fintech OS

Innovation Studio 21.1.6.3

User Guide

Copyright © Fintech 2021. All rights reserved.

TOC

Overview 24

Innovation Studio Personas 25

Search Menu 27

To Hide/ Close the pop-up 29

Evolutive Data Core 30

Data Models 30

Entities 30

Attributes 30

Relationships 31

Extensibility 31

Building with Data at the Core 32

Data Integration 32

Data Model Designer 33

Introduction 33

Accessing Data Model Designer 33

The User Interface 34

Entities Panel 34

Toolbar 34

Work Area Panel 35

Creating Data Models 35

Introduction 35

How to Create a Data Model 36

STEP 1. Add the Data Model 36

STEP 2. Add Entities to the Data Model 37

STEP 3. Define Relationships 38

Working with Data Models 39
Loading Data Models 39
Editing Data Model Details 39
Editing Data Models 40
Deleting Data Models 40
Data Model Explorer 41
Business Entities 41
Introduction 41
Types of entities 42
Viewing existing business entities 43
Creating Entities 44
Editing Entities 52
Deleting Entities 53
Referential Integrity Check 53
Attributes 54
System-generated attributes 55
Types of Attributes 56
Adding Attributes 67
Adding Option Set Attributes 71
Introduction 71
STEP 1. Define the option set (picklist) 72
STEP 2. Add new option set items 72
Changing the items order 75
Editing option set items 75
Deleting Option Set Items 75
STEP 3. Define the option set attribute 76
Example: Create an option set 77
Reordering Entity Attributes 77
Entity Unique Constraints 78
Create a Unique Constraint for an Entity 78

Add Unique Constraint Attributes 79
Enable a Unique Constraint 80
Disable a Unique Constraint 81
Entity Relationships 81
1:N Entity Relationships 82
Introduction 82
Creating a one-to-many relationship 83
N:N Entity Relationships 85
Introduction 85
Creating a many-to-many relationship 86
Advanced Entity Find 89
Step 1. Select the attributes you wish to display 90
STEP 2. Apply Filtering Conditions 91
STEP 3. Validate the Fetch Results 95
Data Views 96
View basic action handlers 96
Creating and Designing Views 97
Introduction 97
Adding Views 97
Provide View General Information 98
Defining View Data 98
STEP 1. Fetch entity data 98
Choose Entity Links 99
Apply Filtering Conditions 103
Validate the Fetch Results 109
STEP 2. Define the View Columns 114
STEP 3. Set the default sorting of the view records 116
Generate View Columns 116
The view has no fetch data 117
The view has fetch data 117

Inline Editing for View Records 117
Create Views using Cell Template 120
Creating views default fetch 120
Creating views with custom fetch 121
Customizing Delete Confirmation 123
Show Loading Panel 125
Data Forms 126
Creating an Entity Form 127
Editing an Entity Form 128
General Settings 128
UI 129
Steps 129
Field Options 129
Filtered Fields 130
Advanced 130
Security Roles 130
Transient Data Entities 130
Create transient data entities 130
Define the automation script for load 131
Examples 132
Define the automation script for save 135
Extend platform data entities with transient data entities 136
Step 1. Create a transient data entity extension 136
Step 2. Add virtual attributes (only for transient data entities with single instance outputs) 137
Step 3. Bind entity attributes to the automation script for load input parameters 138
Step 4. Bind entity attributes to the automation script for save input parameters 139
Display transient data entity attributes in form driven flows 140
Display transient data entity attributes for single instance outputs 141

Display transient data entity attributes for collection outputs 142
Sample API Calls 144
Data Import Template 147
Data Governance 151
Sensitive Context 152
Create a Sensitive Context 153
Sensitive Data 153
1 Configure the Sensitive Entity 154
2 Define the Related Sensitive Entities (optional) 156
3 Add Validation Rules 158
Data Anonymization Jobs 160
1 Set up the Anonymization Service 160
2 Set up the Anonymization Service Trigger 161
3 Restart the FintechOS Job Server 161
Data Anonymization Requests 162
1 Create a Sensitive Request 162
2 Check the Returned Sensitive Data 164
3 Approve the Records for Data Anonymization 166
External APIs 169
How to create an External API 170
How to configure External API calls 171
External API Call – Settings 171
External API Call – Parameters 178
External API Call – Custom JavaScript Reference 179
How to call a External API 180
External API General Settings 181
Asynchronous Run 181
Default Timeout 182
Data Model 182

Data Pipes 182
Create the Destination Data Model 185
Step 1. Create the destination entity which will store the replicated data 185
Step 2. Add attributes to the destination entity 186
Replicating the source primary keys 186
Correlated lookup attributes 186
Optionset attributes 189
Set Up the Data Pipes Connections 189
Step 1: Set up the source connection 190
Step 2: Set up the destination connection 192
Set Up the Data Pipes Replication Jobs 196
Run Replication Jobs 199
Digital Journeys 200
Form Driven Flows 203
Creating Form Driven Flows 204
Prerequisite 204
1 Create the form driven flow 204
2 Set the journey default type 212
3 Design the flow UI 212
4 Group information in steps 214
5 Define who has access to the flow 214
6 Save the flow 214
Cloning a form driven flow 215
Adding and Configuring Steps 216
STEP 1. Add step 216
STEP 2. Design the step layout 217
STEP 3. Flow Control 219
STEP 4. Provide the code to be executed after the step is generated (optional) 219

STEP 5. Define who has access to the step 220
STEP 6. Actions 220
After Load 220
After Save 222
STEP 7. Save the step 223
Custom Processor Step 223
Action Step 226
Flow Control 227
Control Digital Journey Flow 227
Overriding the default next step set trough OrderIndex 228
Control Form Driven Flow based on rules 228
Checking with Custom Processor 233
Configuring Field Options 234
How to Customize Form Fields 235
How to Configure Field Rules 239
1 Add field for action 239
2 Add field for condition 239
Defining Form Actions 240
How to create a form action 241
Available form action commands 242
How to add an action to a specific step in a Digital Journey 243
How to attach an endpoint in Form Action 244
Defining Action Groups 245
Prerequisite 245
STEP 1. Add action group 246
STEP 2. Add endpoints 246
How to hide the action button 247
Flow Map 248
Defining Filtered Fields 249
How to add filtered fields 249

Filtered fields on editable views 254
Header Items 255
Linking Labels to Attributes 257
Linking labels to attributes using the HTML editor 257
Linking labels to attributes using the Source code 257
Displaying View from Another Entity 258
Prerequisites 258
How to display a view from another entity 258
Filtering the view results 261
Passing default value 261
Refreshing the view 261
Rendering Custom Data Extensions 262
Prerequisites 262
How to Render Custom Data Extensions 262
Example 263
Creating Custom Search Forms 264
Form Driven Mock-up Flows 267
How to create a form driven mock-up flow 267
How to display a form driven mock-up flow 269
How to convert a form driven mock-up flow into a regular form driven flow 270
Custom Flows 271
Differences between the Form Driven Flow, Custom Flow and Digital Journey. 272
Creating Custom Flows 272
STEP 1. Provide custom flow general information 273
STEP 2. Design the custom flow layout 273
STEP 3. Define the custom flow 274
STEP 4. Define who has access to the journey 275
STEP 5. Save the journey 276

STEP 6. Display the Custom Flow on the Portal 276
Creating Custom Controls 278
Creating custom controls using DevExtreme widgets 278
Modify Control Advanced Properties 279
Creating custom controls using JQuery 280
Code Execution Sequence 281
Entity Forms 281
Entity Form Steps 282
Form Driven Flows 283
Form Driven Flow Steps 284
Form Driven Flows (Wizard Mode) 285
Form Driven Flow Steps (Wizard Mode) 286
Custom Flows 288
Entity Views 289
Charts 291
Digital Journey Map 293
Adding a flow to the map 293
Editing a step 294
Digital Journey Context 295
Apply flow control rules only for specific digital journeys 296
UI Designer 297
STEP 1. Define the form layout 298
Insert Row Templates 298
Move Row Templates 300
Delete Row Templates 301
Procedure protocol 301
STEP 2. Add attributes 303
STEP 3. Configure and add relations 304
STEP 4. Working with Buttons 307

Chart 307
Form Actions Buttons 310
Call Custom processor button 312
Endpoint Buttons 313
Custom Buttons 313
STEP 5. Access predefined HTML Templates 318
STEP 6. Add entity extension child collection support 320
Using Your Own Style Sheets 321
Create a New Style Sheet 321
Apply Style Sheets (No-Code) 322
Apply Style Sheets Using Code 323
Limit Style Impact to Current Form 324
Overwriting Variables 324
Localization 325
Viewing Defined Languages 325
Adding Languages 326
Localizing Generic Resources 329
View generic language resources 329
Localize generic language resource 330
Import Localized Values 330
STEP 1. Export generic resources and localize them 330
STEP 2. Import localized values 330
Localizing Metadata 330
Localizing HTML Templates 332
Localize HTML elements on data forms 332
Localize from Metadata 334
Localize Relationship Labels 335
Localizing Option Set Items 336
Localizing Views 336
Client-side Localization 337

Server-side localization 340
Code Snippets Support 341
Code Snippets Support for the HTML Editor 342
Code Snippets Placeholders 342
Navigating placeholders 342
Replacing placeholders 343
Deactivating placeholders 343
Series of placeholders 343
Nested code snippets 344
Examples of code snippets 344
Examples of code snippets for attributes and relations. 344
Code Snippets Support for JavaScript Text Boxes 345
Code Snippets 345
Code Snippets Placeholders 346
Navigating placeholders 346
Replacing placeholders 346
Deactivating placeholders 347
Series of placeholders 347
Nested code snippets 347
Code snippets for entities and attributes 347
Fintech Automation 349
Business Formulas 349
Business Decisions Processor 349
Digital Product Automation 350
Computer Vision 350
Digital Insurance Product Automation 350
Digital Documents Processor 351
eSign 351
Face Recognition 351

Video Streaming 352
Hyper-Personalization Automation 352
Business Workflows Processor 352
Omnichannel Campaigns 353
Omnichannel Communication Automation 353
Processor settings and mapping 353
Business Formulas 379
Define Formula Inputs 381
Add arguments to a formula input 382
Define Formula Expressions 385
Add steps to a formula 386
Test Your Formula 390
Formula Versioning 391
Activate a Formula 392
Clone a Formula 392
Create a New Formula Version Draft 393
Activate a Formula Version Draft 394
Formula Editor 396
Syntax 397
Formula Arguments 399
Built-in Functions 399
Examples 407
For Simple types 407
For Collection types 416
For Simple Collections types 419
Data Set Calls 424
Data Sets 425
Create a Data Set 425
Define Data Set Discriminants (non Single Value data sets) 427

Add Data Set Values (single value data sets) 428
Add Data Set Values (non Single Value data sets) 428
Data Set Versioning 430
Activate a Data Set 430
Create a New Data Set Version Draft 430
Activate a Data Set Version Draft 431
Formula Parameter Mapping 431
Calling the Business Formulas 433
Calling Formulas in a form-driven flow (no code) 433
Track form driven flows that are using a specific formula 435
Call formulas using server side scripts 435
Export a Formula or Data Set 436
Analytics 438
Advanced Analytics 439
Register App for Power BI 440
Prerequisites 440
How to find the tenant URL 440
Register app for Power BI 441
Embed Power BI Report 445
STEP 1. Get the Power BI report ID 445
STEP 2. Embed the Power BI report in Innovation Studio 445
STEP 3. Add report parameters (Embedding authentication mode only) 447
Add Power BI Report to Dashboard 448
Prerequisites 448
How to add a Power BI report to a dashboard 448
How to add Power BI Reports to Digital Journeys 450
Supported attributes 451
Report parameters in HTML Markup 451
Setting report parameters at runtime 452
PowerBI client-side JavaScript API 453

Custom Reports 454
Creating a custom report 454
STEP 1. Add a report 454
STEP 2. Add report items 456
STEP 3. Insert Report Parameters 457
STEP 4. Define who has access to the custom report 457
STEP 5. Save the report 457
Tabular Reports 457
STEP 1. Add Data Source and Parameters 458
Using Stored Procedures 458
Using Fetch Data 459
STEP 2. Add Report Parameters 460
STEP 3. Add Simple Grid Report 461
STEP 4. Add Report Items 462
STEP 5. Define Report Access Privileges 463
Charts 463
Creating charts 464
Digital Developer Tools 471
DB Tasks 472
Step 1. Add DB tasks 473
Step 2. Add security roles to DB tasks 473
Step 3. Execute DB Tasks 474
Syntax: 474
Request Parameters 474
Returns 475
Advanced Code Editor 475
Features 476
How to Access the Advanced Code Editor 476
General Layout 477

Files Explorer 477
Toolbar 477
Search Nodes 478
Search in Files 478
Properties List 478
Previewing HTML Files 478
How to use the Advanced Code Editor 479
Debugging files from the editor 480
Automation Scripts 480
Event Triggered Automation Scripts 481
On-demand automation scripts 483
Setting the execution order of automation scripts 483
Using Automation Script Libraries 484
Particular automation script libraries 485
Creating Automation Script Libraries 486
Using Web API Client Libraries 487
How to create a Web API client library from an OpenAPI or WSDL specification file 488
How to use a Web API client library in server automation scripts 490
Add certificate support to WebApi client and WCF client 490
Creating Event Triggered Automation Scripts 492
Creating On-demand Server Automation Scripts 496
Customizing Input Parameters 498
Customizing the Output Structure 500
Examples 502
Creating Endpoints 504
1 Create an endpoint 504
2 Attach a security role to an endpoint 505
Calling Actions 506
Scheduling Server Automation Scripts 508

STEP 1. Add schedule job 508
STEP 2. Add schedule services 509
STEP 3. Set the execution order 510
Using Plugin Assemblies 511
STEP 1. Add Plugin Assembly 511
STEP 2. Add the IEbsPlugin Plugin 511
STEP 3. Add UI Processor 512
XML Support 512
Load XML from String 512
Catch XML Load Error 513
Run XPath Queries 513
Run XPath Queries with Namespaces 514
Node API Calls 515
Debugging Automation Scripts 516
Debugging Log 516
Throw Exceptions 517
JavaScript Exceptions 517
Console Debugging 518
Code Blocks 519
1 Add categories 519
2 Add code blocks 520
3 Use code blocks 522
Custom Client-side Functions 523
Defining Custom Functions (using Client Script Libraries) 526
Fluent Queries 527
How to Execute a Fluent Query 528
Comparison Operators 529
Logical Operators 530
Inner Joins 530

Left Joins 530
Attribute Aliases (Projections) 530
Where Clauses 532
Aggregate Functions 533
Working with Fluent Query Result Sets 535
Map result sets to POCO objects 536
Sequencers 537
How to add items to the sequencer 538
How to call the Sequencer 539
Entity versioning 539
How to Set Up Versioning on an Entity 541
1 Configure the versioned entity's attributes 541
2 Configure the versioned entity's data events 541
3 Set Up the Versioning Button 542
4 Set Up the Entity Workflow 542
5 Configure the Version Settings 543
(Optional) Set Up Related Versioned Entities 544
6 Configure the Entity Status Settings 546
How to Version an Entity Record 548
Email Templates 549
Digital Frontends 551
Digital Experience Portals 552
Customizing the Login and Home Page 554
Using Custom Theme 556
STEP 1. Create custom theme 556
STEP 2. Set default custom theme 559
STEP 3. Apply custom theme to the Portal UI 560
Using Custom Icons 560
What files do you need? 560

Files Location 563
Use custom icons 563
Setting Sticky Header 564
Grouping Entities in Menu Items 564
Show Tooltips (for users) 566
Creating HTML Widgets 567
Creating Dashboards 568
Step 1. Add dashboard 568
Step 2. Attach security role to a dashboard 570
Step 3. Add dashboard on a portal profile 570
Adding Widgets to Dashboards 571
Prerequisite 571
Add widgets to dashboards 571
Customize Widgets 572
Resize Widgets 572
Customize Widgets Layout 572
Adding Journeys to Dashboards 573
Adding Charts to Dashboards 576
Prerequisite 576
Add widgets to dashboards 576
Customize Widgets 577
Resize Widgets 577
Customize Widgets Layout 577
Editing Dashboards 578
Using Portal Profiles 578
Step 1. Insert key 580
Step 2. Create portal profiles 581
System Parameters on Portal Profiles 582
Attach Menu Items on Portal Profile 583
Add Dashboards on Portal Profiles 584

Access to Portal Profiles based on Security Roles 584
Setting security roles 584
Restrictions 585
Custom Page Templates 586
Tokens 588
Default Behaviors for Custom Login Pages 591
Examples 592
Configuring the Digital Experience Portal 594
Use Stripped Theme 594
Load Custom Style Sheets 595
Configure the footer text per language 596
Move language selection to the user profile panel 597
Hide Select Theme and Select Palette settings 598
Hide Company Logo 599
Hide the Main Menu 599
Hide search in the Main Menu 600
Hide APPS dashboard 601
Hide My Profile Link from the User Profile panel 602
Keyboard Shortcuts 603
General keyboard shortcuts 603
Shortcuts for date/ date time fields 604
Shortcuts for drop-down fields 604
Shortcuts for radio buttons 605
Shortcuts for check boxes 605
Anonymous Frontends 605
Is it secure to expose digital journeys to unauthenticated users? 605
Setting B2C Environment 606
How to set up the B2C environment 608
STEP 1. Install the FTOS Reverse Proxy 608

STEP 2. Configure the FTOS Reverse Proxy 608
STEP 3. Enable journey to be accessible through the reverse proxy 610
STEP 4. Override default Save on the journey with an endpoint 612
STEP 5. Create and use your own styles sheets (optional) 612
STEP 6. Set anonymous frontends to serve in a specific language (optional) 612
Step 7. Reset an anonymous frontend session 612
Overriding Default Save on Journeys 613
STEP 1. Create an on-demand automation script 613
STEP 2. Create an endpoint and attach the script to it 614
STEP 3. Call the endpoint on the form driven flow 615
Serving User Journeys in a Specific Language 615
Prerequisite 615
Set a journey to serve in a specific language 616
Manage Style Sheets for B2C User Journeys 616
Security 618
Business Units 619
Creating Business Units 619
Managing Business Units 620
Editing Business Units 620
Removing Business Units 621
Security Roles 621
Default Security Roles 621
Creating Security Roles 622
1 Add the security role 623
2 Assign security items to the security role 623
Editing Security Roles 625
Users 626
Adding Users 627

Editing Users 628
Unlock user account 629
Recovering Password (for users) 629
DevOps 631
Exporting a Deployment Package 632
1 Create deployment package 632
2 Add Components to the deployment package 633
3 Export deployment package 641
Importing a Deployment Package 642
Prerequisites: 642
Creating Enhanced Deployment Packages 644
Deployment Package File 644
Deployment Package Folder 644
DataConfigImport Subfolder 645
DataConfigImport.xml File 645
ReportTemplates Subfolder 646
Viewing Deployment Package Logs 647
Sorting Criteria used for XML Sibling Elements 648
Configuration Data Definitions 651
Configuration Data Deployment Package 657
Import data using DevOps 659
1 Deployment package 659
2 Configuration Data Deployment package 660
Fintech Mobile Launcher - Capacitor 663
Setup 663
Configuration 664
Google Services 665
iOS Build 665

Overview

Innovation Studio is a toolkit which allows non-technical users and fintech developers to build software from scratch or extend existing functionality on top of the FintechOS Platform and out-of-the-box accelerators available in the Fintech AppStore.

Using the Innovation Studio, you gain full flexibility and fast time-to-market and can build personalized products, digital journeys and communication through using our integrated hyper-personalization engine.

Complementary to the classical development tools, it also provides a dedicated user interface which is comprised of various editors and configuration engines which can be used for simple and repetitive tasks.

The utility tools and editors carry distinct development logic:

- **Data Model Designer** - is a graphical tool that simplifies data modelling tasks and enhances productivity. Using the Innovation Studio Data Model Designer, consultants and engineers can create, browse and edit logical, relational and physical data models.
- **Data Model Explorer** - enables the configuration of the data model by defining entities, attributes and relationships. It also enables you to extend the data model with data from third-party systems or other instances of FintechOS.
- **Digital Journey Configurator** - features an HTML editor allowing the configuration of forms for different digital journeys, drag & drop list/view editor, and Monaco editor for JavaScript for defining low code login on the client-side.
- **Complex Dev Tools** - managed through automation scripts and libraries – grants developers with editors for writing automation code.
- **Fintech Automation tools** - micro-services with embedded AI which captures data from external data sources and provides you with AI driven insights and actions based on machine learning, big data aggregation and cognitive reasoning.

- **Dashboard** – provides a simple editor for adding widgets such as PowerBI reports, HTML widgets, entity views within dashboards.
- **Security** – allows configuring the organization structure, security roles and managing both internal and external users.
- **DevOps** – set of configurations that create packages with entities, forms or even flows to be deployed in another enviornment or another database.
- **Data Pipes** – enables consultants & developers to easily integrate with third-party systems and other instances of FintechOS.

Using authoring types, Innovation Studio offers a personalized experience, helping you focus on the work that is important to you, thus improving efficiency.

Innovation Studio Personas

Innovation Studio personas (authoring types) narrow the capabilities and refine the focus to suit specific business needs, offering a personalized experience.

- Consultant. Access no-code capabilities.
- Developer. Access advanced capabilities with built-in functions on the Server and Client SDK.

To log in Innovation Studio, you have to choose the persona, provide your Innovation Studio credentials and click Log in. Based on the selected persona, you will have access to specific features. The theme and menu differs based on persona, showing fewer options to consultants.

The table below describes the features available per persona.

Feature	Consultant	Engineer
Evolutive Data Core	yes	yes
Fintech Automation	yes*	yes
Digital Journeys	yes	yes
Custom Flows	no	yes

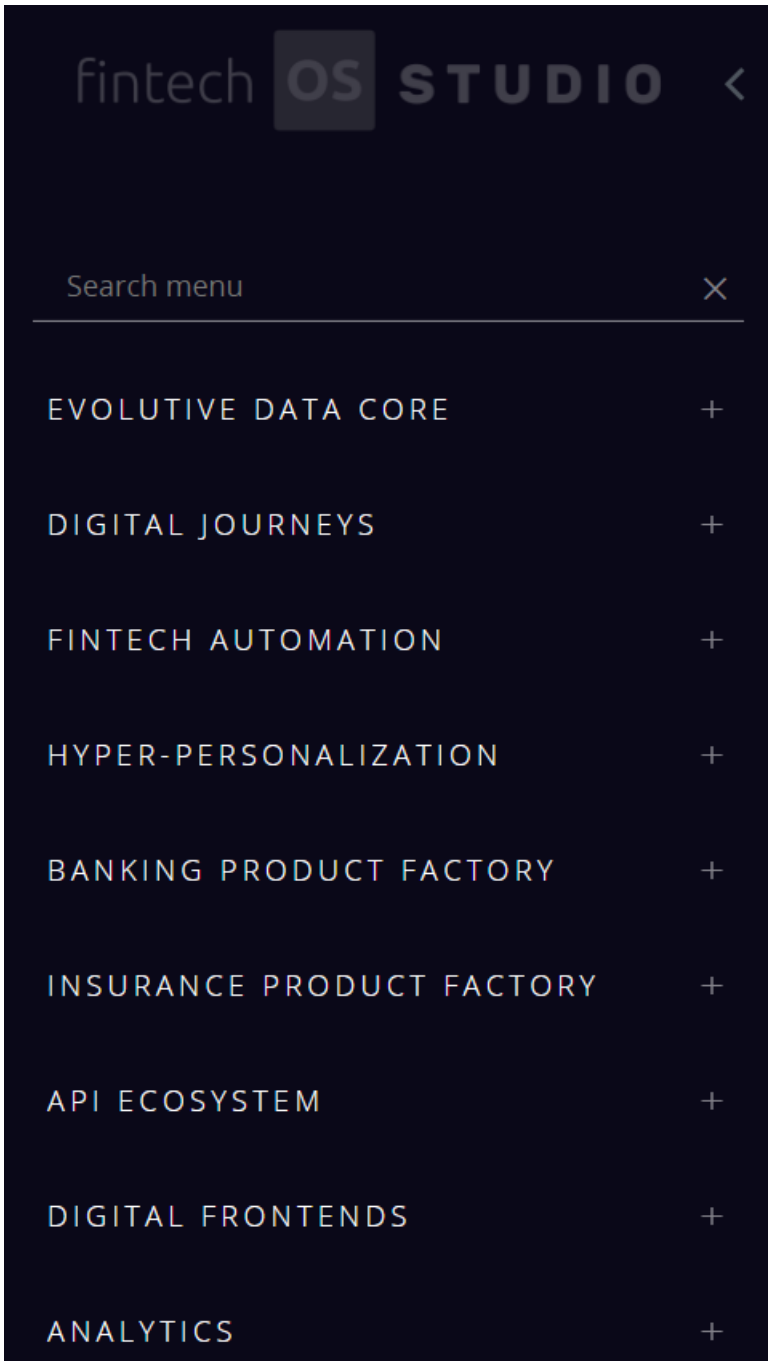
Feature	Consultant	Engineer
Anonymous Frontends Configuration	no	yes
Digital Experience Frontends	yes	yes
Advanced Coding	no	yes
Security	yes	yes
Admin	yes**	yes

*limited to: Digital Documents, Business Workflows, Business Decision, Hyper-personalization, Omnichannel Campaigns, Business Business Formulas.

**limited to: Application Languages, Localization Resources, Option Sets, Settings, User competence settings, Omnichannel Communication Automation, Entity Versioning.

Search Menu

The search button is used in the menu of a Studio environment by our users to look for a particular menu item. The search menu is found both on the Studio and Portal.



By clicking on the button, it is possible to insert data regarding the need the user has. The input is displayed.

When searching a pop-up panel will be displayed below the search input containing the results of the search.

The user can navigate the search results by pressing the up or down keyboard arrows.

By pressing or clicking, the selected search result is displayed.

To Hide/ Close the pop-up

Press or click on the X in the search input, the search string will be cleared and the popup panel will be hidden.

Evolutive Data Core

In modern software development, data represents the vital core of financial and banking solutions architecture.

All business logic and presentation components of a software solution require access to a persistent data storage backbone in order to function. The Evolutive Data Core provides a data persistence layer where the information processed by Fintech digital solutions is stored and managed.

Data Models

Defining the data model is typically the first step of developing FintechOS digital solutions, as the business logic and the presentation layer of a digital solution need data in order to function. Data models identify what data is needed and how it should be organized. They also ensure the quality of data via naming conventions and default values. Innovation Studio data modeling is based on the Entity - Relationship paradigm, which uses the concepts of **entity**, **attribute**, and **relationship** to represent data.

Entities

Entities are representations of the main physical or logical components that require record keeping (data storage). Typical examples include customers, invoices, employees, or products. For more information, see Business Entities.

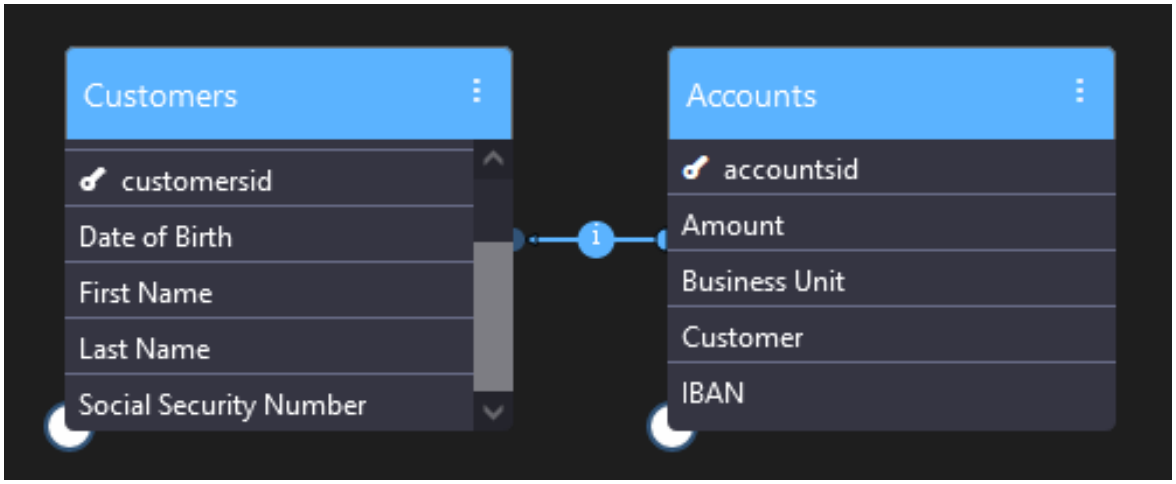
Attributes

Attributes are used to model the specific properties of an entity that you want to record. For instance, for an entity representing customers, you might want to define attributes such as first name, last name, social security number, or date of birth. For

more information, see Attributes.

Relationships

Relationships define links between entities. For instance, in an online banking digital solution, you may need to record data about your customers and their banking accounts. For this purpose, you will create two entities (one for customers and one for accounts) and a relationship between them representing which customer owns a specific account.



In Innovation Studio, relationship modeling is implemented by a special type of attribute called **lookup**. In the example above, the Customer attribute in the Accounts entity is a lookup attribute which helps to define the relationship between the Accounts and the Customers entities. For more information, see Lookup.

Extensibility

Innovation Studio allows you to modify your data model as your digital solutions evolve. You can change the underlying data model of a digital solution in order to customize or integrate it, even while that digital solution is live. Innovation Studio will perform checks to ensure that you don't corrupt data or break any dependencies in

the process. For instance, you will not be able to delete an entity that is referenced in a relationship or to add a required attribute to an entity that is populated with records (and therefore has no matching entries for that attribute in the legacy data).

Building with Data at the Core

Form Driven Flows, which are the main building blocks of FintechOS digital solutions, are each built on top of a dedicated data model. This allows a lot of the default business logic and UI rendering to be inferred from the data model itself and generated automatically, greatly reducing development time and improving the quality and reliability of your solution.

Based on the flow's data model, the form fields used to display entity attributes, the session storage, the user authentication and authorization, the logic for loading and saving records, etc. are generated automatically. You can, of course, further customize them, but most of the heavy lifting is done by the platform.

NOTE
You also have the option to use form driven mock-up flows which allow you to design a form driven flow without an underlying data model. This lets consultants and developers to quickly define the general layout of the user interface. Developers can then attach a data model to the mock-up, map entity attributes to the corresponding form fields, and work on any additional back-end configurations. For more information, see Form Driven Mock-up Flows.

Data Integration

The Evolutive Data Core not only ensures the modeling of the data structure, but can also interact with data from legacy systems, processes, and data repositories, greatly extending the FintechOS data persistence capabilities.

Innovation Studio provides various options to import, interconnect , and replicate data, such as: Data Import Templates, REST APIs, External APIs, and Data Pipes.

Data Model Designer

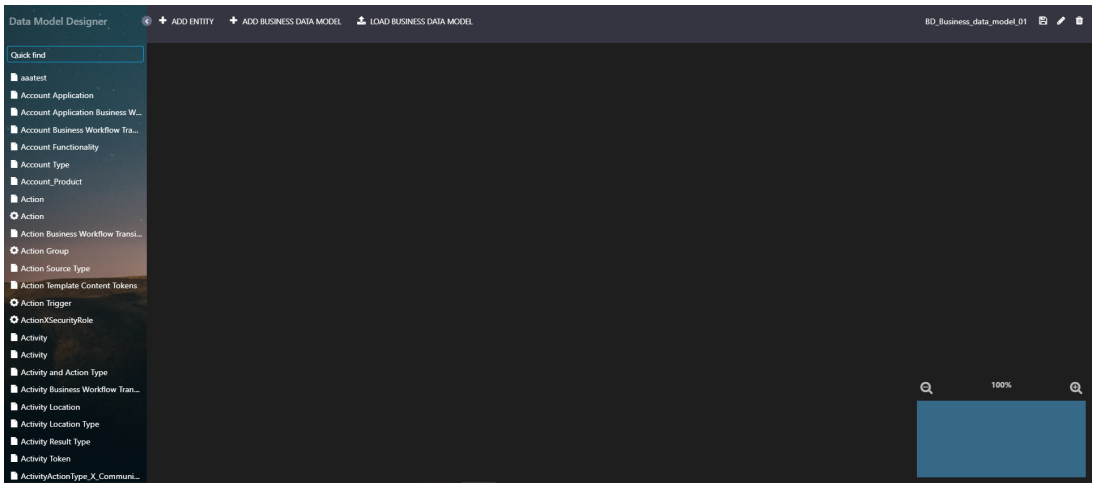
Introduction

Data Model Designer is a graphical tool that simplifies data modelling and increases user productivity when performing data modelling. Consultants and engineers use the Innovation Studio Data Model Designer to create, view, and edit data models type-physical.

Accessing Data Model Designer

To access the Data Model Designer:

1. At the top-right corner of the page, click the menu icon. The navigation menu expands.
2. From the menu, click Evolutive Data Core > Data Model Designer. The **Data Model Designer** appears.



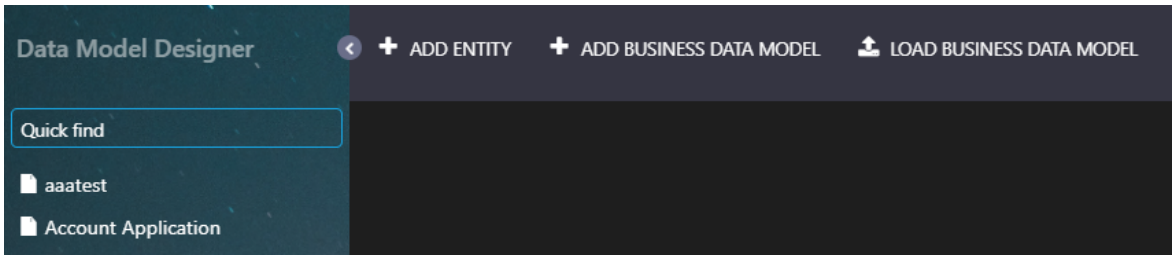
By default, the Data Model Designer displays the most recently used data model. When first opened during a session, it will display an empty canvas.

The User Interface

Innovation Studio's Data Model Designer provides a very user-friendly interface offering the means to add and easily update and extend data models.

Entities Panel

The entities panel lists the system entities and custom entities. You can hide/show this panel by clicking the Minimize / Expand arrow:



HINT

- Use the Quick Find box at the top of the entities panel to filter entities.
- Hovering an entity name in entity panel, displays both entity name and entity display name . This is useful to identify entities with identical display names but different entity names.

Toolbar

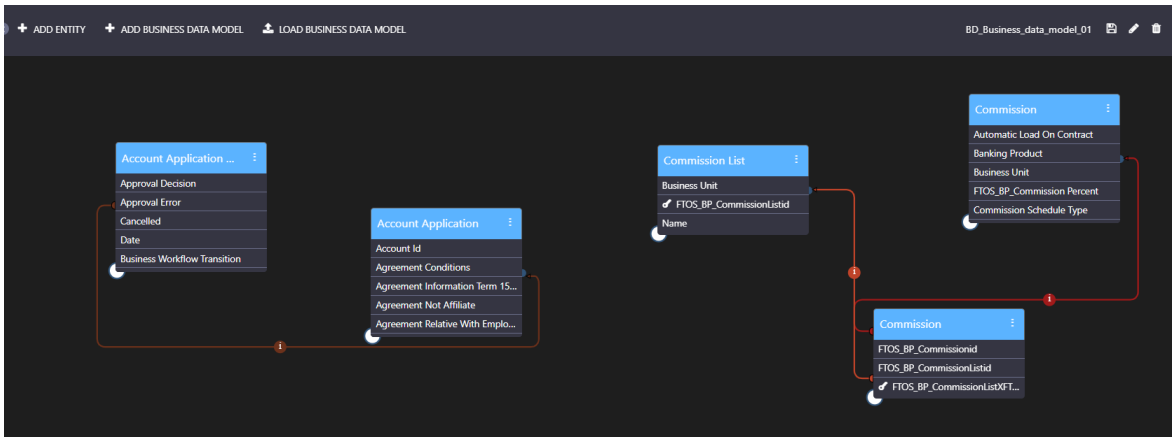
The Toolbar provides you with the buttons needed for performing common functions:

Button	Description
	Opens the Add Business Entity page which allows you to add a new entity.
Add entity Add Business Data Model	Opens the Add Business Data Model dialog which allows you to add a new data model.
Load Business Data Model	Opens the Load Data Model dialog which allows you to load a data model previously created in Innovation Studio.

When a data model is open, the right-side of the toolbar contains additional icons which give you the means to save (), edit ( ), or delete ( ) the current data model.

Work Area Panel

This is where you can design your data model in a visual interface.

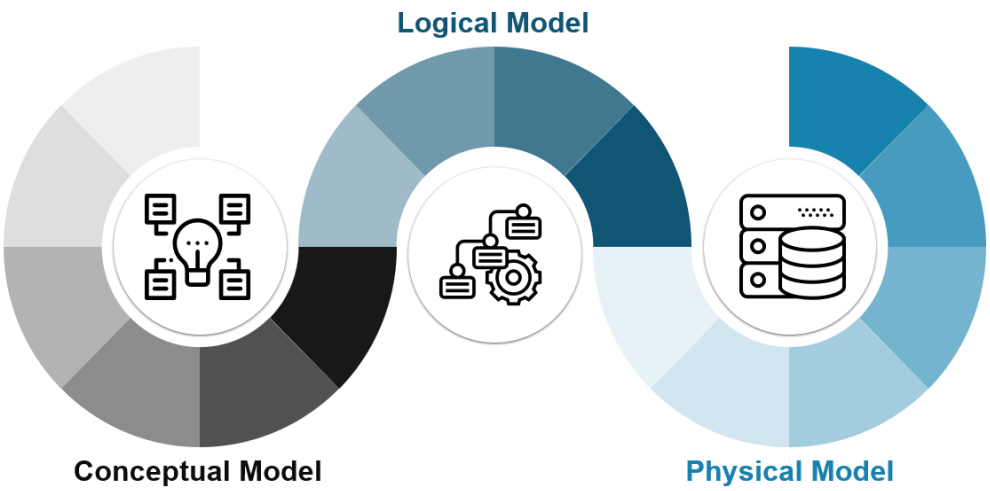


- Drag and drop entities from the entities panel to add them to the data model.
- Drag the bottom left corner of an entity to a related entity to define a relationship.
- The overview panel at the bottom right corner of the work area allows you to zoom in and out and displays the current view's outline within the canvas.

Creating Data Models

Introduction

To create a well-defined data structure in a database, you need to identify the business concepts (entities, attributes , relationships), define data characteristics and also define how the data is related.



A Conceptual Data Model is fundamental to understanding how data is organized and how data elements are related to one another. It identifies basic data concepts, commonly referred to as entities, and the relationships between entities. Entities are comprised of attributes to define the characteristics of entity records – think of them as atomic characteristics such as age, date of birth, height, weight, or eye color.

Defining the attributes is part of the logical model while creating the relationships between entities is the physical model.

This section describes how to create a physical data model using Innovation Studio Data Model Designer.

How to Create a Data Model

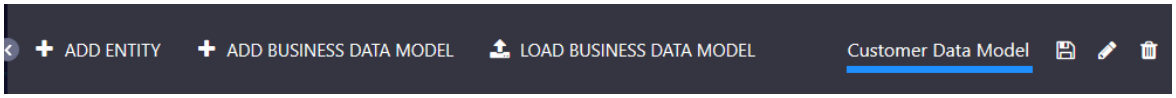
The first step in creating a data model after you open the Data Model Designer is to add the data model.

STEP 1. Add the Data Model

1. Open the Data Model Designer.
2. On the toolbar, click the Add Business Data Model button. The Add Business Data Model page opens.

3. In the Name field, type a name for the data model you're creating.
4. (Optional) In the Description field type a brief description of the data model.
5. Click OK or press Enter on the keyboard to add the data model.

The page closes and the Data Model Designer opens the data model you've previously added, Its name is displayed on the toolbar.



NOTE
The next time you will open the Data Model Designer, it opens on the data model you last worked on.

Now that you've added the data model, you can start adding entities.

STEP 2. Add Entities to the Data Model

The next step in creating a data model is to identify all of the entities you will need. This could be a customer, an invoice, etc.

Once identified, make sure that you add the entities in Innovation Studio. You can add entities in two ways:

- From the Data Model Designer - On the toolbar, click the Add Business Entity button to open the Add Business Entity page, provide the mandatory entity information and save it. The entity has by default a set of system attributes. You can extend the data model by adding new attributes as needed.
- From the Data Model Explorer - Save your current data model by clicking the Save icon, then click the Add Entity button in the toolbar. Fill out the required properties in the Add Business Entity page, and the click Save and Close. In Data Model Designer, locate the newly created entity in the Entities panel, and then drag-n-drop it on the Work Area.

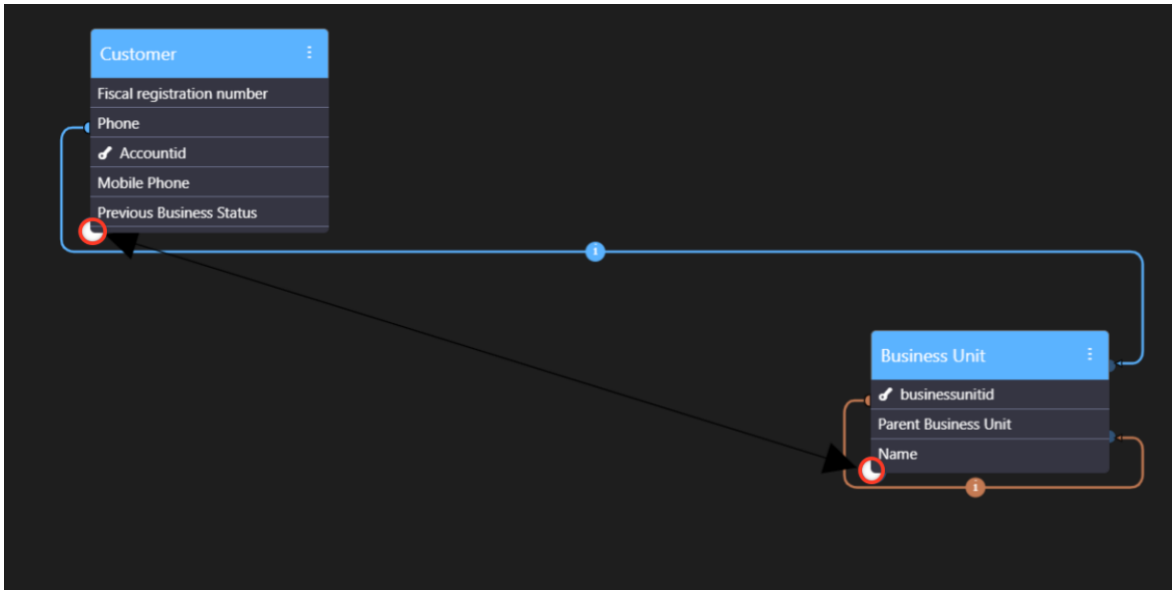
To easily spot the entity you want to add to your data model, you can use the search feature available at the top of the entities panel. If you need new entities, add them through the Data

Model Designer or the Data Model Explorer, go back to the Data Model Designer, drag and drop an entity from the panel into the data model, and that's it!

After you add entities to your data model, the next step is to define the relations between the entities.

STEP 3. Define Relationships

Identify relationships between the entities of your data model. Look at two entities, are they related? How are they related? Describe the relationship. To define relationship between two entities of your data model, in Data Model Designer, select the circle displayed in the left bottom corner of one of the entities and drag it towards the circle of the other entity:



The Add Relationship page appears which allows you to define the relationship. For more information on relationships, their types and how to define them, see Editing Entities.

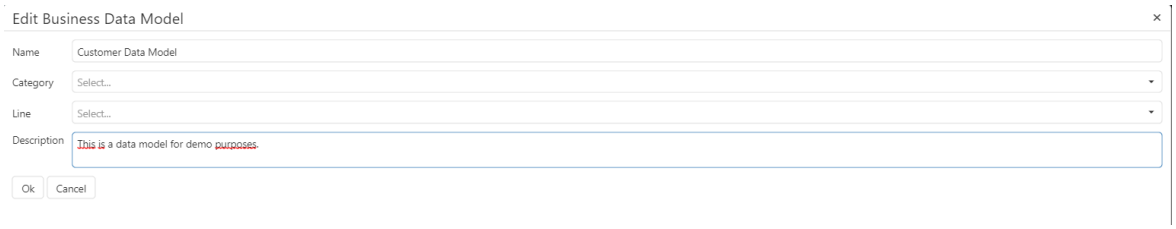
After you save the relationship between two entities, it is displayed as object in Data Model Designer and connectors are shown between it and the entities which it links (that is, the entities between which you created the relationship).

Working with Data Models

Loading Data Models

When you open the Data Model Designer, it opens up the data model you last worked on.

If you want to work on a different data model that you created in Innovation Studio, from the toolbar, click the Load Business Data Model button. The Load Business Data Model page appears which lists all data models that you created in Innovation Studio:



In the list of existing data models, click on the data model you want to load and it opens in the Data Model Designer.

Editing Data Model Details

You can edit the details of a data model (name and description) at any time. To do so, follow these steps:

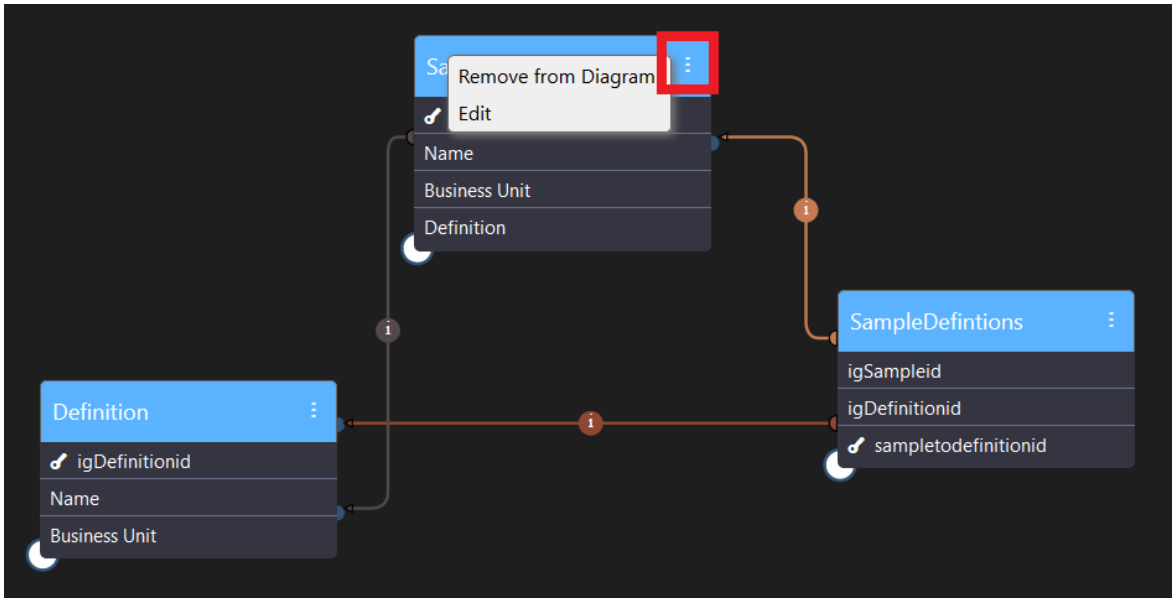
1. Load the data model whose details you want to modify (if it's not the current one).
2. From the right-hand side of the toolbar, click the Edit icon ( ). The Edit Business Data Model page appears.
3. Update the details as preferred (Name and/or Description).
4. Click OK.

> ⚠ **IMPORTANT!**
> Make sure that you save the data model every time you edit it; otherwise, you loose the data when adding an entity or navigating to a different page.

Editing Data Models

In Innovation Studio, you can edit a data model at any time by first loading it in the Data Model Designer, then add, edit or delete entities, attributes and relationships between entities to best suit your business model.

Use the ellipse button at the top right corner of an entity to edit or delete it.



Removing an entity from the diagram also erases the entity's relationship lines, but the lookup attributes used to define those relationships are still preserved in the entity.

Hovering on the letter i displayed on the entity lines displays a short description of the relationship, for example "SampleDefinitions linked to Definition througn <idDefinitionId>".

Relationship lines have different color to enable easier tracking in complex data models. There is no color coding employed.

Deleting Data Models

In the event that you want to remove the current data model, on the toolbar, click the Delete ( ) and in the confirmation pop-up that appears, click Yes.

NOTE
The data model deletion CANNOT be undone, so we recommend you think twice before removing it.

Data Model Explorer

The Data Model Explorer allows you to create business entities, relationships and attributes in an entity-relationship framework, handling in the same time data persistence and automatic database provisioning. It also facilitates data exchange through integration or bulk import of data.

This section covers the following topics:

Business Entities 41
Attributes 54
Entity Unique Constraints 78
Entity Relationships 81
Advanced Entity Find 89
Data Views 96
Data Forms 126
Transient Data Entities 130
Sample API Calls 144

Business Entities

Introduction

An entity is an object in the system that you want to model and store information about. Entities are recognizable concepts which have relevance to the database. Some specific examples of entities are: Customer, Product, Offer or Contract. An entity is similar to a table in the relational model.

Types of entities

In FintechOS, there are three types of entities:

- **Platform Data** - Native FintechOS data which is created and stored within FintechOS.
- **External Source Data** - External persistable data which is created in external systems and replicated within FintechOS. This is historical data (read-only data) which you can view in detail in analytics. External Data Source entities stores data from external systems replicated in FintechOS though Data Pipes. For more information on how to replicate data from external systems in FintechOS, see Data Pipes.
- **Transient Data** - Entities that temporarily store data that has been loaded from or is going to be saved to an external data source. For more information, see "Transient Data Entities" on page 130.

Platform Data entities encompasses:

- **System entities** - are used by FintechOS to run as an integrated operating system (OS). You can add Attributes on system entities, define specific forms and views, but you cannot delete them.

  System entities are found in the database under the following schemas:

  - ebsMetadata - stores entities' metadata. For example, information about the entities which are level 1S options on the Innovation Studio main menu.
  - ebsLocalization - stores records for entitles Language and Currency Code.
  - ebsAudit - stores logging information The ebsAudit schema is comprised of the EbsLogs.UniversalLog and EbsLogs.ApiLog. For more information, see the FintechOS Platform Administration Guide.

  System entities can be used within business processes, if needed. For example, the 'systemuser' entity, which stores information of users authorized to log in the platform can be equally used in 'task management' flows for assigning users to tasks.

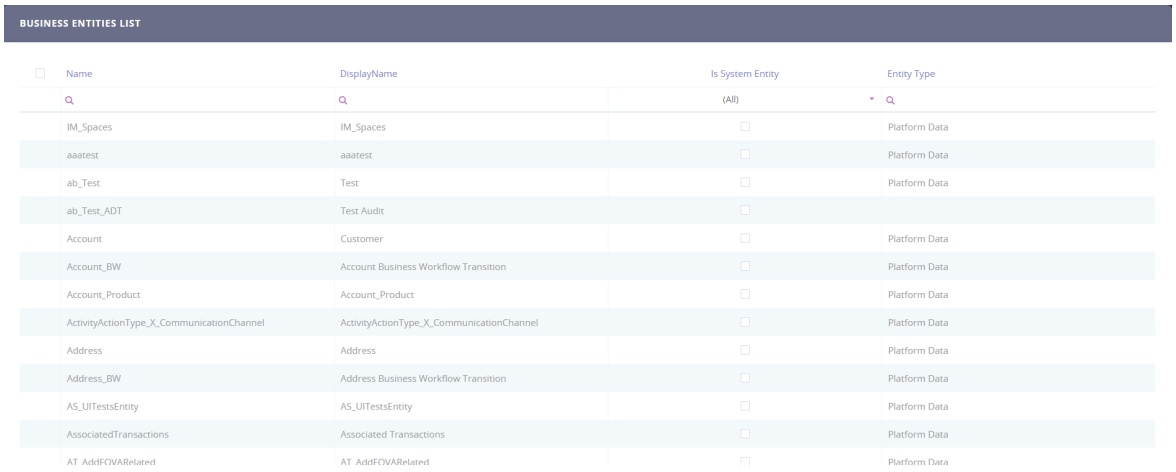
- **Custom entities** - are the entities you define for your application in order to accommodate various business flows. For example: 'Contract', 'Application', 'Legal Agreement'.

Custom entities will be found in the database under the ebs schema or under the schema inserted in the organization table.

Viewing existing business entities

To see the list of entities, follow these steps:

1. At the top-left corner of the page, click the menu icon. The navigation menu expands.
2. From the menu, click Evolutive Data Core > Data Model Explorer. The **Business Entities List** page appears.



The following entity details will be displayed:

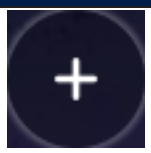
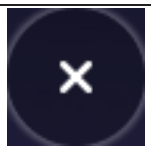
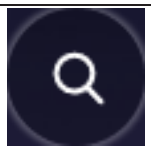
Field	Description
Entity Type	Choose one of the following: • Platform Data • External Source Data • Transient Data.
Name	The entity name as it is stored in the database.
DisplayName	The entity name displayed on views and forms.

Field	Description
Is System Entity	Indicates the entity type. If the value displayed is true, it is a system entity.

Click column headers to order grids ascending or descending.

HINT
You can filter entities by all the search fields displayed in grid, including Name and DisplayName. You can apply multiple filtering criteria at the same time.

At the top right corner of the screen, there is a toolbar with the following buttons:

Button	Description
	Inserts a new business entity. For details, see "Creating Entities" below.
	Deletes the currently selected entities. For details, see "Deleting Entities" on page 53.
	Exports entities. For details, see Data Exports.
	Opens the Advanced Find window, allowing you to define complex filtering criteria based on the entities' attributes. For details, see "Advanced Entity Find" on page 89.

Creating Entities

You can create custom entities to meet your business needs.

To create a new entity, follow these steps:

1. Go to the **Business Entities List** page. For details on how to view the list of entities, see Business Entities.
2. At the top-right corner of the page, click the **Insert** icon. The Add Business Entity page will be displayed.

3. Enter the required fields (attributes).

> **NOTE**
>
> Please note that based on the attribute type you select, you need to fill-in fields mandatory for the selected attribute type. For information on the required fields based on the attribute type, see "Attributes" on page 54.

4. The minimum required fields to create an entity are:

Entity Type

Select one from the list:

- **Platform Data** - Native FintechOS data which is created and stored within FintechOS.
- **External Source Data** - External persistable data which is created in external systems and replicated within FintechOS. This is historical data (read-only data) which you can view in detail in analytics. External Data Source entities stores data from external systems replicated in FintechOS though Data Pipes. For more information on how to replicate data from external systems in FintechOS, see Data Pipes.
- **Transient Data** - Entities that temporarily store data that has been loaded from or is going to be saved to an external data source. For more information, see "Transient Data Entities" on page 130.

Name (only use for add entity)

The unique entity name that will be stored in the database. Provide a relevant name and self-explanatory for what the entity stands for or provide a hint about the business logic it entails.

The entity name is used to identify the entity in the system when working with it (read/ write entity records, define entity relationships, create data models, etc.)

> **NOTE**
>
> You cannot create two entities with the same **Name**.

This field is used by the system and will be displayed only in the application URL. It is not visible to the end-user.

> **NOTE**
>
> A naming convention is an important part in a well-built data model; therefore, we recommend you to use PascalCaseNames (upper camel). The Name starts with an uppercase letter, as do all additional words. Example: StatementPayment.

The field is also used in implementation when calling all the CRUD operations on the specific entity (getByQuery,getById, etc.).

> **NOTE**
>
> On entity creation, the entity's primary key attribute, that is an unique identifier for each entity instance, is automatically generated by the system following this naming convention:'entityname + Id'. The primary key is displayed in the entity's list of attributes and has the attribute type **PK**.

DisplayName

The entity name that will be displayed on views and forms. It is also the label to be localized in different languages.

The Display Name should appear as a noun in singular format (e.g. “Customer”, or “Physical Address”).

DisplayCollectionName

Provide a Collection Name if you want to display the entity on the left-side menu or pin it on the application homepage,

NOTE

The DisplayCollectionName attribute stores all the entity records (instances) within the database; therefore, the naming convention for this attribute is the plural data form of noun used for the entity Name.

TableName (only use for add entity)

The name of the table to be generated in the database, associated with the entity, automatically prefilled by the system based on the entity Name.

NOTE

To avoid affecting data integrity and consistency, do not change the value prefilled by the system.

IMPORTANT!

After you save the entity, you cannot edit this field.

PrimaryAttributeName (only use for add entity)

The name of the main attribute that identifies the entity records from a business perspective. By default, it is a text attribute with a maximum length of 100 characters.

NOTE

A naming convention is an important part in a well-built data model; therefore, we recommend you to provide a name which starts with lowercase letter and all additional words (on the right) start with uppercase.

For example, if the Product entity has three attributes: ID, name and price. For each product, the ID will be unique, so it can be the primary attribute for this entity.

IMPORTANT!

Do not confuse the entity's primary key automatically generated by the system to uniquely identify an entity in the database with the Primary Attribute . For example, you can define the Contract Number as a primary attribute but we do not recommend you to define it as a primary key due to the fact that, for instance, on data import into another system or version of Innovation Studio, the contract number might already exist.

PrimaryAttributeDisplayName (only use for add entity)

The name of the primary attribute, as displayed in the end-user interface on forms and views.

PrimaryAttributeTableColumn (only use for add entity)

The name of the attribute which will be automatically generated in the database. It serves as a primary attribute to identify the PrimaryAttributeName (the main table column stored in the database).

NOTE

A naming convention is an important part in a well-built data model; therefore, we recommend you to use PascalCaseNames (upper camel). Provide a a text which starts with an uppercase letter, as do all additional words.

IMPORTANT!

After you save the entity, you cannot edit this field.

Default Entity Status

Indicates the status corresponding to your work on this entity. To select the entity status, on the right-side of the field, click the drop-down arrow and double-clicking the entity status. The following default entity statuses are available:

- Active – completed work.
- Draft – work in progress

The status selected in this field will appear on entity records when inserting. E.g., if you select by default Active on entity Application, when you insert an Application the field **entityStatusId** will be completed automatically with the value **Active**.

The entity status doesn't impact the entity behavior in any way. It is useful as a way to classify and filter your entities when designing your data models.

At the top right-corner of the page, click the Save and reload icon. The **Add Business Entity** page is replaced by the **Edit Business Entity** page which contains new sections at the bottom which allows you to add entity attributes, forms, views and more.

Other fields that appear when creating an entity for the first time, but they are not required to be filled in when creating the entity.

isAudited

If selected, the checkbox enables you to track all the changes made on the entity record. An audit icon will be displayed on the entity which on click will display the type of changes made on the entity, when the changes have been made and by whom. You will also see a side-by-side comparison between old and new values. For more information on entities audit, see the FintechOS Platform Administration Guide.

Business Workflow

The business workflow that was attached to the entity will be displayed. At first, it is a blank field, after a workflow is associated, the name will be displayed here as read-only. Business workflows allow you to define states and state transitions for your entity records. For more information, see the Business Workflows Processor documentation.

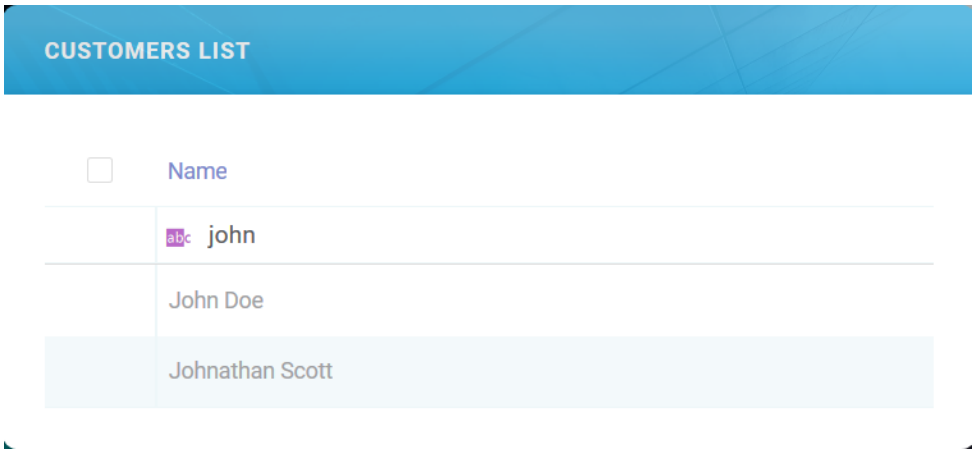
Optimization Search Data (Filter starts with)

By default, the checkbox is not selected, filter set to ‘contains’. This means that when Portal users enter only part of a query attribute (text, number) in one of the entity's "Data Views" on page 96, a filtering is done on records which contain the given part of the query. The search will return the filtered records that contain the part of the query.

For example, a portal user does a partial search (“john”) by Name on the Customers list. The search returns all names that contain “john”.

If the checkbox is selected, filter set to ‘starts with’, the partial search will return only the records that start with the part of the query.

For example, a portal user does a partial search (“john”) by Name on the Customers list. The search returns only names that start with “john”.

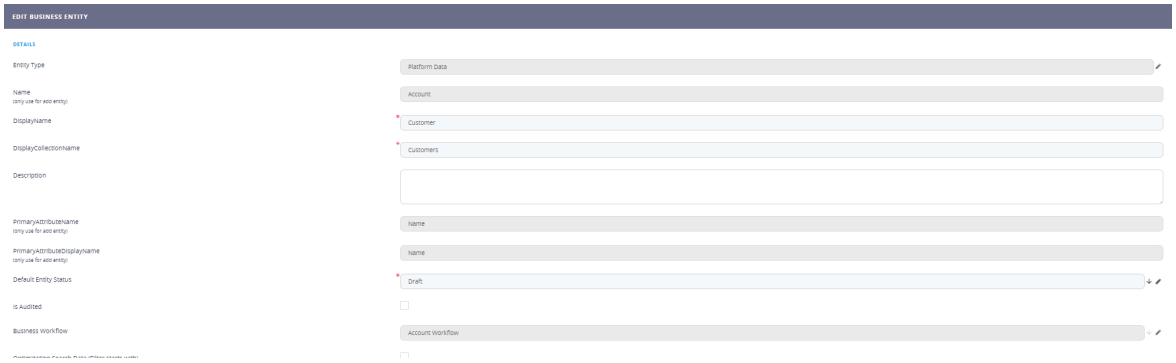


Editing Entities

When a new entity is created, a number of metadata and supporting system records are created for it.

You can edit an entity by editing existing attributes, adding more attributes (Data Model section), extend the data model with data extensions (Extend the Data Model section) and defining relationships.

IMPORTANT!
Attributes can be deleted from the Data Model, however if the attribute has records that have been inserted by an end-user, then the records have to be deleted first then delete the attribute.



NOTE
You cannot edit the following entity attributes: EntityType, Name,

PrimaryAttributeName, PrimaryAttributeDisplayName, once you configure it. The Business Workflow is read-only as well, but can be changed from the Buisness Workflows menu.

Deleting Entities

To delete entities, in the Data Model Explorer, select the entity or entities that you wish to delete and click the × button at the top-right corner of the screen.

IMPORTANT!
Deleting entities might have severe impact on the structure of your system's database; therefore, we strongly recommend you to make sure that the data model requires the entity deletion to address changes on existing business concepts.

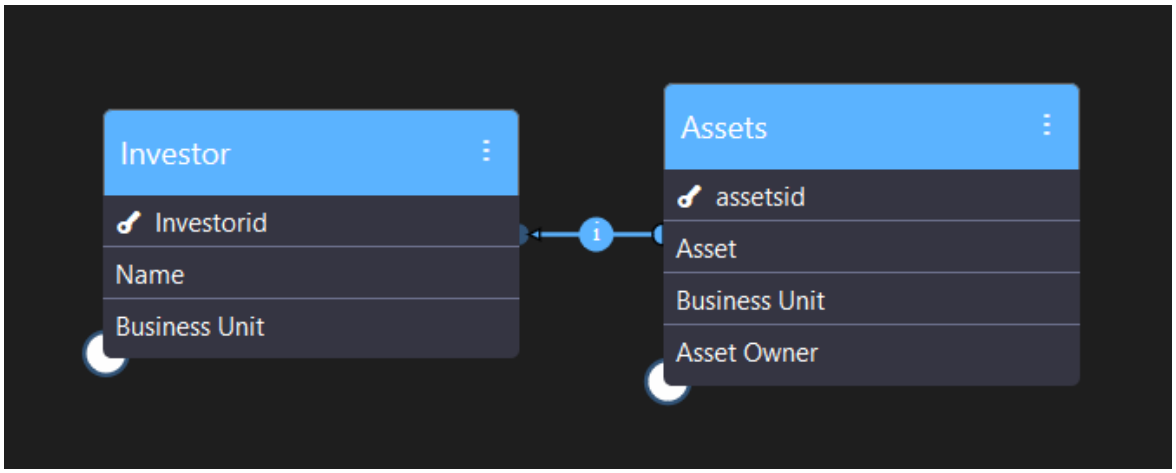
These are a few consequences you need to know before deleting an entity without prior analysis:

- Custom actions defined on deleted entities will fail.
- Widgets linked to a deleted entity will fail.
- Reverse engineering becomes an exhaustive process by dealing with a broken DB structure.

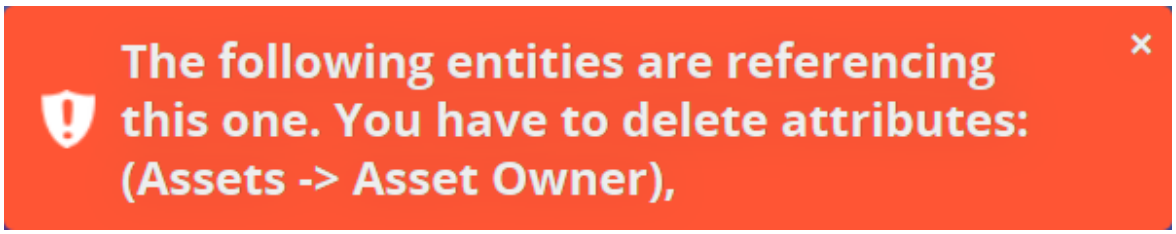
Referential Integrity Check

When deleting entities that are referenced by another entity, you first need to remove the relationship between the entities (delete the corresponding lookup attribute from the referencing entity).

In the example below, the Investor entity is referenced by the Assets entity.



Attempting to delete the Investor entity without deleting the relationship first, will generate the following warning:

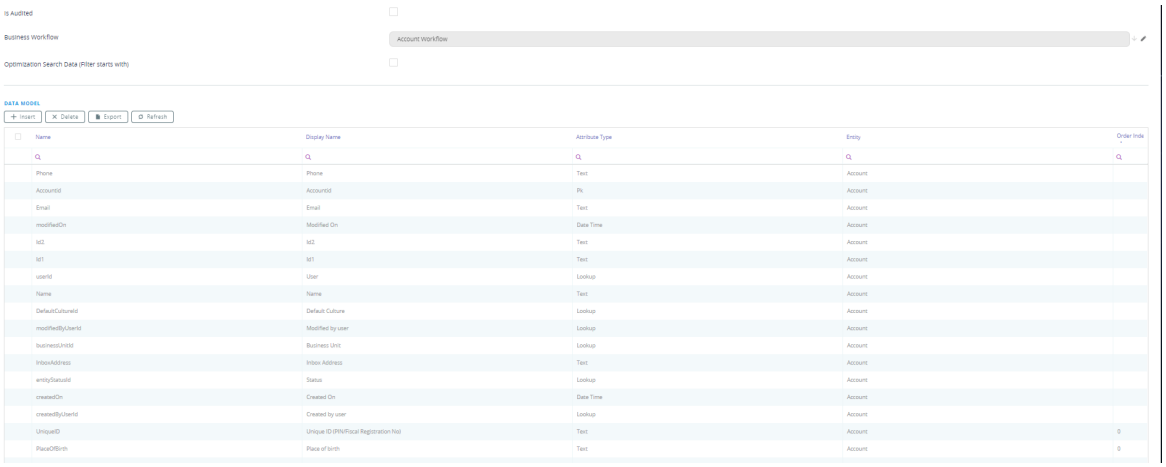


Attributes

An attribute is equivalent to a column in a table, available for the end-user to input or select data. For example, an entity is the ''Account'' and the attributes are ''name'', ''age'', ''product'', ''no.ofcontract'', ''policyno'', ''address'', ''income''. Attributes include primary keys and foreign keys (type lookup) as well.

FintechOS supports a variety of field types, from usual fields like: text, date, boolean or numeric, to advanced fields like lookup (referencing other entities) or optionset (drop-down list).

You can access the entity attributes from the Data Model section of the entity.



The buttons at the top left corner of the section have the following functions:

- Insert - Adds a new attribute. For details, see "Adding Attributes" on page 67.
- Delete - Deletes the currently selected attributes.
- Export - Exports the currently selected attributes' metadata in an Excel file.
- Refresh - Refreshes the list of attributes.

System-generated attributes

When you create an attribute, a new column is added within the table corresponding to your entity in the database.

> **IMPORTANT!**
> All entities have a set of auto-generated attributes that are used for entity auditing purposes. **DO NOT** remove them.

This is the list of system-generated attributes:

Attribute Name	Attribute Display Name	Type	Description
entitynameid	entitynameid	Primary key (PK)	The entity unique identifier, the Name provided when creating the entity name.
createdOn	Created On	Date Time	The date and time when the entity was created.

Attribute Name	Attribute Display Name	Type	Description
modifiedOn	Modified On	Date Time	The date and time when the entity was updated.
userId	User	Lookup	The current user or the owner of the entity record.
createdByUserId	Created by User	Lookup	The user who inserted that record.
modifiedByUserId	Modified by User	Lookup	The user who made the last updates on the entity record.
businessUnitId	Business Unit	Lookup	The business unit associated with the attribute userId. It is the business unit of the user.
entityStatusId	Status	Lookup	The status of the entity record.

HINT
The PrimaryAttributeName is also generated automatically because it is required to create an entity, but the actual name is chosen by the user.

Types of Attributes

This section describes the types of attributes (fields) you can add in Innovation Studio:.

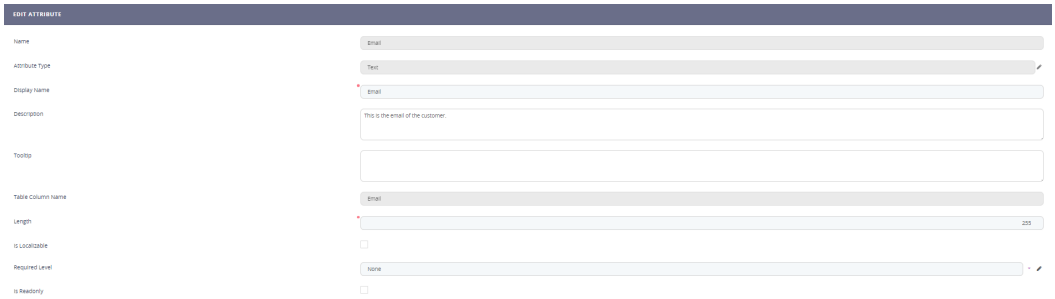
Text

A basic control that enables the user to type one-line set of characters (text). Use it for short alphanumeric attributes such as names or user IDs.

When adding a text attribute to an entity you need to provide the following specific properties:

Property	Description
Length (only for text)	The maximum number of characters users will be allowed to enter in field.
Is localizable (only for text)	If selected, the text fields will be marked as being localizable. For information on how to localize fields, see "Localization" on page 325

This is how a text field is configured.



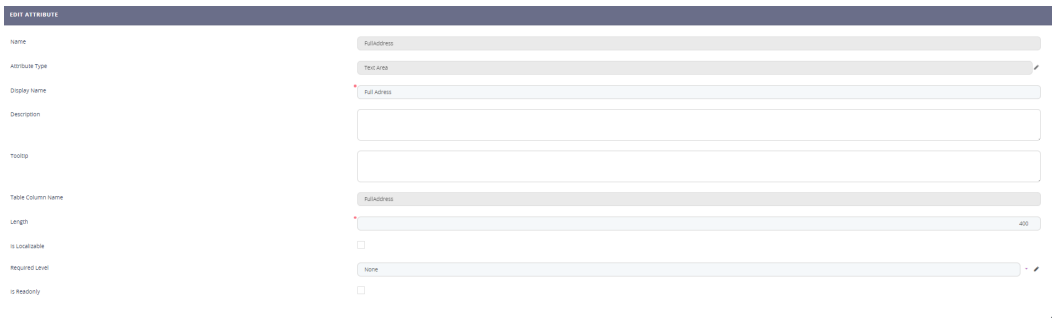
Text Area

Defines a multi-line text input control. Use it when users need to provide large amounts of text that exceeds one line such as descriptions, messages, feedback, etc.

When adding a text area attribute to an entity you need to provide the following specific properties

Property	Description
Length (only for text)	The maximum number of characters users will be allowed to enter in field.
Is localizable (only for text)	If selected, the text fields will be marked as being localizable. For information on how to localize fields, see Localization.

This is how a text area field is configured:



Whole Number

Use this field to insert any integer (whole number) with a value between -2,147,483,648 and 2,147,483,647. Use integers for attributes such as number of children, number of insured persons, number of cars, number of monthly

payments for a credit, maximum number of co-debitors, etc.

When you use integers, you can't add decimals. If you need to make very accurate calculations - for example, for calculating interest rates, and you need to utilize decimals, use numeric attributes instead.

When adding a whole number attribute to an entity you need to provide the following specific properties:

Property	Description
Is identity (it will be read-only)	If selected, the whole number attribute is automatically incremented for each record and becomes read-only. It is useful for unique record identifiers, such as: the contract number or the policy number.

The value inside the whole number fields is by default formatted to the right.

Numeric

Defines a field to enter numeric values. Use it when for attributes that require very accurate calculations, or if you typically use queries that look for values that are equal or not equal to another value, for example interest rates.

When adding a numeric attribute to an entity you need to provide the following specific properties:

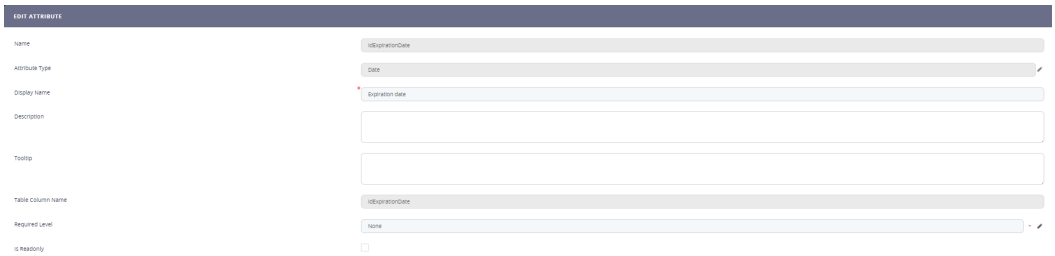
Property	Description
Precision	Specify the number of decimals (up to 9 decimal points of precision) to be displayed in the user interface in case of numeric fields.

Date

Defines a field which has the format option to display date only. No specific properties need to be provided. Use it for attributes such as birth dates, expiration dates, or issuing dates.

When you click on it, the calendar opens, allowing you to select a date.

This is how a date field is configured:



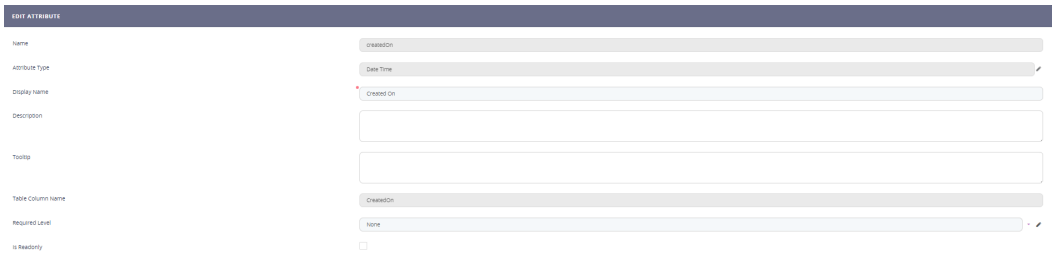
The date format is as defined on the ApplicationLanguage entity, that is, you can have different date formats per language. For details on how you can format the date field throughout FintechOS per language, see Add a New Language.

Date Time

Defines a field which has the format options to display the date and time. No specific properties need to be provided. Use it for attributes that need to record the precise date and time, such as registration when an issue was raised or when a fraud was attempted.

The format of the dateTime field is as defined on the ApplicationLanguage entity, that is, you can have different date formats per language. For details on how you can format the dateTime field throughout FintechOS per language, see Add a New Language.

This is how a dateTime field is configured:



Bool (Boolean)

Defines an attribute which is a checkbox. It can have one of the following values:

- **NULL** - it is the default state of the bool attribute. Indicates that no action has been performed on the checkbox yet.
- **TRUE** - it indicates that the checkbox has been selected.
- **FALSE** - it indicates that the checkbox has not been selected.

Use it for validations, for instance if the customer is a politically exposed person or if he has a criminal record or not.

Lookup

Defines a relation between the entity you're working on and another entity (that is, the parent entity), for example an Asset Owner attribute in an Assets entity that refers to the records in the Customers entity.

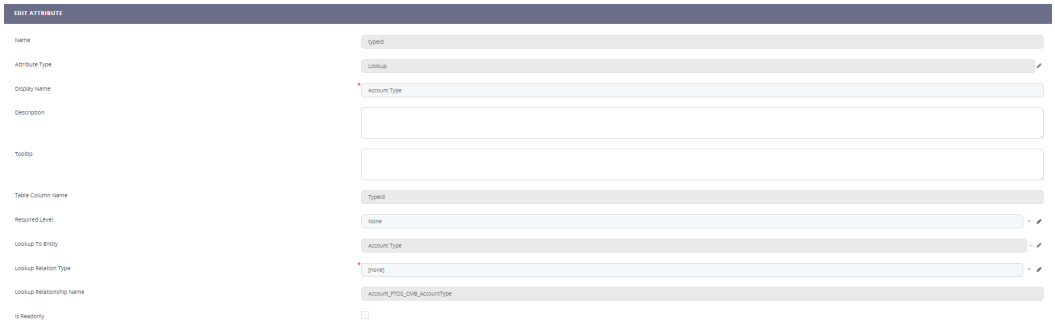
When you create a new lookup attribute you are creating a new Many-to-One (N:1) entity relationship between the entity you are working with and the **Lookup to Entity** defined for the lookup.
All lookup attributes display the primary attribute name of the referenced entity; therefore, you should always provide a value for the primary attribute on the parent entity.

When adding a lookup attribute to an entity you only need to provide the following properties:

Property	Description
Name	The name of the attribute. Make sure that you use the following naming convention: 'the referenced entity' + suffix 'Id', pascal case, no special characters and no blank spaces. For example: a lookup on the 'Contract' entity for the 'ContractType' entity will be named 'ContractTypeId'.
Attribute Type	Select **Lookup** from the drop-down list. The **Lookup to Entity (only for lookups)** field becomes mandatory.
Display Name	The name of the attribute that will be displayed on the data form in the user interface if the **Auto-generate data form** checkbox is selected on the entity level.
Description	Insert a proper description for the attribute.
Tooltip	Insert a proper tooltip to help the user understand what it is about.

Property	Description
Table Column Name	The name of the attribute which will be automatically generated in the database. This field is not visible in the end-user interface. **NOTE** To avoid affecting data integrity and consistency, do not change the value prefilled by the system.
Required level	Select whether the attribute if mandatory to fill in or not: • none • recommended • required.
Lookup to Entity	The parent entity for the entity you are currently working on.
Lookup relationship type	Select one from the list: • Dictionary (default) • IsChildOf (entity A is the child of related entity, it is a 1:N relationship, see "1:N Entity Relationships" on page 82) • One to one (this is a one-to-one relationship, see "Entity Relationships" on page 81)
Lookup Relationship Name	This field is automatically filled-in by the system with the concatenation of the two entity names, following this naming pattern: **ChildEntity_PK_ParentEntity**.

This is how a lookup attribute is configured:



The following actions are available on lookup fields:

- **Select record**: Opens the view with records existing within the referenced entity.
- **Edit record**: Opens the edit data form of the record selected within the lookup attribute.

Map

Defines a map to be displayed in the application. No specific properties need to be provided. Use it for attributes that display location data like where a business's headquarters is located on a map.

When displayed on a form, the user can scroll to zoom in and out, drag to pan the map, and click on the map to set/unset location markers.

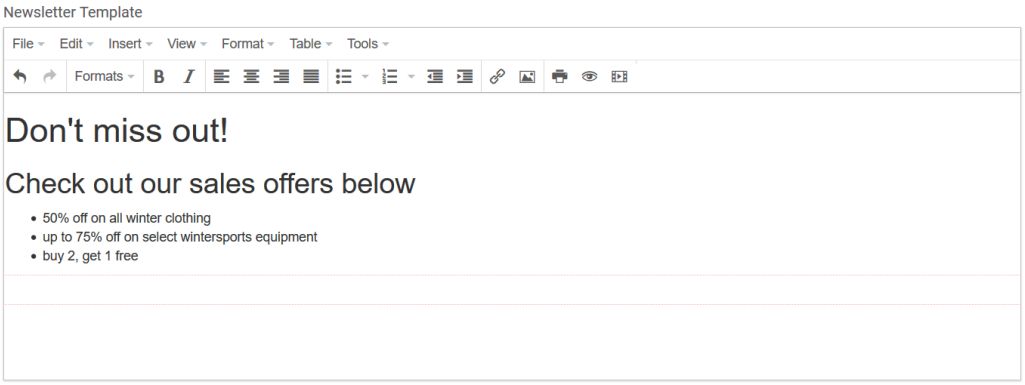
The actual data is saved in the database in the form of a JSON object that stores the geographic coordinates of the location markers and map zoom level.

```
{"markers" : [{"location" : {"lat" : 44.42753573214415,
"lng" : 26.08738040542604}}, {"location" : {"lat" :
44.426929932048424, "lng" : 26.1024179417852}}], "zoom" :
15}
```

HTML

Defines a HTML block of code. HTML fields allow displaying a rich text editor that can be used to quickly generate the underlying HTML code. It uses tinyMCE which interprets the HTML code.

HTML fields are particularly useful when you want to use customer tailored content, for instance to create newsletter templates to be included in your marketing campaigns.



The underlying HTML code that is stored in the database for the example above is:

```
<h1>Don't miss out!</h1>
<h2>Check out our sales offers below</h2>
<ul>
    <li>50% off on all winter clothing</li>
    <li>up to 75% off on select wintersports equipment</li>
    <li>buy 2, get 1 free</li>
</ul>
```

Color

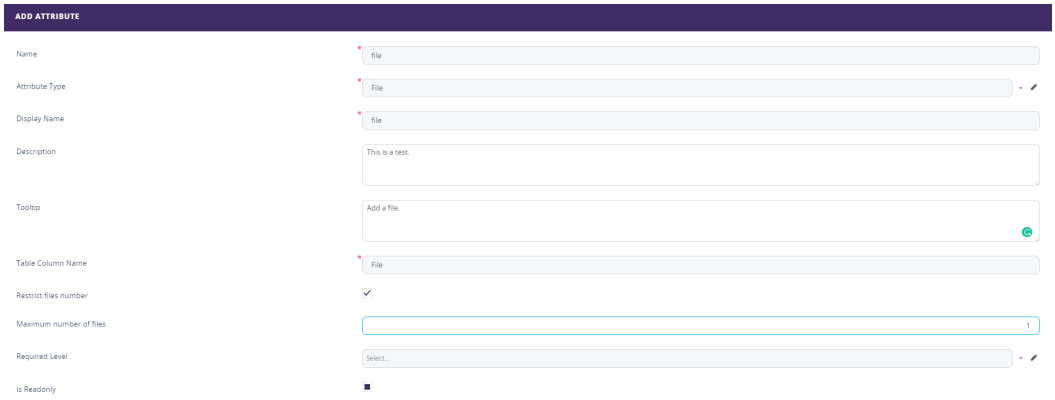
Allows selecting and displaying a color. No specific properties need to be provided. Use it for cosmetic customizations, such as user interface themes.

File

A control that allows end-users to add (upload) an attachment, either by clicking the **Add file** button or by dragging and dropping the file in the corresponding section. Use when the user needs to upload or download a file attachment such as a contract, agreement, or statement.

The uploaded files are stored in the file upload folder configured on the environment. For details about file storage, see the FintechOS Platform Administration Guide.

When adding the file attribute, a new bool will appear ''restrict files number'' to limit the number of file added to the attribute and a secondary filed ''maximum number of files''.



When the bool is true the automatically filed number of maximum files is one, but you can modify it to accept more files. If the bool is false, the number is unlimited. You can add as many file as you wish.

Security considerations you need to know before using File attributes

The malware detection and the file-type upload verification are available on file upload; however, they are disabled by default.

In order to make sure that the files that you upload are malware-free, you need to enable the malware detection feature. To do so, on the server where the FintechOS installation package resides, go to the web.config file, open it and add the following setting:

```
<appSettings>
     ....
     <add key="feature.upload.malware-
detection" value="true" />
</appSettings>
```

NOTE
The Antimalware Scan Interface is available starting with Windows Server

2016 and Windows 10; therefore, if you are using prior versions of these tools, you cannot use the malware detection in FintechOS .

JavaScript (JS)

JS fields allow usage of JavaScript code on data form level. No specific properties need to be provided. Use it for advanced customizations that you wish to add through the user interface.

Order Index

Allows you to drag and drop the rows existing in a grid (view). The order index attributes are not displayed on data form level but are used only in views to order attributes by a particular index number.

When adding a text attribute to an entity you need to provide the following specific properties:

Property	Description
Order index attribute reference (only if order index)	The attribute based on which the records in views will be ordered by.

In order to display optionset items in a particular order in the drop-down list, drag and drop in the order to display them.

Option Set

An option set attribute allows you to define a list with several options available for selection. Use it when the attribute can take a single value from a limited set of options, such as country, city, currency, etc.

When you add an **Option Set** field to a data form, you can specify multiple values that will be available for users to select. For more information, see Add an Option Set Attribute.

Money

It defines a price field which has included a thousand separator. No specific properties are required. Use it for monetary values such as credit values, interests, fees, etc.

Icon Picker

Allows you to select an icon from the list of available (predefined) icons and display it based on your preference. This type of attribute is currently embedded within FintechOS to allow selection of icons to be displayed on shortcuts that will be pinned on the homepage.

To select an icon, you have to click in the **Icon URL** field, and select the desired icon from the icons selection pane which will be displayed.

Invariant Date

Defines a field which has the format option to display date only. This type of field takes into consideration different timezones, Daylight saving time, winter time or leap a specific day based on year (e.g. February 28). Use it for attributes such as birth dates, expiration dates, or issuing dates.

No specific properties need to be provided.

You can use it to define dates like: Inception Date, Start Date, End Date or Due Date.

Unique identifier

This type of attribute is structured with 36 characters from the hexadecimal system (from 0 to F), with four lines.

For example, 43F48DD2-66E7-4ECC-9940-0219F8A5973F.

CSS

Cascading Stylesheets styles and structured the manner in which the HTML code is displayed.

This is how it will look in the FintechOS Portal after you add the code.

Css

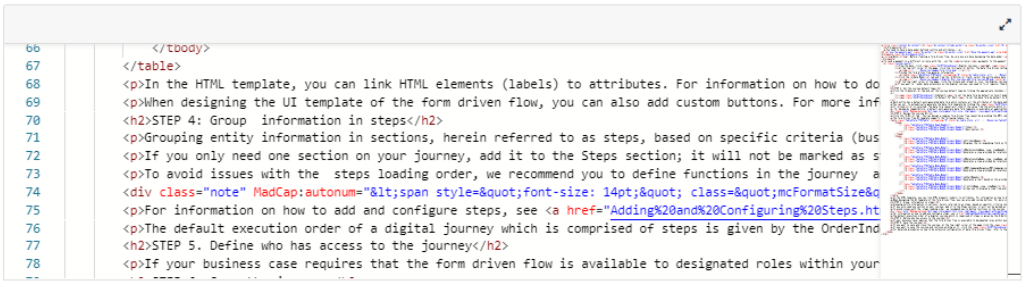
```
  font-size: 16px;
  font-smoothing: antialiased;
  text-rendering: optimizeLegibility;
}
html {
  font-size: 10px;
  -webkit-tap-highlight-color: rgba(0,0,0,0);
}
html {
  line-height: 1.15;
}
.dx-content .dx-anchor__icon:before {
```

HtmlRaw

Displays the filed where you can add HTML cod with Monaco editor and HTML Intellisense support.

This is how it will look in the FintechOS Portal after you add the code.

htmlraw



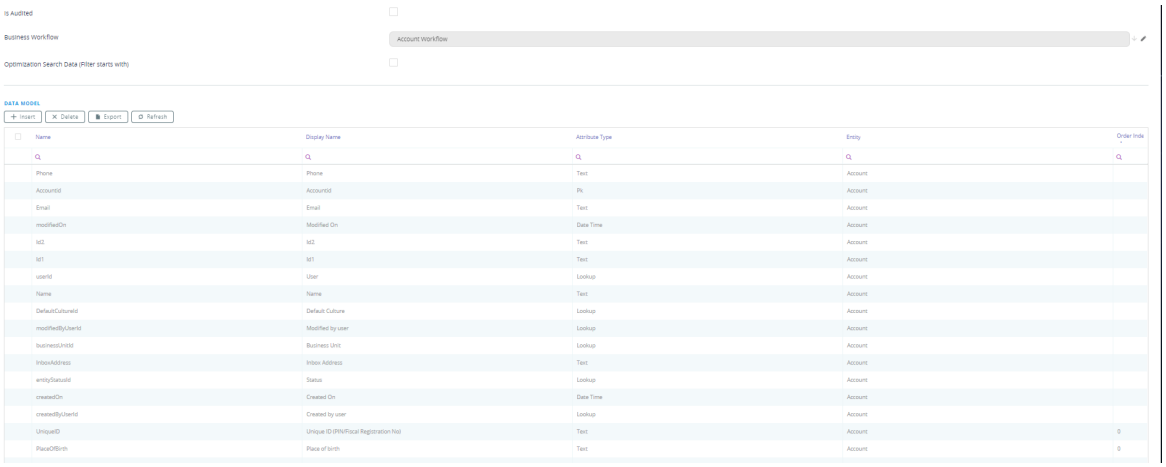
```
        </tbody>
    </table>
    <p>In the HTML template, you can link HTML elements (labels) to attributes. For information on how to do
    <p>When designing the UI template of the form driven flow, you can also add custom buttons. For more inf
    <h2>STEP 4: Group  information in steps</h2>
    <p>Grouping entity information in sections, herein referred to as steps, based on specific criteria (bus
    <p>If you only need one section on your journey, add it to the Steps section; it will not be marked as s
    <p>To avoid issues with the  steps loading order, we recommend you to define functions in the journey  a
    <div class="note" MadCap:autonum="&lt;span style=&quot;font-size: 14pt;&quot; class=&quot;mcFormatSize&q
    <p>For information on how to add and configure steps, see <a href="Adding%20and%20Configuring%20Steps.ht
    <p>The default execution order of a digital journey which is comprised of steps is given by the OrderInd
    <h2>STEP 5. Define who has access to the journey</h2>
    <p>If your business case requires that the form driven flow is available to designated roles within your
```

RawText

It renders information from the log file. It is not validated XSS.

Adding Attributes

To add new custom attributes to an entity, from the Edit Business Entity page, expand the Data Model section by clicking on it, then click the **Insert** button. The Add Attribute page appears where you will provide the properties.



NOTE

Based on the attribute type you select from the **Attribute Type** drop-down list, you need to provide details corresponding to that specific attribute. For more information, see the Types of Attributes.

This is the generic list of properties you need to provide when adding a new attribute (field).

Property	Description
Name	The name of the attribute. This is used to identify the attribute in the data model when you design the user interface, for instance to specify which attribute is displayed in a specific field. **NOTE** A naming convention is an important part in a well-built data model; therefore, we recommend you to use PascalCaseNames (upper camel), except for the first letter. The Name starts with a lowercase letter and all additional words start with an uppercase letter. Example: **a**ccount**I**d.
Attribute Type	From the drop-down list, select the type of attribute you want to add.
Display Name	The name of the attribute that will be displayed on the data form in the user interface. You can overwrite the **Display Name** using other commands directly in the HTML data form.
Description	Insert the proper description.

Property	Description
Tooltip	Insert the tooltip to be displayed. The message inserted here will show when the user will hover over the attribute in the FintechOS Portal. Tooltips can be a powerful UI pattern which help you guide your users to take specific actions within the product; thus, enhancing the user experience. If tooltips are activated on data form driven flows, for all attributes to which you want to show tooltips in the Digital Experience Portal, in the Tooltip field, provide the tooltip text. Optionally, you can add tooltips to specific attributes which can be shown in the Portal UI on data form driven flows.
TabelColumnName	This is the name of the table column.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>Required Level</td><td>From the Required Level drop-down list you can choose if a specific attribute (field) is going to be mandatory, recommended or optional:
None – The field is optional. No error message will be displayed if the field is not completed by the end-user.
Recommended – A blue dot will be displayed on the upper-left corner of the field in the user interface to indicate that it might be useful to fill in the field.
Required - A red dot will be displayed on the upper-left corner of the field in the user interface to indicate that it is a mandatory field. The end-user will not be able to add a new record if the field is not completed.
NOTE
You can only add required attributes to entities which have no records (empty entities). If you try adding a required attribute to an entity for which you already have required attributes stored within the database, you'll receive an error message.
You can add required attributes without creating constraints in the database, from the Forms section by using the After generate events field and the capabilities of field options.</td></tr>
<tr><td>isReadonly</td><td>The attribute is readonly if true, i.e. the front-end user will not be able to insert any data in this attribute.</td></tr>
</table>

Property	Description
isSecurable	Allows you to control access to the attribute using security roles. When this flag is checked, access to the attribute is restricted by default to all user roles except the administrator. To allow access to a secured attribute, you will have to explicitly configure security roles for this purpose (see "Creating Security Roles" on page 622 for details). **NOTE** This option is disabled for system-generated attributes.

Well-designed onboarding is vital for streamlining the user experience. Imagine users being lost within the app, having no clue what specific items on the UI mean or what actions they should take. If only they had the possibility to hover over specific fields and see some tips on what they should be doing.

Always make sure to save your configurations by clicking one of the save icons displayed on the top-right corner of the page.

Adding Option Set Attributes

Introduction

An option set attribute allows you to define a list with several options available for selection.

When you add an option set field to a data form, you can specify multiple values that will be available for users to select.

When users fill out the data form they can select one value displayed in a drop-down list.

Users who are familiar with Innovation Studio (know how to navigate through pages and pop-ups with minimal clicks and actions) can add an option set attribute by adding the option set attribute, then add options sets and items.

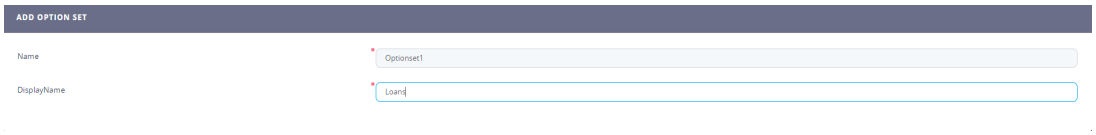
Innovation Studio beginners should define the options set, then add the option set attribute.

This section walks you through the steps you need to follow in order to add and define an option set attribute.

STEP 1. Define the option set (picklist)

1. On the main menu, click Admin and select Options Sets. The Option Sets List page will be displayed.
2. At the top-right corner of the page, click the Insert icon. The Add Option Sets page will be displayed.
3. Fill-in the following fields:

Field	Description
Name	The name of the picklist that will be used by the system. It is not visible in the user interface. The Name field value must be unique.
Display Name	The name of the drop-down list that will be displayed in the user interface.



4. At the top right-corner of the page, click the Save and reload icon. The Add Option Set page is replaced by the Edit Option Set page. The Items section displayed at the bottom of the page allows you to create, edit, delete and change the order in which options are presented.

Now that you've created the option set, you can start adding items.

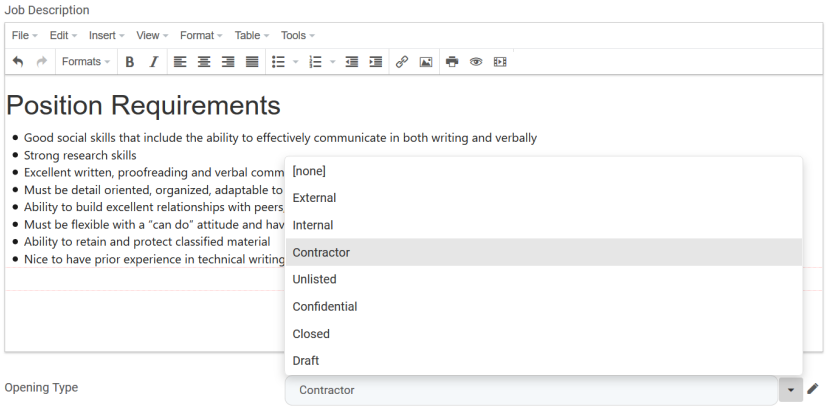
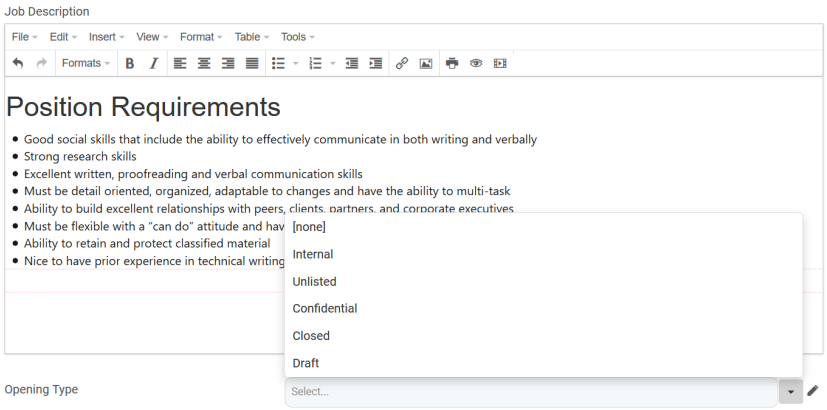
STEP 2. Add new option set items

To add items to an option set, double-click the desired option set and in the Edit Option Set page follow these steps:

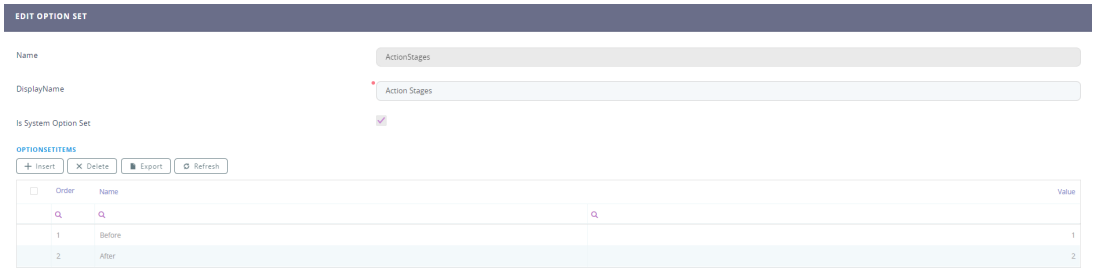
1. From the Items section, click the Insert button. The Add Option Set Item page will be displayed.

2. Fill-in the following fields:

Field	Description
Name	The option set item name which will be used by the system.
Display Name	The option set name which will be displayed in the user interface. **NOTE** The maximum length of the Display Name for Option Set Items is 200 characters.
Value	It is used for mapping (e.g., if you want to get from a script the optionsetitem). The value must be unique within the options. We recommend that you add incremental values in this field as the value might be useful in workflows, flow automation and business rule engines.
Id	Unique ID automatically assigned by the system for the option set item. No user input is required.

Field	Description
StatusId	The drop-down has two options: Active and Inactive. Select **Active** to make the business option available. Option set entity attributes will not list inactive option set items in the user interface. For instance, an option set may contain values for internal, external, contractor, unlisted, confidential, closed, and draft job openings:  During a hiring freeze, a decision is made to suspend external and contractor job openings and the corresponding option set items are set as inactive, making them invisible in the user interface for any entity attribute that uses the option set:  Once the hiring freeze is over, the option set items can be activated again.

3. At the top right-corner of the page, click the **Save and reload** icon. The page refreshes and the option set option is listed in the Items section.



You can add as many option set items as you want to fulfill your business needs.

Changing the items order

You can change the order of the items by dragging the items from one position to another in the items list.

Editing option set items

You can edit an existing option set item from the Edit Option Set page (on the main menu, click Admin and select Options Sets> click the desired option set). In the Items list table double-click on the desired item. The Edit Option Set Item page will be displayed where you can change the Value, Display Name and Status Id.
To save the changes, click the Save and reload icon from the top right-corner of the page.

Deleting Option Set Items

To remove an option set item you have to select it from the Items list by clicking the checkbox that is found in front of it. After you have selected the item that you want to remove, click on the Delete button that is found above the items list. A confirmation dialog will be displayed. Click **Yes** to confirm the deletion of the selected option set item.

> **IMPORTANT!**
> If you remove an option that has already been used in entity records, the data in those entity records becomes invalid; therefore, you should not remove the option but mark the option as obsolete. To do so, double-click it and from the **Status Id** drop-down list select **Inactive**.

STEP 3. Define the option set attribute

To add a new Option Set attribute to an entity, follow these steps:

1. From the menu, click Evolutive Data Core > Data Model Explorer. The Business Entities List page will be displayed.
2. Click on the entity name to which you want to add the option set attribute. The Edit Business Entity page will be displayed.
3. Expand the Data Model section by clicking on it, then on top of the section, click the Insert button. The Add Attribute page will be displayed.
4. Provide the following properties:

Property	Description
Name	The name of the attribute. Make sure that you use the following naming convention: 'optionset name' + suffix 'Id', **Example:** an attribute 'AccountTypeId', where 'Account Type' is the optionset name.
Attribute Type	Select **Option Set** from the drop-down list. The Option set name (only if optionset) field becomes mandatory.
Display Name	The name of the attribute that will displayed on the data form in the user interface if the **Auto-generate data form** checkbox is selected on the entity level.
Table Column Name	The name of the attribute which will be automatically generated in the database. This field is not visible in the end user interface. **NOTE** To avoid affecting data integrity and consistency, do not change the value prefilled by the system.

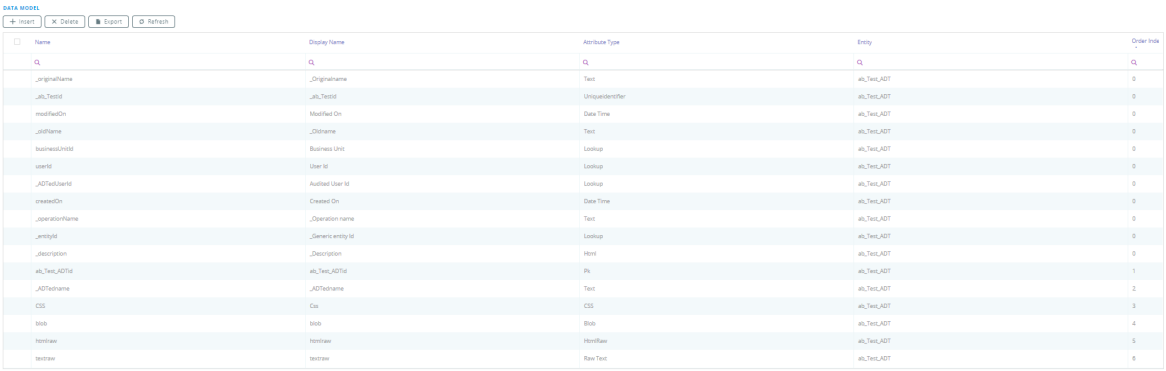
Property	Description
Option Set	The name of the option set (picklist) you defined. You can either enter the option set name or click the drop-down arrow on the right side of the field and double-click it. If you have many option sets displayed within the list, you can use the search feature available to search the one you want to add to the option set attribute.
Option Set - default value	This property allows you to display a default value in the drop-down list on the data form. For usability purposes, enter the value which is more likely selected in most cases by users from the drop-down list on the entity data form.

Example: Create an option set

Reordering Entity Attributes

If you need to change the order of attributes in an entity, go to the edit configuration page of that entity (by double-clicking on the entity in the Business Entities List page), scroll-down to the Data Model section and drag and drop attributes from one position to another in the list. The order index will be automatically updated based on the change you made.

If there are many attributes in the list, the "Please wait..." message will be displayed until the DB is updated. Once the update completes, the platform is loading the attributes in the new order and a message displays at the bottom of the page indicating for how many records the order index has been updated.



Entity Unique Constraints

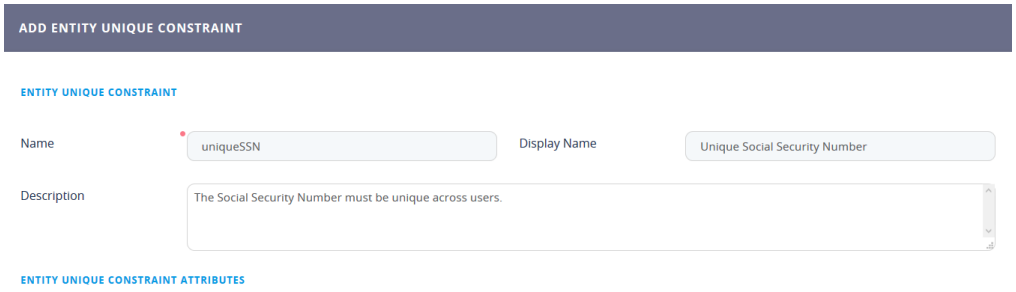
Unique constraints allow you to define attributes or combinations of attributes that must have unique values for each entity record. For instance, in an entity that stores personal data, you may want to set up the Social Security Number to be unique for each person. Or, in an entity that stores invoices, you may want to configure the combined invoice number and invoice series to be unique for each invoice.

Once enabled, unique constraints prevent record inserts or updates that do not meet the defined uniqueness criteria. You can define multiple constraints for the same entity.

Create a Unique Constraint for an Entity

1. Open the Main Menu ( ) in Innovation Studio.
2. Select **Evolutive Data Core**.
3. Select **Data Model Explorer**.
4. In the list of business entities, double click the entity for which you wish to add uniqueness constraints.
5. In the Edit Business Entity page, expand the **Entity Unique Constraints** section and click the **Insert** button.
6. In the **Add Entity Unique Constraint** page:
    a. Enter a **Name** for the constraint. You cannot have multiple constraints with the same name for the same entity, but you can reuse constraint names on different entities.
    b. Optionally enter a **Display Name** for the constraint. This is the name that will be displayed in the user interface.

c. Optionally enter a **Decription** for the constraint.



7. Click the **Save and Reload** ( ) button at the top right corner of the page.

Add Unique Constraint Attributes

1. In the entity unique constraint page, in the **Entity Unique Constraint Attributes** section, click the **Insert** button to add an attribute

> **NOTE**
>
> You cannot insert attributes in enabled constraints. If your constraint is enabled, you must disable it first (see "Disable a Unique Constraint" on page 81 for details).

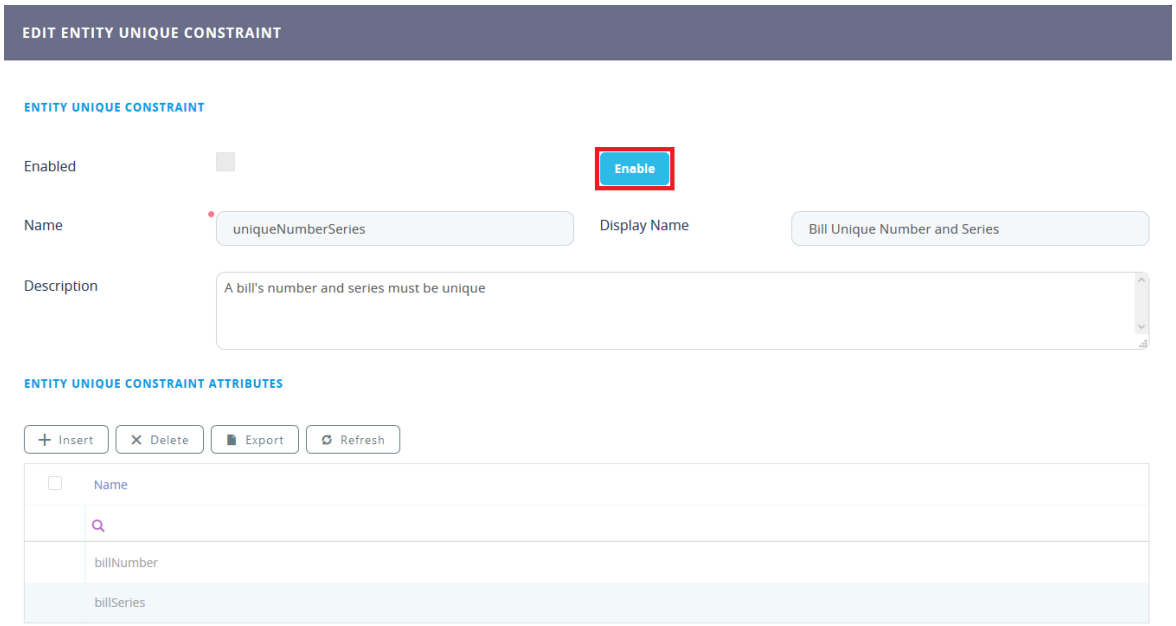
2. Select an **Attribute** that you wish to include in the constraint, from the drop-down list. The **Name** field will be filled-in automatically.

3. Click the **Save and Close** button ( ) at the top right corner of the page.

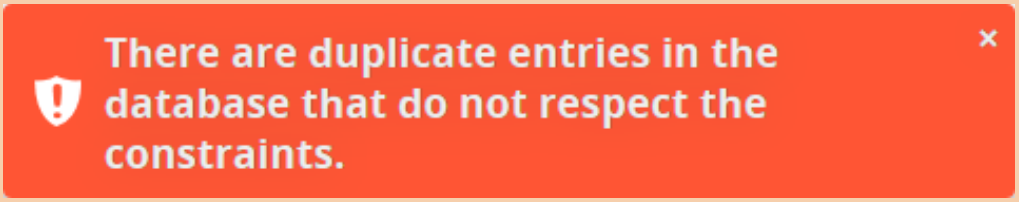
4. Repeat for any additional attributes you wish to include in your constraint. If you add multiple attributes to a constraint, the constraint will evaluate if all the attributes combined are unique, not if each attribute is unique.

Enable a Unique Constraint

When you create a unique constraint, it is disabled by default. To enable the constraint, click the **Enable** button in the entity unique constraint page.

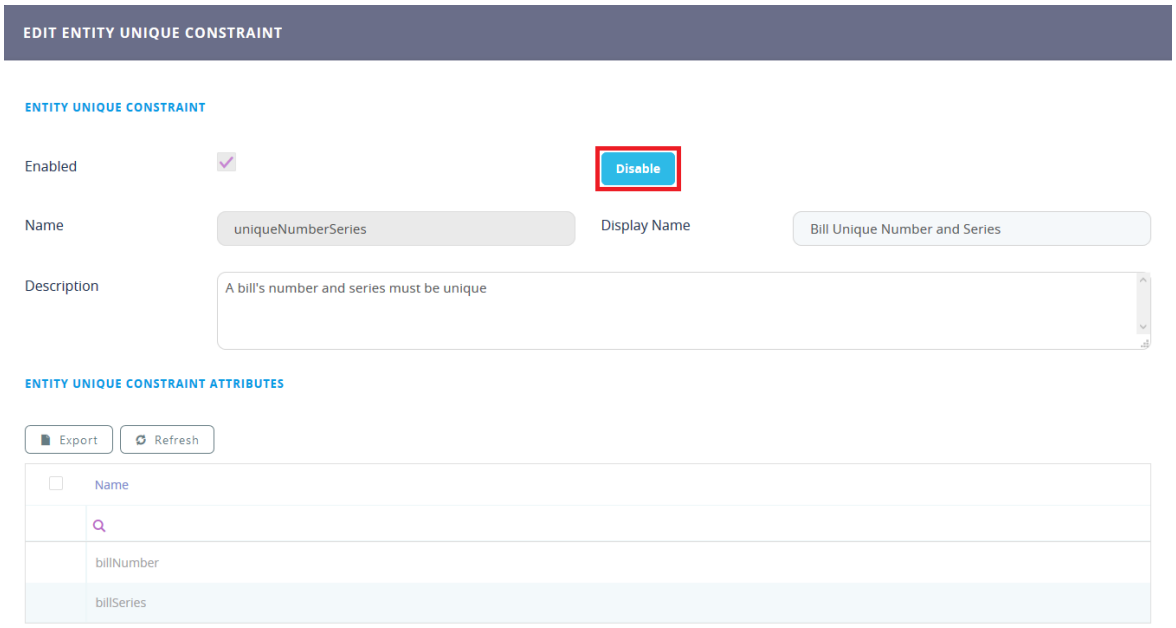


IMPORTANT!
If the existing entity records don't meet the constraint requirements, you will not be able to enable the constraint. The following message will be displayed:



Disable a Unique Constraint

To disable an enabled constraint, click the **Disable** button in the entity unique constraint page.



Entity Relationships

Entity relationships define how records can be related to each other in the database, their associations and dependencies.

There are two types of entity relationships in Innovation Studio:

- 1:N (One-to-Many) - an entity relationship where one entity record for the primary entity (parent entity) can be linked to many other entities (child entities).

  When viewing a parent entity in the user interface, you can also view the list of child entities. For more information on 1:N entity relationships, see "1:N Entity Relationships" on the next page.

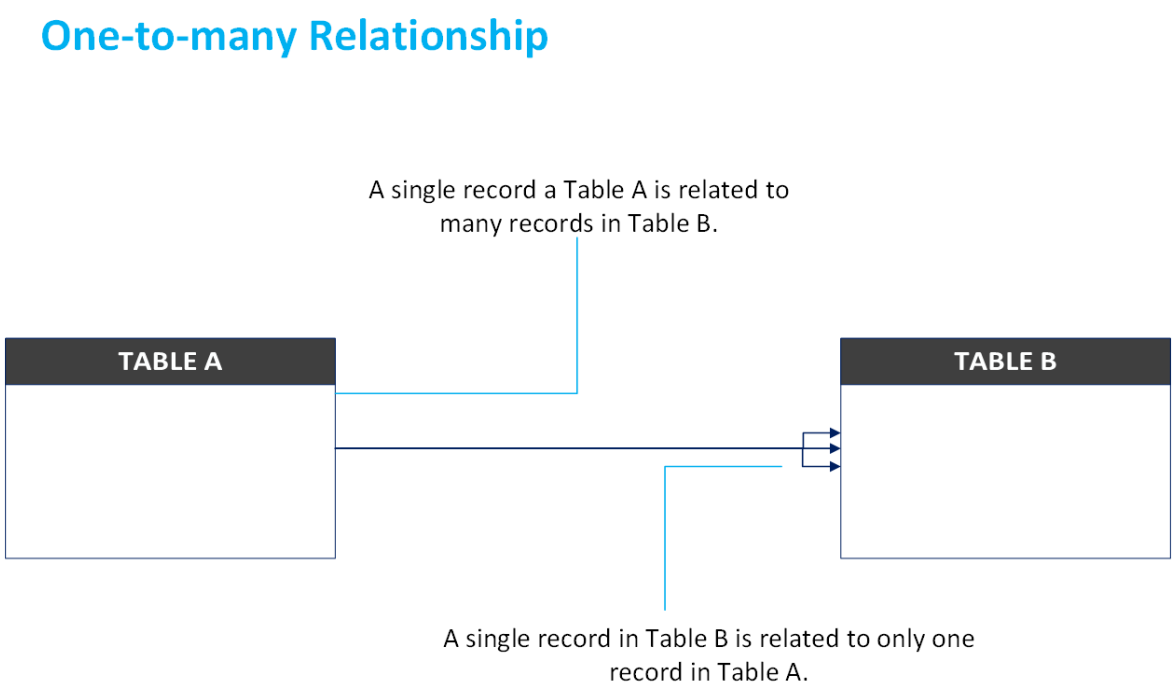
- N:N (Many-to-Many) - an entity relationship where many entity records can be linked to many child entities. For more information on N:N entity relationships, see "N:N Entity Relationships" on page 85.

1:N Entity Relationships

Introduction

A one-to-many relationship is not a property of the data, but a property of the relationship itself.

The one-to-many relationship is only a principle of relational database design. It cannot be explicitly defined in the database structure, but created and enforced by the use of relationships between a parent entity and child entities. One record in a table can be associated with one or more records in another table:



In a one-to-many relationship, the parent entity might have zero, one or multiple child records while a child entity can have one and only one parent entity record.

In Innovation Studio, you can create one-to-many relationships by adding a lookup attribute on the child entity to reference the parent entity. When adding a lookup attribute, the **Lookup Relationship Name** field will be automatically filled in by the

system with the concatenation of the two entity names, following this naming pattern: **ChildEntityName_ParentEntityName**. It allows you to display on entity the following:

- the lookup attribute referencing the parent on child data form
- the list with child records on parent entity

Creating a one-to-many relationship

This section provides the use case scenario for recording customer sales transactions over time. The database will contain two tables, as follows:

- The Investor table used to keep the customer details. The table primary key is the Investorid column.
- The Assets table used to keep the track of individual investor assets. It contains the assetOwner foreign key which references the Investorid column in the Investor table to track the customer to whom the asset belongs.

A single asset can only be associated to one investor while one investor can have many assets. The logic is defined by the one-to-many relationship:

To create a one investor-to-many assets transactions, you need to create a lookup attribute on the Assets entity. To do so, follow these steps:

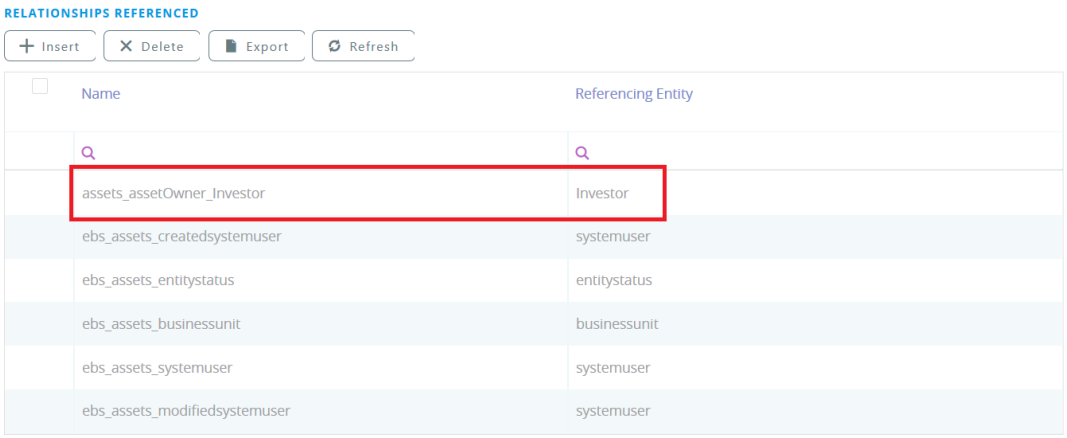
1. From the menu, click Evolutive Data Core > Data Model Explorer. The Business Entities List page appears.
2. Search for the 'Assets' entity and double-click it. The Edit Business Entity page will be displayed.
3. Scroll-down to the Data Model section and click on the section header. The Data Model section expands.
4. Click the Insert button. The Add Attribute page will be displayed.
5. Enter the attribute properties as follows:

Property	Value
Name	assetOwner
Attribute Type	From the drop-down list, select **Lookup**.
Display Name	Asset Owner
Lookup to Entity	Click the down arrow icon from the right side of the field. From the newly displayed page, select **Investor** and double-click it.
Lookup Relationship Name	**assets_assetOwner_Investor** is automatically filled in this field after making the entity selection in the Lookup to entity field.

6. At the top-right corner of the page click the Save and close icon. The lookup attribute will be saved.

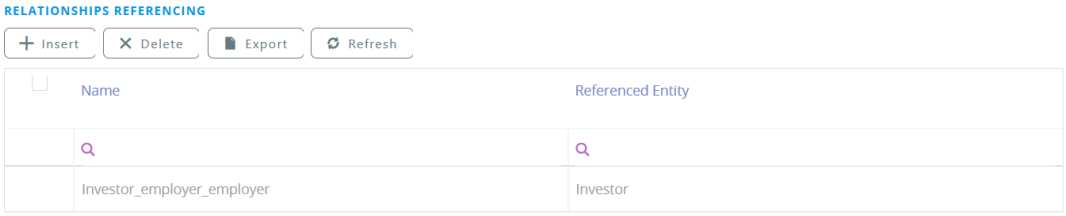
The newly created relationship is displayed on both entities, as follows:

- On the 'assets' entity the relationship is displayed within the Relationships Referenced section, as this section lists the relationships where the current entity is a child.



- On the 'Investor' entity it is displayed within the Relationships Referencing section, as

this section lists the relationships where the current entity is a parent.



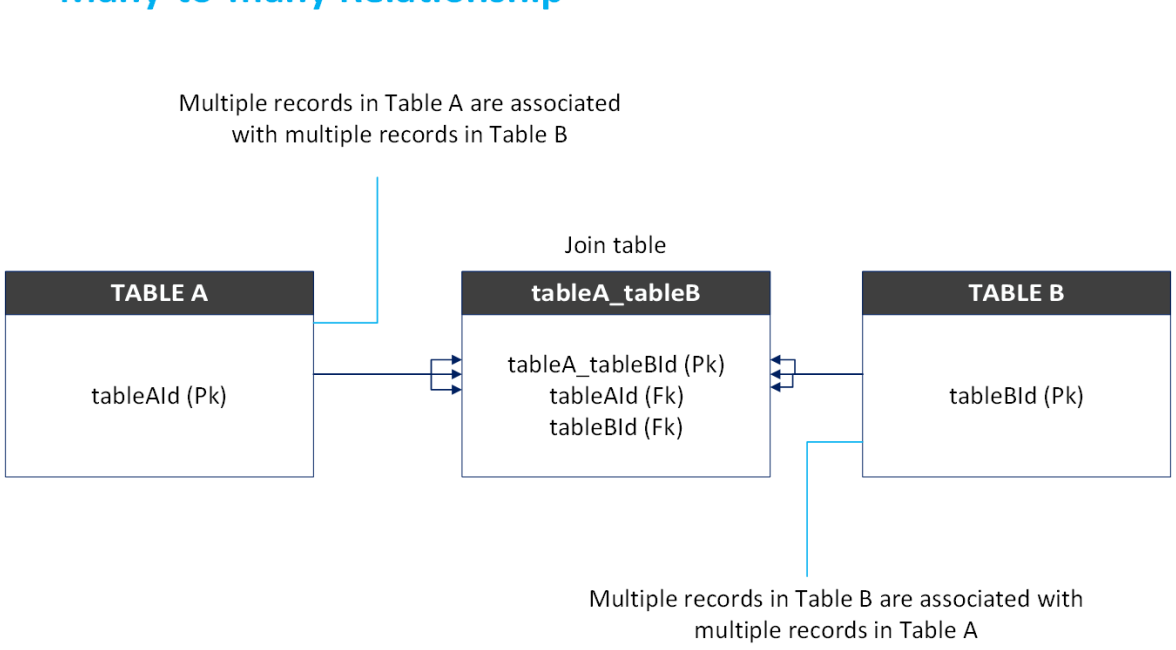
N:N Entity Relationships

Introduction

The N:N entity relationship is used to relate many records of one entity to many records of another entity.

Relational databases do not allow implementing a direct many-to-many relationship between two tables, therefore you cannot create a direct N:N relationship between two entities.

To create a N:N entity relationship, you will need to use an intersect/join entity (table) and create two one-to-many relationships between the two entities and the join entity:



Each record in a join table includes a match field (foreign key) that contains the value of the primary keys of the two tables it joins. The foreign key fields are populated with data when records in the join table are created from either table it joins.

Example of a N:N entity relationship is the relationship between employees and departments. Each department can have any number of employees working on a specific task. Also, an employee can work for multiple departments at the same time. Therefore, any number of departments or employees can simultaneously be linked to each other by creating a join table (Employee_Department) that links them using the Employeeid and the Departmentid.

Creating a many-to-many relationship

In Innovation Studio you can create a many-to-many relationship between two entities by adding a relationship referencing another entity and entering value **2** in the Relationship Type field.

When adding a Relationship referencing another entity you can choose how the system will behave when deleting records which belong to entities linked through relationship. To do so, you will have to enter one of the following values in the Relationship Constraint (0-none, 1-restrict, 2-cascade) field:

- **0** - There will be no restrictions on deleting entity records that have been linked by other entities. When deleting the entity record, the value null will be displayed in that field for all entities linked to that entity. For example, if deleting a product, then all the customers who had the deleted product will have the value "null" in the field corresponding to that specific product.
- **Restrict** – Restricts the deletion of entity records that have been linked by other entities. For example, you are not able to remove a product that is owned by a customer.
- **Cascade** – Allows the deletion of entity records, including recursive cascades.

Use case scenario

Consider a database for recording customer products over time. The database will contain two tables, as follows:

- The Account table used to keep the customer details. The table primary key is the Accountid column.
- The Product table used to keep the track of individual products. The table primary key is the Productid column.

A many-to-many relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers.

To create a many customers-to-many products relationship, follow these steps:

1. From the menu, click Evolutive Data Core > Data Model Explorer. The Business Entities List page appears.
2. Search for the **Product** entity and double-click it. The Edit Business Entity page will be displayed.
3. Scroll-down to the Relationship Referencing section and click on the section header. The Relationship Referencing section expands.
4. On the top of the section, click the Insert button. The Add Relationship page will be displayed.
5. Enter the relationship properties as follows:

Property	Value
Name (only for view)	**Product_Name**. The value in this field should follow this naming convention: concatenation of the two entity names, **ChildEntity_PK_ParentEntity**.
Display Name	Customer
Referenced Entity	Click the down arrow icon displayed on the right side of the field. A new page is displayed. Select **Account** and double-click it.

Property	Value
Referencing Entity (only for view)	**Product** is automatically filled in this field (being the current entity you're working on, the child entity).
Relationship Type (only for view) (0-many to one, 1-one to many, 2-many to many)	Type **2**.
Relationship Constraint (0-none, 1-restrict, 2-cascade)	Type the desired value (0, 1 or 2) based on your business needs.

6. At the top-right corner of the page click the Save and close icon. The lookup attribute will be saved.

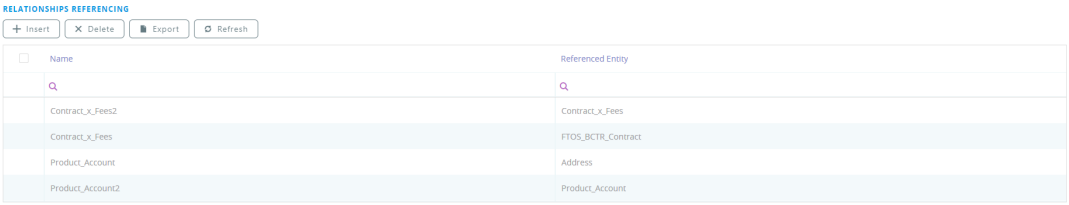
The many-to-many relationship is provisioned in the database and becomes visible in the user interface. Table Product_Account will be the link table between Account and Product.

To see the details of the relationship entity, go to the Entities list (Evolutive Data > Data Model Explorer), search for the **Product_Account** entity and double-click it. The Edit entity page will be displayed.

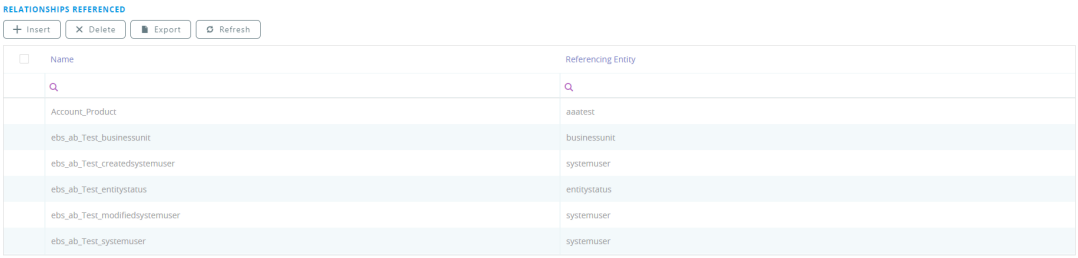
Scroll-down to the Data Model section and click on the section header. The Data Model section expands. It contains the relationship entity primary key attribute (Product_Accountid) and two foreign keys (lookup attributes) to the other two entities (Productid and Accountid):

The newly created relationship entity is displayed on both entities (Product and Account), as follows:

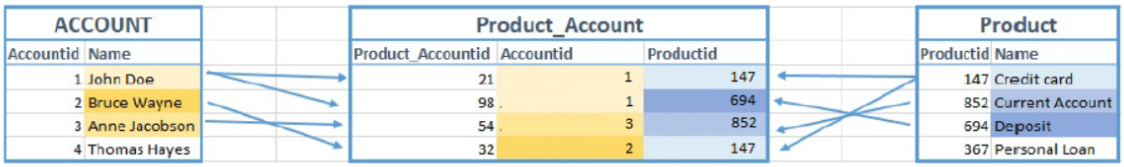
- On the Product entity the relationship entity (**Product_Account**) automatically created by the system is displayed in the Relationship referencing section: Both Product and Account entities reference the **Product_Account** table in the database.



- On the Account entity the relationship entity (**Product_Account**) is displayed in the Relationship referencing section. The referenced entity the **Product_Account** join table, corresponding to the many-to-many relationship between the Product and Account entities.



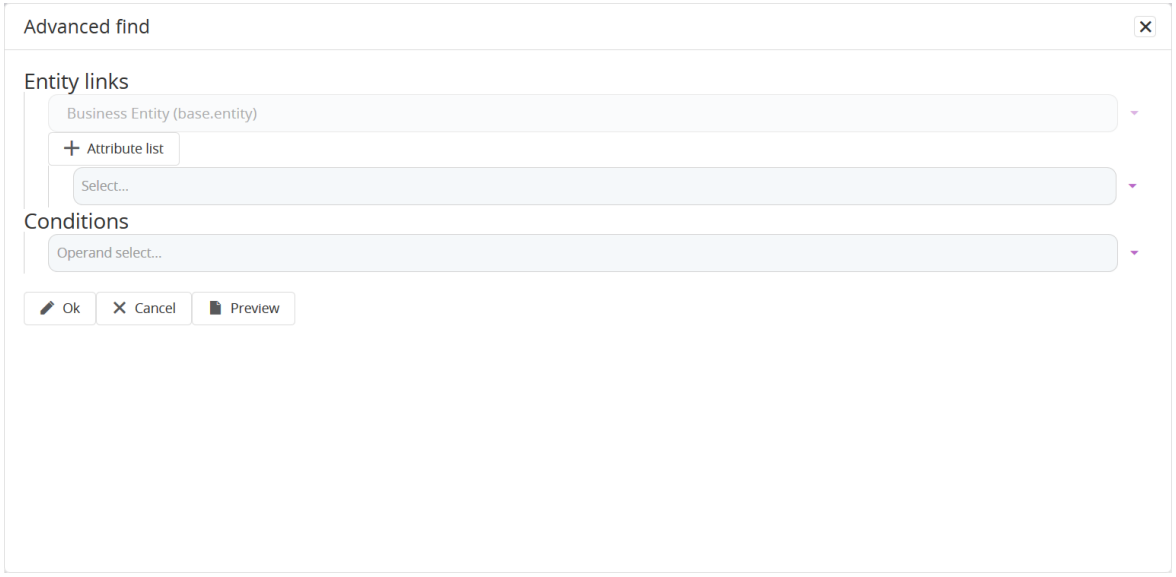
This is how the Product and Account tables will look like in the database when filled with data:



Advanced Entity Find

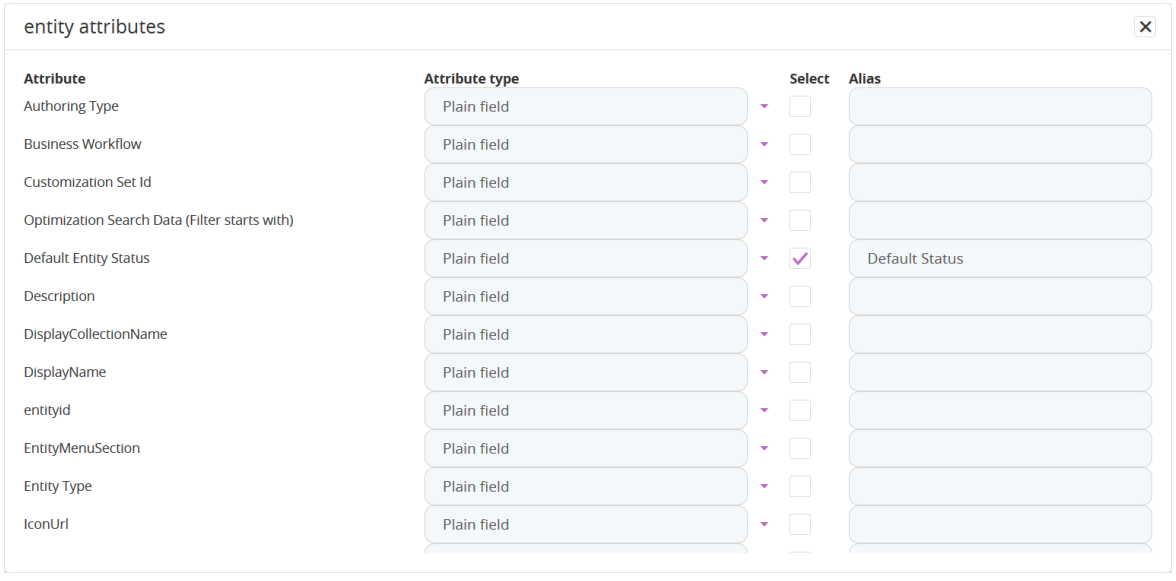
Advanced Find is a powerful DB segmentation tool which allows you to filter and customize the entity grid displayed in the Data Model Explorer.

To access Advanced Find, in the "Data Model Explorer" on page 41, at the top-right corner of the page, click the **Advanced find** ( ) icon. The Advanced find page will be displayed similar to the figure below. The page allows you to choose the criteria and conditions for displaying your entities.



Step 1. Select the attributes you wish to display

Click the **Attribute list** button to open the he Attributes List page. This page lists the attributes that all entities have in common.



Select the attributes you want to include in the fetch by clicking the corresponding checkbox in the Select column.

For all selected attributes, in the Alias column, type a unique alias name.

> **IMPORTANT!**
> The alias is used by the system to distinguish between various records; therefore, providing a unique alias for all selected attributes is mandatory. Do not use the same alias name for two different attributes.

Scroll-down at the bottom of the attributes list and click **OK**. The page listing the entity's attributes will be closed.

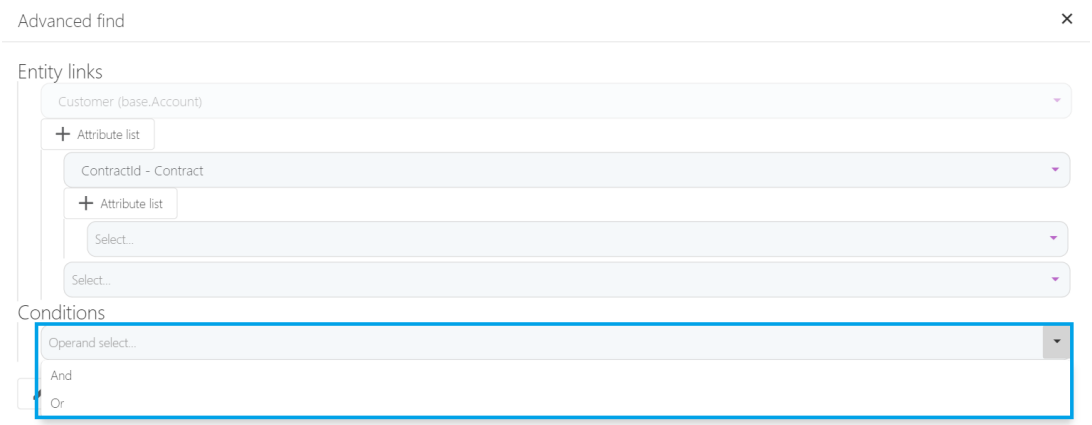
STEP 2. Apply Filtering Conditions

From the Conditions section you need to select the conditions which must be met in order to include a record into the data fetched from the database.

You can apply multiple conditions which need to be met separately or in combination by selecting either AND or OR as logical operator.

To set up the fetch entity data conditions, follow these steps:

1. From the drop-down field displayed in the Conditions section, select the logical operator:



AND – to apply multiple conditions which need to be met separately.

OR – to apply multiple conditions when at least one condition needs to be met.

You can apply multiple conditions by using as many logical operators as you need. Once you select one, another logical operator field will be displayed beneath the first one, and so on.

2. Select the entity on which the condition is to be applied.

> **NOTE**
> In Advanced Find, you can only select the generic base entity.

After selecting an entity, the **Attribute select** field will be displayed below the selected entity.

3. Click in the **Attribute select** field. A drop-down list will be displayed, listing only the attributes that all entities have in common.

4. Select an attribute existing on the entity previously selected. Below the selected attribute, the Select operand field will be displayed. The table below describes the operands you can choose from.

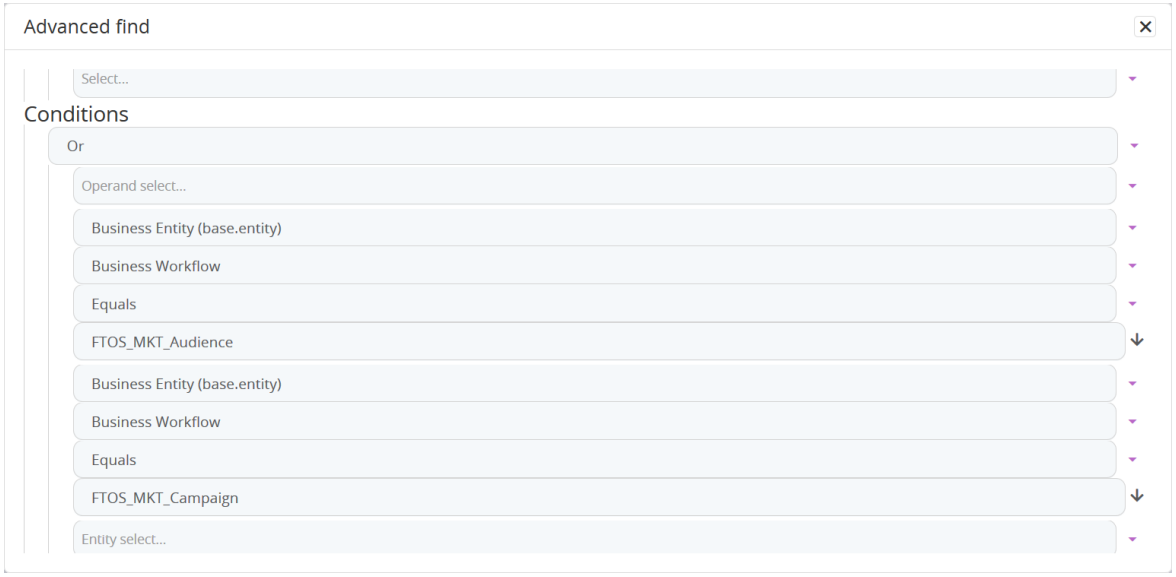
Operands	Description
Contains	Records will be included into the data fetched from the database if the value of the selected attribute contains the value given in the field displayed below the selected operand field.
NotContains	Records will be included into the data fetched from the database if the value of the selected attribute does not contain the value given in the field displayed below the selected operand field.
Equals	Records will be included into the data fetched from the database if the value of the selected attribute equals the value given in the field displayed below the selected operand field.

Operands	Description
NotEquals	Records will be included into the data fetched from the database if the value of the selected attribute is different than the value given in the field displayed below the selected operand field.
Greater Than	Records will be included into the data fetched from the database if the value of the selected attribute is greater than the value given in the field displayed below the selected operand field.
Greater Than or Equal	Records will be included into the data fetched from the database if the value of the selected attribute is greater than or equal to the value given in the field displayed below the selected operand field.
Lower Than	Records will be included into the data fetched from the database if the value of the selected attribute is lower than the value given in the field displayed below the selected operand field.
Lower Than Equal	Records will be included into the data fetched from the database if the value of the selected attribute is lower than or equal to the value given in the field displayed below the selected operand field.
IsNotNull	Records will be included into the data fetched from the database if the value of the selected attribute does not contain a null value.
IsNull	Records will be included into the data fetched from the database if the value of the selected attribute has a null value.
Last X Days	Records from the last 'x' days will be included in the data fetched from database. Where x is the number of days specified in the field displayed below the operand.

Operands	Description
Last X Months	Records from the last 'x' months will be included in the data fetched from database. Where x is the number of months specified in the field displayed below the operand.
Last X Years	Records from the last 'x' years will be included in the data fetched from database. Where x is the number of years specified in the field displayed below the operand.
Last X Weeks	Records from the last 'x' weeks will be included in the data fetched from database. Where x is the number of weeks specified in the field displayed below the operand.
Next X Days	Records which have the creation date greater than today but less than x days from today. Where x is the number of days specified in the field displayed below the operand.
Next X Months	Records which have the creation date greater than today but less than x months from today. Where x is the number of months specified in the field displayed below the operand.
Next X Years	Records which have the creation date greater than today but less than x years from today. Where x is the number of years specified in the field displayed below the operand.

5. Select an operand from the lists.
6. If needed, in the text field below the operand field, enter the match value for the selected operand.
7. Repeat for any additional filtering criteria.

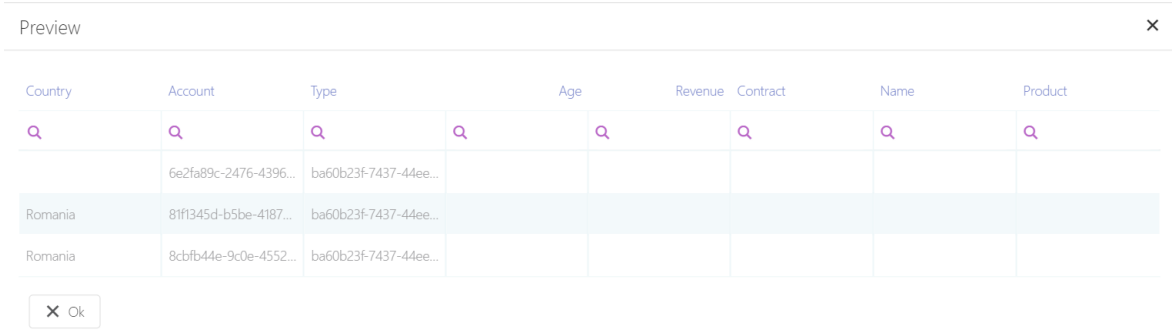
For example, the criteria below will filter entities that have either the FTOM_MKT_ Audience or the FTOS_MKT_Campaign business workflows attached.



Before saving the fetch, we recommend you to preview the fetch and validate the fetch results.

STEP 3. Validate the Fetch Results

To validate the fetch results, preview them by clicking the Preview button at the bottom of the Advanced find page. The Preview page will be displayed. The figure below is a preview example of data fetched from the Account, Contract and Product entities.



Validate the results, then click **OK** to close the page.

To filter the fetch, in the Advanced find page, click **OK**.

Data Views

A view is similar to a table from the database. It displays specific data in columns and rows as referenced in the query that defines the view. The columns and rows are dynamically generated when the view is produced.

The view query can fetch entity data coming from different tables within the database. For details on how to fetch entity data, see Fetch Entity Data.

View basic action handlers

The default view comes with the basic action handlers enabled.

The basic action handlers for views are:

- Insert - allows you to add a new entity record, opens the data form defined as default for insert.
- Delete - allows you to delete the view rows marked to be deleted (applicable only if the user has specific rights). On click, a confirmation pop-up prompts you to confirm deletion.
- Export - allows you to export view data as xlsx files. Two export options are available:
  - **Export current set** - exports the view data displayed within current page.
  - **Export all data set** - exports all data listed in view.

  NOTEWhen exporting view data, the generated .xlsx file is saved into the “UploadEbs\temp” folder.
- Advanced find - allows searching and filtering data based on customized criteria. For details, see "Advanced Entity Find" on page 89.

Action results generate status messages displayed at the bottom of the page.

Creating and Designing Views

Introduction

An entity can have many views defined, but only one will be used as default without mentioning the name. Others views could be used to display data through relationships or through lookup view option.

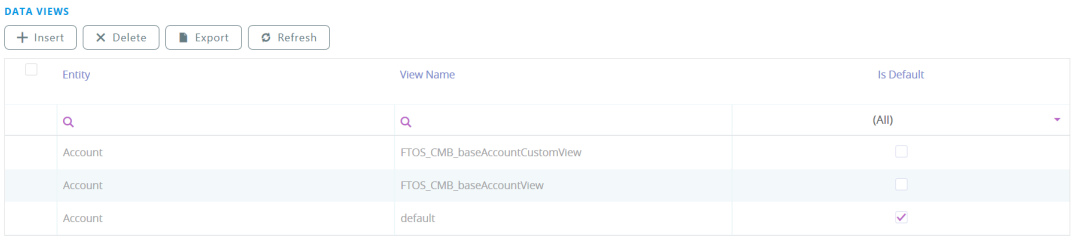
Each entity has a default auto-generated view, which displays by default only the primary attribute (defined when the entity was created), but can be customized at any time.

Adding Views

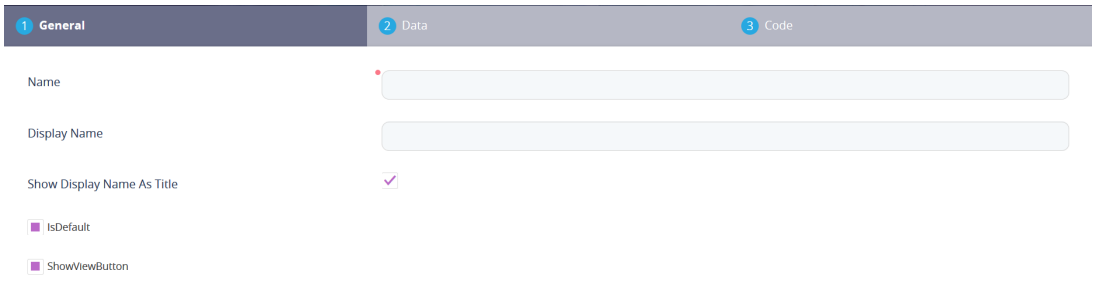
The view configuration page allows you to create a new view or edit an existing one.

To add a view to an entity, follow these steps:

1. Go to the entity edit page and scroll-down to the Data Views section and click the section header. The Data Views section expands.



2. At the top of the section, click the Insert button. The view configuration page appears. It is comprised of three sections and the General tab will be displayed by default. You can navigate through the view configuration sections by clicking the tabs.



3. "Provide View General Information" below
4. "Defining View Data" below.

You can also enable Inline Editing for view and Apply Conditional HTML Formatting.

Provide View General Information

The General tab allows you to provide a name for the new entity view and choose whether it is set as default view and also add a View button to all the view entries.

Property	Description
Name	The name of the view.
IsDefault	If selected, current view will be displayed when users access the specific entity. If you want to display on a data form a view that is not defined as default, add the view name as a token on the data form **{#RelationshipName, view: viewName#.}**. By doing so, the view specified by the token provided will be displayed on the entity data form instead of the default entity view.
ShowViewButton	If selected, adds the View column on the entity view which contains a View button. When clicked, the link redirects the users to the details page of the specific record.

In order to define the view data, you should first provide the view general information and save the view by clicking the Save and reload icon.

Defining View Data

The Data tab allows you to fetch entity data that will be displayed on the view, select the view columns and define the default sort of the view records.

NOTEYou have to save the view before fetching entity data.

To define what data is shown in the view and the sorting order of records, follow these steps:

STEP 1. Fetch entity data

Fetching data allows you to filter entity records displayed in view based on specific criteria (e.g business status, security role).

There are two ways in which you can fetch entity data:

Fetch Entity Data Using the Fetch Designer

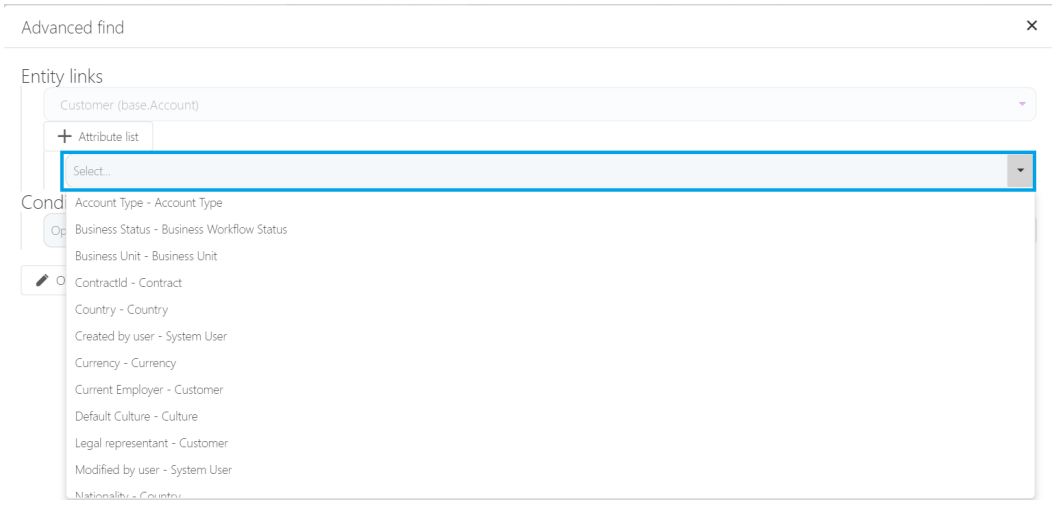
At the upper-right corner of the Data section, click the Show Fetch Designer button. The **Advanced Find** page will be displayed.

Choose Entity Links

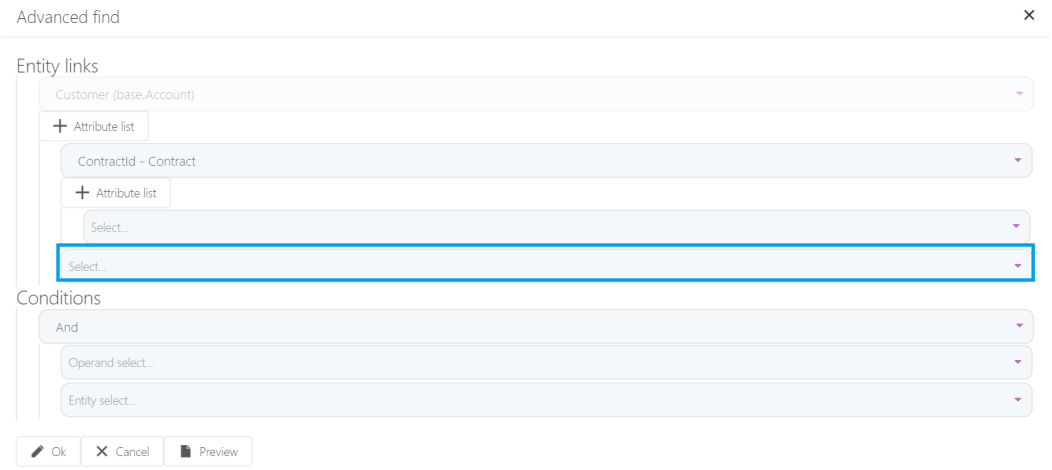
In the Advanced find page, the entity from which you triggered the Advanced search is displayed by default in the Entity Links field and it is non-editable.

Select the entities

Click in the Select field. The list of entities that are linked to the default entity will be displayed.



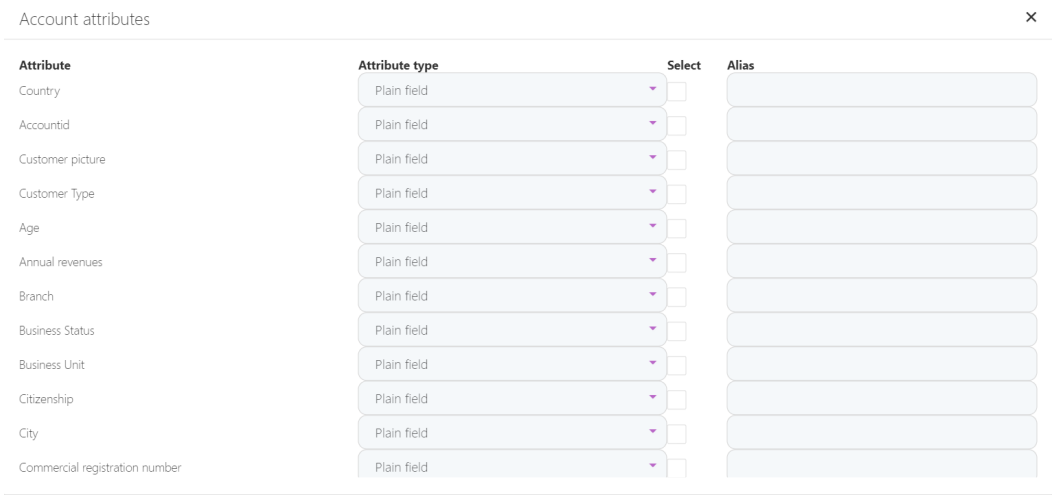
Select the desired entity from which you want to gather data. Another empty field will be displayed which allows you to select another entity in case you want to fetch data from various entities.



You can add as many entities as you need to extract data from. For example, to create profiles for tailored price offerings and communication, you need to fetch data from the following entities: Account, Product and Contract.

Select the attributes

After selecting all the entities from where you want to extract information, for each entity, select the attributes to be included in the fetch. To do so, click the **Attribute list** button displayed below each entity. The Attributes list page will be displayed. It lists all the attributes existing on that entity. The figure below shows the attributes of the Account entity:



Select the attributes you want to include in the fetch by clicking the corresponding checkbox in the Select column.

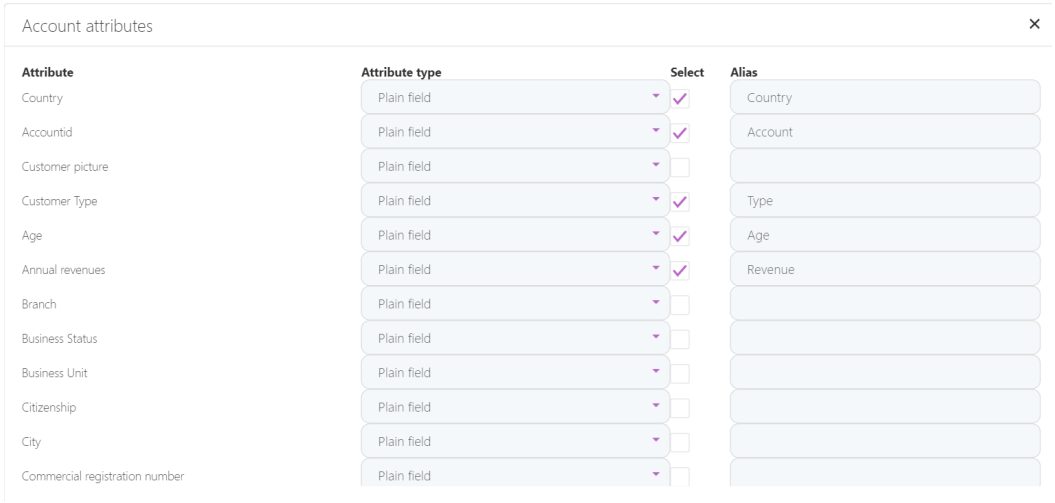
For all selected attributes, in the Alias column, type a unique alias name.

> **IMPORTANT!**
> The alias is used by the system to distinguish between various records; therefore, providing a unique alias for all selected attributes is mandatory. Do not use the same alias name for two attributes of the same or different entities. For example, you should not use the "Mobile" for an attribute existing on the entity "Account" and the "Mobile" alias for an attribute on the entity "Campaign".

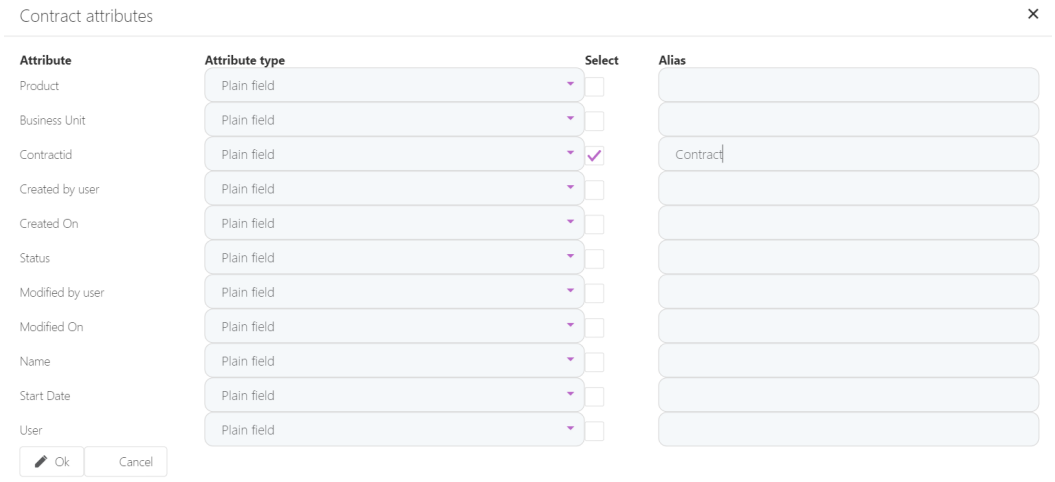
Optionally, you can select how the attribute value will be displayed, by choosing one of the values available in the Attribute type drop-down list:

- **Plain field** - displays the actual value of the attribute. It is selected by default for all attributes.
- **Sum field** - displays the sum of the attribute values.
- **Max field** - displays the maximum of the attribute values.
- **Count field** - displays the number of the attribute values.

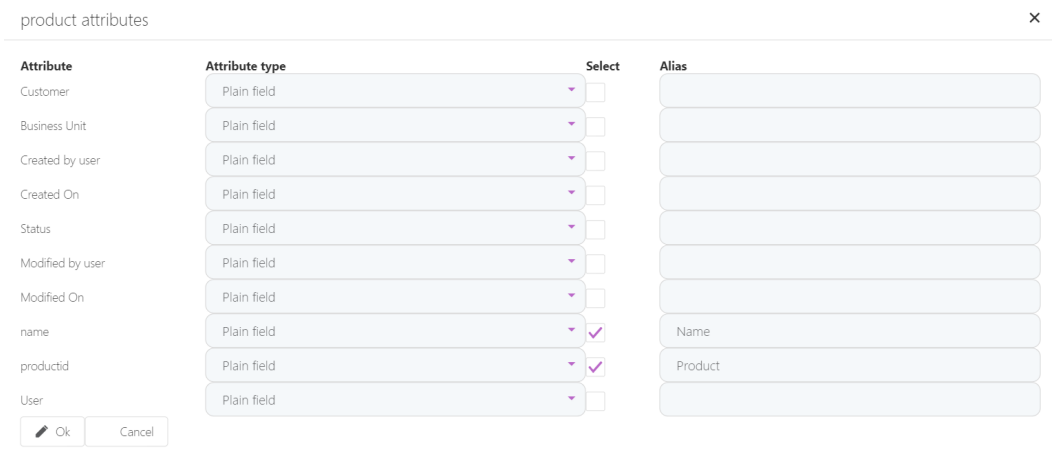
Account Attributes



Contract Attributes



Product Attributes



Scroll-down at the bottom of the attributes list and click **OK**. The page listing the entity's attributes will be closed.

Now you have to set up the fetch conditions.

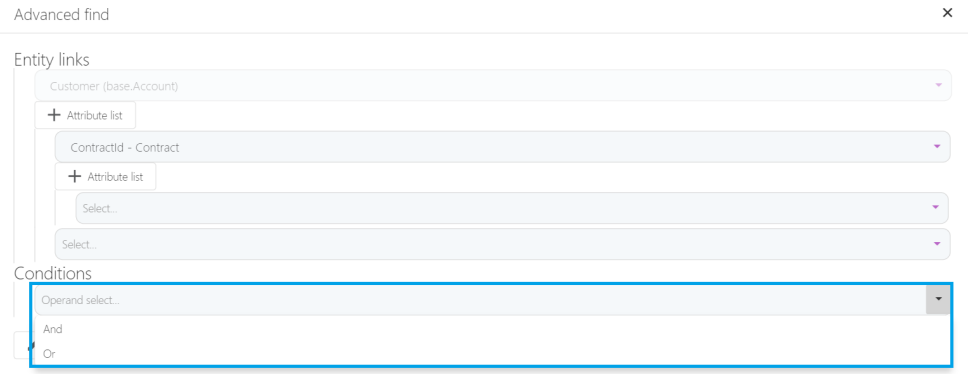
Apply Filtering Conditions

From the Conditions section you need to select the conditions which must be met in order to include a record into the data fetched from the database.

You can apply multiple conditions which need to be met separately or in combination by selecting either AND or OR as logical operator.

To set up the fetch entity data conditions, follow these steps:

1. From the drop-down field displayed in the Conditions section, select the logical operator:

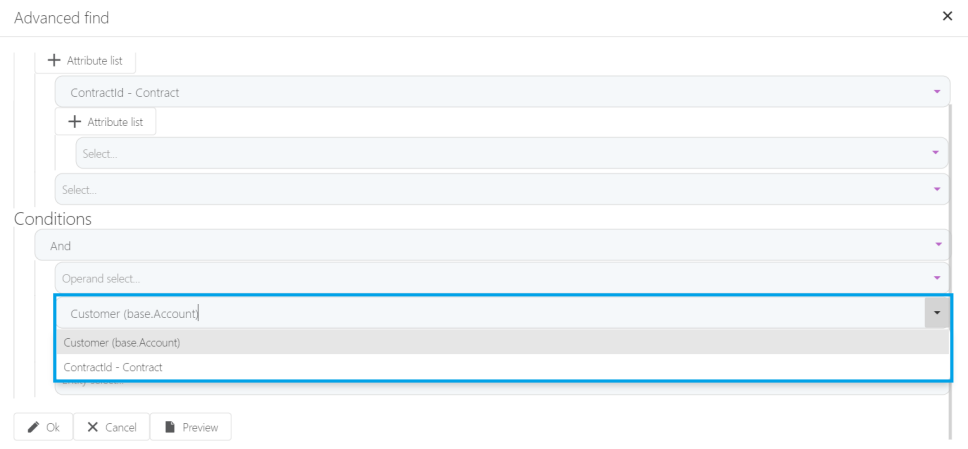


AND – to apply multiple conditions which need to be met separately.

OR – to apply multiple conditions when at least one condition needs to be met.

You can apply multiple conditions by using as many logical operators as you need. Once you select one, another logical operator field will be displayed beneath the first one, and so on.

2. Select the entity on which the condition is to be applied.



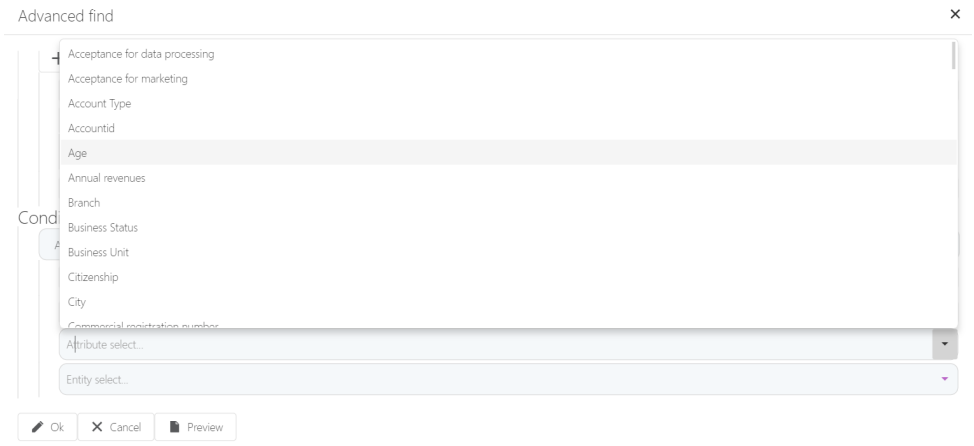
NOTE

You can only select from the list of chosen entities on which the

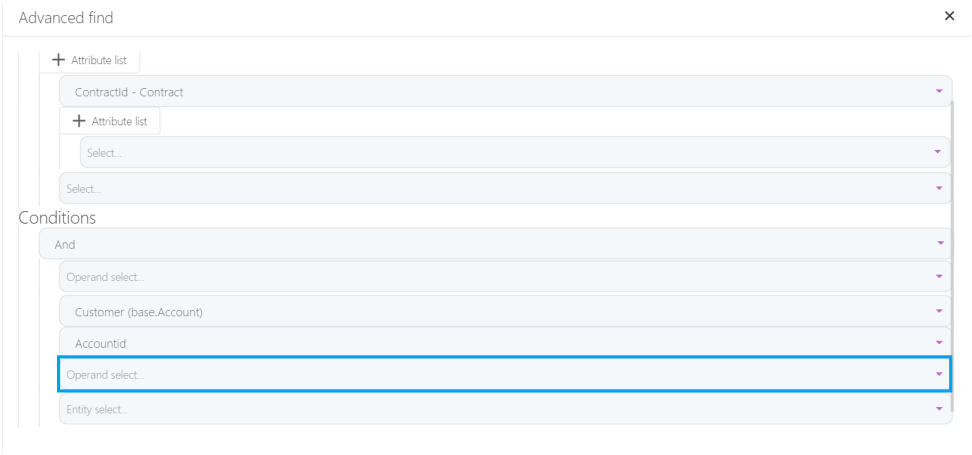
database segmentation will be done (selected in the Entity Links section). For information on how to choose entities for fetching the data, see .

After selecting an entity, the **Attribute select** field will be displayed below the selected entity.

3. Click in the **Attribute select** field. A drop-down list will be displayed, listing only the entity attributes selected for the fetch.



4. Select an attribute existing on the entity previously selected. Below the selected attribute, the Select operand field will be displayed.



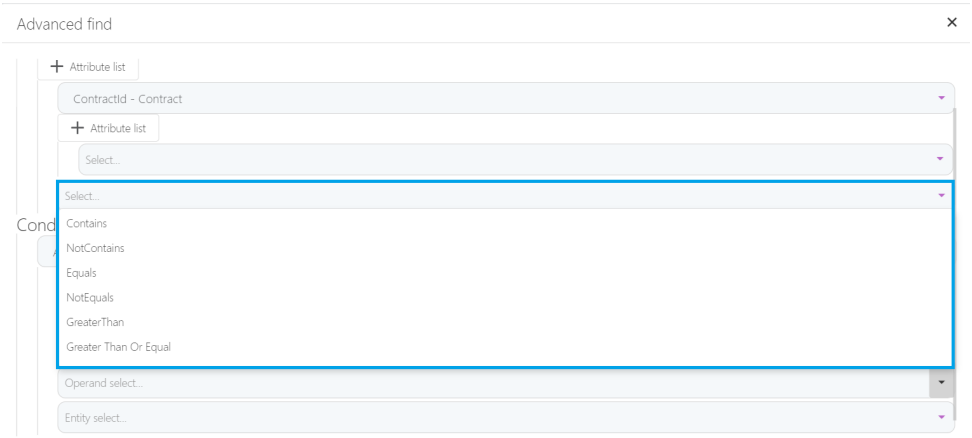
The table below describes the operands you can choose from.

Operands	Description
Contains	Records will be included into the data fetched from the database if the value of the selected attribute contains the value given in the field displayed below the selected operand field.
NotContains	Records will be included into the data fetched from the database if the value of the selected attribute does not contain the value given in the field displayed below the selected operand field.
Equals	Records will be included into the data fetched from the database if the value of the selected attribute equals the value given in the field displayed below the selected operand field.
NotEquals	Records will be included into the data fetched from the database if the value of the selected attribute is different than the value given in the field displayed below the selected operand field.

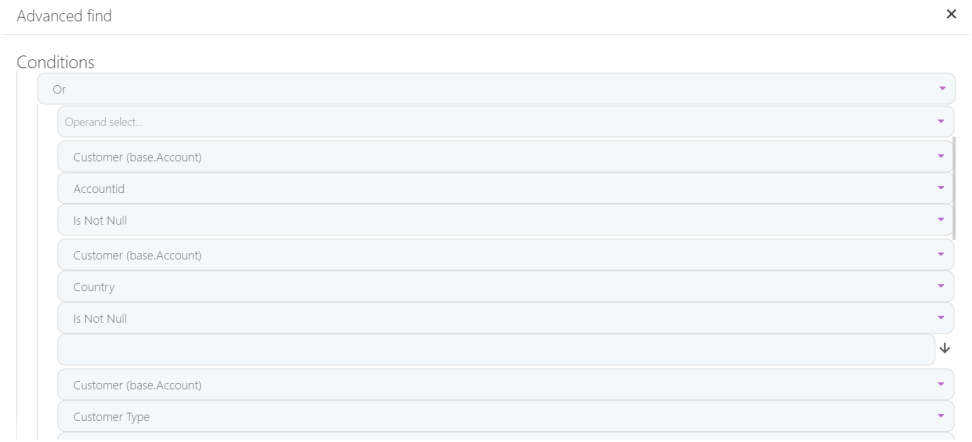
Operands	Description
Greater Than	Records will be included into the data fetched from the database if the value of the selected attribute is greater than the value given in the field displayed below the selected operand field.
Greater Than or Equal	Records will be included into the data fetched from the database if the value of the selected attribute is greater than or equal to the value given in the field displayed below the selected operand field.
Lower Than	Records will be included into the data fetched from the database if the value of the selected attribute is lower than the value given in the field displayed below the selected operand field.
Lower Than Equal	Records will be included into the data fetched from the database if the value of the selected attribute is lower than or equal to the value given in the field displayed below the selected operand field.
IsNotNull	Records will be included into the data fetched from the database if the value of the selected attribute does not contain a null value.
IsNull	Records will be included into the data fetched from the database if the value of the selected attribute has a null value.
Last X Days	Records from the last 'x' days will be included in the data fetched from database. Where x is the number of days specified in the field displayed below the operand.

Operands	Description
Last X Months	Records from the last 'x' months will be included in the data fetched from database. Where x is the number of months specified in the field displayed below the operand.
Last X Years	Records from the last 'x' years will be included in the data fetched from database. Where x is the number of years specified in the field displayed below the operand.
Last X Weeks	Records from the last 'x' weeks will be included in the data fetched from database. Where x is the number of weeks specified in the field displayed below the operand.
Next X Days	Records which have the creation date greater than today but less than x days from today. Where x is the number of days specified in the field displayed below the operand.
Next X Months	Records which have the creation date greater than today but less than x months from today. Where x is the number of months specified in the field displayed below the operand.
Next X Years	Records which have the creation date greater than today but less than x years from today. Where x is the number of years specified in the field displayed below the operand.

5. Click in the Select operand field. A drop-down list with all available operands will be displayed.



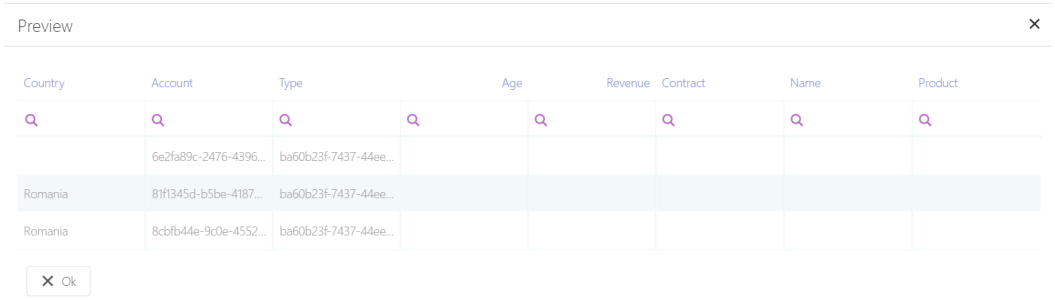
6. Select an operand from the lists.
7. If needed, in the text field below the operand field, enter the match value for the selected operand.



Before saving the fetch, we recommend you to preview the fetch and validate the fetch results.

Validate the Fetch Results

To validate the fetch results, preview them by clicking the Preview button at the bottom of the Advanced find page. The Preview page will be displayed. The figure below is a preview example of data fetched from the Account, Contract and Product entities.



Validate the results, then click **OK** to close the page.

To filter the fetch, in the Advanced find page, click **OK**.

Fetch Entity Data Using the Fetch Object Expression Field

In the Fetch Object Expression field, provide the custom fetch expression, that is a JavaScript object. It might query and return attributes from various entities.

IMPORTANT!
Make sure you always define aliases for fetched entities, otherwise the fetch will not work.

When joining several entities together using the Fetch Object Expression field, the first entity in the fetch expression should have the alias 'a'.

Below is an example on how the fetch expression should look like:

```
Joined entities alias
return {
    "entity": {
            "alias": "a",
            "name": "AccountRelEmployer",
            "attributelist": [
                        {
                            "name": "AccountRelEmployerid"
                        },
                        {
                            "name": "Details",
                        }
                    ],
            "join": [
                    {
                        "type": "left",
                        "entity": {
                        "alias": "entity1",
                        "name": "Account",
                        "attributelist": [
                                    {
                                    "name": "Name",
                                    "attributeType": 3
                                    }
                ]
        },
        "fromto": [
                {
                    "from": "ReferencedAccountId",
                    "to": "Accountid"
                }
               ]
    },
    {
        "type": "left",
        "entity": {
                "alias": "osi",
                "name": "optionsetitem",
```

```
                "attributelist": [
                            {
                                "name": "name",
                                Fintech OS - UI Designer
                                "attributeType": 3
                            }
                        ]
            },
        "fromto": [
                {
                    "from": "EmploymentTypeId",
                    "to": "optionsetitemid"
                }
            ]
        },
    ]
},
where:{}
}
```

To render the view on the related entity data form, add an empty 'where' condition:

```
Empty 'where' condition
    return {
        "entity": {
            "alias": "lookup",
            "name": "AccountRelEmployer",
            "attributelist": [
                        {
                            "name": "AccountRelEmployerid"
                        },
                        {
                            "name": "Details",
                        }
                    ],
            "join": [
                    {
                        "type": "left",
                        "entity": {
                        "alias": "entity1",
                        "name": "Account",
                        "attributelist": [
                                    {
```

```
                                        "name": "Name",
                                        "attributeType": 3
                                    },
                                ]
                        },
                "fromto": [
                        {
                            "from": "ReferencedAccountId",
                            "to": "Accountid"
                        }
                    ]
                },
                {
                    "type": "left",
                    "entity": {
                            "alias": "osi",
                            "name": "optionsetitem",
                            "attributelist": [
                                        {
                                        "name": "name",
                                        "attributeType": 3
                                        }
                                ]
                        },
                    "fromto": [
                            {
                                "from": "EmploymentTypeId",
                                "to": "optionsetitemid"
                            }
                        ]
                },
            ]
        },
where:{}
}
```

> **IMPORTANT!**
> The Fetch Designer is a visual tool that allows you to populate the Fetch Object Expression field without having to code. When you define a fetch in the Fetch Designer, any existing content in the Fetch Object Expression field will be overwritten.

STEP 2. Define the View Columns

You have two options to define the view header (that is, the columns to be displayed on the view):

- In the Data field, list the name of the fields which will define the columns displayed by the view.

  Make sure that you list the name of the fields separated by comma.

- In the Entity View Columns section, insert the desired view columns.

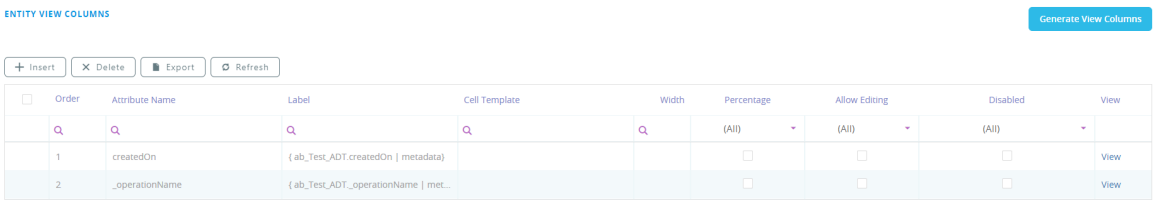
> **NOTE** The view columns added in the Entity View Columns section have a higher priority than the ones mentioned in the Data field. If both entity view columns and data columns are defined, the entity view will display only the columns added in the Entity View Columns section.

To define the view columns from the Entity View columns, click the **Insert** button at the top of the section. The Add Entity View Column page will be displayed. Provide the following view column details:

Property	Description
Entity View	The field is automatically filled in with the name of the selected view.
Width	The column width. Enter a numeric value.

Property	Description
Cell Template	A custom HTML element which defines the cell layout on the entity view column. In this field you can add text, HTML or tokens. A token can be any attribute returned by the fetch expression. Attributes are referenced as tokens by their name prefixed by the alias of the fetch inside curly brackets (e.g., {base.Name}). Cell template example: `<div onclick="window.location.hash ='/entity/MKT_Activity/edit/{a.MKT_Activityid}' " style="cursor: pointer; color: white; display: inline; float: left; border: 2px solid rgba(0, 0, 0, 0); background: #27a8e1; text-align: center; margin: 2px; padding: 2px;">POP</div>` For more information on how to use cell templates for views, see Create Views using Cell Template.
Width Is Percentage	If selected, it indicates that the value provided in the **Width** field is percentage.
Attribute Name	The name of the attribute whose values will be displayed in the column. You can overwrite the attribute when providing a cell template.
Label	The column name that will be displayed on the entity view in the user interface.
Allow Editing	If selected, the column view can be edited.

At the upper-right corner of the page, click the **Save and close** icon. Add as many view columns as you need. They will be listed in the Entity View Columns section.



The view columns are displayed from left to right in the order of their index (OrderIdx). The first column has the OrderIdx set to 1.

To change the order in which the columns are displayed on the view, in the Entity View Columns list drag and drop rows.

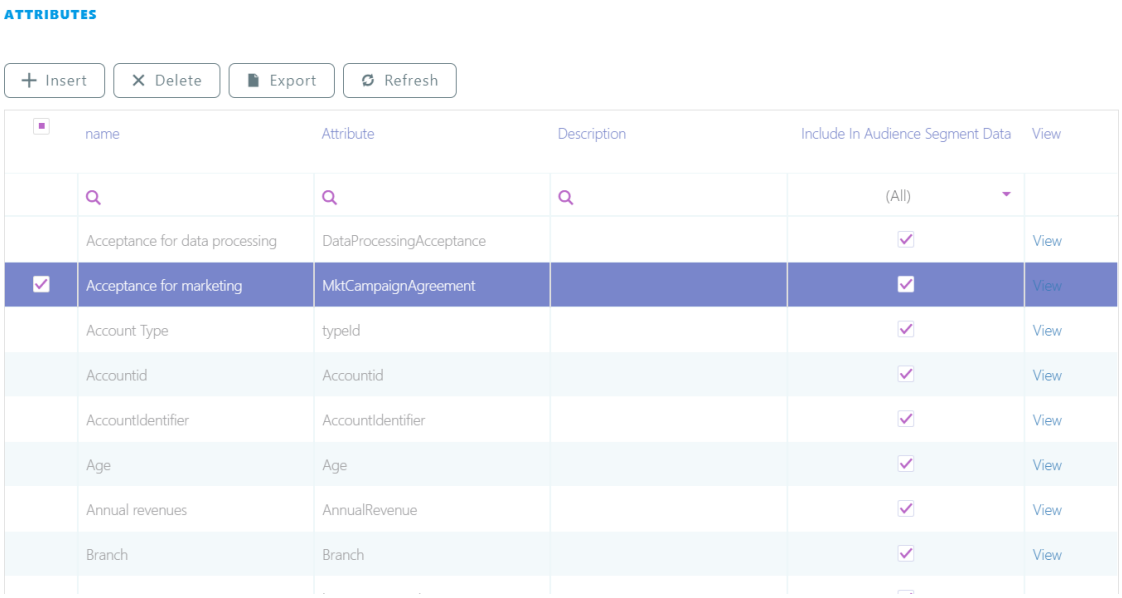
You can also auto-populate the view columns. For more information, see Generate View Columns..

STEP 3. Set the default sorting of the view records

By default the view records are sorted ascending by the entity primary attribute.

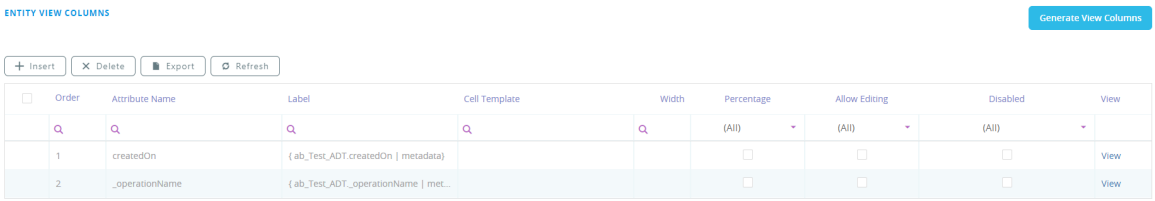
To change the default view records sorting, in the Sort Expression field, enter the attribute name by which the records will be sorted, followed by an asterisk (*) symbol and the sorting direction: asc for ascendant and desc for descendant.

The figure below shows how a view looks like in the user interface.



Generate View Columns

To auto-populate the columns of a view, in the view configuration page, Data tab, scroll-down to the Entity View Columns section, click the Generate View Columns button and in the confirmation pop-up, click Yes..



The view has no fetch data

If the view has no fetch data, the view columns will be populated for all attributes specified in the text field if the field name does not already exist in view column. The column name will be set with the attributes label in the current language.

If you want to replace the display name of an attribute with its localized value, in the **Label** column of the entity view write the following syntax: **{entityName.attributeName | metadata}**.

Clicking the Generate View Columns button will add the missing columns.

NOTEThe system checks if the column already exists by label but it will not delete any column, You will have to manually remove columns.

The view has fetch data

If the view has fetch data, the view columns will be populated for all returned fields specified in the fetch (with alias to) field name that does not already exist as view column. The column name will be set with the attributes label in the current language.

NOTEIf in the view's fetch there is an alias for the attributeName, you should use **aliasAttribute**; otherwise use **entityAlias.attributeName**.

When exporting the view using deployment packages, the label is localized so the label pattern: **{entityName.attributeName | metadata}** will be replaced with the **displayName** of the attribute.

Inline Editing for View Records

Inline editing for views is available which allows you to quickly make changes to view records directly in the view without opening edit forms.

You can set inline editing for views from the view configuration page, Code tab.

Click the Display Options tab and define the view display options using code similar to the following one:

```
{
    "scrollHorizontal":true,
    "allowEdit":true,
    "editMode":"cell",
    "showColumnHeaders":true,
    "showFilterRow":false,
    "pageSize":10
}
```

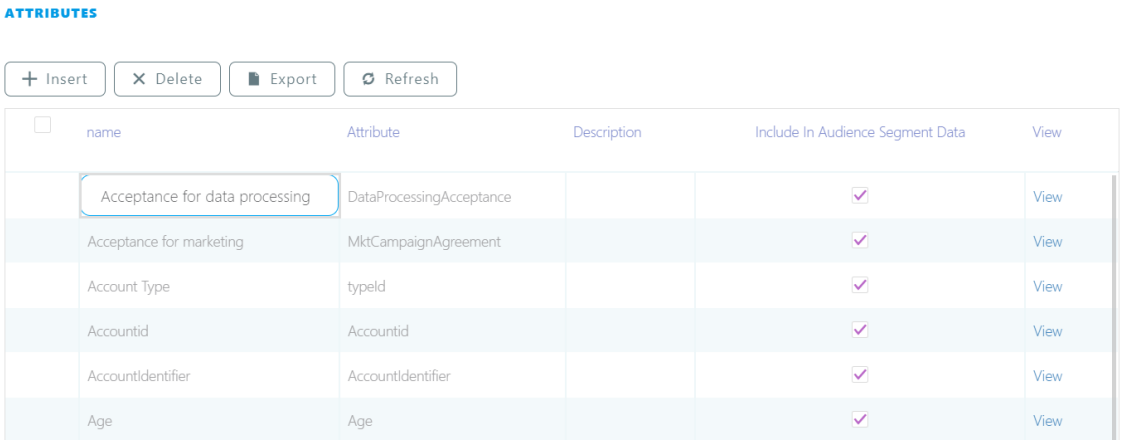
The table below describes the available view display options.

Display Option	Description
Display Option	Description
scrollHorizontal	If you have many columns in the view, set it to **true**.
allowEdit	To allow inline editing on the view, set it to **true**.
editMode	Sets the mode of the inline editing on the view. The following values are possible. **cell** - allows you to edit view records cell by cell. **row** - allows you to edit a view record by editing the cells in a row then saving the view record changes. **batch** - allows you to edit several view records and then save the changes in batch. **data form** - allows you to edit a view record similar to a data form. When clicking on a view record, the selected record information will be displayed similar to a data form.
showColumnHeaders	If you want the view to show the columns header, set it to **true**.
showFilterRow	If you want to show the filter row which allows you to filter the view records, set it to **true**.

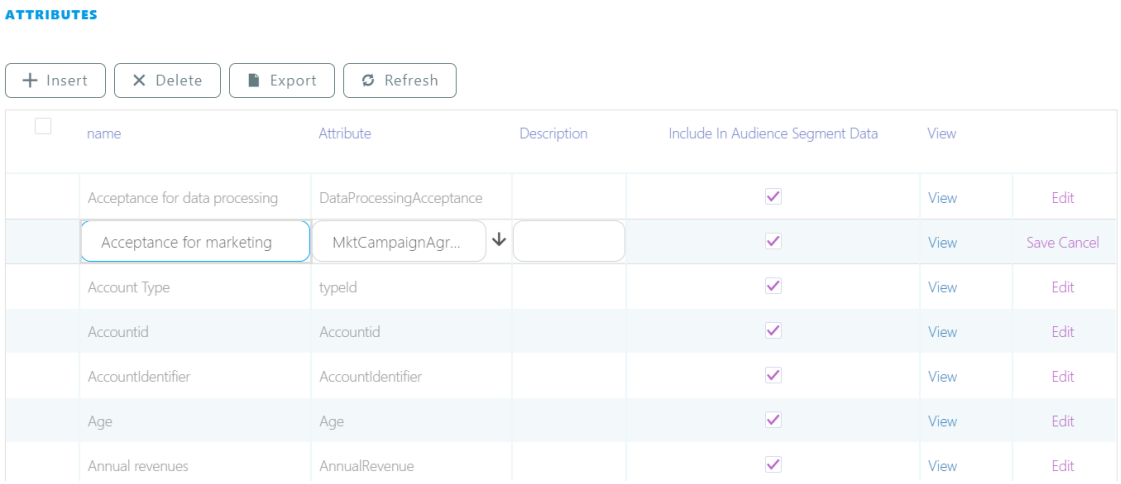
At the upper-right corner of the page, click the Save and close icon. The view changes are saved.

The following are some examples of how inline editing looks like in the user interface based on the editMode.

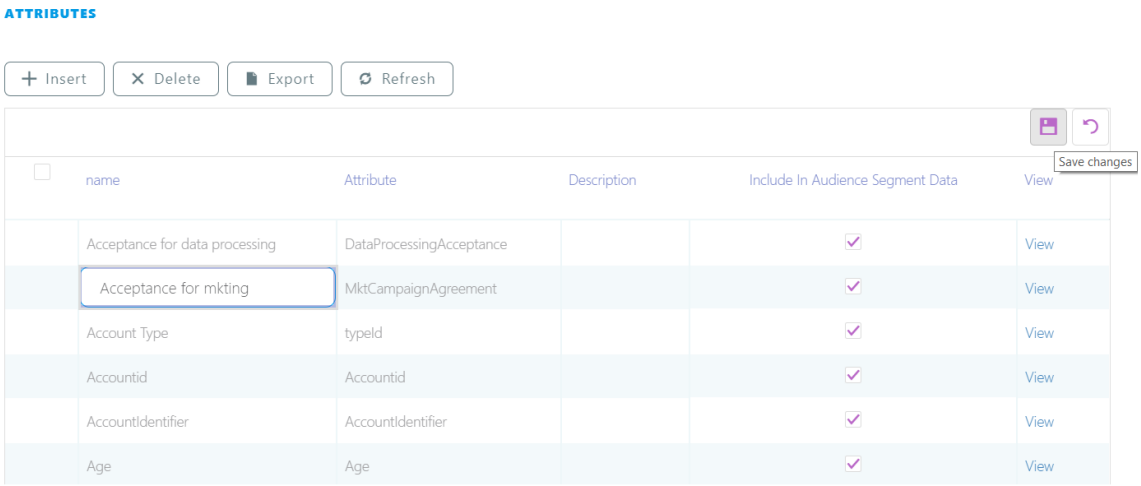
Cell edit mode



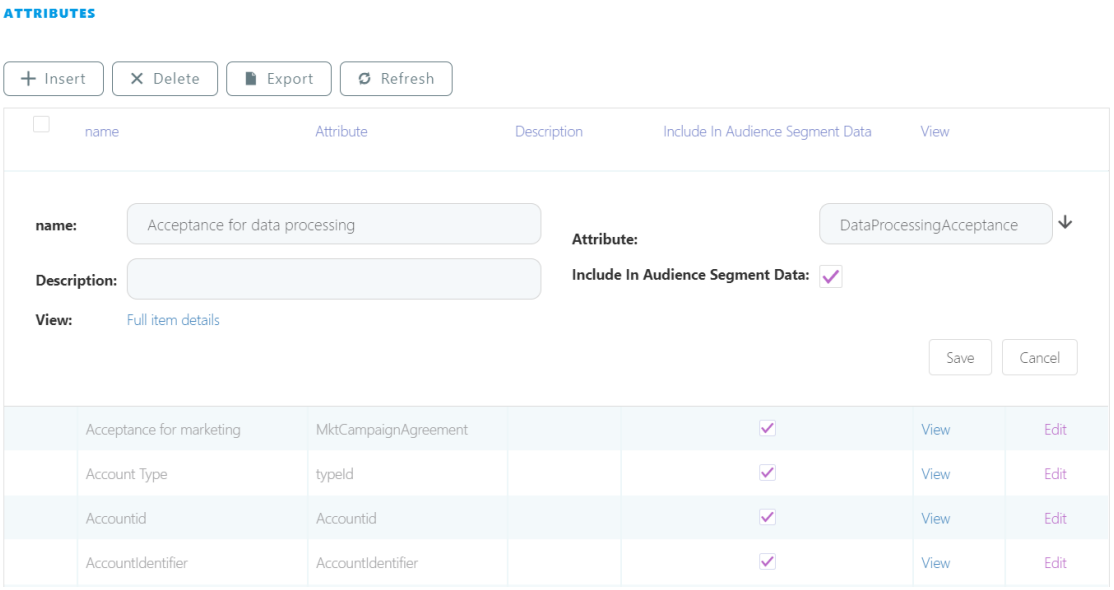
Row edit mode



Batch edit mode



Form edit mode



Create Views using Cell Template

You can create views using cell templates from the view configuration page (Edit Business Entity > Data Views >click insert to create a new one or double-click on an existing view).

Cell template is a text editor element or a collection of elements. Using cell templates, you can create views with default or custom fetch.

Creating views default fetch

Reference the attributes by using the fetch alias or by their name.

Cell Template

```
return {
    "entity": {
        "alias": "base",
        "name": "Account",
        "attributelist": [
            {
                "name": "Accountid"
            },
            {
                "name": "typeId"
            },
            {
```

```
            "name": "Email"
            },
            {
                "name": "EmploymentStatusId"
            },
            {
                "name": "MAI"
            },
            {
                "name": "MobilePhone"
            },
            {
                "name": "Name"
            },
            {
                "name": "PIN"
            }
            ],
        "join": [
                {
                "type": "inner",
                "entity": {
                    "alias": "BWstatus",
                    "name": "BWstatus",
                    "attributelist": [{
                        "name": "label",
                        "alias": "label"
                    }],
                },
            "fromto": [
                    {
                    "from": "businessStatusId",
                    "to": "BWstatusid"
                    }
                ]
            }
        ]
    }
}
```

Starting with FintechOS you can also fetch view data using the Code Editor.

Creating views with custom fetch

To create views with custom fetch, reference the attributes by using the custom alias:

```
//Fetch object expression text editor
return {
        "entity": {
            "alias": "base",
            "name": "Account",
            "attributelist": [
                {name:"Accountid"},
                {name:"Name"},
                {name:"UniqueID"},
                {name:"PIN"},
                {name:"FiscalRegistrationNo"},
                {name:"CommercialRegistration"},
                {name:"FirstName"},
                {name:"LastName"},
                {name:"MobilePhone"}
            ],
        "join": [{
            "type": "left",
            "entity": {
            "alias": "accType",
            "name": "FTOS_CMB_AccountType",
            "attributelist": [{name: "FTOS_CMB_AccountTypeid"},
{name: "name", alias: "TypeName"}]
        },
        "fromto": [{
            "from": "typeId",
            "to": "FTOS_CMB_AccountTypeid"
            }]
        }]
        }
};
```

On the view, in the Code tab, After Generate JS section, provide the code to apply the desired logic.

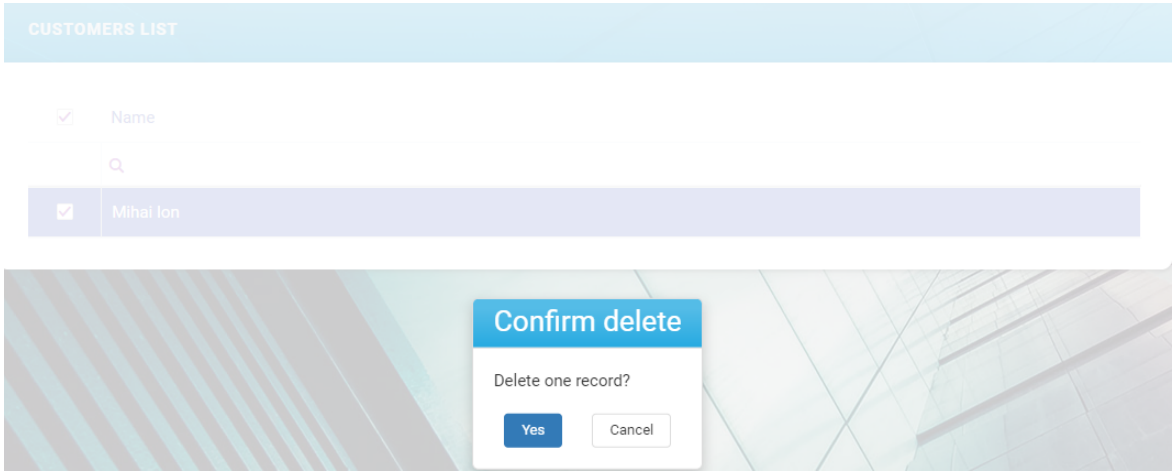
```
var view = {editFormName:"FTOS_CMB_BaseAccount",
insertFormName:"FTOS_CMB_BaseAccount"};
var gridId = 'ebsContainerContent';
var dataGrid = $('#' + gridId).dxDataGrid('instance');
     dataGrid.option('onRowClick', function (h) {
        console.log("Custom click");
var component = h.component,
        prevClickTime = component.lastClickTime;
        component.lastClickTime = new Date();
if ((prevClickTime && (component.lastClickTime - prevClickTime <
300)) || ebsIsOnMobile) {
```

```
        var nameAttr = dataGrid.option("nameAttribute");
        var key = h.key;
        var entityName = dataGrid.option("entityName");
        window.location.href = ebs.getEditNavigationUrl(entityName,
key, null, view.editFormName);
        }
});
var formData = ebs.getFormData();
if(currentPageActionHandlers.find(function(x){ return x.text ==
"Insert"; }))
        {
        currentPageActionHandlers.find(function(x){ return x.text
== "Insert"; }).handler = function(){
        ebs.generateInsert("ebsContainer", "Account", null, null,
"FTOS_CMB_BaseAccount");
        };
}
```

Customizing Delete Confirmation

You can customize the delete confirmation dialog for view records.

The figure below provides the default delete confirmation dialog:



To customize the delete confirmation dialog displayed when deleting records from a specific entity view:

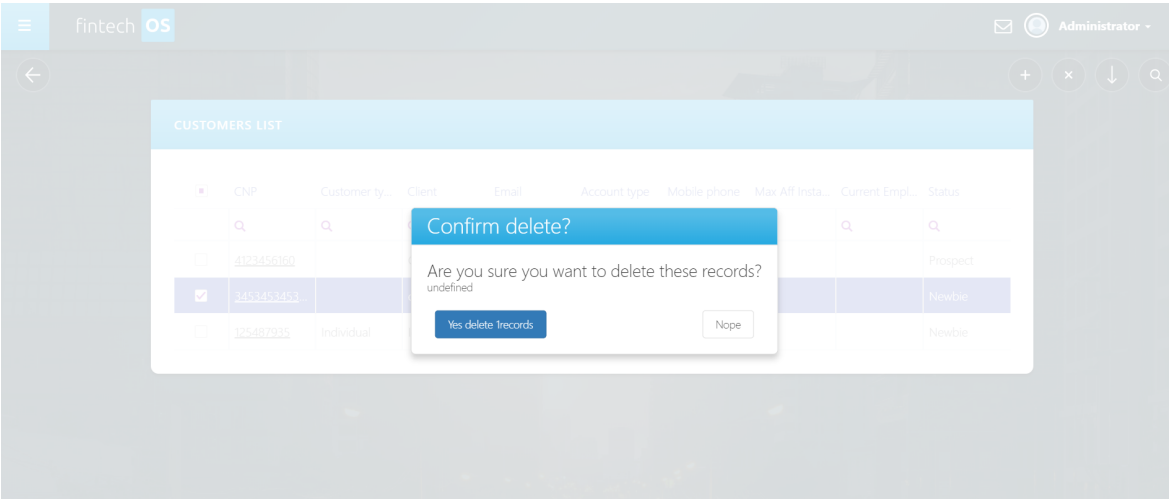
1. Go to the entity view configuration page, **Code** tab.
2. Click the **Display Options** tab

3. Customize the delete confirmation dialog by updating the code provided by default.

```
ebs.createViewDisplayOptionsObject({
   deleteConfirmation: {
        yesButtonLabel: "Yes, sure",
        noButtonLabel: "Nope",
        getDeleteConfirmationOptions: function (selectedRowsData,
nameAttribute) {
            //console.log(selectedRowsData, nameAttribute);
            var msg = "<h4>Are you sure you want to delete these
records?</h4>";
            var noOfRows = selectedRowsData.length;
            var cancel = false;
            for (var i = 0; i < noOfRows; i++) {
                msg += "<div>" + selectedRowsData[i][nameAttribute]
+ "</div>";
                if(selectedRowsData[i]["a_status"] == "pending")
                cancel = true;
            }

            if(cancel){
                ebs.showMessage("You cannot delete pending
records", "info");
            }
            return {
                yesButtonLabel: "Yes delete " + noOfRows +
"records",
                 message: msg,
                cancelDelete: cancel
            }
        }
    }
});
```

The figure below shows how the delete confirmation dialog customized above will look like:



The table below describes the properties of DeleteConfirmation.

Property	Type	Required	Default Value
title	string	no	$resources.DeleteRecords_ Confirmation_Title
message	string	no	$resources.DeleteRecords_ Confirmation_Message | DeleteRecords_Confirmation_ Message_Singular
yesButtonLabel	string	no	$resources.DeleteRecords_ Confirmation_Yes_Label
noButtonLabel	string	no	$resources.DeleteRecords_ Confirmation_No_Label
cancelDelete	boolean	no	false
silentDelete	boolean	no	false

Setting the **cancelDelete** property to **true** will prevent the selected record(s) deletion and no confirmation will be displayed.

Setting the **silentDelete** property to **true** will delete the selected record without confirmation.

If both **cancelDelete** and **silentDelete** are set to **true**, the selected record(s) will not be deleted and no confirmation will be displayed.

Show Loading Panel

To show the loading panel on views (which have inline editing enabled) when editing view records which trigger the execution of a workflow, go to the entity view configuration page, Code tab, click the Display Options tab and set the

showLoadingPanel option to **true**: `"showLoadingPanel":true`

You have an entity FTOS_CASE_Case and a child entity, CaseXSelectedRisk2. The child entity has a view, FTOS_CASE_Case, which has the cell edit mode enabled. The view has records which when updated trigger the execution of a workflow on the child entity, CaseXSelectedRisk2

To show loading when changing the value of such records, on the child entity, CaseXSelectedRisk2, in the list of views, double-click the FTOS_CASE_Case view. In the entity view configuration page, click the **Code** tab, click the **Display Options** tab and set the **showLoadingPanel** option to true: `"showLoadingPanel":true`

```
{
  "allowEdit" : true,
  "editMode" : "cell",
  "showLoadingPanel" : true
}
```

To hide the loading panel, either set the **showLoadingPanel** option to **false** or remove the option from the Display Options tab.

Data Forms

Forms are user interface elements that allow you to interact with individual entity records. You can access the entity forms from the Data Forms section of the entity.

You can create up to 2 forms for the same entity, depending the different contexts in which users interact with the entity. Some forms may allow users to add records to the entity, some only to edit existing records, some may be read-only, some may have multiple pages and advanced controls and validations, etc.

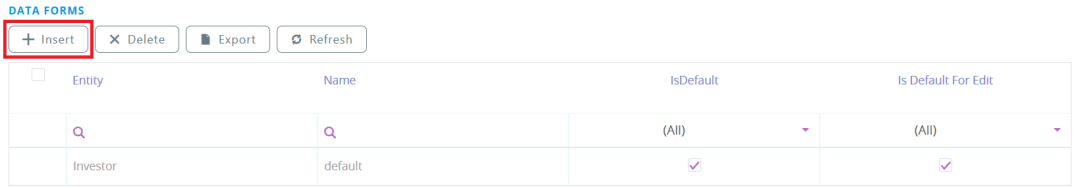
When you create an entity, a default form is created automatically which allows you to do basic record inserts and edits. The default form will be used to insert or edit records when an entity is exposed in the Digital Experience Portal or other front-ends without designating a custom form for inserts or edits. The automatically created

default form uses the Auto Generate Template option (see below), but can be customized at any time. An entity can have only one default form for record inserts and one default form for record edits at a time.

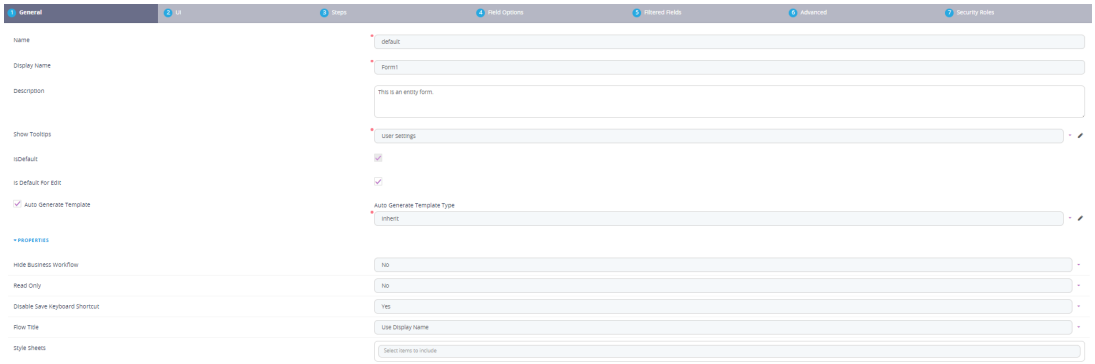
System entities do not have default forms created automatically.

Creating an Entity Form

1. Expand the Data Forms section of an entity and click **Insert**.



2. Fill in the form's general information.



   - Name - Name used to identify the entity's form. If the entity has another form defined (each entity can have up to two forms) this name has to be different from the other form's name.
   - Description - Optional description of the form's function to simplify development.
   - Show Tooltips - Select if form fields display tooltips on mouse over. The default is to inherit the user setting.

- Is Default - Check to make this the default form for inserting and editing entity records when displaying entity views in the user interface. You can have only one default form for an entity at a time.
- Is Default for Edit - Check to make this the default form for editing entity records when displaying entity views in the user interface. You can have only one default edit form for an entity at a time.
- Auto Generate Template - Check to automatically generate the form fields based on the entity's attributes using a 1, 2, 3, or 4 column layout. The template will include all entity attributes, except the primary key and system attributes.
- Hide Business Workflow - Hides the entity record's state and state transition options in the end-user interface. For more information about business workflows, see the Business Workflows Processor documentation.
- Read Only - Prevents end-users from making any changes to the displayed form fields.
- Disable Save Keyboard Shortcut - Prevents end-users from saving and reloading the form by pressing the Ctrl+S keyboard shortcut.

3. Click the **Save and Close** ( ) button.

Editing an Entity Form

To edit an entity form, expand the Data Forms section of an entity and double click the form. The form's settings are displayed.

General Settings

- Name - Name used to identify the entity's form. If the entity has another form defined (each entity can have up to two forms) this name has to be different from the other form's name.
- Description - Optional description of the form's function to simplify development.

- Show Tooltips - Select if form fields display tooltips on mouse over. The default is to inherit the user setting.
- Is Default - Check to make this the default form for inserting and editing entity records when displaying entity views in the user interface. You can have only one default form for an entity at a time.
- Is Default for Edit - Check to make this the default form for editing entity records when displaying entity views in the user interface. You can have only one default edit form for an entity at a time.
- Auto Generate Template - Check to automatically generate the form fields based on the entity's attributes using a 1, 2, 3, or 4 column layout.
- Hide Business Workflow - Hides the entity record's state and state transition options in the end-user interface. For more information about business workflows, see the Business Workflows Processor documentation.
- Read Only - Prevents end-users from making any changes to the displayed form fields.
- Disable Save Keyboard Shortcut - Prevents end-users from saving and reloading the form by pressing the Ctrl+S keyboard shortcut.

UI

Use the UI designer to define the form's layout. It is possible to add entity extension child collection support. For details on how to use the UI designer, see "UI Designer" on page 297.

Steps

Use the Steps section to set up forms with multiple pages. For details on how to work with steps, see "Adding and Configuring Steps" on page 216.

Field Options

The Field Options section allows you to customize form fields based on how users have filled out other fields in the form: show field values, show or hide specific fields, etc. For details on how to work with field options, see "Configuring Field Options" on

page 234.

Filtered Fields

A filtered field restricts the values available in a form field based on entries in other fields. For details, see "Defining Filtered Fields" on page 249.

Advanced

The Advanced section allows you to write custom client-side JavaScript code that is executed before or after the form is rendered. For details on the embedded JavaScript extensions , see the Client SDK Reference Guide.

Security Roles

Use the Security Roles section to select the security roles required for users that can access the form. For details on how to define security roles, see "Security Roles" on page 621.

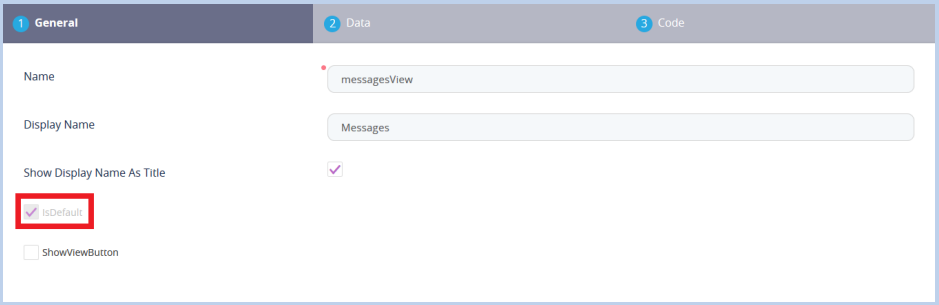
Transient Data Entities

Transient data entities allow your "Form Driven Flows" on page 203 to temporarily store and display data that has been loaded from or is going to be saved to an external data source. Thus, legacy systems benefit the most from the use of transient data and the data and metadata are not lost. The transient data entity's attributes are included in the flow's data model through data extensions (see Extend the Data Model for details) and their values are preserved only while the flow is open. Transient data entities use automation scripts for load/save that are triggered when such a flow is displayed or when it is saved to facilitate data transfers from/to external sources.

Create transient data entities

1. Open Innovation Studio in developer mode.
2. From the **Main Menu** (☰), select **Data Model Explorer**.
3. Click the **Insert** button (+) at the top right corner of the page.
4. In the **Entity Type** field, select Transient Data.

5. Enter a **Name** for the transient entity. This is a unique name used to identify the entity internally by the system.
6. Enter a **Display Name** for the transient entity. This is the name that will be displayed in the end-user interface.
7. Click **Save and reload** ( ) at the top right corner of the page.
8. In the **Data Model** section, add the entity's data attributes (for details, see "Attributes" on page 54).
9. If you want the transient data entity to appear if form driven flows as a grid of multiple records (not a single record instance), use the **Data View** section to define a view that includes the attributes you wish to display. For details see "Data Views" on page 96.

> **NOTE**
>
> Make sure that the view is configured as the default view of your transient data entity, by checking the **IsDefault** checkbox.
>
> 
>

10. Click **Save and reload** ( ) at the top right corner of the page.

Define the automation script for load

The automation script for load is triggered when a form driven flow with a data extension linked to the transient data entity is displayed. This is useful, for instance, if you use the transient data entity to fetch data from an external source and display it in the form driven flow.

To add an automation script for load:

1. Open the transient data entity in the editor.
2. Use the **Automation Script for Load** field to create the automation script.
    - The automation script must be on-demand (see "Creating On-demand Server Automation Scripts" on page 496 for details).
    - The output structure must be set to your transient data entity (see "Customizing the Output Structure" on page 500 for details). If you will use the transient data entity to show a grid of records (not a single record instance) when displayed in a form driven flow, make sure that the **Output Parameter Type** is set as Collection. Otherwise, choose Single Instance.

Examples

List the user accounts from an external FintechOS instance as a single record

In this automation script for load:

- We use the **ftosOpenAPI** Web API client library that stores the API specifications for another FintechOS instance to make API calls to that instance. For more details about Web API client libraries, see "Using Web API Client Libraries" on page 487.
- We read the incoming **webserver**, **username**, and **password** input parameters that we have set up in the automation script. For details, see "Customizing Input Parameters" on page 498.
- We authenticate to the above web server with the provided credentials and we run a query on the **systemuser** entity.
- We extract each **userName** from the **Records** object in the result set in the **users** variable, which we then save to the **importedData** attribute we have set up for the transient data entity.

```
context.result = createResult();

let client = importWebApiClient('ftosOpenApi',
context.parameters.webserver);
try {
    let authToken = client.authorize.getToken("client_id",
context.parameters.username, context.parameters.password,
"", "");
    if (authToken && authToken.access_token) {
        let data = client.openApi.query({
            apiInfo: {
                token :  authToken.access_token
            },
            request: {
                entity: {
                    name: "systemuser",
                    alias: "t"
                },
                distinct: false
            }
        });
        let users = [];
        data.Records.forEach(myFunction);
        function myFunction(value){
            users.push(value.t_userName)
        };
        context.result.importedData = toJson(users)
    }
    else
        throw new Error('Invalid authentication!');
}
catch(err) {
    context.result.importedData = err
}
```

List the visitors and messages saved in a guest book database on an external FintechOS instance as a collection of records

In this automation script for load:

- We use the **ftosOpenAPI** Web API client library that stores the API specifications for the **https://anotherFTOS/Pulsarb71i02Portal** FintechOS instance to make API calls to that instance. For more details about Web API client libraries, see "Using Web API Client Libraries" on page 487.
- We authenticate to the above web server with the **host** and **1234567** credentials and we run a query on the **guestBook** entity.
- We store each **visitor** and **message** we find in the guest book in an **entry** object, and we push each entry object in the **result** collection.
- We set the result collection as the script's output, (we have configured the script to have a collection output parameter type).

```
let result = [];
let entry = {};

let client = importWebApiClient('ftosOpenApi',
'https://anotherFTOS/Pulsarb71i02Portal');
try {
    let authToken = client.authorize.getToken("client_id",
'host', '1234567', "", "");
    if (authToken && authToken.access_token) {
        let data = client.openApi.query({
            apiInfo: {
                token :  authToken.access_token
            },
            request: {
                entity: {
                    name: "guestBook",
                    alias: "t"
                },
                distinct: false
            }
        });
        data.Records.forEach(myFunction);
        function myFunction(value){
            entry = {};
            entry.visitor = value.t_visitor;
            entry.message = value.t_message;
            result.push(entry)
```

```
        };

        setResult(result)
    }
    else
        throw new Error('Invalid authentication!');
}
catch(err) {
    context.result.importedData = err
}
```

Define the automation script for save

The automation script for save is triggered when a form driven flow with a data extension linked to the transient data entity is saved. This is useful, for instance, if you use the transient data entity to save data displayed it in the form driven flow to an external destination.

To add an automation script for save:

1. Open the transient data entity in the editor.
2. Use the **Automation Script for Save** field to create the automation script.

Example: Update an entry in the Swagger sample pet store

In this automation script for save:

- We use the **petstore** Web API client library that stores the API specifications for the **https://petstore.swagger.io/v2** web service. For more details about Web API client libraries, see "Using Web API Client Libraries" on page 487.
- We call the **updatePet** endpoint to update the **id**, **name**, and **tags** attributes of an entry with the values stored in the respective **petID**, **petName**, and **petTags** input parameters that we have set up in the automation script. For details, see "Customizing Input Parameters" on page 498.
- We leave the **photoUrls** attribute empty.

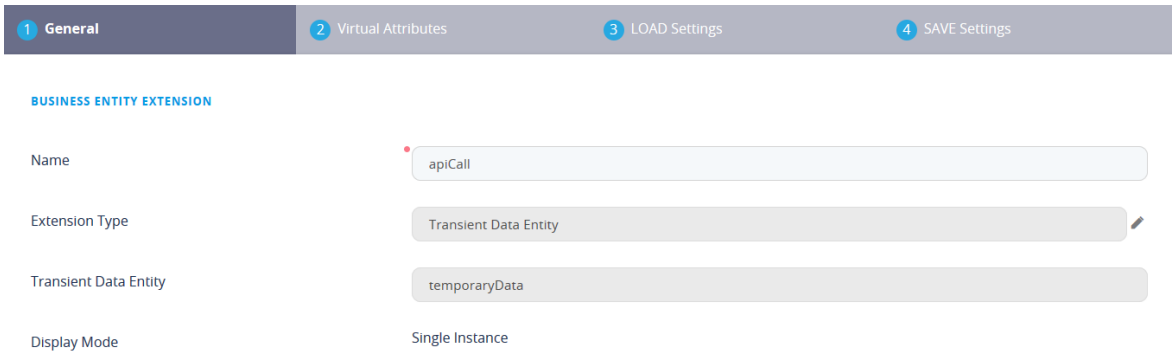
```
var client = importWebApiClient("petstore",
"https://petstore.swagger.io/v2");
client.updatePet({
    id: context.parameters.petId,
    name: context.parameters.petName,
    tags: context.parameters.petTags,
    photoUrls: []
})
```

Extend platform data entities with transient data entities

To include a transient data entity in a form driven flow, you must extend the flow's source entity data model with the transient data entity. This will allow you to interact with the transient data entity attributes via fields in your form driven flow. For more information about data model extensions, see Extend the Data Model.

Step 1. Create a transient data entity extension

1. Open a platform data entity in the editor.
2. Expand the **Extended Model** section.
3. Click **Insert**.
4. Enter a **Name** for your entity extension.
5. In the **Extension Type** field, select Transient Data Entity.
6. In the **Transient Data Entity** field, select a transient data entity you created earlier (see "Create transient data entities" on page 130 for details).
7. Click **Save and reload** ( ) at the top right corner of the page.

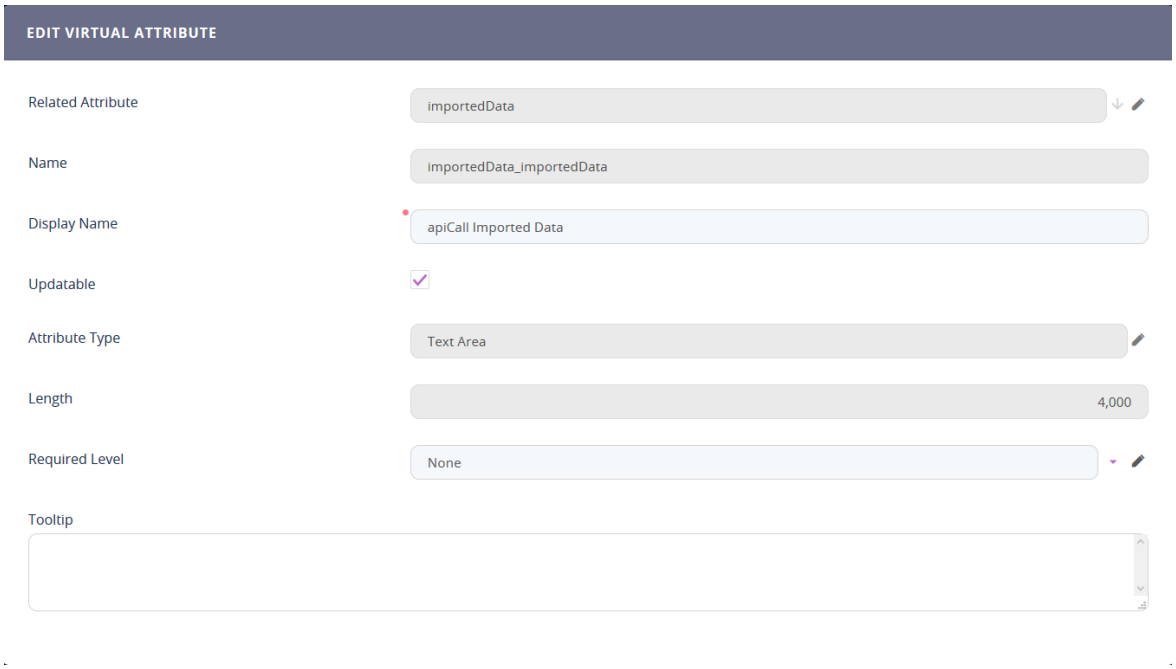


Step 2. Add virtual attributes (only for transient data entities with single instance outputs)

1. Open the entity extension for your transient data entity.
2. Select the **Virtual Attributes** tab.
3. Click **Insert** and fill in the virtual attribute's settings.
    - **Related Attribute** - Select the transient entity attribute linked to your virtual attribute.
    - **Name** - Enter a name for your virtual attribute. This is a unique name used to identify the virtual attribute internally by the system.
    - **Display Name** - This is the field name that will be displayed in the end-user interface.
    - **Updatable** - Select to have the data extension value updated automatically.
    - **Attribute Type** - Will be updated automatically to match the Related Attribute.
    - **Length** - Will be updated automatically to match the Related Attribute.
    - **Required Level** - Select if the attribute is optional, required, or recommended to be filled.

- **Tooltip** - If tooltips are set to be shown on forms and digital journeys and you want to have a tooltip explaining this data extension in the user interface, provide the desired text in the Tooltip text area field.

4. Click **Save and Reload** ( ) at the top right corner of the page.



Step 3. Bind entity attributes to the automation script for load input parameters

If the transient data entity's automation script for load includes input parameters (see "Define the automation script for load" on page 131 for details), you must bind those parameters to attributes in your extended entity. When a form driven flow is displayed, the input parameters will be populated based on the matching entity attributes.

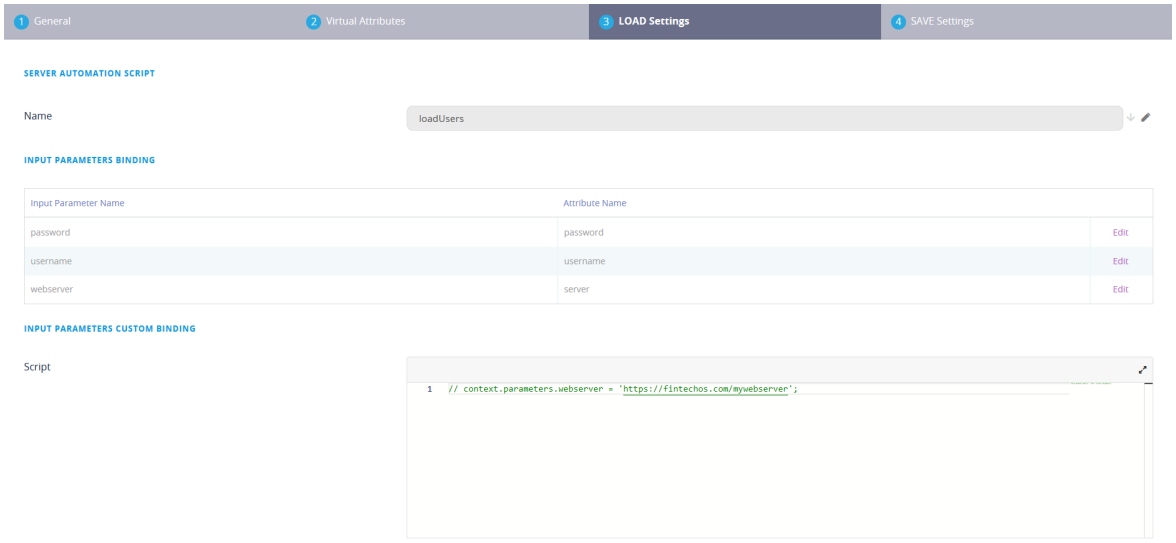
To bind input parameters to entity attributes:

1. Open the entity extension for your transient data entity.
2. Select the **LOAD Settings** tab.
3. The server automation script **Name** will be populated automatically with the transient data entity's automation script for load.

4. In the **Input Parameters Binding** section, match each input parameter to the entity attribute used to populate it. Click the Edit/Save links at the right end of each row to set the attribute names.
5. Optionally, you can manually enter a script in the **Input Parameters Custom Binding** to define custom input values for your input parameters.

> ⚠ **IMPORTANT!**
>
> **Input Parameter Custom Binding** settings override settings in the **Input Parameters Binding** section

6. Click **Save and Reload** (💾) at the top right corner of the page.

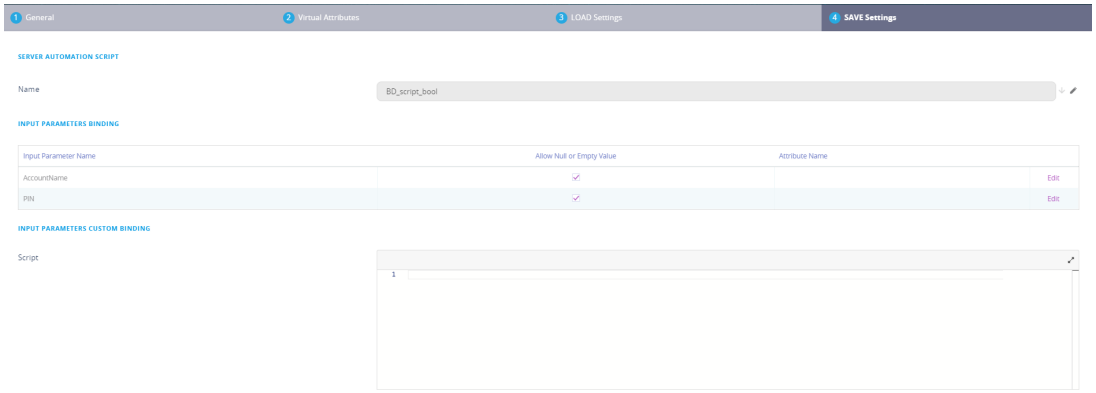


Step 4. Bind entity attributes to the automation script for save input parameters

If the transient data entity's automation script for save includes input parameters (see "Define the automation script for save" on page 135 for details), you must bind those parameters to attributes in your extended entity. When a form driven flow is saved, the input parameters will be populated based on the matching entity attributes.

To bind input parameters to entity attributes:

1. Open the entity extension for your transient data entity.
2. Select the **SAVE Settings** tab.
3. The server automation script **Name** will be populated automatically with the transient data entity's automation script for save.
4. In the **Input Parameters Binding** section, match each input parameter to the entity attribute used to populate it. Click the Edit/Save links at the right end of each row to set the attribute names.
5. Optionally, you can manually enter a script in the **Input Parameters Custom Binding** to define custom input values for your input parameters.



> **IMPORTANT!**
>
> **Input Parameter Custom Binding** settings override settings in the **Input Parameters Binding** section

6. Click **Save and Reload** ( ) at the top right corner of the page.

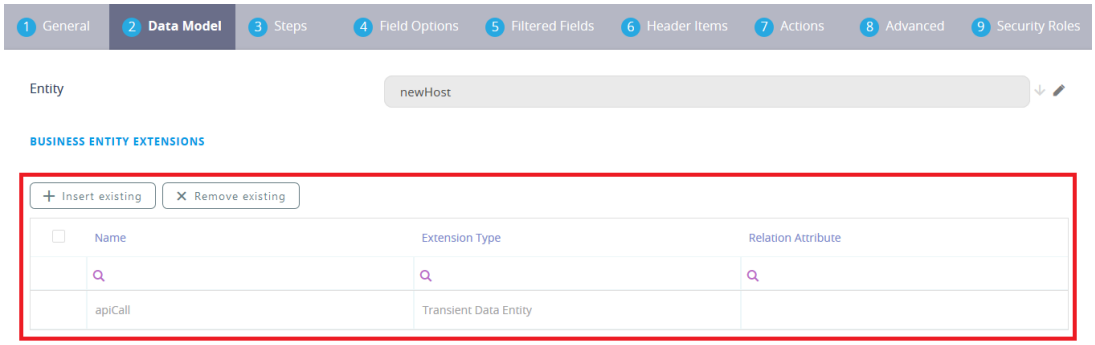
Display transient data entity attributes in form driven flows

Once you extend a platform entity data model with a transient data entity (see "Extend platform data entities with transient data entities" on page 136), you can display the transient data entity's attributes in the extended entity's form driven flows. Transient data entities with single instance output structures allow you to

display individual fields for each attribute, while those with collection output structures can be displayed in grids based on the transient data entity's default view (see "Create transient data entities" on page 130 for details).

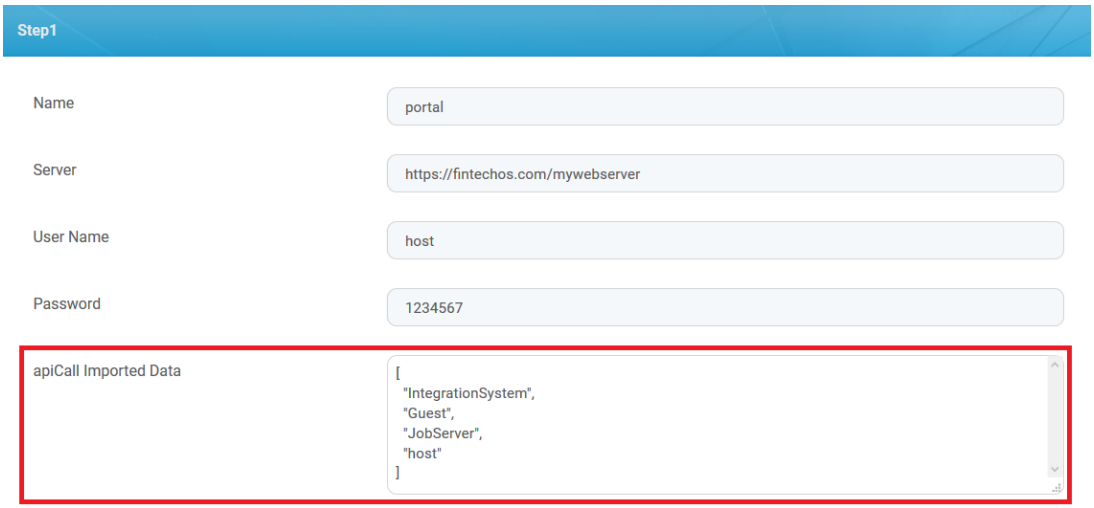
Display transient data entity attributes for single instance outputs

1. Create a form driven flow based on a platform entity that was extended with a transient data entity with single instance output structure.
    - For information on how to create form driven flows, see "Form Driven Flows" on page 203.
    - For information on how to extend a platform data entity with a transient data entity, see "Extend platform data entities with transient data entities" on page 136.
    - For information on how to create automation scripts with single instance output structures, see "Customizing the Output Structure" on page 500.
2. Make sure to include the transient data entity extension in the form driven flow's data model.



3. In the form driven flow's UI editor, you will be able to insert data templates with the transient data entity's virtual attributes.

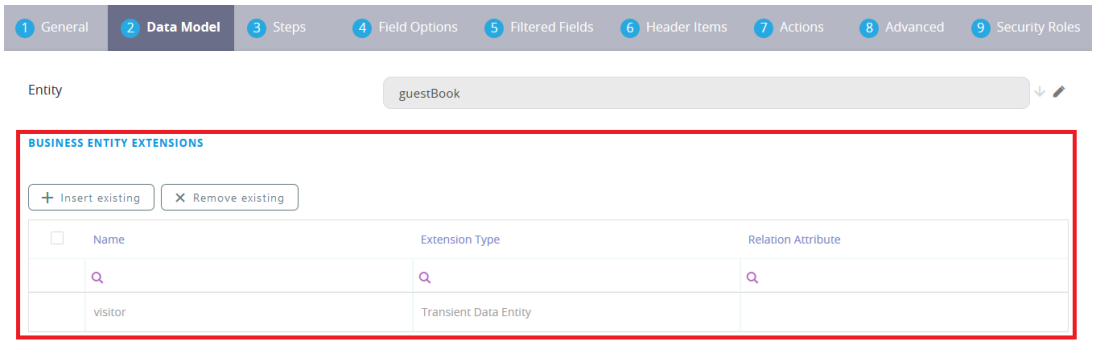
4. The corresponding transient data entity's attributes will be displayed in the user interface.



Display transient data entity attributes for collection outputs

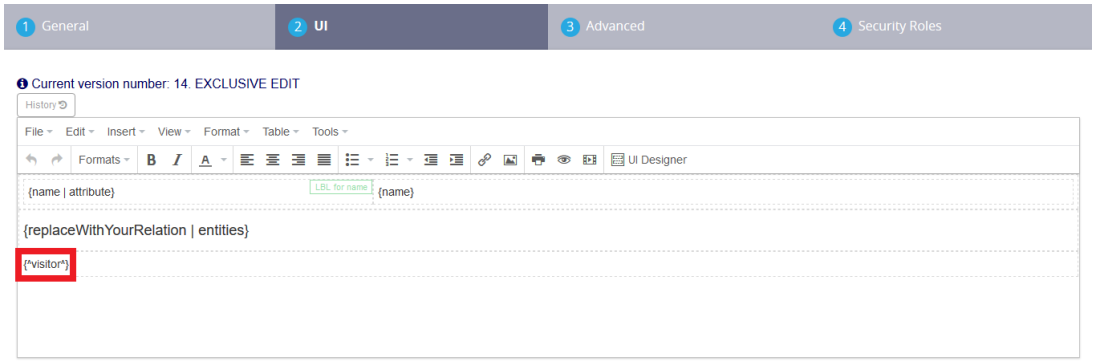
1. Create a form driven flow based on a platform entity that was extended with a transient data entity with collection output structure.
    - For information on how to create form driven flows, see "Form Driven Flows" on page 203.
    - For information on how to extend a platform data entity with a transient data entity, see "Extend platform data entities with transient data entities" on page 136.
    - For information on how to create automation scripts with collection output structures, see "Customizing the Output Structure" on page 500.

2. Make sure to include the transient data entity extension in the form driven flow's data model.

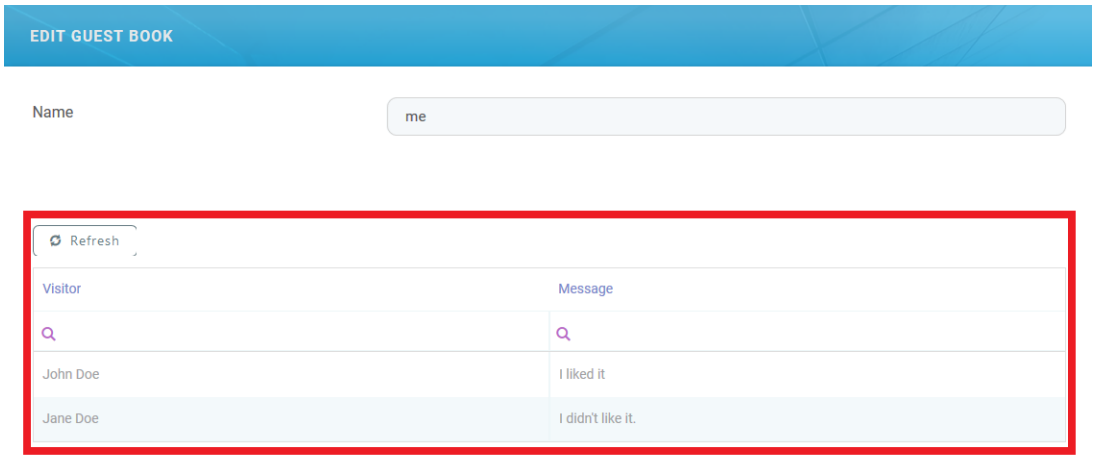


3. In the form driven flow's UI editor, insert a relation container with the value based on the entity extension's name, using the following syntax:

```
{^entity extension name^}
```



4. The corresponding transient data entity's attributes will be displayed in the user interface in a grid based on the transient entity's default view.



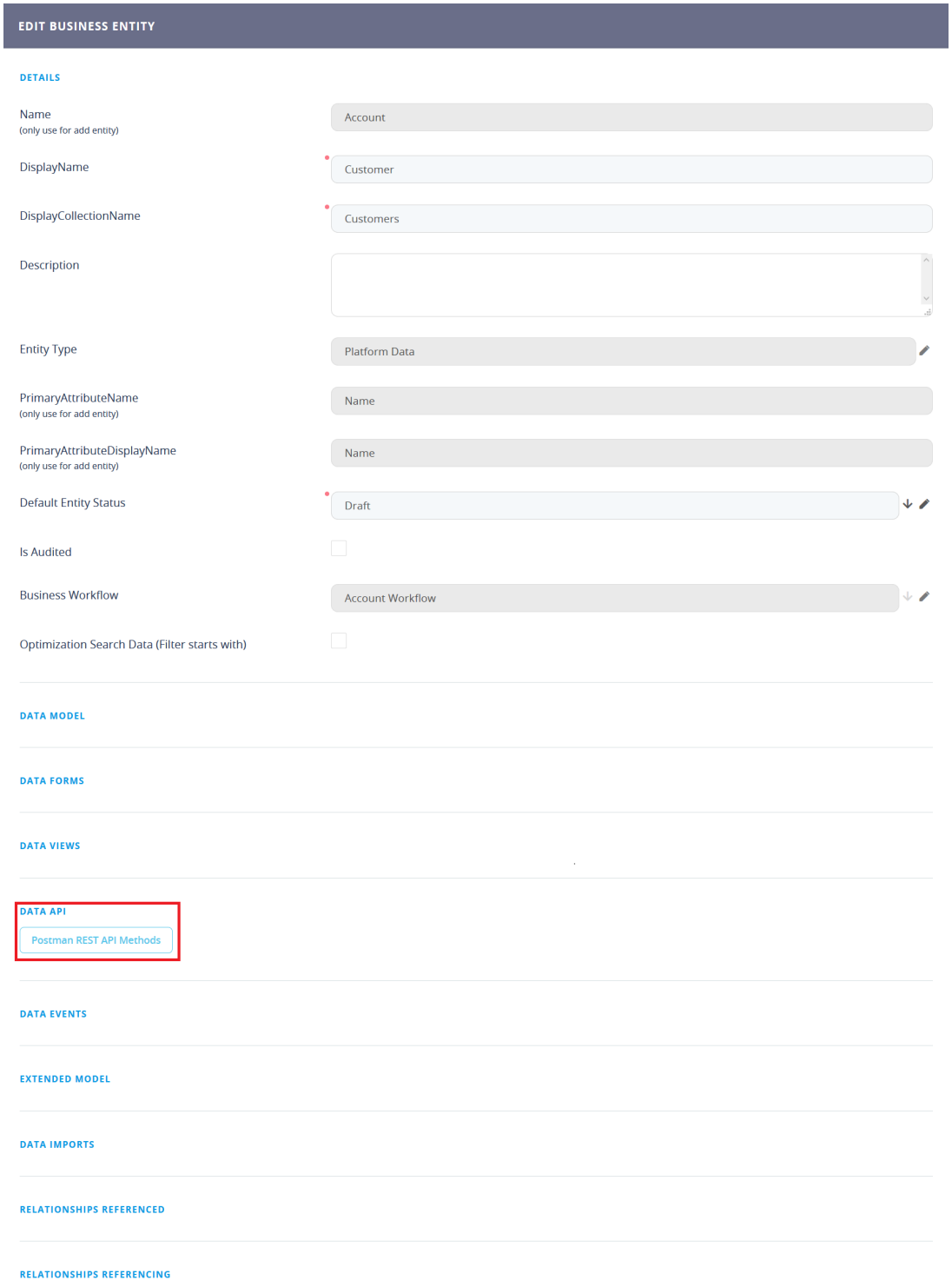
Sample API Calls

The Data API section allows you to download a Postman API Client collection with sample API calls pre-populated with the entity's attributes. For detailed information about the API calls supported in FintechOS, see the API Reference Guide.

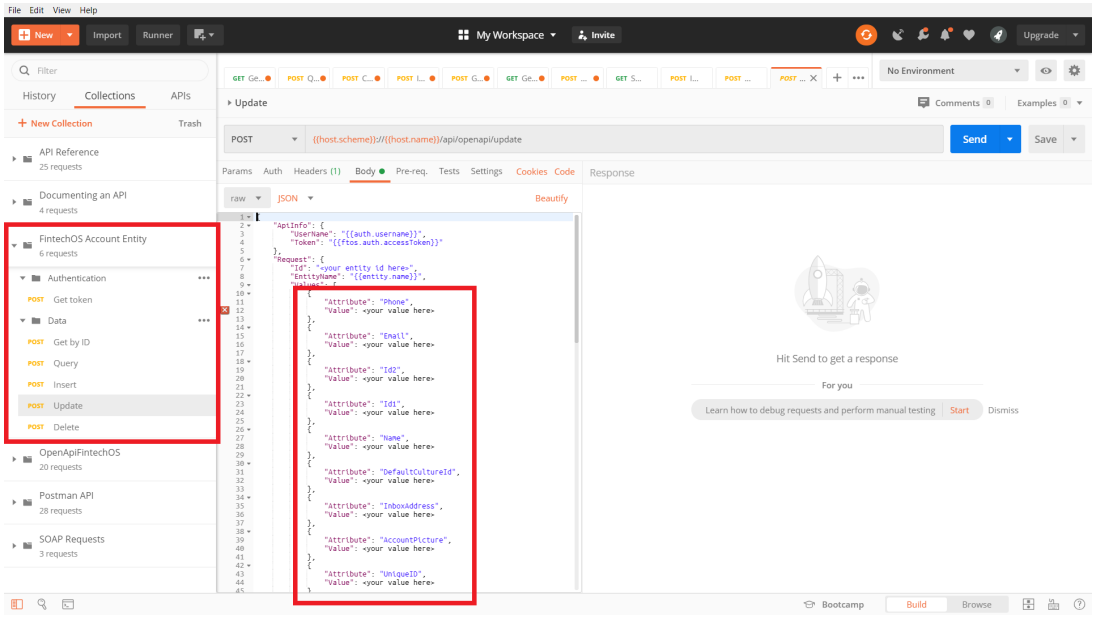
To download the sample API calls:

1. Open the entity in the Data Model Explorer.

2. Expand the **Data API** section and click the **Postman REST API Methods** button.



3. A JSON file containing the Postman collection will be downloaded on your local machine.
4. In Postman, click **File** > **Import** and select the downloaded JSON file.
5. Once the import finishes, the API calls collection will be visible in Postman.
6. The collection contains basic API calls for authentication and CRUD operations with the entity's attributes names pre-populated.



Data Import Template

To import a package or data with its metadata from another system source follow these steps:

HINT
Prior importing data in bulk in FintechOS, go to the legacy system and export the data in an Excel file.
Do not include the source primary keys in the data exports. Primary keys are generated automatically by the system.

Make sure to include in the source export any attributes that are required in the destination entity, otherwise the import will fail.

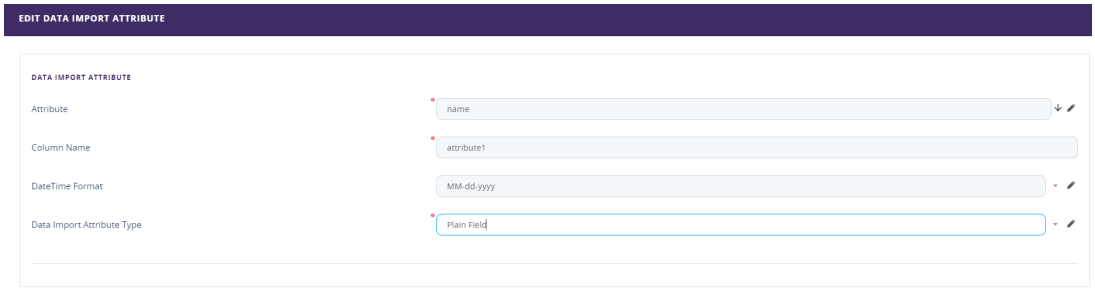
1. Open the Innovation Studio and select the **Evolutive Data Core**.
2. Click on the **Data Import Template** menu item. To open an existing template search in the list for the desired one. To create a new one click on ''**Insert**''. To erase a template, click on the ''**Delete**'' button on the right side of the screen.
3. Fill in the following:

Field	Data Type	Description
Name	Text	Insert a name for the package.
Entity	Option set	Choose the entity you wish to have the import done on.
Unique constraint	Option set	Select the constraint. For more information, see "Entity Unique Constraints" on page 78. For each entity record, there will be unique values.

4. Click the ''save and reload'' button. Two grids will show up: LIST OF DATA IMPORT ATTRIBUTES and LIST OF DATA IMPORTS.
5. In the list of data import attributes, click the ''insert'' button to insert an attribute by filling in the following fields:

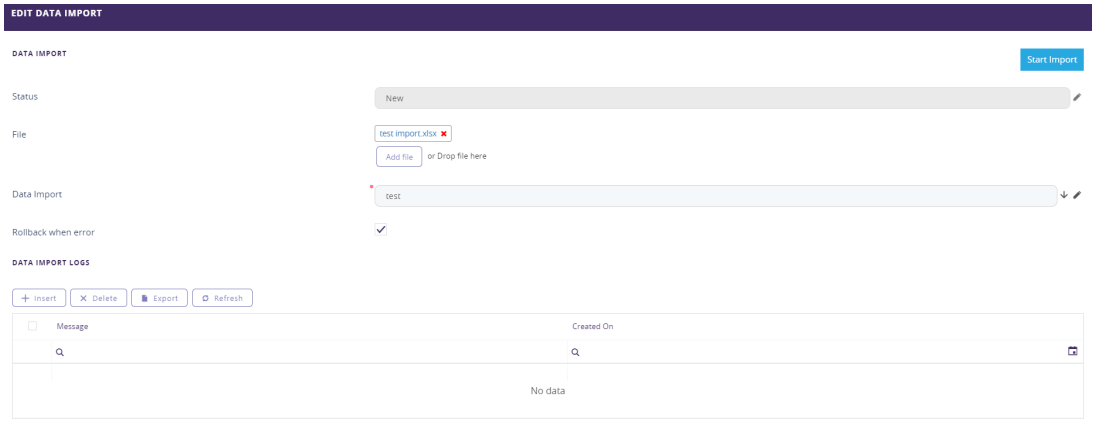
Field	Data Type	Description
Attribute	Option set	Select the attribute you wish to include. Be sure to include the attribute which has the unique constraint.

Field	Data Type	Description
Column name	Text	Insert a name for the column. **IMPORTANT!** The name must be the same as the name of the column in the Excel file.
Date time format	Option set	Select the way in which to write the date, if the attribute is a date type: • none • dd/MM/yyyy • MM/dd/yyyy • dd-MM-yyyy • MM-dd-yyyy.
Data Import Attribute Type	Option set	Select the type: • Plain Field • Lookup Field • Optionset Field.



6. Click the ''Save and close'' button.

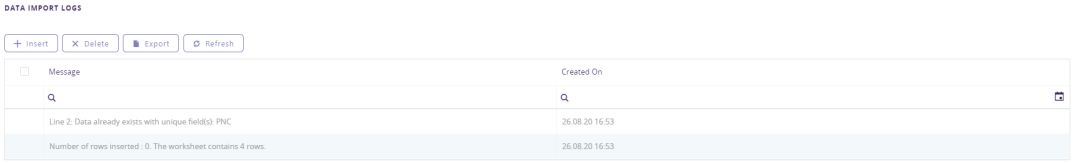
7. Repeat for as many attributes as needed. It is possible to delete an attribute by clicking the ''Delete'' button next to the ''Insert'' button.
8. In the list data imports, click the ''**Insert**'' button.



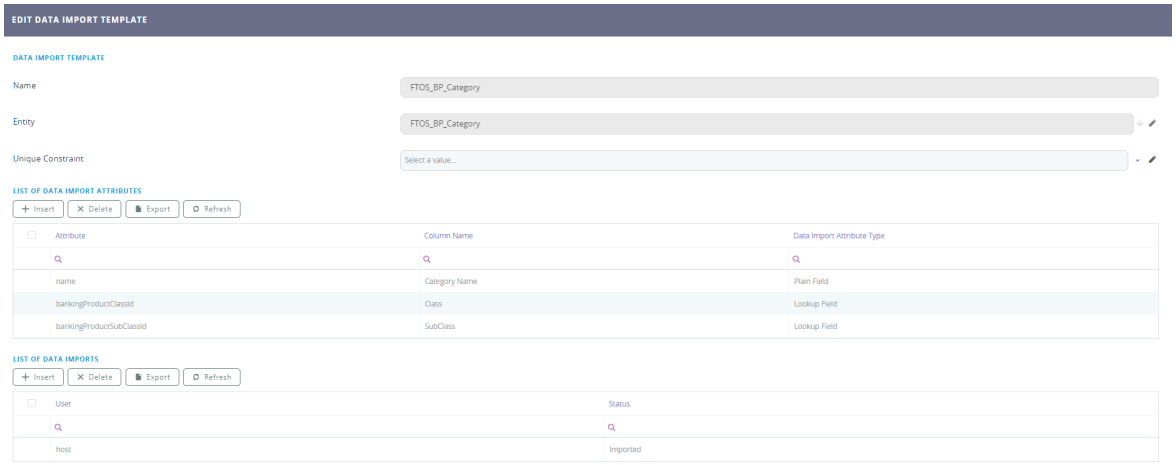
9. Fill in the following:

Field	Data Types	Description
Status	Option set	This is auto-filled. The options are: • new (this is the initial status of an import before the ''Start import'' button has been clicked) • imported (this is the confirmation if the import was successful) • error (this is the status if the import failed).
File	File	Add the Excel file here.
Data Import	Option set	Name of the template.
Rollback when error	Bool	This checkbox will enable the process of coming back to the initial state of the data base when the import fails.

10. Click the ''Start Import'' button. In the DATA IMPORT LOGS grid, the history of the process will be recorded. If the is an error the messages will be shown here.



11. Click the ''**Save and close**'' button. Repeat as many times as needed.



Data Governance

Data governance allows you to classify sensitive data and anonymize it on request. This is particularly useful to ensure compliance with data protection regulations, such as the GDPR art. 17 Right to erasure (Right to be forgotten).

> ⚠ **IMPORTANT!**
> When anonymizing data, the relevant record attributes are permanently deleted and replaced with generic information, such as *Sensitive data deleted*.

The data anonymization process consists of multiple steps, involving both back-end and front-end operations, as described below:

1. **Sensitive Data Classification** - In this stage, digital developers or consultants who are familiar with the data model define the "Sensitive Context" below and the "Sensitive Data" on the next page that is subject to anonymization in Innovation Studio.
2. **Issuing Anonymization Requests** - Once the sensitive data has been classified, operators can use FintechOS Portal to make "Data Anonymization Requests" on page 162. The operators search for the relevant data to be anonymized based on generic criteria, such as the sensitive context and the type of sensitive data (name, telephone number, identification code, etc.), not based on specific entities and attributes from the data model. Then, they identify, validate, and approve the data to be anonymized.
3. **Data Anonymization** - After the anonymization requests are approved, "Data Anonymization Jobs" on page 160 can process the requests and anonymize the actual data. For this purpose, you must configure the job server to run an anonymization service for the corresponding sensitive context.

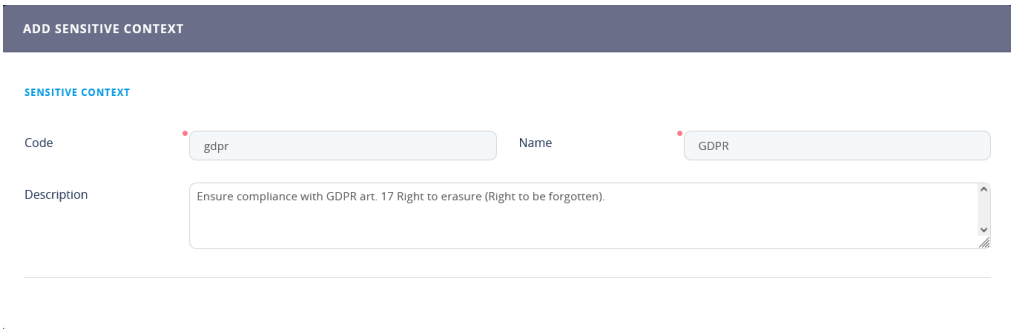
Sensitive Context

The sensitive context groups components that are involved in data anonymization for a specific purpose, such as ensuring GDPR compliance. Specifically, for such a purpose, the sensitive context will help:

- Define the "Sensitive Data" on the next page.
- Provide a reference for the "Data Anonymization Jobs" on page 160 to find the relevant data to be anonymized.

Create a Sensitive Context

1. In Innovation Studio, go to **Main Menu** > **Evolutive Data Core** > **Data Governance** > **Sensitive Data Settings**.
2. Click the **Insert** button at the top right corner of the screen.
3. In the Add Sensitive Context page, fill in the following fields:
    - **Code** - The sensitive context identifier.
    - **Name** - The sensitive context name.
    - **Description** - Optional description for the data anonymization purpose of the sensitive context.



4. Click the **Save and Close** button at the top right corner of the screen.

Sensitive Data

Sensitive data definitions specify the data anonymization settings applicable for a sensitive context, such as:

- The attributes used by operators to search for the records that contain sensitive data.
- The attribute values that must be anonymized once those records are identified.
- The validations required to allow the data anonymization to proceed.

Each sensitive data definition is built around a sensitive entity (and, optionally, its related entities) on which the anonymization is applied. The sensitive data definition also specifies the validation rules that must be fulfilled in order to allow the anonymization.

1 Configure the Sensitive Entity

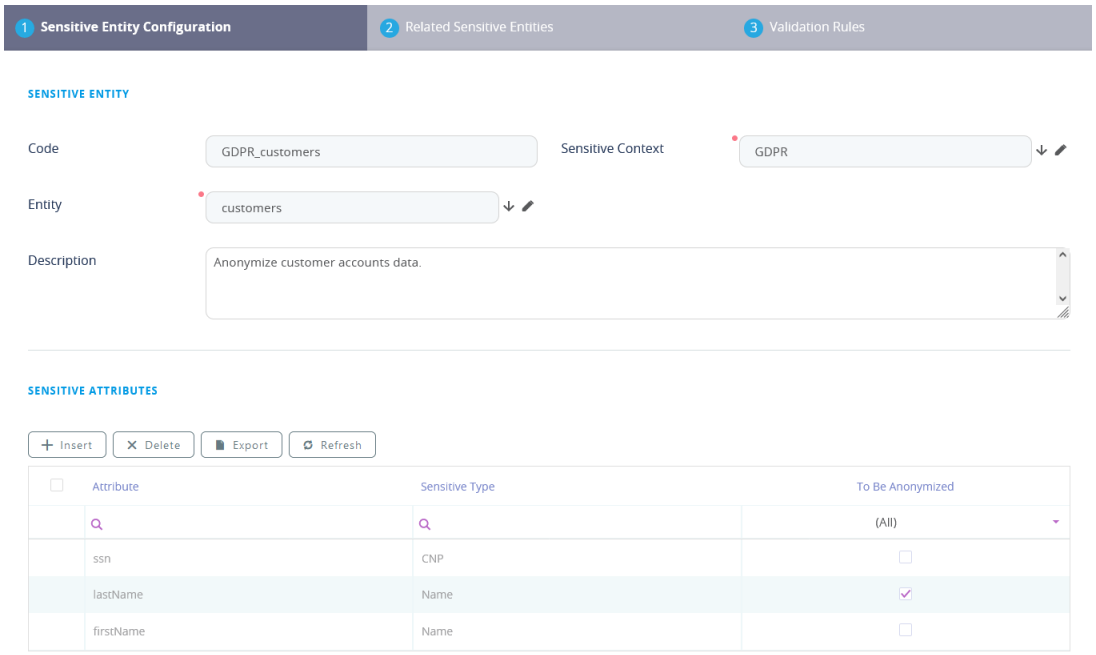
1. In Innovation Studio, go to **Main Menu** > **Evolutive Data Core** > **Data Governance** > **Sensitive Data Definitions**.
2. Click **Insert** at the top right corner of the screen.
3. In the Sensitive Entity Configuration tab, fill in the following information:
    - **Code** - The identifier for the sensitive data definition.
    - **Sensitive Context** - The applicable "Sensitive Context" on page 152.
    - **Entity** - The sensitive entity on which the anonymization is based.
    - **Description** - Optional detailed description of the sensitive data definition.
4. Click the **Save and Reload** button at the top right corner of the screen.
5. In the Sensitive Attributes list, click the **Insert** button to add the sensitive entity's attributes that are relevant for the anonymization (either as search criteria or as data to be anonymized). For each attribute, specify the following values:
    - **Sensitive Entity** - This is automatically populated with the sensitive entity name.
    - **Attribute** - The attribute's name.
    - **Sensitive Type** - Attribute classification used by operators as search criterion when performing "Data Anonymization Requests" on page 162. The following values are available: CNP, Name, Phone, First Name, Last Name.

> **NOTE**
>
> Only attributes with sensitive types can be searched in data anonymization requests.

- **To Be Anonymized** - Check to indicate that, if a sensitive entity record is subject to data anonymization, this attribute will be anonymized in accordance with its data type:

Attribute Data Type	Anonymized Attribute Value
Text	Sensitive data deleted.
File	[]
Text Area	Sensitive data deleted.
Numeric	0
Whole Number	0
Date Time	01.01.1900
Date	01.01.1900
Bool	NULL
Option Set	NULL
Lookup	NULL

6. After you finish adding all the sensitive attributes, click the **Save and Reload** button at the top right corner of the page.



2 Define the Related Sensitive Entities (optional)

The related sensitive entities allow you to propagate data anonymization to a chain of entities that are related to the main sensitive entity. For instance, when anonymizing a customer's data, you may want to also anonymize the data for all the bank accounts that the customer holds (which is stored in a related entity).

You can also include entities that are related to other related entities, extending the data anonymization scope to a tree of entities originating from the main sensitive entity.

To define a related sensitive entity:

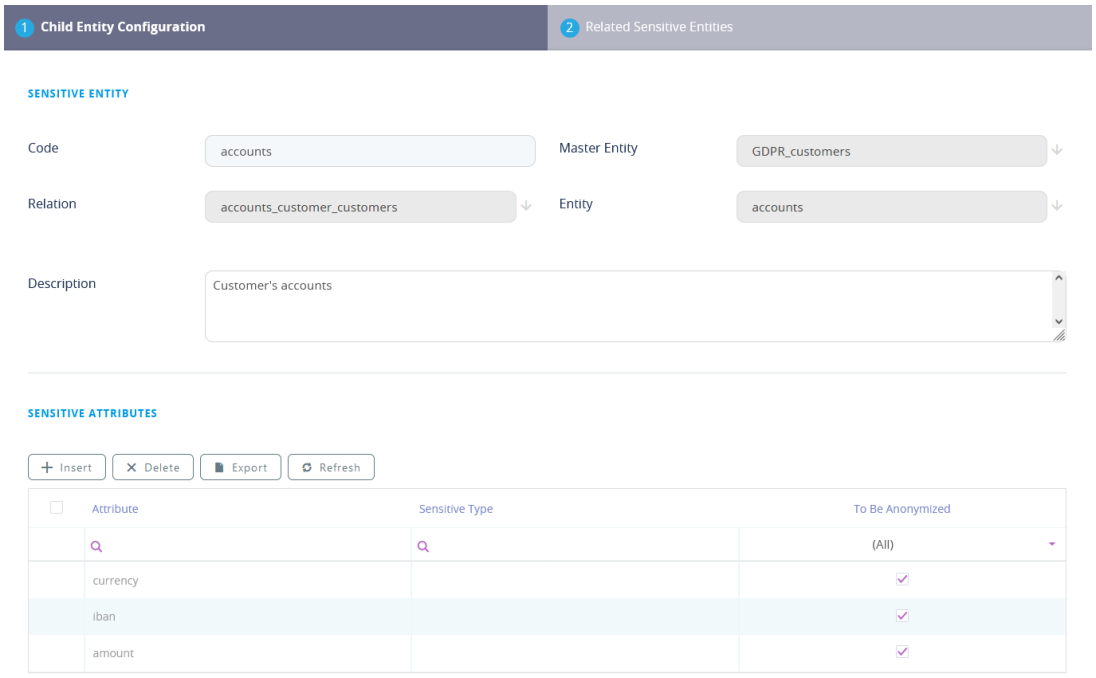
1. In the sensitive data definition editor, select the **Related Sensitive Entities** tab.
2. In the Related Entities list, click the **Insert** button to add the sensitive entities that are part of the data anonymization scope. For each related entity, specify the following values:
    - **Code** - The identifier for the related entity.
    - **Master Entity** - Automatically populated with the sensitive entity identifier.

- **Relation** - Select from the list of relationships of the sensitive entity.
- **Entity** - Related entity name. Automatically populated based on the selected relation.

3. Click the **Save and Reload** button at the top right corner of the screen.
4. In the Sensitive Attributes list, click the **Insert** button to add any entity attributes that are part of the data anonymization scope. For each attribute, specify the following values:
    - **Sensitive Entity** - This is automatically populated with the related sensitive entity name.
    - **Attribute** - The attribute's name.
    - **To Be Anonymized** - Check to indicate that, if a sensitive entity record is subject to data anonymization, this attribute will be anonymized in accordance with its data type.

5. Click **Save and Close** at the top right corner of the screen.

6. Repeat from Step 4 to add any remaining sensitive attributes.



7. You can further define downstream related entities by selecting the **Related Sensitive Entites** tab and repeating the process.
8. Click **Save and Close** at the top right corner of the screen.

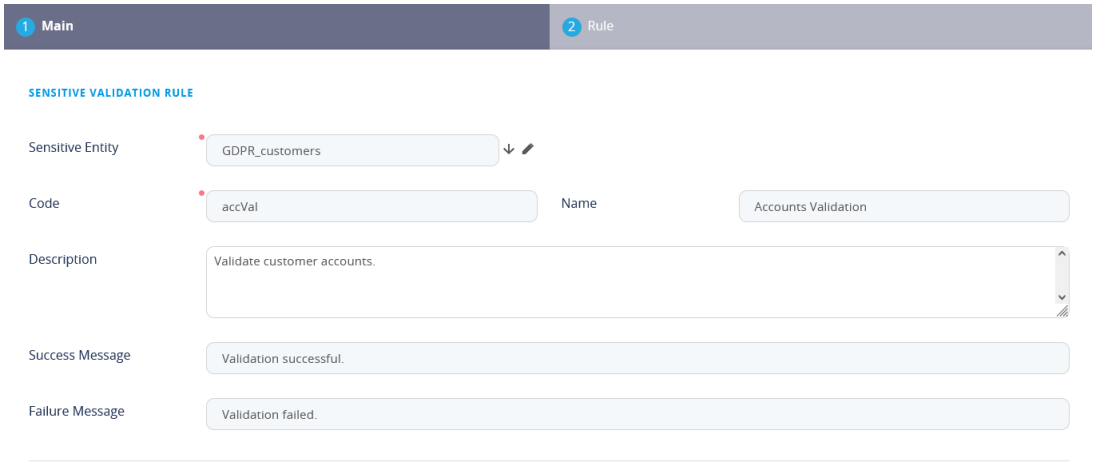
3 Add Validation Rules

Validation rules can prevent data anonymization if certain criteria are not met. For instance, you may want to allow data anonymization of a bank account data only if the account balance is 0.

To add a validation rule for a sensitive data definition:

1. In the sensitive data definition editor, select the **Validation Rules** tab.
2. In the Sensitive Validation Rules list, click the **Insert** button to add a rule.

3. In the **Main** tab, specify the following values:
    - **Sensitive Entity** - The identifier for the sensitive data definition.
    - **Code** - The identifier of the validation rule.
    - **Name** - The name of the validation rule.
    - **Description** - Optional detailed description of the validation rule.
    - **Success Message** - Message displayed in the operator's interface when validating the result of a "Data Anonymization Requests" on page 162 if the validation is successful.
    - **Failure Message** - Message displayed in the operator's interface when validating the result of a "Data Anonymization Requests" on page 162 if the validation fails.



4. In the **Rule** tab specify the expression of the validation rule. The expression must return **true** in case of a successful validation or **false** if the validation fails.

5. Click **Save and Close** at the top right corner of the screen.

Data Anonymization Jobs

Approved records are anonymized using an SQL procedure. For each "Sensitive Context" on page 152, you must configure the FintechOS job server to run a corresponding anonymization service.

1 Set up the Anonymization Service

Open the FintechOS Job Server's **services.config** file in a text editor, and add a service based on the following model:

```
<service>
    <name>FTOS.Anonimization</name>
    <type>mixed</type>
    <steps>
        <step>
            <name>Run Sensitive_Context_Name anonymization</name>
            <type>sql</type>
            <command>exec ebs.uspFTOS_SNSD_DeleteSensitiveData
'Sensitive_Context_Name'</command>
            <sqlConnection>Data Source=SqlServerName ;Initial
Catalog=DataBaseName;User
ID=userName;Password=userPassword</sqlConnection>
            <mandatory>true</mandatory>
        </step>
    </steps>
</service>
```

2 Set up the Anonymization Service Trigger

Open the FintechOS Job Server's **schedule.config** file in a text editor, and add a trigger for the anonymization service based on the following model:

```
<trigger>
    <name>FTOS.Anonimization</name>
    <!--<calendar>holyday1</calendar>-->
    <startTime>02.01.2019 11:00</startTime>
    <endTime>03.11.2080 11:02</endTime>
    <poolTime>60</poolTime>
    <repeatCount>-1</repeatCount>
    <rescheduleAfterRun>false</rescheduleAfterRun>
    <async>false</async>
    <expression>0/10 * * * * ?</expression>
    <services>
        <service>FTOS.Anonimization</service>
    </services>
</trigger>
```

The following settings apply:

- `<poolTime>` is expressed in seconds.
- `<repeatCount>` when set to -1 will run indefinitely. When set to a positive value (eg. 5) will run for only 5 times at each pool time.
- `<expression>` is a Cron expression. When present, the Cron expression will override the `<poolTime>`.

3 Restart the FintechOS Job Server

After editing the services.config and schedule.config files, you need to restart the FintechOS Job Server for your changes to take effect:

1. On the FintechOS Job Server environment, open the **Command Prompt** application in **Administrator mode**.

2. Enter the following command and wait for the job server service to stop:

```
net stop FTOS.JobServer.<instance name>
```

3. Enter the following command and wait for the job server to restart:

```
net start FTOS.JobServer.<instance name>
```

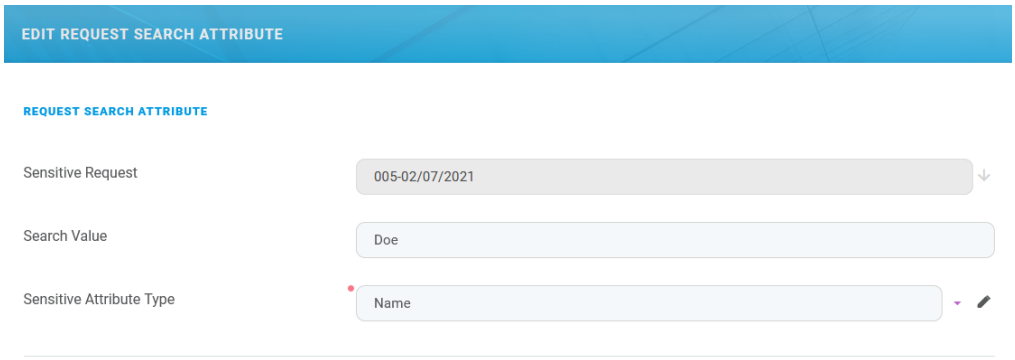
Data Anonymization Requests

Data anonymization requests allow FintechOS Portal operators to mark specific records within a "Sensitive Context" on page 152 for anonymization. Once the relevant data is identified, validated, and approved for anonymization by the operator, a data anonymization job can process the marked records.

1 Create a Sensitive Request

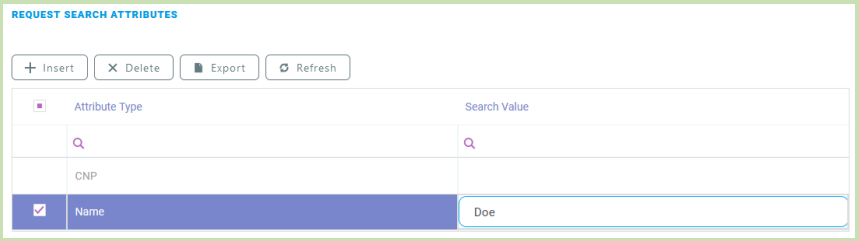
1. In FintechOS Portal, go to **Main Menu** > **Data Anonymization Requests**.
2. Click the **Insert** button at the top right corner of the screen.
3. Fill in the following sensitive request values:
    - **Request no** - Identifier for the data anonymization request.
    - **Request Date** - Select a reference date for the data anonymization request.
    - **Sensitive Context** - Select the "Sensitive Context" on page 152 relevant for the request.
4. Click the **Save and Reload** button at the top right corner of the screen.
5. In the Request Search Attributes grid, you will see the attributes that are available for search according to the sensitive context's "Sensitive Data" on page 153 definition. Edit the search attribute you wish to use as search criteria:

- **Sensitive Request** - This field is populated automatically with the data anonymization request identifier.
- **Search Value** - The key term you wish to use for your search, such as a user's first name or SSN.
- **Sensitive Attribute Type** - The clasification for the sensitive attribute type. The following values are available: CNP, Name, Phone, First Name, Last Name.



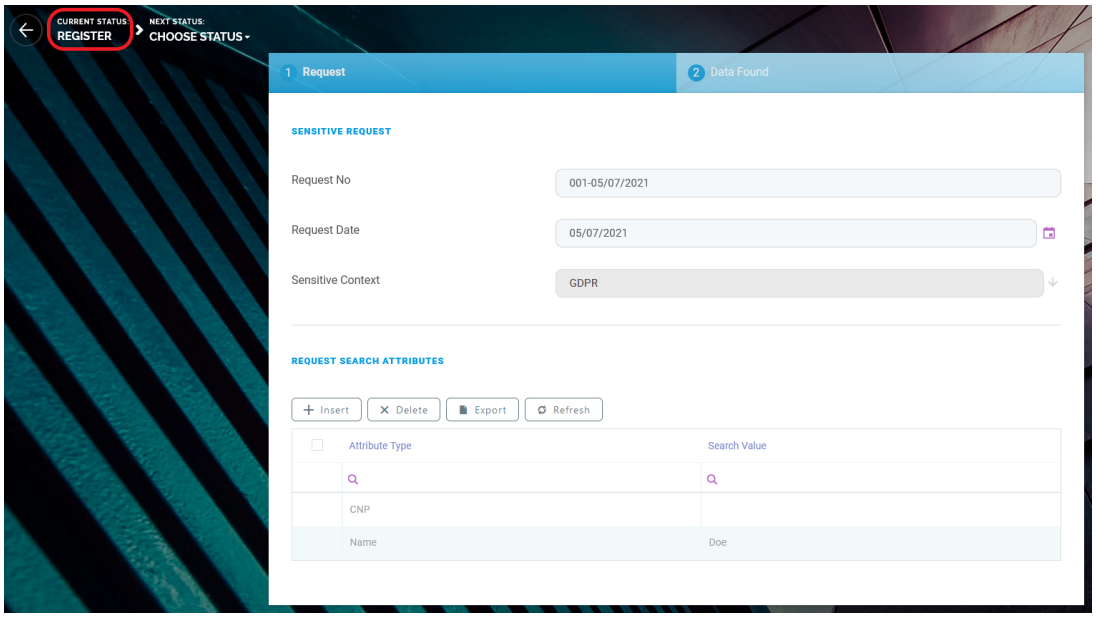
HINT

Although you can define any sensitive attribute type, the search will only work for the values that match the "Sensitive Data" on page 153 definition. This is why the best approach is to edit the desired search value directly in the Request Search Attributes grid.



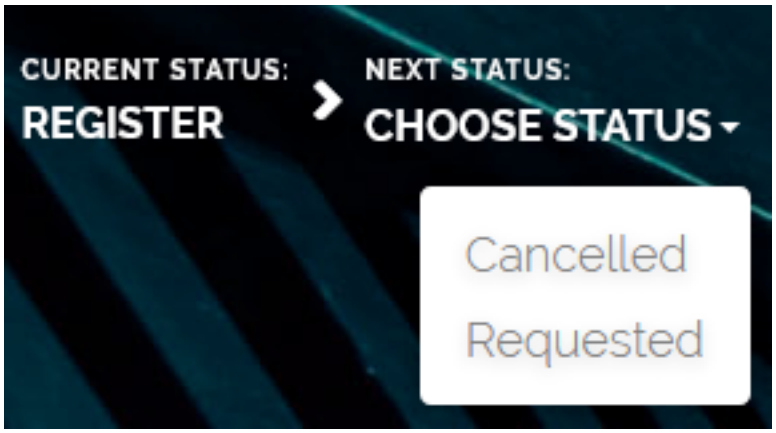
- Click **Save and Close** at the top right corner of the screen.

6. The data anonymization request is saved with the **Register** status.

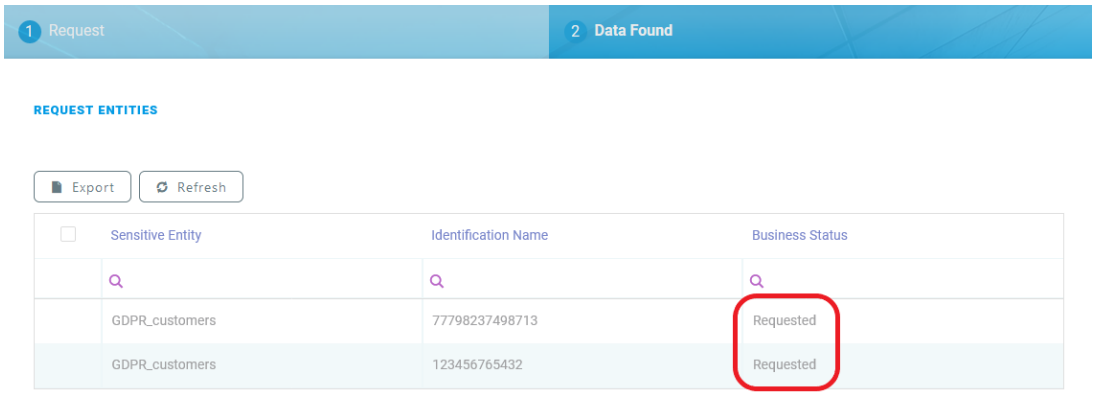


2 Check the Returned Sensitive Data

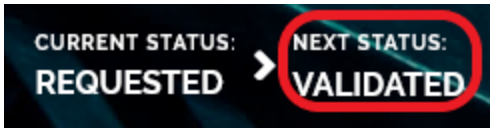
1. Once all the settings of the Data Anonymization Request are configured, you can choose to either cancel the request or proceed with the search by selecting the desired status from the top left corner of the screen.



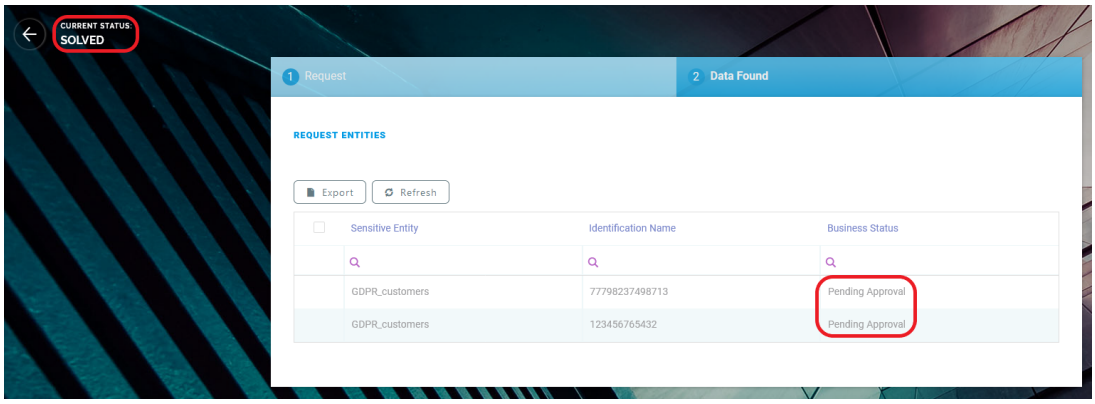
2. Select the **Requested** status to lock the search criteria and perform the search. Once the status is updated, the search results are populated in the Data Found tab of the request. Each search result will also be in the Requested business status.



3. You can double click on the desired entities to do a preliminary check on the individual results. When ready, select the Validated status at the top right corner of the screen to run the data validations that were configured in the sensitive data (for details, see " 3 Add Validation Rules" on page 158).

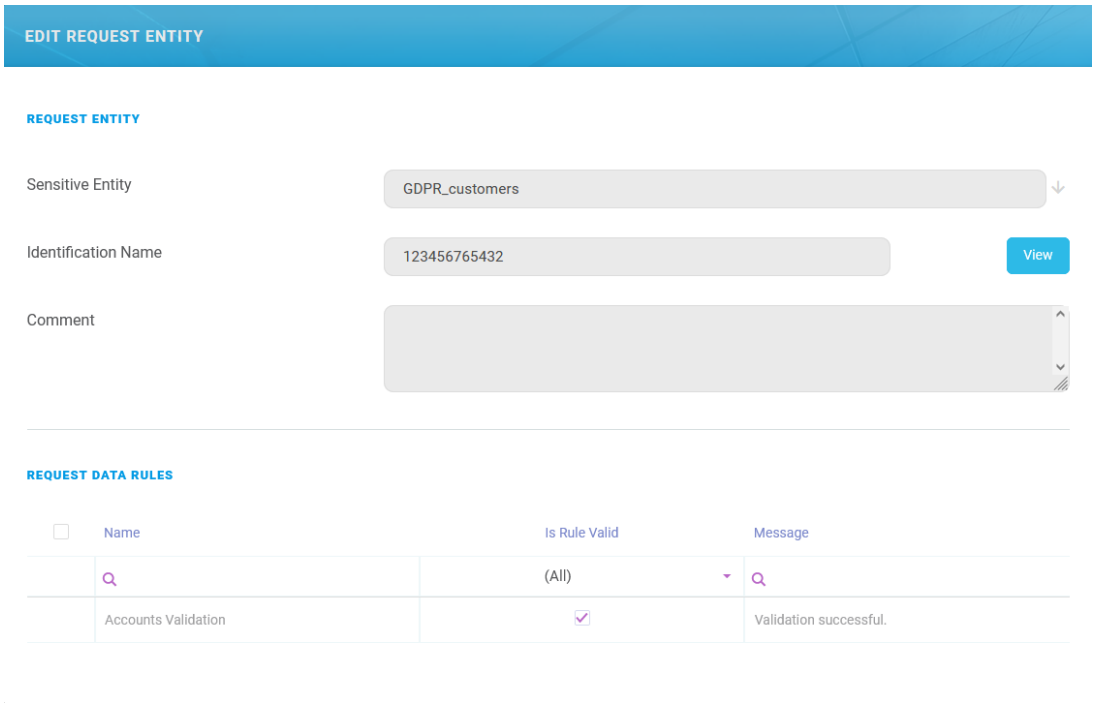


4. The request will advance to the Solved status and each search result will go into the Pending Approval status.



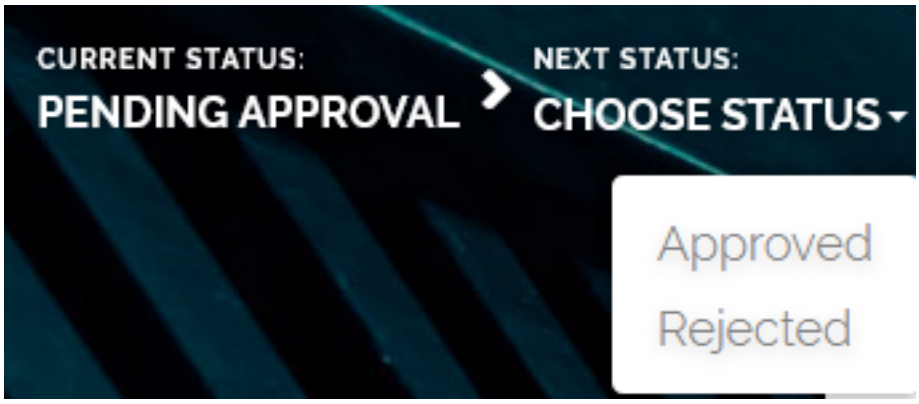
3 Approve the Records for Data Anonymization

1. Open each search result in the grid and check it:



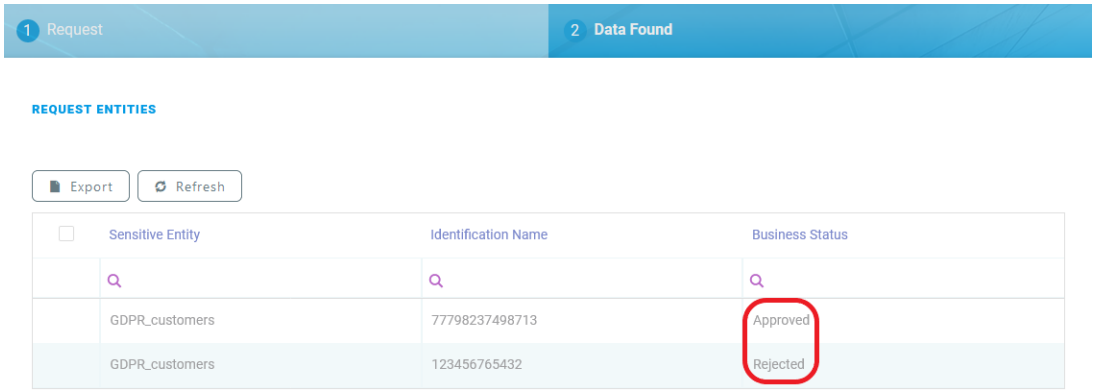
   - **Sensitive Entity** - Sensitive entity based on which the search was performed. For details, see " 1 Configure the Sensitive Entity" on page 154.
   - **Identification Name** - Primary attribute of the returned sensitive entity record.
   - **View** - Click this button to view the sensitive entity record in its default edit form.
   - **Comment** - Sensitive entity description (if provided).
   - **Request Data Rules** - The results of each validation rule applied to the returned data.

2. Based on the search result's data, from the top left corner of the screen, choose to either approve or reject data anonymization for the record.

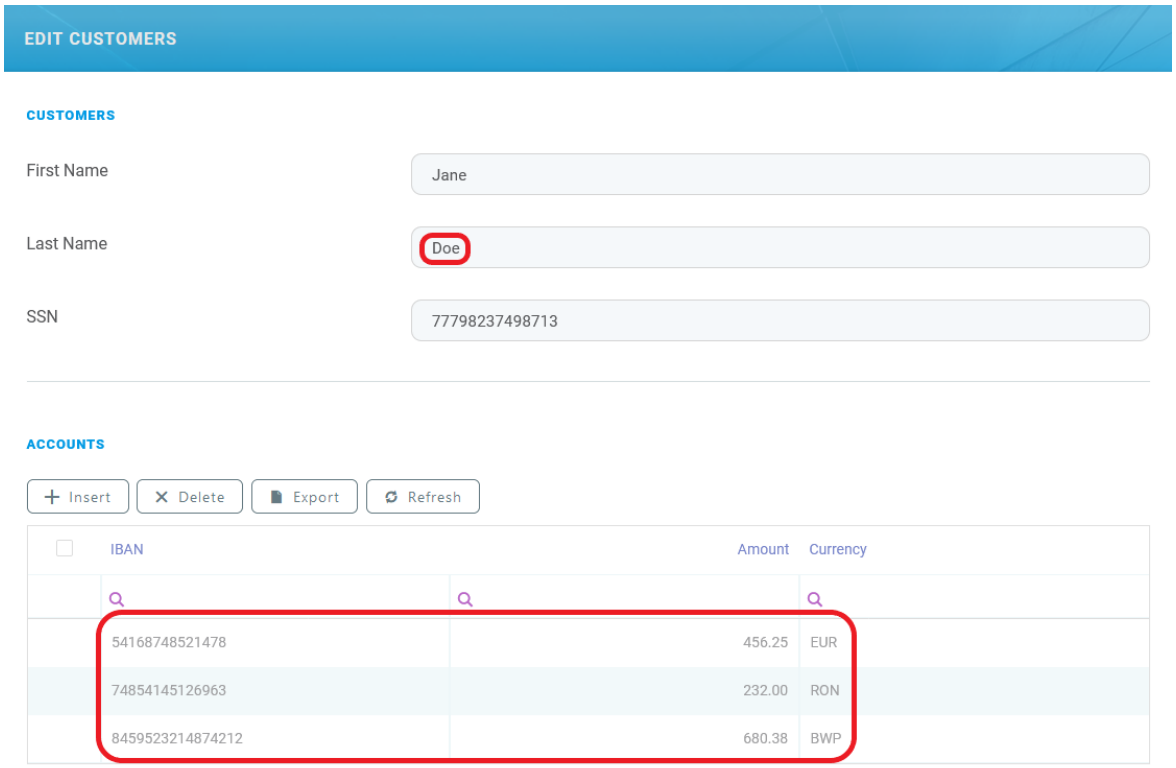


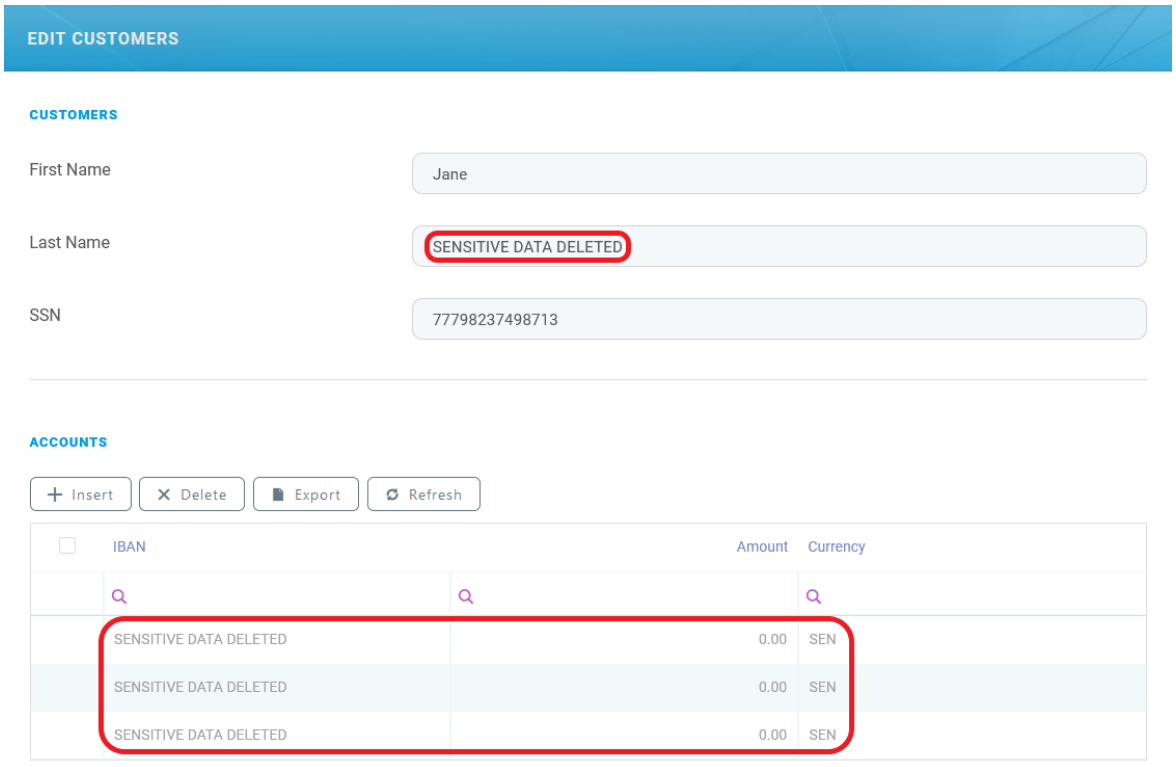
> **IMPORTANT!**
>
> When you approve a record, the anonymized record attributes are permanently deleted and replaced with generic information, such as *Sensitive data deleted*.

3. Repeat for the remaining search results and either approve or reject them.

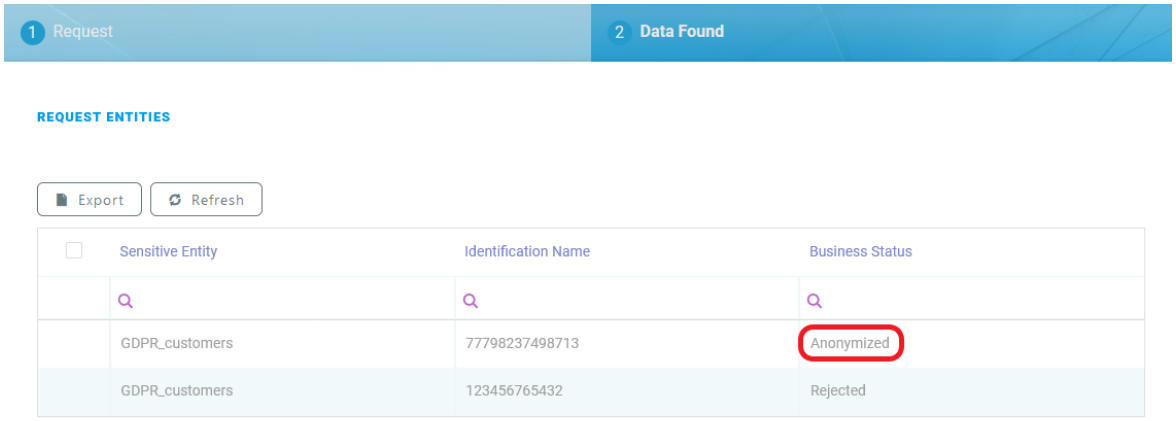


When the FintechOS Job Server runs the corresponding "Data Anonymization Jobs" on page 160, the approved sensitive data is anonymized.


EDIT CUSTOMERS
CUSTOMERS
First Name
Jane
Last Name
Doe
SSN
77798237498713
ACCOUNTS
Insert
Delete
Export
Refresh
IBAN
Amount
Currency
54168748521478
456.25
EUR
74854145126963
232.00
RON
8459523214874212
680.38
BWP


EDIT CUSTOMERS
CUSTOMERS
First Name
Jane
Last Name
SENSITIVE DATA DELETED
SSN
77798237498713
ACCOUNTS
Insert
Delete
Export
Refresh
IBAN
Amount
Currency
SENSITIVE DATA DELETED
0.00
SEN
SENSITIVE DATA DELETED
0.00
SEN
SENSITIVE DATA DELETED
0.00
SEN

The business status of the sensitive entity is set to Anonymized.



External APIs

External APIs allow you to access third party resources that are available through RESTful web services. External APIs are predefined collections of API calls that are executed in sequence within the same context, effectively turning Innovation Studio into an API client with advanced capabilities.

This allows you, for instance, to use a credit reporting agency's API to automatically check the creditworthiness of an applicant.

Besides defining a sequence of API calls, some of the External API's advanced features include:

- Manage the API calls' paths and authentication in a centralized and secure mode, from the *web.config* file.
- Rules for the execution sequence of the API calls, such as canceling subsequent calls in the sequence in case of failure or establishing dependencies where one call only executes after the successful completion of other calls.
- Custom JavaScript code to be executed before or after a call.

- Call timing configurations such as expiration time of the authentication token, retries on error, number of retries, or delays between retries.
- Logging for the API calls' requests and responses.

HINT
You can take advantage of the External API on the client side and the Fintech API endpoints on the server side to exchange data between different FintechOS instances. For more information about Fintech API resources, see the API Reference Guide.

How to create an External API **170**
How to configure External API calls **171**
How to call a External API **180**
External API General Settings **181**
Data Model **182**

How to create an External API

1. Open Innovation Studio in Developer mode.
2. Open the **Main Menu**
3. Select **API Ecosystem** > **External API**. This opens the list of external APIs.
4. Click the **Insert** button at the top right corner of the page to add a new External API. This opens the Add External API page.
5. In the page that opens, enter a **Code** and, optionally, a **Name** for your External API.
6. Click the **Save and Close** button at the top right corner of the page.

This creates an empty External API. To add API calls to the External API, see "How to configure External API calls" on the next page.

How to configure External API calls

1. Open Innovation Studio in Developer mode.
2. Open the **Main Menu**
3. Select **Fintech Automation** > **External API**. This opens the External API List page.
4. In the External API List, double click the External API you wish to edit. This opens the Edit External API page.
5. In the External API Details grid:
    - **To add a new API call to the pipe**, click the Insert button.
    - **To edit an existing API call**, double click the API call from the grid.
6. In the page that opens, fill in the API call's details and click the **Save and Close** button at the top right corner of the page when done. For details on how to set up an API call, see the following:
    - "External API Call – Settings" below
    - "External API Call – Parameters" on page 178
    - "External API Call – Custom JavaSript Reference" on page 179

External API Call – Settings

Setting	Description
Code	API call's code. This setting is filled automatically based on the following schema [*External API code*].[*API call Order No.*]. For details, see "How to create an External API" on the previous page and "OrderNo" below.
Name	Enter a descriptive name for API call.
OrderNo	The order in which the API call is executed in the pipe. This allows you to configure multiple API calls to be executed in sequence in the same External API.

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>baseUrlIsKeyInConfigFile</td><td>
Check to use the base URL defined in the Innovation Studio web.config file for the API call's path ([base URL]+[endpoint]).

NOTE
Checking this option, disables the "BaseURL" on the next page field.

To configure a base URL in the web.config file, open the web.config file in a text editor and add the following entry in the <appSettings> section:

<pre><add key="FTOS_IntegrationProcessBaseUrl_
[<i>External API Detail Code</i>]" value=
[<i>base URL</i>]/></pre>
The External API's code and the API call's order number in the External API detail code are optional. For example:

• To use a base URL for the first API call in the P04 External API, enter:

<pre><add key="FTOS_IntegrationProcessBaseUrl_
P04.01" value="http://www.example.com/example_
Api"/></pre>
• To use a base URL for all the API calls in the P04 External API, enter:

<pre><add key="FTOS_IntegrationProcessBaseUrl_
P04" value="http://www.example.com/example_
Api"/></pre>
• To use a base URL for all the API calls of all the External API, enter:

<pre><add key="FTOS_
IntegrationProcessBaseUrl"
 value="http://www.example.com/example_Api"/></pre>
</td></tr>
</table>

Setting	Description
	If multiple base URL definitions in the *web.config* file are applicable for the same API call, the first one will be applied.
BaseURL	Base URL part of the API call's path (`[base URL]+[endpoint]`).
MethodName	Endpoint part of the API call's path (`[base URL]+[endpoint]`).
HttpMethod	HTTP request method used by the API call. Available options are: GET, POST, PUT, and DELETE.
hasHttpAuthentication	Check to make "httpAuthenticationType" on the next page mandatory.

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>httpAuthenticationType</td><td>
Allows you to select Basic HTTP authentication based on user-id/password pairs (RFC7617) or Bearer tokens based authentication (RFC6750).

To configure a user-id/password authentication scheme, open the Innovation Studio web.config file in a text editor and add the following entries in the <code><appSettings></code> section:

<pre><add key="FTOS_IntegrationProcessAuthUser_
[<i>External API Detail Code</i>]" value="
[<i>user-id</i>]"/>
<add key="FTOS_IntegrationProcessAuthPassword_
[<i>External API Detail Code</i>]" value="
[<i>password</i>]"/></pre>
The External API's code and the API call's order number in the External API detail code are optional. For example:

• To set credentials for the first API call in the P04 External API, enter:

<pre><add key="FTOS_IntegrationProcessAuthUser_
P04.01" value="guest"/>
<add key="FTOS_IntegrationProcessAuthPassword_
P04.01" value="guest"/></pre>
• To set credentials for all the API calls in the P04 External API, enter:

<pre><add key="FTOS_IntegrationProcessAuthUser_
P04" value="guest"/>
<add key="FTOS_IntegrationProcessAuthPassword_
P04" value="guest"/></pre>
• To set credentials for all the API calls of all the External API, enter:

<pre><add key="FTOS_
IntegrationProcessAuthUser" value="guest"/></pre>
</td></tr>
</table>

Setting	Description
	<add key="FTOS_IntegrationProcessAuthPassword" value="guest"/> If multiple credentials in the *web.config* file are applicable for the same API call, the first one will be applied.
additionalHeaders	Code for additional configurations to include in the HTTP request header.
StopOnError	When checked, if an error occurs during the API call, the remaining API calls in the sequence will not be executed. Otherwise, the External API will attempt to run the next API call in the sequence.

beforeJS	Custom JavaScript code to be executed before the API call. For details, see "External API Call – Custom JavaSript Reference" on page 179.
	
	IMPORTANT!
	When this page is saved/refreshed for the first time, the following code is automatically added in the beforeJS text box:
	
	`var instanceId = "$$instanceId$$";`
	`var integrationProcessDetailId = "$$integrationProcessDetailId$$";`
	`var contextEntityName = "$$contextEntityName$$";`
	`var contextUniqueId = "$$contextUniqueId$$";`
	`var runAsync = "$$runAsync$$";`
	`var requestParamsBeforeJs = {};`
	`requestParamsBeforeJs = $$requestParamsBeforeJs$$;`
	
	`/*set token example*/`
	`/*requestParamsBeforeJs["TOKEN"] = getAuthorizationTokenFromIntegrationProcessDetailId(instanceId, integrationProcessDetailId);*/`
	
	`////mandatory return object`
	`return {requestParams: requestParamsBeforeJs, skippedFromBeforeJs: false };`

Setting	Description

afterJS	Custom JavaScript code to be executed after the API call. For details, see "External API Call – Custom JavaSript Reference" on page 179.
	
	IMPORTANT!
	When this page is saved/refreshed for the first time, the following code is automatically added in the afterJS text box:
	
	`var responseAsString = "";`
	`responseAsString = $$responseAsString$$;`
	`var instanceId = "$$instanceId$$";`
	`var instanceDetailId = "$$instanceDetailId$$";`
	`var contextEntityName = "$$contextEntityName$$";`
	`var contextUniqueId = "$$contextUniqueId$$";`
	`var runAsync = "$$runAsync$$";`
	
	`var responseAsJson = JSON.parse (responseAsString);`
	`log("responseAsJson:"+ toJson (responseAsJson));`
	
	`/*save token example*/`
	`/*`
	`if(responseAsJson.TOKEN){`
	`update("FTOS_IntegrationProcessInstanceDetail", instanceDetailId, {authorizationToken: responseAsJson.TOKEN, authorizationTokenObtainedOn: responseAsJson.TOKEN_DATE});`
	`}else{`
	`resultAsjson.isSuccess = true;`
	`resultAsjson.message = "Nu a putut fi obtinut tokenul:" + (responseAsJson.errorMessage?responseAsJson.errorMessage:"null");`
	`}`
	`*/`
	`////mandatory return object`
	`var resultAsjson = {};`
	`resultAsjson.isSuccess = true;`

Setting	Description
	⚠ ``` resultAsjson.message = "!OK!"; return resultAsjson; ```
expiresInSeconds Multiplier	The number of seconds after which the authentication token expires.
secToWaitBeforeStart	Configures a delay in seconds before running the API call.
retryOnError	Check to resend the API request if the API call fails (the "resultAsjson" on page 179 object has the `isSuccess` key set to `false`). The number of retries is set by the "numberOfRetries" below setting.
numberOfRetries	Number of attempts to resend an API request if the API call fails (if the "retryOnError" above setting is enabled).
secToWaitBeforeRetry	Number of seconds to wait before resending an API request if the API call fails (if the "retryOnError" above setting is enabled).
Pass response in main result	Includes the API call's response in the response object of the External API. Otherwise, it will be available only within the External API's context in the `responseAsJson` variable in the "afterJS" on the previous page setting.
logRequest	Logs the call's HTTP request details in the FTOS_ `IntegrationProcessInstanceDetailLog` entity for debugging purposes.
logResponse	Logs the call's HTTP response details in the FTOS_ `IntegrationProcessInstanceDetailLog` entity for debugging purposes.
External API Detail Parameters	Defines the parameters that will be passed to the API endpoint. For details, see "External API Call – Parameters" on the next page.

External API Detail Dependencies	Defines the API calls that must be completed successfully prior to running the current API call.
	⚠ **IMPORTANT!**
	If the dependency is based on a security context where the dependee call must pass an authorization token to the current API call, check the `hasAuthorizationToken` setting and add the following code in the "beforeJS" on page 175 textbox:
	``
	requestParamsBeforeJs["TOKEN"] = getAuthorizationTokenFromIntegrationProcessDetailId(instanceId, integrationProcessDetailId);
	``

External API Call – Parameters

This screen configures each parameter that will be populated and passed by the API call to the REST API web service. Parameter settings must match the specifications of the called REST API.

Parameter Setting	Description
IntegrationProcessDetail	API call's code (see "Code" on page 171).
ParameterType	• Query – Parameter is appended to the API call's path. • Body – Parameter is passed in the HTTP request body. • Formdata – Parameter is passed using the multipart/form-data media type.
isMandatory	Parameter is mandatory.
isEnumList	Check if the Body type parameters are passed in the HTTP request in enumeration format (a single line JSON such as `{firstName: "John", lastName: "Doe"}`).
Name	Parameter name.
IsAuthentificationUser	Applicable for all non-Body parameters with "isEnumList" above not checked. **IMPORTANT!** Do not set user-id static parameters in this screen. See "httpAuthenticationType" on page 174 for details on how to configure authentication credentials.
IsAuthentificationPassword	Applicable for all non-Body parameters with "isEnumList" above not checked. **IMPORTANT!** Do not set password static parameters in this screen. See "httpAuthenticationType" on page 174 for details on how to configure authentication credentials.

External API Call – Custom JavaSript Reference

You can use the "beforeJS" on page 175 and "afterJS" on page 176 settings to set up custom JavaScript code to be executed before and after an API call. The following objects are available in this context:

Object	Description
instanceId	ID of the External API instance.
integrationProcessDetailId	ID of the API call.
contextEntityName	Name of the entity for which the External API is run.
contextUniqueId	ID of the "contextEntityName" above record.
runAsync	Either true or false depending if the External API is run asynchronously or not.
requestParamsBeforeJs	Includes static parameters and their values to be passed to the HTTP request.
requestParams	Parameter - value pairs to be passed to the API call. You can edit this object to include "requestParamsBeforeJs" above and/or dynamic parameters, such as an authentication token obtained in a prior step or values from the "contextEntityName" above record. **IMPORTANT!** This is a mandatory object that must be returned by the "beforeJS" on page 175 code.
skippedFromBeforeJs	If set to true, the API call is skipped. Otherwise, it should be set to false. **IMPORTANT!** This is a mandatory object that must be returned by the "beforeJS" on page 175 code.
responseAsString	HTTP response of the API call in string format.
resultAsjson	Indicates if the API call was successfull (`resultAsjson.isSuccess = true;`) and the response message (`resultAsjson.message = "!OK!";`). **IMPORTANT!** This is a mandatory object that must be returned by the "afterJS" on page 176 code.

How to call a External API

To call a External API, use the `FTOS_IntegrationProcessLibrary` object and `callIntegrationProcess` method:

```
var integrationProcessId = FTOS_
IntegrationProcessUtils.getIdFromCode("FTOS_IntegrationProcess",
"code", restPipeCode);
var ip = new FTOS_IntegrationProcessLibrary();
ip.logEnabled = false;
var response = ip.callIntegrationProcess(integrationProcessId,
contextEntityName, contextUniqueId, requestParams, runAsync);
```

where:

Parameter	Description
integrationProcessId	External API ID based on the pipe's code (see "How to create an External API" on page 170 for details).
contextEntityName	Name of the entity for which the External API is run.
contextUniqueId	ID of the "contextEntityName" above record.

requestParams	Includes static parameters and their values to be passed to each API call in the following format:
	`requestParams[External API code].[API call Order No.] = {[parameter 1]:[value 1], [parameter 2]:[value 2], ...};`
	For example:
	`requestParams["P01.01"] = {CUI: "36438401", CNP: "78787878"};`
	`requestParams["P01.02"] = {CUI: "36438401", CNP: "78787878"};`
	These parameter values will be available in the API call's custom JavaScript code in the "requestParamsBeforeJs" on the previous page object.

runAsync	If set to true, runs the External API as an asynchronous process.

Parameter	Description
response	The resulting response of the External API call will have the following format: `response:{ "ipInstanceId": "523ee20b-705c-45b3-b881-caeec1bde15e", "isSuccess": true, "errorMessage": null, "mainResponse": { "P04.02": { "requestId": 89696.0, "errorMessage": null, "IsSuccess": true } } }`

Example

```
var ip = new FTOS_IntegrationProcessLibrary();

var integrationProcessId = FTOS_
IntegrationProcessUtils.getIdFromCode("FTOS_
IntegrationProcess", "code", "P04");
var requestParams = {};
requestParams["P01.01"] = {CUI: "36438401", CNP:
"78787878"};
requestParams["P01.02"] = {CUI: "36438401", CNP:
"78787878"};

ip.logEnabled = true;
var response = ip.callIntegrationProcess
(integrationProcessId, "P666", "31a85d94-0c9c-429b-8bc6-
8c4a5e2d91a7", requestParams, false);
```

External API General Settings

Asynchronous Run

Asynchronous run of a External API is performed by the `FTOS_IntegrationProcess_JobServer_RunAsyncInstances` workflow. Use the `FTOS_IntegrationProcess_NumberOfAsyncInstancesToProcess` key in the

<appSetting> section of the *web.config* file to set up the number of asynchronous External API runs that can be queued.

Default Timeout

Use the FTOSIntegrationProcessDefaultTimeOut key in the <appSetting> section of the *web.config* file to set up the default timeout for calling the web service.

Data Model

The following entities are used to implement External API:

Entity	Description
FTOS_IntegrationProcess	External API definitions
FTOS_IntegrationProcessDetail	
FTOS_ IntegrationProcessDetailParameters	
FTOS_ IntegrationProcessDetailDependency	
FTOS_IntegrationProcessInstance	Data about External API instance runs.
FTOS_IntegrationProcessInstanceDetail	Data about External API' API calls.
FTOS_ IntegrationProcessInstanceDetailLog	Logs data about HTTP request and responses of the API calls.
FTOS_IntegrationProcess_AsyncProces	Data about asynchronous runs
FTOS_IntegrationProcess_ AsyncBatchProces	How many External API are asynchronously processed in real time.

Data Pipes

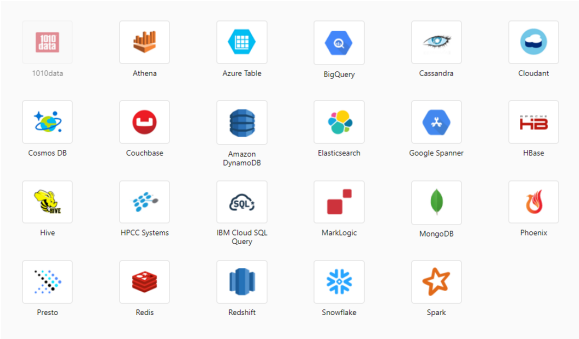
Data Pipes extract data from external data sources and replicate (load) it in the FintechOS database or other data management systems. By replicating and synchronizing data from outside sources, you can work with external data sets as if it they were native FintechOS database records.

NOTE
To have the Data Pipes working this key is added to web.config in Studio: <add

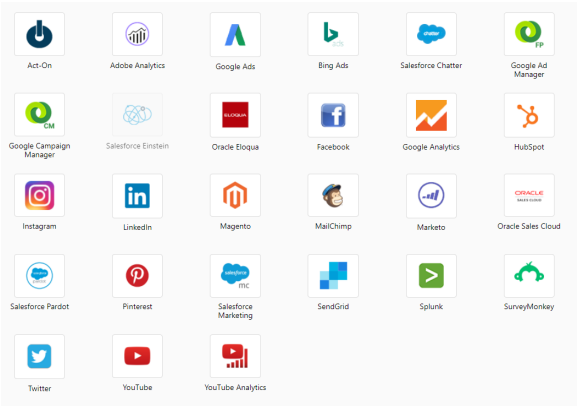
key="core-setting-cdata-proxy-url" value="http://192.168.30.18:7090" /> (value should be changed accordingly).

Data Pipes come with built-in connectors which allow you to replicate data from a variety of external data sources:

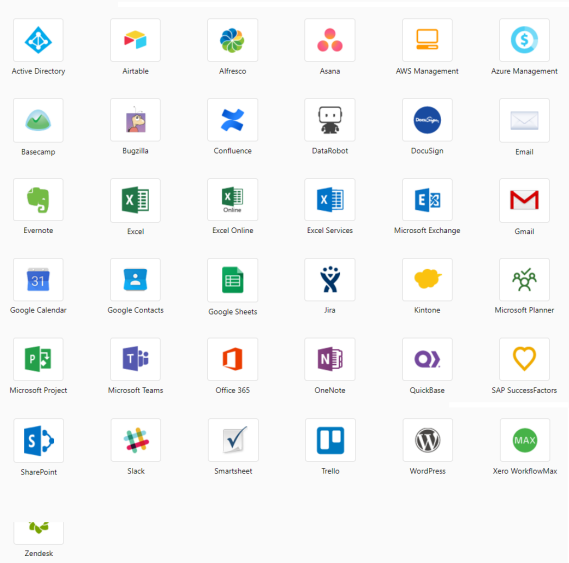
Big Data & NoSQL



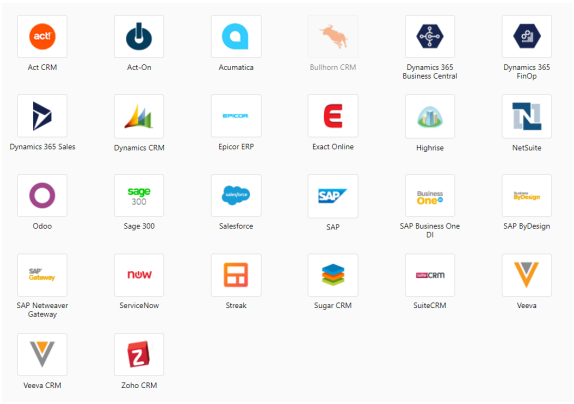
Marketing



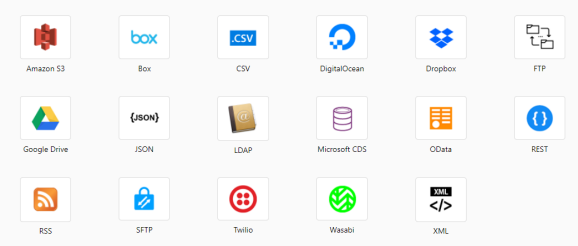
Collaboration



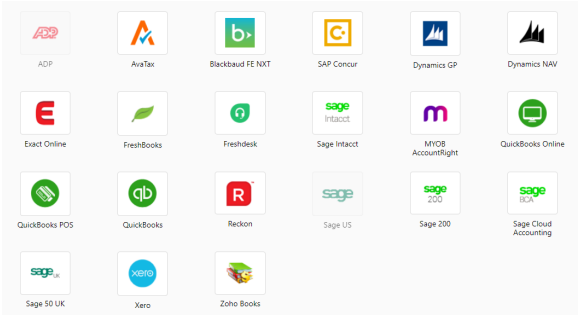
CRM & ERP



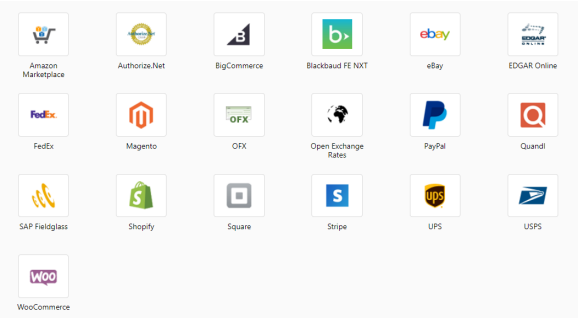
File & API



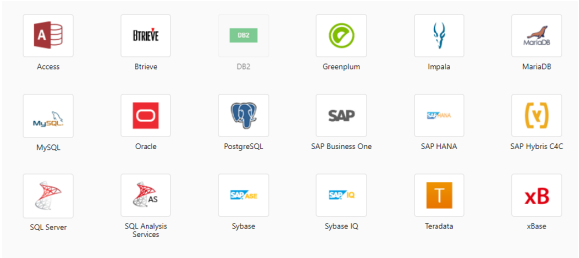
Accounting



E-Commerce



RDBMS



To replicate data from external data sources, follow the steps below:

Create the Destination Data Model 185
Set Up the Data Pipes Connections 189

Set Up the Data Pipes Replication Jobs 196

Create the Destination Data Model

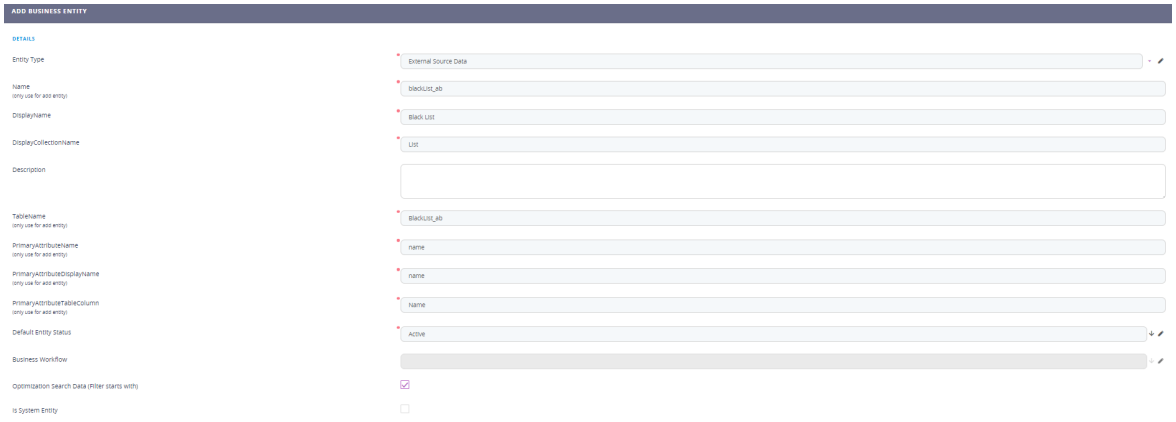
In order to replicate data from external data sources in FintechOS using Data Pipes, you need to create a data model that matches the data you will extract from the external system. To do that, make sure that you know how the external data is structured.

NOTE
The data replicated using Data Pipes is historical data (read-only data) and you cannot transform it. Use replicated data for analysis and reporting.
The only attributes that can be updated in a replica entity are the *entityStatusID* and *businessUnitId* system attributes.

To create a data model replica of the external data model, follow these steps:

Step 1. Create the destination entity which will store the replicated data

To create a data model which stores data replicated from an external system, in FintechOS, create an entity following the Creating Entities procedure. Make sure to select **External Data Source** in the Entity Type field. This entity will be the **destination** for your data replication.



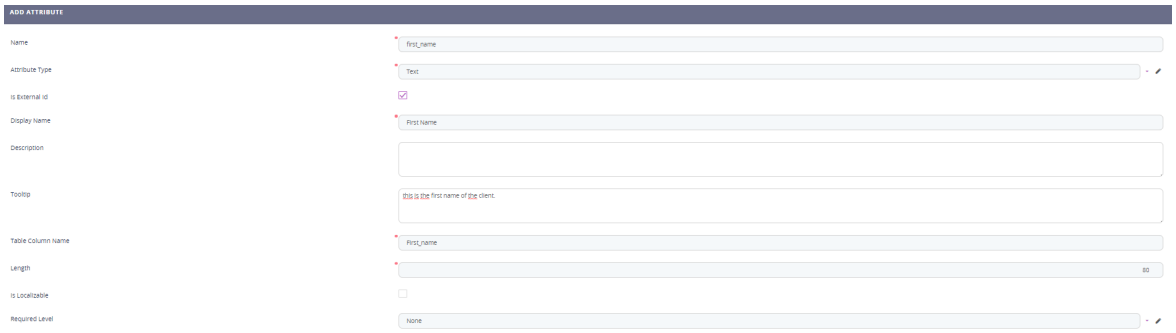
Step 2. Add attributes to the destination entity

Follow the Adding Attributes procedure to add all the necessary attributes matching the external data model to the destination entity (add an attribute for each column in the table you will extract from the source data system).

> **IMPORTANT!**
> Make sure that the attribute data types match the data types of the source table columns.

Replicating the source primary keys

If you want the data replication though dataPipes to be differential, your replication will have to include the primary keys from the source system. The destination attributes which will host the source primary keys must have the **Is External ID** checkbox selected.



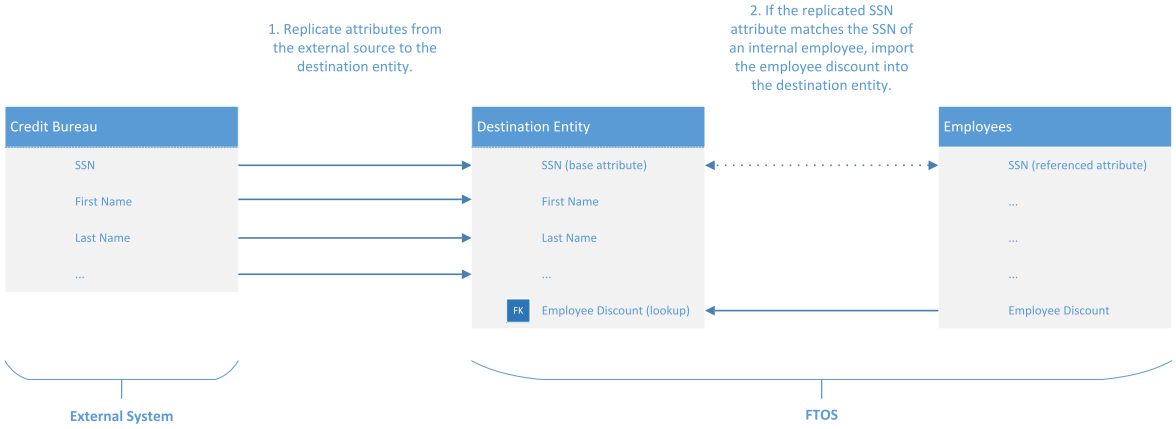
If multiple attributes have the checkbox ticked, they will form a composite external ID.

> **IMPORTANT!**
> You can change the attribute(s) which will form the external ID (tick/ untick the **Is External ID** checkbox on attributes) only before replicating external data. Once you run a replication job you can no longer change the external IDs.

Correlated lookup attributes

You can automatically populate specific lookup attributes based on the incoming data when a replication job is run. For instance, if you collect credit reference information from the Credit Bureau about credit applicants and you also provide a fee discount for

credits granted to your own employees, when collecting credit reference about an applicant that is registered as an employee, you can automatically populate the internal employee discount in the replicated record.



To create a correlated lookup attribute:

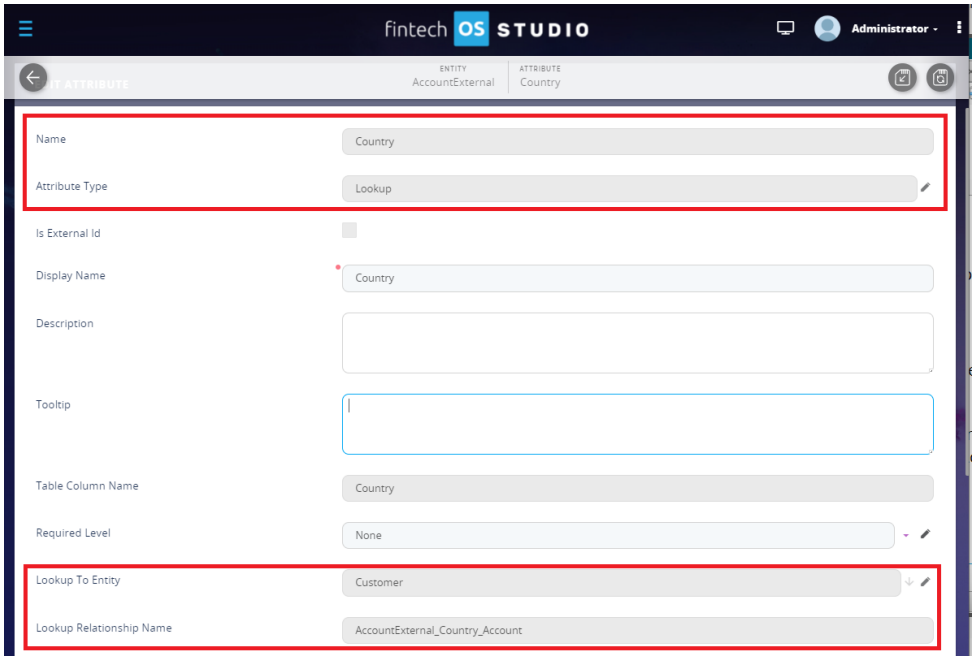
1. On the entity which will store replicated data, go to the Data Model and create the lookup attribute you want to correlate with an attribute from another FintechOS entity.
2. Click the **Save and Reload** icon.
3. Scroll down to the Lookup Correlation Attributes section and click the **Insert** button. The Add Lookup Correlation Attribute page appears.
4. In the **Base Attribute** field, select the matching attribute on the destination entity (current entity).
5. In the **Referenced Attribute** field, select the matching attribute on the entity referenced by the lookup attribute.
6. At the top right corner of the page, click the **Save and close** icon to save the correlation attribute. The record is displayed in the Lookup Correlation Attributes section.

After the data replication is complete (that is, a replication job is run), based on the defined correlations, the system will try to automatically populate the replicated entity's lookup attribute(s) based on the correlations you defined.

Example

In this example:

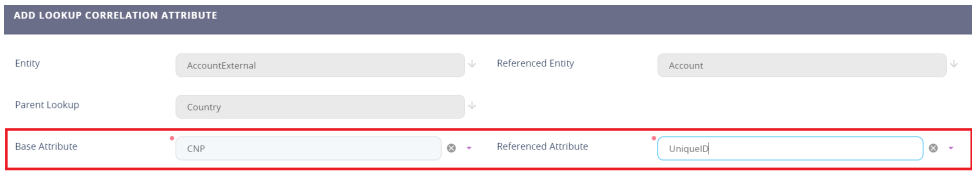
- **CNP** and **Country** are attributes replicated on the **Account_External** entity from an external data source. The Country attribute is a lookup to the FintechOS **Account** entity.



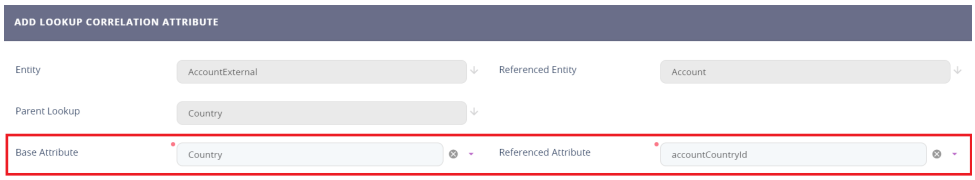
- In FintechOS, the **Account** entity has two attributes **UniqueID** (PIN as Display Name) and **accountCountryId** (Country as Display Name).

To correlate Account replicated data with FintechOS Account data by **CNP** and **Country**, we create two lookup correlations as follows:

- **CNP** with **UniqueID**



- **Country** with **accountCountryId**



After the data replication is complete, the system will try to automatically populate the Account ID field lookup to Account by CNP and Country.

Optionset attributes

If the attribute on the entity storing external data is of type optionset, after the replication is complete, the system will try to join the replicated value with the value of the optionset existing in FintechOS on the related entity.

Set Up the Data Pipes Connections

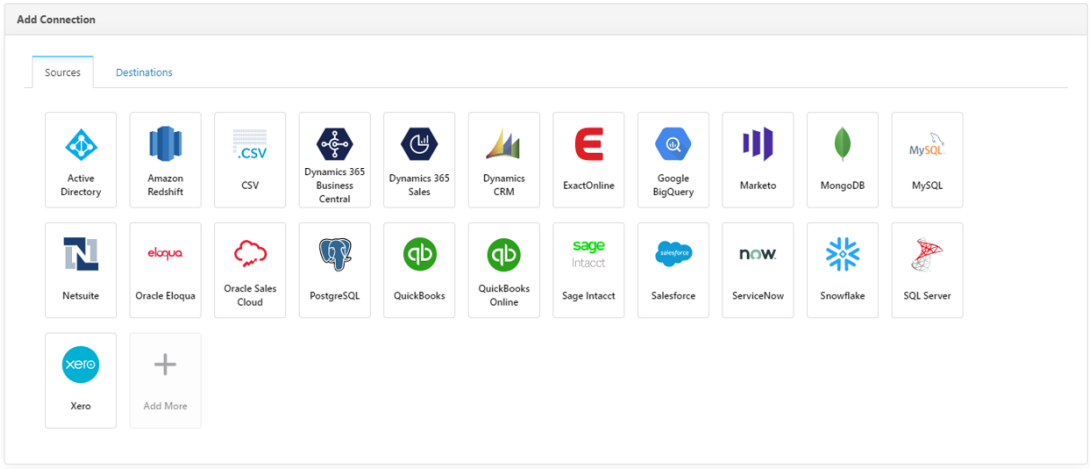
IMPORTANT!
Data Pipes connections require the CData Sync service to be installed and running on the FintechOS environment. For information on how to configure the CData Sync service, see the FintechOS Platform Administration Guide.

Data Pipes data replication jobs require a source - destination pair of connections. The source connection represents the external data source of the replicated data and the destination connection is where the replicated data is stored. In the case of FintechOS, the destination will be the underlying Microsoft SQL database server.

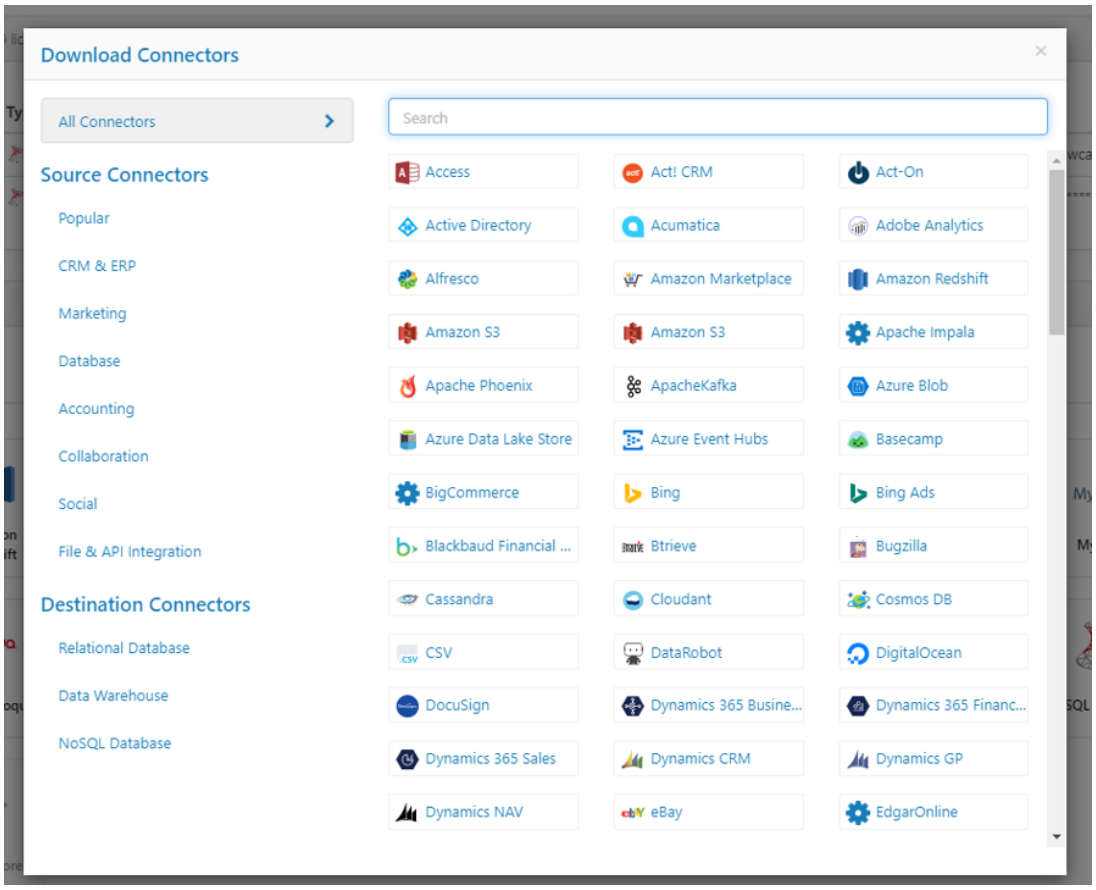
NOTE
When you open the Data connections page, you may have to enter the CData Sync username and password. If you don't know the CData Sync credentials, contact your administrator.

Step 1: Set up the source connection

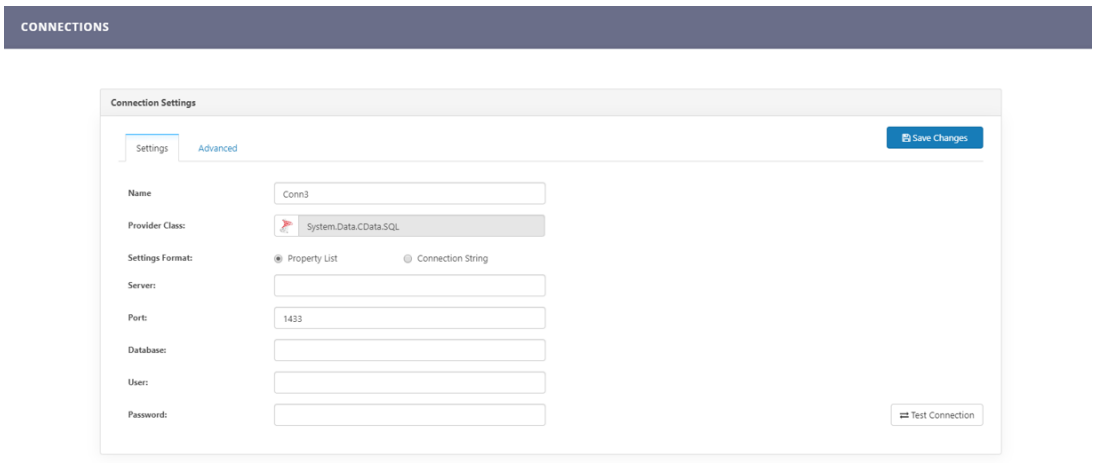
1. In Innovation Studio, on the main menu, click **Evolutive Data Core** > **Data Pipes** > **Connections**.
2. Scroll down to the **Add Connection** section, select the **Sources** tab, and select the connector that matches the source system for the desired data replication.



You can install additional connectors if you click on **Add More** and select the connector from Download Connector page.



3. Configure the connection settings specific to the selected connector. For details about each specific data source settings, see the CData Sync documentation.

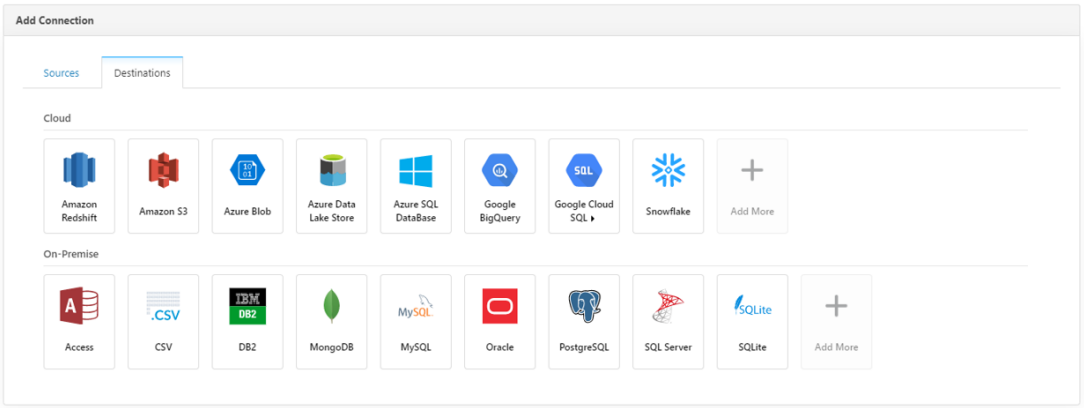


4. Click **Test Connection** to see if you properly configured the connection.

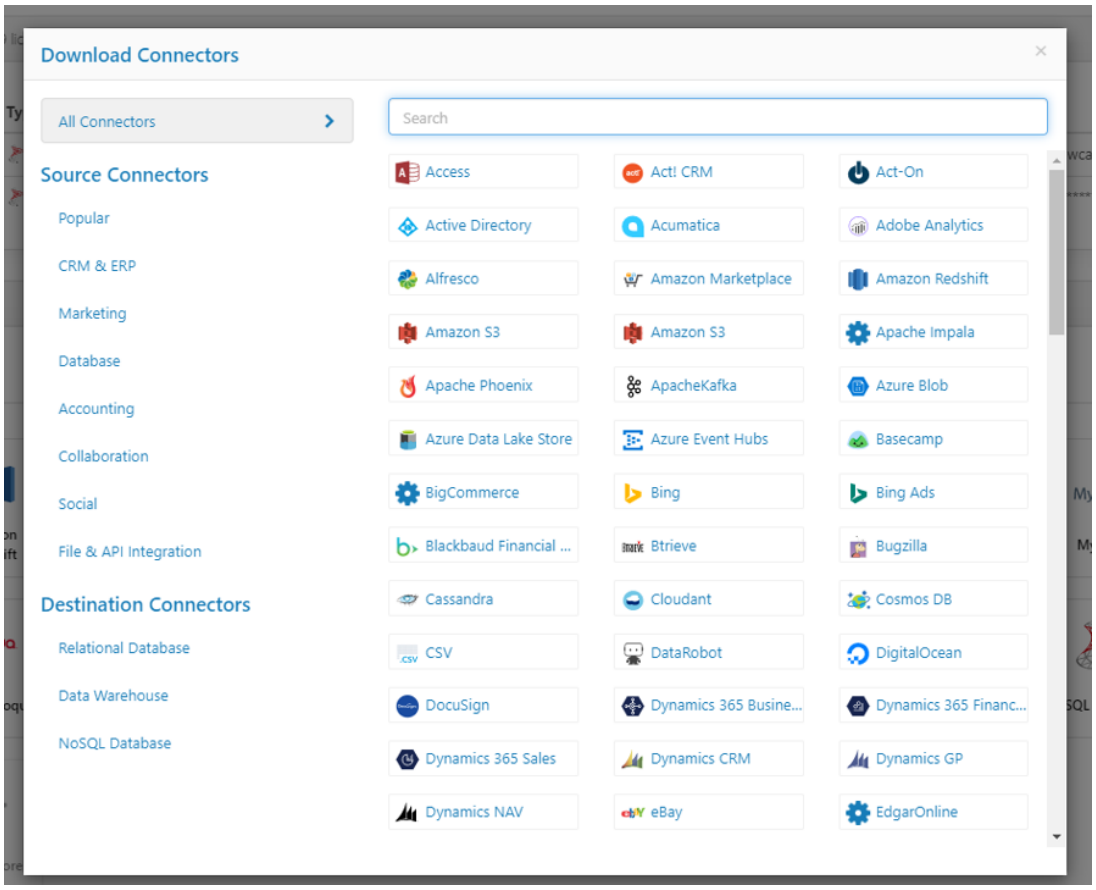
5. Click **Save changes** to save the data source connection.

Step 2: Set up the destination connection

1. In Innovation Studio, on the main menu, click **Evolutive Data Core** > **Data Pipes** > **Connections**.
2. Scroll down to the **Add Connection** section, select the **Destinations** tab, and select the connector that matches the destination system for the desired data replication. In the case of FintechOS, this will be SQL Server.



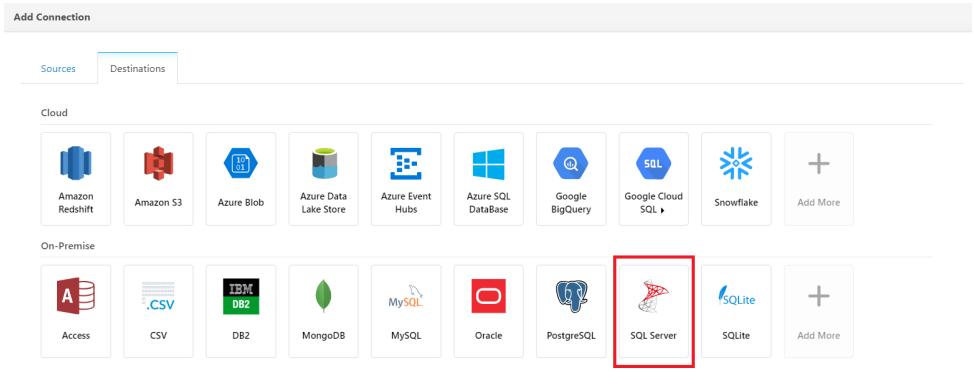
You can install additional connectors if you click on **Add More** and select the connector from Download Connector page.



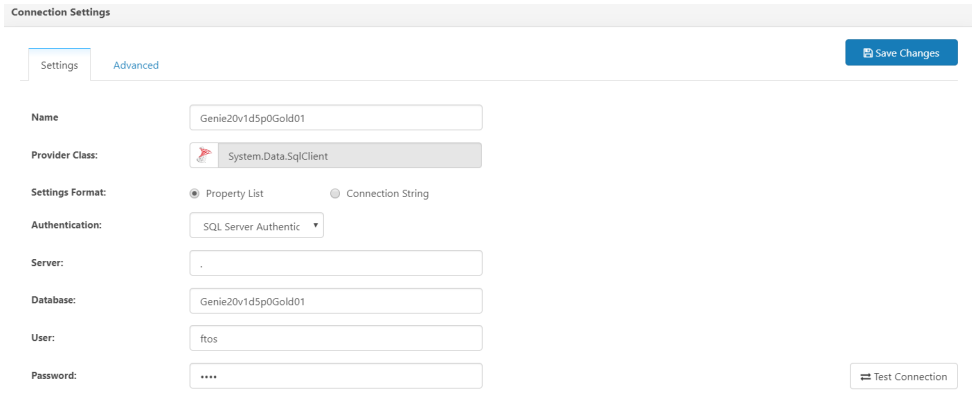
3. Configure the connection settings specific to the selected connector. For details about each specific data source settings, see the CData Sync documentation.
4. Click **Test Connection** to see if you properly configured the connection.
5. Click **Save changes** to save the data destination connection.

Example: How to configure the local FintechOS database as a destination connection

1. In Innovation Studio, on the main menu, click **Evolutive Data Core** > **Data Pipes** > **Connections**.
2. Scroll down to the **Add Connection** section, select the **Destinations** tab, and select the **SQL Server** connection.



3. Configure the connection settings specific to the FintechOS database server.



Setting	Description
Name	Enter a representative name for your connection.
Settings Format	If you prefer to enter all the connection settings using a single database connection string, select **Connection String**. Otherwise, leave it as **Property List**.
Authentication	Leave SQL Server Authentication.

Setting	Description
Server	Enter the name of the SQL Server running on your machine. **HINT** Entering a dot "." will default to your local SQL Server.
Database	Enter the database on the SQL Server used by FintechOS.
User	Enter a user account that has access to the above database. You can use the user account FintechOS is using to access the database.
Password	Enter the password for the above user account.

HINT

You can refer to the `<connectionStrings>` entry in the *Web.config* file to find out details about the FintechOS database connection.

```
<connectionStrings>
    <add
name="EbsSqlServer" connectionString="Data
Source=GenieDB;Initial
Catalog=Genie20v1d3b339;Integrated
Security=False;User
ID=ftos;Password=abcdefg"
 providerName="System.Data.SqlClient" />
</connectionStrings>
```

4. Click **Test Connection** to see if you properly configured the connection.
5. Click **Save changes** to save the data destination connection.

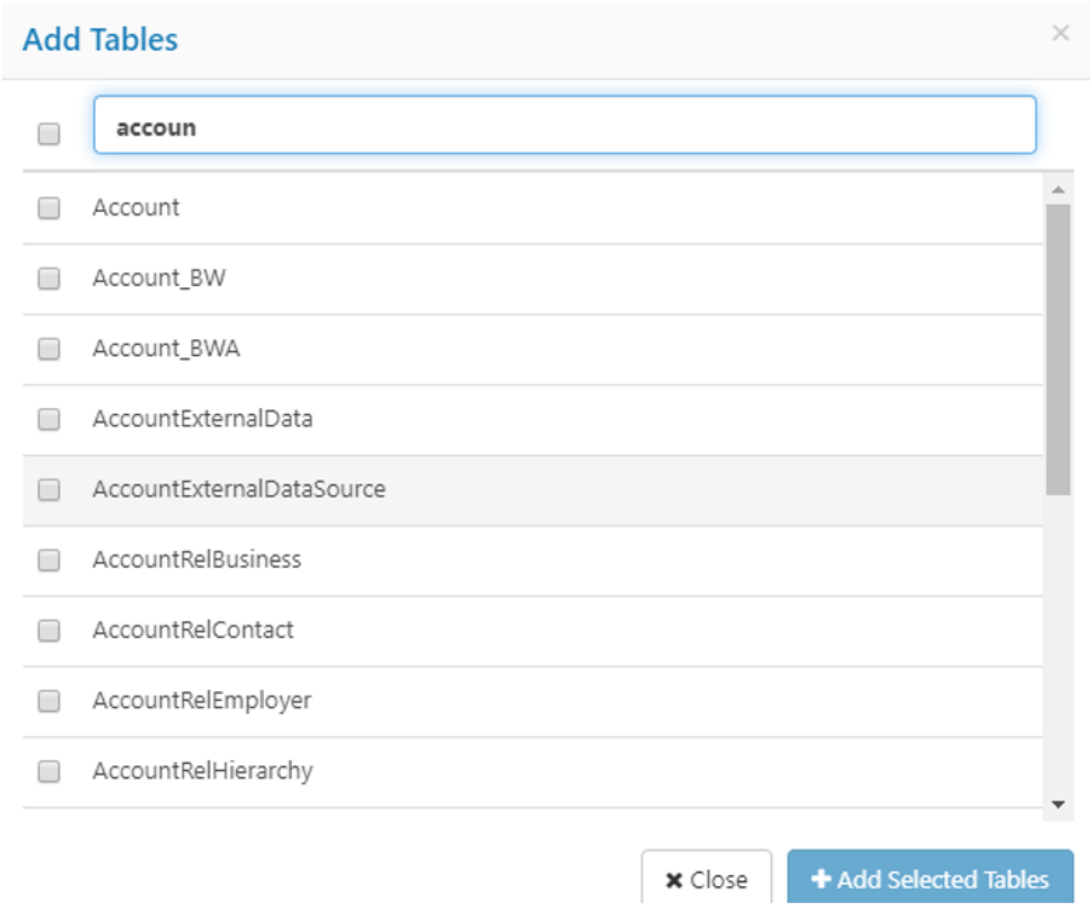
Set Up the Data Pipes Replication Jobs

IMPORTANT!
Data Pipes jobs require the CData Sync service to be installed and running on the FintechOS environment. For information on how to configure the CData Sync service, see the FintechOS Platform Administration Guide.

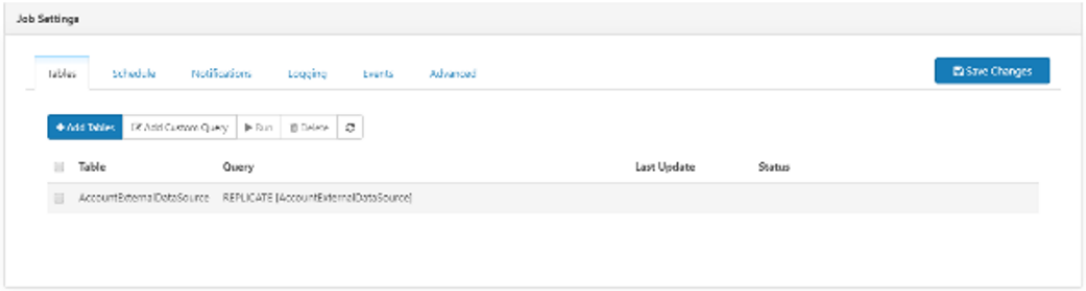
The active action of data replication between a source connection and a destination connection is called a **job**. To set up a job:

1. On the main menu, click **Evolutive Data Core** > **Data Pipes** > **Jobs**.
2. Click **Add Job**.
3. Name your job and select the **source** and **destination** connections (the ones you set up following the "Set Up the Data Pipes Connections" on page 189 procedure). You can reuse the same connections for multiple jobs.

4. Configure the replication job. To do so, from the Job Settings section click **Add Tables**

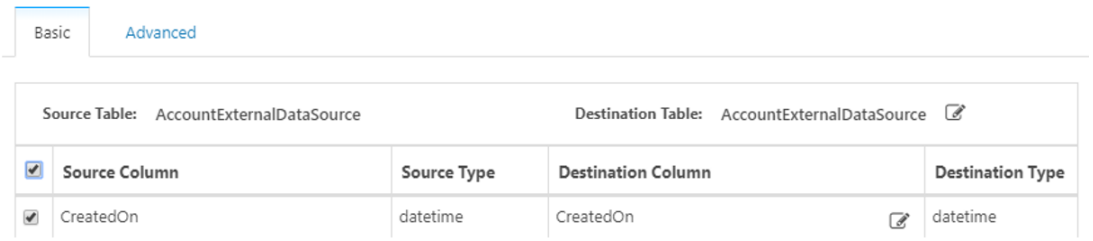


5. Select the source table(s) that you would like to replicate and click the **Add Selected Tables** button. The selected tables are added to the Job Settings (Tables tab).

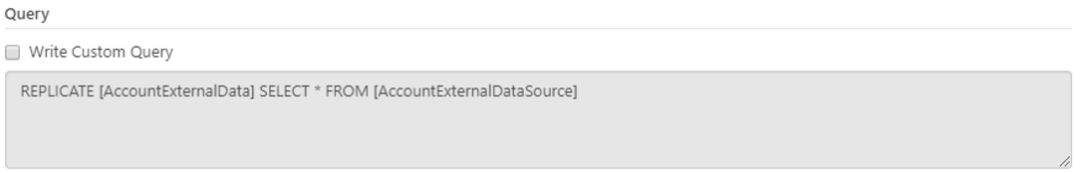


6. Click on the table entry and change the replication query. First select the destination table. To so, click on the edit button next to the **Destination table** and change its name

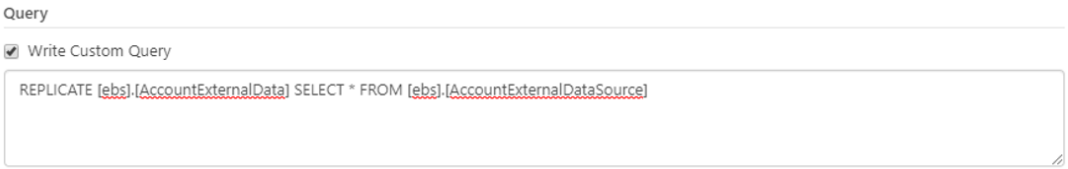
to the name of the destination table.



The figure below provides an example of how the query might look like:



7. If you need to specify a specific schema, tick the **Write Custom Query** checkbox and change the query to match your schema (the schema name for FintechOS tables is *ebs*).



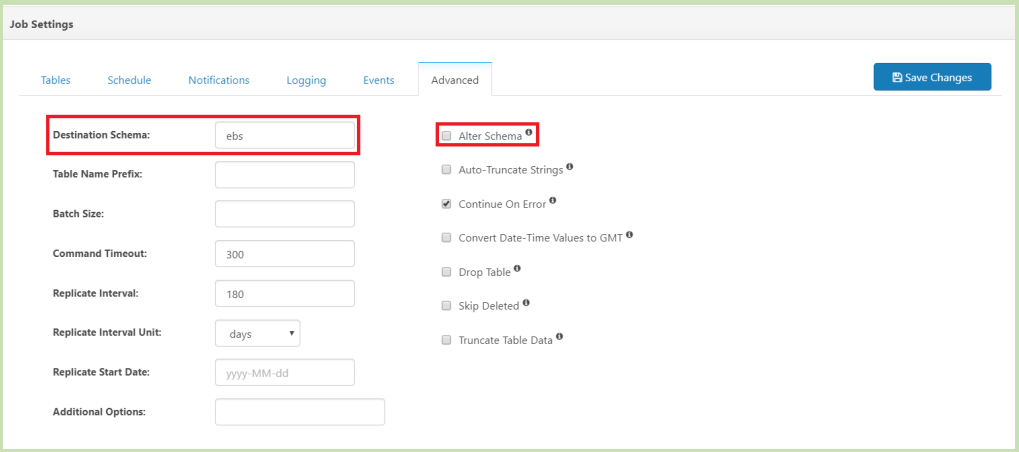
You can include multiple source - destination table pairs in the same replication job.

If you wish the replication job to update the existing data in the destination table prior to replication (for instance, if some records in the source system have been deleted, but they are still archived in the destination table), add a custom query at the beginning of the job with the following model:

```
CHECKCACHE [DestinationTable] AGAINST [SourceTable] WITH REPAIR
```

For detailed instructions on how to set up all the advanced job parameters, see the CData Sync documentation.

HINT
When replicating data into FintechOS, it is recommended to set the destination schema to *ebs* and to uncheck the **Alter Schema** checkbox in the job's Advanced tab.



This way, you will not have to manually specify the ebs schema when referring to destination internal FintechOS tables and the replication job will not alter your data model if there is a mismatch between the source table metadata and the destination table metadata.

Run Replication Jobs

You can run a job manually or you can schedule it to run at specific time intervals.

To run a specific data replication job, go to the list of jobs, select the job you want to run and click on the **Start** button. The external data is replicated in the FintechOS external data source entity or in the data storage system you selected as destination when setting up the connections.

The replicated data saved in FintechOS has the **IntegrationSystem** user ID and the **root** business unit.

To schedule a replication job, click the **Schedule** tab and set the time intervals when you want the job to run, then click **Save changes**.

Digital Journeys

Digital journeys are a visual representation of every experience (path) users might have when using an app. A digital journey is comprised of a set of steps and decision points which take the users throughout a process, carrying them from one step to another to achieve a goal such as an onboarding process, application for a home policy or a life insurance. FintechOS Digital Journey anatomy is comprised of Evolutive Data Core, Form Driven Flows, User Interface and Fintech Automation Processors. FintechOS digital journeys are founded on the Evolutive Data Core which organizes, manages and displays various data collected in a business process.

In Innovation Studio, you can define how users interact with the apps based on their needs and expectations, improving the customer journey and providing a positive customer experience.

There are two types of flows a user is able to create and use to build a digital journey with multiple flows:

- **Form driven flows** enable you to group business-wise information in a logical and comprehensive manner. They are defined on entity. Another type of flow is the Mock-up Flow, which is a preliminary version of a Form Driven Flow, useful for prototyping and defining the information needed.
- **Custom flows** enable you to create custom URLs which redirect the user to a specific data form or view, generate custom filtered views based on security roles or add buttons which trigger specific actions.

A digital journey can combine several flows: a customer flow and an operator flow, depending on the business need. For example, it is possible to build a SME current account onboarding, by creating first a mock-up flow turning it into a form driven flow and attaching an operator flow. This way a client will fill in his information and have a video call with an operator to confirm any further data.

For insurance, it is possible to build a journey for a client to apply for a motor policy by declaring information about this car and sign the contract in just a few minutes using Business Automation processors.

This section covers the following topics:

Form Driven Flows **203**
Creating Form Driven Flows 204
Adding and Configuring Steps 216
Custom Processor Step 223
Action Step 226
Flow Control 227
Configuring Field Options 234
Defining Form Actions 240
Defining Action Groups 245
Flow Map 248
Defining Filtered Fields 249
Header Items 255
Linking Labels to Attributes 257
Displaying View from Another Entity 258
Rendering Custom Data Extensions 262
Creating Custom Search Forms 264
Form Driven Mock-up Flows **267**
How to create a form driven mock-up flow 267
How to display a form driven mock-up flow 269
How to convert a form driven mock-up flow into a regular form driven flow 270
Custom Flows **271**
Differences between the Form Driven Flow, Custom Flow and Digital Journey. 272
Creating Custom Flows 272
Creating Custom Controls 278
Code Execution Sequence **281**
Entity Forms 281
Entity Form Steps 282
Form Driven Flows 283
Form Driven Flow Steps 284
Form Driven Flows (Wizard Mode) 285

Form Driven Flow Steps (Wizard Mode) 286
Custom Flows 288
Entity Views 289
Charts 291
Digital Journey Map 293
Adding a flow to the map 293
Editing a step 294
Digital Journey Context 295
Apply flow control rules only for specific digital journeys 296
UI Designer 297
STEP 1. Define the form layout 298
STEP 2. Add attributes 303
STEP 3. Configure and add relations 304
STEP 4. Working with Buttons 307
STEP 5. Access predefined HTML Templates 318
STEP 6. Add entity extension child collection support 320
Using Your Own Style Sheets 321
Create a New Style Sheet 321
Apply Style Sheets (No-Code) 322
Apply Style Sheets Using Code 323
Limit Style Impact to Current Form 324
Overwriting Variables 324
Localization 325
Viewing Defined Languages 325
Adding Languages 326
Localizing Generic Resources 329
Localizing Metadata 330
Localizing HTML Templates 332
Localizing Option Set Items 336
Localizing Views 336

Client-side Localization 337
Server-side localization 340
Code Snippets Support 341
Code Snippets Support for the HTML Editor 342
Code Snippets Support for JavaScript Text Boxes 345

Form Driven Flows

Form driven flows provide you with the mechanisms to make seamless user experience across the entire digital journey. An ordered collection of components which address an entire need of a digital actor. It is a part of the FintechOS Digital Journey anatomy.

If properly created, they provide you with opportunities to address both your team's and customers' pain points.

It is a low-code steady stream of steps that an user goes through to achieve a goal. The way the actor gets from one place to another in a steady stream, i.e. process steps. It is the starting point to build a complex digital journey with several flows, a multitude of steps, sometimes collecting data about a client or an insured item to satisfy a business need.

For example, by using a digital journey, a banking representative may want to add a new customer which becomes connected to data sources. The added customer is visible to a set of related applications that expand customer's business profile. The business profile is available for inspection and review as part of the customer information central hub.

This section covers the following topics:

Creating Form Driven Flows 204
Adding and Configuring Steps 216
Custom Processor Step 223
Action Step 226

Flow Control 227
Configuring Field Options 234
Defining Form Actions 240
Defining Action Groups 245
Flow Map 248
Defining Filtered Fields 249
Header Items 255
Linking Labels to Attributes 257
Displaying View from Another Entity 258
Rendering Custom Data Extensions 262
Creating Custom Search Forms 264

Creating Form Driven Flows

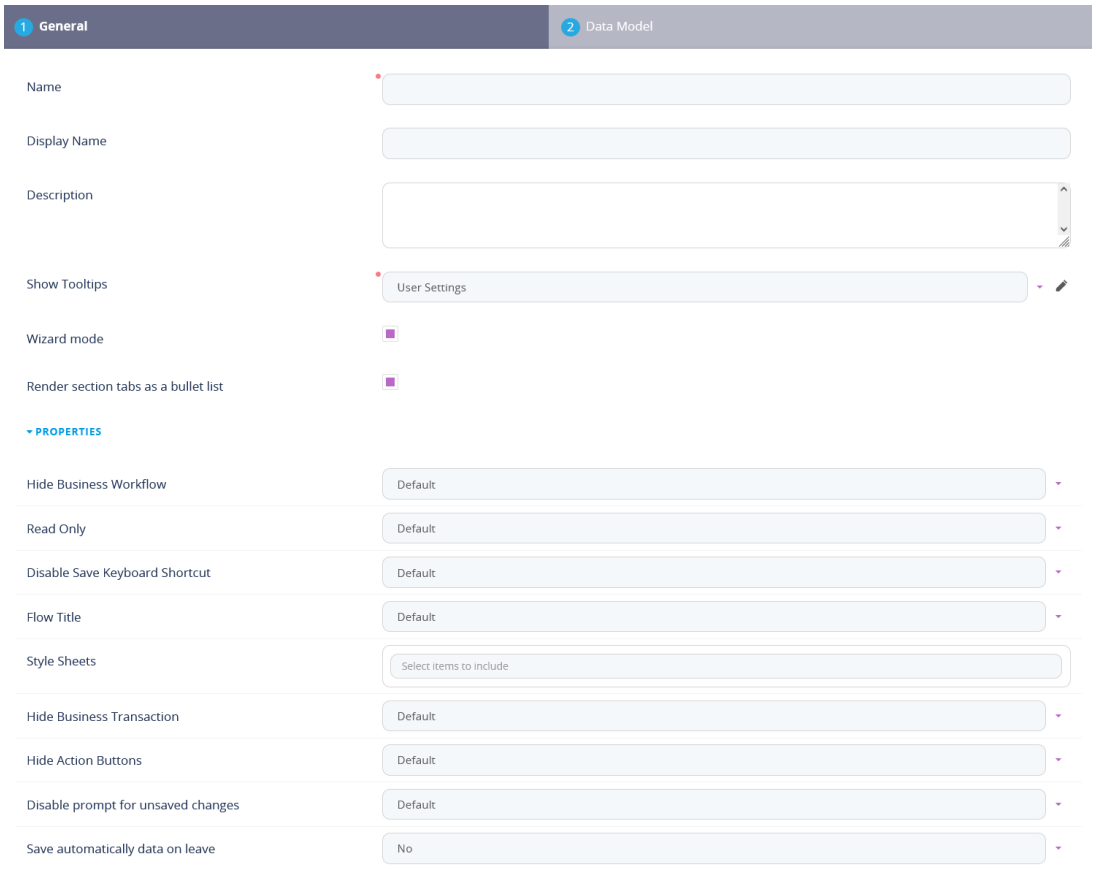
This section walks you through the steps that you need to follow in order to create a form driven flow.

Prerequisite

You need to have the flow's underlying data model defined (entity and attributes), including any required data extensions (presented in the Extend the Data Model).

1 Create the form driven flow

1. From the menu, click **Digital Journeys** > **Form Driven Flows**. The Form Driven Flows page appears.
2. At the top-left corner of the page, click the **Create** button. The data form driven configuration page appears which is comprised of two sections. It displays by default on the **General** section.

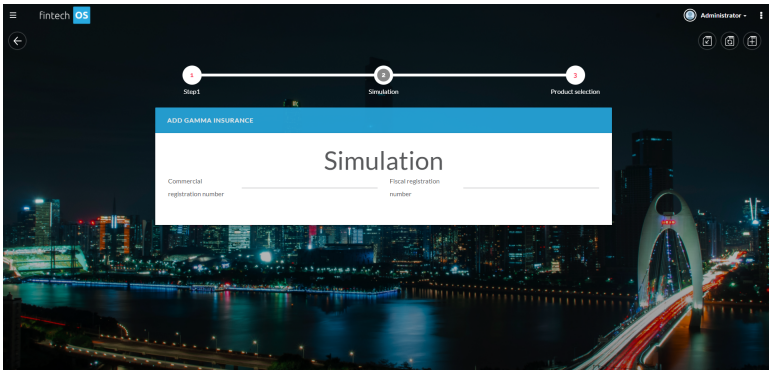


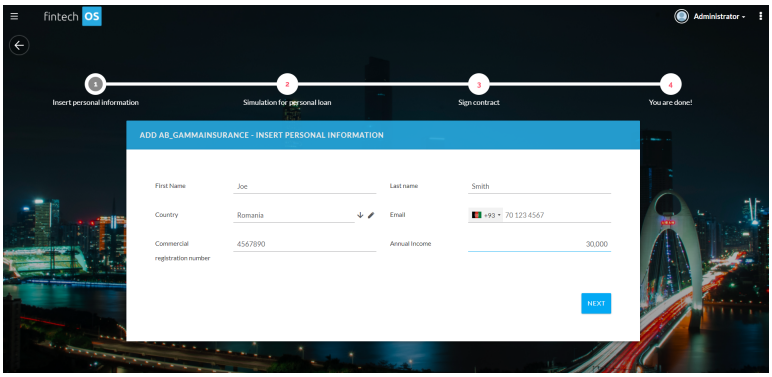
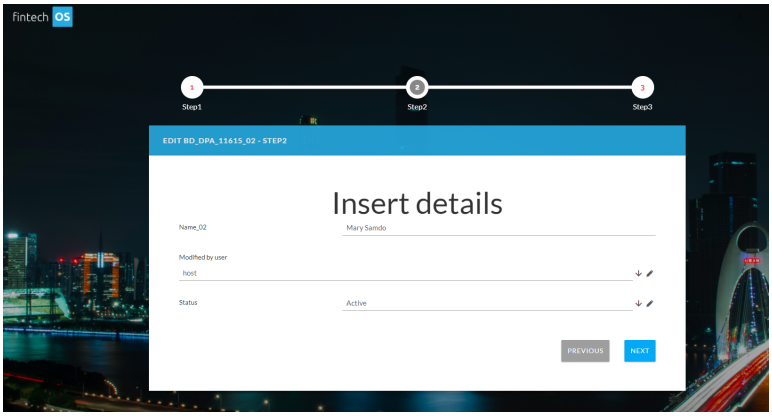
3. Provide the form driven flow general information:

Field	Description
Name	The name of the form driven flow used internally by the system. This field is mandatory. **NOTE** A naming convention is an important part in a well-built data model; therefore, we recommend you to use PascalCaseNames (upper camel). The Name starts with an uppercase letter, as do all additional words. For example: FTOS_CMB_BaseAccount.

Field	Description
Display Name	This is the name that will be shown in the user interface. Choose a suggestive name.
Description	Optional description of the form driven flow.
Show Tooltips	Select how you want the tooltips to be shown for specific attributes on journeys in the user interface. The following values are available: • User Settings (default value). Give users the possibility to show tooltips by toggling on / off the Tooltips button displayed at the top-right corner of the UI. • YES. Always show tooltips in the Portal UI the digital journeys when users hover their mouse on the attributes on fields which have tooltips. The users do not have the possibility to toggle the tooltips off. • No. Never show tooltips in the Portal UI on the digital journey.

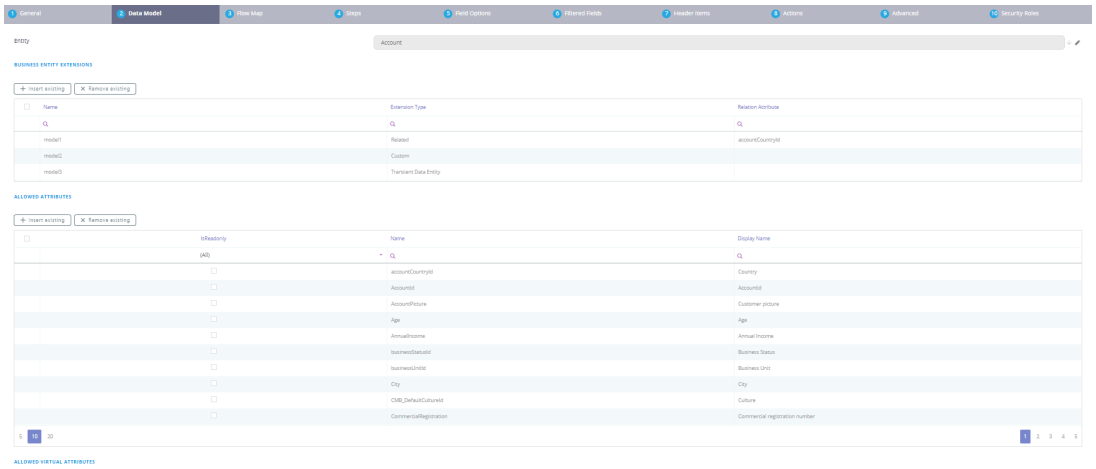
Field	Description
Wizard Mode	The steps of a digital user journey are by default displayed as tabs. If you tick **Wizard Mode**, the digital journey will have a sequenced design in the user interface, with navigation buttons (Next, Previous, Finish). If you select this field, you will be able to control the flow of the digital journey and also visualize the journey map (graphical representation of the steps). By enabling the wizard mode, the tabs "Flow Control" on page 227 and "Flow Map" on page 248 are displayed. ⚠ **IMPORTANT!** In a wizard-like digital journey, the data is saved into the database when navigating from a section to the next one, so be careful how you manage the mandatory fields. We recommend you to add these fields on the first step of the digital journey. You can customize the layout of the wizard: button labels, positioning an their colors. For more information, see Client SDK Reference. If the bool is true a new field will be displayed ''Show Bullets Progress Bar''.

Field	Description
Render section tabs as a bullet list	Not available in wizard mode. The steps of a digital user journey are by default displayed as tabs. If you tick this checkbox, the steps will be shown in the Digital Experience Portal as a bullet list and you can navigate between steps by clicking on the round numbered bullets. 
Hide Business Workflow	Hides the entity record's state and state transition options in the end-user interface. For more information about business workflows, see the Business Workflows Processor documentation.
Read Only	Prevents end-users from making changes to the displayed form fields.
Disable Save Keyboard Shortcut	Prevents end-users from saving and reloading the form by pressing the Ctrl+S keyboard shortcut.

Field	Description
Show Bullets Progress Bar	Wizard mode only. In addition to the Next and Previous buttons, the numbering of the steps appear on the progress bar. Choose one of the following: • Default. The progress bar is not shown. • Yes. The progress bar is shown and the navigation is done through the Next/ Previous buttons. • No. The progress bar is not shown. This is how it looks like in the Portal.  

Field	Description
Flow Title	Select a title from the list: • use display name (this name will appear everywhere, on every step this name will be exhibited) • show only step display name (each step will have its own name exhibited).
Style sheets	Select the style sheet you wish to use. Multiple style sheets can be added. The order of how they are added is respected in the execution. For details, see "Using Your Own Style Sheets" on page 321.
Hide Business Transaction	Hides the Business Transactions button at the top right corner of the screen, which displays the business workflow transitions of the record. For more information about business workflows, see the Business Workflows Processor documentation.
Hide Action Buttons	Hides the Actions button at the top right corner of the screen, which allows you to open action groups or reports associated with the form driven flow (for details, see "Defining Action Groups" on page 245 and "How to add Power BI Reports to Digital Journeys" on page 450).
Disable prompt for unsaved changes	When navigating away from the form driven flow, for instance by clicking the Go Back button in the top left corner of the screen, if the form contains unsaved data, the user will not be prompted to save changes.
Save automatically data on leave	When navigating away from the form driven flow, for instance by clicking the Go Back button in the top left corner of the screen, any unsaved data in the flow will be automatically saved.

4. Click the **Data Model** tab and from the **Entity** field, select the entity whose data model you'll be using.



5. At the top-right corner of the page, click the **Save and reload** icon. The digital journey is added into the system and the data form driven configuration page displays on the **Data Model** tab which has the Business Entities Extensions section expanded. For more information on how to add Business Entities Extensions, see Extend the Data Model.

   If you want to render data extensions on the digital journey, you need to first register them by adding the entity extension, For more information, see "Rendering Custom Data Extensions" on page 262.

6. For a specific digital journey, after selecting the entity, select the attributes from it that you wish to use in particular. Click the ''**Insert existing**'' button and select from the list the attributes.

7. To add virtual attributes, in the third grid of the Data Model, click the ''Insert pre-existing'' and tick the virtual attributes to be added to the form.

8. If you wish to restrict the form's access only to specific attributes in the data model, insert the corresponding values in the **Allowed Attributes** and **Allowed Virtual Attributes** sections. By default, all attributes in the data model are available to the form's context. Once you start populating the allowed attributes sections, the form will be restricted from accessing any attributes except the ones explicitly defined in these sections.

2 Set the journey default type

Click the **General** tab and set the form driven flow default type by ticking the appropriate checkbox:

- IsDefault to set the data form by default on Insert mode.
- Is Default For Edit to set the data form by default on Edit mode.

Each entity has a default auto-generated data form which contains all the attributes of the data model you're using. If you want to make changes to the default data form, leave the Auto Generate Template checkbox unticked, otherwise on entity updates (e.g., add a new data form attribute), the data form will be overwritten and all the updates will be lost.

You can set to automatically generate the data form template by ticking the Auto Generate Template checkbox and selecting from the Auto Generate Template Type field one of the options available: Inherit, 1 Column, 2 Columns, 3 Columns and 4 Columns.

If **inherit** is selected, the data form layout will inherit the value from the entity which is parent for the current entity.

For backwards compatibility, a default auto-generate data form template is available at application level in the web.config file, 1-column data form template.

NOTE
If the checkbox is selected, on data form save, the existing data form template will be overwritten with the auto-generated one.

3 Design the flow UI

Click the **UI** tab . You can design a complex form driven flow layout by providing the HMTL code or you can create a classic data form layout by using the HTML elements available on the toolbar of the HTML editor. You can also create the HMTL template of a form driven flow by using the Advanced Code Editor or the UI Designer.

The form driven flow UI template supports the following tokens:

Token	Description
{AttributeName}	Displays the corresponding field on the data form. **NOTE** The attribute name must be included between curly brackets; otherwise, a simple text will be displayed on the page instead of the actual field.
{#RelationshipName, view: viewName#}	Generates a view provided by relationship and by view. The viewName is optional and specifies which view to generate. If viewName is not provided, the default view will be displayed on the data form.
{#RelationshipName, view: viewName, editmode:cell#}	Generates a view provided by relationship and by view. This view allows inline editing meaning that you can edit cells one by one directly in the grid, without opening specific records. **NOTE** In order to activate inline editing for a specific cell, you must select the **Allow Editing** checkbox displayed on the entity view column (View > View columns).
{#MKT_CampaignResponse_ MKT_ Campaign,nodelete,noinsert#}	Generates a view provided by relationship and by view, but the **Delete** and **Insert** buttons are not displayed on the view. You can apply the same logic (similar tokens) for the Export (noexport) or Refresh (norefresh) if you no longer need them on the view.
{$ChartName$}	Generates a chart based on the provided chart name.
{? entityName, view: viewName ?}	Allows you to display a view from another entity. For information on how to use it, see Display View from Another Entity.

In the HTML template, you can link HTML elements (labels) to attributes. For information on how to do it, see Link Labels to Attributes.

When designing the UI template of the form driven flow, you can also add custom buttons. For more information, see STEP 5. Working with Buttons.

4 Group information in steps

Grouping entity information in sections, herein referred to as steps, based on specific criteria (business, operational or other relevant to you) is useful especially in complex financial activities when you have to display a lot of information on digital journeys. You can add as many steps as you need in the Steps section (tab) based on your criteria.

If you only need one section on your journey, add it to the Steps section; it will not be marked as step in the Digital Experience Portal.

To avoid issues with the steps loading order, we recommend you to define functions in the journey and only call them within the steps. Rendering a step automatically loads, the main journey is also rendered.

> **NOTE**
> Before changing the behavior of an element existing on a step by using logic in a different step, make sure that both steps were previously rendered.

For information on how to add and configure steps, see Adding and Configuring Steps.

The default execution order of a digital journey which is comprised of steps is given by the OrderIndex of the steps as set on the digital journey > **Steps** tab. You can change the default flow of a digital journey by using the Flow Control feature.

5 Define who has access to the flow

If your business case requires that the form driven flow is available to designated roles within your organization, click the Security Roles tab and add the security roles who should have access to them. If no security roles are added here, all users will be able to view the journey.

6 Save the flow

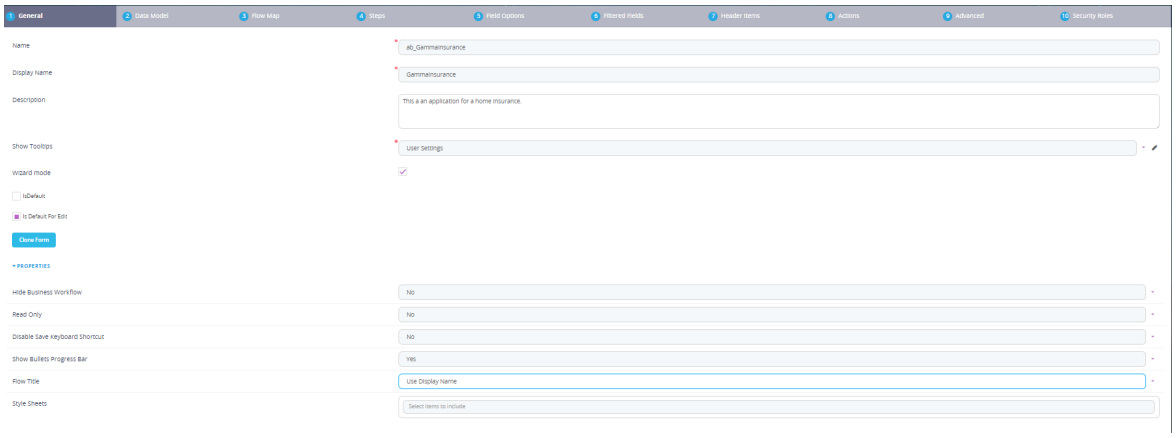
If you want to save and close the flow, at the top-right corner of the page click the Save and close icon.

If you want to save the flow and continue working on it, click the Save and reload icon.

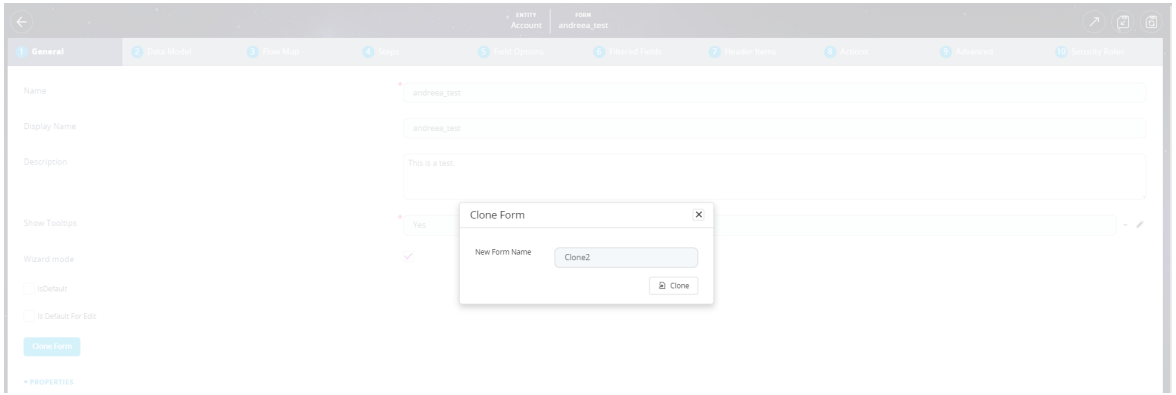
For detailed procedures on how to do extensive configuration of data form driven flows, refer to the Related Topics.

Cloning a form driven flow

To duplicate a form driven flow with all of its configurations, in the General tab, click the **Clone Form** button.



Insert a name for it and click ''**Clone**''. The new form driven flow will open immediately for editing.



NOTE

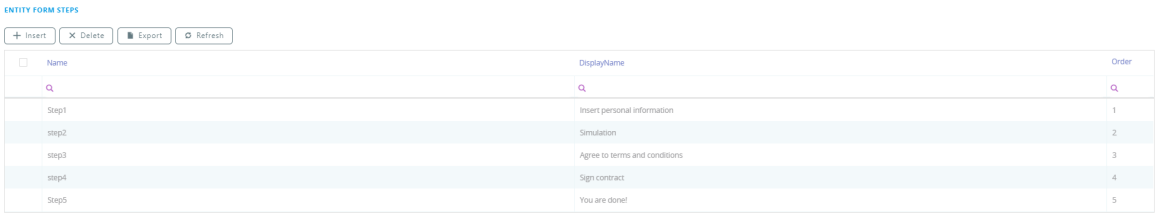
The data set behind it is not cloned, simply the form and its settings.
The Is Default and Is Default For Edit settings are not replicated in the cloned form driven flow, since you can have only one default form driven flow per entity.

Adding and Configuring Steps

Grouping entity information in steps based on specific criteria (business, operational or other relevant to you) is useful especially in complex financial activities when you have to display a lot of information on the entity forms and user journeys.

This sections walks you through the configurations that you need to follow to add steps and configure them.

STEP 1. Add step



1. On the configuration page of the form driven flow whose information you want to group in steps, click the **Steps** tab.
2. At the top of the **Entity Form Steps** section, click the Insert button. The step configuration page appears, displaying only the general tab.
3. Provide the following fields:

Property	Description
Name	The name of the step that will be used by system. **NOTE**A naming convention is an important part in a well-built data model; therefore, we recommend you to use PascalCaseNames (upper camel). The Name starts with an uppercase letter, as do all additional words.

Property	Description
Display Name	The name of the step. It is displayed in the Digital Experience Portal.
Step Type	Select either: • default. It is the basic type of ordinary step. • processor. For more information, see "Custom Processor Step" on page 223. • action. For more information, see "Action Step" on page 226.

4. At the top-right corner of the page, click the Save and reload icon. The step configuration page appears, containing more tabs, displayed by default on the General tab. You can configure the step, by clicking the tabs and making the desired settings.

STEP 2. Design the step layout

Click the UI tab . You can design a complex form driven flow layout by providing the HMTL code or you can create a classic data form layout by using the HTML elements available on the toolbar of the HTML editor.

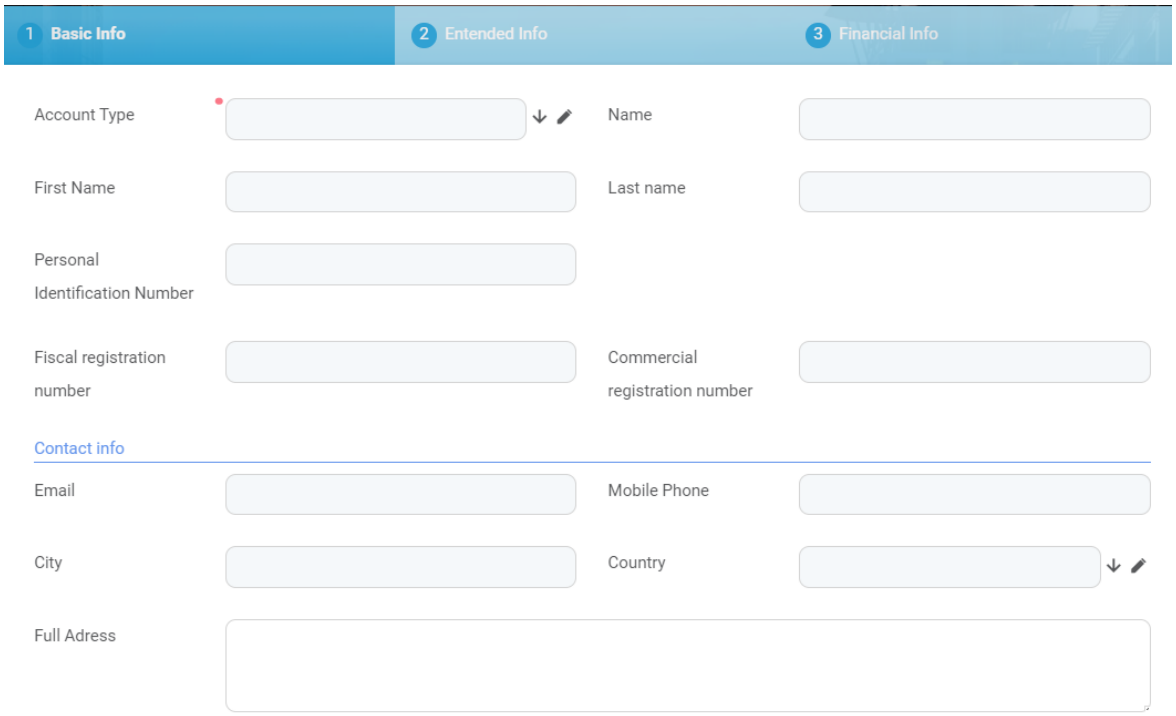
You can also create the HMTL template of a step by using the Code Editor or the UI Designer.

The step UI template supports the following tokens:

Token	Description
{AttributeName}	Displays the corresponding field on the step. **NOTE** The attribute name must be included between curly brackets; otherwise, a simple text will be displayed on the page instead of the actual field.

Token	Description
{#RelationshipName, view: viewName#}	Generates a view provided by relationship and by view. The viewName is optional and specifies which view to generate. If viewName is not provided, the default view will be displayed on the data form.
{#RelationshipName, view: viewName, editmode:cell#}	Generates a view provided by relationship and by view. This view allows inline editing meaning that you can edit cells one by one directly in the grid, without opening specific records. **NOTE** In order to activate inline editing for a specific cell, you must tick the **Allow Editing** checkbox displayed on the entity view column (View > View columns).
{#MKT_CampaignResponse_ MKT_ Campaign,nodelete,noinsert#}	Generates a view provided by relationship and by view, but the **Delete** and **Insert** buttons are not displayed on the view. You can apply the same logic (similar tokens) for the Export (noexport) or Refresh (norefresh) if you no longer need them on the view.
{$ChartName$}	Generates a chart based on the provided chart name.
{? entityName, view: viewName ?}	Allows you to display a view from another entity. For information on how to use it, see Display View from Another Entity.

This is how a step might look like in the Portal UI:



In the HTML template, you can link HTML elements (labels) to attributes. For information on how to do it, see Link Labels to Attributes.

When designing the UI template of a step, you can also add custom buttons. For more information, see STEP 5. Working with Buttons.

STEP 3. Flow Control

For details on how to configure the step within the digital journey's flow, see "Flow Control" on page 227.

STEP 4. Provide the code to be executed after the step is generated (optional)

Click the **Advanced** tab and, in the **After Events** field, provide the code to be executed after the step is generated (opened).

In the upper-right corner of the page, click the **Save and Close** icon. The new section will be displayed in the Entity Form Sections list.

NOTE
Variables and functions declared in other steps or in the form driven flow's Advanced section are not visible in the current step.
To access such variables and functions, use the formData.formScope object.

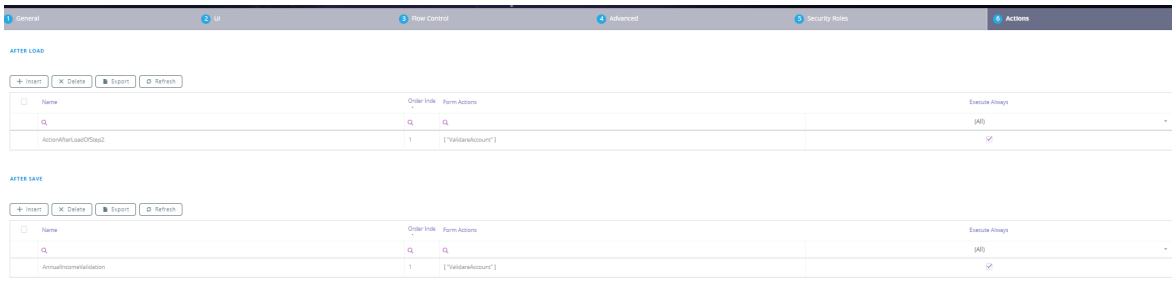
STEP 5. Define who has access to the step

If your business case requires that specific steps are available to designated roles within your organization, click the Security Roles tab and add the security roles who should have access to them. If no security roles are added here, all users will be able to view the section.

STEP 6. Actions

After having added an action in "Flow Control" on page 227, a user is able to set multiple formActions that automatically will be triggered after save, before navigation, if a specified condition (similiar to navigation rule) is true.

There are two grids displayed: **After Load** and **After Save**.

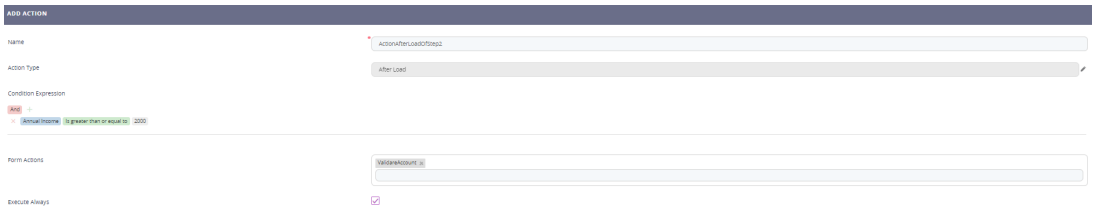


After Load

1. Click the ''Insert'' button to add a new action.
2. Fill in the following:

Field	Required	Data type	Description
Name	Yes	Text	Insert a name for the After Load action.

Field	Required	Data type	Description
Action type	No	Option set	This is automatically filled in. It is the moment of navigation. It correspondents to the grid.
Condition expression	No		Insert an expression. For details, see "Flow Control" on page 227.
Form Action	No	Option set	Select one action configured in the "Defining Form Actions" on page 240: • Change Business Status from Status. For details, see Business Workflow. • Generate Digital Document. For details, see Digital Documents Processor. • Call Custom Processor. • Call Business Matrix. For details, see Business Decisions. • Call Formula with data mapping. For details, see "Formula Parameter Mapping" on page 431.
Execute always	No	Bool	If this bool is true, the action will be executed every time it is triggered.



3. Click the ''Save and Close'' button.

After Save

1. Click the ''Insert'' button to add a new action.
2. Fill in the following:

Field	Required	Data type	Description
Name	Yes	Text	Insert a name for the After Save action.
Action type	No	Option set	This is automatically filled in. It is the moment of navigation. It correspondents to the grid.
Condition expression	No		Insert an expression. For details, see "Flow Control" on page 227.

Field	Required	Data type	Description
Form Action	No	Option set	Select one action configured in the "Defining Form Actions" on page 240: • Change Business Status from Status • Generate Digital Document • Call Custom Processor • Call Business Matrix • Call Formula with data mapping.
Execute always	No	Bool	If this bool is true, the action will be executed every time it is triggered.

3. Click the ''Save and Close'' button.

STEP 7. Save the step

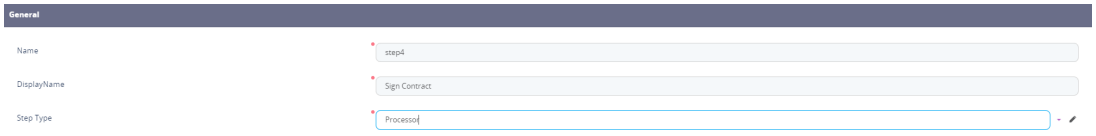
At the top-right corner of the page, click the Save and close icon.

Add as many steps as you need, then either continue with the journey configuration or save the changes by clicking the Save and close icon.

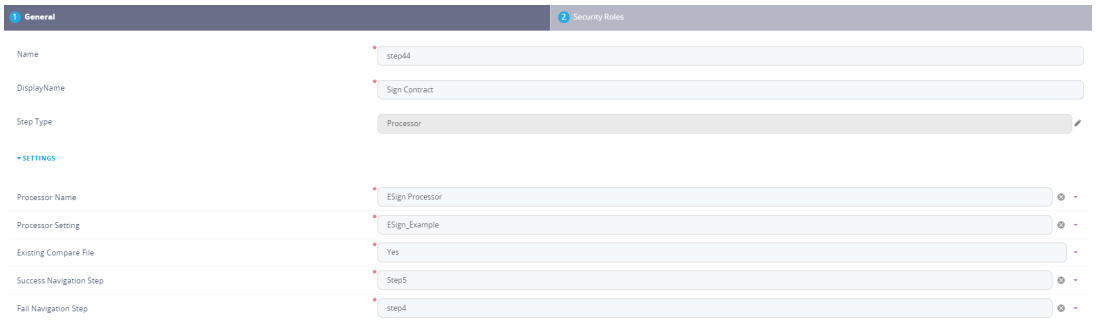
Custom Processor Step

This feature makes it possible for a user to add an automation processor such as Computer Vision or eSign or Video Streaming Processor or Face Recognition to a step without writing code. This functionality aids the user to rapidly build a fully automated digital journey with multiple steps and to easily call processors to the flow. By doing so, the time is halved and the navigation between steps is clear to the building of a form driven flow. To do so:

1. Open the Innovation Studio, open the main menu, select Digital Journey, click on the Form Driven Flow.
2. Read the list and select the Form Driven Flow you wish to edit by adding a processor to one of its steps.
3. After opening the form driven flow, click on ''4. Steps'' to add a new step. For more information, see "Adding and Configuring Steps" on page 216.
4. Click on ''Insert'' and fill in the name of the step, display name and for Type from the Option set select the **Processor**.



5. Click the ''Save and reload'' button. The general page will open.
6. Fill in the fields:



Fields	Required	Data type	Description
Name	Yes	Text	It is automatically filled. It is the name of the step.
Display name	Yes	Text	It is automatically filled. It is the name of the step.

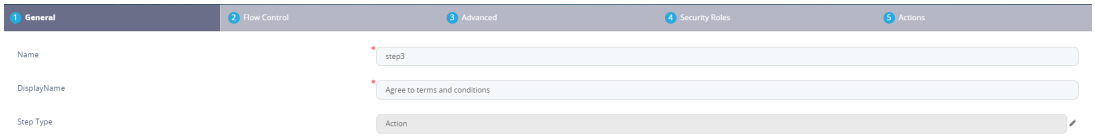
Fields	Required	Data type	Description
Step Type	Yes	Option set	It is automatically filled. It is the name of the step.
Processor name	Yes	Option set	Select the name of the processor.
Processor settings	Yes	Option set	Select the settings you configured earlier for the processor.
Existing Compare file	Yes	Option set	When using the Face recognition or Face recognition with Liveness, this field enables the system to compare the selfie with the photo for an ID.
Success navigation step	Yes	Option set	Select the step you wish to be next if the processor step is a success.
Fail navigation step	Yes	Option set	Select the step you wish to be next if the processor step is a failure.

7. Click the ''**Save and reload**'' button.
8. Add the proper security roles for this step by clicking the ''**Insert**'' button and choosing one of the security roles. For more details, see "Creating Security Roles" on page 622. Click the ''**Save and close**'' button.
9. Repeat for as many steps as needed.

Action Step

This feature makes it possible to trigger an action created in "Defining Form Actions" on page 240 and adding it to its own step. It looks like the default step without an UI.

1. Open the Innovation Studio, open the main menu, select Digital Journey, click on the Form Driven Flow.
2. Read the list and select the Form Driven Flow you wish to edit by adding an action to one of its steps.
3. After opening the form driven flow, click on ''**Steps**'' to add a new step. For more information, see "Adding and Configuring Steps" on page 216.
4. Click on ''**Insert**'' and fill in the name of the step, display name and for Type from the Option set select the **Action**.



5. Click the ''**Save and reload**'' button. The **General** page will open.
6. Fill in the fields to configure the "Flow Control" on the next page, Advanced, "Creating Security Roles" on page 622 and "Defining Form Actions" on page 240. They behave as for a default step.
7. Click the ''**Save and reload**'' button.
8. Add the proper security roles for this step by clicking the ''**Insert**'' button and choosing one of the security roles. For more details, see "Creating Security Roles" on page 622. Click the ''**Save and close**'' button.
9. Repeat for as many steps as needed.

Flow Control

The Flow Control feature enables you to create the path through the Form Driven Flow or Digital Journey for customers and Portal users.

> ⚠ **IMPORTANT!**
> The flow control is available only for Form Driven Flows of type wizard.

Before setting the flow control of a Form Driven Flow, you need to understand how it is working.

For a form driven flow comprised of steps, the execution order is the following:

1. The main Form Drievn Flow is executed. First the beforegenerate.js of the digital journey is executed and then the aftergenerate.js.
2. For each step of the Form Driven Flow when activated: step elements (HTML, CSS) are loaded and the Form Driven Flow is also rendered. Then aftergenerate.js of the step is executed.

Control Digital Journey Flow

The execution order of a digital journey which is comprised of steps is given by the OrderIndex of the steps as set on the digital journey > **Steps** tab, if not otherwise specified on the steps.

There are two ways in which you can change the execution order of a digital journey . On step you can:

- override the default next step.
- set a specific rule and choose the next step or cancel the transition and display error message.

When clicking **Next** in the digital journey, the flow is:

1. BeforeSave on the step is executed.
2. The conditions of type Cancel Navigation are evaluated and only the first one is executed.
3. The data is saved.
4. The conditions of type NavigateToSection are evaluated and only the first one is executed. It changes the default nextStep given by the OrdexIndex with the step specified on step.
5. The AfterSave on the step is executed and the digital journey takes the user to the next step .

Overriding the default next step set trough OrderIndex

To override the default next step set though the digital journey order index, go to the step from which the transition will be done, select the next step using the Option set.

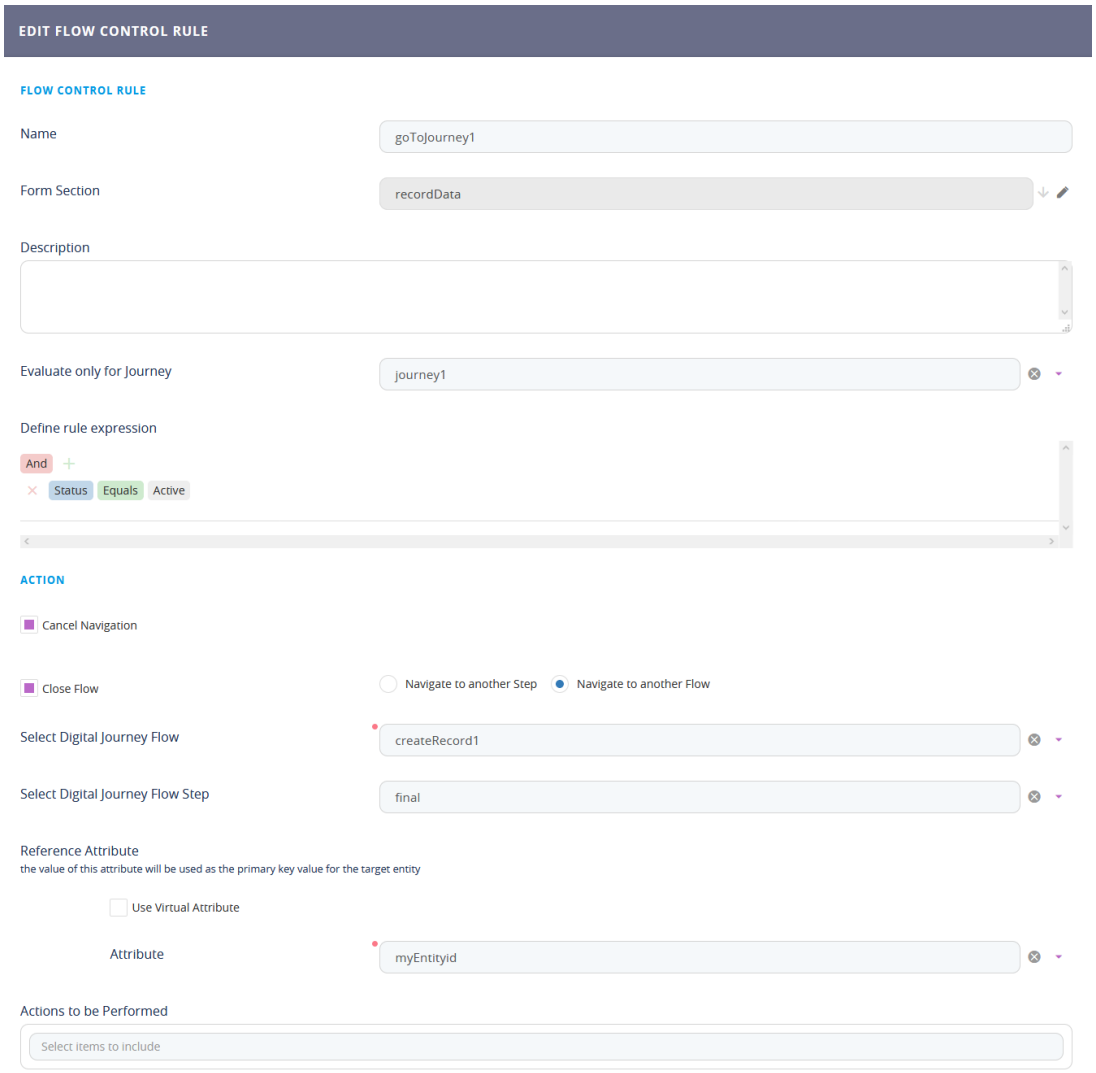
In the Default Navigation Rule, if the ''Close flow'' bool is true after the step 2 is displayed, the flow is closed. If the bool ''Navigate to another step'' is true, the flow will continue to another step selected in the option set. If the bool ''Navigate to another flow'' is true, then the next step will belong to another flow.

At the top-right corner of the page, click the **Save and close** button to save the step.

Control Form Driven Flow based on rules

To add rule at a step level to control the default flow of a digital journey, follow these steps::

1. Go to the desired step edit form and click the **Flow Control** tab.
2. Click the **Insert** button. The **Add Flow Control Rule** page appears.



3. Provide a **Name** for the rule to be run on step. In the Form section, the name of the step you are editting will be automatically filled in.
4. Insert the proper description for the rule.
5. If you wish to evaluate the rule only in the context of a specific digital journey, select the corresponding journey from the **Evaluate only for Journey** drop-down box. For information, see "Digital Journey Context" on page 295.

6. **Define rule expression** based on which specific action happen. It is possible to **add a condition** or to **add a group**. When adding a condition it is possible to add a ''**Checking with Custom Processor**''.

   When adding a condition, first select the attribute on which the condition will apply. For some attributes if you click on the little black arrow on the left of the attribute itself it will open a new set of options such as:

   - minutes since
   - minutes until
   - hours since
   - hours until
   - days since
   - days until
   - day of
   - weekdays of
   - days until anniversary
   - has anniversary today
   - months since
   - months until
   - month of
   - years since
   - years until
   - year of
   - date of

- The following choice is to select the rule. Based on the type of attribute a set of possible rules will be shown:
    - contains
    - does not contain
    - starts with
    - ends with
    - equals
    - does not equal
    - is blank
    - is not blank

    or

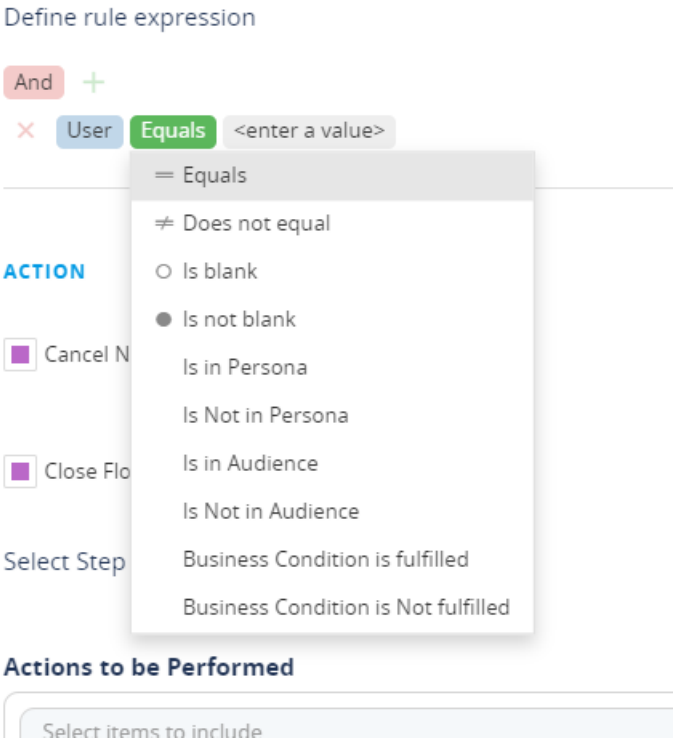
    - equals
    - does not equal
    - is less than
    - is greater than
    - is less than or equal to
    - is greater than or equal to
    - is blank
    - is not blank
    - is in between

    For attributes such as user it is possible to set:

    - equals
    - does not equal

- is blank
- is not blank
- is in Persona
- is not in Persona (see Customer Persona)
- is in Audience (see Audience management)
- is not in Audience
- Business condition is fulfilled
- Business condition is not fulfilled

After selecting the two fields enter a value.

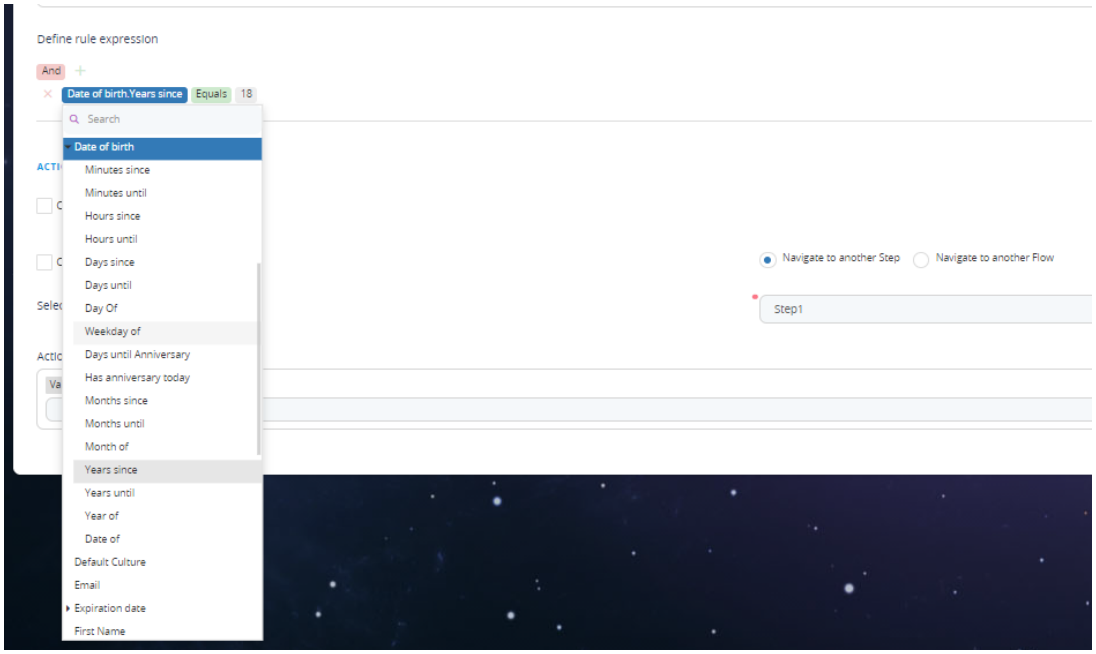


For example, add a condition for the age of a client to be greater than 18 years old.

7. Select the action:

   Navigate to section – navigates to the specified step instead of the next step

Cancel Transition – displays an error message and nothing happens.

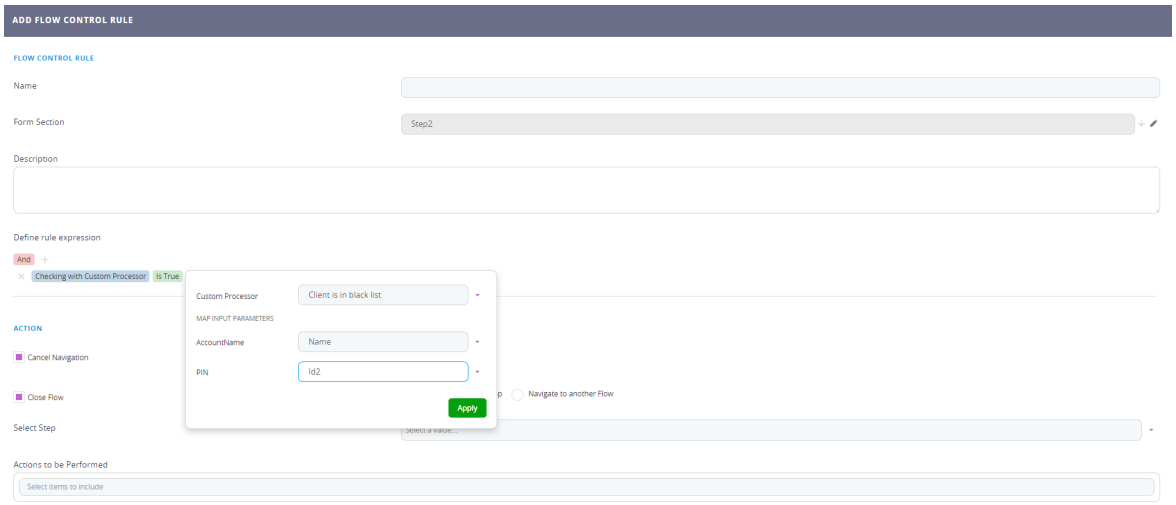


If you choose to **Cancel Transition** to the next default step, you need to provide a **Cancel Navigation Message**.

8. Select the ''**Actions to be Performed**''. From the list created in "Defining Form Actions" on page 240 select the Action to be triggered when the rule is applied.
9. Save the rule by clicking the ''save and close'' button and add as many rules as you need.

Checking with Custom Processor

It is possible to create a rule that checks the attribute mappings of an endpoint to return the values true or false from the boolean attribute.



1. Create the endpoint with the input and output mappings as bool in Server Automation Scripts. In the code section write the following line:

```
context.result = true;
```

2. Navigate to the Form Driven Flow, to the Flow Control, click the insert Rule.
3. In the rule add the condition with checking the custom processor.
4. Select from the option set the name of the endpoint and map the input and out put to the bool attributes you wish to have the data in.
5. Click the ''Apply'' button.
6. Select the next step where to navigate and click the ''Save and close'' button.

Configuring Field Options

Using field options enables you to build dynamic forms and user journeys. This feature allows you customize the fields displayed in the user interface and to create rules which apply to specific fields based on how users have filled out other fields in the form driven flow: show field values, show or hide specific fields, etc.

Customizing the user experience by using field options ensures that users always see the fields that are relevant to them.

Some use case scenarios for using this feature:

- Show relevant cities based on selected country
- Automatically change the account currency based on selected country
- Show CIF/CNP fields based on selected customer account type.

This section walks you through the prerequisites and the steps that you need to follow customize form fields and to create a field rule.

How to Customize Form Fields

To customize form fields for specific attributes, in the configuration page of the form driven flow, select the **Field Options** tab and click the **Insert** button to customize the field for a specific attribute. Then, fill in the corresponding settings for the field:

Field	Description
Use Virtual Attribute	Tick when rendering fields for data extensions.
Attribute	At the right-side of the **Attribute** field, click the drop-down and select the attribute for which you wish to customize the form field. This field is mandatory.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Show Tooltip</td><td>Inherit (default value). Inherits the tooltip show options selected on the General tab. If no options have been previously selected the default will be User Settings, which means that the users will be able to show tooltips by toggling on / off the Tooltips button displayed at the top-right corner of the UI.

NOTE
The Tooltips toggle button is available in the Digital Experience Portal only when tooltips are available on the journey.

• YES. Always show tooltips in the Portal UI on forms and user journeys when users hover their mouse on the attributes on fields which have tooltips. The users do not have the possibility to toggle the tooltips off.
• No. Never show tooltips in the Portal UI on the data form / digital journey. The users do not have the possibility to toggle the tooltips on.</td></tr>
<tr><td>Custom Tooltip</td><td>The text which will be shown in the tooltip in the Portal Ui. The tooltip text is localizable.

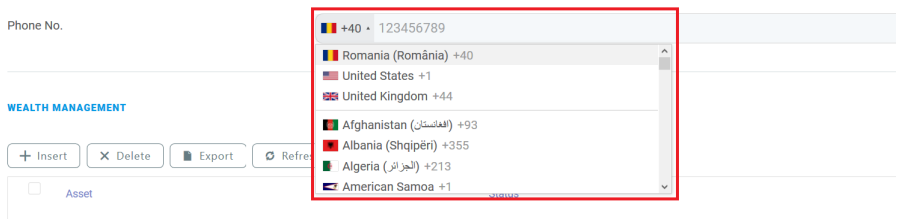
NOTE
The maximum length of the tooltip text is 500 characters.</td></tr>
<tr><td>Lookup View Name</td><td>Lookup attributes only! Predefined view used to select values for the attribute. If no value is selected, the attribute's default view is used.</td></tr>
<tr><td>Lookup as DropDown</td><td>Lookup attributes only! Display the lookup attribute values in a drop down list instead of a separate grid window.</td></tr>
<tr><td>Show Edit Button</td><td>Lookup attributes only! Display the pencil button allowing you to edit the selected lookup attribute.</td></tr>
<tr><td>Lookup Edit Form</td><td>Lookup attributes only! Predefined form used to edit the selected lookup attribute. If no value is selected, the attribute's default edit form is used.</td></tr>
<tr><td>Lookup Show Insert Button</td><td>Lookup attributes only! If the lookup attribute is selected from a grid window, not from a drop down list, the grid will not include the Insert button (you will not be able to add new lookup attribute records).</td></tr>
<tr><td>On File Upload Error</td><td>Attribute of type file only! Allows execution of a client side script if the file upload fails.</td></tr>
</table>

Field	Description
Field is Read Only	The field will be non-editable.
Field Required Level	The **Required Level** drop-down allows you to choose if a specific attribute (field) is to be mandatory, recommended or optional: • **None** – The field is optional. No error message will be displayed if the field is empty. • **Recommended** – A blue dot will be displayed on the upper-left corner of the field in the user interface to indicate that it might be useful to fill in the field. • **Required** - A red dot will be displayed on the upper-left corner of the field in the user interface to indicate that it is a mandatory field. The end user will not be able to add a new record if the field will be left blank.
UI Template	Allows you to select placeholder for either email or phone. Select the template available for your attribute. After the selection is done, a HTML code will be displayed.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>UI Template Options</td><td>
This field is shown only if the UI Template is selected. You can customize the selected placeholder by modifying the code based on your preferences. For the Phone Placeholder UI templates, the following options are available:

type – "tel"
autocomplete – true/false. Selects the use of the browser's autocomplete feature.
placeholder – Displays a grayed out value for exemplification purposes when the field is empty.
default_country – Phone number format based on the ISO 3166-1 aplha-2 country codes used by default.
onlyCountries – List of allowed phone number formats based on the ISO 3166-1 aplha-2 country codes. When empty, all country-specific phone number formats are available.
preferredCountries – Phone number formats based on the ISO 3166-1 aplha-2 country codes displayed at the top of the list.
excludeCountries – Phone number formats based on the ISO 3166-1 aplha-2 country codes excluded from the list.

For instance, the template options below
<pre><code>return {
    type: "tel",
    autocomplete: true,
    placeholder: "123456789",
    default_country : "ro",
    onlyCountries: [],
    preferredCountries: ['ro', 'us', 'gb'],
    excludeCountries: [],
}</code></pre>
will create the following field:
</td></tr>
</table>

Field	Description
	
Attribute Change Event	Code to be executed each time a form field is updated. For instance, you can dynamically populate a lookup field based on the value selected in another field. For more information, see "How to Configure Field Rules" below.

How to Configure Field Rules

To create field rules, on the entity linked to your form driven flow, you should have at least two attributes defined, onefor the condition and the second for the action. For example, if you want to display relevant cities based on the selected country, on the **accountInsertForm** digital journey of the account entity, you have to add the country field and the city field on which you define the action (where both fields are lookup fields). If the data form does not have the two fields, add them.

1 Add field for action

Configure the field used as filtering criterion (eg. country) as described in the "How to Customize Form Fields" on page 235 section.

2 Add field for condition

Add the field which is filtered (e.g., city) and configure it as described in the "How to Customize Form Fields" on page 235 section.

Then, in the Attribute Chagne Event field, provide the piece of code to be executed each time the lookup value of the country field changes, for instance:

```
var accountCountryId = ebs.getFormAttributeValue
("ebsContainerContent", "Country")
ebs.getByQuery({
        entity: {
                alias: "a",
                name: "accountCountryId",
                attributelist: [
                        {
```

```
                        name: "Cities",
                        }
            ]
        },
        "where": {
            "type": "and",
            conditionlist: [
                        {
                        first: "a.accountCountryId",
                        type: "equals",
                        second: "val("+ accountCountryId +")"
                        }
                    ]
        },
function(e){
        if(e.Records != null && e.Records != undefined &&
e.Records.length > 0)
            {
            ebs.setFormAttributeValue("ebsContainerContent",
e.Records[0].a_accountCountryId);
            }
})
```

Save the field, then save the form driven flow.

In the user interface, each time you select a country, the list of cities will include only cities that belong to that country (i.e., the cities relevant for the country).

Defining Form Actions

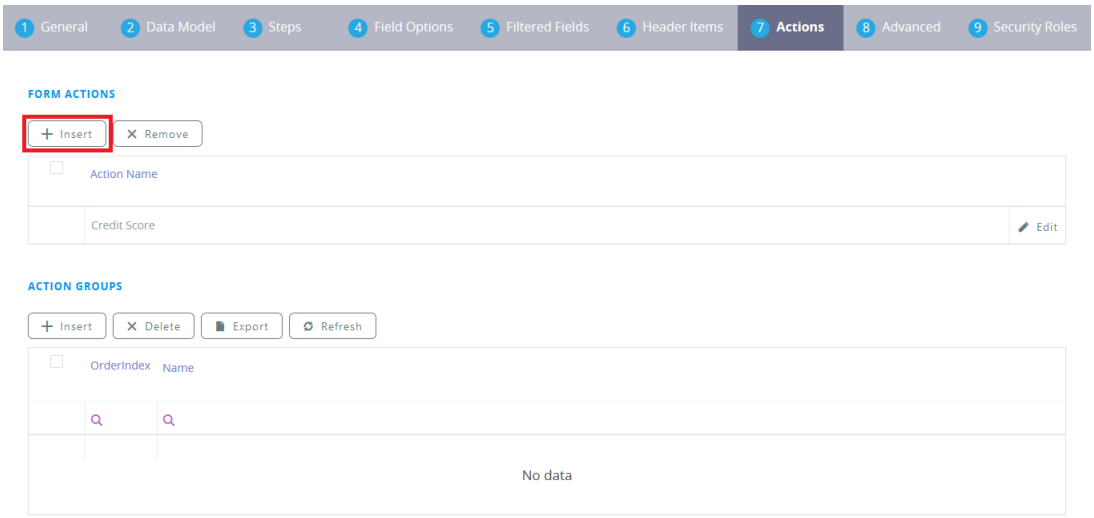
Form actions provide a no-code method to:

- change a record's business status
- generate a report
- run a server side script
- run a business decision matrix
- call a formula.

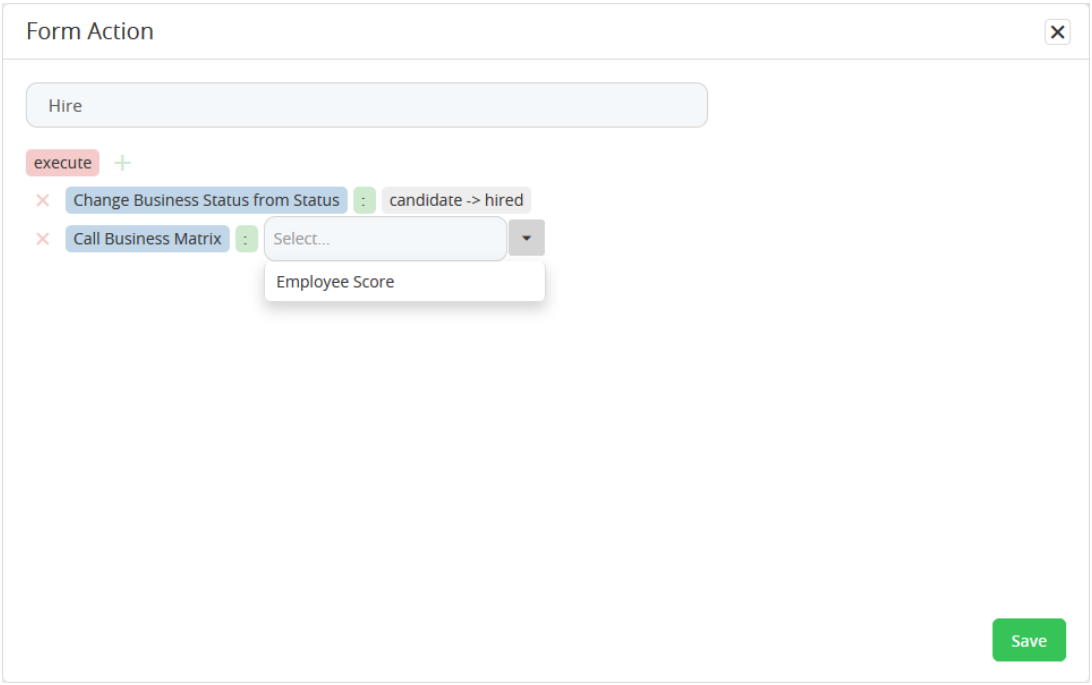
Once defined, form actions can be triggered on-demand, for example by form buttons (see Form Actions Buttons for details).

How to create a form action

1. On the configuration page of the form driven flow, click the **Actions** tab. Two grids will be shown: one is **Form Actions** and the other is **Action Group**.
2. At the top of the **Form Actions** section, click the **Insert** button.



3. In the **Form Action** window:
   a. Enter a name for the form action.
   b. Click the Plus (+) sign next to the execute label to add a command.
   c. Click the labels in the command to select the desired operands such as change business status from status/ generate digital document/ call custom processor/ call business matrix/ call a formula with mapping. Then, select the status/ the document/ the processor/ the matrix.
   d. Go back to step b. if you wish to add additional commands. To remove commands from the list, click the × button.
   e. Click the **Save** button at the bottom right corner of the page.



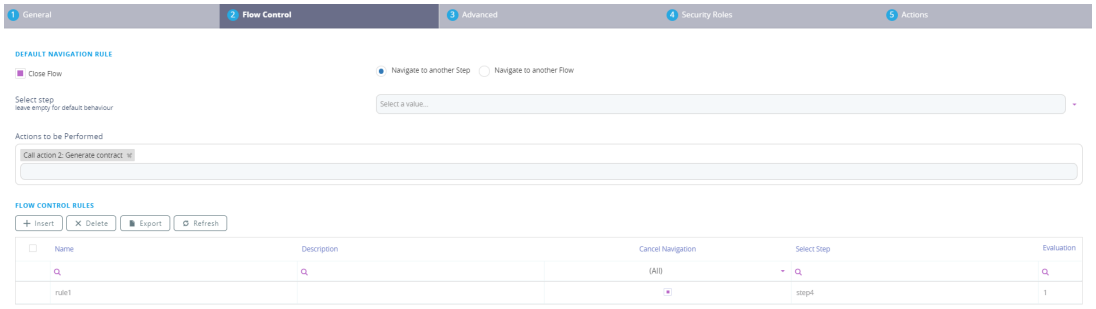
Available form action commands

Command	Description	Operands	Examples
Change Business Status from Status	Changes the record's workflow status based on the entity's attached business workflow. For details, see the Business Workflows Processor documentation.	• Initial status • Final status	Change the status from Draft to Active
Generate Digital Document	Generates a predefined report. For details, see "Analytics" on page 438.	Report name.	Generate a contract or an agreement
Call Custom Processor	Runs a predefined on-demand server automation script. For details, see "Creating On-demand Server Automation Scripts" on page 496.	Server automation script name.	Call the E-sign processor or an endpoint.
Call Business Matrix	Runs a predefined business decision matrix. For details, see the Business Decisions Processor documentation.	Business decision matrix name.	Call the eligibility matrix.
Call Formula with data mapping	Runs a predefined formula for calculation of input data. See "Calling the Business Formulas" on page 433.	Business Formulas	Call the formula for calculating the policy of an insurance.

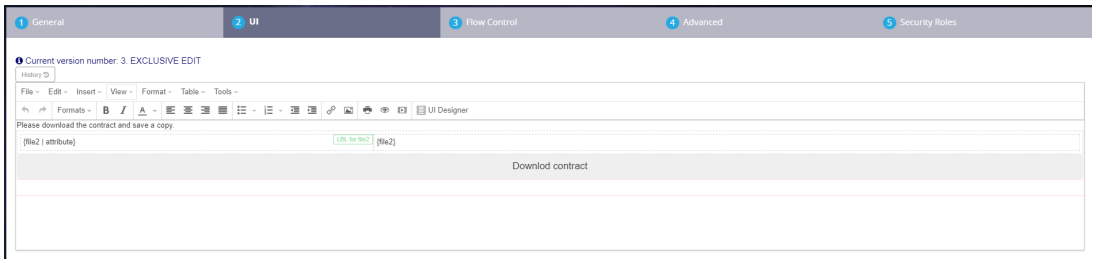
Alternatively, it is possible to trigger an action from the actual step of a digital journey.

How to add an action to a specific step in a Digital Journey

1. Open the main menu and click on the Digital Journey menu item. Open **Form Driven Flows**.
2. Select the flow you wish to work on. Click the ''**Steps**'' tab. Select the step you wish to modifiy.
3. Inside the step, click on **Flow Control** tab.



4. In the **Actions to be performed** section, select the action you wish to have in that particular step. The available actions are those configured earlier from the **Available form action commands** table.
5. Click the ''**Save and Reload**'' button.
6. Optionally, if you have selected the Generate contract action, go to the **UI** tab and add the button for this action and the file attribute created in the Data Model of the entity. For more information, see Digital Documents Processor.



7. Click the ''**Save and Reload**'' button.

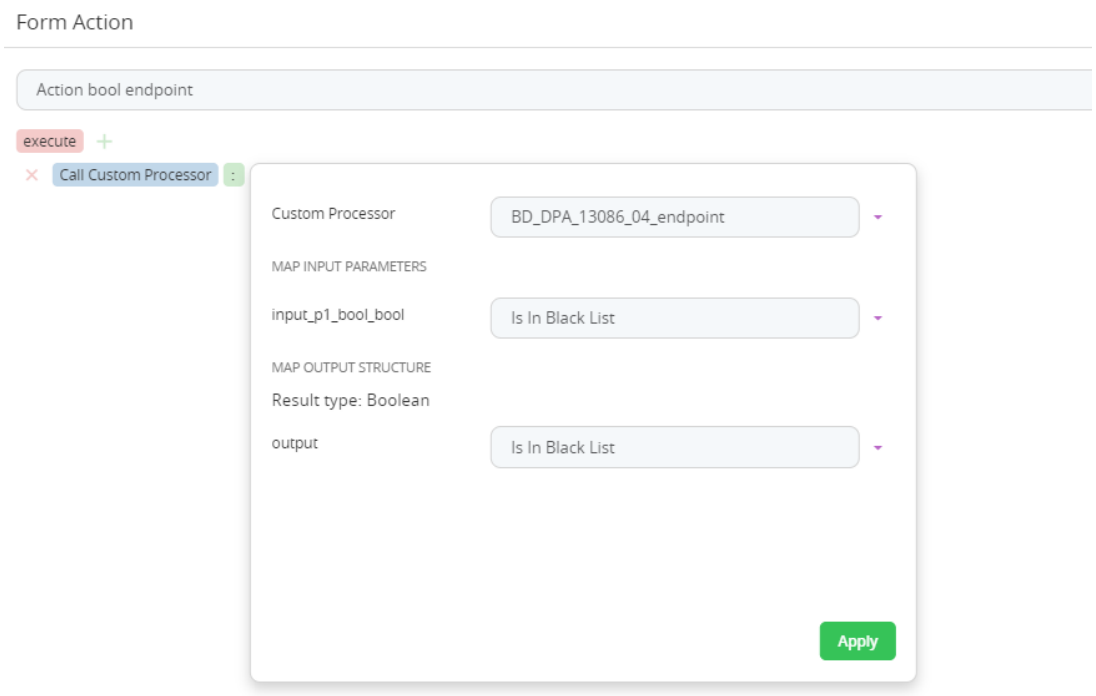
How to attach an endpoint in Form Action

In form Driven Flow, in form Actions, a user can map between an automation script input/output parameters and form attributes.

1. Create one Platform Data Entity that is default for a FDF type wizard.
2. Create a server script with Input Parameters and the Output structure is one of the following, depending on your needs:
   - [none]
   - “Entity"
   - “Custom“
   - “Boolean“.

   If the endpoint has no output structure type, then no output mapping form will be displayed. For more details, see "Creating On-demand Server Automation Scripts" on page 496.
3. Navigate to the Form Driven Flow, **Actions** tab and in the first grid click the ''Insert'' button.
4. Insert a name for the action.
5. Click the execute button and select the ''Call custom processor''.
6. Select for the drop-down the endpoint created.
7. Map the input attributes.

8. Map the output attributes.



9. Click ''**Apply**''. Click ''**Save**''.
10. Click the ''**Save and close**'' button.

Defining Action Groups

Innovation Studio allows you to create a custom group of actions that can be triggered on demand on data form driven flows, when a button is clicked.

This section walks you through the steps you need to follow to create an action group.

Prerequisite

- You need to have an on-demand automation script defined on the entity for which you create the form driven flow.

STEP 1. Add action group

On the configuration page of the form driven flow, click the Action Groups tab.

At the top of the Action Groups section, click the Insert button. The Add Action Group page appears. Provide the following properties:

Property	Description
Name	The name of the action group. Make sure that you use the following naming convention: pascal case, no special characters and no blank spaces.
Display Name	The name of the action group that will displayed in the Digital Experience Portal.
Entity Form	Select the data form for which you define the action group.

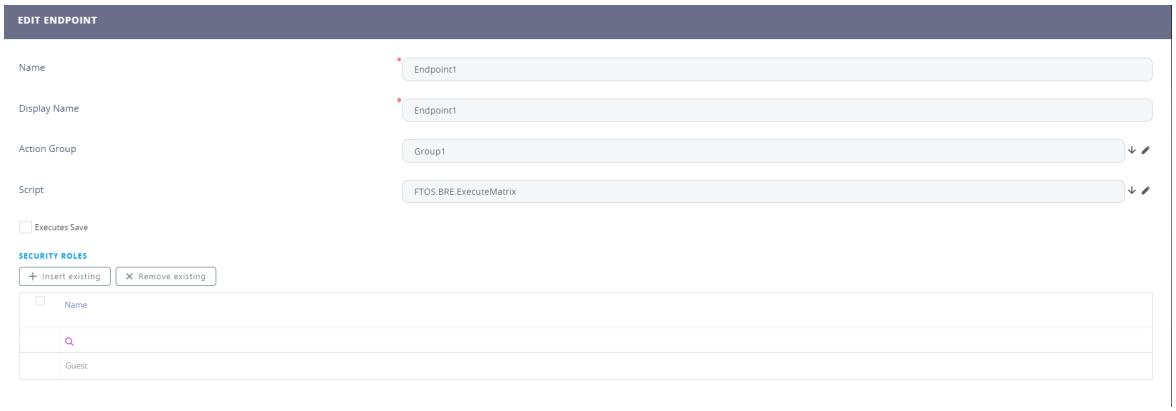
Click the Save and reload icon. The Add Action Group page is replaced by the Edit Action Group page and the Actions section becomes available.

STEP 2. Add endpoints

Add the endpoints on which actions defined in the selected on-demand automation scripts will be run on button click. To do so, from the Endpoints section, click the Insert button. The Add Endpoint page will be displayed. Provide the following action properties:

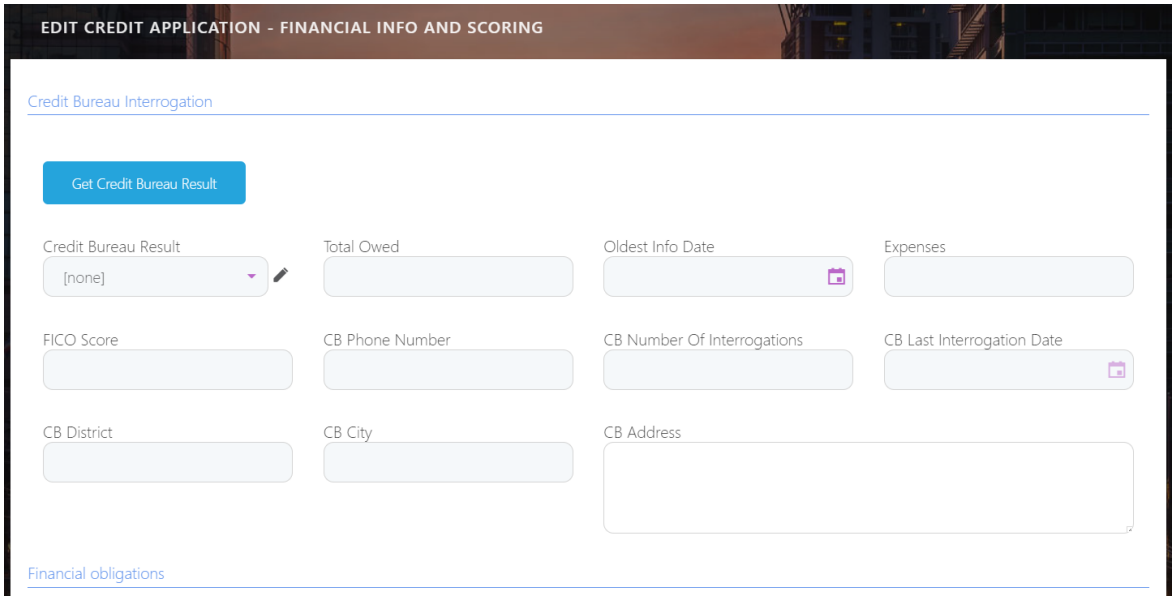
Property	Description
Name	The name of the endpoint that is used by the system
Display Name	The name of the endpoint that will be displayed in the Portal UI.
Script	From the **Script** drop-down, select the on-demand automation script which will be run on button click. For more information on automation scripts, see Server Scripts. If you have no on demand scripts defined, you can add one by clicking the Script drop-down, clicking the Insert button in the page listing the available scripts and in the Add Scripts page providing the script properties.
Executes Save	Saves after the script execution.

At the top-right corner of the page, click the Save and close icon.



You can add as many endpoints as you need. The endpoints are displayed in order of their index. To change their order index, drag and drop the desired action row.

The figure below shows the actions defined in the Action Groups for the credit bureau interrogation in the user interface.



How to hide the action button

To hide the action button, click the Advanced tab, then click the After Events tab and in the field type the following JavaScript code:

```
$("div[data-action-group-name="<the name of the action group you want to hide>"]").hide();
```

At the top-right corner of the page, click one of the save icons to save the changes. The action button will be hidden.

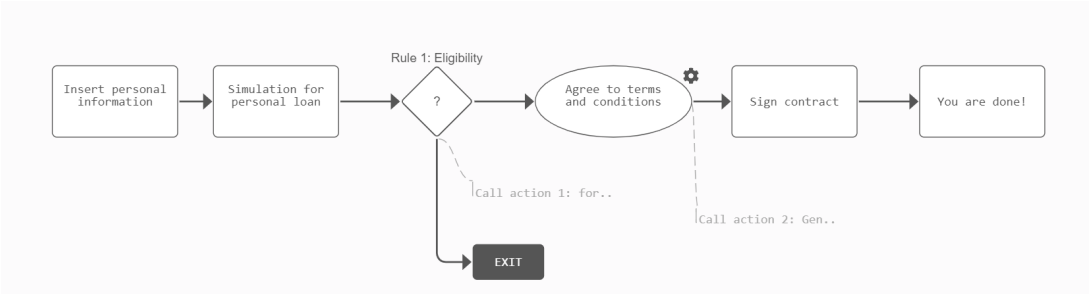
Flow Map

This map shows the user the steps in a flow, the rules and transitions. The difference from the "Digital Journey Map" on page 293 is that this map only shows one flow as oppose to the digital journey map that shows all flows in a journey.

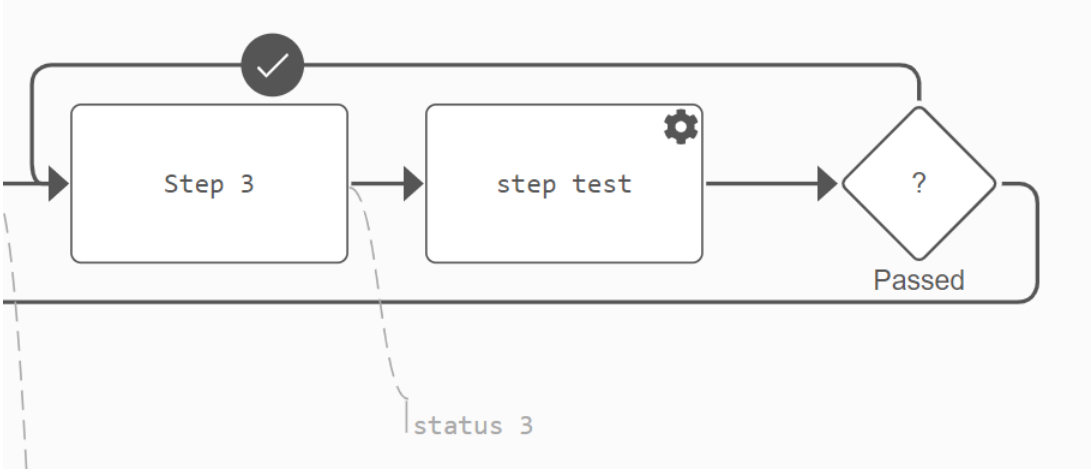
> **IMPORTANT!**
> The flow control is available only for Form Driven Flows of type wizard.

The Flow map exhibits the actions performed on a step configured in "Flow Control" on page 227. To do so:

1. Open the Innovation Studio, open the main menu, select the Digital Journey chapter, click on the Form driven flow.
2. Select the existing Form driven flow you wish to modify or create a new one.
3. After opening the flow, click on ''**Flow map**''.



4. Click on a step to see the UI configurations or the rule for flow control. Configure to add a step or an action or a transition.



For example, step 3 is a default step, step test is a processor step, the passed step is a rule and the status 3 is a action performed after the step 3 data is saved and the action is triggered.

5. Click the ''**Save and reload**'' button to save the changes.

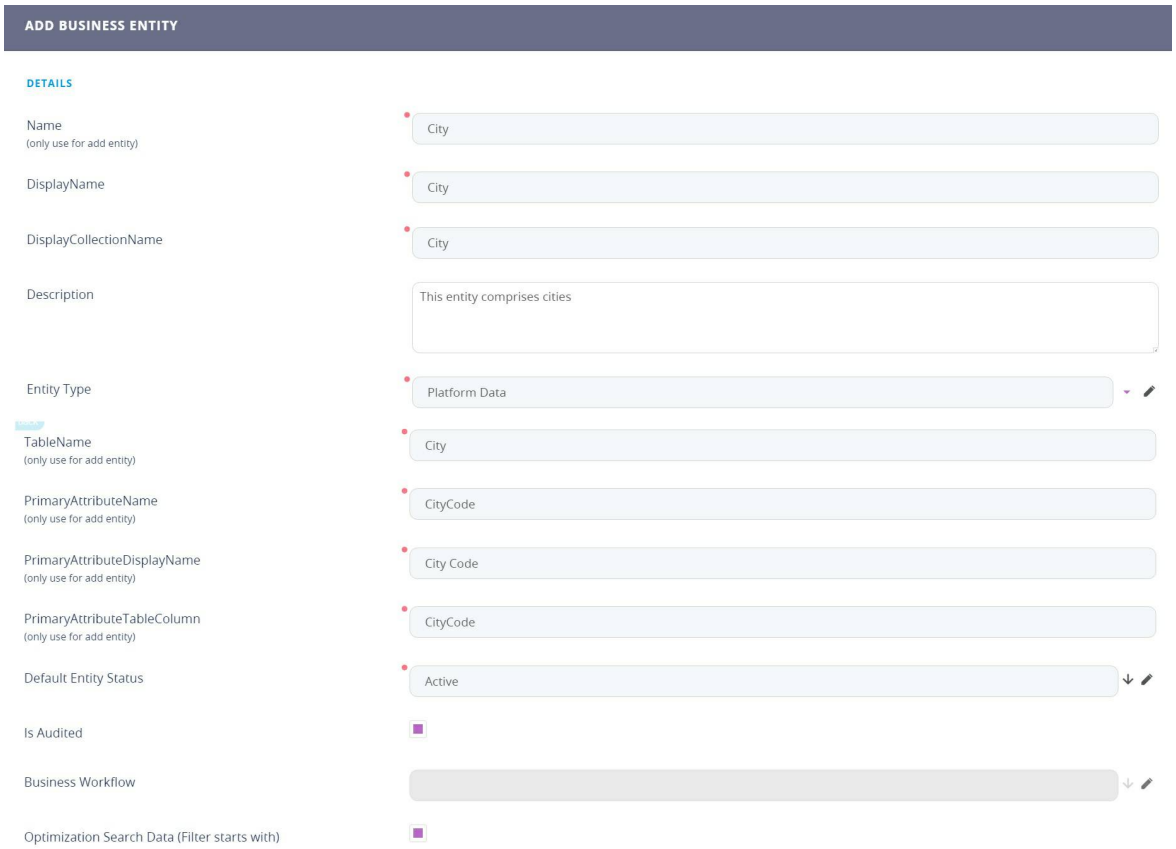
Defining Filtered Fields

The **Filtered Fields** tab allows you to fetch entity data from lookup fields. You can dynamically gather data from the platform and display it on forms / user journeys as best suits your business needs.

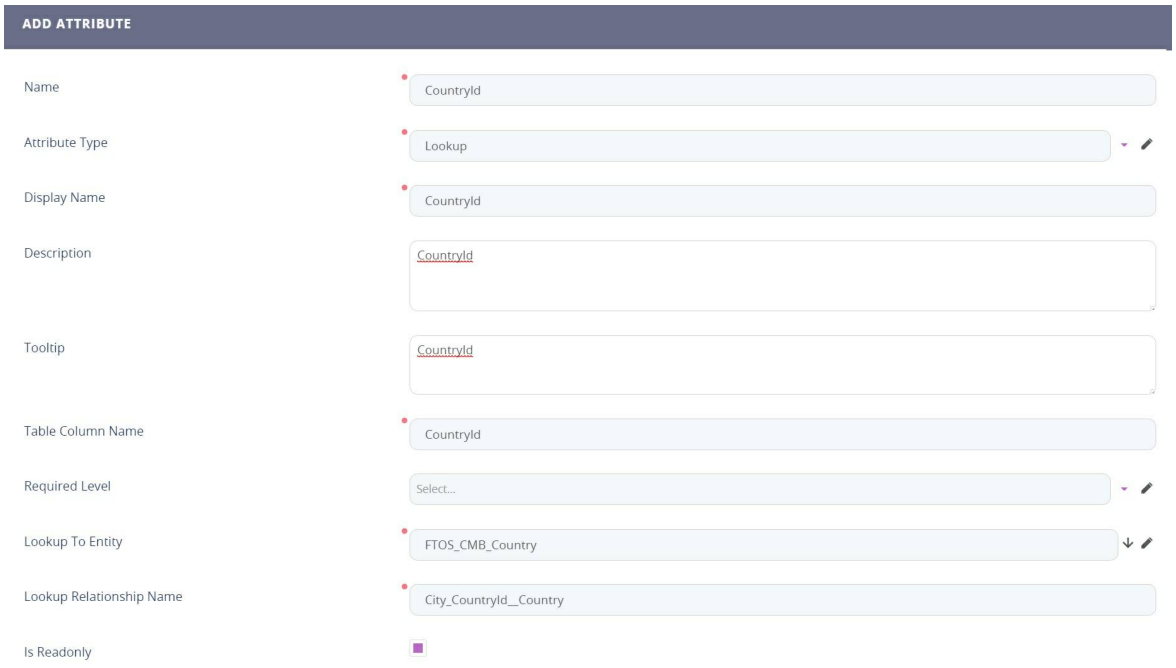
With Filtered Fields you can control the values that can be used in a field on forms based on the value selected in another field. For example, the Countries and States / Provinces fields. When a user selects a country, you might want to display the states / provinces corresponding to the selected country.

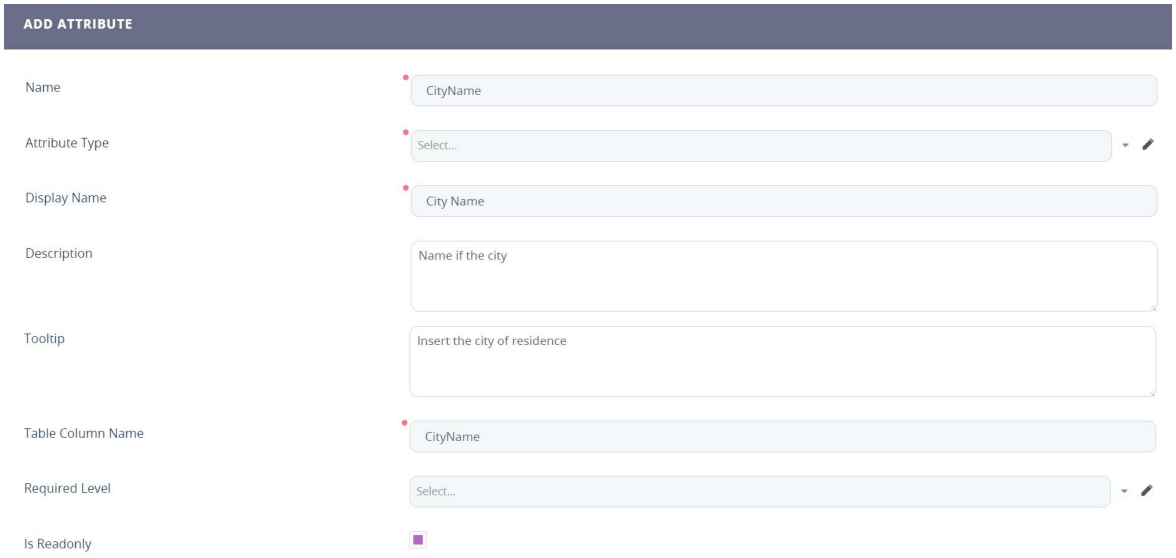
How to add filtered fields

Before starting the process, make sure that there are three entities created, two entities will serve as look-up to one another (for example entity A and entity B) and a third on which the digital journey is built (entity C).

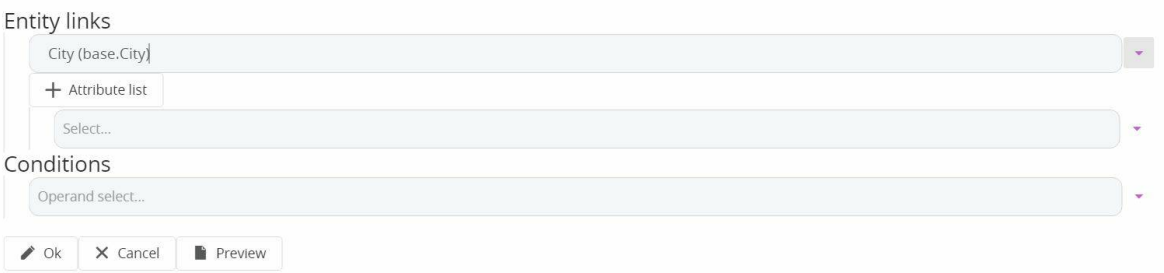


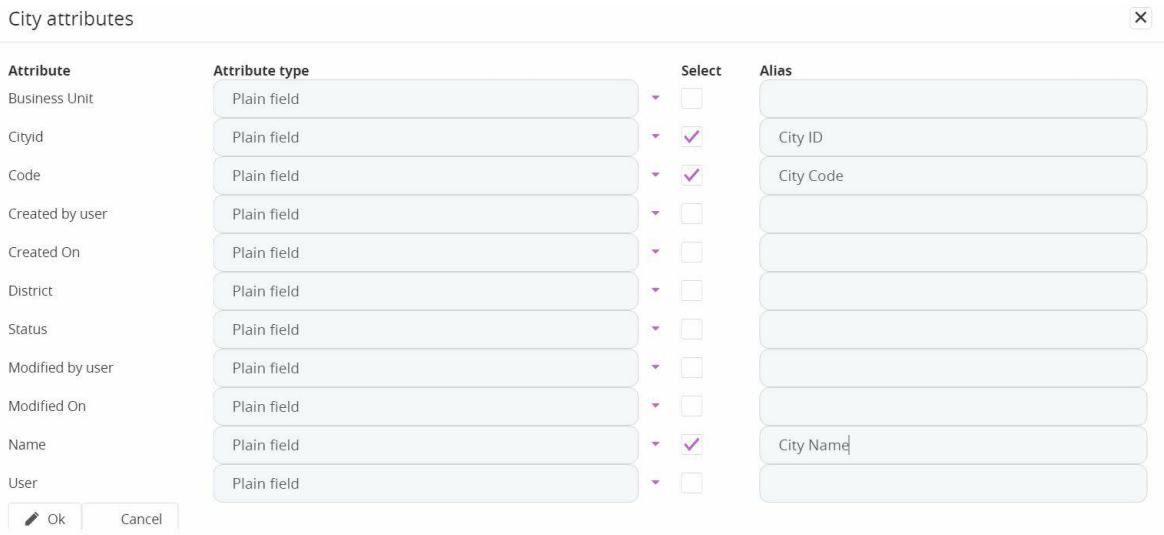
The two entities A and B need a primary attribute name (see "Attributes" on page 54), then create a secondary attribute for each of the entities. Afterwords, create a look-up attribute from entity B to A and the name of this attribute must coincide with PK name of the other entity e.g. in entity B create a look-up attribute to entity A named entityAID.



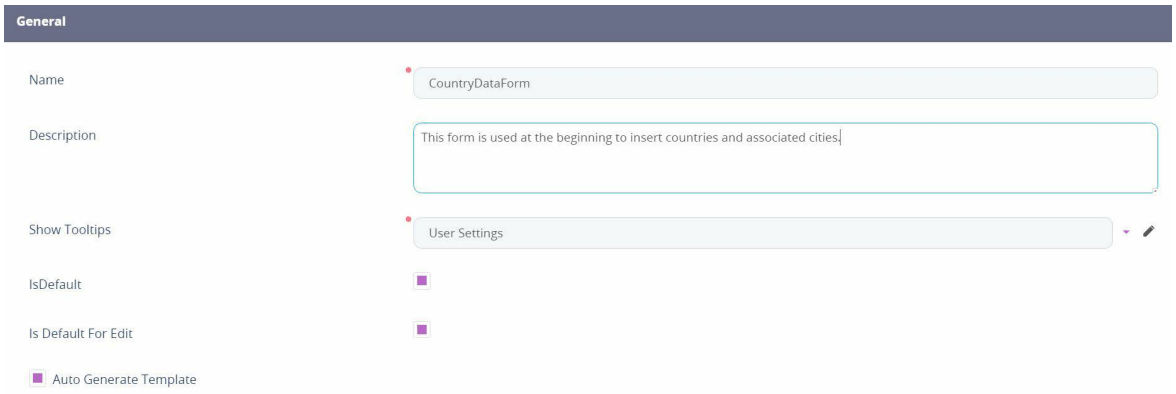


Moreover, configure the data view on both to show these attributes (see "Data Views" on page 96).

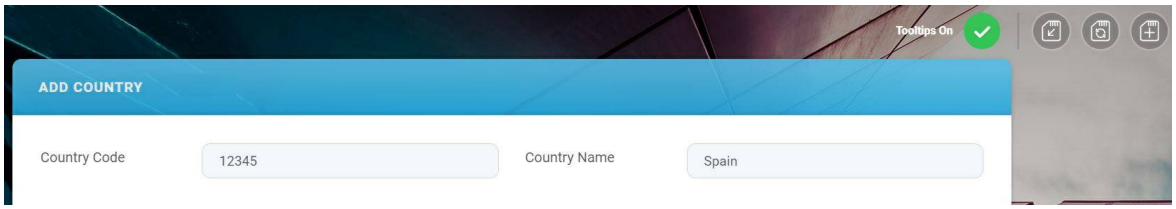


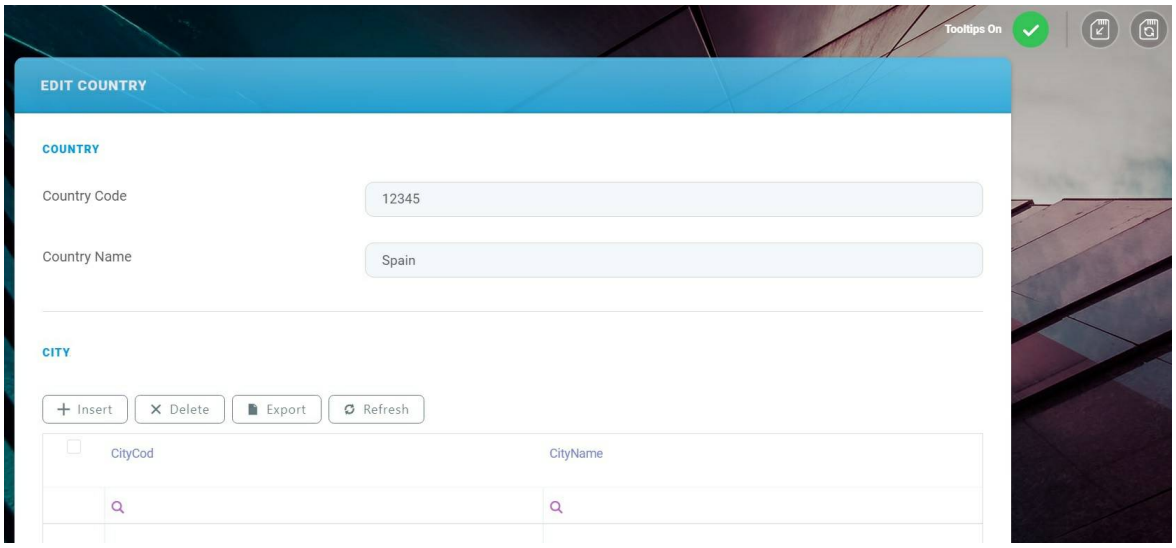


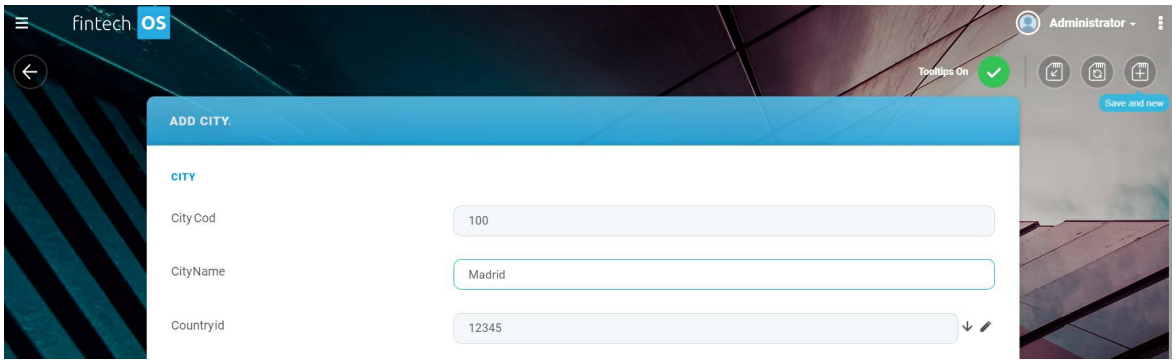
Next, configure the data form (see "Data Forms" on page 126) on entity A, where in the UI tab insert the two attributes created and a relational container where a user must select the relation between entities A and B and tick the view grid default which was edited earlier.



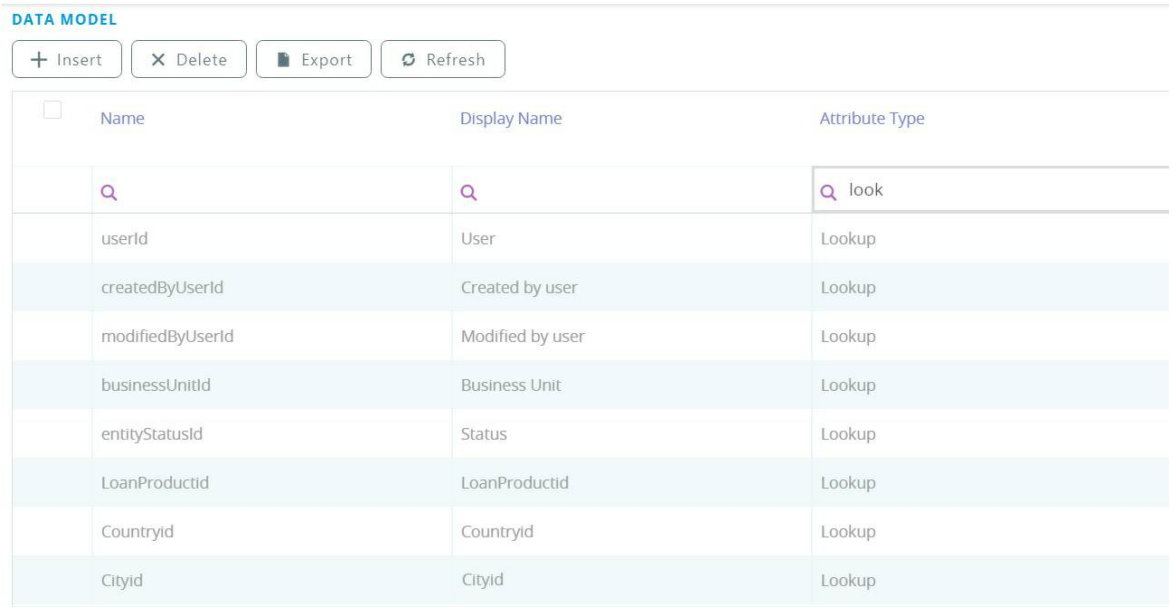
Create a shortcut to the entity A of the type Insert on one of the available Dashboards. Where map the values form one field to the other.







Inside entity C, create two attributes type look-up to A and B. The names must coincide with the PK e.g. entityAID and entityBID.



Going into the digital journey or flow, open the step where the attributes are needed and insert the two attributes created before e.g. entityAID and entityBID.

To add filtered fields, in the **Filtered Fields** section, from the Filter Fields list, click the **Insert** button. The Add Filtered Field page will be displayed. Read the on-screen description and provide the properties:

At the upper-right corner of the page, click the Save and Close icon. The record is saved and is displayed in the Filtered Fields list.

For example, entity A is a country entity and entity B is a city entity and C is Account or client entity.

Filtered fields on editable views

The filtered fields also apply on editable views. If you set up inline editing for a view, the filtered fields settings that you find on the data form will also apply on the view.

In the Attribute To Filter Reference field, you can also specify the alias when you use a custom view for the lookup that will be filtered and this field is not on the main entity (the entity of the lookup).

When using a custom view for the lookup field that will be filtered and this field is not on the main entity (the entity of the lookup), in the Attribute To Filter Reference field, you can also provide the alias.

In the example above, **Attribute to filter reference** is "f2.Field3Id" and "f2" is the alias from the entity Field3 which is related to Field1 which is the entity from the lookup.

Header Items

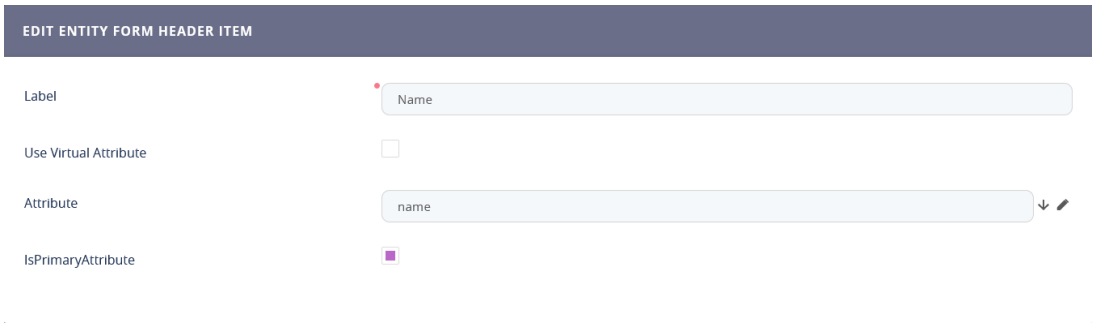
When creating user journeys, the **Header Items** tab allows you to provide users visibility to the most relevant information (attributes). It also allows you to make the form driven flow's header sticky on scroll, which is useful when a data form has many attributes and users have to scroll-down to complete it.

To add a new header item:

1. Click the **Insert** button. The Add New Header Item page will be displayed.
2. Fill-in the following fields:

Field	Description
Label	The name of the header item which will be displayed on the form driven flow.
Use Virtual Attribute	Select if you wish to display a virtual attribute from the entity's extended data model, instead of a regular attribute. This will replace the Attribute field with a drop-down called Virtual Attribute which will allow you to select virtual attributes instead of regular attributes.
Attribute/Virtual Attribute	The value of the entity attribute which will be displayed on the journey header item. If you selected the Use Virtual Attribute checkbox, you will be able to choose virtual attributes from the entity's extended data model.

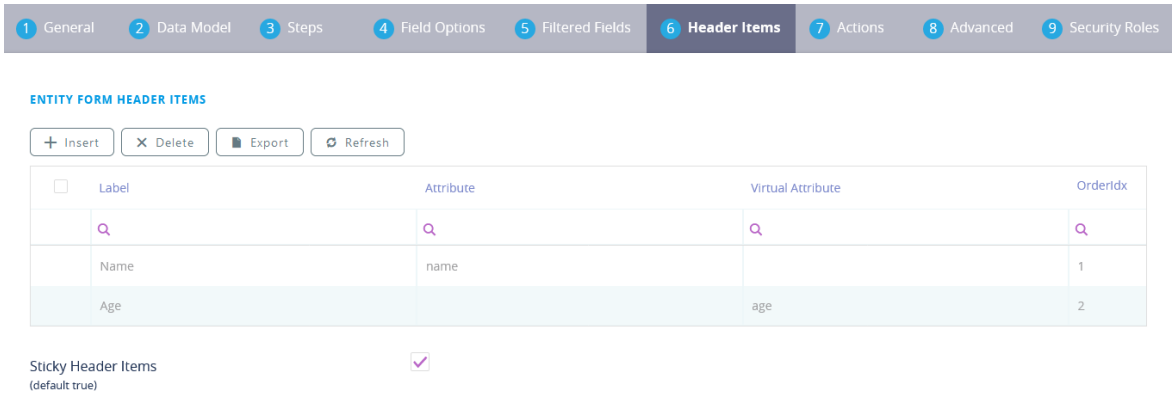
Field	Description
Is Primary Attribute	Tick only if the attribute to be displayed in the digital journey header is also the entity's primary attribute.



3. In the upper-right corner of the page, click the **Save and close** icon. The page refreshes and the header item will be displayed in the Entity Form Header Items list.

By default, the header item has the order index set to 1. Add as many header items as you need, then from the Entity Form Header Items list set their display order on the data form, by drag and dropping rows. The header items order is ascendant, whereas the first row (header item) has the order index set to 1.

The data form header is by default sticky on scroll. If you do not want to have a sticky data form header, tick clear the **Sticky Header Items** checkbox.



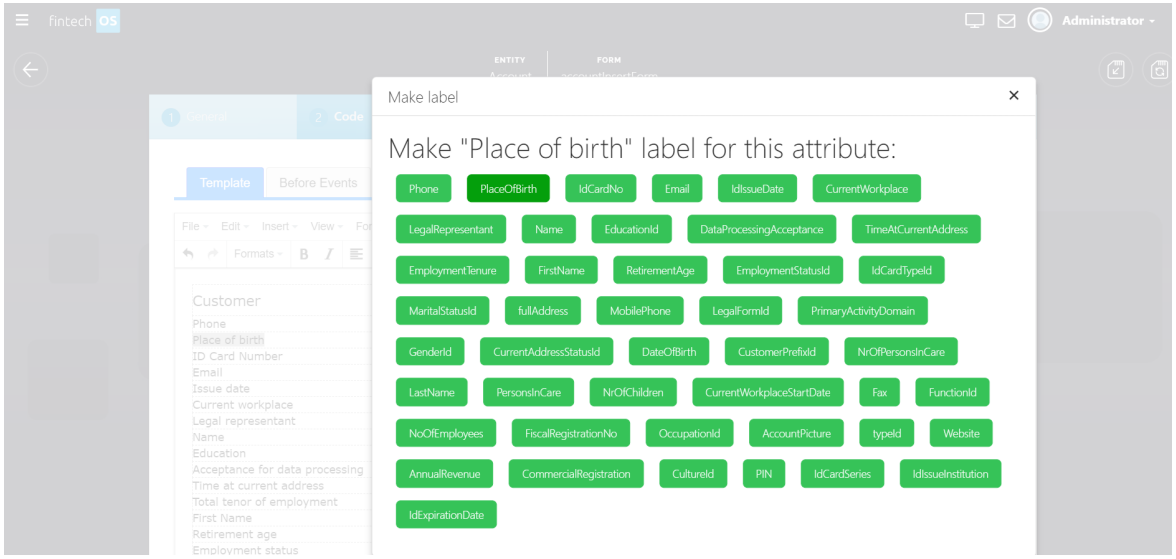
Linking Labels to Attributes

When creating HTML templates (**UI** tab) you can link labels (elements) to attributes. You can do that by using either the HTML editor or the Source code.

Linking labels to attributes using the HTML editor

To link a label to an attribute using the HTML editor, in the template, right-click on the text which will be label and select Make Label. A pop-up will be displayed, listing the attributes you can choose from.

The figure below presents the pop-up might look like:



Select the desired attribute. The pop-up closes and the selected text will be label for the selected attribute.

To remove the link, right-click on the text (label) and select Remove Label.

Linking labels to attributes using the Source code

To link a label to an attribute control using the source code, from the Tools menu, click Source code. In the Source code page, add the HTML attribute data-label-for to that label , set its value to the name of the attribute and click OK.

Source code for linking the label "My Atribute" to the attribute "myAttribute"

```
<div class="row">
     <div class="col-xs-12 data form-label" data-label-
for="myAttribute">My Atribute</div>
     <div class="col-xs-12">{myAttribute}</div>
</div>
```

To remove the link, delete from the source code the `data-label-for` HTML attribute.

Displaying View from Another Entity

Let's say you want to show a view from one entity data form on another. For example, you would like to see on a data form specific details available in a view on a different entity, saving the extra clicks of having to switch from a data form to another one to check those details.

Starting with FintechOS 18.1.10, you can achieve this by using a token on the data form where you want to render the view from the other entity.

Prerequisites

- You have two entities.
- One entity has a view defined (can be the default view) and the other entity has a digital journey or a data form defined.

How to display a view from another entity

1. On the entity where you want to render the view from another entity, go to the edit configuration page of the digital journey or data form. Click the UI tab.
2. On the left panel of the UI Designer select the Relation Data Template and drag it to the container where you wish to place it.
3. After dropping it, click on it to open the right panel and configure it. Select the relation from the drop-down list.

4. Replace the relation tag with the token which will render the view from another entity. The token should follow this format: **{? EntityName, view: viewName ?}**.

> **NOTE**
>
> If you copy/paste the token, the HTML editor might break its formatting and it might not work as intended. We recommend you to either copy/paste the token using a text editor or within the HTML Editor from the Tools menu, select <>Source code and check the token formatting.

The table below describes the token elements you can use when rendering the view.

Token Element	Description
entityName	The name of the entity from which you will render the view.
view:	The name of the view to be displayed.
noheader	Does not display the view header. Display the default view from entity FTOS_CMB_AccountType without header using token **{? FTOS_CMB_AccountType, view: default, noheader ?}**
nofilter	Does not display the view filtering / search. Display the default view from entity FTOS_CMB_AccountType without filtering using token **{? FTOS_CMB_AccountType, view: default, nofilter ?}**
noinsert	Does not display the Insert button on the view toolbar. Display the default view from entity FTOS_CMB_AccountType without the Insert button using token **{? FTOS_CMB_ AccountType, view: default, noinsert ?}**

Token Element	Description
nodelete	Does not display the Delete button on the view toolbar. Display the default view from entity FTOS_CMB_AccountType without the Delete button using token **{? FTOS_CMB_ AccountType, view: default, nodelete ?}**
noexport	Does not display the Export button on the view toolbar. Display the default view from entity FTOS_CMB_AccountType without the Export button using token **{? FTOS_CMB_ AccountType, view: default, noexport ?}**
norefresh	Does not display the Refresh button on the view toolbar. Display the default view from entity FTOS_CMB_AccountType without the Refresh button using token **{? FTOS_CMB_ AccountType, view: default, norefresh ?}**
notoolbar	Does not display the view toolbar. Display the default view from entity FTOS_CMB_AccountType without the toolbar using token **{? FTOS_CMB_AccountType, view: default, notoolbar ?}**
collapse:	The text displayed as a collapse panel. Display the default view from entity FTOS_CMB_AccountType after clicking on the Name text using token **{? FTOS_CMB_ AccountType, view: default, collapse: Name ?}**.
data form:	If the entity from which you render the view has multiple forms and you want a specific data form on edit directly from this view, use **form:** in the token followed by the data form name. **NOTE**Do not use no insert within the token when using **form:**.

Token Element	Description
insertForm:	If the entity from which you render the view has multiple forms and you want a specific data form on insert directly from this view, use **insertForm:** in the token followed by the data form name. **NOTE**Do not use no insert within the token when using **data form:**.

5. After you finish customizing the view layout, save the data form and the entity.

Filtering the view results

You can filter the results displayed in the view by using a custom fetch for the view.

Passing default value

If you want to pass a default value for the insert data form, you must use the `context.on("goToInsert", function(e){})` function;

```
context.on("goToInsert", function(e){
        var pId = "18352c17-0ca4-4b6d-8037-28510e6186d1"
        e.options.defaultVals = "parentId*" + pId;
})
```

Refreshing the view

If you want to programmatically refresh the view, use the `ebs.refreshGrid(gridName)` function.

If no **view:** is specified, gridName will be the entityName token element.

```
For {? myEntity ?}
ebs.refreshGrid("myEntity")
```

If **view:** is specified, gridName will be the entityName token element concatenated with _ and the **view:** token element.

```
For {? myEntity, view: myEntityView ?}
ebs.refreshGrid("myEntity_myEntityView")
```

Rendering Custom Data Extensions

Adding custom data extensions to forms and user journeys is a powerful and flexible technique which allows you to customize the user interface.

Prerequisites

- You have extended the entity with data extensions. For more information, see Extend Data Model.
- In the Data Model section of the form driven flow, add the entity extension which contains the data extensions that you want to render.

How to Render Custom Data Extensions

To render a data extension on a form driven flow, in Innovation Studio, follow these steps:

1. Go to the configuration page of the form driven flow on which you want to render the data extension. The configuration page appears by default on the General tab.
2. Click the Advanced tab, then the After Events tab and provide the code to get the values of the attributes existing in the DB and set the value of the data extension.
3. Click the Field Options tab and add the attributes whose values will be used by the data extension. Make sure to tick the Use Virtual Attribute checkbox when adding the attributes in the list of field options.
4. In the UI template of the form driven flow(journey configuration page > UI tab or in the UI template of the step where you want to render the data extension (data form configuration page > Steps tab > section configuration page > UI tab) add the data extension similar to normal entity attributes by using tokens.

The values of data extensions flow with the save data request, and their values are available for processing in server side scripts inside the `context.AdditionalAttributes.VirtualAttributes` array.

Example

This section teaches you how to automatically calculate Total Expenses as the sum of Unreported Expenses and Expenses, and display the amount in the Exposure section of a loan application.

Prerequisites

- The **FTOS_BAPer_LoanApplication** entity has the attributes **unreportedExpenses** and **expenses**.
- The entity FTOS_BAPer_LoanApplication has been extended with the data extension TotalExpenses relating to attributes **unreportedExpenses** and **expenses**.
- On the LoanApp_ConsumerFlow journey, in the Data Model section, linked to the entity extension is added.

To use the Total Expenses data extension on the LoanApp_ConsumerFlow journey, follow these steps:

1. Go to the configuration page of the LoanApp_ConsumerFlow journey.
2. Click the Advanced tab, then the After Events tab. Provide the code for getting the values of the UnreportedExpenses and Expenses attributes and setting the TotalExpenses data extension so that it returns the sum of the two attributes.

```
var unreportedExpenses = ebs.getFormAttributeValue
('ebsContainerContent', 'UnreportedExpenses');
var expenses = ebs.getFormAttributeValue
('ebsContainerContent', 'Expenses');

ebs.setFormAttributeValue('ebsContainerContent',
'TotalExpenses', unreportedExpenses + expenses);
```

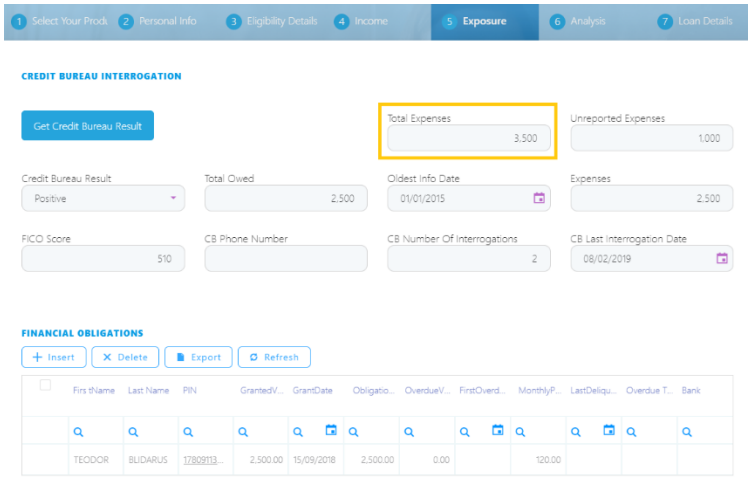
3. Click the Field Options tab and to the list of Entity Form Fields, add the two attributes: unreportedExpenses and expenses. When adding each of the attributes, in the Attribute Change Event field, provide the following code:

```
var unreportedExpenses = ebs.getFormAttributeValue
('ebsContainerContent', 'UnreportedExpenses');
var expenses = ebs.getFormAttributeValue
('ebsContainerContent', 'Expenses');

ebs.setFormAttributeValue('ebsContainerContent',
'TotalExpenses', unreportedExpenses + expenses);
```

4. Click the Steps tab, and in the sections list, double-click the Exposure section (record) and in the **UI** tab, add the Total Expenses data extension by using the token: `{TotalExpenses}`.

In the Portal, when updating any of the fields Unreported Expenses or Expenses, the Total Expenses field will be automatically filled-in (recalculated).



Creating Custom Search Forms

To create a custom search data form based on input fields, search button and list results, follow these steps:

1. On the UI template (UI tab)of the journey / journey step, add a data form container and a button element to the HTML source code:

```
<div id="searchAccount" style="display: inline-block; width:
80%;"></div>
<div id="BtnId" style="cursor: pointer; color: white; border:
2px solid rgba(0, 0, 0, 0);
background: #25a4dc; text-align: center; vertical-align:
middle; padding: 3px; width: 125px;
height: 100%; display: inline-block; margin-bottom: 30px;
margin-left: 15px;">Search</div>
```

2. On the form driven flow, click the Advanced tab, then click the After Events section and provide the code to generate the search.

```
/Generate search data form based on a custom data form
(AccountSearch) with only one input field
(Name):
$("#searchAccount").html("<div id="searchAccountForm"
display: inline-block;"></div>");
var formData = {
        entityName: "account",
        formName: "AccountSearch",
       mode: "insert",
        name: "searchAccountForm",
        mainHtmlId: "searchAccountForm",
        disableControlNavigation: true,
        noRenderIntent: true
    }
ebs.generateForm(formData, function(e) {
});
//Generate grid
var fetchResults = new Object();
fetchResults.entity = new Object();
fetchResults.entity.name = "TestEntity";
fetchResults.entity.alias = "base";
var viewResults = new Object();
viewResults.fetch = fetchResults;
viewResults.viewName = "default";
var afterGenerateCallback = function (e) {
    var gridId = "AccountResult";
    var dataGrid = $("#" + gridId).dxDataGrid("instance");
    dataGrid.option("paging",{pageSize: 10});
```

```
    dataGrid.option("onSelectionChanged", function (g) {
        if (g.selectedRowsData != null &&
g.selectedRowsData.length > 0) {
            $.each(g.selectedRowsData, function(index,value)
{
                currentAccountName = value.base_Name;
                $("#ClientsSelected").text
(currentAccountName);
            });
        }
    });
};
ebs.generateGrid("AccountResult", viewResults, "auto", {},
afterGenerateCallback, {
selectionMode: "single" });
//search function and grid refresh as search result
var onClickSearch = function() {
    var where = {
      type: "and",
      conditionlist: []
    };
    var productName = ebs.getFormAttributeValue
("searchAccountForm","Name");
    if (productName !== null && productName !== "") {
        where.conditionlist.push({
            first: "base.Name",
            type: "contains",
            second: "val(%" + productName + "%)"
        });
    }
    if (where.conditionlist.length == 0) where = null;
    var fetch = $("body").data("AccountResultmyData_
dxquery").fetch;
    fetch.where =
where;
//new updated fetch
    $("#AccountResult").dxDataGrid("instance").option
("dataSource", ebs.getDataStore(ebs.
getEbsDataUrl(), fetch));
    $("#AccountResult").dxDataGrid("instance").refresh();
}
$("#BtnId").click(onClickSearch);
```

Form Driven Mock-up Flows

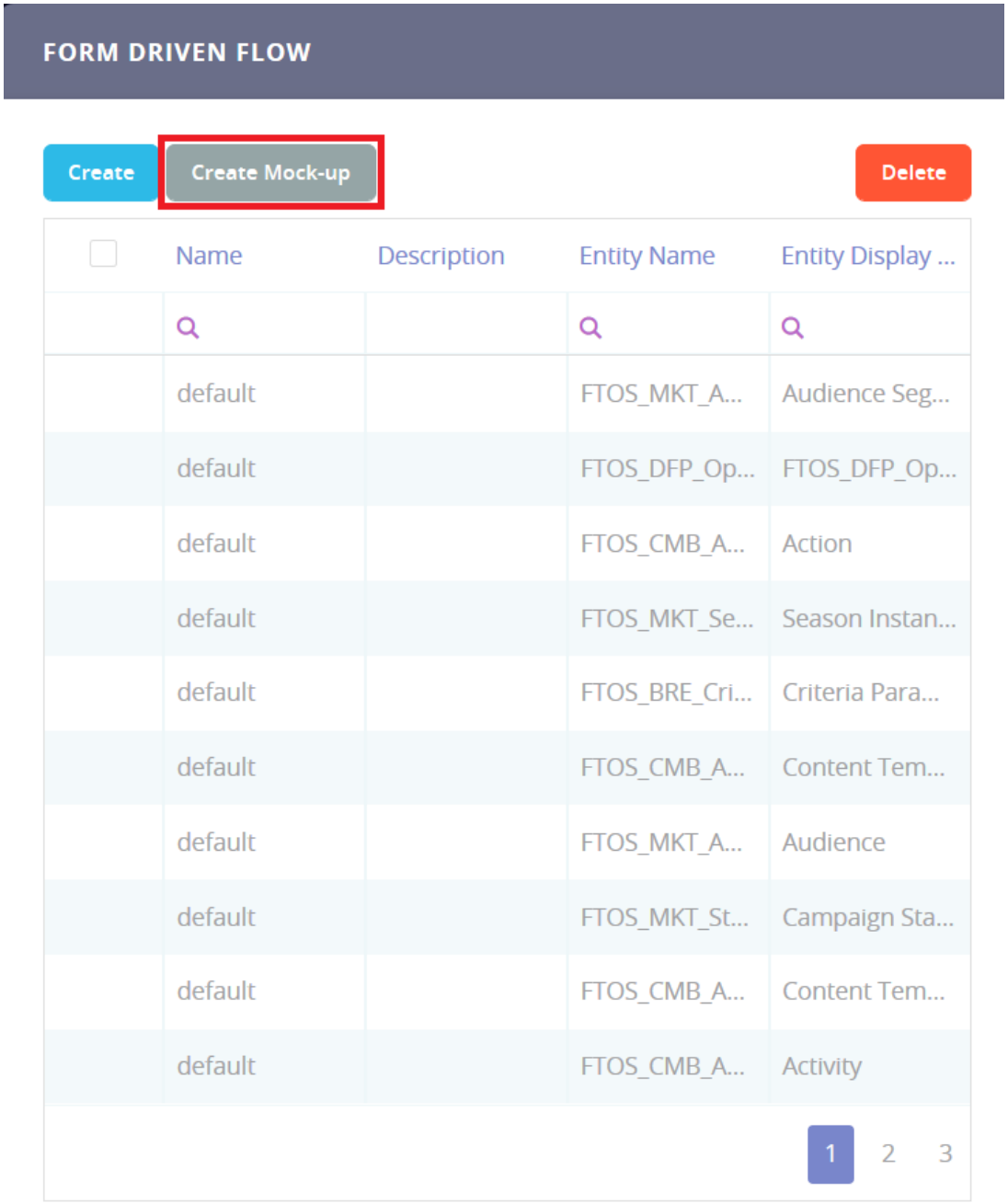
Form driven mock-up flows allow you to design a form driven flow without having an underlying data model. This lets consultants & developers to quickly define the general layout of the user interface. Developers can then attach a data model to the mock-up, map entity attributes to the corresponding form fields, and work on any additional back-end configurations.

How to create a form driven mock-up flow **267**
How to display a form driven mock-up flow **269**
How to convert a form driven mock-up flow into a regular form driven flow **270**

How to create a form driven mock-up flow

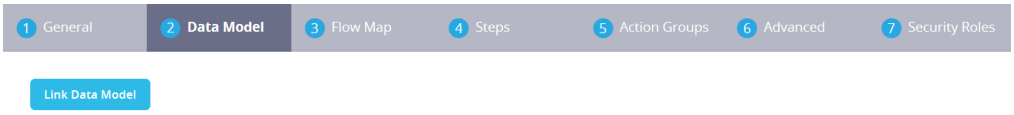
1. Open Innovation Studio.
2. From the main menu, select **Digital Journeys** > **Form Driven Flows**.

3. In the Form Driven Flow screen, click **Create Mock-up**.



4. Follow the instructions in the "Form Driven Flows" on page 203 chapter to create a form driven flow mock-up. The following exceptions apply:

- The Data Model tab contains only a button called **Link Data Model**. This button is used to attach the flow to a business entity, once you are ready to turn the mock-up into a regular flow.

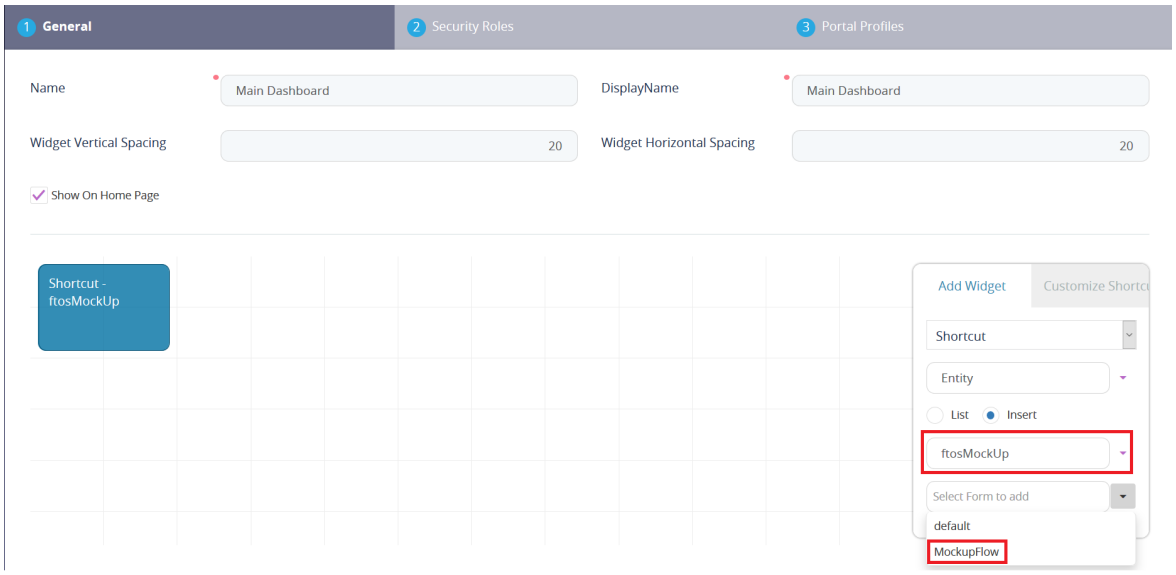


- The UI, Field Options, and Filtered Fields tabs are not available in form driven flow mock-ups.
- The UI Designer allows you to type in any attribute labels and values. The UI Designer allows you to type in any attribute labels and values. Specific for mock-ups only, it is possible to add a custom list without creating a option set. Add the records from the list separated by a comma, otherwise it will not work.

How to display a form driven mock-up flow

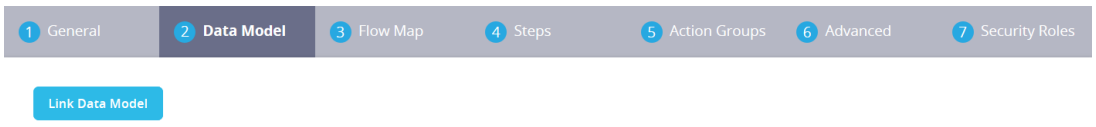
To display a form driven mock-up flow in a digital experience portal, follow the instructions in the "Digital Experience Portals" on page 552 chapter.

Form driven mock-up flows are always attached to the **ftosMockUp** entity. In other respects, they behave like regular form driven flows.



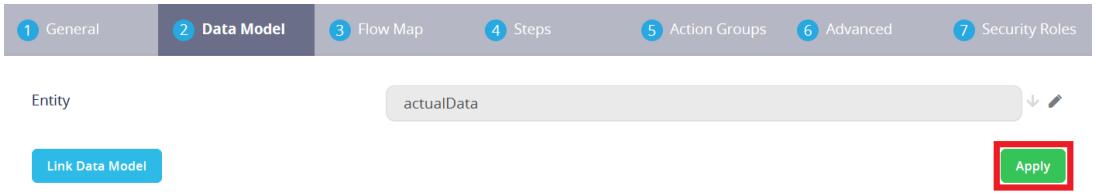
How to convert a form driven mock-up flow into a regular form driven flow

1. Create the corresponding business entity your flow will attach to. For details, see "Evolutive Data Core" on page 30.
2. Open the form driven mock-up flow in Innovation Studio and, in the Data Model tab, click the **Link Data Model** button.

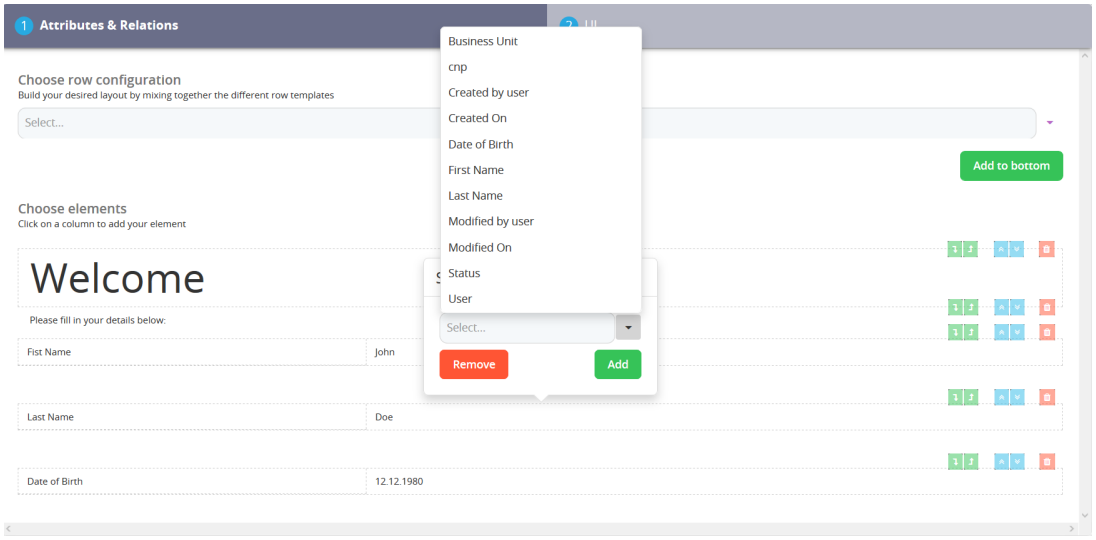


3. Select the business entity you created at step 1.

4. Click **Apply**.



5. You now have a regular form driven flow. In the UI Designer, you can now replace the dummy fields with actual attributes from your business entity.



Custom Flows

A custom flow is an ordered collection of components which address a singular need in the direction of componentization. It represents a business flow that can be the base for a digital journey, but it is not associated with an entity. It is an implementation of a business sub-process that addresses a single business need of the process.

Innovation Studio enables you to create custom flows and use them in the following the following business cases:

- to create custom URLs which redirect the user to a specific data form or view.
- to generate custom filtered views based on security roles.
- to add buttons which trigger specifics actions.

Differences between the Form Driven Flow, Custom Flow and Digital Journey.

Feature	Form Driven Flow	Custom Flow	Digital Journey
Associated to an entity	Yes	No	Yes
Has steps	Yes	No	Yes
Before and after actions	Yes	No	By accessing the Form Driven Flow
Uses source code editor	Yes	Yes	By accessing the Form Driven Flow
Uses buttons with endpoint	Yes	Yes	By accessing the Form Driven Flow
Flow map	Yes	No	By accessing the Form Driven Flow
Digital Journey Map	No	No	Yes
Accesses UI Designer	Yes	Yes	By accessing the Form Driven Flow
Invoked Ad-hoc	No	Yes	No
Uses mock-up steps	Yes	No	No

This section covers the following topics:

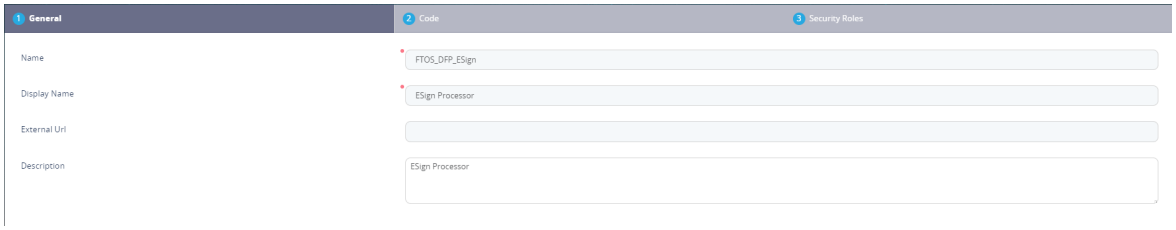
Creating Custom Flows

This section walks you through the steps that you need to follow to create a custom flow:

STEP 1. Provide custom flow general information

From the menu, click Digital Journeys > Custom Flows. The Custom Flows page appears.

At the top-left corner of the page, click the Create button. The custom flow configuration is displayed which consists of three sections (tabs). The configuration page is displayed by default on the General tab.



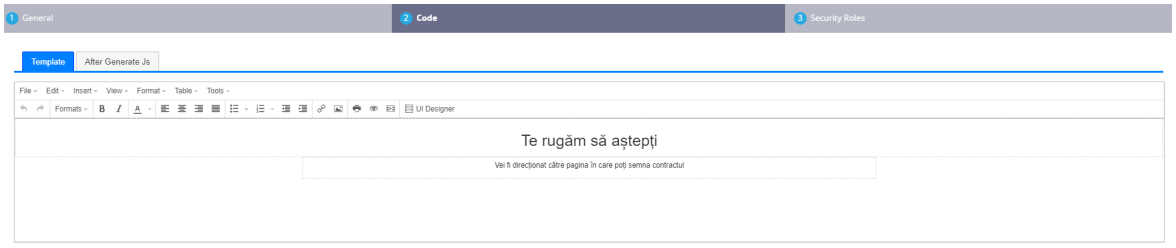
Fill-in the fields based on your needs:

Field	Description
Name	The name of the custom flow to be used by the system. The field is mandatory.
Display Name	The name of the custom flow that will be displayed in the Digital Experience Portal. The field is mandatory.
External Url	If the custom flow redirects to an external Url, provide the URL here.
Description	Description of the custom flow.

Click the ''**Save and reload**'' button and new tabs will be available.

STEP 2. Design the custom flow layout

Click the Code tab. The Code section is displayed on the Template tab. In the Template section, provide the HTML code which defines the layout of the custom flow.



You can also create the HTML template of a form driven flow by using the Advanced Code Editor or the UI Designer.

Using custom flows you can also create custom controls. For more information, see Creating Custom Controls.

STEP 3. Define the custom flow

In the Code section, click the After Generate Js tab and provide the JavaScript code which defines the digital journey (redirect users to a specific data form or view, generate custom filtered views, add buttons which trigger specific actions).

Generate contract to be signed off by the customer and add button for the customer to sign off the document

```
var params = sessionStorage.getItem("sessionParameters");
params = JSON.parse(params);
params.fileIsBase64 = false;
var signContract = function()
{

    if(params.existingFile){
        ebs.showLoadingPanel();
        if(params.externalURL == true){
                params.settings.maskNextStepURLSuccess =
params.settings.maskNextStepURLSuccess;
        }     else {
            params.settings.maskNextStepURLSuccess = ebs.getBaseUrl
() + "/" + params.settings.maskNextStepURLSuccess;
        }
    console.log("params.settings.maskNextStepURLSuccess " +
params.settings.maskNextStepURLSuccess);
    ebs.callActionByName("FTOS_DFP_ESign_Endpoint", {params:
params}, function(result) {

        ebs.hideLoadingPanel();
        params = result.UIResult.Data.params;

        //remember to delete
        //console.log(JSON.stringify(params));

        sessionStorage.setItem("sessionParameters", JSON.stringify
(params));
        if(params.settings.redirecttoNamirialLink == true){

                window.location.href = params.site;
                } else {
                window.history.back();
```

```
                }
        });
    } else ebs.showMessage("There is no file to sign!", "error");
}
signContract();
// Upload document button
$('#btnUpload').click(function() {
        if(!params.existingFile){
                $('#fileInput').trigger('click');
                $('#fileInput').on('change', function(event) {
                var fileNotBase64 = event.target.files[0];
                params.existingFile = true;
                $('#fileInput').removeAttr('style');

                var reader = new FileReader();
                reader.onload = function(fileLoadedEvent) {
                var fileBase64 = fileLoadedEvent.target.result;
                fileBase64 = fileBase64.substring
(fileBase64.indexOf(",")+1);
                params.file = fileBase64;
                params.fileIsBase64 = true;
                };
                reader.readAsDataURL(fileNotBase64);
                });
        } else ebs.showMessage("A file was already uploaded!",
"error");
});

// Sign document button
$("#btnSign").click(function(e) {
                signContract();
});
```

To have a custom flow rendered on a data form-driven journey, you should include an empty div within the Monaco editor: `<div> </div>`.

STEP 4. Define who has access to the journey

If your business case requires that the custom flow is available to designated roles within your organization, click the Security Roles tab and add the security roles who should have access to them. If no security roles are added here, all users will be able to view the journey. For more information, see "Creating Security Roles" on page 622.

STEP 5. Save the journey

If you want to save and close the journey, at the top-right corner of the page click the Save and close icon.

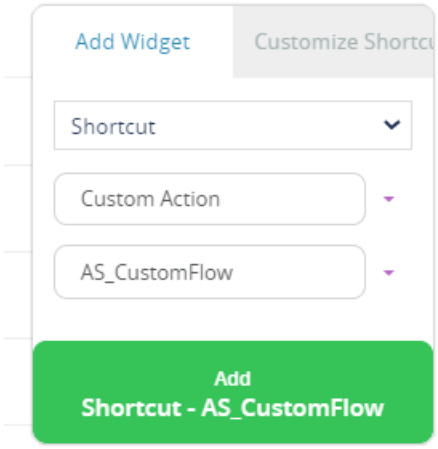
If you want to save the journey and continue working on it, click the Save and reload icon.

STEP 6. Display the Custom Flow on the Portal

To add the flow to the **dashboard**, complete the following steps:

1. Navigate to the main menu, open the Digital Frontends, select the Digital Experience Portals.
2. Click on the Dashboards and from the list choose the dashboard name where you wish to place the flow.
3. The layout of the dashboard will open. Navigate to the right panel where there is a control table which adds the widgets.

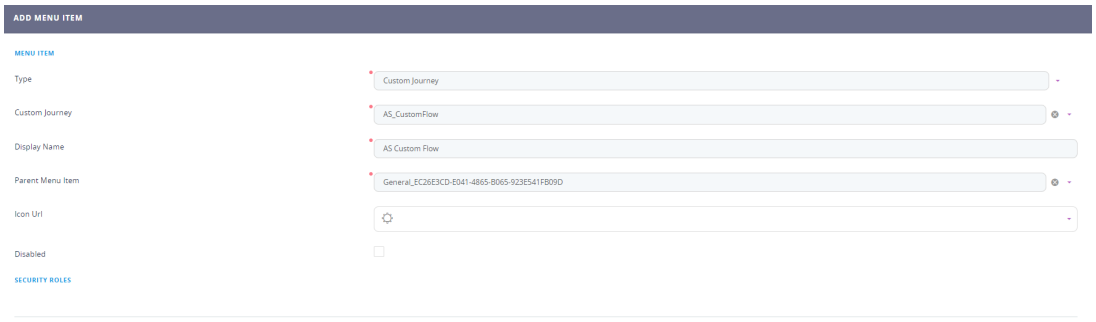
> **HINT**
>
> If the dashboard does not have the checkbox ''Show on homepage'' equal true, then the dashboard will not be shown in the FintechOS Portal. Thus, any widget added to that dashboard will not be displayed.



4. Select the type ''Shortcut'', followed by the ''Custom Action'', and lastly the entity.
5. Click on the button which says ''Add Shortcut-EntityName''.

To add the flow as a **menu item**, complete the following steps:

1. Navigate to the main menu, open the Digital Frontends, select the Digital Experience Portals.
2. Click on the Menu Items. Click on ''Insert''.



3. The ''Add menu item'' page will open. Fill in the following:

Field	Description
Type	Select from the list: • Entity • **Custom journey** • Report • Menu section.
Custom Journey	This field will open once you have chosen the type. Select the flow you wish to add.
Display Name	Insert the name of the flow.
Parent menu item	This field is optional, only if you are adding the flow inside another menu item. It shows the name of the menu item inside where the flow will appear.
Icon URL	Select the desired icon.
Disabled	Tick this checkbox if you wish to disable it.

4. Click the ''Save and reload''. The Edit menu item page will appear.

5. Add security roles by clicking the ''Insert existing.''

Creating Custom Controls

You can create custom controls you can use in your custom flows, by using either DevExtreme widgets or jQuery.

Creating custom controls using DevExtreme widgets

Based on the control you want to add, use the corresponding DevExtreme widget. For the complete list of supported widgets, see DevExtreme UI_Widgets.

To add a custom control using widgets:

1. Add the control holder element.
2. Generate the desired control.
3. Set the value for the generated control.

```
//Add control holder element
$("table").append("<tr><td><div> Age </div></td><td><div
id="controlHolder "></div></td></tr>");

//generate desired control
$("#controlHolder").dxNumberBox({
                min: 0,
                max: 100,
                showSpinButtons: true,
                step: 1
                });

//set value for generated control
$("#controlHolder").dxNumberBox("instance").option("value", 20);

//change property
$("#controlHolder").dxNumberBox("instance").option("step", 3);

//set event listener
$("#controlHolder").dxNumberBox("instance").option("onChange",
function() {
console.log("value changed")
}
);
```

Modify Control Advanced Properties

You can modify the advanced properties of an existing control created using DevExtreme widgets. For information on the controls available properties, see the specific control documentation provided by DevExtreme.

```
//get control
var control = $("#controlHolder").dxNumberBox("instance")
//change propreties
control.option("step", 3);
control.option("onChange", function() {
    console.log("value changed")
```

```
});
control.option("visible", false);
```

Creating custom controls using JQuery

> **IMPORTANT!** We do not guarantee the Client -side methods functionality when using jQuery.

To create a custom control using jQuery, follow these steps:

1. Add the control holder element.
2. Generate the desired control.
3. Set the value of the generated control.
4. Set the event listeners.

```
createCustomControl
//Add control holder element
$("table").append("<tr><td><div> Age </div></td><td><div
id="controlHolder"></div></td></tr>");

//Generate desired control
ebs.generateGenericControl({
                AttributeTypeValue: 5
                }, "controlHolder"
);

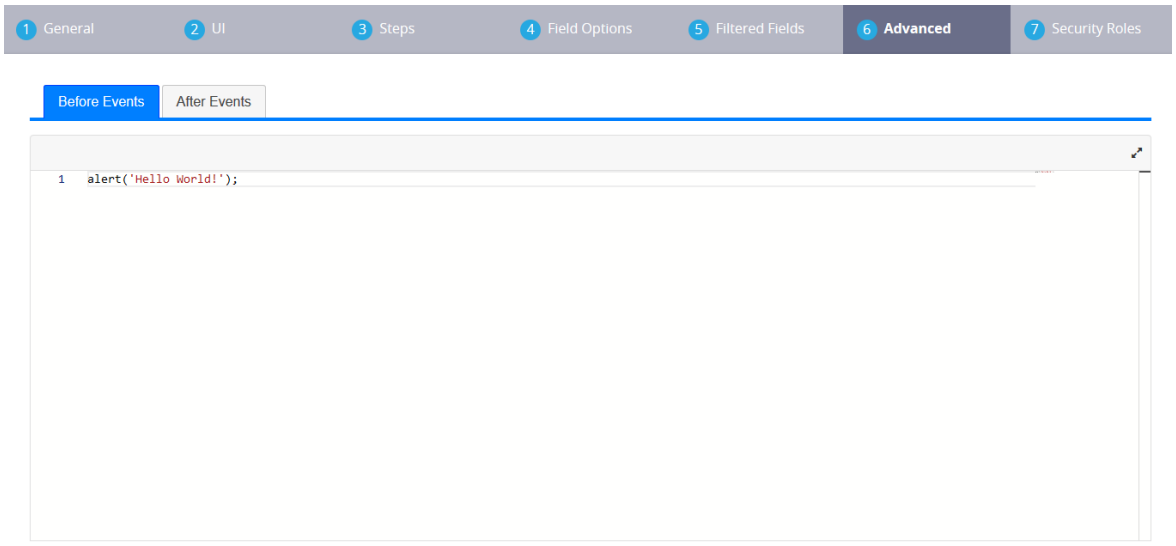
//Set value of generated control
$("#controlHolder input").val(20);

//Set event listeners
$("#controlHolder input").change(function() {
                    alert("Handler for controlHolder called.");
});
```

Code Execution Sequence

In Innovation Studio, you can attach client-side scripts in multiple places within forms, flows, and views. The script's location determines when the code is executed, for instance after a view is loaded, before a form is displayed, between two form steps, etc.

Entity Forms

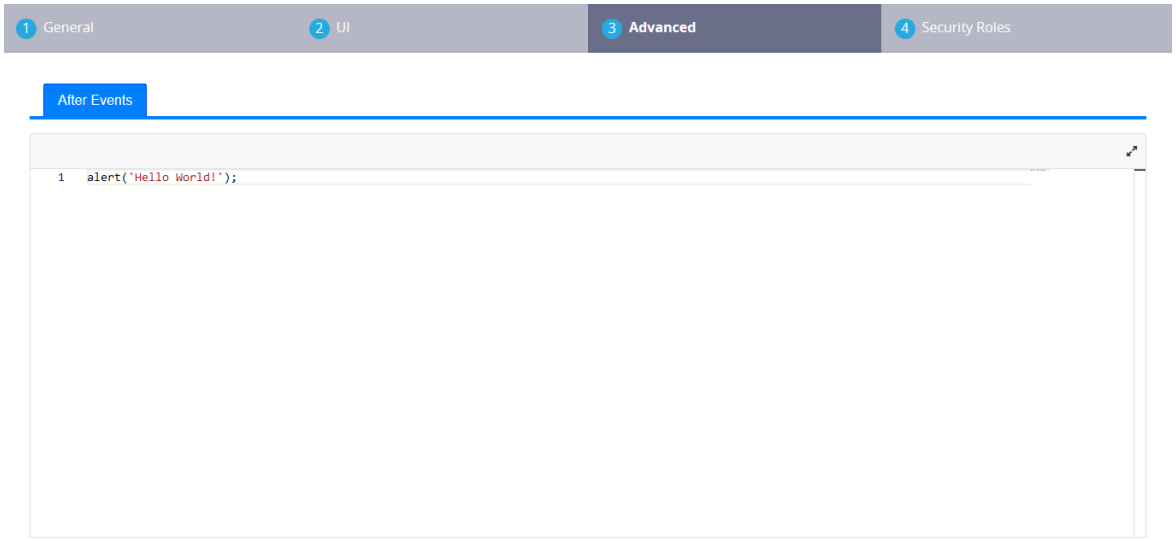


In the **Advanced** section of entity forms, there are two tabs called **Before Events** and **After Events** where you can add client code that will run when the form is loaded.

When an entity form with no steps is loaded, the code executes in the following sequence:

1. Code in the **Before Events** tab.
2. Form HTML code.
3. Code in the **After Events** tab.

Entity Form Steps



In the **Advanced** section of entity form steps, there is a tab called **After Events** where you can add client code that will run when the step is loaded.

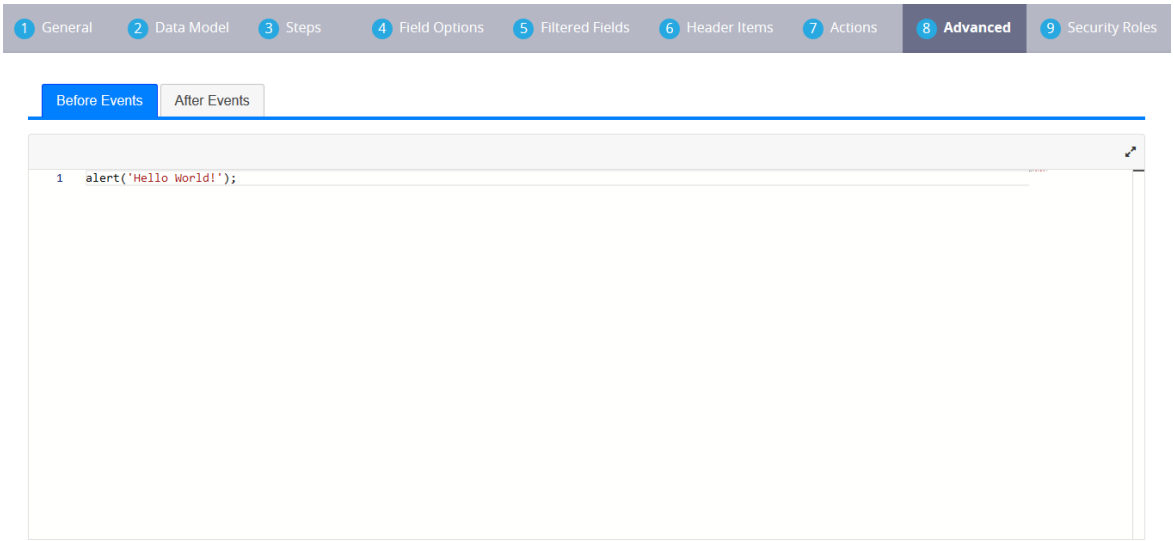
When an entity form with steps is loaded, the code executes in the following sequence:

1. Code in the form's **Before Events** tab.
2. HTML code for all steps.
3. When a step is displayed:
    4. Step HTML code.
    5. Form **After Events** code.
    6. Step **After Events** code.
7. Repeat from Step 3. for each subsequent step that is displayed.

> ⚠ **IMPORTANT!**
> Step **After Events** code is executed only the first time that the step is displayed. If, in

the entity form, you navigate away from a step and later return to the same step, the step's **After Events** code will not run again.

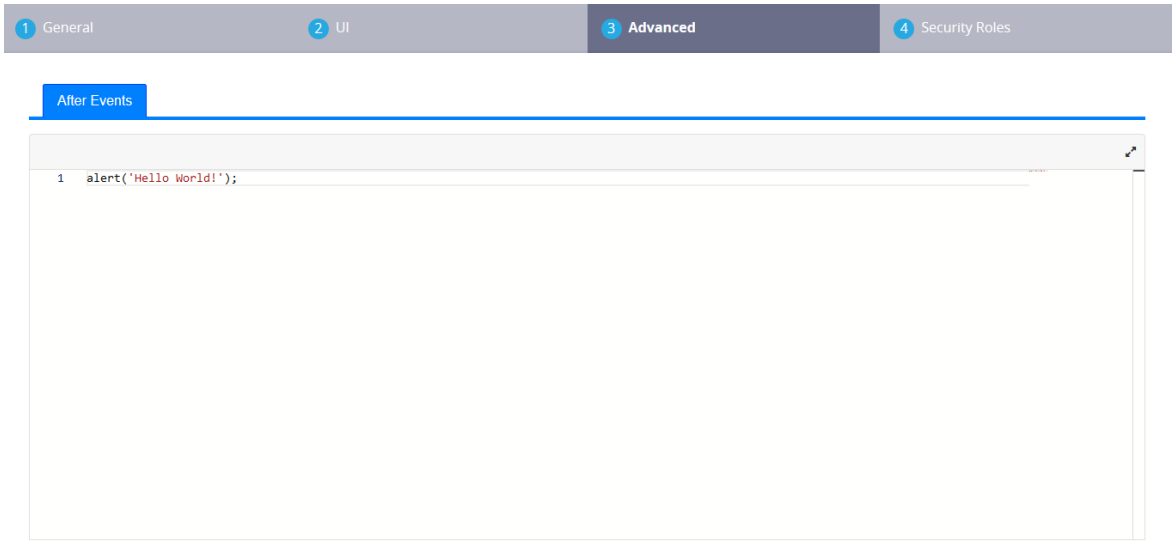
Form Driven Flows



In the **Advanced** section of form driven flows, there are two tabs called **Before Events** and **After Events** where you can add client code, that will run when the flow is loaded.

Form driven flows always include steps. Check the **Form Driven Flows Steps** section for details about the code execution sequence.

Form Driven Flow Steps



In the **Advanced** section of form driven flow steps, there is a tab called **After Events** where you can add client code that will run when the step is loaded.

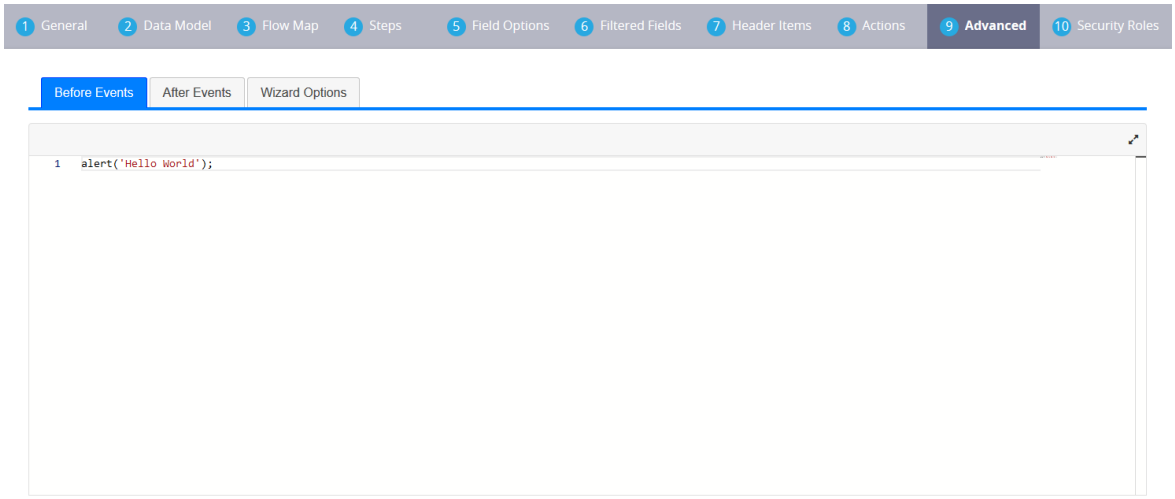
When a form driven flow is loaded, the code executes in the following sequence:

1. Code in the flow's **Before Events** tab.
2. HTML code for all steps.
3. When a step is displayed:
    4. Step HTML code.
    5. Flow **After Events** code.
    6. Step **After Events** code.
7. Repeat from Step 3. for each subsequent step that is displayed.

> ⚠ **IMPORTANT!**
> Step **After Events** code is executed only the first time that the step is displayed. If,

during the form driven flow, you navigate away from a step and later return to the same step, the step's **After Events** code will not run again.

Form Driven Flows (Wizard Mode)



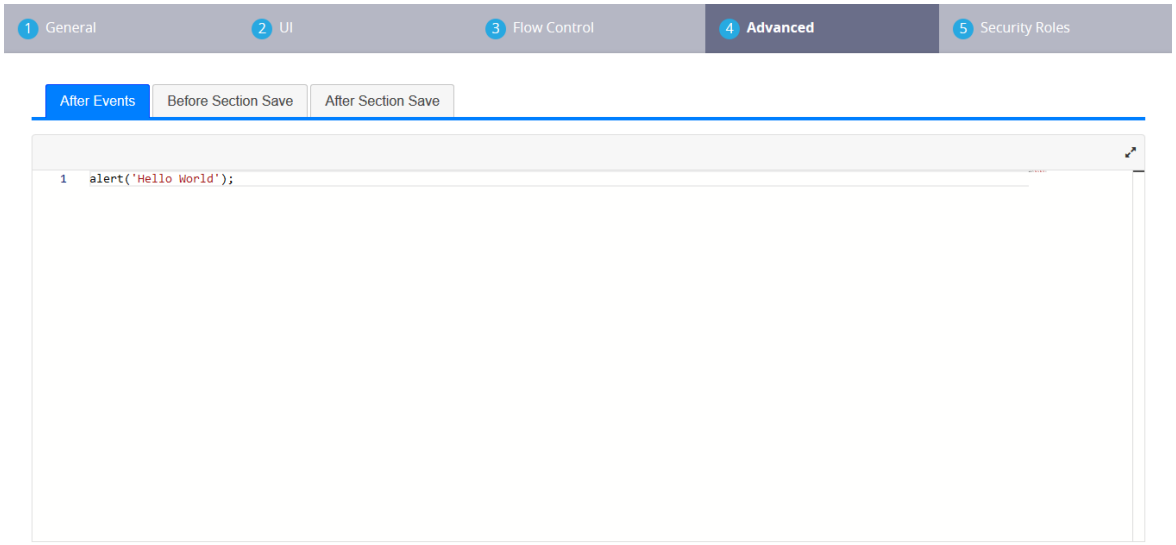
In the **Advanced** section of form driven flows in wizard mode, there are three tabs called **Before Events**, **After Events**, and **Wizard Options**, where you can add client code.

The code in the **Before Events** and **After Events** tabs will run when the flow is loaded.

The code in the **Wizard Options** tab allows you to define settings for the wizard's look and feel, such as labels and positioning for the navigation buttons, colors, or title bar. For details, see the Client SDK documentation for the ebs.createWizardObject function.

Form driven flows in wizard mode always include steps. Check the **Form Driven Flows Steps (wizard mode)** section for details about the code execution sequence.

Form Driven Flow Steps (Wizard Mode)



In the **Advanced** section of Form Driven flow steps (in wizard mode), there are three tabs called **After Events**, **Before Section Save**, and **After Section Save** where you can add client code.

The code in the **After Events** tab will run when the step is loaded.

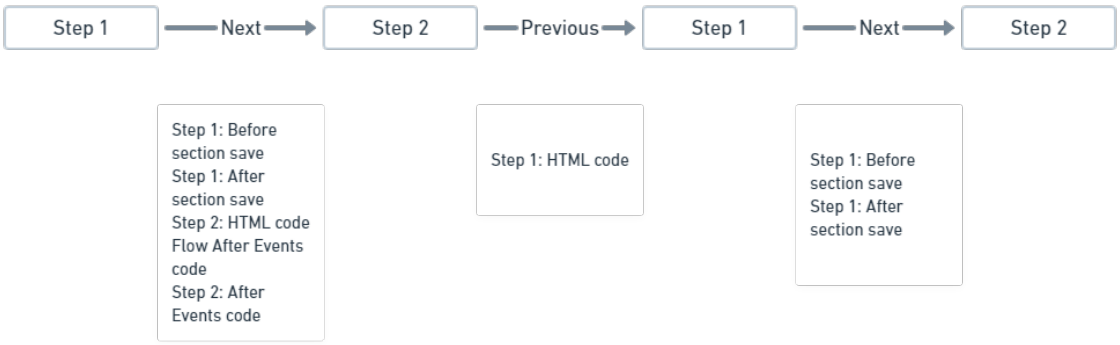
The code in the **Before Section Save** and **After Section Save** tabs will run when you advance either to the next step in the wizard (click the Next button) or, if this is the final step in the flow, you conclude the wizard (click the Finish button). This allows you to run distinct operations before and after the step's data is committed to the database.

When a Form Driven Flow in wizard mode is loaded, the code executes in the following sequence:

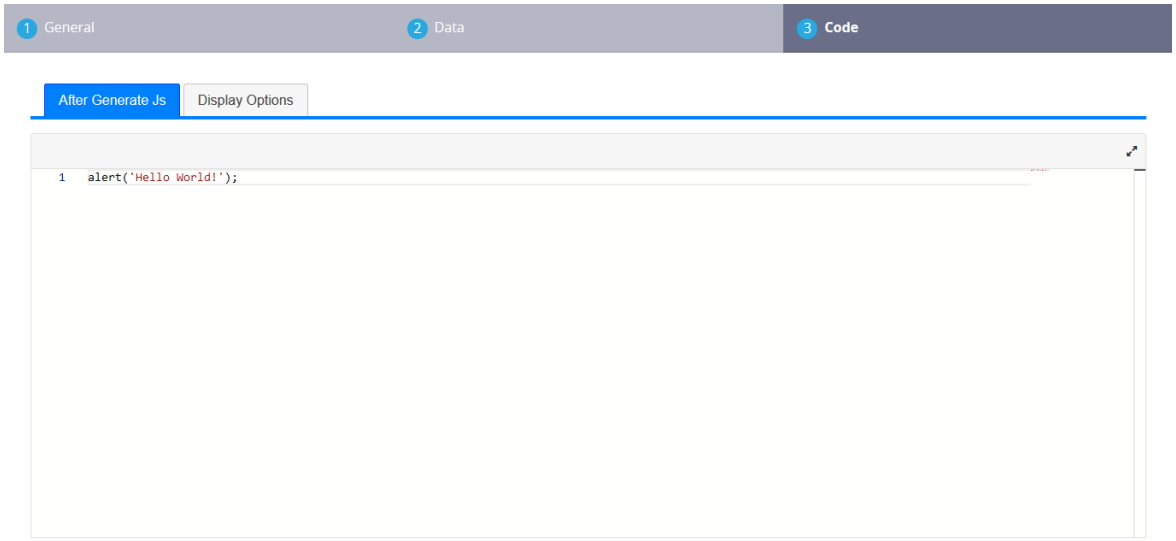
1. Code in the flow's **Before Events** tab.
2. HTML code for all steps.
3. Flow Wizard Options.
4. When a step is displayed:

5. Step HTML code.
6. Flow **After Events** code.
7. Step **After Events** code.
8. When you advance (click the Next or Finish button):
    9. Step **Before Section Save** code.
    10. Step **After Section Save** code.
11. Repeat from Step 4. for each subsequent step in the wizard.

> ⚠ **IMPORTANT!**
> When you go back to a previous step in the wizard (using the Previous button), no **After Events** or **Section Save** code (Steps 6, 7, 9, or 10) is executed. Then, if you advance again to a step that you opened previously (using the Next button), only the current step's **Section Save** code (Steps 9 and 10) is executed. The next step's **After Events** code (Steps 6 and 7) is not executed again.



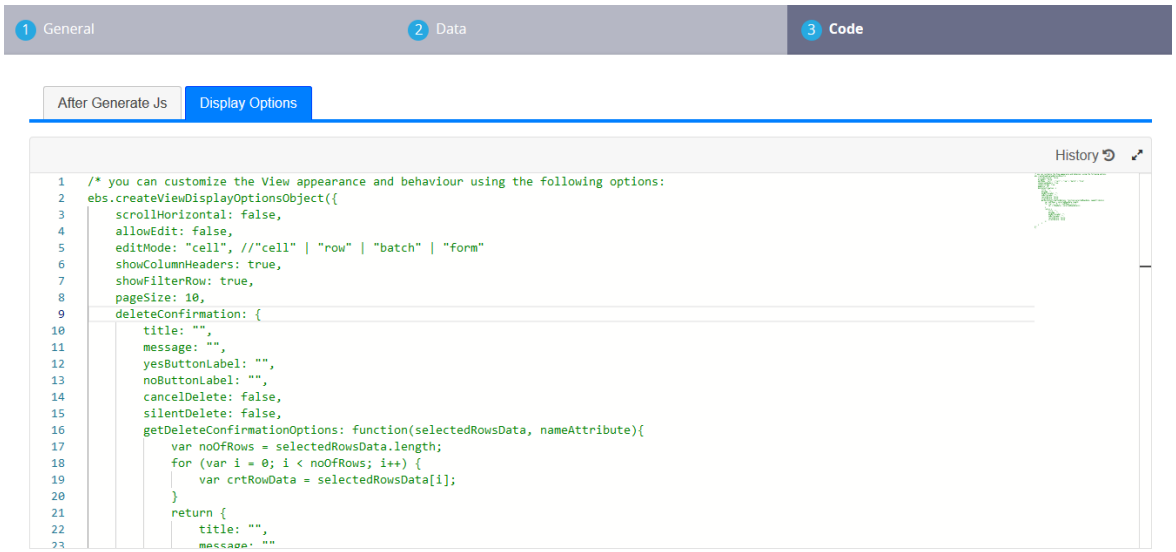
Custom Flows



In the **Code** section of custom flows, there are two tabs called **Template** and **After Generate Js** where you can add client code.

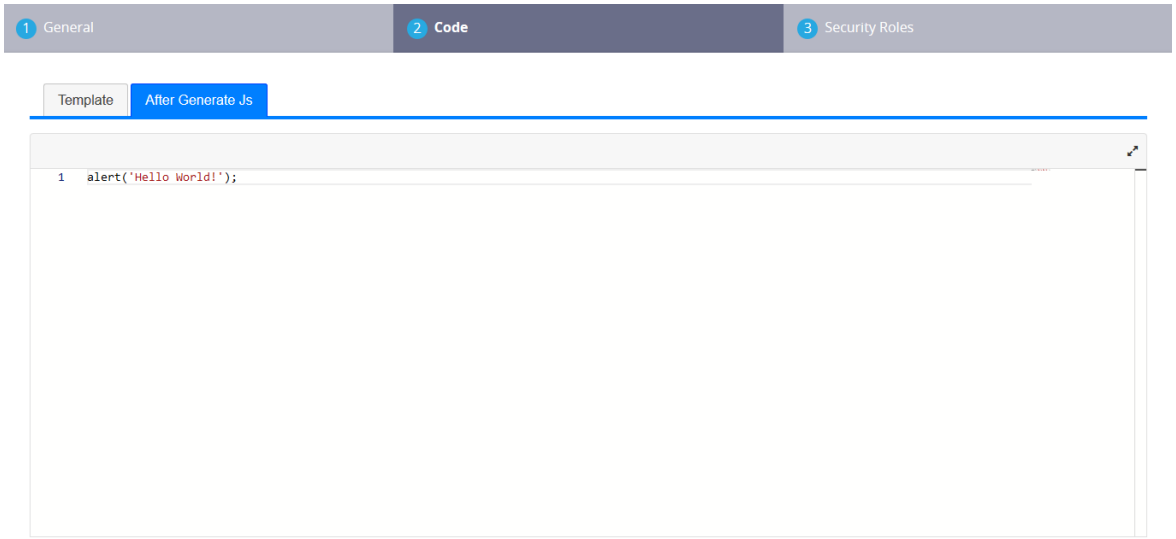
The code in the **Template** tab contains the flow's HTML code. Since this is a custom design, you must provide the form's entire content in the **Template** tab.

The code in the **After Generate Js** tab is executed after the flow's HTML code is rendered.



1. Flow **Template**.
2. Flow **After Generate Js**.

Entity Views



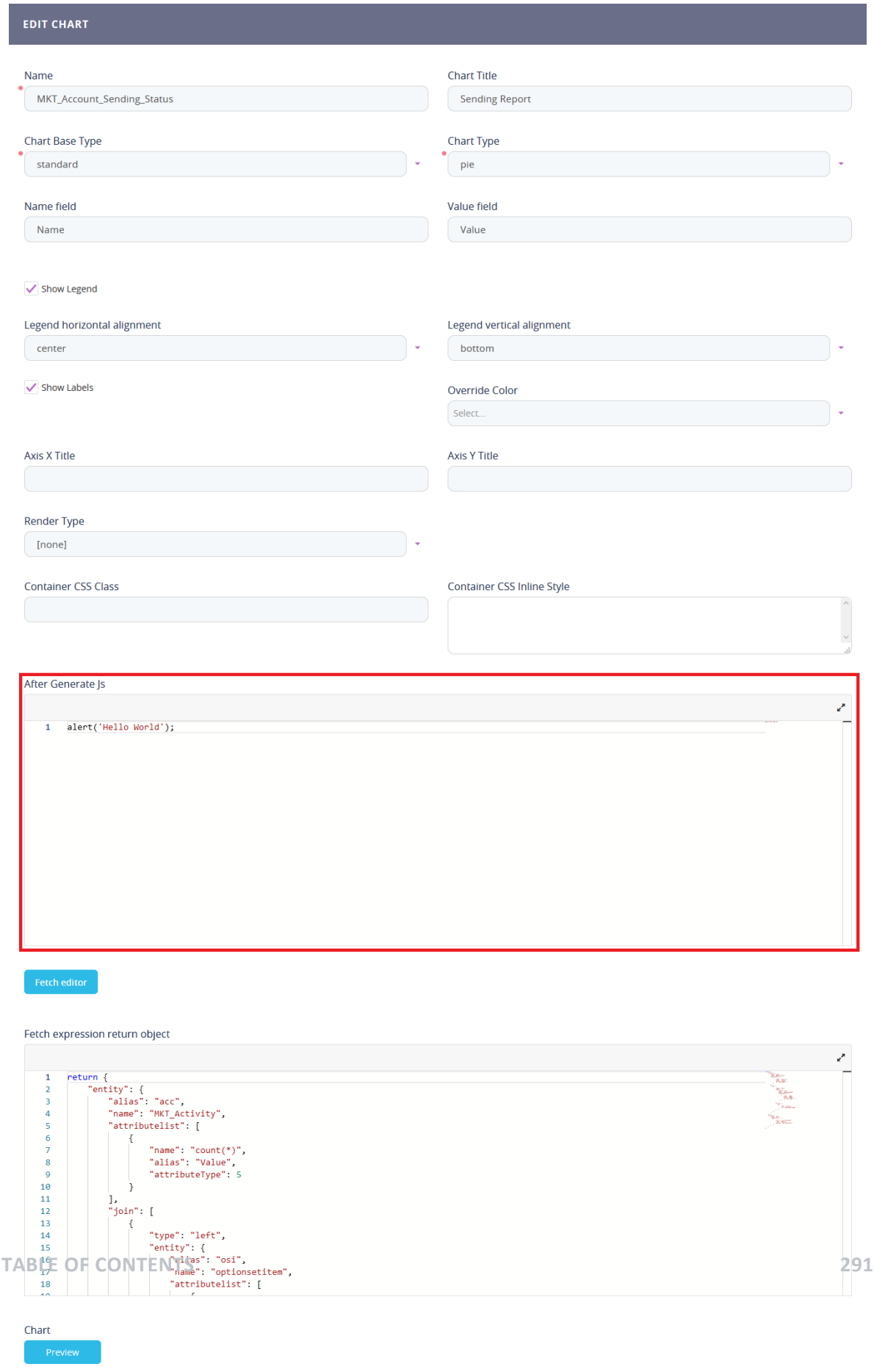
In the **Code** section of entity views, there are two tabs called **After Generate Js** and **Display Options** where you can add client code.

The code in the **After Generate Js** tab runs after the view is loaded.

The code in the **Display Options** tab allows you to define settings for the view's look and feel, such as scrolling, headers, filter row, page size, or the ability allow records editing in the view. The **Display Options** tab is pre-populated with the available settings commented out which you can modify according to your preference. These settings supplement the no-code view customizations.

1. Display view using any configured **Display Options**.
2. Run the **After Generate Js** code.

Charts


EDIT CHART
Name
MKT_Account_Sending_Status
Chart Title
Sending Report
Chart Base Type
standard
Chart Type
pie
Name field
Name
Value field
Value
Show Legend
Legend horizontal alignment
center
Legend vertical alignment
bottom
Show Labels
Override Color
Select...
Axis X Title
Axis Y Title
Render Type
[none]
Container CSS Class
Container CSS Inline Style
After Generate Js
1 alert('Hello World');
Fetch editor
Fetch expression return object
1 return {
2 "entity": {
3 "alias": "acc",
4 "name": "MKT_Activity",
5 "attributelist": [
6 {
7 "name": "count(*)",
8 "alias": "Value",
9 "attributeType": 5
10 }
11],
12 "join": [
13 {
14 "type": "left",
15 "entity": {
16 "alias": "osi",
17 "name": "optionsetitem",
18 "attributelist": [
Chart
Preview

In the chart editing windows, there is a section called **After Generate Js** where you can add client code that is executed when the chart is rendered.

Digital Journey Map

When building a digital journey, it is possible to have many form driven flows dedicated to the same digital journey, but each has a different set of steps that may be related to the other flows. In order to build visually aesthetic journeys with customer flows and operator flows, FintechOS makes it possible to see all the flows on one journey in a map.

This helps users to see all the steps, rules and the flow of the journey.

To access the map:

1. Open the Innovation Studio, open the main menu.
2. Expand the Digital Journey sub-menu, and select Digital Journeys to open the list with created journeys. From the list, select the digital journey you are interested in.
3. Click on the tab that says ''2. Digital Journey Map''. From here, it is possible to do an array of actions by clicking the buttons on the right side of the page.

Adding a flow to the map

Click on the ''**Add a flow**'' button on the left side of the screen. A new grid will appear.

Fill in the following fields:

Field	Data type	Description
Name	Text	Fill in the name.
Description	Text Area	Add a description with more details.
Type	Option set	Select the type of flow: • Form driven flow • Custom flow.
Form Driven Flow	Option set	Select the desired flow from the list.

Click the ''Save and reload'' button. Add as many flows as needed.

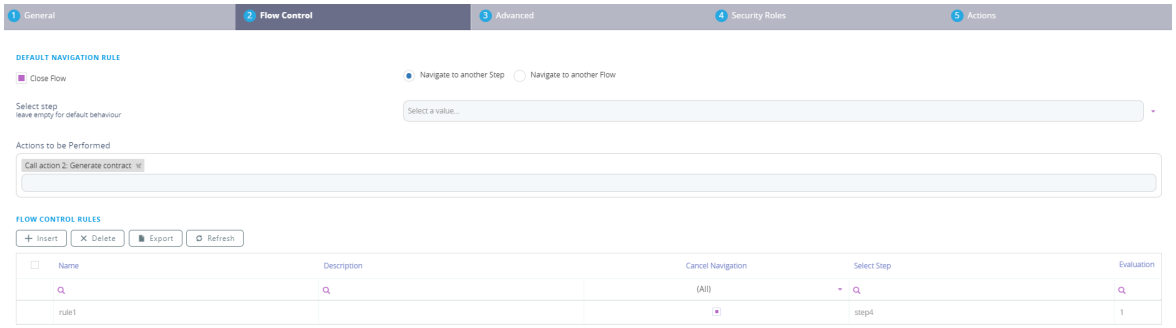
Editing a step

By clicking on a step, the following buttons activate: Details and Delete.

By clicking the Details button or by double-clicking the step, the configurations for the step are shown and it is possible to edit any needed element such as the UI, the flow and the security roles.

The Delete button will erase the step from the digital journey map.

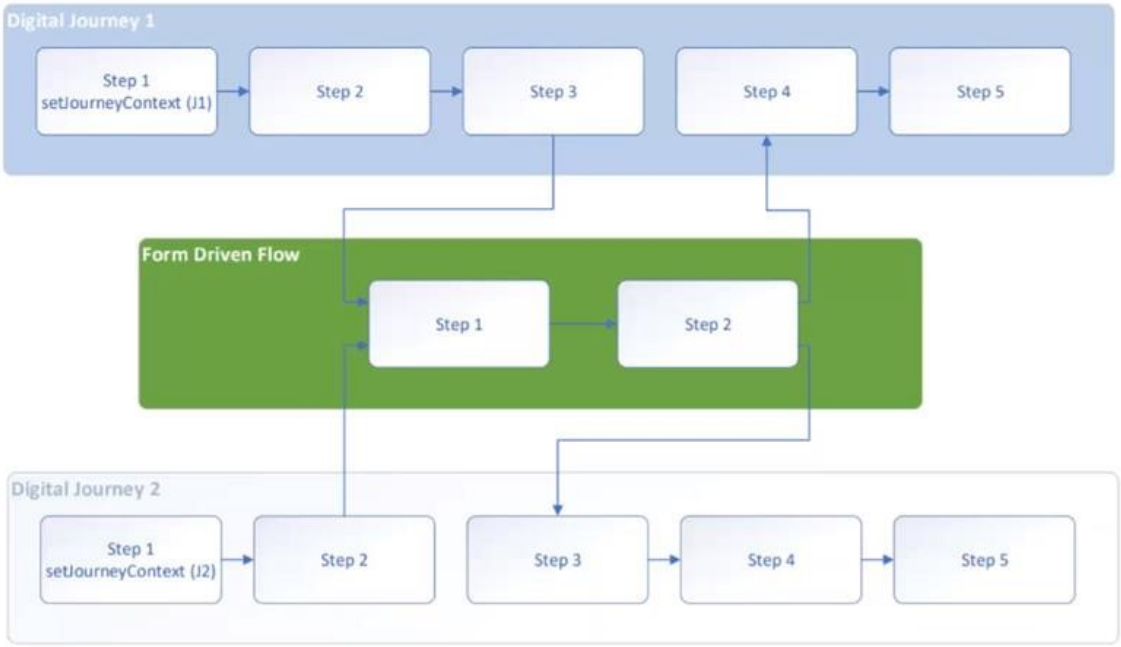
By double-clicking on a rule, the following page will open. The edit flow control rule page will allow the user to edit the rule as needed. For more details, see "Flow Control" on page 227.



The advantage is the easy navigation between steps. By visualizing the flows and each step, it is straightforward to build digital journeys with several flows which have transitions between them. Rules are too shown on the map to visualize the whole process.

Digital Journey Context

Form driven flows can be shared between multiple digital journeys. The digital journey context allows you to customize a flow's behavior based on the journey that initiated it.



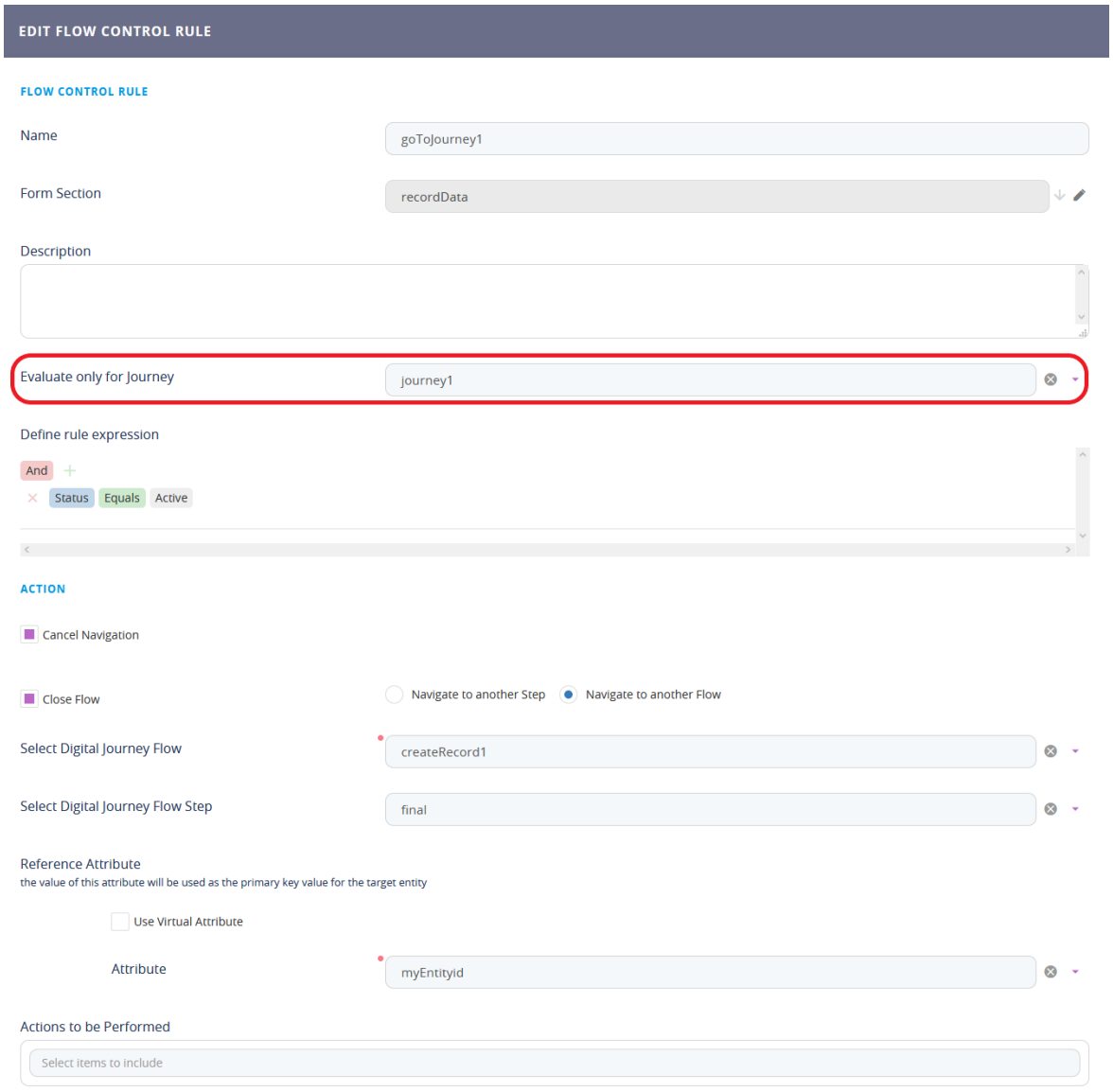
To set a digital journey context, use the ebs.setJourneyContext function on a form driven flow step (ideally on the first form driven flow step of the journey). This value will be propagated in subsequent journey steps.

To read the digital journey context, use the ebs.getJourneyContext function (typically in a downstream form driven flow that is shared between multiple journeys).

You can also use the ebs.cleanJourneyContext function to clear the digital journey context from session storage.

Apply flow control rules only for specific digital journeys

To apply a flow control rule only for a specific digital journey, use the **Evaluate only for Journey** setting in the flow control rule editor.



IMPORTANT!
In a prior step of the journey, you must use the ebs.setJourneyContext function to

⚠ set the digital journey context to the exact name of the digital journey. Otherwise, the evaluation will not work.

UI Designer

The built-in UI Designer enables FintechOS engineers to easily add HTML structures containing attributes, relations, or predefined UI elements without writing code, all from the user interface.

The UI Designer is available in the following components:

Entity	Attribute
Entity Form (entity and form driven flow)	Template
Entity Form Step (entity and form driven flow)	Template
Custom Form (custom flow)	Template
Widget	Html

You can access the UI Designer, as follows:

- For entity forms, entity form steps, and custom flows, from the **UI** tab, by clicking UI Designer from the HTML Editor toolbar,
- For Widgets, from the **Code** tab of the Digital Frontends section to the Digital Experience Portals to the menu item Widget.

When using the UI Designer on custom flows and widgets, it will have only one section, HTML.

You can also access it from an HTML field when inserting or editing entity records, but only with limited functionality. Using the UI Designer when inserting or editing records, you will not be able to:

- Change or remove attributes and relations
- Configure relations

- Add automation scripts to UI elements
- No automatically generated on click events after adding UI elements

The UI Designer contains two panels, as follows:

- the left panel contains the components, data templates and containers
- the right panel opens the settings of an element selected for the left panel.

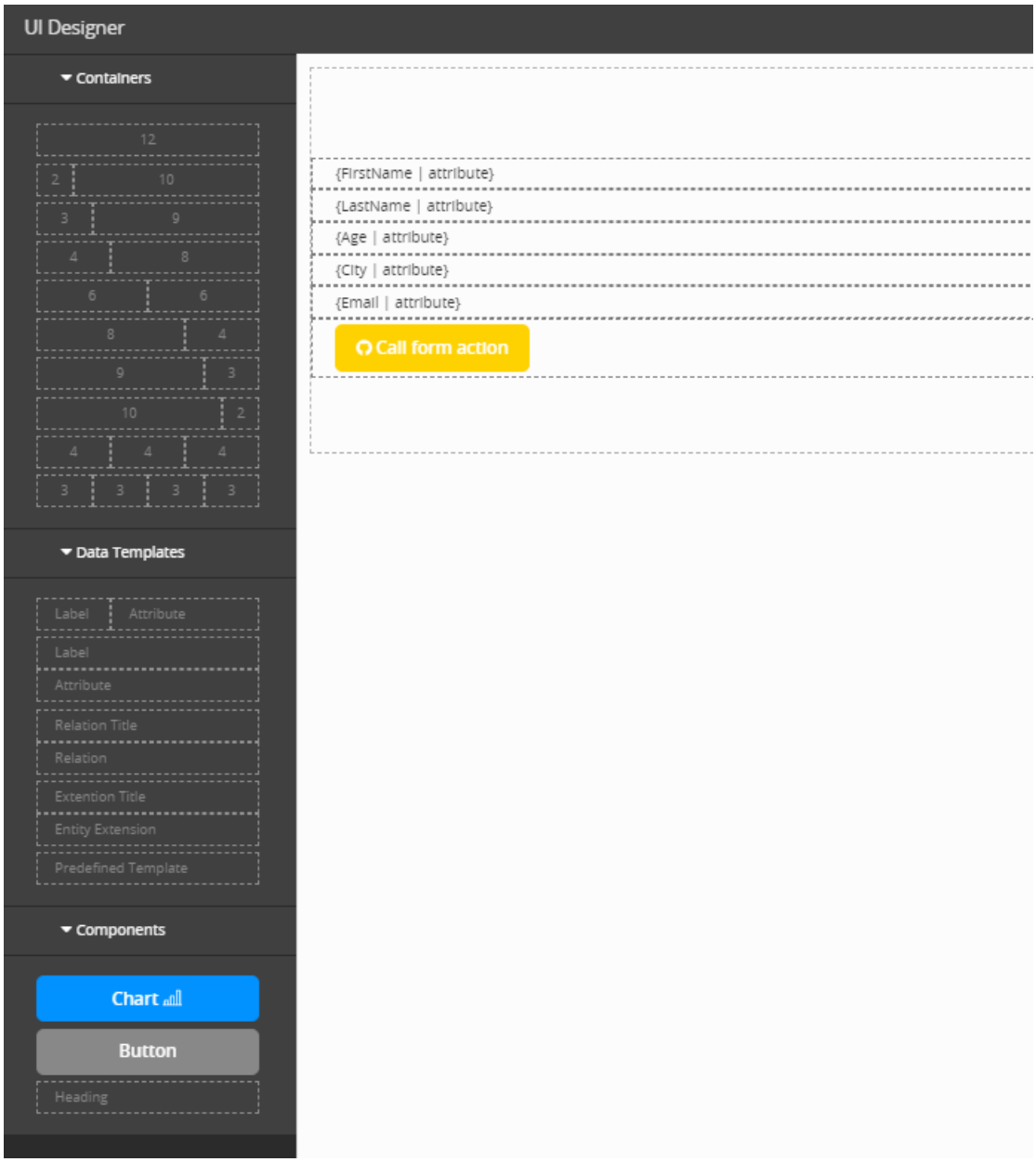
STEP 1. Define the form layout **298**
STEP 2. Add attributes **303**
STEP 3. Configure and add relations **304**
STEP 4. Working with Buttons **307**
STEP 5. Access predefined HTML Templates **318**
STEP 6. Add entity extension child collection support **320**

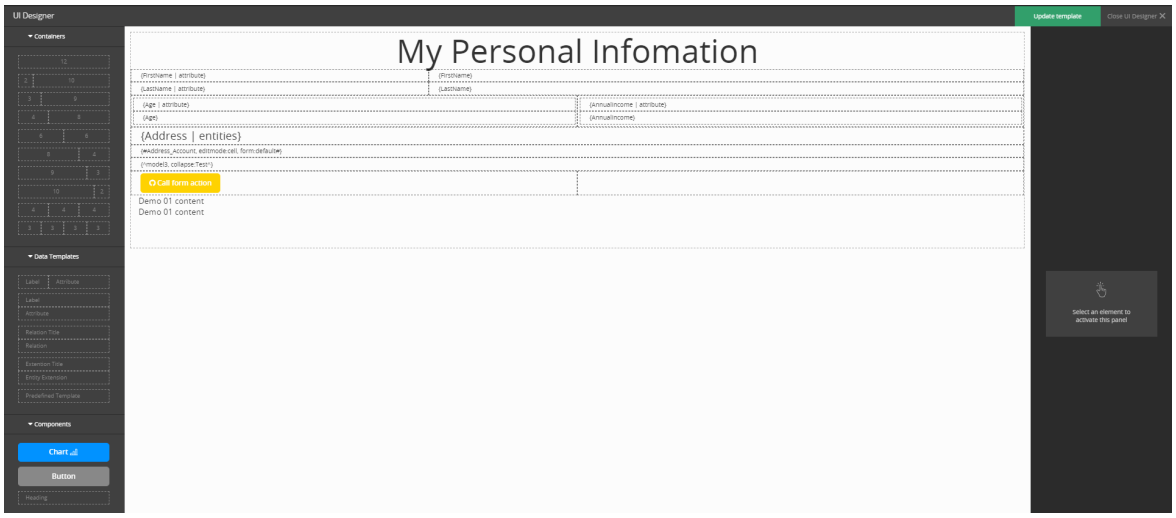
STEP 1. Define the form layout

Insert Row Templates

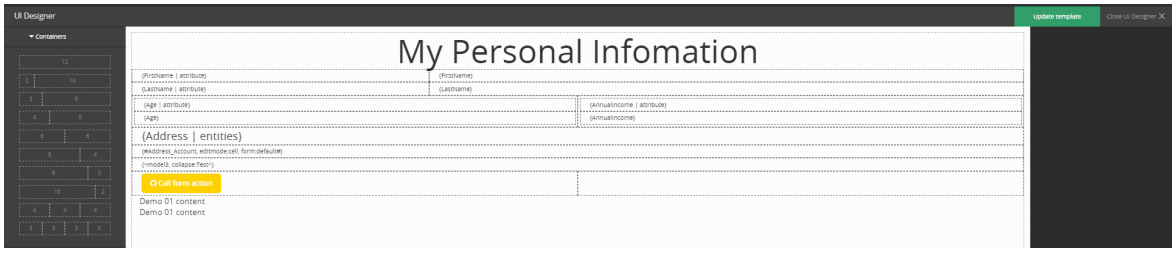
In the UI Designer, on the left panel there are the types of layout displayed. You can insert containers for entity attributes as per your needs. The left panel has:

- containers
- data templates
- components (chart, button, heading).


UI Designer
Containers
12
2
10
3
9
4
8
6
6
8
4
9
3
10
2
4
4
4
3
3
3
3
Data Templates
Label
Attribute
Label
Attribute
Relation Title
Relation
Extention Title
Entity Extension
Predefined Template
Components
Chart
Button
Heading
{FirstName | attribute}
{LastName | attribute}
{Age | attribute}
{City | attribute}
{Email | attribute}
Call form action

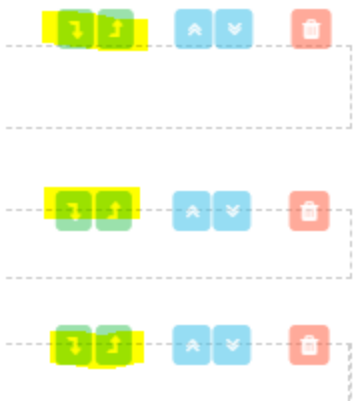


There are ten types of containers. Drag and drop the preferred layout. Inside each container it is possible to add a data template or component as in the example below.



Move Row Templates

To move a row in the grid, click the up or down buttons available on each row.



Delete Row Templates

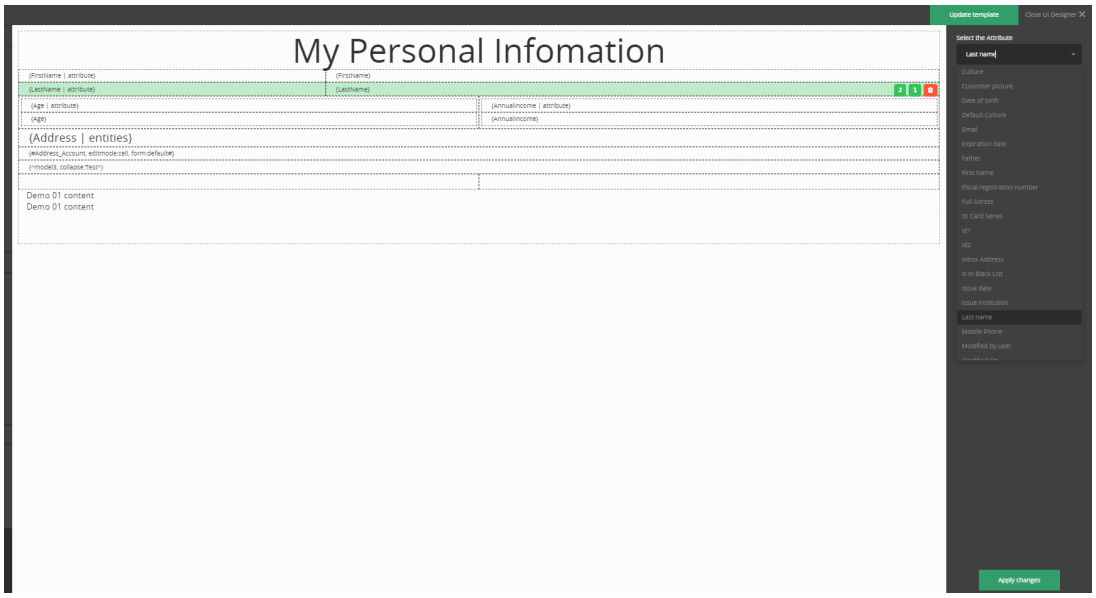
To delete a row from the grid, click the **Delete** button available on the right-side of the row. Once, you've finished building the grid, you can start adding attributes on the grid.

> **IMPORTANT!**
> Always, after exiting the UI designer, click the ''Save and reload'' button to actually save the template to the step.

Procedure protocol

1. Select the containers for the whole step.
2. Drag and drop the data templates inside each container. Click on the data template to replace with attribute/ relation/ entity extension/ predefined template. For the first three, on the left side of the screen select the actual data to go inside.

   For example, for an attribute, select Age and click the ''Apply changes'' button on the right side of the screen.



   For an relation, select Address_Account and configure the relation mode of display in the digital journey:

- edit
- collapse
- view
- Form
- Insert Form
- No header
- No filter
- No insert
- No insert for Insert Existing
- No delete
- No export
- No refresh
- No toolbar.

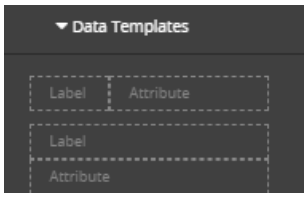
Click the ''Apply changes'' button on the right side of the screen.

For an entity extension, select Model 1 and click the ''Apply changes'' button on the right side of the screen.

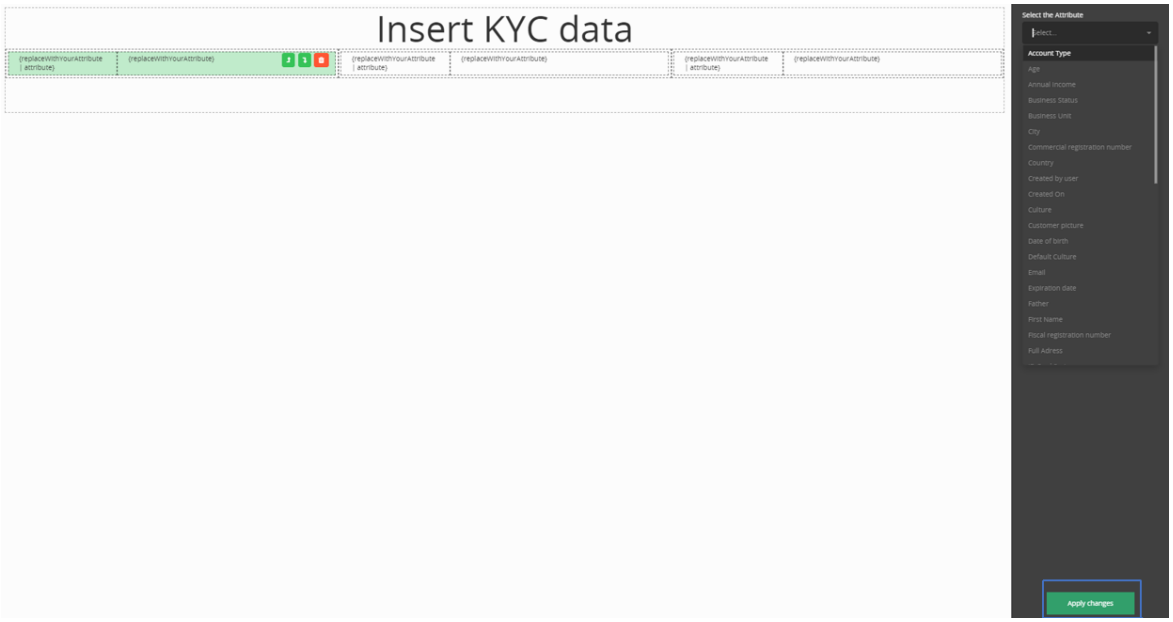
3. Insert the buttons in the containers created in set 1.
4. Insert the charts. Click the ''Apply changes'' button on the right side of the screen.
5. Insert the header. Edit the text header in the right side of the screen panel. Click the ''**Apply changes**'' button on the right side of the screen.
6. Click the ''**Update Template**'' button on the right side upper corner and then click the ''**Save and reload**'' button.

STEP 2. Add attributes

To add an attribute, in the UI designer, select one of the vertical or horizontal attribute templates and drag it to the white panel. It is possible to add attribute and their label vertically or horizontally.



From the drop-down on the right side of the screen select the desired attribute and click the **Apply changes** button.



> **IMPORTANT!**
> Always after selecting the attribute to place in the container, press **Appy changes** or the data will not be saved.

To change an attribute from a column, click on the column. From the drop-down, select the new attribute and click the **Apply changes** button.

> ⚠ **IMPORTANT!**
> Do not use the same attribute twice; otherwise errors might occur.

After setting up the attributes, click the **Update template** button to save your changes and the ''**Save and reload**'' button.

The UI Designer allows you to type in any attribute labels and values. Specific for mock-ups only, it is possible to add a custom list without creating a option set. Add the records from the list separated by a comma, otherwise it will not work.

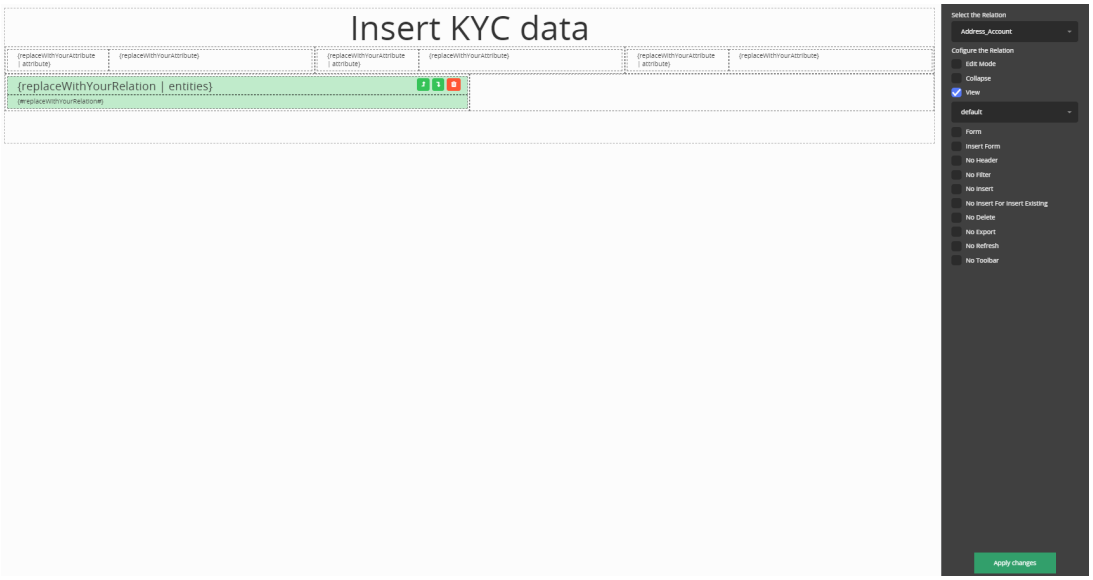
STEP 3. Configure and add relations

Prerequisites:

- You should have at least one entity with attributes, besides the entity for which you create the HTML template.
- You should add one relationship between the entity on which you define the HTML template and the other entity(ies).

1. To add relations, you first need to add a container.
2. Then from data templates add the relation. Select the desired relation on the right side of the screen.

3. The configuration for that relation will be displayed.



Configure the relation based on your needs. The table below describes the configuration options.

Option	Description
Edit Mode	Sets the mode of the inline editing on the view. Click the checkbox and from the drop-down below, select the desired options: • **cell** - allows you to edit view records cell by cell. • **row** - allows you to edit a view record by editing the cells in a row then saving the view record changes. • **batch** - allows you to edit several view records and then save the changes in batch.
Collapse	The text displayed as a collapse panel. Click the checkbox and in the field below, provide the name of the collapse panel.
View	The name of the view from another entity to be displayed.
Form	If the entity from which you render the view has multiple forms and you want a specific data form on edit directly from this view, select this checkbox and from the **Insert Form** drop-down, select the desired data form.
Insert Form	If the entity from which you render the view has multiple forms and you want a specific data form on insert directly from this view, from the **Insert Form** drop-down, select the desired data form name.
No Header	Does not display the view header.
No Filter	Does not display the view filtering / search.
No Insert	Does not display the **Insert** button on the view toolbar. **NOTE** Do not select **No Insert** when using **Form**. and **Insert Form**; otherwise, issues might occur.

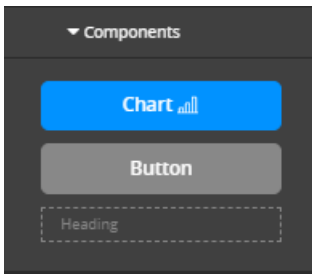
Option	Description
No Delete	Does not display the **Delete** button on the view toolbar.
No Export	Does not display the **Export** button on the view toolbar.
No Refresh	Does not display the **Refresh** button on the view toolbar.
No Toolbar	Does not display the view toolbar.

> ⚠ **IMPORTANT!**
>
> Do not intersect two relation layouts mixing them by moving relations row up and down; otherwise, errors might occur.

4. Click the ''Apply changes'' button on the right side panel.
5. Click the ''Update template'' on the top right corner of the screen.
6. Click the ''Save and reload'' button.

STEP 4. Working with Buttons

Button are added from the left side of the screen named ''Components'' by clicking and dragging the button itself to the row where the button is to be placed in a row container.

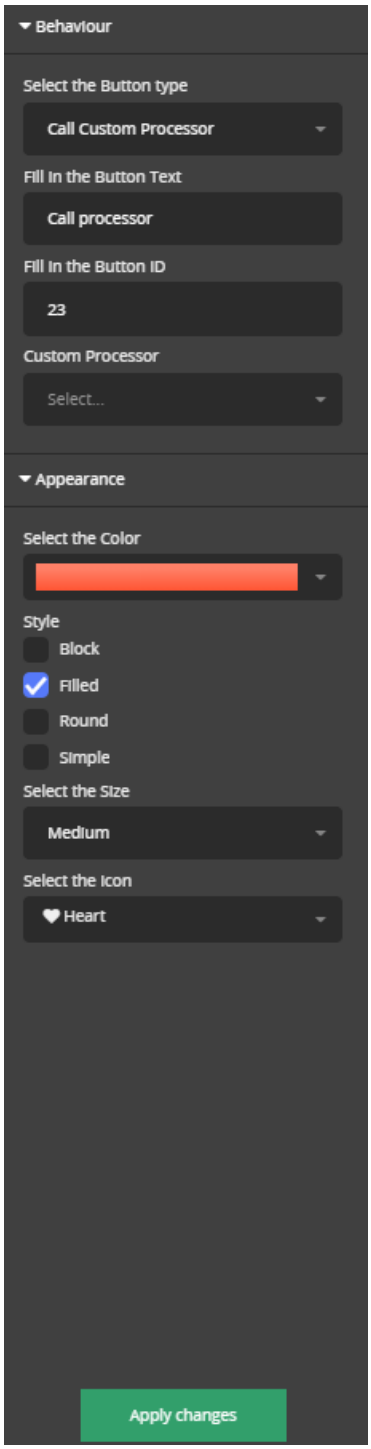


Chart

Pre-existing charts can be added to a form driven flow by dragging the button from the left-side of the panel called Data Templates to the template.

1. Access the entity form or a Form Driven Flow, in the UI section, select UI Designer.
2. In Data Templates, under the Components collapse, select the button named Charts with an indicative icon.
3. Drag and drop the Charts button in the middle section of the screen, the chart placeholder appears.
4. Click the placeholder to open the right section with a dropdown list with all the existing charts in the database.
5. Select the desired chart and click ''Apply'', then ''Update template''. The placeholder has the name of the selected chart.

The right side of the screen the panel with the following configurations opens:


Behaviour
Select the Button type
Call Custom Processor
Fill in the Button Text
Call processor
Fill in the Button ID
23
Custom Processor
Select...
Appearance
Select the Color
Style
Block
Filled
Round
Simple
Select the Size
Medium
Select the Icon
Heart
Apply changes

Field	Description
Select button type	Select one of the options: • custom • call custom processor • call form action.
Fill in the button text	Write the words to be displayed on the button for the user to read.
Fill in button ID	Unique name used to identify the button internally.
Appearance	
Select the button color	Select one of the available colours.
Style	Tick one of the options: • Filled • Block • Round • Simple.
Select the size	Select one of the options: • large • medium • small • extra small.
Select the icon	The text that will be shown on the button in the user interface.

Depending on the chosen type of button new fields will open.

Form Actions Buttons

These buttons allow you to trigger one or more form actions defined in the form. For details, see "Defining Form Actions" on page 240.

Field	Description
Select button type	Select one of the options: • custom • call custom processor • call form action.
Fill in the button text	Write the words to be displayed on the button for the user to read.
Fill in button ID	Unique name used to identify the button internally.
Select the Form action	Select the form action configured before.
Navigate to next step	If the action is triggered, and the result is true, tick the navigation to the next step.
Fill in the Form Actions Message	Once the action is triggered, display this message.
Appearance	
Select the button color	Select one of the available colours.
Style	Tick one of the options: • Filled • Block • Round • Simple.
Select the size	Select one of the options: • large • medium • small • extra small.
Select the icon	The text that will be shown on the button in the user interface.

Call Custom processor button

Fill in the following fields, to add a customer processor triggering button in the UI of a digital journey or an entity view.

Field	Description
Select button type	Select one of the options: • custom • call custom processor • call form action.
Fill in the button text	Write the words to be displayed on the button for the user to read.
Fill in button ID	Unique name used to identify the button internally.
Select the Custom processor	Select the custom processor configured before.
Appearance	
Select the button color	Select one of the available colors.
Style	Tick one of the options: • Filled • Block • Round • Simple.
Select the size	Select one of the options: • large • medium • small • extra small.
Select the icon	The text that will be shown on the button in the user interface.

Endpoint Buttons

These buttons allow you to call a predefined endpoint. For details, see "Creating Endpoints" on page 504.

Field	Description
ID	Unique name used to identify the button internally.
Label	The text that will be shown on the button in the user interface.
Endpoint	The name of the endpoint you wish to call.
MAP INPUT PARAMETERS	Select the input parameter an attribute where to mat the data.
MAP OUTPUT STRUCTURE	Select the output parameter an attribute where to mat the data. **HINT** The endpoint can be added to a Digital Journey using the "Defining Form Actions" on page 240

Custom Buttons

These buttons do not have any predefined behavior and must be configured using HTML elements or DevExtreme widgets.

Once you click the **Add** button, the form step's After Events section is opened automatically allowing you to configure the button's behavior. The section is pre-populated with a placeholder on-click event code.

```
/* Click event for the test button */
$('#test').on('click', function (event) {
    console.log("The button test was clicked");
});
```

You can modify the existing code or replace it completely with custom code, for instance using DevExtreme widgets:

```
eventHandler $("#test").dxButton({
    text: "Click me",
    onClick: function()
        {
            alert("Button clicked");
```

```
        }
});
```

NOTE

When removing a button from the template, the corresponding on-click event in the **After Events** tab will not be removed, as it might be useful in case you accidentally delete the button.

Example: Creating a Search Customer button

This example describes how to add a **Search Customer** button on a simulated digital journey for unsecured loan applications. This enables agents to check if a customer applying for a loan is an existing customer and if so, get the customer's information from the database and auto-fill in specific customer information. This example presents how to add the custom button using HTML elements.

Prerequisite: In Innovation Studio, a form driven flow has been created and customized to serve the simulation of unsecured loan applications. The journey has been configured to allow bank agents to simulate unsecured loan applications.

All steps presented below are performed in Innovation Studio.

To add a custom button that performs a custom action on user click, follow these steps:

1. Go to the configuration page of the Customer Info section . By default, it is displayed on the General tab.
2. Click the UI tab.
3. On the HTML Editor toolbar, from the Tools menu , select <> Source code.

   The Source code pop-up appears.
4. Add the div elements for PIN field and the **Search Customer** button, as follows:

```
<div class="row col-lg-12 col-md-12 col-sm-12 col-xs-
12" style="margin: 20px; background-color: white;
border: solid 1px #E0E0E0; width: 90%;">
    <h3 style="font-family: Ubuntu; font-size: 17px;
color: #662d91; font-weight: 500; margin-left: 15px;
margin-bottom: 0.5rem; margin-top: 30px; letter-
spacing: 2.4px; line-height: 1.2; text-transform:
uppercase;">Already our customer?</h3> >
    <div id="searchPIN" class="panel-body">
        <div class="col-lg-4 col-md-4 col-sm-12 col-
xs-12" style="color: #333333;">Please check it out.
It takes less than 1 minute.</div>
        <div class="col-lg-4 col-md-4 col-sm-12 col-
xs-12">
                <input
id="inputSearchPIN" class="form-
control" type="text" placeholder="Search PIN..." />
        </div>
    <div class="col-lg-4 col-md-4 col-sm-12 col-xs-12
"><span class="input-group-btn"> <button
id="getPersonalData" class="btn btn-
azure" style="pading-left: 15px;" type="button">
Search Customer</button></span></div>
    </div>
 </div>
```

5. Click **OK**. The pop-up closes.

6. Click the **Advanced** tab, then the **After Events** tab and add the click event handler for the button:

```
// Search Customer by PIN
$('#getPersonalData').click(
    function() {
        var PIN = $("#inputSearchPIN").val();
        ebs.getByQuery(
            {
            "entity": {
                "alias": "a",
                "name": "Account",
                "attributelist": [
                    {"name":"Accountid"},
                    {"name":"LastName"},
```

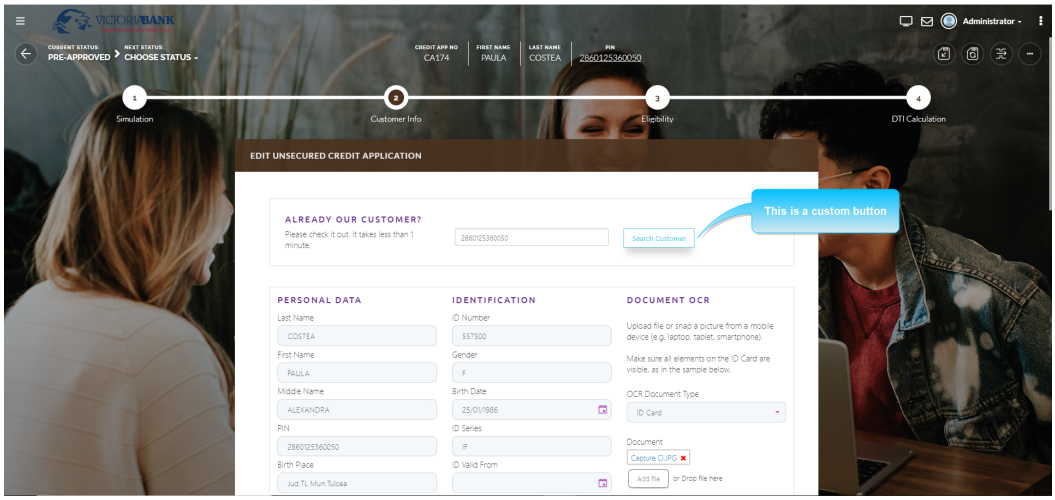
```
                    {"name":"FirstName"},
                    {"name":"UniqueID"},
                    {"name":"DateOfBirth"},
                    {"name":"PlaceOfBirth"},
                    {"name":"Age"},
                    {"name":"IdCardNo"},
                    {"name":"IdCardSeries"},
                    {"name":"IdCardIssuedBy"},
                    {"name":"IdCardIssueDate"},
                    {"name":"IdCardExpiryDate"},
                    {"name":"GenderId"}
            ]
        },
        "where": {
            "type": "and",
            "conditionlist": [
                {
                    "first": "a.PIN",
                    "type": "equals",
                    "second": "val(" + PIN + ")"
                    }
            ]
        }
    },function(e){
        if(e.Records.length > 0){
            ebs.setFormAttributeValue
("ebsContainerContent", "LastName", e.Records[0].a_
LastName);
            ebs.setFormAttributeValue
("ebsContainerContent", "FirstName", e.Records[0].a_
FirstName);
            ebs.setFormAttributeValue
("ebsContainerContent", "PIN", e.Records[0].a_
UniqueID);
            ebs.setFormAttributeValue
("ebsContainerContent", "BirthDate", e.Records[0].a_
DateOfBirth);
            ebs.setFormAttributeValue
("ebsContainerContent", "BirthPlace", e.Records[0].a_
PlaceOfBirth);
            ebs.setFormAttributeValue
("ebsContainerContent", "Age", e.Records[0].a_Age);
```

```
            ebs.setFormAttributeValue
("ebsContainerContent", "IDNumber", e.Records[0].a_
IdCardNo);
            ebs.setFormAttributeValue
("ebsContainerContent", "IDSeries", e.Records[0].a_
IdCardSeries);
            ebs.setFormAttributeValue
("ebsContainerContent", "IDIssuedBy", e.Records[0].a_
IdCardIssuedBy);
            ebs.setFormAttributeValue
("ebsContainerContent", "IDValidFrom", e.Records
[0].a_IdCardIssueDate);
            ebs.setFormAttributeValue
("ebsContainerContent", "IDValidTo", e.Records[0].a_
IdCardExpiryDate);
            }
        else{
            ebs.showMessage("The customer was not
found in our database. Please proceed to OCR.",
"warning");
            ebs.setFormAttributeValue
('ebsContainerContent', 'LastName', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'FirstName', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'BirthPlace', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'BirthDate', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'Age', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'BirthDate', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'IDNumber', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'IDSeries', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'IDIssuedBy', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'IDValidFrom', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'IDValidTo', null);
            ebs.setFormAttributeValue
('ebsContainerContent', 'PIN',
$("#inputSearchPIN").val());
```

```
                }
        });
});
```

7. Save the changes by clicking **Save and close** at the top-right corner of the configuration page.

This is what the bank agent will see in the Digital Experience Portal:



To check if the customer is an existing customer, the agent will provide the PIN and click the **Search Customer** button. If the PIN exists in the database, the customer's Personal Data is automatically filled-in.

STEP 5. Access predefined HTML Templates

There are two types for templates:

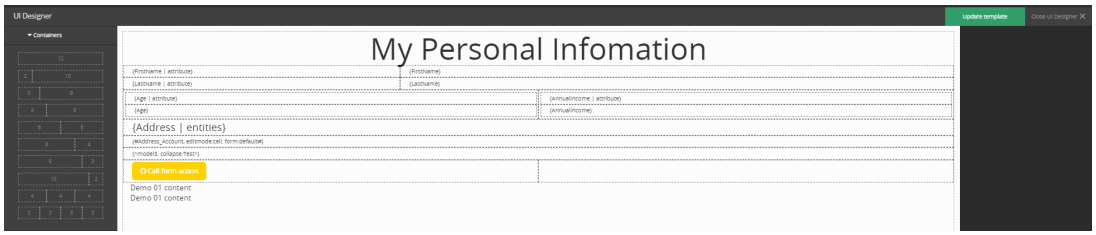
- Banking
- Insurance

> **IMPORTANT!**
> To add templates for a digital journey, the FintechOS environment needs a special

key in the web.config file with the url from the server where the templates are found to be able to access them in the Innovation Studio.

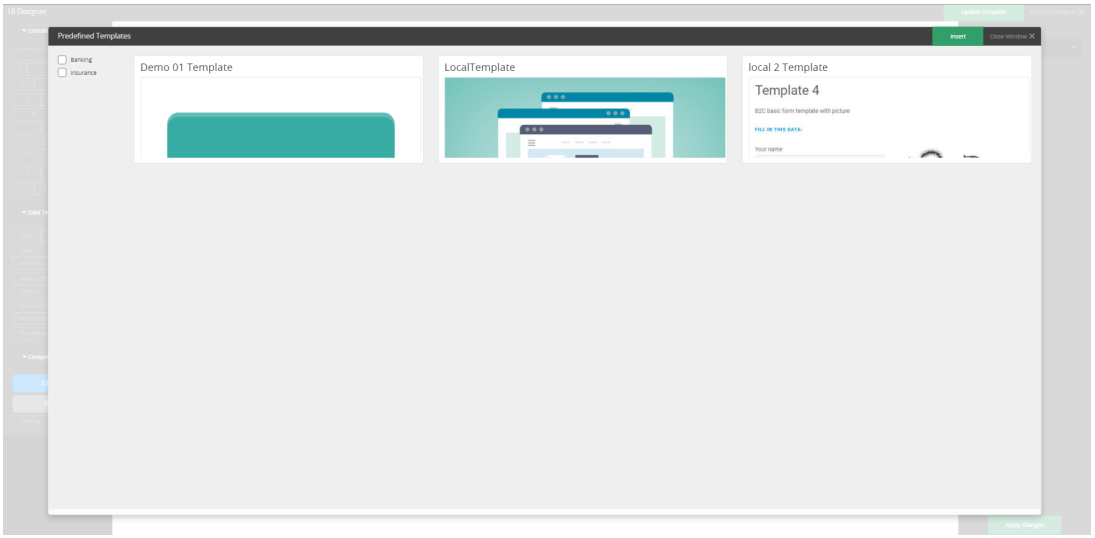
To add a template, follow these steps:

1. Open the Innovation Studio, open the main menu, select the Digital Journey submenu and open the Form driven flow or a mockup depending on what project you are working on.
2. After selecting the desired flow, open the step you wish to add the template to or create a new step where to add the template.
3. Open the UI section with the UI designer.



4. Drag the ''Predefined Templates'' on the left bottom of the page to the body of the page. Select the branch of industry you are interested in: Banking or Insurance. Select the desired template for your step. Click ''Insert'' on the right-upper corner of the page.

5. Modify the name of the attributes and select the attributes type.



6.

> **NOTE**
>
> It is possible to add multiple static templates to the same step.

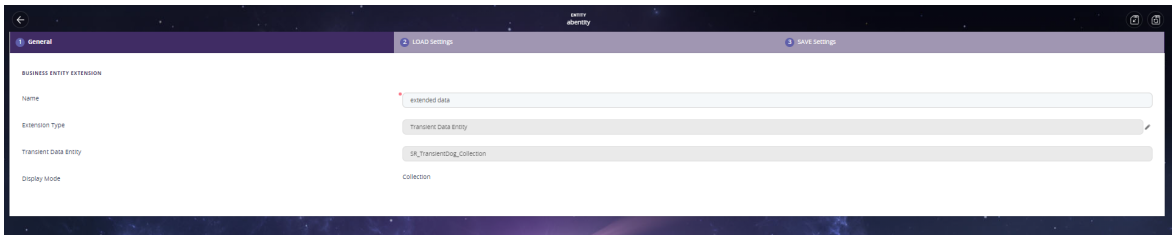
7. Click the ''Update template'' button.

8. Click the ''Save and close'' button. Repeat for any other step as needed.

STEP 6. Add entity extension child collection support

This feature makes it possible to render in the data form the extended model with a transient data entity of type collection.

The perquisite is for this feature is to access the Evolutive Data Core -> Data Model Explorer -> Business Entities List -> select the desired entity -> Extended Model. The external model needs to be linked to a extension type of transient data with the display model collection, not a single instance.



1. Open the UI designer in the data form.
2. Select the Entity Extension Container, drag and drop it.
3. Select the entity extension needed and configured in the Extended Model.
4. Select the type of view/ collapse/ no refresh/ no toolbar viewing mode. Click the ''Insert Entity extension''.
5. Click the ''Apply changes'' button.
6. Click the ''Update Template''.
7. Click the ''Save and reload'' button. Repeat for any other data form. Optionally, attach the data form to the Portal to see the result in the Portal. For more information, see "Creating Dashboards" on page 568.

Using Your Own Style Sheets

Style sheets allow you to define your own styles for forms and digital journeys for better accessibility and improved usability for your own comfort. Using style sheets, you can apply your own text style, text color, padding, etc.

Create a New Style Sheet

Creating a new style sheet in Innovation Studio is an easy task:

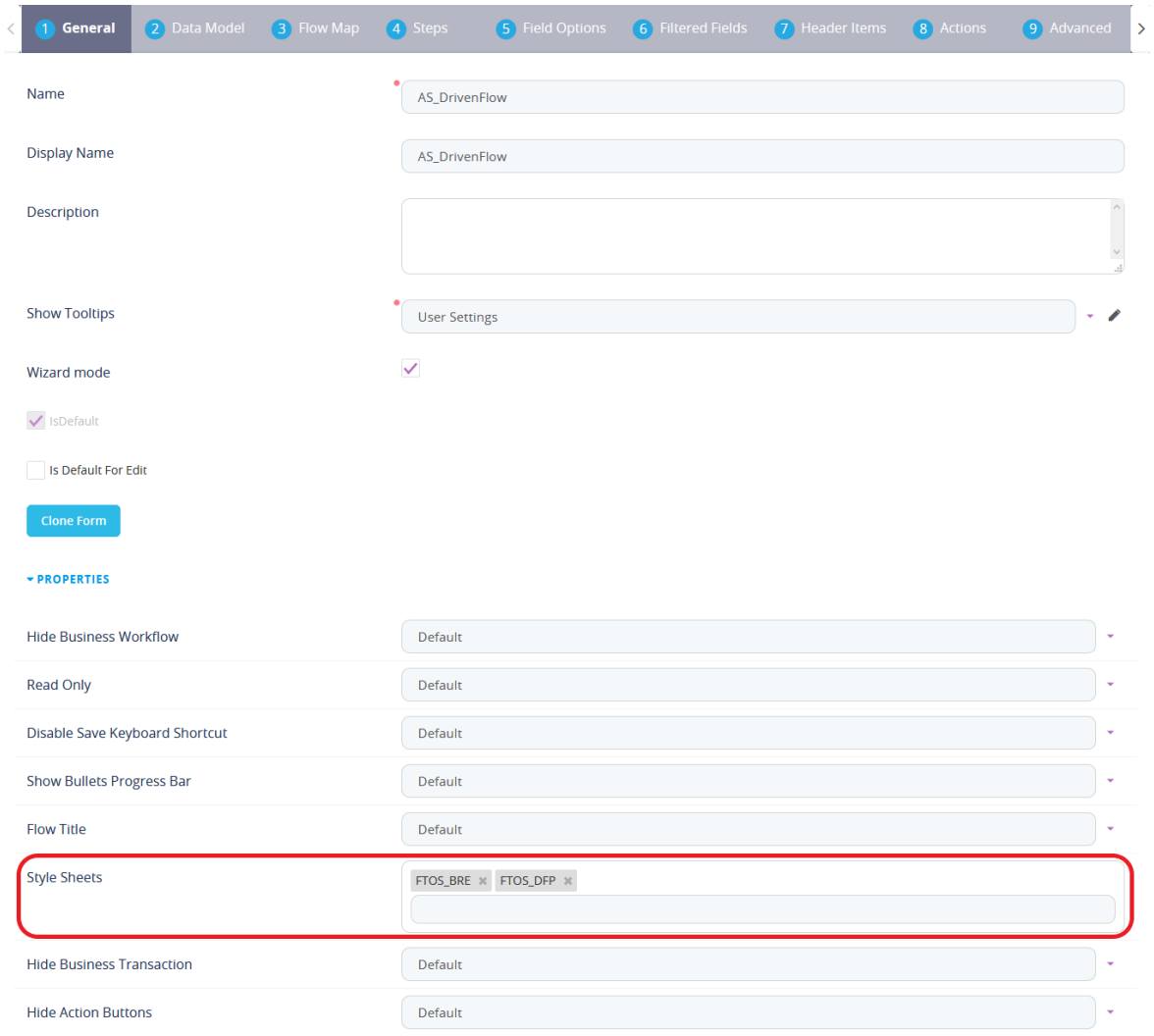
1. From the menu, click **Advanced** > **Style Sheets**. The Style Sheets List page appears.
2. At the top-right corner of the page, click the **Insert** icon. The Add Style Sheet page appears.
3. In the **Name** field, enter the name of the new style sheet.
4. In the **Code** field, provide the CSS classes that define your styles.

   > **NOTE**
   > If in the custom style sheet you reference files (e.g., images), make sure that you're using the following referencing convention **../custom/<filename>.extension** instead of **custom/<filename>.extension**; otherwise the files will not load.

   You can limit the style impact on the current form driven flow and you can also overwrite css variables. For more details, see the sections below.
5. At the top-right corner of the page, click the **Save and close** icon. The new style sheet will be displayed in the Style Sheets List page.

Apply Style Sheets (No-Code)

In the form or form driven flow's General section, use the Style Sheets property to select the style sheets you wish to apply.



You can select multiple style sheets for the same form. In this case, the order in which the style sheets will be applied matches the order in which you entered them.

Apply Style Sheets Using Code

You can apply your own style sheets to the following entities and their corresponding attributes:

Entity	Attributes
Entity Form	Before Events, After Events
Entity Form Section	After Events, After Section Save, Before Section Save

Entity	Attributes
Custom Form	After Generate Events
Entity View	After Generate Js
HtmlWidget	JavaScript

To apply your own colors and font styles you have created in your own style sheet, on the desired data form or digital journey, in the **Code** tab use the ebs.importStyleSheet method.

If you want to remove your own styles applied on data forms and digital journeys, use the ebs.removeAllImportedStyleSheets method.

For information on how to use these methods, see Attribute Control Methods.

Now you can apply and use your own style sheets. For information on how to do it, see "Manage Style Sheets for B2C User Journeys" on page 616.

Limit Style Impact to Current Form

To better qualify your selectors, use the following css class:

`form_{entityName}_{formName}`, with the spaces removed.

```
>.form_myEntity_myForm .mySelector{
        color: #f00;
        }
```

Overwriting Variables

If you want to style content based on its relationship with the parent and sibling content, use the :root selector. It enables you to target the highest-level element of the DOM, overwriting css variables.

```
:root{
        --defaultFontSize: 12px;
        }
```

Localization

The aim of localization is to give a product the look and feel of having been created specifically for a target market, no matter their language, culture, or location.

Localization comprises various elements:

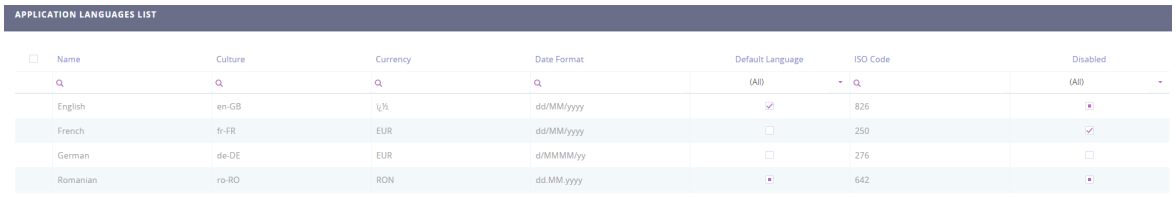
- Translation
- Design and layout to properly display translated text
- Converting to local requirements (such as currencies and units of measure)
- Using proper local formats for dates, addresses, and phone numbers
- Addressing local regulations and legal requirements

FintechOS supports the localization of static and dynamic elements, as well as the localization of customized messages and metadata.

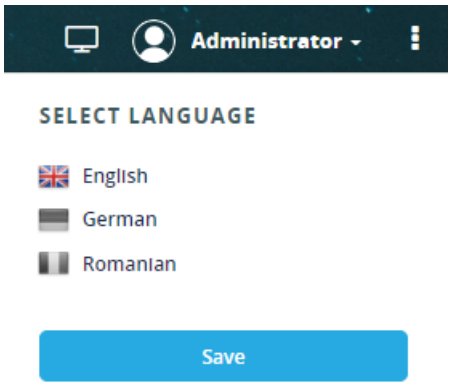
NOTEBefore localizing in a new language, you need to prepare your environment to easily identify the resources to be localized resources. For information on how to set up the localization, see the FintechOS Platform Administration Guide.

Viewing Defined Languages

To view the list of available languages, from the menu, click Admin > Application Languages The Application Language List page appears.



By default, the page lists the standard system languages: English (EN) and Romanian (RO). To use any of the standard system languages, on the right-side of the navigation bar, next to your name, click the User Settings icon and select the language that you want to use.



The page refreshes and the user interface text is displayed in the selected language.

NOTE
The **applicationLanguage** entity stores information and configuration related to the supported languages.

Adding Languages

NOTEWhen adding a new language, the system automatically creates all the additional database fields for the new language, for all the attributes marked as localizable.

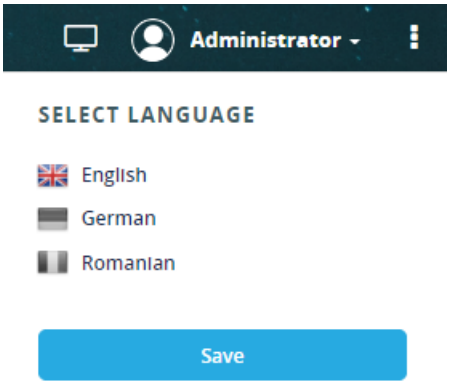
To add a new language, follow these steps:

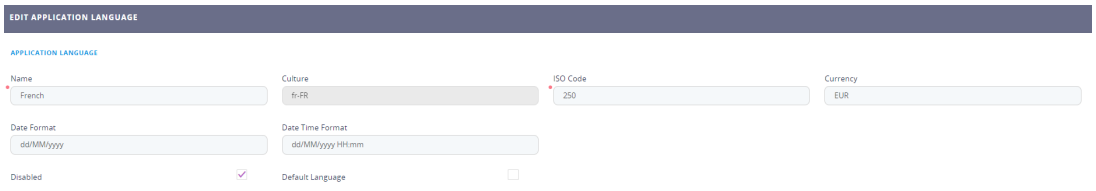
1. From the menu, click Admin > Application Languages The Application Language List page appears..

2. At the top-right corner of the page, click the Insert icon. The Add Application Language will be displayed.

3. Enter the language details. The table below provides description of the fields:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Name</td><td>The name of the language you want to add. The field is mandatory.</td></tr>
<tr><td>Culture</td><td>The .NET language culture. Make sure that you enter one from the available .NET cultures. The field is mandatory.
By default, two cultures are available: en-GB and ro-RO.</td></tr>
<tr><td>ISO Code</td><td>The ISO country code. View the list with ISO country codes.
NOTEYou must enter the ISO Numeric Code UN M49 Numerical Code; otherwise you will receive an error message when saving the new language.
Based on the country ISO Code, the corresponding flag and language name appear in the user settings.</td></tr>
<tr><td>Currency</td><td>The default currency for the specified language.</td></tr>
<tr><td>Default language</td><td>The user default language after login.</td></tr>
<tr><td>Date Format</td><td>The format of attributes of type Date Time. You can define a different format per language:
<table><tr><td>Date Attribute Format</td><td>Example Output</td></tr><tr><td>dd/MM/yyyy</td><td>05/10/2018</td></tr><tr><td>dd/MMM/yyyy</td><td>05/Oct/2018</td></tr><tr><td>dd/MMMM/yyyy</td><td>05/October/2018</td></tr><tr><td>d/MMMM/yy</td><td>5/October/18</td></tr><tr><td>MMM dd, yyyy</td><td>Oct 05, 2018</td></tr></table></td></tr>
</table>

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Date Time Format</td><td>The format of attributes of type DateTime. You can define a different format per language:
<table>
<tr><td>DateTime Attribute Format</td><td>Example Output</td></tr>
<tr><td>dd/MM/yyyy HH:mm</td><td>05/10/2018 15:25</td></tr>
<tr><td>dd/MM/yyyy hh:mm</td><td>05/10/2018 03:25</td></tr>
<tr><td>dd/MM/yyyy hh:mm tt</td><td>05/10/2018 03:25 PM</td></tr>
<tr><td>dd/MMM/yyyy h:mm</td><td>05/Oct/2018 3:25</td></tr>
<tr><td>dd/MMMM/yyyy HH:mm</td><td>05/October/2018 15:25</td></tr>
<tr><td>d/MMMM/yy HH:mm</td><td>5/October/18 15:25</td></tr>
<tr><td>MMM dd, yyyy - HH:mm:ss</td><td>Oct 05, 2018 - 15:25:00</td></tr>
</table></td></tr>
<tr><td>Disabled</td><td>True, then the language is disabled and will not appear in the list of available languages.
 
</td></tr>
</table>



4. At the top-right corner of the page, click the Save and close icon to save the language.

   If you want to save the new language and stay on the Add Application Language page, click the Save and reload icon.

> **NOTE**
> To have your app localized in a language other than default languages, you should localize all the application resources. For information on how to localize generic resources, see Localizing Generic Resources.

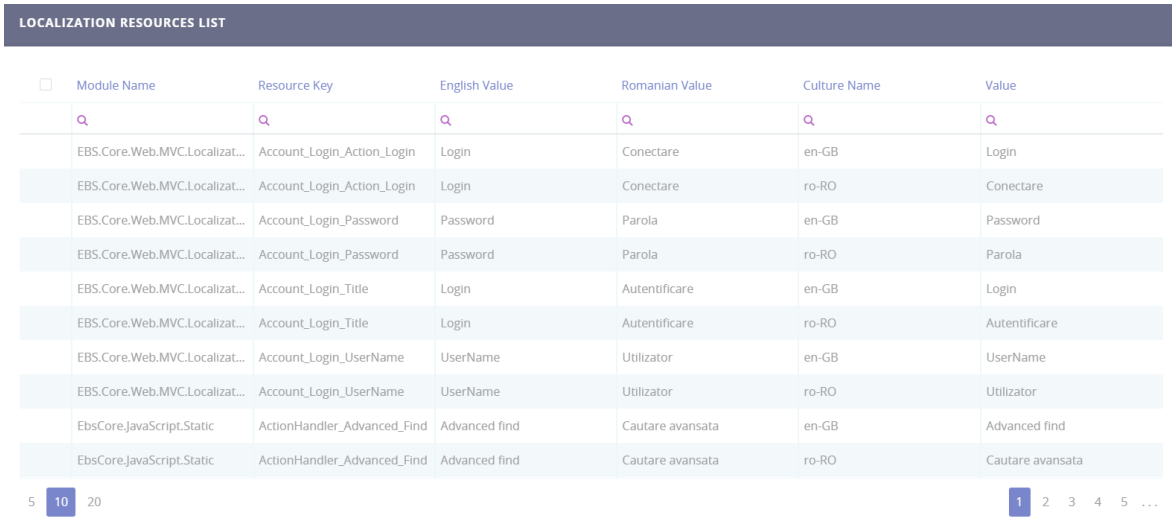
Localizing Generic Resources

By default, FintechOS provides you with a list of generic resources in both ro-RO and en-EN cultures.

This section describes how to view the list of generic language resources and how to localize them.

View generic language resources

To view the list of UI generic resources, from the menu, click Admin > Localization Resources. The Localization Resources List page appears. It lists all UI localization resources sorted automatically by module, key and culture. You can filter these resources at your convenience.



Localize generic language resource

The Value field, supports inline editing, so you do not have to open each record to edit it.

To easily localize the generic UI resources, click on the Value field corresponding to resource to be localized and type in the localized text.

Import Localized Values

To localize all generic resources available, you can use the bulk data import functionality provided by FintechOS by creating an import template that will only UPDATE the list of localizable resources list with the new values for the new or old languages inserted in the application.

To import localized values, follow these steps:

STEP 1. Export generic resources and localize them

1. From the menu, click Admin > Localization Resources. The Localization Resources List page appears.
2. At the top-right corner of the page, click the Export icon and select Export all data set.All localizable resources within the list are exported to an excel file. For details on the content of the excel file, see Data Exports.

Open the exported excel file and fill in the values translated for the desired languages, then save the file.

STEP 2. Import localized values

Import the excel file which contains the localized values following the Bulk Data Import procedure.

When creating the data import template, in the **Entity** field, select the LocalizationResource entity.

Localizing Metadata

For data localization, some of the most visible metadata entities have an additional **DisplayName** field, that is used when rendering the user interface instead of the **Name** field.

FintechOS comes with two predefined languages: en-GB and ro-RO; where en-GB is the default language.

The following attributes are marked localizable by default:

entity	displayName
entity	displayCollectionName
attribute	displayName
action	displayName
actiongroup	displayName
optionset	displayName
optionsetitem	displayName
entityformheaderitem	label
entityformsection	displayName
entityviewcolumn	label
customAction	displayName
relationship	displayName
report	displayName

At metadata attribute level, there is a new property **IsLocalizable**. If thecheckbox is ticked on Text attributes, the system automatically creates additional fields in the database for each application language except for the default one (that is, en-GB).

NOTEThe primary attributes are not localizable.

To localize metadata, follow these steps:

1. Provide the value for the display name in en-GB.
2. Switch to the language in which you want to localize resources. If the Debug mode has been activated, the value to be localized is marked with the question mark.
3. Replace the en-GB value displayed in the Display Name field with the one corresponding to the language you want to localize.
4. At the top-right corner of the page, click the Save and close icon.

The localization updates are saved in the database for the field corresponding to the current language selected from the **User Settings**. Inserts will save the data in all additional fields for each localizable attribute.

Localizing HTML Templates

FintechOS allows you to localize the content of HTML template embedded within the entity data forms, data form sections and custom actions.

NOTE

- Localization applies to the text part of the HTML element and it does not support children mixed with text. When mixing text with other HTML elements split the text in spans.
- The metadata displayed on forms, are localized using the DisplayName, you do not have to insert resources for them. If you want to translate the same metadata with different text, depending on the context, on different forms with different rules, then add the localization resources separately from the forms.

Localize HTML elements on data forms

To localize HTML elements that will be displayed on data forms, on the HTML Editor, click the Tools button, then click Source code and add the attributes data-resource-key and data-culture to the element.

The data-culture attribute should be a valid .NET culture name. For details on the valid .NET culture names, see: MSDN Table of Language Culture Names, Codes, and ISO Values Method [C++].

When not specified, the data-culture attribute is set by default to English UK (en-GB).

You can add additional translations for the same label by using the custom ebs-resource tag.

NOTEThe HTML editor does not allow the ebs-resource tag inside tr and td elements, so you need to add it to the div level.

```
<div>
<!-- localization for myAttr specific for italian language,
augments the localization inside td tag -->
<ebs-resource data-resource-key="myAttr" data-culture="it-
IT">TestCinci-IT</ebs-resource>
<table>
<tr>
<!-- localization by resource key -->
<td data-resource-key="myAttr" data-culture="ro-
RO">Attributul meu</td>
<td>{MyOptionSetAttribute}</td>
</tr>
</table>
</div>
```

On save, the resource keys are automatically inserted within the LocalizationResource and available for edit in the Localization Resources editor. The resource keys will be saved with the module name following these naming conventions:

Entity data form template naming convention

entities / entiyName / forms / formName / html

Example: *entities/ProductPromotion/forms/default/html*

Form section template naming convention

entities / entiyName / forms / formName / sections / sectionName / html

Example: *entities/ProductPromotion/forms/default/sections/Section 2/html*

Custom action naming convention

customActions / customActionName / html

Example: *customActions/Product Promotion/html*

Localize from Metadata

To get localization from metadata, use piped arguments inside the bracket expressions:

- `{ xxx | entity }` will be replaced with the display name of the entity named xxx. The entity name is case sensitive.
- `{ xxx | entities }` will be replaced with the display collection name of the entity named xxx. The attribute name is case sensitive.
- `{ myAttribute | attribute }` will be replaced with the display name of the attribute named myAttribute. The entity is the data form owner. The attribute name is case insensitive.
- `{ xxx_yyy| relationship}` will be replaced with the display name of the relationship between entity xxx and entity yyy. The relationship name is case insensitive.

HTML Template Localization

```
<div style="padding: 20px; background-color: white; border:
solid 1px #E0E0E0;">
    <table style="width: 100%;">
        <tbody>
            <tr>
                <td colspan="2">
                    <!-- entity pipe: display name of entity
test1 (case sensitive) -->
                    <h4>{test1 | entity }</h4>
                    <!-- entities pipe: display collection
name of entity test1 (case sensitive) -->
                    <h4>{test1 | entities }</h4>
                </td>
            </tr>
            <tr>
                <!-- localization by resource key -->
                <td data-resource-key="myAttr" data-
culture="ro-RO">Attributul meu</td>
                <td>{MyOptionSetAttribute}</td>
            </tr>
```

```
            <tr>
                <!-- localization by resource key -->
                <td data-resource-key="name" data-
culture="ro-RO">label pentru name</td>
                <td>{Name}</td>
            </tr>
            <tr>
                <td>{Name | attribute }</td>
                <td>{Name}</td>
            </tr>
            <tr>
                <!-- attribute pipe: shows display name of
attribute Field1 (case insensitive)-->
                <td>{Field1 | attribute }</td>
                <td>{Field1}</td>
            </tr>
            <tr>
                <td colspan="2">
                    <hr />
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

Localize Relationship Labels

To render relationships inside the HTML template, use a syntax similar to:

`{#sys_entity_sys_attribute,collapse:Attributes#}`

Where:

- sys_entity_sys_attribute is the name of the relationship.
- Attributes is a user-defined label to be rendered as title.

To automatically localize this construct, use the localizable displayName attribute from the Relationship by specifying `$displayName` as the label.

`{#sys_entity_sys_attribute,collapse:$displayName#}`

Localizing Option Set Items

The `dxSelectBox.option` function is used in customization code to access the name of the option set directly from the control, so localization of optionSetItem would return the localized DisplayName instead of the expected Name.

The rendering for the select box has been modified so that it uses the **DisplayName** attribute of the option set item as display name instead of the **Name** attribute.

The below will return the option set item name and not the translated text:

```
$("#ebsContainerContent_xxx_list").dxSelectBox("instance").option
("text")
$("#ebsContainerContent_xxx_list").dxSelectBox("instance").option
("displayValue")
```

Localizing Views

To localize view columns, follow these steps:

1. On the right-side of the navigation bar, next to your name, click the **User Settings** icon. The User Settings menu expands.
2. Select English (en-GB). The page refreshes and the UI texts are displayed in English.
3. On the desired entity (Edit Entity page), scroll-down to the Data Views section and click on it. The Data Views section expands.
4. Double-click the desired view. view configuration page appears.
5. Click the Data tab, scroll-down to the Entity View Columns section and click the Insert button.
6. Fill-in the following fields: Attribute Name and Label.

7. Go to **User Settings** again and switch the language (e.g., fr-FR). The page refreshes and the Label field is marked with the question mark (?) indicating that it has not been localized yet.
8. In the Label field, enter the localized value then at the top-right corner of the page click one of the save icons (based on your needs).

NOTE

- Localization is backward compatible with previous implementations, in which view columns have been added in the Data field, instead of the View columns grid.
- Once localized, the resource will be displayed in the language selected, regardless the display option used within the view.

Client-side Localization

You can specify any "design-time" language specific messages. The resources are automatically exported when the metadata is saved and are available for edit in other defined languages in the Localization Resources editor.

It is not mandatory to define message values for all languages using JavaScript. You should define at least one language (en-GB or ro-RO) using JavaScript. For other languages, go to the Localization Resource entity and fill-in the **Value** field corresponding to the resources to localize.

At run-time the application will resolve the translations from the database with fallback to the explicit values defined in the script.

Code Snippet

```
var rsMyMessage = new EbsResource({
            key : "myMessage",
                "en-GB" : "message in English",
                "ro-RO" : "mesaj in romana",
                "de-DE" : "Meldung auf Deutsch"
```

```
            });

var rsMyMessageFmt = new EbsResource({
            key : "myMessageFmt",
                "en-GB" : "Field {0} is empty",
                "ro-RO" : "Campul {0} nu este completat",
                "de-DE" : "Der Feld {0} ist leer"
            });
console.log( rsMyMessage.getString()); // when culture is ro-RO
outputs: mesaj in romana
console.log( rsMyMessageFmt.getString("Field1"));  // when culture
is ro-RO outputs: Campul Field1 este necompletat
```

ebs.showMessage

The existing method for ebs.showMessage has been modified to accommodate localization.

You can verify the code snippet either in the 'Developer Tools' (in Chrome) or in the **After Events** field.

Code snippet from Developer Tools:

```
> ebs.showMessage
        < f (englishMessage, type, romanianMessage) {
            //type:success, warning, info, error
            //englishMessage can be a resource identifier

            var Localized = null;
            if (englishMessages && englis...
```

Code snippet for After generate events:

```
ebs.showMessage(englishMessage, type, romanianMessage);
```

NOTENew localization is backwards compatible, it passes both English and Romanian message strings; however, when using EbsResource it is not mandatory that you pass the Romanian message.

Code snippet from Developer Tools:

```
> EbsResource
```

```
< f EbsResource(data) {
            if (data) {
               for (var prop in data) {
                    this[prop] = data[prop];
                }
            }
    }
```

ebs.showMessage

```
var rsMyWarning = new EbsResource({
                key : "myWarning",
                    "en-GB" : "Warning!",
                    "ro-RO" : "Atentie!",
                });
// other code
ebs.showMessage(rsMyWarning.getString(), 'warning');   //new style
ebs.showMessage("english message", 'warning', "mesaj in
romana");  // still works for backwards compatibility
```

When saved, the resource keys are automatically inserted in the database and are available for edit in the **Localization Resource** entity. The resource keys will be saved with the module name following these naming conventions:

Entity data form aftergeneratejs naming convention

aftergenerateJs: entities / entiyName / forms / formName / aftergeneratejs

Example:

entities/Product/forms/default/aftergeneratejs

Form section aftergeneratejs naming convention

entities / entiyName / forms / formName / sections / sectionName / aftergen eratejs

Example:

entities/Product/forms/default/sections/Section 1/aftergeneratejs

Entity views aftergeneratejs naming convention

entities/ entityName/ views/ viewName / aftergeneratejs

Example: entities/Product/views/default/aftergeneratejs

Form attribute change aftergeneratejs naming convention

entities / entityName/ forms / formName./ attributes / attributeName/ attributeChangeEventsJs

Example:

entities/Product/forms/default/attributes/Name/attributeChangeEventJs

Custom actions aftergeneratejs naming convention

customActions / actionName / aftergeneratejs

Example:

customActions/Product Promotion/aftergeneratejs

Server-side localization

JavaScript localization support is available in workflows and workflow libraries. The pattern is similar to the one used in Form AftergenerateJS, with a small difference at call time:

```
var resource1 = new EbsResource({
    key : "myKey",
        "ro-RO" : "mesaj romana",
        "en-GB" : "English message"
});

throwException( getString(resource1), 1 );  //the resource must be
passed as parameter to getString

//support for message formatting

var resource2 = new EbsResource({
    key : "myKey",
```

```
    "ro-RO" : "Atributul {0} din entitea {1} nu poate fi null",
    "en-GB" : "Attribute {0} in entity {1} cannot be null"
});

// getString can be called as string.Format in .NET, passing the
formatting args as an array argument

throwException( getString(resource2,
["MyManadatoryAttribute","MyEntity"]) , 1 );
```

Upon save, the resource keys are automatically inserted within the LocalizationResource and available for edit in the Localization Resources editor. The resource keys will be saved with the module name following these naming conventions:

Workflow naming convention

workflows / workflowname / js

Example:

workflows/Product wf/js

Workflow library naming convention

workflowlibraries / workflowlibname / js

Example:

workflowLibraries/Product wf library/js

Code Snippets Support

Code snippets are small sequences of reusable code that can be inserted using a combination of shortcut keys. They are very useful as they make it easier to remember specific formatting of functions and avoid spending time on typos and syntax errors.

Code snippets are an aid of type Intellisense/auto-complete and most of the times they are incomplete and / or syntactically incorrect and require further processing thereof.

Code snippets support is available via the Monaco Intelisense for text boxes and controls of type JavaScript. and for the HTML editor.

NOTEBackward compatibility is not necessary, so the snippets, their name, their content and the way they are organized in drop-downs may change from one Innovation Studio release to another without prior notice.

Code Snippets Support for the HTML Editor

Code snippets are also available in the HTML editor available on forms and user journeys, via the Monaco Editor. To use these code snippets on the data form or digital journey, go to the **Code** section, **Template** tab. From the HTML editor toolbar, click **Tools** and select **Source code**. The Source code page will be displayed using the Monaco Editor.

Pressing the **CTRL+Space** keys opens a drop-down which contains not only entries corresponding to certain code snippets but also attributes and relations belonging to the entity.

Code Snippets Placeholders

The code snippets might contain hint words, known as placeholders which can be easily selected and changed. The placeholders are highlighted with a background color.

Navigating placeholders

Upon snippet insertion, the cursor will be placed on the first placeholder (if any). If no placeholder is available, the cursor will go to the end of the snippet.

To navigate from one placeholder to another, press the **TAB** key or **SHIFT**+**TAB**.

Pressing **TAB** when the cursor is over the last placeholder will move the cursor at the end of the snippet.

Replacing placeholders

There might be placeholders without underlying text, visible due to a narrow gray line on that position.

When the cursor is over a placeholder, starting to type will remove the placeholder text.

> ⚠ **IMPORTANT!**Replace all placeholders, otherwise, errors might occur as the code snippet syntax might be broken or the syntax remains unbroken but it is logically incorrect.

The presence of one or more cursors indicate that those placeholders will be filled in simultaneously with the same text. Additionally, such group of placeholders might have identical texts beneath.

Make sure to replace the placeholders and click **OK** to save the source code changes.

Deactivating placeholders

When the cursor falls outside the snippet, the placeholders will be deactivated and any text which has not been replaced will remain in the snippet.

When the placeholders are deactivated:

- They are no longer highlighted with a background color
- Navigating from one placeholder to another is no longer possible using **TAB** and **SHIFT**+**TAB**.
- Typing when cursor is on top of the placeholders will not remove the placeholder text.

Series of placeholders

Some snippets may have for some fields' series of placeholders in a row, like: true false

To eliminate the unwanted variants and keep the desired one, navigate placeholders using **TAB** or **SHIFT+TAB** and when on top of the placeholder that you want to eliminate, press the **delete** key.

Nested code snippets

Snippets might be nested, meaning that in a placeholder you can insert another snippet, without deactivating the placeholders of the first snippet.

Examples of code snippets

Pressing **CTRL+Space** and selecting **attribute rows by 2** from the drop-down will insert the following code snippet:

```
<div class="row">
   <div class="col-lg-6 col-md-6 col-sm-6 col-xs-12" >
        <div class="row">
            <div class="col-lg-4 col-md-4 col-sm-4 col-xs-12 data
form-label"></div>
           <div class="col-lg-8 col-md-8 col-sm-8 col-xs-12"></div>
        </div>
   </div>
   <div class="col-lg-6 col-md-6 col-sm-6 col-xs-12" >
        <div class="row">
            <div class="col-lg-4 col-md-4 col-sm-4 col-xs-12 data
form-label"></div>
            <div class="col-lg-8 col-md-8 col-sm-8 col-xs-
12"></div>
        </div>
   </div>
  </div>
```

Pressing **CTRL+Space** and selecting **< = “”></>** from the drop-down will insert the following code snippet:

```
<tag = “ “>text</tag>
```

NOTEIn cases like the previous example, two or more placeholders might be written simultaneously, more than one cursor being available.

Examples of code snippets for attributes and relations.

- Pressing **CTRL+Space** and selecting **{#relView#} ebs_AAA_Ent_8966_businessunit** will insert {#ebs_AAA_Ent_8966_businessunit#}

- Pressing **CTRL+Space** and selecting **{attrLabel|attribute} businessunitid** will insert `{businessunitid|attribute}`
- Pressing **CTRL+Space** and selecting **{attrName}** businessunitid will insert `{businessunitid}`
- Pressing **CTRL+Space** and selecting **{relName|entities} ebs_AAA_Ent_8966_ businessunit** will insert `{ebs_AAA_Ent_8966_businessunit|entities}`

Code Snippets Support for JavaScript Text Boxes

Using Monaco Intellisense available via the Monaco Editor, the platform provides two mechanisms for using code snippets in Java Script text boxes, as follows:

- **$s.** - displays code snippets
- **$m.** - displays code snippets for entities and attributes.

The sections below describe how to use the two mechanisms.

Code Snippets

Writing **$s.** opens a drop-down list which offers the possibility to choose a snippet name or another. There might be one or more entries in the list which are not snippets but group names in which case typing a dot after them will open new drop-down menu.

The snippets can be organized in menus or drop-downs on multiple levels in structures like hierarchies.

A snippet name together with its selection path may look like:

```
$s.ebs.functions.callAction
```

Placing the cursor right at the end of the snippet path/name (full name) and pressing the **TAB** key, the system will replace the string with:

```
ebs.callAction( actionId, { id:ebs.getCurrentEntityId() }, function
( e ) { function body; });
```

NOTE If you do not place the cursor at end of the snippet path / name (full name) before pressing the **TAB** key, the snippet name will be breaking syntax.

In the example provided above, “ebs.“ is inserted in front of the function to reduce further typing.

Code Snippets Placeholders

The code snippets might contain hint words, known as placeholders which can be easily selected and changed. The placeholders are highlighted with a background color.

In the example provided above, the **ebs.callAction** function snippet, “actionId” and “function body“, although not visible in the text above, might be placeholders.

Navigating placeholders

Upon snippet insertion, the cursor will be placed on the first placeholder (if any). If no placeholder is available, the cursor will go to the end of the snippet.

To navigate from one placeholder to another, press the **TAB** key or **SHIFT**+**TAB**.

Pressing **TAB** when the cursor is over the last placeholder will move the cursor at the end of the snippet.

Replacing placeholders

There might be placeholders without underlying text, visible due to a narrow gray line on that position.

When the cursor is over a placeholder, starting to type will remove the placeholder text.

IMPORTANT! Replace all placeholders, otherwise, errors might occur as the code snippet syntax might be broken or the syntax remains unbroken but it is logically incorrect.

The presence of one or more cursors indicate that those placeholders will be filled in simultaneously with the same text. Additionally, such group of placeholders might have identical texts beneath.

Make sure to replace the placeholders and click **OK** to save the source code changes.

Deactivating placeholders

When the cursor falls outside the snippet, the placeholders will be deactivated and any text which has not been replaced will remain in the snippet.

When the placeholders are deactivated:

- They are no longer highlighted with a background color
- Navigating from one placeholder to another is no longer possible using **TAB** and **SHIFT+TAB**.
- Typing when cursor is on top of the placeholders will not remove the placeholder text.

Series of placeholders

Some snippets may have for some fields' series of placeholders in a row, like: left inner right

To eliminate the unwanted variants and keep the desired one, navigate placeholders using **TAB** or **SHIFT+TAB** and when on top of the placeholder that you want to eliminate, press the **delete** key.

Nested code snippets

Snippets might be nested, meaning that in a placeholder you can insert another snippet, without deactivating the placeholders of the first snippet.

Code snippets for entities and attributes

The **$m** mechanism has been introduced to enable FintechOS engineers to easily select entities and attributes names, transforming them in strings if necessary.

Writing **$m.** opens a drop-down list containing existing entities. Selecting an entity and typing a dot opens a new drop-down containing the attributes of that entity. Pressing **TAB** after the inserted entity or attribute transforms the expression into a string which contains only the name of the entity or attribute depending on the selection(s) made from drop-down(s).

If selecting the current entity from the drop-down, data extensions (if any will also be displayed) in the drop-down list.

- Pressing **TAB** after `$m.Account.Accountid` transforms the expression into "Accountid"
- Pressing **TAB** after `$m.Account` transforms the expression into "Account"

To eliminate the quotes around the string, press **TAB** a second time .

NOTEIf there is a dot before “$m.” (**.$m.**), the expression is not transformed into a string, keeping the selected name.
Pressing **TAB** after `.$m.Account.Accountid` transforms the expression into **.Accountid**

Fintech Automation

An automation processor is a micro-service with embedded AI which captures data from external data sources and provides you with AI driven insights and actions based on machine learning, big data aggregation and cognitive reasoning.

Using the automation processors in digital journeys enables you to run 100% paperless, digital, AI-enabled digital journeys.

Innovation Studio offers you with a list of built-in automation processors that you can use to streamline your digital journeys.

Business Formulas

Formulas process different inputs from your digital journeys (such as income, age, assets, risk class, etc.) in order to generate desired outputs (such as credit scores, insurance premiums, interests, etc). The Business Formulas supports multi-step calculations, using a variety of data types and built-in functions. A data sets feature is included that allows you to reference predefined value mappings (such as risk matrices).

For details, see "Business Formulas" on page 379.

Business Decisions Processor

The Business Decisions Processor allows you to define evaluations (credit scores, product eligibility, insurance premiums, interest rates, etc.) that you can use at various decision points in a digital journey. It relies on decision matrices that incorporate a variety of evaluation criteria, allowing you to implement complex decision modeling for your business processes.

For details, see the Business Decisions Processor documentation.

Digital Product Automation

The Digital Product Automation allows you to create and maintain banking products that you can use in FintechOS digital journeys. The banking products are configured from a graphical user interface while Digital Product Automation automatically populates and maintains the underlying data in a consistent data model. Consequently, you have a reliable data model for your product portfolio that you can reference when you build your digital journeys. This allows you to manage your product portfolio at will, without having to re-code your digital journeys every time a banking product is added, updated, or retired.

For details, see the Digital Product Automation documentation.

Computer Vision

The Computer Vision automation processor allows you to automatically populate entity records in your FintechOS applications with text extracted from document scans or photos.

For details, see the Computer Vision documentation.

Digital Insurance Product Automation

The Digital Insurance Product Automation allows you to create and maintain insurance products that you can use in FintechOS digital journeys. The insurance products are configured from a graphical user interface while Digital Insurance Product Automation automatically populates and maintains the underlying data in a consistent data model. Consequently, you have a reliable data model for your product portfolio that you can reference when you build your digital journeys. This allows you to manage your product portfolio at will, without having to re-code your digital journeys every time an insurance product is added, updated, or retired.

For details, see the Digital Insurance Product Automation documentation.

Digital Documents Processor

The Digital Documents Processor enables you to leverage intelligent document automation to reduce errors, boost productivity and maximize business outcomes. Automatically generate dynamic, personalized & accurate essential business documents – including customized contracts and agreements, by merging real-time data.

For details, see the Digital Documents Processor documentation.

eSign

The eSign automation processor allows customers to electronically sign digital documents streamlining the customer experience. The qualified electronic signatures turn documents into sealed, tamper-evident PDFs throughout the signing process life-cycle.

For details, see the eSign documentation.

Face Recognition

The Face Recognition enables liveness detection to validate an individual's identity. It uses Machine Learning to compare ID/Driving License or any other picture that was uploaded during the OCR process, with the selfie to certify that they belong to the same person.

For details, see the Face Recognition documentation.

Video Streaming

The Video Streaming automation processor provides a seamless experience for users and video call center operators, allowing them to have a secure conversation. It provides both the customer and the consultants a guided, predefined process to complete the customer identity verification and provide support.

For details, see the Video Streaming documentation.

Hyper-Personalization Automation

The Hyper-Personalization Automation processor allows you to create personalized content and business-tailored segments of audience (customer personas) to further create effective campaigns and meaningful interactions with customers.

For details, see the Hyper-Personalization Automation documentation.

Business Workflows Processor

The Business Workflows Processor allows you to implement state machine computation modeling in your FintechOS digital journeys by defining states and state transitions for your business entities.

For details, see the Business Workflows Processor documentation.

Omnichannel Campaigns

The Omnichannel Campaigns automation processor empowers you with the ability to create effective and user-tailored ways of interacting with the customer. Automate completely personalized campaigns, populate unique emails for each individual, ensuring a personalized communication with your customers based on their needs.

For details, see the Omnichannel Campaigns documentation.

Omnichannel Communication Automation

The Omnichannel Communication Automation processor provides email delivery and real-time email events tracking directly from within FintechOS.

For details, see the Omnichannel Communication Automation documentation.

Processor settings and mapping

For the following processors Computer Vision, eSign, and Face Recognition with and without Liveness there is a no-code form in the FintechOS Portal where the settings and mappings are defined.

The settings allow for the following types of controls:

String: will create a normal input control.

Number: will create a numeric input control with spinner.

Boolean: will create a dropdown control with following displayed values: Default, Yes, No.

Lookup: will create a control which opens a grid for selecting the desired value. The grid allows filtering.

Object: will render nested controls inside it.

Array: will render nested controls inside it. Add and remove buttons are available to manipulate sets of controls

To render the proper form, follow these steps in the Innovation Studio and in the FintechOS Portal:

1. In the Innovation Studio, open the entity FTOS_DFP_ProcessorSettings.
2. Open the data form, and in the Advanced tab fill in the following code:

   In the Before events tab:

```
// "settings" in this case represents the name of the entity
attribute
formData.formScope.jsonEditor = new ebs.JsonEditorControl
("settings");
```

   In After Events tab:

```
var settings = [{}] //will be discussed later

//add listener event on processor type
ebs.addFormChangeEvent("ebsContainerContent",
"digitalProcessorTypeId", setSettingsAndMappingsControls);

async function setSettingsAndMappingsControls(settingsJSON){
    //get the value of processorType which is a GUID
    var digitalProcessorType = ebs.getFormAttributeValue
("ebsContainerContent", "digitalProcessorTypeId");
    //get the optionSetName using the processorTypeId
    var optionSetItemName = await
ebs.getOptionSetItemNameById("FTOS_DFP_
DigitalProcessorTypeOptionSet", digitalProcessorType);
    if(digitalProcessorType) {
        //filter the settings to take the configuration for
the desired processor type
        processorSettingsConfiguration = settings.filter
(function(item){
            return item.key.toLocaleLowerCase() ==
optionSetItemName.toLocaleLowerCase();
        })[0];
```

```
        //call the configController method of the editor
created in 'Before events'
        formData.formScope.jsonEditor.configController
(processorSettingsConfiguration.value, settingsJSON);
    }
}

//call the above method with the previously saved JSON in DB
setSettingsAndMappingsControls(ebs.getFormAttributeValue
("ebsContainerContent", "settings"));
```

The settings available are the following (This is an example, please modify as needed):

```
var settings = [
{
 key:'FaceRecognition',
 value: {
        documentationUrl:
'https://fintechos.com/documentation/APs/FaceRecognition/2.0
/UserGuide/Content/Settings.htm',
        props: [
                {
                    name: "isLiveness",
                    label: "Is Liveness",
                    type: "boolean"
                },
                {
                    name: "DestinationEntityName",
                    label: "Destination Entity Name",
                    type: "lookup",
                    lookupEntityName: "entity",
                    lookupViewName: "default",
                    displayProp: "displayName"
                },
                {
                    name: "SourceEntityName",
                    label: "Source Entity Name",
                    type: "lookup",
                    lookupEntityName: "entity",
                    lookupViewName: "default",
                    displayProp: "displayName"
                },
```

```
                {
                    name: "SourceLookupDestinationName",
                    label: "Source Lookup Destination Name",
                    type: "lookup",
                    lookupEntityName: "attribute",
                    lookupViewName: "default",
                    parentPropertyName: "SourceEntityName",
                    attributeToFilterReference: "entityid",
                    displayProp: "displayName"
                },
                {
                    name: "FileAttributeName",
                    label: "File Attribute Name",
                    type: "string"
                },
                {
                    name: "MaxRetry",
                    label: "Max Retry",
                    type: "number"
                },
                {
                    name: "MinimumAcceptedConfidence",
                    label: "Minimum Accepted Confidence",
                    type: "number"
                },
                {
                    name: "maskNextStepURLSuccess",
                    label: "Mask Next Step URL Success",
                    type: "object",
                    items: [
                            {
                                name: "entity",
                                label: "Entity",
                                type: "lookup",
                                lookupEntityName: "entity",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            },
                            {
                                name: "form",
                                label: "Form",
                                type: "lookup",
                                lookupEntityName:
"entityform",
                                parentPropertyName: "entity",
```

```
                                attributeToFilterReference:
"entityid",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            },
                            {
                                name: "section",
                                label: "Section",
                                type: "lookup",
                                lookupEntityName:
"entityformsection",
                                parentPropertyName: "form",
                                attributeToFilterReference:
"entityFormId",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            }
                        ]
                    },
                    {
                        name: "maskNextStepURLFail",
                        label: "Mask Next Step URL Fail",
                        type: "object",
                        items: [
                                {
                                    name: "entity",
                                    label: "Entity",
                                    type: "lookup",
                                    lookupEntityName: "entity",
                                    lookupViewName: "default",
                                    displayProp: "displayName"
                                },
                                {
                                    name: "form",
                                    label: "Form",
                                    type: "lookup",
                                    lookupEntityName:
"entityform",
                                    parentPropertyName: "entity",
                                    attributeToFilterReference:
"entityid",
                                    lookupViewName: "default",
                                    displayProp: "displayName"
                                },
                                {
```

```
                                name: "section",
                                label: "Section",
                                type: "lookup",
                                lookupEntityName:
"entityformsection",
                                parentPropertyName: "form",
                                attributeToFilterReference:
"entityFormId",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            }
                    ]
                },
                {
                    name: "businessStatusSuccess",
                    label: "Business Status Success",
                    type: "string"
                },
                {
                    name: "businessStatusFail",
                    label: "Business Status Fail",
                    type: "string"
                }
        ]
    }
},
{
    key: 'VideoStreaming',
    value: {
        documentationUrl:
'https://fintechos.com/documentation/APs/VideoStreaming/2.0/
UserGuide/Content/Settings.htm',
        props: [
                {
                    name: "DestinationEntityName",
                    label: "Destination Entity Name",
                    type: "lookup",
                    lookupEntityName: "entity",
                    lookupViewName: "default",
                    displayProp: "displayName"
                },
                {
                    name: "SourceEntityName",
                    label: "Source Entity Name",
                    type: "lookup",
```

```
                    lookupEntityName: "entity",
                    lookupViewName: "default",
                    displayProp: "displayName"
                },
                {
                    name: "SourceLookupDestinationName",
                    label: "Source Lookup Destination Name",
                    type: "lookup",
                    lookupEntityName: "attribute",
                    lookupViewName: "default",
                    parentPropertyName: "SourceEntityName",
                    attributeToFilterReference: "entityid",
                    displayProp: "displayName"
                },
                {
                    name: "QueueParameters",
                    label: "Queue Parameters",
                    type: "array",
                    items: [
                            {
                                name: "ParamName",
                                label: "Parameter Name",
                                type: "string",
                            },
                            {
                                name: "ParamValue",
                                label: "Parameter Value",
                                type: "string",
                            }
                    ]
                },
                {
                    name: "maskNextStepURLSuccess",
                    label: "Mask Next Step URL Success",
                    type: "object",
                    items: [
                            {
                                name: "entity",
                                label: "Entity",
                                type: "lookup",
                                lookupEntityName: "entity",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            },
                            {
```

```
                                name: "form",
                                label: "Form",
                                type: "lookup",
                                lookupEntityName:
"entityform",
                                parentPropertyName: "entity",
                                attributeToFilterReference:
"entityid",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            },
                            {
                                name: "section",
                                label: "Section",
                                type: "lookup",
                                lookupEntityName:
"entityformsection",
                                parentPropertyName: "form",
                                attributeToFilterReference:
"entityFormId",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            }
                        ]
                    },
                    {
                        name: "maskNextStepURLFail",
                        label: "Mask Next Step URL Fail",
                        type: "object",
                        items: [
                                {
                                    name: "entity",
                                    label: "Entity",
                                    type: "lookup",
                                    lookupEntityName: "entity",
                                    lookupViewName: "default",
                                    displayProp: "displayName"
                                },
                                {
                                    name: "form",
                                    label: "Form",
                                    type: "lookup",
                                    lookupEntityName:
"entityform",
                                    parentPropertyName: "entity",
```

```
                                attributeToFilterReference:
"entityid",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            },
                            {
                                name: "section",
                                label: "Section",
                                type: "lookup",
                                lookupEntityName:
"entityformsection",
                                parentPropertyName: "form",
                                attributeToFilterReference:
"entityFormId",
                                lookupViewName: "default",
                                displayProp: "displayName"
                            }
                        ]
                    },
            ]
        }

    },
    {
        key:'OCR',
        value:{
            documentationUrl:
'https://fintechos.com/documentation/APs/ComputerVision/1.5/
UserGuide/Content/Settings.htm',
            props: [
                        {
                            name: "SourceEntityName",
                            label: "Source Entity Name",
                            type: "lookup",
                            lookupEntityName: "entity",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        },
                        {
                            name: "Entities",
                            label: "Entities",
                            type: "array",
                            "items":[
                             {
                                name: "DestinationEntityName",
```

```
                            label: "Destination Entity Name",
                            type: "lookup",
                            lookupEntityName: "entity",
                            lookupViewName: "default",
                            displayProp: "displayName"
                          },
                          {
                            name: "SourceEntityName",
                            label: "Source Entity Name",
                            type: "lookup",
                            lookupEntityName: "entity",
                            lookupViewName: "default",
                            displayProp: "displayName"
                          },
                          {
                            name:
"SourceLookupDestinationName",
                            label: "Source Lookup Destination
Name",
                            type: "lookup",
                            lookupEntityName: "attribute",
                            parentPropertyName:
"SourceEntityName",
                            attributeToFilterReference:
"entityid",
                            lookupViewName: "default",
                            displayProp: "displayName"
                          }]

                    },
                    {
                        name: "FileAttributeName",
                        label: "File Attribute Name",
                        type: "string"
                    },
                    {
                        name: "MaxRetry",
                        label: "Max Retry",
                        type: "number"
                    },
                    {
                        name: "OptionSets",
                        label: "Option Sets",
                        type: "array",
                        "items":[
```

```
                                {
                                    name: "OptionSetName",
                                    label: "Option Set Name",
                                    type: "lookup",
                                    lookupEntityName:
"optionset",
                                    lookupViewName: "default",
                                    displayProp: "displayName"
                                },
                                {
                                    name: "MappingName",
                                    label: "Mapping Name",
                                    type: "string"
                                },
                                {
                                    name: "OptionSetItems",
                                    label: "Option Set Items",
                                    type: "object",
                                    items:[
                                            {
                                                name: "M",
                                                label: "M",
                                                type:
"lookup",
                                                lookupEntityN
ame: "optionsetitem",
                                                lookupViewNam
e: "default",
                                                parentPropert
yName: "OptionSetName",
                                                attributeToFi
lterReference: "optionSetId",
                                                displayProp:
"displayName"
                                            },
                                            {
                                                name: "F",
                                                label: "F",
                                                type:
"lookup",
                                                lookupEntityN
ame: "optionsetitem",
                                                lookupViewNam
e: "default",
                                                parentPropert
yName: "OptionSetName",
```

```
                                                attributeToFi
lterReference: "optionSetId",
                                                displayProp:
"displayName"
                                            }
                                    ]
                                }
                        ]

                    },
                    {
                        name: "LookupEntities",
                        label: "Lookup Entities",
                        type: "array",
                        "items":[
                                {
                                   name: "MappingName",
                                   label: "Mapping Name",
                                   type: "string"
                                },
                                {
                                   name: "EntityName",
                                   label: "Entity Name",
                                   type: "lookup",
                                   lookupEntityName:
"entity",
                                   lookupViewName: "default",
                                   displayProp: "displayName"
                                },
                                {
                                   name: "AttributeKey",
                                   label: "Attribute Key",
                                   type: "lookup",
                                   lookupEntityName:
"attribute",
                                   parentPropertyName:
"EntityName",
                                   attributeToFilterReferenc
e: "entityid",
                                   lookupViewName: "default",
                                   displayProp: "displayName"
                                },
                                {
                                   name: "Parent",
                                   label: "Parent",
```

```
                                    type: "object",
                                    items: [
                                            {
                                               name:
"AttributeParentKey",
                                               label:
"Attribute Parent Key",
                                               type:
"lookup",
                                               lookupEntityN
ame: "attribute",
                                               lookupViewNam
e: "default",
                                               parentPropert
yName: "EntityName",
                                               attributeToFi
lterReference: "entityid",
                                               displayProp:
"displayName"
                                            },
                                            {
                                                name:
"MappingParentName",
                                                label:
"Mapping Parent Name",
                                                type:
"string"
                                            }
                                     ]
                                }
                        ]

                    },
                    {
                       name: "Validations",
                       label: "Validations",
                       type: "array",
                       "items":[
                                {
                                   name: "type",
                                   label: "Type",
                                   type: "string"
                                },
                                {
                                   name: "Validations",
                                   label: "Validations",
```

```
                                    type: "string"
                                },
                                {
                                    name: "CheckScriptName",
                                    label: "Check Script
Name",
                                    type: "string"
                                }
                            ]
                        },
                        {
                            name: "AvailableDocumentTypes",
                            label: "Available Document Types",
                            type: "array",
                            "items":[
                                        {
                                            name: "type",
                                            label: "Type",
                                            type: "string"
                                        },
                                        {
                                            name: "DocumentType",
                                            label: "Document Type",
                                            type: "string"
                                        },
                                        {
                                            name: "Country",
                                            label: "Country",
                                            type: "string"
                                        },
                                        {
                                            name: "Provider",
                                            label: "Provider",
                                            type: "string"
                                        }
                            ]
                        },
                        {
                            name: "maskNextStepURLSuccess",
                            label: "Mask Next Step URL Success",
                            type: "object",
                            items: [
                                        {
                                        name: "entity",
                                        label: "Entity",
```

```
                                type: "lookup",
                                lookupEntityName: "entity",
                                lookupViewName: "default",
                                displayProp: "displayName"
                                },
                                {
                                    name: "form",
                                    label: "Form",
                                    type: "lookup",
                                    lookupEntityName:
"entityform",
                                    parentPropertyName:
"entity",
                                    attributeToFilterReferen
ce: "entityid",
                                    lookupViewName:
"default",
                                    displayProp:
"displayName"
                                },
                                {
                                    name: "section",
                                    label: "Section",
                                    type: "lookup",
                                    lookupEntityName:
"entityformsection",
                                    parentPropertyName:
"form",
                                    attributeToFilterReferen
ce: "entityFormId",
                                    lookupViewName:
"default",
                                    displayProp:
"displayName"
                                }
                        ]
                    },
                    {
                        name: "maskNextStepURLFail",
                        label: "Mask Next Step URL Fail",
                        type: "object",
                        items: [
                                {
                                name: "entity",
                                label: "Entity",
                                type: "lookup",
```

```
                                lookupEntityName: "entity",
                                lookupViewName: "default",
                                displayProp: "displayName"
                                },
                                {
                                    name: "form",
                                    label: "Form",
                                    type: "lookup",
                                    lookupEntityName:
"entityform",
                                    parentPropertyName:
"entity",
                                    attributeToFilterReferen
ce: "entityid",
                                    lookupViewName:
"default",
                                    displayProp:
"displayName"
                                },
                                {
                                    name: "section",
                                    label: "Section",
                                    type: "lookup",
                                    lookupEntityName:
"entityformsection",
                                    parentPropertyName:
"form",
                                    attributeToFilterReferen
ce: "entityFormId",
                                    lookupViewName:
"default",
                                    displayProp:
"displayName"
                                }
                            ]
                        },
                        {
                            name: "businessStatusSuccess",
                            label: "Business Status Success",
                            type: "string"
                        },
                        {
                            name: "businessStatusFail",
                            label: "Business Status Fail",
                            type: "string"
                        },
```

```
                    {
                        name: "DocumentType",
                        label: "Document Type",
                        type: "string"
                    }


                ]
    }
},
{
    key: 'ESign',
    value: {
        documentationUrl:
'https://fintechos.com/documentation/APs/eSign/2.1/UserGuide
/Content/Settings.htm',
        props: [
            {
                name: "EntityName",
                label: "Entity Name",
                type: "lookup",
                lookupEntityName: "entity",
                lookupViewName: "default",
                displayProp: "displayName"
            },
            {
                name: "FileAttributeNameList",
                label: "File Attribute Name List",
                type: "array",
                items: [
                        {
                            name: "fileAttributeName",
                            label: "File Attribute Name",
                            type: "string"
                        },
                        {
                            name: "fileToBeSignedName",
                            label: "File To Be Signed Name",
                            type: "string"
                        }

                ]
            },
            {
                name: "MaxRetry",
```

```
                label: "Max Retry",
                type: "number"
            },
            {
                name: "signedDocumentName",
                label: "Signed Document Name",
                type: "string"
            },
            {
                name: "WebhookUrl",
                label: "Webhook Url",
                type: "string"
            },
            {
                name: "WebhookStatusUrl",
                label: "Webhook Status Url",
                type: "string"
            },
            {
                name: "SignatureSteps",
                label: "Signature Steps",
                type: "array",
                items: [
                        {
                            name: "order",
                            label: "Order",
                            type: "number"
                        },
                        {
                            name: "signatureTag",
                            label: "Signature Tag",
                            type: "string"
                        },
                        {
                            name: "signatureTypeTemplate",
                            label: "Signature Type Template",
                            type: "string"
                        },
                        {
                            name: "signatureType",
                            label: "Signature Type",
                            type: "string"
                        },
                        {
                            name: "SignatureData",
```

```
                                label: "Signature Data",
                                type: "array",
                                items: [
                                        {
                                            name:
"SourceEntityName",
                                            label: "Source Entity
Name",
                                            type: "lookup",
                                            lookupEntityName:
"entity",
                                            lookupViewName:
"default",
                                            displayProp:
"displayName"
                                        },
                                        {
                                            name:
"SourceLookupDestinationName",
                                            label: "Source Lookup
Destination Name",
                                            type: "lookup",
                                            lookupEntityName:
"attribute",
                                            lookupViewName:
"default",
                                            parentPropertyName:
"SourceEntityName",
                                            attributeToFilterRef
erence: "entityid",
                                            displayProp:
"displayName"
                                        },
                                        {
                                            name:
"DefaultFields",
                                            label: "Default
Fields",
                                            type: "object",
                                            items:[
                                                    {
                                                        name:
"languageCode",
                                                        label:
"Language Code",
```

```
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"documentType",
                                                    label:
"Document Type",
                                                    type:
"string"
                                                }
                                            ]
                                        },
                                        {
                                            name: "MappedFields",
                                            label: "Mapped
Fields",
                                            type: "object",
                                            items: [
                                                {
                                                    name:
"languageCode",
                                                    label:
"Language Code",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"documentType",
                                                    label:
"Document Type",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"email",
                                                    label:
"Email",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"phoneMobile",
```

```
                                                    label:
"Phone Mobile",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"firstName",
                                                    label:
"First Name",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"lastName",
                                                    label:
"Last Name",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"documentIssuedBy",
                                                    label:
"Document Issued By",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"socialSecurityNumber",
                                                    label:
"Social Security Number",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"documentExpiryDate",
                                                    label:
"Document Expiry Date",
                                                    type:
"string"
                                                },
                                                {
```

```
                                                    name:
"documentIssuedOn",
                                                    label:
"Document Issued On",
                                                    type:
"string"
                                                },
                                                {
                                                    name:
"documentNumber",
                                                    label:
"Document Number",
                                                    type:
"string"
                                                },
                                            ]
                                        },
                                        {
                                            name: "OptionSets",
                                            lable: "Option Sets",
                                            type: "string"
                                        },
                                        {
                                            name:
"LookupEntities",
                                            lable: "Lookup
Entities",
                                            type: "string"
                                        }
                                    ]
                                },
                                {
                                    name: "smsText",
                                    label: "SMS Text",
                                    type: "string"
                                },
                                {
                                    name: "clickMsg",
                                    label: "Click link text",
                                    type: "string"
                                },
                                {
                                    name: "disableEmail",
                                    label: "Disable Email",
                                    type: "boolean"
                                }
```

```
                ]
            },
            {
                name: "maskNextStepURLSuccess",
                label: "Mask Next Step URL Success",
                type: "object",
                items: [
                        {
                            name: "entity",
                            label: "Entity",
                            type: "lookup",
                            lookupEntityName: "entity",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        },
                        {
                            name: "form",
                            label: "Form",
                            type: "lookup",
                            lookupEntityName: "entityform",
                            parentPropertyName: "entity",
                            attributeToFilterReference:
"entityid",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        },
                        {
                            name: "section",
                            label: "Section",
                            type: "lookup",
                            lookupEntityName:
"entityformsection",
                            parentPropertyName: "form",
                            attributeToFilterReference:
"entityFormId",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        }
                ]
            },
            {
                name: "maskNextStepURLFail",
                label: "Mask Next Step URL Fail",
                type: "object",
                items: [
```

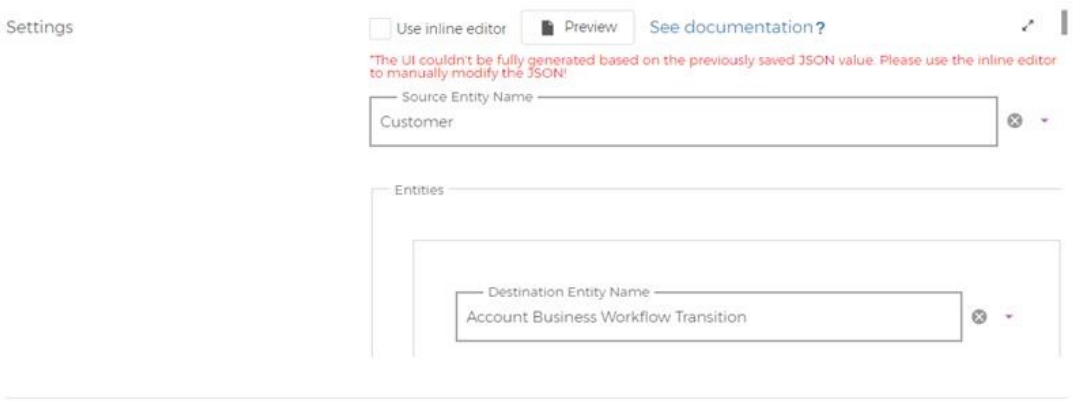
```
                        {
                            name: "entity",
                            label: "Entity",
                            type: "lookup",
                            lookupEntityName: "entity",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        },
                        {
                            name: "form",
                            label: "Form",
                            type: "lookup",
                            lookupEntityName: "entityform",
                            parentPropertyName: "entity",
                            attributeToFilterReference:
"entityid",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        },
                        {
                            name: "section",
                            label: "Section",
                            type: "lookup",
                            lookupEntityName:
"entityformsection",
                            parentPropertyName: "form",
                            attributeToFilterReference:
"entityFormId",
                            lookupViewName: "default",
                            displayProp: "displayName"
                        }
                    ]
                },
                {
                    name: "businessStatusSuccess",
                    label: "Business Status Success",
                    type: "string"
                },
                {
                    name: "businessStatusFail",
                    label: "Business Status Fail",
                    type: "string"
                },
                {
                    name: "redirecttoNamirialLink",
```

```
                label: "Redirect To Namirial Link",
                type: "string"
            }
        ]
    }
}
]
```

IMPORTANT!The fields available have to be configured with care. Each field must be tailed to the specific use-case.

To configure the processor itself, follow these steps:

1. Open the FintechOS Portal, Digital Flow Settings → Flow Settings → Processor Settings section. To insert a new processor, click on the Insert icon on the right- top corner of the screen. to edit an existing processor, select the desired processor from the list.
2. In the Settings tab, a no-code form will be displayed. What has been configured in the code snippet above will be rendered here in a no-code approach.



3. Fill in the name of the processor. Select for each option set provided the needed information e.g. source entity name, type, item etc.

   Based on the values added/selected by him in each field, it is possible to generate the JSON file. Click the ''Preview'' button to see the JSON file.

To update the JSON as done for the previous releases, check the ''Use inline editor'' bool and the Innovation Studio will generate the JSON:

- on Edit mode, the inline editor will be populated with the previously saved value
- on Add mode, the inline editor value is empty.

To consult the documentation on the settings for an automation processor, click the ''See documentation'' hyperlink which will redirect the user to the documentation website.

The two arrows on the right side of the screen open the editor in full screen to see the whole list of the possible no-code selection.

When trying to save the configurations on Edit mode, however the JSON file cannot be rendered in the UI of the data form due to it having been created using inline editor or the configuration is not correct, a message is displayed below the top buttons above the source entity name in red.

On Save mode:

- if the Use inline editor is checked, then the inline editor value is saved
- If the Use inline editor is not checked, then the JSON generated using the UI is saved.

Business Formulas

Formulas process different inputs from your digital journeys (such as income, age, assets, risk class, etc.) in order to generate desired outputs (such as credit scores, insurance premiums, interests, etc). It is mainly used to define mathematical and logical calculations that support various business needs.

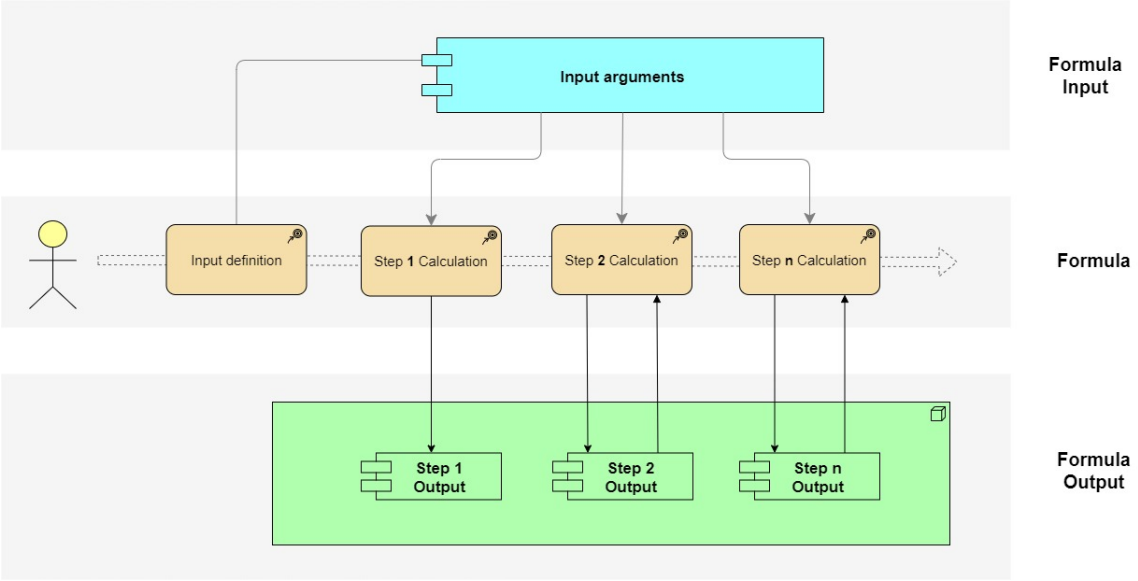
The Business Formulas allows users to create advanced computations based on data imported from Excel.

You can implement multi-step calculations, using a variety of data types and built-in functions. An embedded data sets feature allows you to reference predefined value mappings between sets of discriminants (such as age, sex, driving experience, etc) and specific metrics (such as a risk coefficient).

The advantages are:

- Ability to import large Excel data sets such as risk matrices
- Intellisense assistance for writing your formulas in the Formula editor
- Large array of built-in functions

Formulas use a simple syntax and can incorporate several steps where the result of a step is used in subsequent steps. They can take into account data sets imported in Excel format from third-party systems.



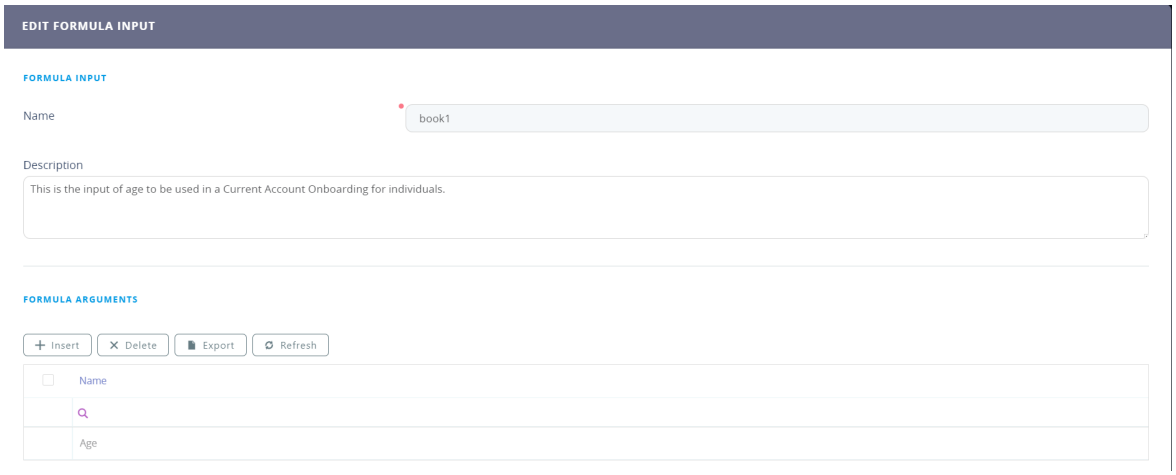
In the following chapters, you will read about the way to define the input for the formula using arguments, how to built the actual formula in steps and test it, how to use the editor and, finally, how to import data sets from an Excel fine and how you can use the built-in functions for each type of data.

Define Formula Inputs 381
Define Formula Expressions 385
Formula Editor 396
Data Sets 425

Formula Parameter Mapping 431
Calling the Business Formulas 433
Export a Formula or Data Set 436

Define Formula Inputs

Formula Input is a grouping of parameters that can be used on more than one formula. Each input has a set of arguments used in several formulae at a time.



To define the set of arguments that go into a formula:

1. Open the Main Menu ( ) in Innovation Studio.
2. Select **Business Formulas**.
3. Select **Formula Input**.
4. Click the **Insert** button ( ) at the top right corner of the page.
5. Enter a **Name** for the formula input. This name must be unique. This name will be referenced by formulas that process this specific set of arguments.
6. Optionally enter a **Description** for the formula input.
7. Click the **Save and Reload** ( ) button at the top right corner of the page.

Alternatively, from stage 4, it is possible to delete an input by clicking the ''**X**'' sign on the right corner of the screen. Updating or deleting a Formula Input that is used in a Formula which is not in status Draft is not allowed. This will also apply to all Formula Arguments declared in the Formula Input.

Add arguments to a formula input

Formula argument is the entity which holds a single value (be it simple or collection) which will be used as input for the Business Formulas. They are practically the parameters for the formula.

For example, for an insurance product it is possible to have as arguments the construction year of the house, the construction type, the seismic zone, the risks and the insured amount. For a banking product such as a loan, the arguments can be education, age, income, expenses, FICO score etc.

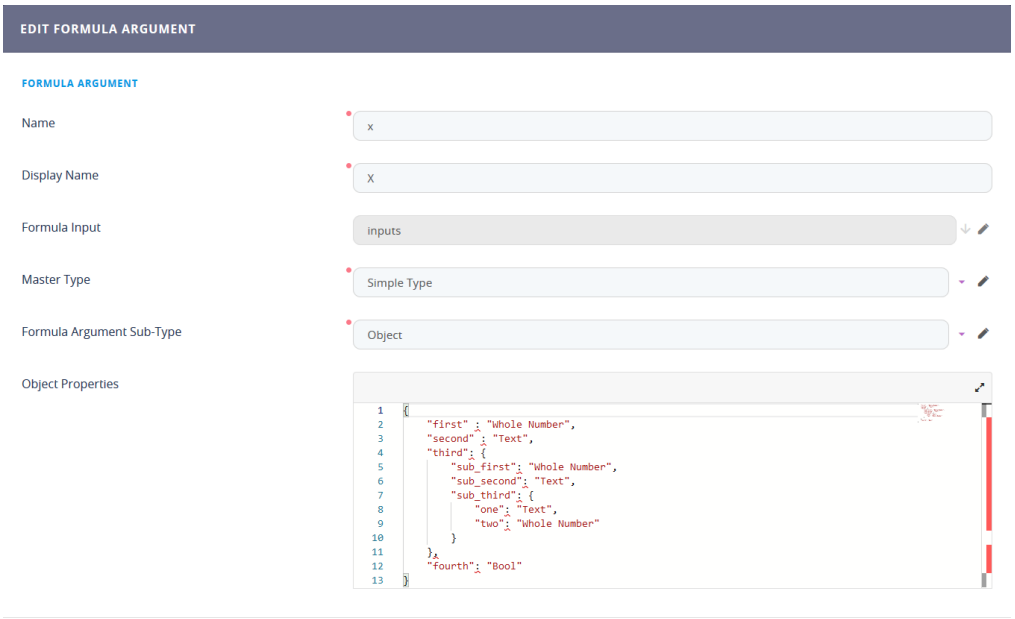
1. In the **Formula input** page, in the **Formula Arguments** section, click the **Insert** button to add an argument.
2. Enter a **Name** for the argument. This is a unique name that identifies the argument in the formula input e.g. Formula Arguments must have a unique name per each Formula Input (you can reuse argument names for different formula inputs).
3. Enter a **Display Name** for the argument. This is how the argument will be displayed in the end-user interface.
4. The **Formula Input** field is automatically populated with your formula input name.
5. Select the **Master Type** of your argument as either Simple Type (a single value) or Collection (a set of multiple values; decimal and whole number collections only). For

example, a **Simple Type - Decimal** can be the amount that the client must pay in a month for a credit he has taken, while a **Collection – Decimal** represents a list containing all the monthly payments installments that the client must pay in order to fully repay the credit he has taken.

6. In the **Formula Argument Sub-Type** field, enter the data type for your argument:

   - Whole Number
   - Decimal
   - Boolean - only for Simple Type master types
   - Text - only for Simple Type master types
   - Object - only for Simple Type master types. When this data type is selected, an **Object Properties** field will open where you must enter the JSON code containing the object's keys and data types. This will be presented in intellisense to the user in the Formula Step where he can use the argument. The object should look similar to:

     ```
     {
     "first" : "WholeNumber",
     "second" : "Text",
     "third": "Decimal",
     "fourth": "Bool"
     }
     ```



For example, you can have a collection master type with a whole number sub-type for the argument moto risks e.g. for each risk, fire, vandalism, hurricane. Each risk there is a number e.g. 1, 2, 3, 4.

7. Click the **Save and Close** button (💾) at the top right corner of the page.

8. Repeat for any additional arguments you wish to include in your formula. The order for adding the arguments is not relevant.

   After you've added all the arguments, click the **Save and Close** button (💾) at the top right corner of the page.

> **HINT**
> To change or delete a Formula Argument, you must make sure that the parent Formula Input is not used in an active formula.

After having set the parameters for the formula, it is time to set the steps for the calculation of each formula.

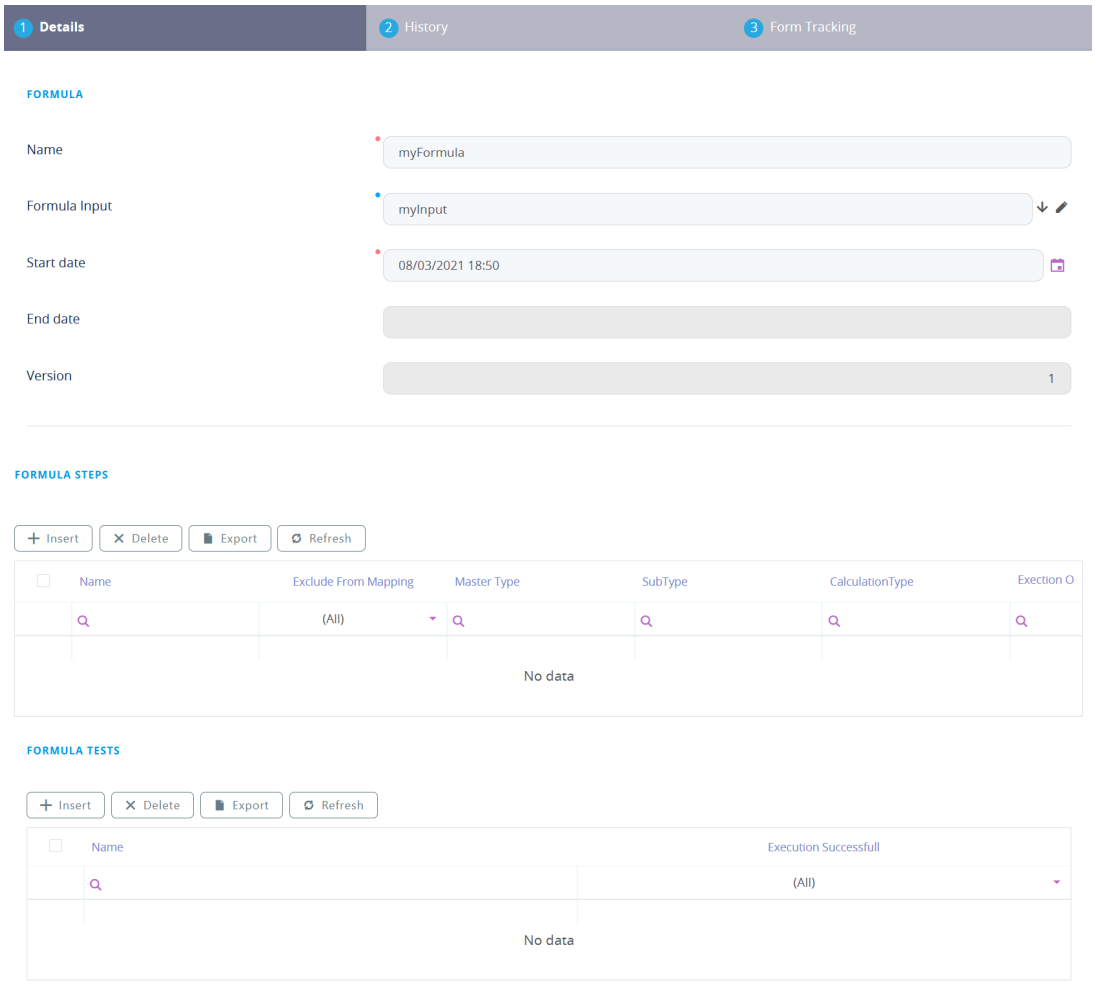
Define Formula Expressions

This is where you build business formulas based on your business needs. Using pre-defined arguments, you can structure a formula into steps and test it. The formula expressions support different types of results (simple or collection).

To define a formula expression:

1. Open the Main Menu ( ) in Innovation Studio.
2. Select **Business Formulas**.
3. Select **Formula**.
4. Click the **Insert** button ( ) at the top right corner of the page.
5. Enter a **Name** for the formula.
6. Select the **Start Date** at which the formula becomes active. Automatically, it sets the current date, but it is possible to change it. It selects the time as well.
7. In the **Formula Input** field, select the set of arguments that will be processed by the formula (see "Define Formula Inputs" on page 381 for details).
8. The **End Date** and **Version** fields are populated automatically based on the formula's versioning. For details, see "Formula Versioning" on page 391.

9. Click the **Save and Reload** ( ) button at the top right corner of the page.

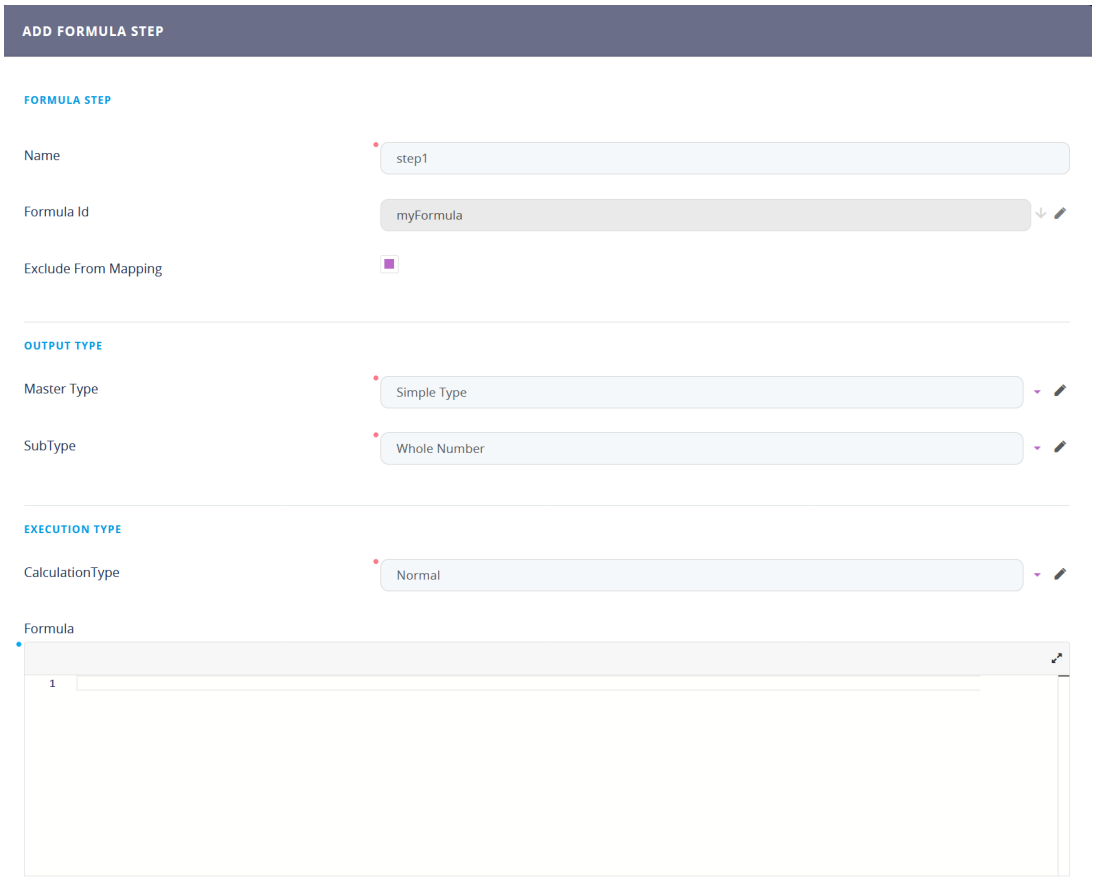


Add steps to a formula

Steps allow you to process a formula in successive stages and to use the output from a step as an input argument in subsequent steps.

For example, you can create a formula that calculates the premium amount in the first step, then use the result as an input in the second step to calculate the monthly payment installments (by dividing the premium amount to the number of months).

1. In the formula screen, in the **Formula Steps** section, click the **Insert** button to add a step. This will open the Add Formula Step screen.



2. Enter a **Name** for the step. The name must be unique in the formula (Two steps can be named the same, but must be in different formulae.) This name can be used in subsequent steps as an input argument.
3. The **Formula Id** field is automatically populated with your formula name.
4. Check the **Exclude From Mapping** checkbox if you want to disable the output parameter mapping for the step (see "Formula Parameter Mapping" on page 431 for details). This is useful, for instance, if you wish to safely test your formula without affecting entries in the database.
5. In the **Master Type** field select the data type for the step's result. This can be either Simple Type (a single value) or Collection (a set of multiple values).
   If the master type is a **Collection**, you will have to specify in the **Number of Iterations**

field the formula argument that will be iterated to generate the collection.

For example, to generate a collection with the first x numbers in Fibonacci sequence, use a whole number input argument called x for the **number of iterations** (make sure x is greater than or equal to 2) and use the following formula expression:

“result[0] = 1;

result[1] =1;

result[i] = result[i-1] + result[i-2];”

Simple Type

This stands for the following types of data parameters:

Data type	Example
Whole number	30000 euros - income
Decimal	350.78 GBP - rent expenses
Boolean	Is married - marital status
Text	Construction type e.g. wood - quality of the object for insurance
Object	It is a grouping of arguments that are connected. e.g. “Age”: “WholeNumber”, “Salary”: “Numeric”, “Education”: “Text”, “IsMarried”: “Bool”

Collection

This stands for the following types of data parameters:

Data type	Example
Whole number	30000 euros -income
Decimal	350.78 GBP - rent expenses
Text	List of insurable goods in a house.

> **HINT**
>
> Where the formula body is complex, needs periodic update or must be simplified for transparency and traceability, we suggest you split it in separate steps, each with its own expression. This architecture allows calling the result of step N-1 in step N (“result = step 1 + a;”).

6. In the **Sub-Type** field, enter the data type for the step's result:
   - Whole Number
   - Decimal
   - Boolean - only for Simple Type master types
   - Text - only for Simple Type master types
   - Object - only for Simple Type master types
7. **Calculation Type** can be set as either Normal or as an Iteration, in which case you will have to specify the formula argument that will be iterated in your formula. Current Formula Step cannot be set as iteration argument.
8. Enter the expression for the formula step in the **Formula field**. For details, see "Formula Editor" on page 396.
9. Click the **Save and Close** button ( ) at the top right corner of the page.
10. Repeat for any additional steps you wish to include in your formula. The output from a step can be used as an input argument in subsequent steps.

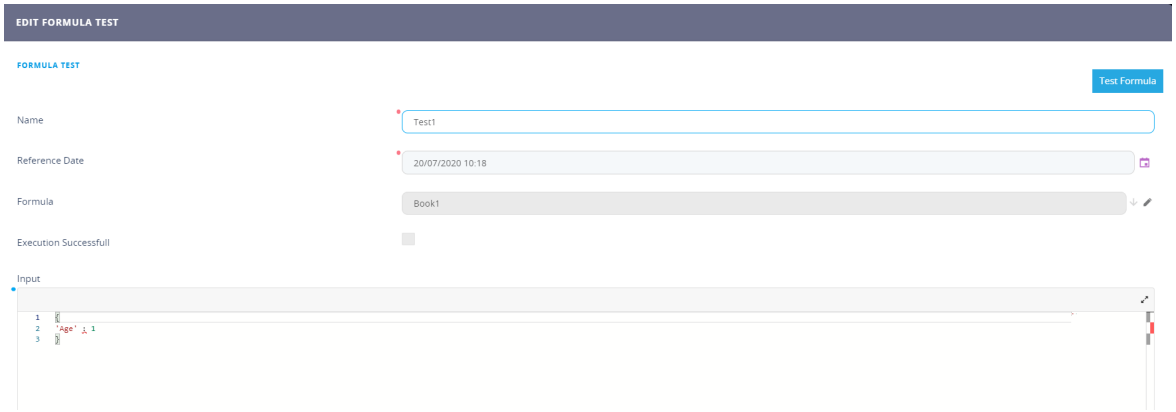
> **NOTE**
>
> New formula steps cannot be added on an active formula. To add steps to a formula create a new versioning (see "Formula Versioning" on the next page).

After you've added all the steps, click the **Save and Close** button ( ) at the top right corner of the page.

> **HINT**
>
> The order used for each step is indicated in the Oder Index column on the right in the grid.

Test Your Formula

You can create various tests for a formula to see how it performs.



To create a formula test:

1. In the formula screen, in the **Formula Tests** section, click the **Insert** button to add a test.
2. Enter a **Name** for your test.

3. Enter a **Reference Date** for the test. By default this will be the current date and time. Depending on the Reference Date that the user chooses, the engine will use the version of the formula that is active at the reference date.
4. The **Formula** field will be automatically populated with the formula name.
5. Click the **Save and Reload** button ( ) in the top right corner of the screen. A new button will appear called ''**Test Formula**''.
6. Edit the **Input** field with the desired input parameters. This field is filled in automatically with the input form the formula and with default values for each argument from this input. The user can just insert some values, not necessarily arguments.
7. Click the **Test Formula** button at the top right corner of the page.
8. Check the **Execution Successful** checkbox and the **Output** field to investigate the formula execution.

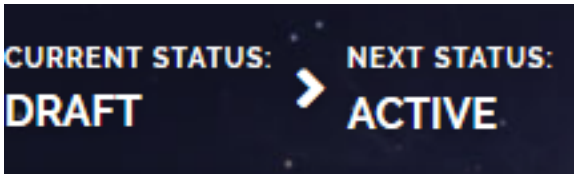
If the result is an error a new text box will appear. In the text box, the system will retrieve the reason for the error.

Formula Versioning

When you first create a formula, it will be in a **Draft** state, meaning that it can be edited and tested, but not used by the system. User cannot create a new version from this state by clicking the button ''Create new version'' on the right-side of the screen.

Activate a Formula

To activate a formula draft, in the formula page, click the **Next Status**: Active field in the top left corner of the screen.

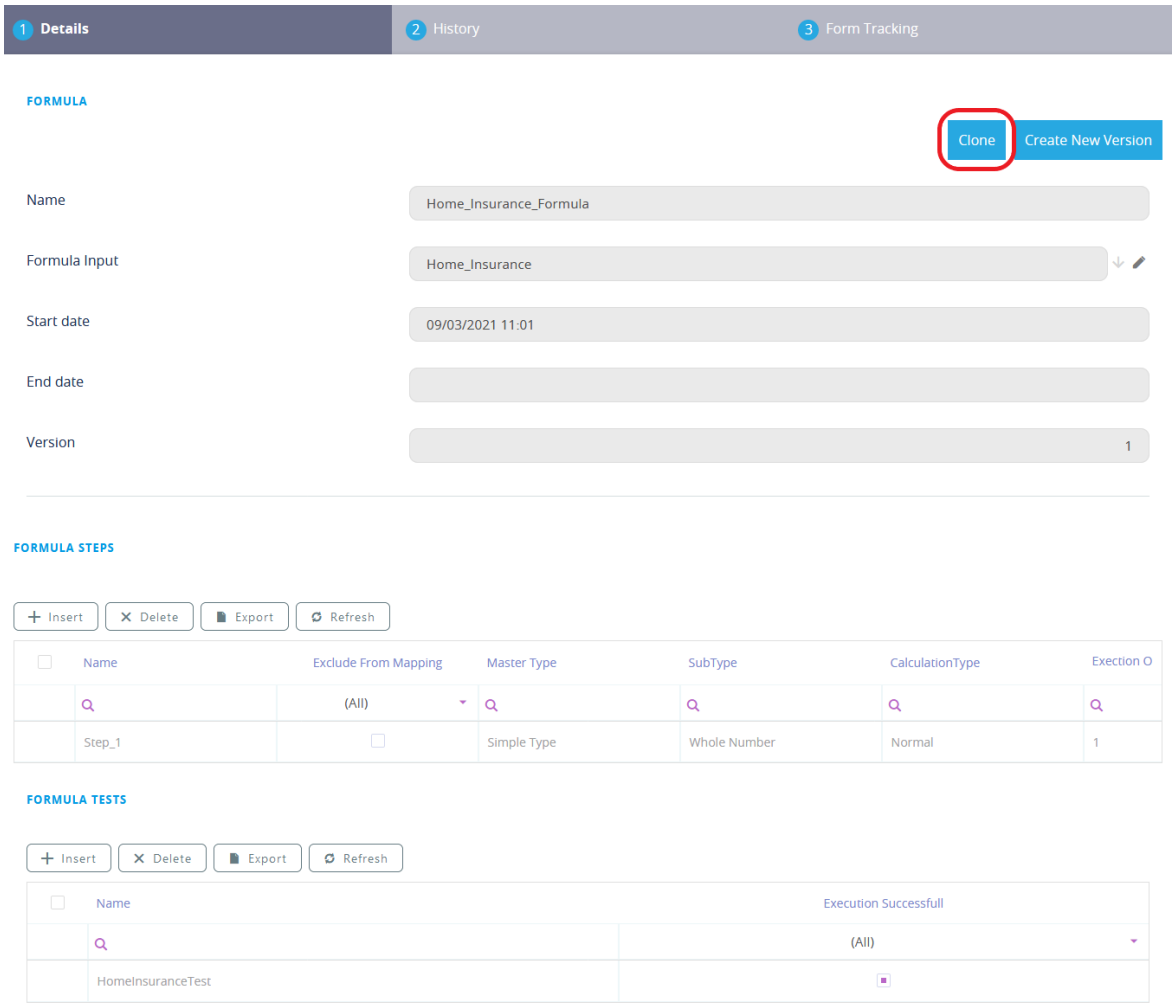


The activation stage implies new conditions for the formula:

- it is possible to create a new version of the formula
- the formula has at least one step
- the form is readonly
- it can only go into the **Closed** state. For any other changes to this formula, a new version needs to be created.
- when a versioned formula goes into **Active** status, the parent formula end date will be updated with the versioned formula start date and its status will be updated to **Closed**.

Clone a Formula

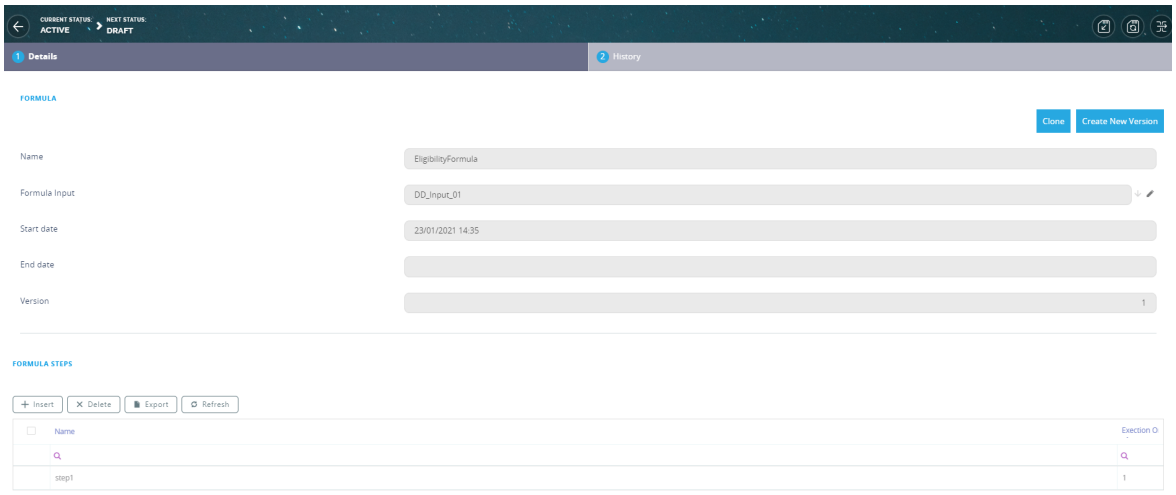
After activation, you can create a clone of that formula by clicking the **Clone** button at the top-right corner of the screen.



This allows you create variations based on the same formula as a starting point.

Create a New Formula Version Draft

Once a formula has been activated, it cannot be modified. Instead, a ''**Create New Version**'' button will appear in the top right corner of your formula page. This button will allow anyone to create a new version of the same formula and a user will be able to make modifications for the new version. It will create a new version with start date today or the start date of the formula if it is in the future.

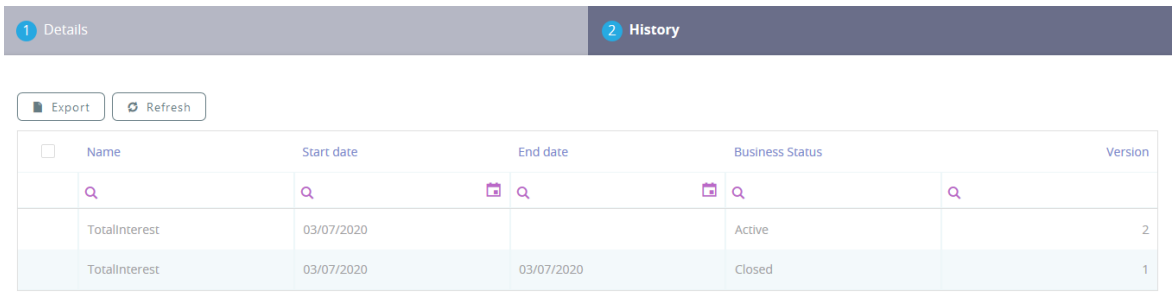


Click the **Create New Version** button to create a new **Draft** version based on the active formula. You can edit the draft version while the active formula is still enabled. It will have the same name as the original one, but the version number will be increased by one, and the start date will be current date, or start date of the previous version plus day. To find the New Formula Version Draft, open the formula you wish to change and click on the ''**History**'' tab where all modifications are shown.

Activate a Formula Version Draft

Once you finish updating the draft version, change its status from **Draft** to **Active** as shown in the "Activate a Formula" on page 392 section. The previously active version will be set to a Closed state, and the draft version will become the currently active version.

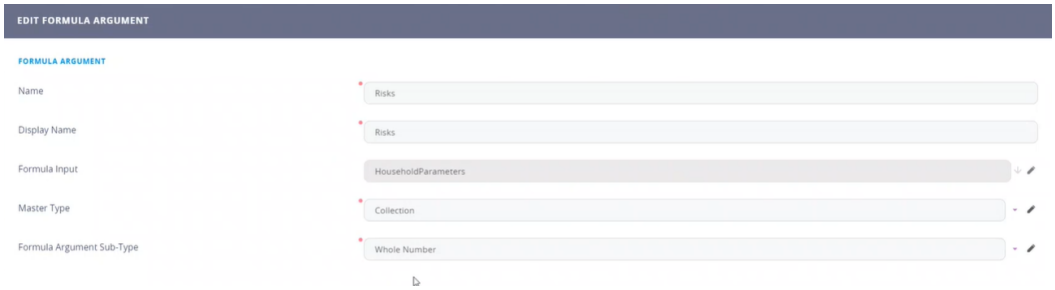
You can track the formula versions in the **History** tab of the formula page.



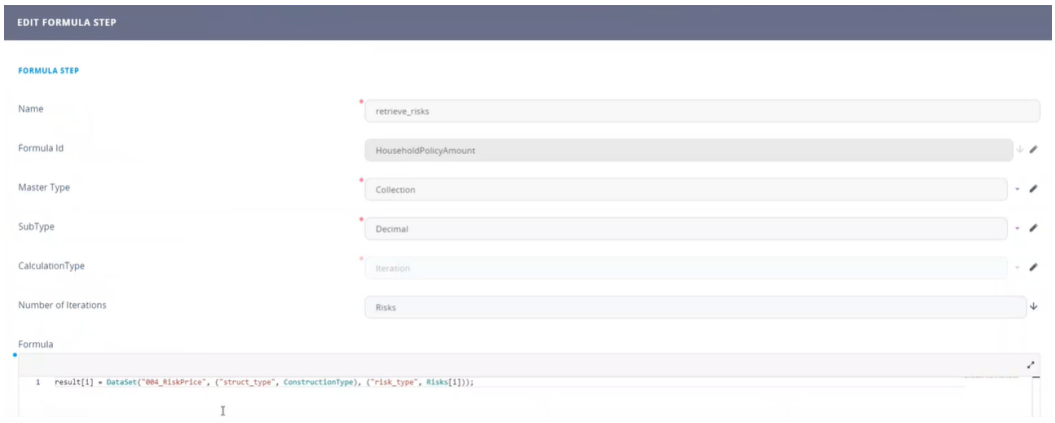
Example

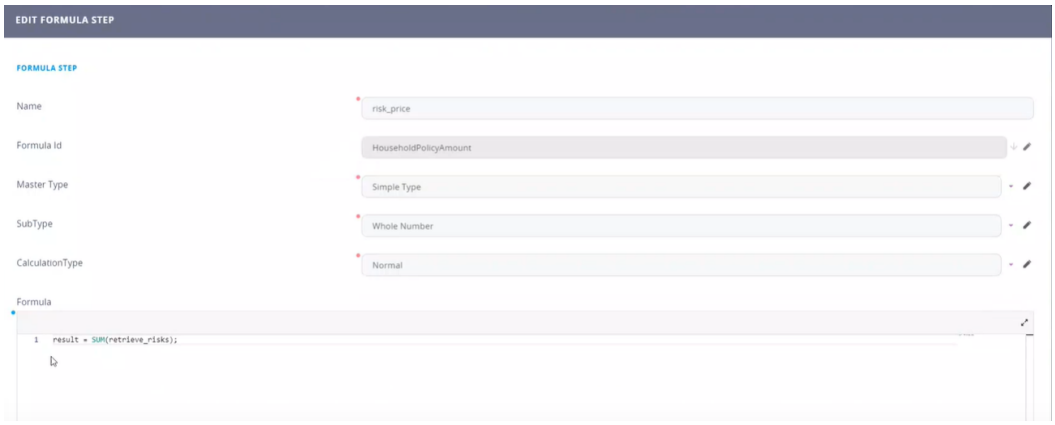
Let's say you have to build a formula to determine the price of the risks for an home insurance. Depending on the construction type and the array of risks, a formula will be written to return the price. The array for the risks are actually coefficients that are whole number e.g. 1, 2, 3, 4, 5.

Firstly, create the input data by creating an argument. In this case we need as arguments the construction type, the structure type and the risks.

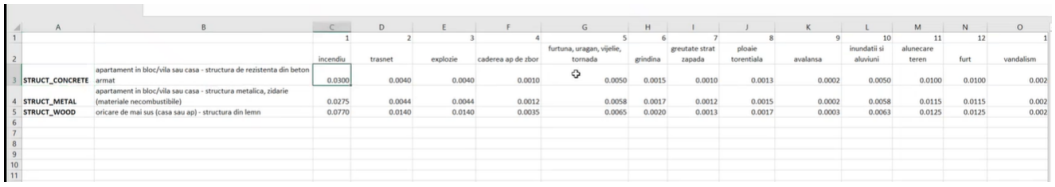


Secondly, create the two steps of the formula, where the first step becomes the input argument of the second step. In the first step, we will calculate the array of prices for the risks and in the second step calculate the sum of those prices.





Thirdly, test the formula by adding a data set and in the formula test grid inserting a structure type, a construction type and an array of risk (the coefficient of those risks that mark a real risk).

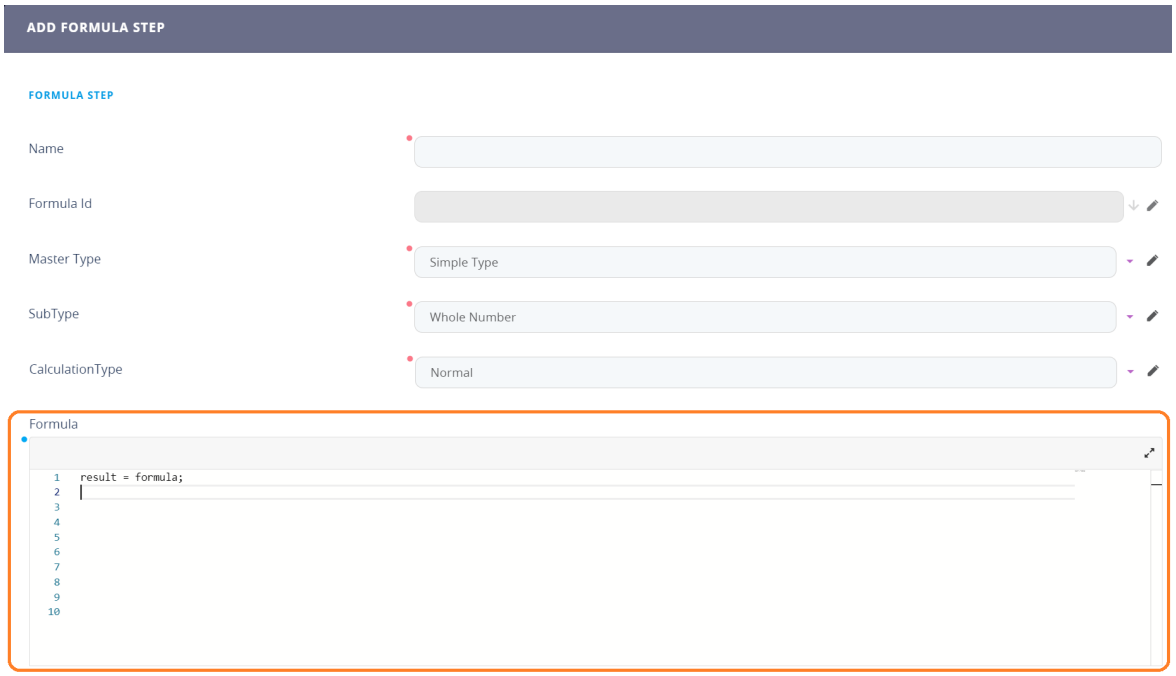


			1	2	3	4	5	6	7	8	9	10	11	12	1
			incendiu	trasnet	explozie	caderea ap de zbor	furtuna, uragan, vijelie, tornada	grindina	greutate strat zapada	ploaie torentiala	avalansa	inundatii si aluviuni	alunecare teren	furt	vandalism
3	STRUCT_CONCRETE	apartament in bloc/vila sau casa - structura de rezistenta din beton armat	0.0300	0.0040	0.0040	0.0010	0.0050	0.0015	0.0010	0.0013	0.0002	0.0050	0.0100	0.0100	0.002
4	STRUCT_METAL	apartament in bloc/vila sau casa - structura metalica, zidarie (materiale necombustibile)	0.0275	0.0044	0.0044	0.0012	0.0058	0.0017	0.0012	0.0015	0.0002	0.0058	0.0115	0.0115	0.002
5	STRUCT_WOOD	oricare de mai sus (casa sau ap) - structura din lemn	0.0770	0.0140	0.0140	0.0035	0.0065	0.0020	0.0013	0.0017	0.0003	0.0063	0.0125	0.0125	0.002

Formula Editor

The FintechOS Formula Editor is a complex tool found in the Studio. It is the place where a user writes an actual mathematical expressions and calculations. The FintechOS Formula Editor is found at the bottom of the “Add/Edit formula step” form.

For example, based on what a client inserts as data in his/hers KYC, the editor takes that information and uses the calculation formula inserted into the Editor to return data such as the net income of a loan applicant or the age limit for a contract or the insurance premium for a policy using complex arguments and functions. It is possible to test the formula inserted here by running a test. For more information about running a test, see "Define Formula Expressions" on page 385.



Syntax

In the Editor, a user inserts the formula that is built based on a given mathematical syntax whose structure is made of the formula body and the call for calculation which is the actual result.

> **IMPORTANT!**
> When writing the formula, make sure that each formula matches to the step you are working on because the result may be used as input in the following steps.

Use the following syntax in the formula editor to define a formula output:

```
result = <formula expression>;
```

For formulas that return collections, use the following syntax (make sure that the formula expression includes an iteration argument):

```
result[i] = < ... iterationArgument[i] ... >;
```

For recursive formulas, define a simple type whole number formula and include a whole number argument in the **Number of Iterations** field (see "Define Formula Expressions" on page 385 for details). For example, to generate a collection with the

first x numbers in Fibonacci sequence, use a whole number input argument called x for the number of iterations (make sure x is greater than or equal to 2) and use the following formula expression:

```
result[0] = 1; result[1] =1; result[i] = result[i-1] + result[i-2];
```

When the formula body is complex, needs periodic update or must be simplified for transparency and traceability, we suggest you split it in separate steps, each with its own expression.

NOTE

You can include multi-line C# code in the formula expression, as long as you assign a result value:

```
var a = 1;
var b = 2;
result = a + b;
```

The order of execution between operations such as addition “+”, substraction “-“, multiplication “*” and division “/” is respected and you can further control this by using parantheses () or by splitting the formula in steps.

In a formula receiving an argument data of a complex type with properties A,B,C of type string and D,E of type numeric, we can write:

```
var result = FROM(data).GROUPBY(["A","B"]).SUM("D");
```

The result will be a collection of items with properties A,B and C, where C will be the SUM for the group determined by the content of properties A and B

Pressing Ctrl+Space will launch the **Intellisense** that can help you learn more about the formula you are editing, keep track of parameters you’re typing, add calls to functions and various information with only a few keystrokes.

Formula can be of the following types:

- constant f = sum(1,n)
- linear f(x) = 2*x+1 , where x = 1,100
- 2-dimensional f(x,y) = x*y+30;

- n-dimensional etc.
- recursive f(x) = f(x-1)+20.

> **HINT**
> Already pre-defined Steps also appear in Intellisense since they can be called in any subsequent step.

Click the ''**Save and Close**'' button.

Formula Arguments

Primary formula arguments are defined in the formula input (see "Define Formula Inputs" on page 381). For example, if we have defined an argument called *days* in our formula input, we can create a formula step called *years* to convert the number of days into years with the following expression:

```
result = days / 365;
```

We can also use previous steps' outputs as arguments for subsequent step inputs. For example, after the above step, we can create a step that calculates an interest by multiplying the *principal* and *rate* input parameters with the *years* result from the previous step:

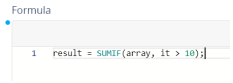
```
result = principal * rate * years
```

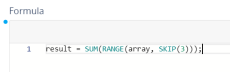
Built-in Functions

You can include the following built-in functions in your formula expressions:

Function	Description	Example	Result
SUM(Vector)	SUM function can be used to add all numbers in a range of cells. The arguments can be numbers, cells references or formula-driven numeric values. For instance, if you would like to calculate the sum of all items part of an array then you need to use an iteration formula.	SUM([1, 2, 3])	6
ABS(Number)	It returns the absolute value of the number.	ABS(-3)	3
POWER(Number, Exponent)	Calculates how many times to use the number in a multiplication.	POWER(10,3)	1000
ODD(Number)	Specifies if the number is odd.	ODD(12)	false
EVEN(Number)	Specifies if the number is even.	EVEN(26)	true
TRUNC(Number)	Truncates the number to a specified number of decimal places.	TRUNC(17.51)	17
ROUND(Number, Precision)	Rounds the number to the specified number of digits.	ROUND(17.51)	18

Function	Description	Example	Result
ROUNDUP(Number)	Rounds the number upward to the specified number of digits.	ROUNDUP(-0.6)	0
ROUNDDOWN(Number)	Rounds the number downward to the specified number of digits.	ROUNDDOWN(1.9)	1
FLOOR(Number)	Receives one parameter and rounds the number down.	FLOOR(6.7)	6
IIF(Condition, TrueExpression, FalseExpression)	IF function is a "conditional function" because it returns a value based on the condition that you specify.		
COUNT(Vector)	Returns the number of numerical values (numbers and dates) in the list of arguments.		
COUNTIF(Vector, FilterExpression)	Counts the number of cells within the range that meet the specified criteria.		
MIN(Vector)	Returns the minimal value from the list of arguments in a row.	MIN(1,2,3,5,6)	1
MAX(Vector)	Returns the maximum value from the list of arguments in a row.	MAX(111,22,33,44,55)	55
AVERAGE(Number1, Number2, Number3, …)	Returns the average of the arguments.	AVERAGE (1234,67543,5752)	24843

Function	Description	Example	Result
AVERAGE(Vector)	Returns the average of the arguments in a row.	AVERAGE ([4643,652348,83284])	246758
SUMIF(Vector, FilterExpression)	Adds up the cells in a specified range that meet a certain condition. SUMIF can evaluate only a single criteria.  "it" is a mandatory term, standing for "item" of an array, for all Filter Expressions in all formulas which evaluate an expression.		
SELECT(Vector, TransformExpression)		SELECT (IncomeList, it * Database (''IncomeAdjuster''))	
RANGE(Vector, RangeOperators)	Navigate inside an array	RANGE ([1,2,3,4,5,6],SKIP(2)])	VECTOR (3,4,5,6)

Function	Description	Example	Result
SKIP(Number)	Combined with RANGE function, you can skip an item of an array, in case for example you would like to add all items except the 3rd one in the array. 		
TAKE(Number)	Combined with RANGE function, you can take as many array items as indicated by the input parameter.		

Function	Description	Example	Result
RATE (nper, pmt, pv, [fv], [type], [guess])	Returns the interest rate per period of an annuity. You can use RATE to calculate the periodic interest rate, then multiply as required to derive the annual interest rate. Arguments: • nper - The total number of payment periods. • pmt - The payment made each period. • pv - The present value, or total value of all loan payments now. • fv - [optional] The future value, or desired cash	RATE(36, 150, -5000)	0.42

Function	Description	Example	Result
	balance after last payment. Default is 0. • type - [optional] When payments are due. 0 = end of period. 1 = beginning of period. Default is 0. • guess - [optional] Your guess on the rate. Default is 10%.		

Function	Description	Example	Result
PMT (rate, nper, pv, [fv], [type])	Calculates the payment for a loan based on constant payments and a constant interest rate. Arguments: • rate - The interest rate for a loan. • nper - The total number of payments for a loan. • pv - The present value, or the total amount that a series of future payments is worth now; also known as the principal. • fv - [optional] The future value, or a	A 3-year (36 months) loan for 10000 USD at 8% interest rate: PMT(0.08/12, 36, 10000)	-313.36

Function	Description	Example	Result
	cash balance you want to attain after the last payment is made. If fv is omitted, it is assumed to be 0 (zero), that is, the future value of a loan is 0. • type - [optional] Indicates when payments are due: • 0 or omitted - at the end of the period • 1 - at the beginning of the period		

Examples

For Simple types

IIF

Returns one of two values, depending on whether the Boolean Condition evaluates to true or false.

Syntax

```
/**
* @param booleanCondition - condition that has to return
true/false
* @param trueValue - the value that is returned if
booleanCondition is evaluated as true
* @param falseValue - the value that is returned if
booleanCondition is evaluated as false
* @returns - return trueValue or falseValue
*/
IIF(booleanCondition, trueValue, falseValue): boolean
```

Example

```
//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 250
*/
result = IIF(COUNT(simpleCollection) > 0, SUM
(simpleCollection)/COUNT(simpleCollection) , 0);
```

AVERAGE

Returns the average of the numbers received as parameters.

Syntax

```
/**
* @param {number} number1 - required, the first number for
which you want the average
* @param {number} number2, ... - optional, additional
numbers for which you want the average
* @returns {number} - the average of the numbers
```

```
*/
AVERAGE(number1, [number2], ...): number
```

Example

```
/*
Input:

Output:
    result = 250
*/
result = AVERAGE(100, 200, 300, 400);
```

ABS

Returns the absolute value of a number

Syntax

```
/**
* @param {number} number - the number of which you want the
absolute value
* @returns {number} - the absolute value
*/
ABS(number): number
```

Example

```
/*
Input:
    num = -100
Output:
    result = 100
*/
result = ABS(num);
```

POWER

Raises a number to a power

Syntax

```
/**
* @param {number} number - the base number
* @param {number} exponent - the exponent to which the base
number is raised
* @returns {number} -
*/
POWER(number, exponent): number
```

Example

```
/*
Input:
    num = 10
    exp = 2
Output:
    result = 100
*/
result = POWER(num, exp);
```

ODD

Returns true if the integer number is odd, otherwise false.

Syntax

```
/**
* @param {number} number - the number the needs to be
verified
* @returns {boolean} -
*/
ODD(number): boolean
```

Example

```
/*
Input:
    num = 3
Output:
```

```
    result = true
*/
result = ODD(num);

/*
Input:
    num = 2
Output:
    result = false
*/
result = ODD(num);
```

EVEN

Returns true if the integer number is even, otherwise false.

Syntax

```
/**
* @param {number} number - the number the needs to be
verified
* @returns {boolean} -
*/
EVEN(number): boolean
```

Example

```
/*
Input:
    num = 3
Output:
    result = false
*/
result = EVEN(num);

/*
Input:
    num = 2
Output:
    result = true
*/
```

```
result = EVEN(num);
```

TRUNC(Number)

Calculates the integral part of a specified decimal number.

Syntax

```
/**
* @param {number} number - the number the needs to be
truncated
* @returns {number} -  the integral part of the number
*/
TRUNC(number): number
```

Example

```
/*
Input:
    num = 17.53M
Output:
    result = 17
*/
result = TRUNC(num);

/*
Input:
    num = -17.53M
Output:
    result = 17
*/
result = TRUNC(num);
```

ROUND(Number, [Precision])

Rounds a decimal value to a specified number of fractional digits.

Syntax

```
/**
* @param {number} num - the number the needs to be rounded
* @param {number} precision - optional,  number of decimal
places in the return value. The default value is 0
* @returns {number} -  The number nearest to num that
contains a number of fractional digits equal to precision.
*/
ROUND(num, [precision]): number
```

Example

```
/*
Input:
    num = -17.51M
Output:
    result = -18
*/
result = ROUND(num);

/*
Input:
    num = 17.51M
    precision = 1
Output:
    result = 17.5
*/
result = ROUND(num, precision);
```

ROUNDUP

Returns the smallest integral value that is greater than or equal to the specified decimal number.

Syntax

```
/**
* @param {number} num - the number the needs to be rounded
up
* @returns {number} -  the smallest integral value that is
greater than or equal to the specified decimal number.
*/
```

```
ROUNDUP(num): number
```

Example

```
/*
Input:
    num = -17.51M
Output:
    result = -17
*/
result = ROUNDUP(num);

/*
Input:
    num = 17.51M
Output:
    result = 18
*/
result = ROUNDUP(num);
```

ROUNDDOWN

Returns the largest integer less than or equal to the specified decimal number.

Syntax

```
/**
* @param {number} num - the number the needs to be rounded
down
* @returns {number} -  the largest integer less than or
equal to the specified decimal number.
*/
ROUNDDOWN(num): number
```

Example

```
/*
Input:
```

```
    num = -17.51M
Output:
    result = -18
*/
result = ROUNDDOWN(num);

/*
Input:
    num = 17.51M
Output:
    result = 17
*/
result = ROUNDDOWN(num);
```

FLOOR

Returns the largest integer less than or equal to the specified decimal number.

Syntax

```
/**
* @param {number} num - the number
* @returns {number} -  the largest integer less than or
equal to the specified decimal number.
*/
FLOOR(num): number
```

Example

```
/*
Input:
    num = -17.51M
Output:
    result = -18
*/
result = FLOOR(num);

/*
Input:
    num = 17.51M
```

```
Output:
    result = 17
*/
result = FLOOR(num);
```

For Collection types

SELECT

Applies a function to each element of the collection and returns a new collection with the results of the function invocation.

```
//Example
/*
Input:
    productCollection = [
        {"name":"Product1", "priceWithVAT": 100 },
        {"name":"Product2", "priceWithVAT": 200 }]

Output:
    result = [100, 200]
*/
result = FROM(productCollection).SELECT(x=>x.priceWithVAT);

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = [200, 400, 600, 800]
*/
result = FROM(simpleCollection).SELECT(x=>x * 2);
```

WHERE

Applies a function to each element of the collection and returns a new collection with the filtered elements that respect the condition

```
//Example
/*
Input:
    productCollection = [
        {"name":"Product1", "priceWithVAT": 100 },
        {"name":"Product2", "priceWithVAT": 200 }]

Output:
    result = [200]
*/
result = FROM(productCollection).WHERE
(x=>x.priceWithVAT>150).SELECT(x=>x.priceWithVAT);
```

FIRSTORDEFAULT

Returns first element of the collection or the default value (0 for numeric elements). It goes well when used with WHERE and you are sure only one record is returned. ```javascript //Example /* Input: productCollection = [{"name":"Product1", "priceWithVAT": 100 }, {"name":"Product2", "priceWithVAT": 200 }]

Output:

result = 200 */ result = FROM(productCollection).WHERE (x=>x.priceWithVAT>150).SELECT(x=>x.priceWithVAT).FIRSTORDEFAULT(),

```
### GROUPBY ###
**Can only be used with agregate function SUM**

Returns a new collection grouped by a property of the
object.
```javascript
//Example
/*
Input:
    productCollection = [
        {category: "Cat1", name:"Product1", priceWithVAT:
100, quantity: 1 },
        {category: "Cat2", name:"Product2", priceWithVAT:
200, quantity: 2 },
        {category: "Cat1", name:"Product3", priceWithVAT:
300, quantity: 3 }]
```
```

```
Output:
    result = [
      {
        "category": "Cat1",
        "priceWithVAT": 400.0
      },
      {
        "category": "Cat2",
        "priceWithVAT": 200.0
      }
    ]
*/
result = FROM(productCollection).GROUPBY("category").SUM
("priceWithVAT");
```

EXTENDELEMENTS

Returns a new collection with a new property added to all elements in the collection. Also, for each element it assigns a value for the new added property.

```
//Example
/*
Input:
    productCollection = [
        {category: "Cat1", name:"Product1", priceWithVAT:
100, quantity: 1 },
        {category: "Cat2", name:"Product2", priceWithVAT:
200, quantity: 2 },
        {category: "Cat1", name:"Product3", priceWithVAT:
300, quantity: 3 }]

Output:
    result = [
      {
        "category": "Cat1",
        "name": "Product1",
        "priceWithVAT": 100.0,
        "quantity": 1.0,
        "totalPriceWithVAT": 100.0
      },
```

```
      {
        "category": "Cat2",
        "name": "Product2",
        "priceWithVAT": 200.0,
        "quantity": 2.0,
        "totalPriceWithVAT": 400.0
      },
      {
        "category": "Cat1",
        "name": "Product3",
        "priceWithVAT": 300.0,
        "quantity": 3.0,
        "totalPriceWithVAT": 900.0
      }
    ]
*/
result = FROM(productCollection).EXTENDELEMENTS
("totalPriceWithVAT", x=>x.priceWithVAT * x.quantity);
```

Count

Count is a property of collections, it returns the number of elements from the collection.

```
//Example
/*
Input:
    productCollection = [
        {"name":"Product1", "priceWithVAT": 100 },
        {"name":"Product2", "priceWithVAT": 200 }]

Output:
    result = 2
*/
result = FROM(productCollection).WHERE
(x=>x.priceWithVAT>=100).Count;
```

For Simple Collections types

RANGE with SKIP and/or TAKE

Generates a sequence of numbers within a specified range.

Syntax

```
/**
* @param {number[]} simpleCollection - simple collection of
decimals
* @param rangeOperators - range operators of type SKIP and
TAKE
* @returns {number[]} -  the sequence of numbers within a
specified range.
*/
RANGE(simpleCollection, rangeOperators...): number[]
```

Example

```
//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = [100, 200]
*/
result = RANGE(simpleCollection, TAKE(2));

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = [200, 300]
*/
result = RANGE(simpleCollection, SKIP(1), TAKE(2));

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = [200, 300, 400]
*/
```

```
result = RANGE(simpleCollection, SKIP(1));
```

MIN / MAX

Returns the min/max from the collection.

Syntax

```
/**
* @param {number[]} simpleCollection - simple collection of
decimals
* @returns {number} -
*/
MIN(simpleCollection): number
MAX(simpleCollection): number
```

Example

```
//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 100
*/
result = MIN(simpleCollection);

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 400
*/
result = MAX(simpleCollection);
```

AVERAGE

Returns the average of the collection of numbers received as parameter

Syntax

```
/**
* @param {number[]} simpleCollection - simple collection of
decimals

* @returns {number} - the average of the numbers
*/
AVERAGE(simpleCollection): number
```

Example

```
//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 250
*/
result = AVERAGE(simpleCollection);
```

SUM / COUNT

Returns the sum/count of the elements from the collection.

Syntax

```
/**
* @param {number[]} simpleCollection - simple collection of
decimals
* @returns {number} -
*/
SUM(simpleCollection): number
COUNT(simpleCollection): number
```

Example

```
//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 1000
*/
result = SUM(simpleCollection);

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 4
*/
result = COUNT(simpleCollection);
```

SUMIF / COUNTIF

Returns the sum/count of the elements from the collection that respect the condition

Syntax

```
/**
* @param {number[]} simpleCollection - simple collection of
decimals
* @param  filter - the condition for filtering
* @returns {number} -
*/
SUMIF(simpleCollection, filter): number
COUNTIF(simpleCollection, filter): number
```

Example

The it from the filter is just a convention for naming an item of the collection.

```
//Example
```

```
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 700
*/
result = SUMIF(simpleCollection, it > 250);

//Example
/*
Input:
    simpleCollection = [100, 200, 300, 400]

Output:
    result = 2
*/
result = COUNTIF(simpleCollection, it > 250);
```

Data Set Calls

To extract value mappings from data sets (see "Data Sets" on the next page for details), use the following syntax:

```
DataSet("<data set name>", ("<discriminant 1 name>", <discriminant 1 value>), ("<discriminant 2 name>", <discriminant 2 value>) ... )
```

If all the discriminants are given as a parameter, the result will be a value. However, if one of the discriminants is not sent as a parameter, the result will be an array.

IMPORTANT!
You can only call data sets that are active. For details, see "Data Set Versioning" on page 430.

Example

In the example below, we return a risk coefficient from a data set called *RiskPrice* with two discriminants (*structure_type* and *risk_type*). We retrieve the value corresponding to the structure and risk stored in the *myStructure* and *myRisk* input parameters.

> ⚠ **IMPORTANT!**
> The sub-types that are text are written using quotation marks.

```
result = DataSet("RiskPrice", ("structure_type",
myStructure), ("risk_type", myRisk));
```

Data Sets

Data sets are mappings that associate values for a set of discriminants (such as age, sex, driving experience, etc) to values for a specific metric (such as a risk coefficient). You can call data sets from your functions, allowing you to reference the predefined metric value that matches a set of arguments (for details, see "Data Set Calls" on the previous page).

Data set values are populated by importing mapping values from Excel files and/or by manual entries.

For example, you can import a file with coefficients for the age of a client.

	A	B
1	Age	Value
2	[;17]	0
3	[18;25]	50
4	[26;35]	100
5	[36;50]	150
6	[51;]	75

Create a Data Set

Data set discriminants is the grid where a user can teach the system the two dimensional data in rows and columns.

The value of an argument is dependent to a discriminant contained in the Excel file imported. For example, a column or row could be the age of the client, his income, his status, the item insured, the type of car, risk zones etc. For each of this text items you can set coefficients in Excel.

> ⚠ **IMPORTANT!**
> If the Excel file has more than one sheet, those other sheets will not be imported. Please, import one sheet at a time.

1. Open the Main Menu ( ) in Innovation Studio.
2. Select **Business Formulas**.
3. Select **Data Set**.
4. Click the **Insert** button ( ) at the top right corner of the page.
5. Enter a **Name** for the data set. This is a unique name used to identify the data set in the system.
6. Optionally enter a **Description** for the data set.
7. In the **Value Types** field, select if the data set returns **Numeric** or **Text** values.
8. Check the **Single Value** box if the data set includes a mapping between a single discriminant key and a single metric value.
9. Select the **Start Date** at which the data set becomes active. Select the time as well. This makes it possible to activate and later on deactivate the formula for minute to minute.
10. The **End Date** and **Version** fields are populated automatically based on the data set's versioning (see "Data Set Versioning" on page 430 for details).
11. The **Has Column Description** and **Has Row Description** checkboxes indicate that the Excel files used to import value mappings include field descriptions in the second row and/or second column (key values are set in the first row and first column).
12. Click the **Save and Reload** ( ) button at the top right corner of the page.

Define Data Set Discriminants (non Single Value data sets)

1. In the data set screen, in the **Data Set Discriminants** section, click the **Insert** button to add a discriminant.
2. Enter a **Name** for the discriminant.
3. The **Data Set** field is automatically populated with your data set name.
4. Optionally enter a **Description** for the discriminant.
5. In the **Values Type** field, select the data type for the discriminant. This can be either **Text**, **Numeric**, or **Option Set**. In the case of option sets, an **Option Set** field will be displayed, allowing you to select the option set that the discriminant values must belong to (see "Adding Option Set Attributes" on page 71 for details).
6. Select **Is Interval** if the discriminant values are in the form of a value range.
    - Intervals must be entered using the following syntax: [; *<maximum value>*], [*<minimum value>*; *<maximum value>*], or [*<minimum value>*;].
    - If the minimum value is not specified, the interval covers all values smaller than or equal to the maximum value. If the maximum value is not specified, the interval covers all values greater than or equal to the minimum value.
    - You can use closed intervals […], open intervals (…), or half-open intervals […) (…].
7. Select **Is Row Key** if discriminant values are represented in the first column of the imported Excel file used to add data set values (see "Add Data Set Values (non Single Value data sets)" on the next page). You can have a only one row key discriminant per data set.
8. Select **Is Column Key** if discriminant values are represented in the first row of the imported Excel file used to add data set values (see "Add Data Set Values (non Single

Value data sets)" on the next page). You can have only one column key discriminant per data set.

9. Click the **Save and Close** button ( ) at the top right corner of the page.
10. Repeat for any additional discriminants you wish to include in your formula.

Add Data Set Values (single value data sets)

For this kind of data set, no Excel file is needed. The values are inserted in the form and the ''**Start Import**'' button is missing.

1. In the data set screen, in the **Data Set Values** section, click the **Insert** button.
2. The **Data Set** field is automatically populated with your data set name.
3. Enter a **Name** for the value mapping.
4. Optionally enter a **Description** for the value mapping.
5. Click the **Save and Reload** ( ) button at the top right corner of the page.
6. 6. At the bottom of the page:

   i. Enter the discriminant value in the left column.

   ii. Enter the metric value in the right column.
7. Click the **Save and Close** button ( ) at the top right corner of the page.

Add Data Set Values (non Single Value data sets)

Data set value mappings are imported via Excel files for discriminants that are column/row keys and entered manually for discriminants that are not.

HINT
If you your data set includes more than two discriminants, set the discriminants with the top two highest cardinalities as row key and column key. This way, you will need fewer sets of values.
For instance, if you have 3 discriminants, such as age (4 age brackets), education (5 education levels), and sex (2 sexes), you can set age and education as row and column key values. This will allow you to populate the data set with only two value

sets: one Excel file with age/education mappings for males and another with age/education mappings for females.

1. In the data set screen, in the **Data Set Values** section, click the **Insert** button to add a set of values.
2. The **Data Set** field is automatically populated with your data set name.
3. Enter a **Name** for the set of values.
4. Optionally enter a **Description** for the set of values.
5. Insert in the **Import file** the Excel file you wish to use.
6. Click the ''**Start Import**'' button on the right side of the corner.
7. Click the **Save and Reload** ( ) button at the top right corner of the page.
8. Enter keys for any non-row/column key discriminants:
    i. Click the **Insert** button in the Data Set Discriminant Values section.
    ii. Enter a **Name** for the discriminant.
    iii. The **Data Set Value** is populated automatically.
    iv. Select the corresponding **Data Set Discriminant** from the list.
    v. Enter the discriminant key value in the **Discriminant Value Text** field.
    vi. Click the **Save and Close** button ( ) at the top right corner of the page.
    vii. Repeat for any remaining non-row/column key discriminants.
9. Click the **Add file** button to select the Excel import file for the row/column key discriminants. The Excel file must match the discriminants' data type settings and formatting (description row/description column).
10. Click the **Start Import** button at the top right corner of the page. After the import finishes, the imported data will be displayed at the bottom of the page.

11. Click the **Save and Close** button ( ) at the top right corner of the page.

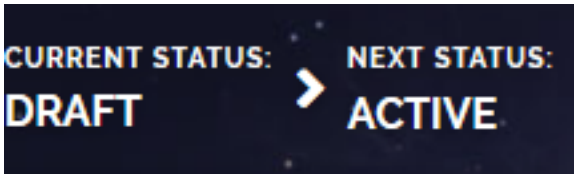
12. Repeat for any additional value sets you wish to include in your data set.

Data Set Versioning

When you first create a data set, it will be in a **Draft** state, meaning that it can be edited, but not used by the system.

Activate a Data Set

To activate a data set draft, in the data set page, click the **Next Status: Active** field in the top left corner of the screen.

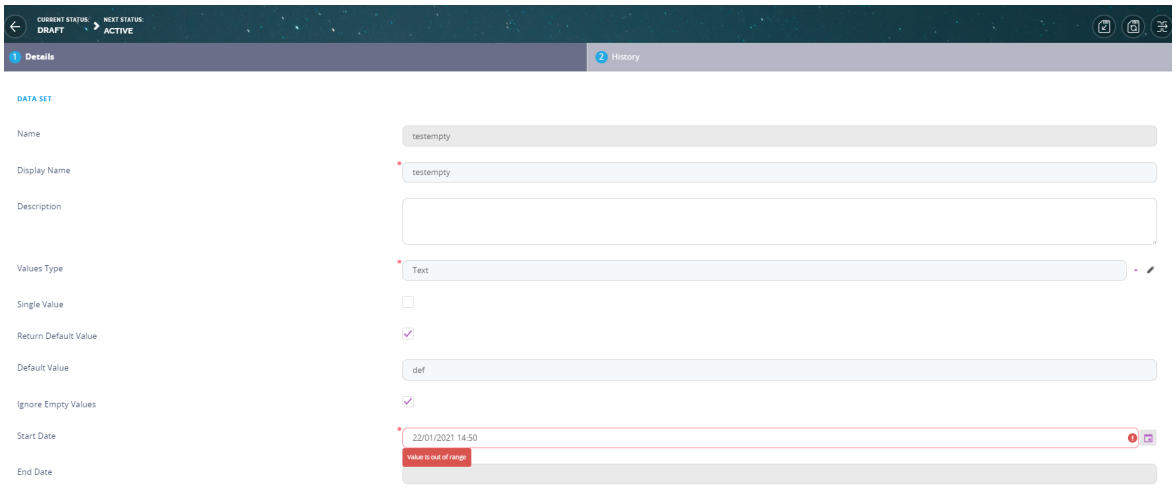


Once activated, a data set can be used in formulas (see "Data Set Calls" on page 424 for details).

Create a New Data Set Version Draft

An active data set cannot be modified. Instead, a Create New Version button will appear in the top right corner of your data set page.

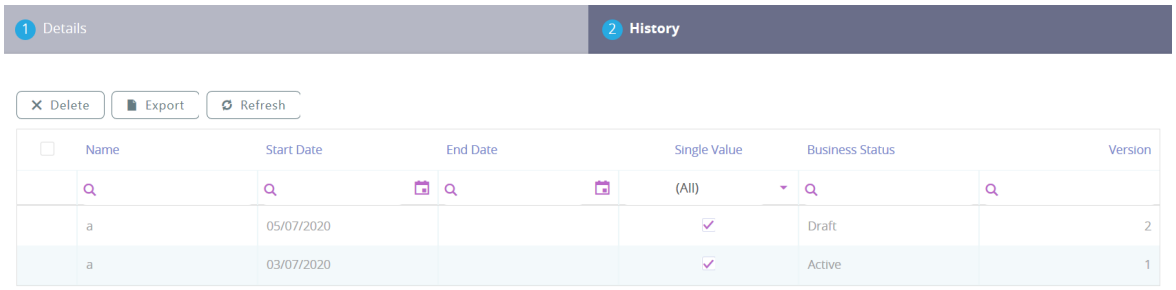
Click the **Create New Version** button to create a new draft version based on the active data set. You can edit the draft version while the active data set is still enabled. It will create a new version with start date today or the start date of the data set it is in the future. If the new version created takes the current date as start date (the date and time when it was created e.g. 10:45) then the start date will become out of range when displayed and it will have to be changed.



Activate a Data Set Version Draft

Once you finish updating the draft version, change its status from Draft to Active as shown above. The previously active version will be set to a Closed state, and the draft version will become the currently active version.

You can track the data set versions in the **History** tab of the data set page.

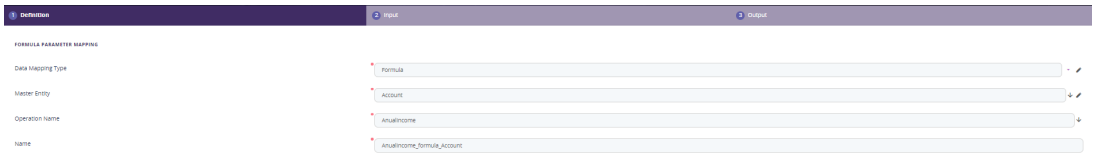


Formula Parameter Mapping

After defining a formula, as by the configurations in "Define Formula Expressions" on page 385, create the mapping needed for the use of the formula. To do so:

1. Open the Innovation Studio, open the main menu, select the Business Formulas, click on the Formula Parameter Mapping.

2. Click the ''Insert'' button to add a new one or click the ''Delete'' button to erase a mapping.
3. To create a new one, fill in the following:



Field	Data type	Description
Data Mapping Type	Option set	Select the type from the list: • formula • insurance type.
Master Entity	Option set	Choose the corresponding entity from where the user wishes to get the data. **NOTE** Be sure to select the same entity as the entity for the formula created before.
Operation Name	Option set	Choose the formula from the list.
Name	Text	Insert a name for the mapping.

4. Click the ''Save and reload'' button.
5. Click on ''**Input**'' to map the input to the fields from the entity.
6. Click the ''Save and reload'' button.
7. Select ''**Output**'' to map the results from the formula to the fields from the account.

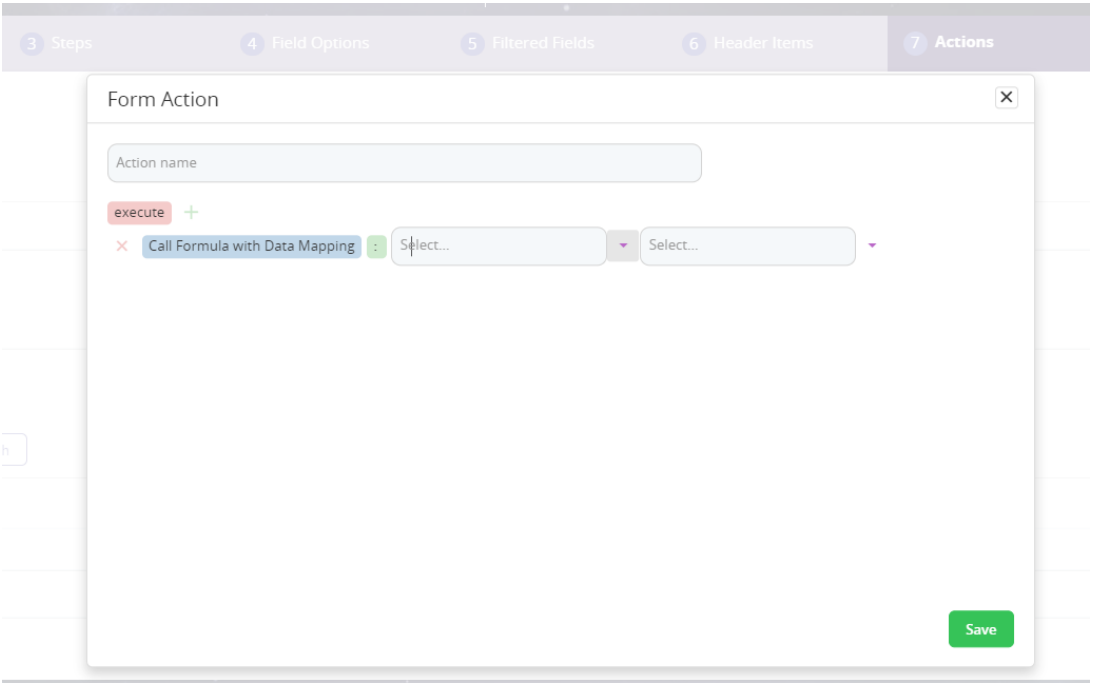
8. Click the ''Save and reload'' button.

Calling the Business Formulas

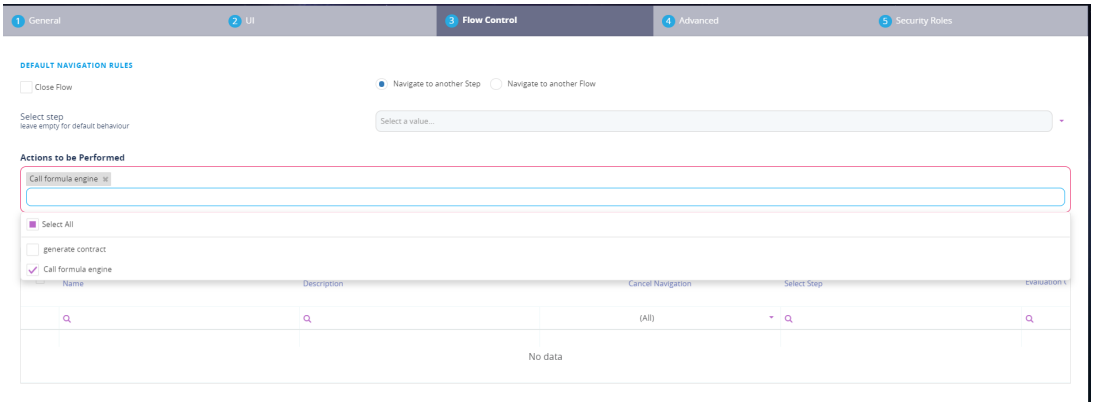
After you configured a data set, a formula expression in the editor, and activated the formula, you can use the formula in a journey or entity form.

Calling Formulas in a form-driven flow (no code)

1. Open Innovation Studio, open the main menu and select **Digital Journey**.
2. Open the digital journey you wish to work on.
3. Click on **7. Actions**, then click on the ''**Insert**'' button.
4. Click on + and select the call formula with data mapping and select the formula you wish to execute and he mapping you have done.



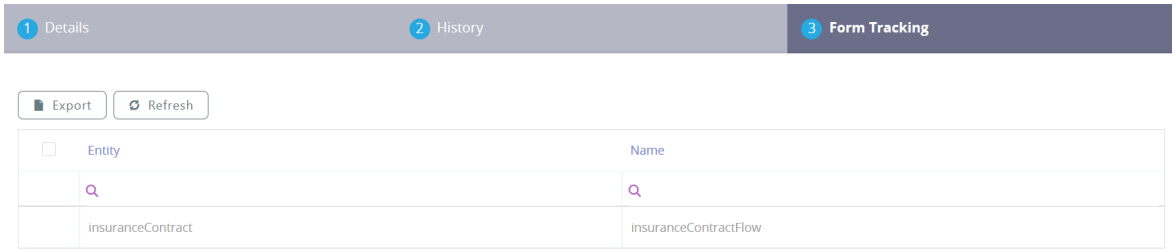
5. Click ''**Save**'' and then click ''**Save and reload**''.
6. Continue to add as any as you have or need.
7. Open each step and add the action by clicking the ''3. Flow control''.



8. In the section ''Actions to be performed'' add the action that calls the formula.
9. Click the **Save and Reload** button.

Track form driven flows that are using a specific formula

After you configure a form driven flow to call a formula, the form will be listed in the Form Tracking tab of the formula's edit form.



Call formulas using server side scripts

Formulas can be called from server side scripts using the following method:

```
/**
 * Call formula by name
 * @param formulaName is the name of the formula
 * @param input is the input that must be provided in order to
compute the formula
 * @param options are the formula runtime options of type
IFtosRunFormulaOption. A property that can be set here is
referenceDate if you need to call a past version of the formula
 * @return the object with the calculated values
*/
server.formulas.runFormula(formulaName: string, input: any,
options: any): any
```

Example

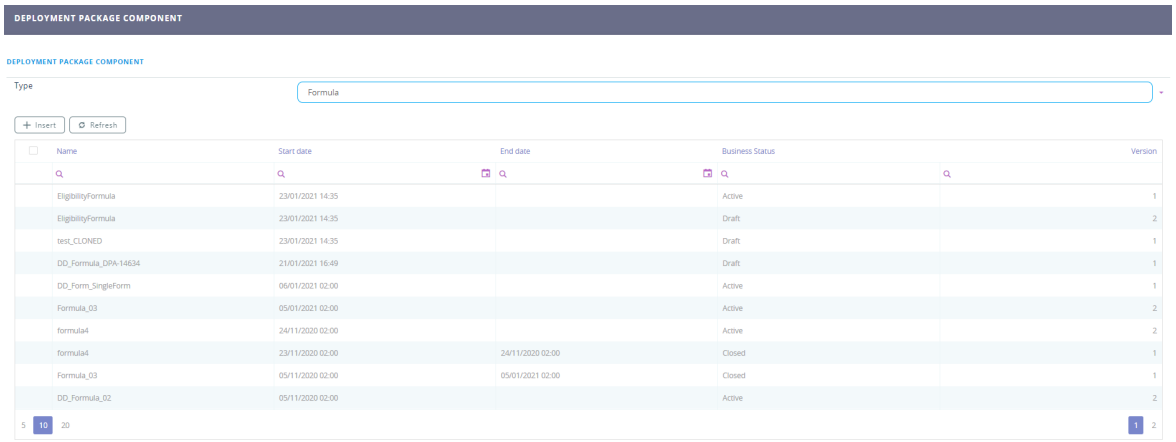
```
var input = {
    age: 20,
    region: "test"
}
server.formulas.runFormula("formulaName", input, {});
```

For more information, see the Server SDK Reference Guide.

Export a Formula or Data Set

When creating a package to be exported from the current environment to another environment, the digital actor is going to select the formula version to be included in the deployment package, regardless of its status:

- draft
- active
- closed.



To deploy a package containing a formula from one environment to another, the system will proceed as indicated below by checking:

- the formula name to see if there are any existing formulas on the deployment environment. If the formula exists on the deployment environment, the input parameter and output formula steps will be checked and if they are the same (environment vs. package) the deployed formula will become the latest version of the existing formula.
- on the deployment environment if the existing formula has a draft version. If there is an existing draft version the system will display an error message on importing the

package. (The user will have to remove/delete the draft version to deploy/import the formula from the package.)

- If there is no draft version, the deployed formula will be imported with Draft status, begin date = current date when beginning data < current date and all links of parent formula and master formula will be reinitialized.
- If the input and output parameters are different from package formula the formula could be imported as a new one with change name "name_importdate" or the import will raise an error.
- If the name is not found then the formula will be imported as a new one, in draft status, version 1 and begin date = current date.

Analytics

Innovation Studio aggregates data from various sources, supporting a centralized view of data and omni-channel data insights.

FintechOS has embedded Microsoft Power Bi integration allowing you to use more complex and interactive analytic dashboards.

The dashboards allow aggregating lists fed from the open data model with the most relevant data (e.g., to-do lists ordered by SLA or severity) and enable the usage of security roles to manage data visibility and ownership.

Advanced Analytics **439**

Register App for Power BI 440

Embed Power BI Report 445

Add Power BI Report to Dashboard 448

How to add Power BI Reports to Digital Journeys 450

Custom Reports **454**

Creating a custom report 454

Tabular Reports **457**

STEP 1. Add Data Source and Parameters 458

STEP 2. Add Report Parameters 460

STEP 3. Add Simple Grid Report 461

STEP 4. Add Report Items 462

STEP 5. Define Report Access Privileges 463

Charts **463**

Creating charts 464

Advanced Analytics

As organizations own large amount of data, emerging technologies have bought in new ways to deal with data, analyze it, and understand the business trends.

Advanced analytics bring autonomous data examination using business intelligence (BI) to process large amounts of unstructured data, discover deeper insights, make predictions, and generate recommendations.

Integrated into FintechOS, Microsoft Power BI brings advanced analytics to process and transform data into coherent, visually immersive, and interactive insights, thus providing solutions to your business problems. With the use of data mining and various BI systems, it helps identify data patterns, an important analytical need when trying to obtain meaningful, useful, and actionable information hidden in data through data analysis and exploration.

Microsoft Power BI integrated into Innovation Studio helps analyze data, determine which metrics are driving more opportunities and success, and share insights across all levels of an organization. In addition, you can embed interactive reports and visuals into your app.

A Power BI report is a multi-perspective view into a dataset with visualizations that represent different findings and insights from that dataset.

To use a Microsoft PowerBI Report in Innovation Studio, follow these steps:

1. Register App for Power BI
2. Embed Power BI Report
3. Add Power BI Report to Dashboard
4. Add Power BI Reports to Digital Journeys

Register App for Power BI

Prerequisites

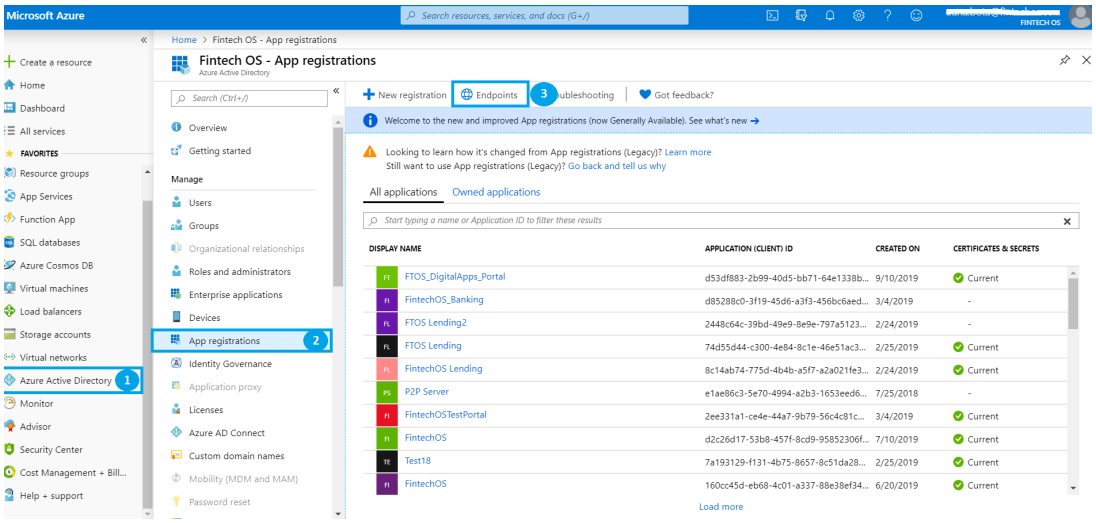
- You should know the ID of your company's **Azure Active Directory** domain (also known as tenant URL).
- You should have a **Power BI** account.

How to find the tenant URL

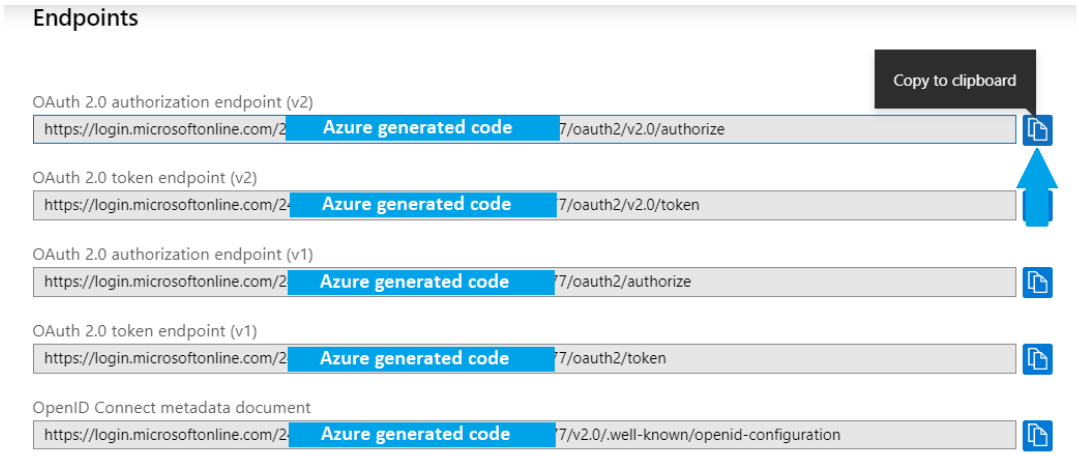
NOTETo find the tenant URL, you should have a **Microsoft Azure** account provided by your organization.

Follow these steps to find the tenant URL:

1. Go to https://portal.azure.com/. You will be automatically logged in with your **Microsoft** account.
2. From the main menu, click **Azure Active Directory**. Your company's **Azure Active Directory** overview appears.
3. From the left side **Manage** menu, click **App registrations**. The list of all apps registered by your company in **Microsoft Azure** appears.



4. On the toolbar, click **Endpoints** to open the list of available endpoints.



5. In the list, search for the **OAUTH 2.0 authorization endpoint** regardless the version, and click the **Copy to clipboard** icon corresponding to the endpoint.

You will need the tenant URL after you register the app for **Power BI** in **Azure**.

Register app for Power BI

Once you configure the **Power BI** reports in the user interface, you need to register your application for **Power BI**:

1. Access the **Power BI** registration link. Follow on-screen instructions.

   > **NOTE** You need to log in to your **Power BI** account to be able to register your app for **Power BI**.

2. In the **App Name** field type a descriptive name for your app as it will be displayed on the login page. The **App Type** field is set by default to **Server-side Web App**.

3. In the **Redirect URL** field, type the application URL and add the suffix **Azure/Redirect.asp**.

   E.g., `https://188.210.90.229/EBSCore_CRM/Azure/Redirect.aspx`

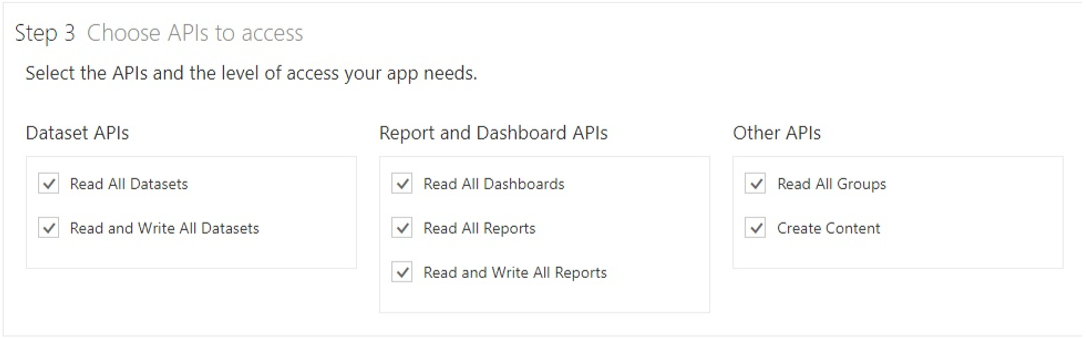
   > **NOTE** Provide a secure redirect URL, otherwise the app registration will fail.

4. In the **Home Page URL** field, type the application URL and add the suffix `/Main`.

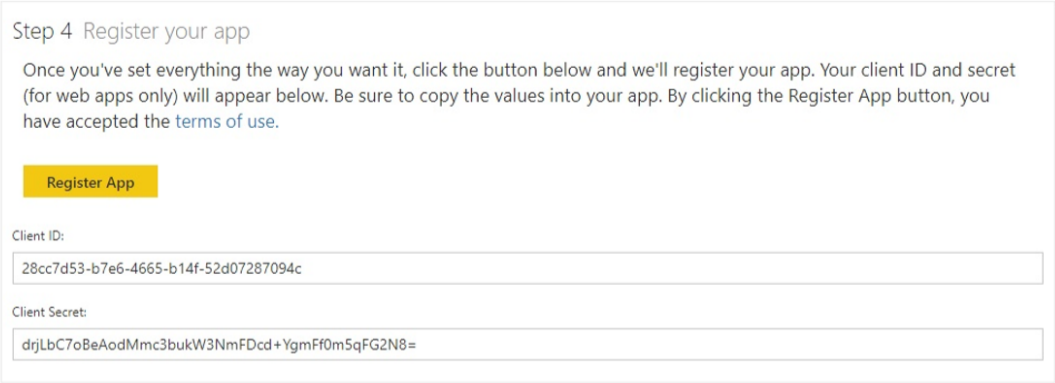
   E.g., `https://188.210.90.229/EBSCore_CRM/Main`

   > **NOTE** Provide a secure application URL, otherwise the app registration will fail.

5. Select the **APIs** and the level of access your app needs:



6. Click the **Register App** button. **Power BI** will generate unique keys (**Client ID** and **Client Secret**) for your application.



7. Go to your app **web.config** file and provide the configuration for embedding the **Power BI**. You can use one of the following configurations to embed **Power BI** in Innovation Studio:

Configuration for embedding using master password:

```
<configSections>
...
<section
name
="powerBI"

type
="EBS.Core.Utils.PowerBiConfig.PowerBIConfigurationSection,
EBS.Core.Utils"/>
</configSections>
<powerBI>
    <tenants>
        <tenant name="default">
        <services>
            <service name="default"
            appId="{<span style="background-color:
#faebd7;">Azure application id</span>}"
            apiUrl="https://api.powerbi.com/"
            authorityUrl="https://login.microsoftonline.com/
common"
            resourceUrl="https://analysis.windows.net/powerb
i/api">
```

```
            <masterUser
                userName="{<span style="background-color:
#faebd7;">userName</span>}" password="{<span
style="background-color: #faebd7;">password</span>}">
            </masterUser>
            </service>
        </services>
        </tenant>
    </tenants>
</powerBI>
```

Configuration for embedding using service principal:

```
<configSections>
...
<section
name
="powerBI"

type
="EBS.Core.Utils.PowerBiConfig.PowerBIConfigurationSection,
EBS.Core.Utils"/>
</configSections>
    <powerBI>
        <tenants>
            <tenant name="default">
                ><services>
                    <service name="default"
                        appId="{<span style="background-
color: #faebd7;">Azure application id</span>}"
                        apiUrl="https://api.powerbi.com/"
                        authorityUrl="https://login.microsof
tonline.com/common"
                        resourceUrl="https://analysis.window
s.net/powerbi/api">

                        <servicePrincipal
                        applicationSecret="{<span
style="background-color: #faebd7;">secret</span>}"
                        tenant="{<span style="background-
color: #faebd7;">Azure tenant id</span>}">
                        </servicePrincipal>
                    </service>
                </services>
            </tenant>
```

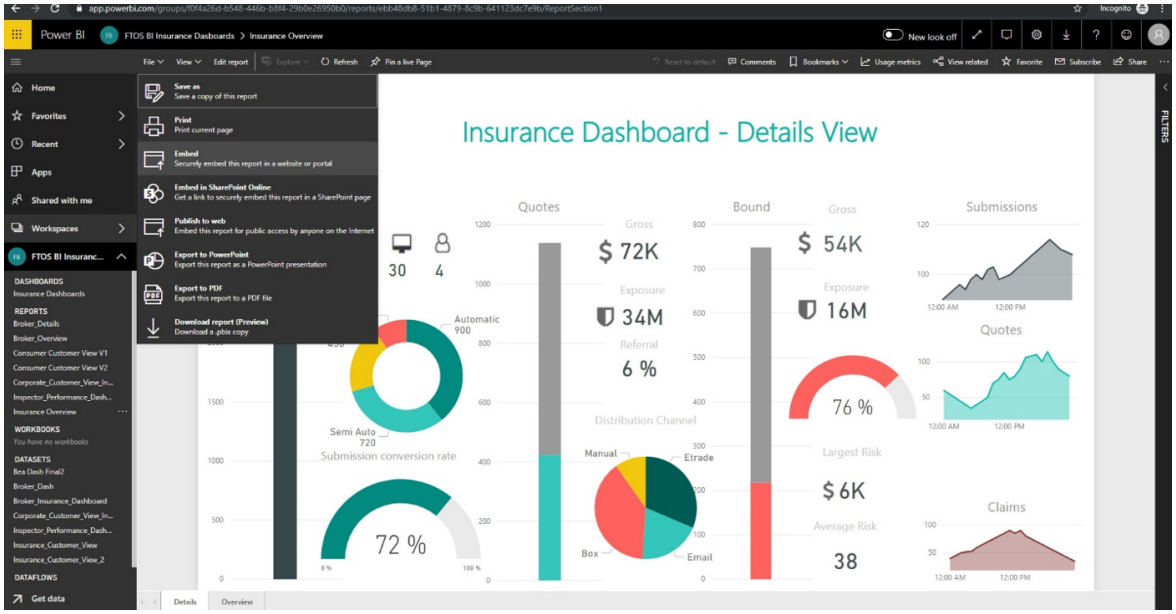
```
        </tenants>
</powerBI>
```

Embed Power BI Report

STEP 1. Get the Power BI report ID

Go to http://app.powerbi.com and log in using the credentials given by your organization. Go to the Power BI report and get the report ID.

This is how a power BI report might look like in the **Microsoft Power BI** app:



STEP 2. Embed the Power BI report in Innovation Studio

To embed and display a Power BI report in Innovation Studio, follow these steps:

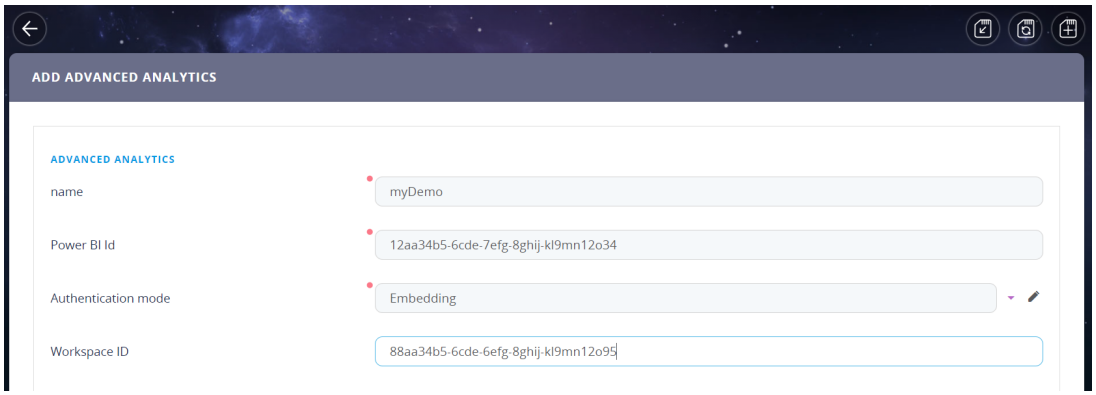
1. From the menu, click **Analytics** > **Advanced Analytics**. The **Advanced Analytics List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add Advanced Analytics** page appears.

3. In the Name field, type the name of the Power BI report (it will be displayed in the user interface).
4. In the Power BI Id field, type the ID given by Power BI to your report, that is the ID displayed in the Power BI URL (not the ID of the Power BI Dashboard).
5. Select the **Authentication mode**. Two options are available:

   Requires Sign-In (legacy) - user needs to authenticate with Azure AD in order to view the report.

   > **NOTE**Supported only in HTML Widgets of type Report, this option does not support passing any parameters.

   Embedding - an authorization token is generated by the server, and the user can view the report without authentication in **Azure AD**. This mode supports a richer API to interact with report parameters with markup components and the **Client SDK**.
6. In the **Workspace ID** field, type the unique identifier of the Power Bi workspace holding the report. This parameter is required when the authentication mode is **Embedding**.

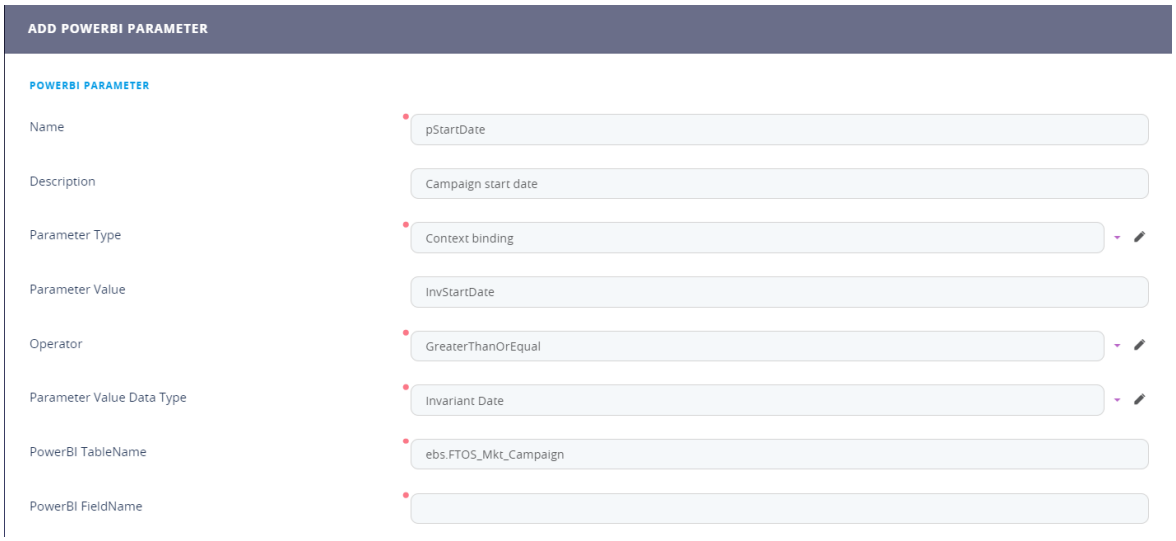


7. If you selected the authentication mode **Embedding**, at the top-right corner of the page click **Save and reload** to continue adding report parameters, following the steps explained in the next section. Otherwise, click the **Save and close** icon.

STEP 3. Add report parameters (Embedding authentication mode only)

Power BI report parameters define mappings to remote Azure Power BI report parameters. The table below describes the Power BI report parameters.

Parameter	Description
Name	The name of the parameter that will be used by the system to reference the parameter. The field is mandatory.
Description	The description of the parameter. Can be localized.
Parameter Type	Mandatory field for choosing the report parameter type. Two options are available: • **Context binding** - at runtime, the parameter will be bound automatically to a context property (the current entity property). If selected, in the **Parameter Value** field, provide the property name. • **Constant** - The parameter value attribute contains the actual value.
Parameter Value	Available only if the **Context binding** parameter type was chosen. This field holds the property name.
Operator	The comparison operator used to compose the filter expression. It is comprised of the parameter, and the value.
Parameter Value Data Type	The data type mapping for the parameter.
Power BI TableName	The name of the Power BI table containing the field to be mapped.
Power BI FieldName	The name of the Power BI field that will be mapped to the specified parameter.



After you finish providing the report parameter details, at the top-right corner of the page click the **Save and Close** icon to save the parameter. After you finish mapping to remote Azure Power Bi report parameters (adding all needed report parameters), at the top-right corner of the **Edit Advanced Analytics** page, click the **Save and Close** icon to save the Power BI report settings.

Now you can add and use the Power BI report in your digital journeys. For more information, see "How to add Power BI Reports to Digital Journeys" on page 450.

Add Power BI Report to Dashboard

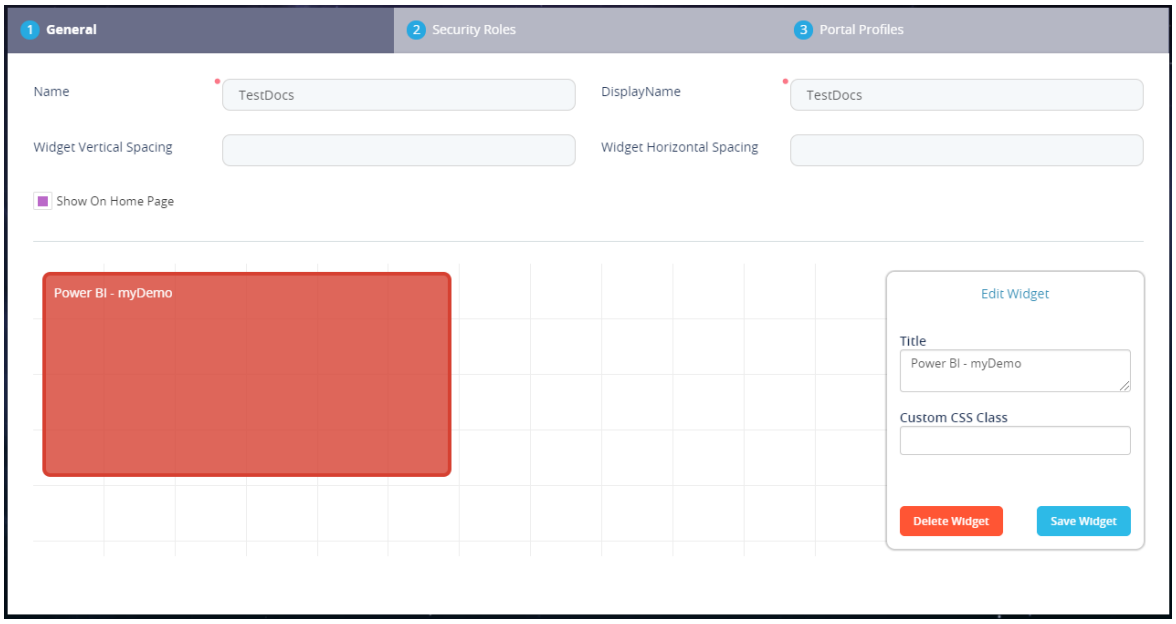
Prerequisites

- You should have created a dashboard.
- You should have registered **Power BI** for apps.
- You should have embedded the Power BI report into Innovation Studio

How to add a Power BI report to a dashboard

Follow these steps to add a Power BI report to a dashboard,:

1. From the menu, click **Digital Frontends** > **Digital Experience Portal** > **Dashboards**. The **Dashboards List** page appears.
2. Double-click on the desired dashboard. The dashboard configuration page appears.
3. In the **Add Widget** area, from the first drop-down, select **Power BI Report**.
4. From the second drop-down, select the Power BI report you want to add to the current dashboard. The name of the Power BI report appears in a rectangle below the toolbar.
5. Click the **Add** button. The report widget is added to the dashboard, and the **Add Widget** area is replaced by **Edit Widget**.
6. Optionally, you can resize the widget by placing the cursor on the bottom-right corner of the report rectangle. A resize icon will be displayed. Click to drag and drop, and resize as preferred.
7. At the top-right corner of the page, click the **Save and close** icon. The Power BI report is added to the dashboard.

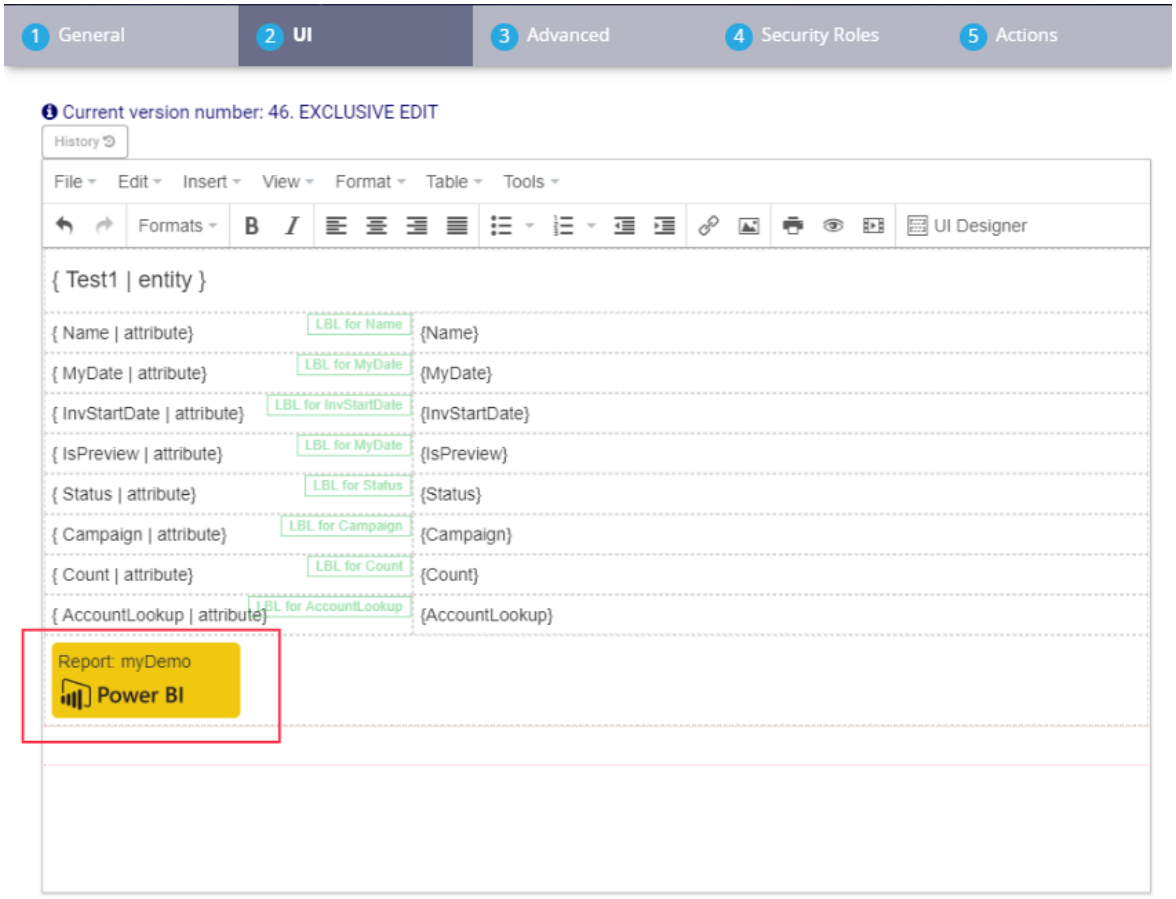


How to add Power BI Reports to Digital Journeys

You can define a Power BI report in HTML markup (on the digital journey, step, **UI** tab), by using the following code:

```
<os-powerbi-report id="{<span style="background-color:
#faebd7;">report ID</span>}" name="{<span style="background-color:
#faebd7;">report name</span>}" height="{<span style="background-
color: #faebd7;">display height</span>}">
</os-powerbi-report>
```

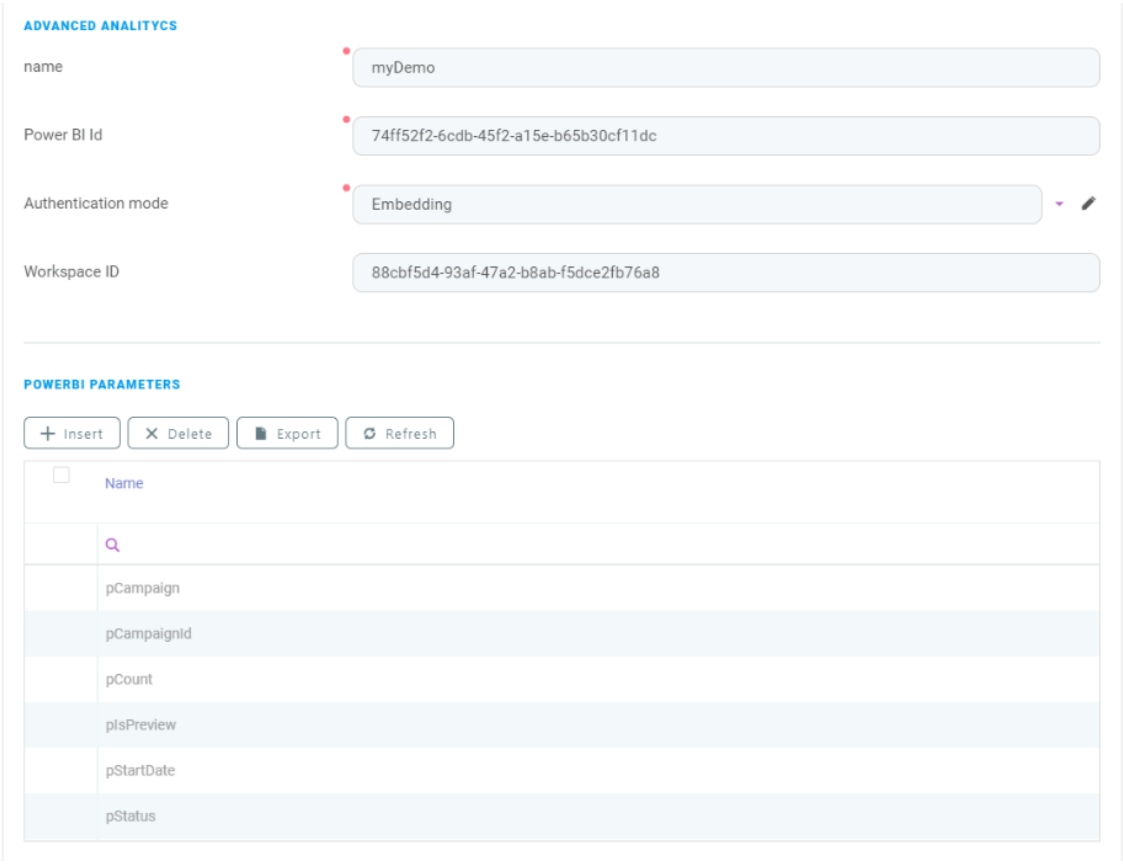
The **HTML** preview has a custom rendering for Power BI components by rendering the report name as title and showing a Power BI logo icon:



Supported attributes

The following are the supported attributes for **Power BI**:

- **name** - the name of the PowerBI record in FintechOS.
- **height** - the height in pixels or percent (default 100%) used for the report icon.
- **width id** - the width in pixels or percent (default 100%) used for the report icon.



Report parameters in HTML Markup

When no parameters are specified in HTML markup, report parameters are initialized based on the FintechOS Power BI parameter records in the database.

You can override the global configuration from the parameter record, by specifying the report parameters in the HTML markup. The following parameters are supported:

- **name** - the name of PowerBI parameter record, child of PowerBI report record from FintechOS.
- **context-binding** - specifies that the value will be bound to a context expression. The only expression implemented at this moment is a context property expression. The parameter will initialize from the context (current entity) property and will react to changes.
- **value** - specifies a constant value binding.

Example

In this example, the report parameter **pStatus** will be bound at runtime to the **Status** property of the current entity, and report parameter **pIsPreview** is initialized with the constant value true.

```
<os-powerbi-report
id="myReport" name="myDemo" height="500px">

     <os-powerbi-parameter name="pStatus" context-
binding="Status">
</os-powerbi-parameter>
<os-powerbi-parameter name="pIsPreview" value="true">
     </os-powerbi-parameter>
</os-powerbi-report>
```

Setting report parameters at runtime

You can set report parameters at runtime using the needed code in the **After Generate** field (on digital journey steps).

Example:

```
os.powerBIReport("myReport").setParameters({ pStatus : 'Expired'});
```

You can also set multiple parameters in a single API call:

```
os.powerBIReport("myReport").setParameters(
        {
```

```
            pStatus : 'Expired',
            pIsExpired : true
});
```

IMPORTANT!Setting the value of a context-bound parameter to a constant value will remove the binding, and the parameter will not react to further changes of the context property.

NOTEParameters of type **OptionSet** can be configured using the **optionsetitem id** or **optionsetitem name**. The value passed to the **PowerBI** service will be the name.

PowerBI client-side JavaScript API

Always specify an ID attribute for the components in HTML markup. The ID is used for component identification.

Example HTML:

```
<os-powerbi-report id="myReport" name="myDemo" height="500px">
        </os-powerbi-report>
```

Example JavaScript:

```
//make the report fullscreen
        os.powerBIReport("myReport").fullscreen();
        //populate report parameters with dynamic values
        os.powerBIReport("myReport").setParameters(
            {
                pStatus : 'Expired',
                pIsExpired : true
        });
```

Custom Reports

Many financial organizations use **SQL Server Reporting Services (SSRS)** to generate custom reports that comply with financial regulations, and are used for statutory reporting to legal authorities.

FintechOS integrates with SSRS successfully, and addresses to a wide variety of reporting needs including managed enterprise reporting, ad-hoc reporting, embedded reporting, and web based reporting to enable organizations to deliver relevant information, wherever it's needed.

FintechOS enables you to make these reports available directly within the platform. They can be generated on demand based on the selected time frame and the inclusion/exclusion criteria. The reports can be accessed based on previously defined security roles, or automatically generated on predefined milestones and dates.

You can export the reports to multiple file formats, or deliver them to subscribers by e-mail or to a shared file. Reports can also be generated from the application menu.

Creating a custom report

Prerequisites:

- A SSRS user account with credentials at hand (username and password).
- A report in SSRS.

STEP 1. Add a report

1. From the main menu, click **Analytics** > **Reports**. The **Reports List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add Report** page appears.
3. Fill in the fields.

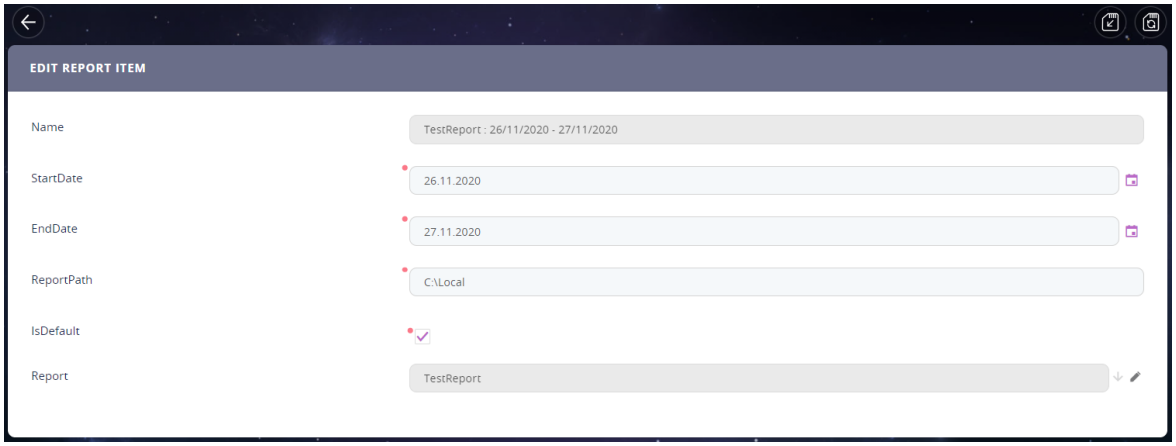
Field	Description
Name	Enter the report name that will be used by the system.
Display Name	The name of the report that will be displayed in the **Portal**. This field is mandatory.
Entity Menu Section	Select the entity menu section where users will be able to generate the report.
Show In Menu	Select the check box only if you selected the entity menu section where users will be able to generate the report.
Scope	Select **General or Entity.**
Type	Select **Document, Custom Report** or **Simple Grid Report. Document** is available only for the scope **Entity. Simple Grid Report** is available only for scope **General.**
For scope **General** or **Entity,** and type **Custom Report**	
Server Url	Enter the URL of the **SQL Server Reporting Services (SSRS).**
Domain	Enter the domain name where you host the **SSRS**.
Username	Enter the username associated to your **SSRS** user account.
Password	Enter the password of your **SSRS** user account.
Always Return File	Tick the checkbox to return the file. Only available for scope **Entity.**
For scope **Entity** and type **Document** or **Custom Report**	
Output Method	Select the output method: **Attach to entity** or **Download file**.
Destination Field	Set the destination field for your report. Only available for **Attach to entity** output method.
Destination File Name	Add the destination file name.
Report Document Type	Only available for type **Document**.

4. At the top-tight corner of the page, click the **Save and reload** icon. The **Edit Report** page appears.

IMPORTANT!You have to add a report item; otherwise the report cannot be generated.

STEP 2. Add report items

This section is used for setting the time interval from which the report will select the data. Each insertion of a report item is sort of a template for that report. Hence, it is possible to add more templates the same report with different dates.



1. In the **Edit Report** page, scroll-down to the **Report Items** section. At the top of the section, click the **Insert** button. The **Add Report Item** page will be displayed.
2. The **Name** field is automatically filled with the name of the report inserted in Step 1.3.
3. Select **Start Date** and **End Date**. Upon the report generation, it will gather data within the specified time interval (between the start date and the end date).
4. In the **Report Path** set the path for where this report will be stored.
5. For the bool **IsDefault**, tick if you wish this template to be the default for the report. If not, leave empty.
6. The **Report** field is automatically filled in with the name of the report.
7. At the top-right corner of the page click the **Save and reload** icon if you want to add another report item, otherwise, click the **Save and close** icon.

STEP 3. Insert Report Parameters

This section allows setting one or more attributes for the report. This is available only if the scope of the report was set to **Entity**. Fill in the following fields:

Field	Data type	Description
Name	Text	Insert a name for the parameter.
Attribute	Option set	Select an attribute for the list.
Report	Text	It is automatically filled in with the name of the report inserted at Step 1.3.

Click the **Save and close** button. Repeat as many times as needed.

STEP 4. Define who has access to the custom report

If your business case requires that the custom report is available to designated roles within your organization, in the **Edit Report** page, scroll-down to the **Report Security Roles** section. Click the **Insert existing** button, and select the security roles that should have access to report. If no security roles are added here, all users will be able to view the report. For details, see "Creating Security Roles" on page 622.

STEP 5. Save the report

If you want to save and close the report, at the top-right corner of the page click the **Save and close** icon.

If you want to save the report and continue working on it, click the **Save and reload** icon.

Tabular Reports

Tabular reports offer a tabular view of data; each column representing a field and each row representing a record.

FintechOS gives you the possibility to create tabular reports based on data returned by your own stored SQL procedures or by fetching data.

To create a tabular report, follow these steps:

STEP 1. Add Data Source and Parameters **458**

STEP 2. Add Report Parameters 460
STEP 3. Add Simple Grid Report 461
STEP 4. Add Report Items 462
STEP 5. Define Report Access Privileges 463

STEP 1. Add Data Source and Parameters

1. From the Main Menu, click **Analytics** > **Data Sources**. The **Data Sources List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add Data Source** page appears.
3. Enter the **Name** of the data source that will be used by the system.
4. Enter the **Display Name** of the data source that will be displayed in the UI.
5. If your data source is a stored SQL procedure, tick the **Use Stored Procedure** checkbox, and in the **Stored Procedure** field, enter the name of the SQL procedure following this convention: `procedure_name_as_stored_in_DB @Id @EntityName @UserId`. For more information, see "Using Stored Procedures" below.
6. If your data source is fetch data, tick clear the **Use Stored Procedure** checkbox and fetch the data. For information on how to fetch data, see "Using Fetch Data" on the next page.

Using Stored Procedures

Prerequisite: Create the SQL procedure that you want to use.

The following parameters of the stored SQL procedure are automatically mapped to specific values, as described in the table below:

Parameter	Value mapped to
@Id	The record ID of the entity item that has the **Report** linked to the **Report Document**.

Parameter	Value mapped to
@EntityName	The name of the entity that has the **Report** linked to the **Report Document.**
@UserId	The ID of the user that runs the **Report** linked to the **Report Document**.

If you want to use other parameters declared within the stored SQL procedure, you have to save and reload the **Add Data Source** page, and add the parameters in the **PARAMS** section first, then append them in the **Stored Procedure** field using the following convention `@parameter_name`. For information on how to add parameters, see Add Report Parameters.

Using Fetch Data

Prerequisites:

- Make sure there are at least two entities in the system. For information on how to add entities, see Creating Entities.
- Create relationships between the entities on which you do the fetch. For more information on relationships, see Entity Relationships.
- Add custom attributes to each entity for which you do the fetch; you will use the attributes when defining the fetch. For information on how to add attributes, see Adding Attributes.

To fetch the data, you can write the fetch directly into the **Fetch Object Expression** field. You can also use the **Fetch Designer** by clicking the **Show Fetch Designer** button, and choosing the criteria and conditions for clustering the database and choosing the data to be included in the report.

Basic fetch:

```
return {
    "entity": {
        "alias": "base",
        "name": "entity",
        "attributelist": null
    }
}
```

Change the values that you want to be dynamically replaced by the parameter values by using `getParamValue`.

Dynamically change property values with the parameter values when generating the report.

In the example above, we will change the value of the **name** property as follows:

```
return {
    "entity": {
            "alias": "base",
            "name": "entity",
            "attributelist":null
    },
    where:{
            type:"and",
            conditionlist:[{
                first:"base.defaultEntityStatusId",
                type:"equals",
                second:"val(<span style="background-color:
#fff0f0">getParamValue(entityStatus</span>))"
            }]
    }
}
```

When generating the report, the system will use the value of the `entityStatus` parameter for the **name** property.

The platform supports multiple parameters at once, therefore you can use `getParamValue` to dynamically change properties value with the parameters value as many times as you need.

NOTEYou have to save and reload the **Add Data Source** page and in the PARAMS section, add the parameters whose values will replace the values of properties as defined in the fetch. For information on how to add parameters, see Add Report Parameters.

STEP 2. Add Report Parameters

Report parameters are used for inputting data into document reports, used to filter the data when generating the report.

Adding parameters is very easy, similar to adding an attribute on an entity:

1. In the **Edit Data Source** page, scroll down to the **PARAMS** section and click the **Insert** button. The **Add Data Source Param** page appears.

2. Provide the **Name** of the parameter matching the name that you will use in the custom fetch, or in the stored procedure.

   When generating the report, the value of the "name" property will to be replaced with the value of the "entityName" parameter provided in the fetch, we will add the "entityName" parameter.

3. In the **Display Name** field, enter the name of the parameter as it will be displayed in the UI.
4. Select the **Attribute** type. For more information on the types of attributes available in the platform, see Types of Attributes.
5. If you want to add multiple parameters, click **Save and reload** and add the parameters. Otherwise, click **Save and close**.

STEP 3. Add Simple Grid Report

1. From the menu, click **Analytics** > **Reports**. The **Reports List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add Report** page appears.
3. Fill in the fields, as follows:

Field	Description
Name	Enter the report name which will be used by the system.
Display Name	The name of the report which will be displayed in the **Portal**. This field is mandatory.
Entity Menu Section	Select the entity menu section from where users will be able to generate the report in the **Portal**.

Field	Description
Show In Menu	Select the checkbox only if you selected the entity menu section from where users will be able to generate the report.
Scope	Select **General**.
Type	Select **Tabular Report**.

4. At the top-tight corner of the page, click the **Save and reload** icon. The **Edit Report** page appears.

IMPORTANT!You have to add a report item, otherwise the report cannot be generated.

STEP 4. Add Report Items

A report item represents a configuration for the report that will be added to the report, and will gather data within the specified dates.

You can have many report items, but only the one set as default will be used upon the report generation.

To add items to a simple grid report, follow these steps:

1. In the **Edit Report** page, scroll down to the **Report Items** section and on top of the section, click the **Insert** button. The **Add Report Item** page appears.
2. Select the **Start Date** and **End Date**. Upon the report generation, it will gather data within the specified time interval (between the start date and the end date).
3. Select the **Data Source** for the report.
4. Select the **Is Default** checkbox if this is the item that you want to be used when generating the report.

5. At the top right corner of the page click the **Save and reload** icon if you want to add another report item, otherwise, click the **Save and close** icon.

If needed, you can restrict users' access to the report by adding security roles to the report. For more information on security roles, see STEP 5. Define Report Access Privileges.

STEP 5. Define Report Access Privileges

If your business case requires that the simple grid report is available to designated roles within your organization, in the **Edit Report** page, scroll down to the **Report Security Roles** section, click the **Insert existing** button, and select the security roles that should have access to report. If no security roles are added here, all users will be able to view the report.

Once you finish adding the security roles, click the **Save and close** icon to save the report. For more details on security roles, see "Security Roles" on page 621.

Charts

Charts are visual presentation of data that help you convey information, and understand data in a visual way.

By using charts, data in your report is displayed in a clear way allowing you to easily compare sales figures or highlighting a trend.

Use charts when a tabular report won't adequately show relationships between data points.

Creating charts

Prior to creating charts, think about the outcome that you want to achieve, or the specific data that you want to show. Make your charts simple to keep your audience focused on relevant information and avoid any confusion.

Follow the steps below to add a chart:

1. From the main menu, click **Analytics** > **Charts**. The **Charts List** page appears.
2. At the top-right corner if the page, click the **Insert** icon. The **Add Chart** page appears.
3. Configure the chart by providing the attributes:

Chart Attribute	Description
Name	The chart name used by the system.
Chart Title	The chart name that will be displayed in the Digital Experience Portal.
Chart Base Type	Two options are available: • **Standard** - represents a Name/Value series on the X/Y axes. For example, income= 20k, education=high school. • **Series** - represents more series from a given fetch. For example, client Paul Mathew: income=15k, education=undergraduate. When you have a third variable it is suitable to use series.

Chart Attribute	Description
Chart Type	Select one of the following options from the drop-down: • **bar** - displays rectangular bars. Use bar charts to present or compare data from the same category, e.g., the sales value or product volume over a period of time. • **line** - displays a two-dimensional scatter-plot of ordered values connected by lines following their order. Use lines charts to display the relationship between multiple sets of data over a period of time. • **Pie** - displays percentage values as a slice of a pie. Use pie charts to focus on the big picture, drawing attention to important information. • **doughnut** - similar to pie charts, this type of charts have a number of elements including the division of segments and the meaning of arc for an individual segment. It is useful to present the relationship between proportions of different data groups.

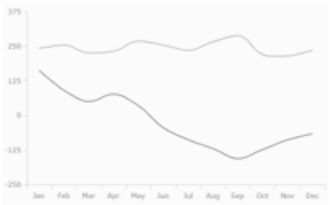
Chart Attribute	Description
	 • **spline** - similar to line charts, the spline chart has data points connected with smooth curves instead of straight lines. It usually display the change of a variable over time (in Cartesian charts) or simply along the X-axis (for Scatter charts). Spline charts are the basis for spline area charts.  • **scatter** - allow analyzing how different goals have been achieved around a main topic and their different dimensions over a time period. For example, compare types of products based on budgets and selling prices. • **area** - similar to line charts, they have solid plot lines. They're useful for displaying trends over a period for single or several categories, or the change between several data groups.

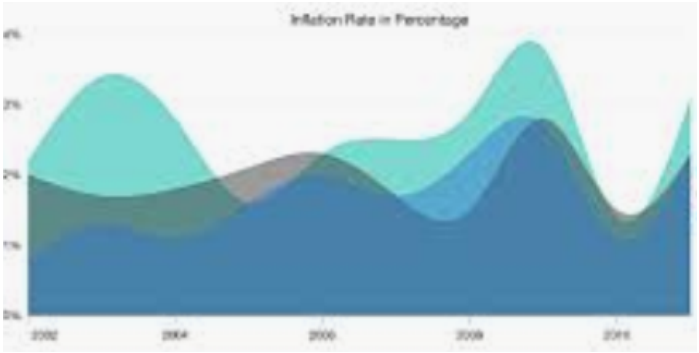
Chart Attribute	Description
	• **spine area** - an area chart in which data points are connected by smooth curves. The area between the line segments and the X-axis is colored to emphasize the magnitude of change over time.  **NOTE**For base charts of type **series**, the following chart types are not available: **pie** and **doughnut**.
Name field	Alias used in the fetch which maps the name on X axis.
Value field	Alias used in the fetch which maps the value on the Y axis.
Series argument field	The pairing of an argument and its value are represented on a diagram's axes as their X and Y coordinates. This field holds the third. For example, this would be the client variable.
Show legend	Tick to display data about the datasets that appear on the chart. You can choose where you want the legend to be shown on the chart: **Legend horizontal alignment** (left, right or center) and **Legend vertical alignment** (top or bottom).
Show labels	Tick to add data labels to the data points of the chart. Labels help you quickly identify data series in a chart.

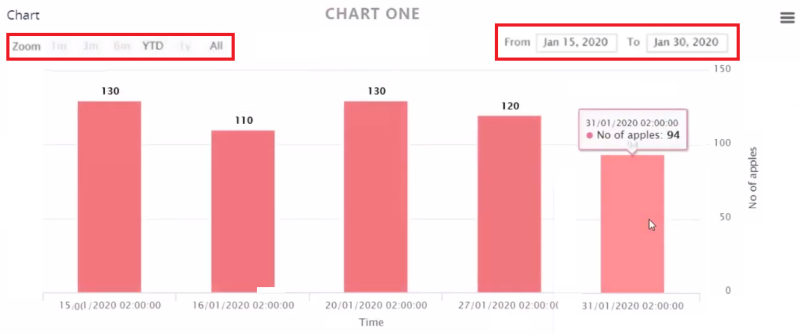
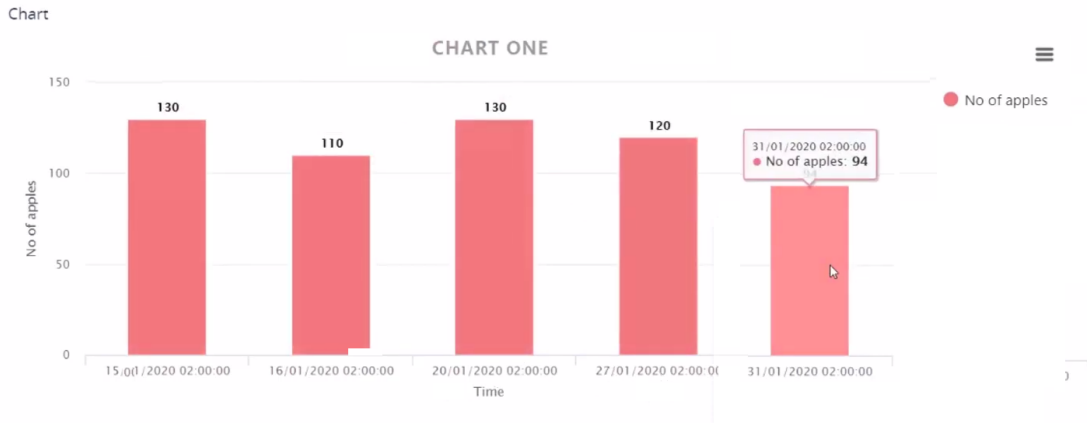
Chart Attribute	Description
Override Color	Change the default rendering color of the chart.
Axis X Title	The title to be displayed on the chart's X axis.
Axis Y Title	The title to be displayed on the chart's Y axis.
Render Type	The chart rendering type: **Chart** or **Finchart**.
Render As Time Series	Tick to render chart values over a period of time. It enables you to zoom in/out chart data per time intervals (1 month, 3 months, 6 months, year, or all) and also filter the data within specific from - to dates. The figure below presents a standard bar **Finchart** rendered as time series: 
Container CSS Class	The name of the css class to add to the graphic container.
Container CSS Inline Style	The css changes you want to apply to the chart.
After Generate Js	JavaScript code to be executed after the chart is rendered.

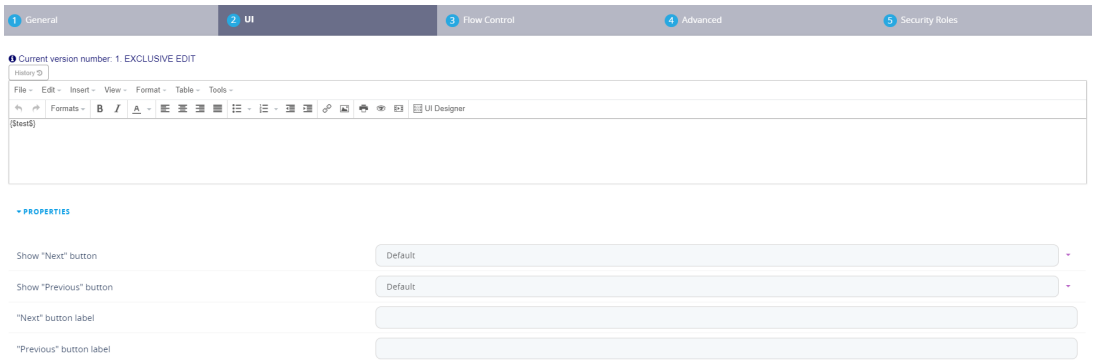
Chart Attribute	Description
Fetch expression return object	Returns the fetch data for the chart.

4. Save the changes by clicking the **Save and close** icon. If you want to see how the chart looks like, click the **Save and Reload** icon and scroll down to the Chart section which displays the chart. The figure below shows an example of a standard bar Finchart which is not rendered as time series:

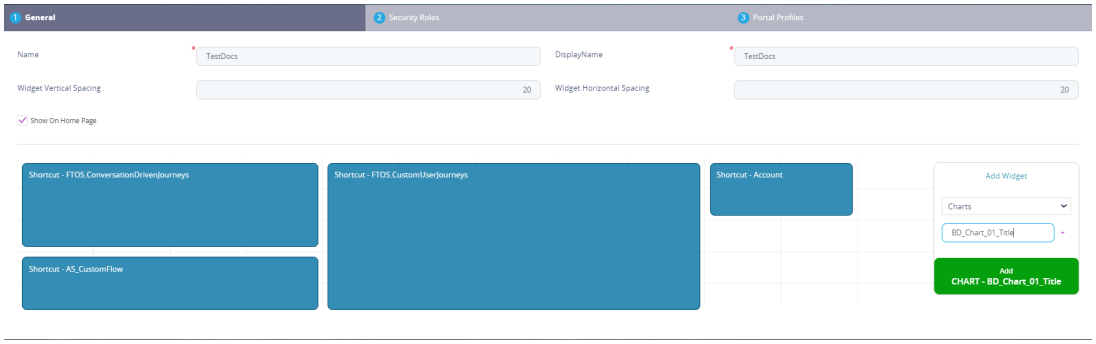


There are various ways in which you can render charts:

- on forms by using the token **{$chartName$}**. By creating a form driven flow (see Form Driven Flows), and in the **Steps** tab, create the step you wish to insert the chart in e.g. Step 4 and in the **UI tab** insert the token with the name of the chart created. Click the **Save and close** button. Display the shortcut for the Digital Journey in a dashboard and navigate to Step 4 to see the chart.



- add charts to **Digital Experience Portal** dashboards. Go to the desired dashboard and select the **charts widget** select the name of the chart and click on ''Add xyz'' chart as shown below.



- programmatic by using the **ebs.generateChart** method. For details, see ebs.generateChart in the Client SDK user guide.

Digital Developer Tools

Innovation Studio provides developers with extensive capabilities:

- manage HTML, CSS, and JavaScript attributes at source code level
- create scripts and use scripts in digital journeys and forms
- create script libraries to avoid writing the same lines of codes repeatedly, resulting in increased productivity.

NOTE
Some advanced features are not available to consultant users.

This section covers the following topics:

DB Tasks **472**
Step 1. Add DB tasks 473
Step 2. Add security roles to DB tasks 473
Step 3. Execute DB Tasks 474
Advanced Code Editor **475**
Features 476
How to Access the Advanced Code Editor 476
General Layout 477
How to use the Advanced Code Editor 479
Debugging files from the editor 480
Automation Scripts **480**
Event Triggered Automation Scripts 481
On-demand automation scripts 483
Setting the execution order of automation scripts 483
Using Automation Script Libraries 484
Using Web API Client Libraries 487

Creating Event Triggered Automation Scripts 492
Creating On-demand Server Automation Scripts 496
Creating Endpoints 504
Calling Actions 506
Scheduling Server Automation Scripts 508
Using Plugin Assemblies 511
XML Support 512
Debugging Automation Scripts 516
Code Blocks 519
1 Add categories 519
2 Add code blocks 520
3 Use code blocks 522
Custom Client-side Functions 523
Defining Custom Functions (using Client Script Libraries) 526
Fluent Queries 527
How to Execute a Fluent Query 528
Working with Fluent Query Result Sets 535
Sequencers 537
How to add items to the sequencer 538
How to call the Sequencer 539
Entity versioning 539
How to Set Up Versioning on an Entity 541
How to Version an Entity Record 548
Email Templates 549

DB Tasks

A stored procedure is a set of Structured Query Language (SQL) statements which are saved in the database (DB).

If the stored procedures are performing massive computing, (e.g. automated calculation of data from imported tabular files), for high-performance computing, we recommend you to use DB tasks to execute such stored procedures under the 'Ebs' DB schema.

Step 1. Add DB tasks

To add a DB task:

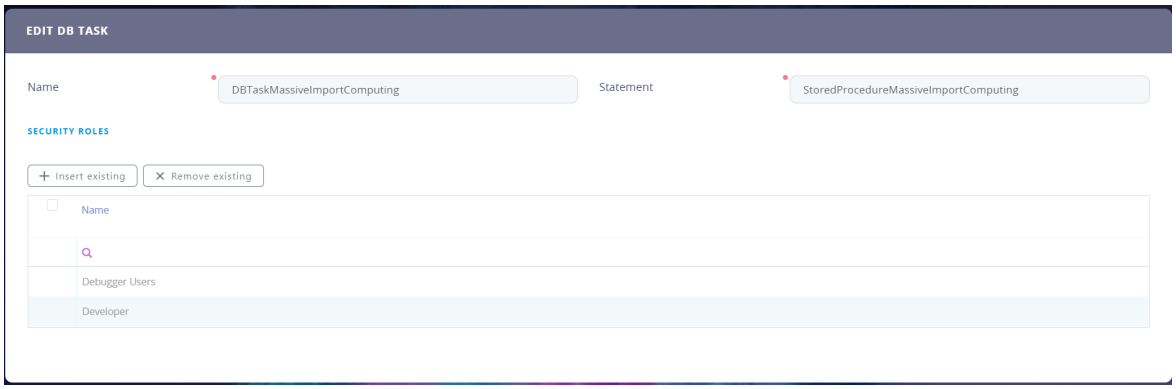
1. Log in the Innovation Studio using developer mode.
2. Click the Main Menu, then click **Advanced** > **DB Tasks**. The **DB Tasks List** page appears.
3. At the top-right corner of the page, click the **Insert** icon. The **Add DB Task** page appears.
4. Type a **Name** for the DB task. It is the name used by the system. You need it when executing the DB task.
5. In the **Statement** field, type the name of the stored SQL procedure.

If you want to add security roles to DB tasks, at the top right corner of the page, click the Save and reload icon. and go to STEP 2; otherwise, click the **Save and close** icon.

Step 2. Add security roles to DB tasks

If your business case requires that the DB tasks are executed only by designated roles within your organization, in the Edit DB Task page, click the Insert existing button at the top of the **SECURITY ROLES** section. A pop-up listing all defined security roles appears.

Double-click the desired security role from the list, The pup-up closes and the selected security role is displayed in the SECURITY ROLES section.



Add the security roles that comply with your organization's security policies.

NOTE If no security roles are added, all users will be able to execute the DB task.

After you finish adding the security roles, at the top-right corner of the page, click the **Save and close** icon to save the DB task updates.

Step 3. Execute DB Tasks

Users who have a security role defined on a DB task can execute that DB task in server automation scripts by using the following Server SDK function:

Syntax:

```
function executeDbTask(dbTaskName: string, parameters: any):
IExecuteDbTaskResult
```

Request Parameters

Parameter	Description
dbTaskName	The name of the DB task to be executed.

Parameter	Description
parameters	The array of parameters and their values specified in the stored procedure. It is a json object with the following format: `parameterObject = [ { paramName: "paramName", paramType: "ParamType", paramValue: "paramValue"} .... ]`

Returns

The function returns an array of objects mapped to the columns from **select** in the stored procedure.

NOTE If the DB Task has no security roles associated, all users can execute the DB task.

Advanced Code Editor

The Advanced Code Editor provides developers with a simple and powerful interface that allows them to insert and edit HTML, CSS and JavaScript attributes by using code.

The table below lists the attributes that can be inserted or edited using the Advanced Code Editor.

Entity	Attribute
EntityForm	After Events
EntityForm	Before Events
EntityForm	Template
Entity Form Field	Attribute Change Event
Entity Form Section	After Events
Entity Form Section	After Section Save
Entity Form Section	Before Section Save
Entity Form Section	Template

Entity	Attribute
EntityView	Fetch Object Expression
EntityView	After Generate Js
EntityView	Display Options
Automation Script	Code
Automation Script Libraries	Code
Client Script Library	Definition
Client Script Library	Code
Custom Form (Custom Flow)	After Generate Js
Custom Form (Custom Flow)	Template
Html Widget	JavaScript
Html Widget	Html
Style Sheet	Code

Features

- Browsing files and nodes
- Searching for specific nodes or specific content in files
- Simplified code editing using code snippets
- Live preview of HTML files
- Debugging right from the editor
- Insert code blocks and customize them as best suit your needs

How to Access the Advanced Code Editor

You can access the Advanced Code Editor in two ways:

- From the menu, click Advanced > Advanced Code Editor. By doing so, you will see all files and nodes available in Innovation Studio.

- When editing one of the entities listed in the table above, click at the top-right corner of the configuration page the Open in Code icon. By doing so, you will be able to browse the files of the current entity.

General Layout

The Advanced Code Editor has a common user interface which is comprised of the following panels: an explorer on the left, showing all of the folders and files you have access to, main editor in the center, showing the content of the files you have opened, a property list on the right, showing the values of the attributes from the file you have opened, a toolbar on top and two search tabs at the bottom.

Files Explorer

Displays in a tree view the files you have access to. You can browse and select the attribute files you want to edit. The files are organized in folders.

Toolbar

The toolbar displayed on top provides the controls to perform basic operations like saving or closing file(s) and also debugging the code directly in the editor or live previewing the HTML files. The table below describes the controls available.

Control	Description
Check file history	Displays the version history of the currently opened file (if any).
Preview Mode	Available only for HTML files, allows you to toggle between the source code or the live preview of the HTML file. When choosing live preview, below the HTML source code, a panel will be displayed within the main editor showing how the HTML file will look like in the UI.
Close all open tabs	Closes all opened files. When you open multiple files, they are displayed as tabs.
Save all	Saves all opened files. You can also save all opened files by pressing **CTRL+SHIFT+S**.
Save	Saves the file you're currently working on. You can also save the file you're working on by pressing **CTRL+S.**

Search Nodes

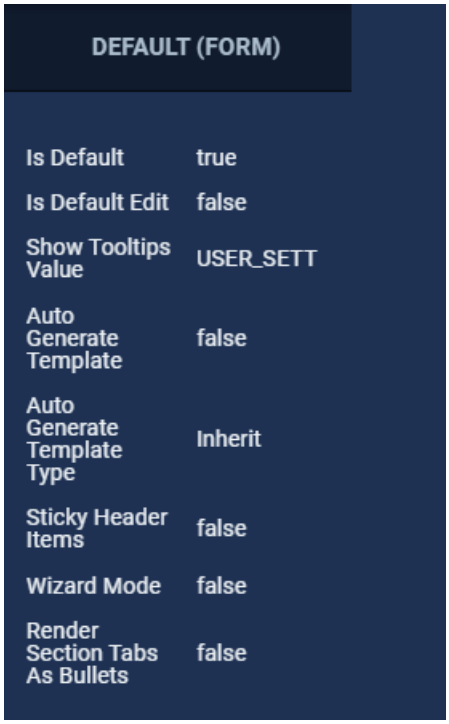
The tab allows you to search available nodes by name. Enter the name of the node you want to search for and press **Enter**. The search returns the list of nodes partially matching the name of the node you provided.

Search in Files

The tab allows you to search for specific content within the available files. Enter the content to search for (e.g., an attribute, function) and press **Enter**. The search returns the list of nodes which contain the content you provided.

Properties List

For some of the nodes a list of properties is displayed, on node selection, on the right-side panel.



Previewing HTML Files

To preview an open HTML file, from the toolbar toggle the **Preview Mode** button on.

Below the preview panel, a toolbar with three colored icons is available. The icons allow you to perform specific actions, as follows:

- Green icon - maximize the live preview to the main editor panel.

  To maximize the preview panel even ore (by hiding the search tabs), click the down-arrow.

- Orange icon - minimize the live preview.

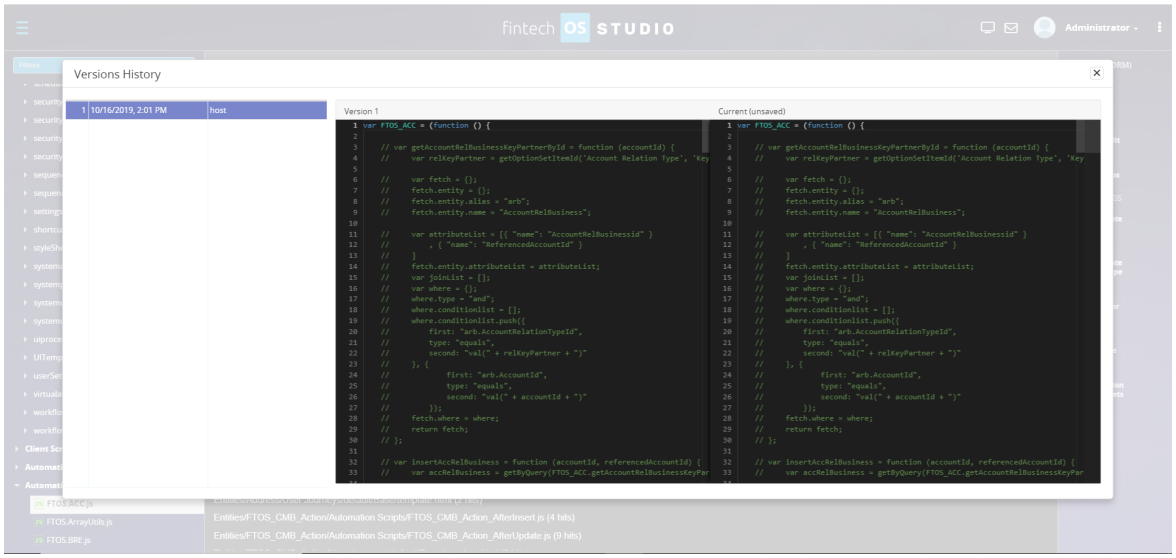
- Red icon – close the live preview. The Preview Mode button is automatically toggled off.

How to use the Advanced Code Editor

Browse or search for the files you want to edit and open them by clicking on the files. Provide the code which defines how and what the files should do and save the changes. You can also insert predefined code blocks and change them to best suit your needs.

To write the code faster and typo-free, use the built-in Intellisense and code snippets. For information on how to use code snippets, see Code Snippets Support.

Debugging files from the editor



You can debug a file directly in the editor. To do so, open the file and on the toolbar, click the Check file history button ( ). The panel on the left shows file versions in descending order, who made the change, and the date and time of the change. The panel on the right shows a side-by-side comparison of current and previous versions. Changes performed in current version are highlighted.

Automation Scripts

FintechOS allows you to manage complex automation and validation tasks, triggering the execution of scripts on business status transition.

Automation scripts are executed synchronously, which means that the execution happens in a single series. Next operations cannot be performed until the current operations is finished.

Key points of synchronous automation scripts:

- Are created by using an automation script entity record.
- The event-triggered automation scripts can execute before (pre-operation), after (post-operation), or after the transaction is completed.
- Whether configured to run on-demand or event triggered by specific operations (read, update, insert, delete), the script runs immediately.
- Log errors only when logging is enabled.
- Execute in the current transaction.

Reusable blocks of code can be included in automation script libraries and associated afterwards to scripts.

In Innovation Studio, you can create and use two types of automation scripts, as follows:

Event Triggered Automation Scripts

These automation scripts are automatically triggered when an event of read, update, insert or delete occurs in the user interface:

Event	Description
Read	The most complex type of automation scripts. Upon its execution, all information from the target entity is read, including automation scripts or `getByQuery`methods. You should be careful not to alter other functionality related to the entity.

Event	Description
Update	If an automation script updates another entity that has another automation script on update, that will also be triggered, and if any automation script in the execution chain throws exception, the entire transaction is roll-backed. That means that every operation is enlisted to the master Ebs Core transaction. The "After Transaction" stage executes a script only after the transaction completes. It should be used only in rare cases, if you make a get/post call to another web-service which tries to update/alter data of the same record. Inside every automation script code you have a context variable that holds data about the current scope. ⚠ **IMPORTANT!** If you write an update script inside which you update the same record, it will trigger in recursion. Although the EBS has been build to prevent such situations, the transaction will be rolled back after several iterations.
Insert	If you do not have a record 'before insert', use 'after insert' instead.
Delete	Can be used in conjunction with: • The 'before' stage, otherwise, the automation script tries deleting a record ID which has already been deleted. • The 'after insert' script. Do not try to use it 'before insert' as you do not have a record ID yet to relate to.

Event-triggered automation scripts are bound to a ‘before’, ‘after’ or ‘after transaction’ stage execution, while on-demand automation scripts have no such dependencies and can be executed whenever the case.

A list of predefined methods and functions with corresponding code-snippets are available within a dedicated development library (automation script library), covering most common use cases and technical applicability scenarios.

On-demand automation scripts

While event-triggered automation scripts are context-based and linked to a specific entity within the open data model, on-demand automation scripts are context independent and available for being called from any object or context.

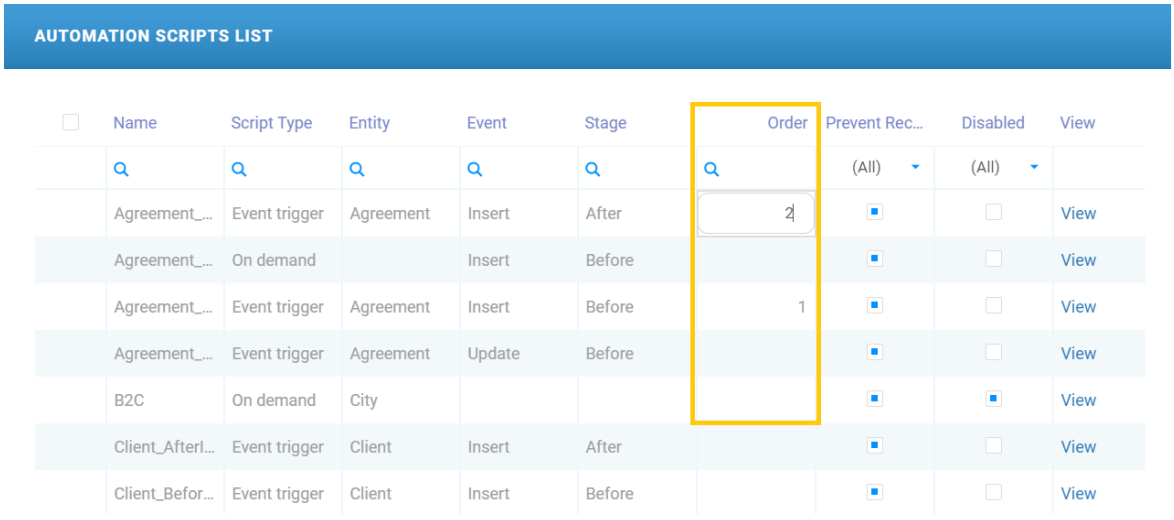
On-demand automation scripts can be triggered manually if they are attached to an action. For usability purposes, you can organize actions performed on an entity into action groups.

To use an on-demand automation script, follow these steps:

1. Create an on-demand automation script.
2. Create endpoint.
3. Call action.

Setting the execution order of automation scripts

You can set the execution order for similar automation scripts (entity / event type / stage) from the Server Automation Scripts List page (Advanced > Server Automation Scripts) by setting the execution order in the **Order** column.



Using Automation Script Libraries

Use automation script libraries to organize code on the server and reuse the code in automation scripts by attaching the automation script library to the desired automation scripts.

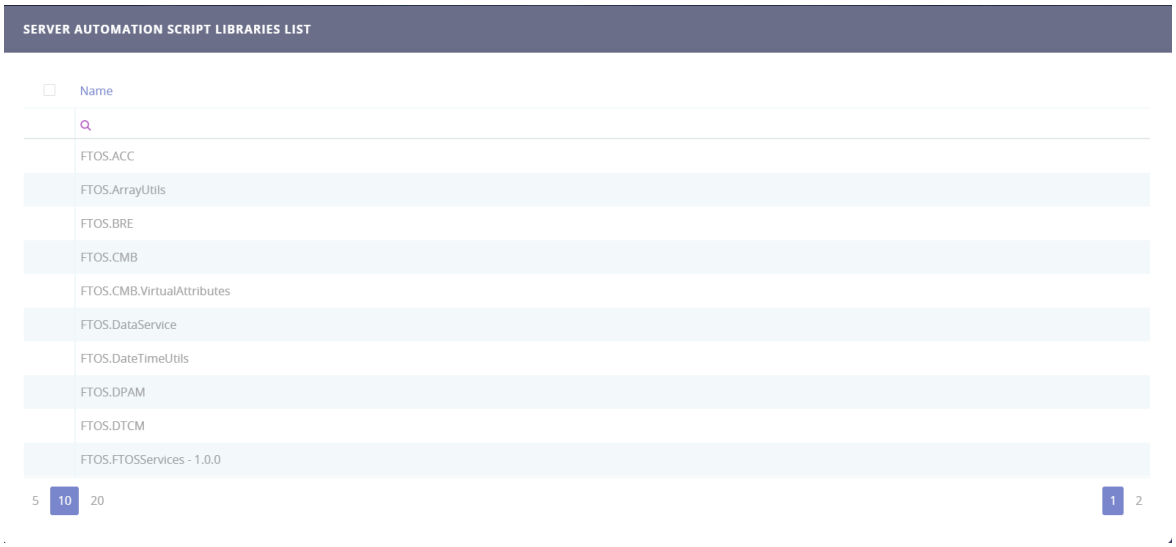
NOTEYou can attach multiple automation script libraries to automation scripts but you cannot attach automation script libraries to other automation script libraries.

You will be able to easily maintain the code by modifying it once on the server (within the automation script library) and the code updates will be automatically distributed to all the automation scripts calling the automation script library.

The typical use case scenarios in which an automation script library is used:

1. Executing a specific block of code after an entity business status is changed. To use an automation script library to execute a block of code after a business status is changed, you need to create it first. For information on how to create an automation script library, see Create an Automation Script Library.
2. Call the automation script library from a server script using server-side functions.

To see the list of defined automation script libraries, from the menu, click Advanced > Server Automation Script Libraries. The Server Automation Script Libraries List page appears:



Particular automation script libraries

The code provided in automation script libraries with a name which follows the following naming convention **statusChange_EntityName** are executed automatically on entity stage transition. These libraries exist in the context of previous and current execution.

```
var currStatusName = getBusinessStatusName("EntityName",
context.BusinessStatus);
var previousStatusName = getBusinessStatusName("EntityName ",
context.PreviousBusinessStatus);
```

To view the state transition, you should use after update server scripts using the following commands:

```
var ent = getById("EntityName",context.Id);

svar prevBussinessStatusId = context.BeforeValues.businessStatusId;
var bussinessStatusId = ent. businessStatusId;
if (businessStatusId != prevBussinessStatusId) {
        // state transition occurred from prevBussinessStatusId to
businessStatusId
}
```

Creating Automation Script Libraries

To create an automation script library, follow these steps:

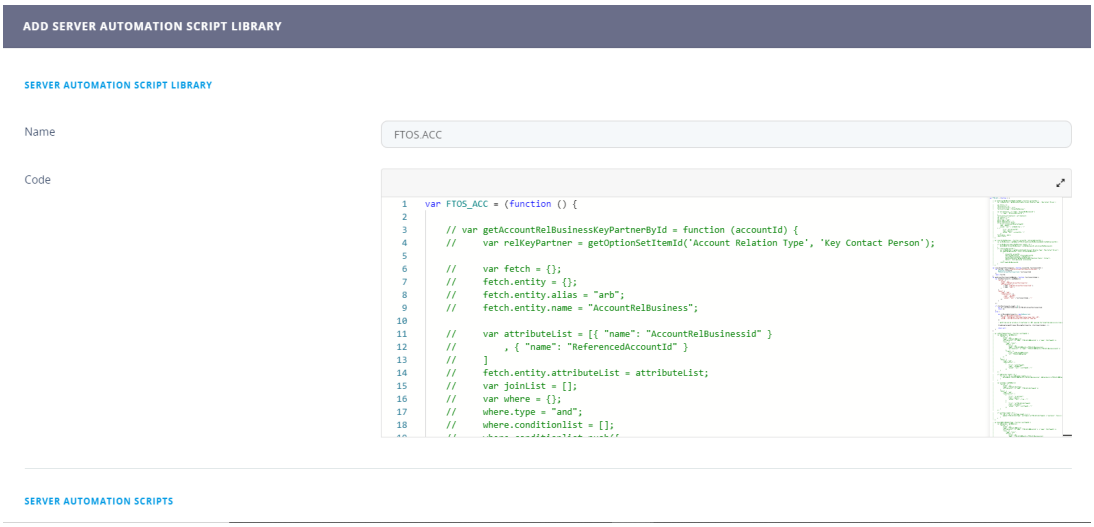
1. At the top-right corner of the Server Automation Script Libraries List. page, click the **Insert** icon. The Add Server Automation Script Library page appears.
2. In the Name field, enter the name of the library.
3. In the Code field, enter the script code (write code using Server SDK functions).

NOTEMake sure to include a function within the code, you will call it from the server script in order to use the automation script library.

```
var FTOSExample = FTOSExample || (function () {
        var count = 100;

        this.getCount = function () {
                return count;
        }

        return {
            getCount: this.getCount
        }
});
```



To avoid calling the wrong methods and attributes, you can use the '$m' mechanism when writing the code. For more information on how to the mechanism, see $m Mechanism.

4. At the top-right corner of the page, click one of the save icons. The server automation script library is saved into the system and will be displayed in the Server Automation Script Libraries List.

Using Web API Client Libraries

Web API client libraries allow you to work with external APIs using proxy methods native to the FintechOS development environment. Web API client libraries are generated automatically by Innovation Studio from OpenAPI or WSDL specification files supplied by the web service provider. Once generated, you can import the library into any server automation script that requires access to the API.

Some of the advantages of Web API client libraries are:

- less code to write and maintain
- simplified authorization and authentication
- better code consistency and robustness

HINT
You can access the API specification files of a FintechOS instance at the following locations:

- **OpenAPI**: <*host_address*>/api/openapi/swagger
- **WSDL**: <*host_address*>/Services/ApiService.svc

How to create a Web API client library from an OpenAPI or WSDL specification file

IMPORTANT!

When importing WSDL specification files, Innovation Studio uses the Windows Communication Foundation (WCF) **dotnet-svcutil** tool to generate the Web API client libraries. Make sure that dotnet-svcutil is installed on the deployment server either as a global tool or as a local tool in the bin directory.

You should install the .Net core runtime version that corresponds to the FintechOS instance, even if there are other versions installed on the machine. For more details regarding the installation, go to https://docs.microsoft.com/en-us/dotnet/core/additional-tools/dotnet-svcutil-guide.

Additionally, depending on your .NET Core SDK version, run the following commands:

- For .NET Core SDK 3.0 or above, run:

```
dotnet tool install dotnet-svcutil --tool-path
<WEB_APP_FOLDER>\bin
```

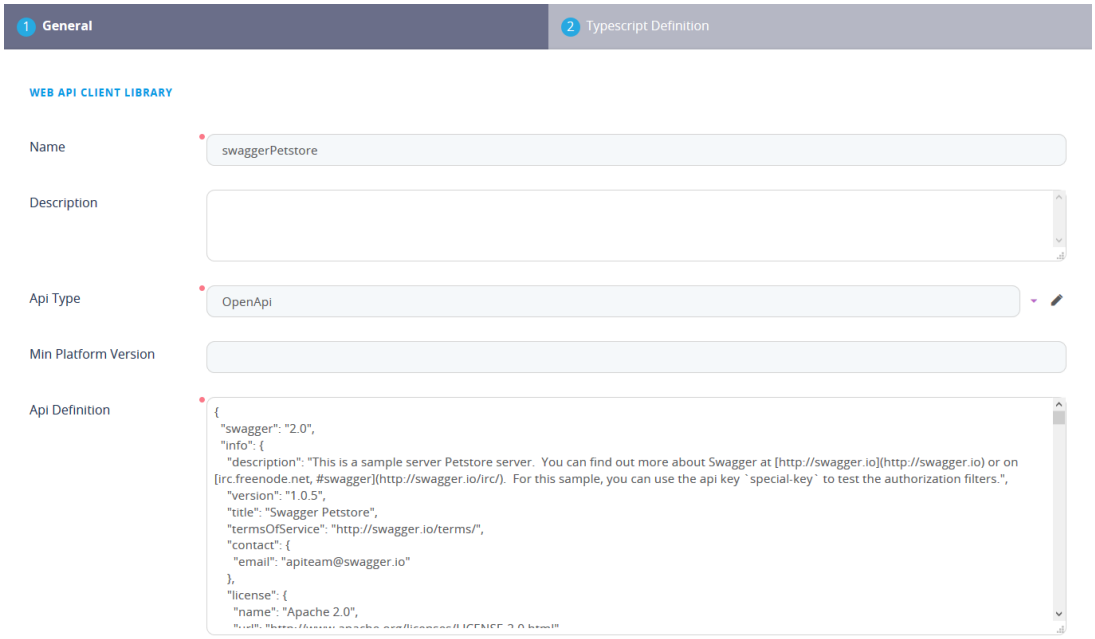
- For .NET Core SDK 2.2 or below:

1. Run

```
dotnet tool install dotnet-svcutil --global
```

2. Copy the <USERPROFILE_FOLDER>\.dotnet\tools folder content to %WEB_APP_FOLDER%\bin

1. In the Main Menu, go to **Advanced** > **Web API Client Libraries**.
2. In the Web API Client Libraries List page, click the Insert (+) button at the top right corner of the page.
3. Fill in the library's details.



- Name - Enter a name for the Web API client library.
- Description - Optional library description.
- API Type - Select whether the source file uses the OpenAPI or WSDL standard.

- Min Platform Version - Optionally select the oldest API version endpoints you wish to extract from the source file.
- API Definition - Paste the contents of the OpenAPI or WSDL specification file.

4. Click the **Save and Reload** ( ) button at the top right corner of the page. After the library is generated, you can review its definitions in the Typescript Definition tab.
5. Click the **Save and Close** ( ) button at the top right corner of the page.

How to use a Web API client library in server automation scripts

In your automation script, use the **importWebApiClient** function to import the Web API client library in a JSON object that contains the API specifications. The object exposes methods matching the API's endpoints and has full IntelliSense auto-complete support.

For details, see the Server SDK Reference Guide.

Add certificate support to WebApi client and WCF client

WebAPI client offers a way for consuming external web services by providing a definition for the external service with an OpenAPI or Wsdl file and import it into the platform and provide an object to use. This feature allows a user to work with web services and web sites that require a certificate authentication for example Certsign.

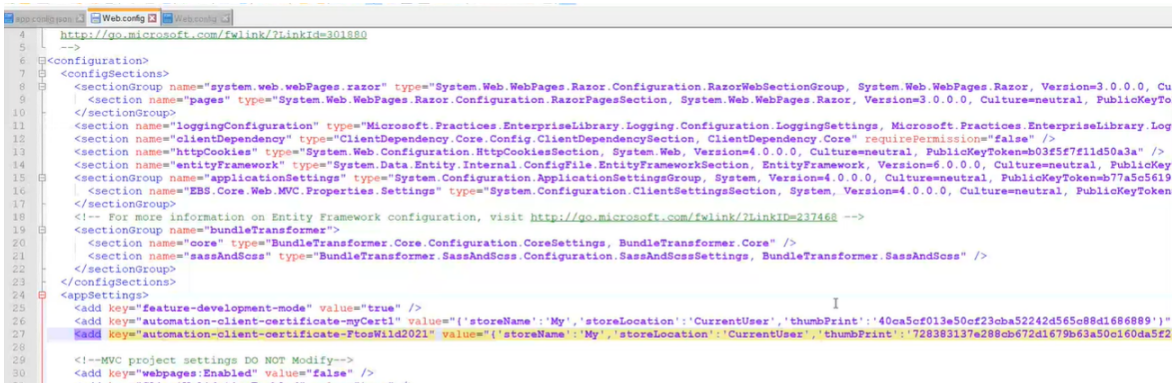
HINT
To do so, the user needs a valid certificate.

```
let client = importWebApiClient('SecureFTOS
client.setCertificate(server.clientCertificates.get());

try {
    var authToken = client.authorize.getToken({
        client_id: 'client_id',
        username: 'host',
        password: '1234567'
    });
    if (authToken && authToken.access_token) {
        let data = client.openApi.query({
            apiInfo: {
                userName : "host",
                token :  authToken.access_token
            },
            request: {
                entity: {
                    name: "webApiClientLibrary",
                    alias: "api"
                },
                distinct: false
            }
        });
        log(data);
    }
    else
        throw new Error('Invalid authentication!');
}
catch(err) {
    log(err);
    throw err;
}
```

setCertificate(**workflowClientCertificate: WorkflowClientCertificate**): void

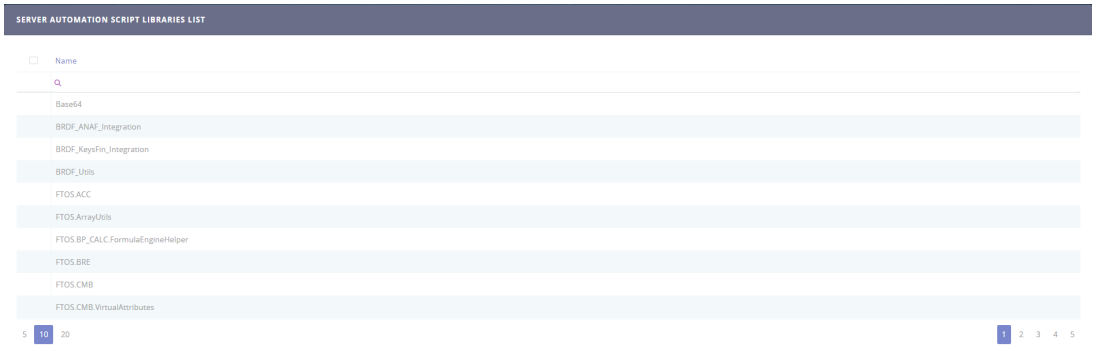
There is a method for this process setCertificate which accepts workflow client certificate, this is to actually use a certificate. By using a server method that provides the certificate, this method uses the configurations from web.config. In the web.config file, the certificate needs to be configured by providing a storeLocation, storeName, thumbPrint.



Creating Event Triggered Automation Scripts

To create an event triggered automation script, follow these steps:

1. From the menu, click Advanced > Server Automation Scripts. The **Server Automation Scripts List** page appears:



2. At the top-right corner of the page, click the **Insert** icon. The **Add Server Automation Script** page appears:
3. In the Name field, type the script name.
4. In the Description field type a description of the script logic. The field is optional, but we recommend you to provide description so that developers have a clear view on what the script is intended to do.
5. From the Script Type drop-down, select **Event triggered**.
6. From the Event drop-down, select the event type that triggers the script: **Read**, **Update**, **Insert** or **Delete**.
7. From the Stage drop-down, choose the stage of event that triggers the automation script:

Stage	Description
Before	Executes a block of code before read, update, insert or delete events occur. **Use cases**: To validate information for update / read events and restrict access /filter information for read events.
After	Executes a block of code after read, update, insert or delete event. Using the record ID, you can use it to create single or cascading events.
After Transaction	Waits until the current SQL transaction finishes and then the automation script runs in a new transaction. **Use case**: To insert and update the same record at the same time. If using an after event in the automation script, the second transaction (update) will execute after the first one (insert) completes.

8. From the Entity drop-down, select the entity triggering the server automation script.
9. In the Code field, enter the server automation script code (using Server SDK functions). If you want to use an automation script library, you can do so, by calling functions defined in the library.

```
log('Client - BeforeUpdate - START - ' + context.Id);
log('CONTEXT: ' + serialize(context));
setAdminMode(true);
if(context.ExecutionDeapth < 2){
    var clientName = isNullOrEmpty
(context.Values.AgreementCounterpartyName) ?
context.BeforeValues.AgreementCounterpartyName :
context.Values.AgreementCounterpartyName;
    clientName = clientName.trim();
    context.Values.AgreementCounterpartyName = clientName;
    context.Values.InitialCounterpartyName = clientName;
```

```
    var fenergoClientId = isNullOrEmpty
(context.Values.FenergoClientId) ?
context.BeforeValues.FenergoClientId :
context.Values.FenergoClientId;
    var clientId = context.Id;
    if(!isNullOrEmpty(fenergoClientId)){
        var duplicateClient = getByQuery({
            "entity": {
                "alias": "a",
                "name": "Client"
            },
        "attributelist": [
            {
            "name": "AgreementCounterpartyName"
            },
            {
            "name": "Clientid"
            }
        ],
    "where": {
        "type": "and",
        "conditionlist": [{
            "first": "a.FenergoClientId",
            "type": "equals",
            "second": "val(" + fenergoClientId + ")"
            },
        {
        "first": "a.Clientid",
        "type": "notequals",
        "second": "val(" + clientId + ")"
        }]
        }
    }
);
if(duplicateClient !== null && duplicateClient.length !== 0){
    throwException(getErrorMessage("61111"));
    }
}

    var duplicateClient = getByQuery({
        "entity": {
        "alias": "a",
        "name": "Client"
    },
    "attributelist": [
```

```
            {
            "name": "AgreementCounterpartyName"
            }
        ],
        "where": {
            "type": "and",
            "conditionlist": [{
                "first": "a.AgreementCounterpartyName",
                "type": "equals",
                "second": "val(" + clientName + ")"
            },
            {
            "first": "a.Clientid",
            "type": "notequals",
            "second": "val(" + clientId + ")"
            }]
            }
        }
    );
    if(duplicateClient !== null && duplicateClient.length !== 0){
        throwException(getErrorMessage("61111"));
        }
    }
    log('Client - BeforeUpdate - END - ' + context.Id);
```

To avoid calling the wrong methods and attributes, you can use the '$m' mechanism when writing the code. For more information on how to the mechanism, see $m Mechanism.

If you want to use an automation script library (the one whose functions you appended in the Code field), after saving the automation script, go to the Edit Automation Script page, from the List of Automation Script Libraries section, click the Insert existing button and select the desired script library by double-clicking it.

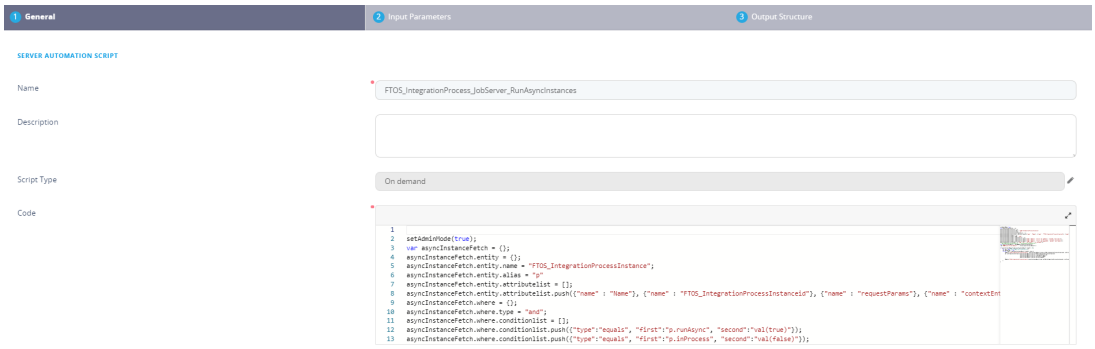
10. To prevent recursive run of the scripts, stick the Prevent Recursivity checkbox.
11. Specify if the script is active or disabled. Tick the Disable checkbox to disable the automation script code execution (select it for debugging purposes or for obsolete automation scripts).

12. At the top-right corner of the page, click the Save and close icon to save the automation script.

Creating On-demand Server Automation Scripts

To create an on-demand server automation script, follow these steps:

1. From the menu, click **Advanced** > **Server Automation Scripts**. The Server Automation Scripts List page appears:
2. At the top-right corner of the page, click the **Insert** icon. The Add Server Automation Script page appears.
3. In the General tab, enter a unique **Name** for the automation script.
4. In the **Description** field type a description of the automation script logic. The field is optional, but we recommend you to provide description, so that developers have a clear view on what the automation script is intended to do.
5. From the **Script Type** drop-down, select **On-Demand**.



6. In the **Code** field, enter the automation script code (using using Server SDK functions). If you want to use an automation script library, you can do so, by calling functions defined in the library.

```
//parameter called input1 received from client <a
href="https://fintechos.com/documentation/ClientSDK/#Other/e
bs.callActionByName.htm">callActionByName</a> function
var input1 = context.Data.input1;

//call to a method defined in server script library
var cnt = new FTOSExample().getCount();

//the custom returned object
var acc = {totalCount: cnt, test:"example1", resInput1:
input1};

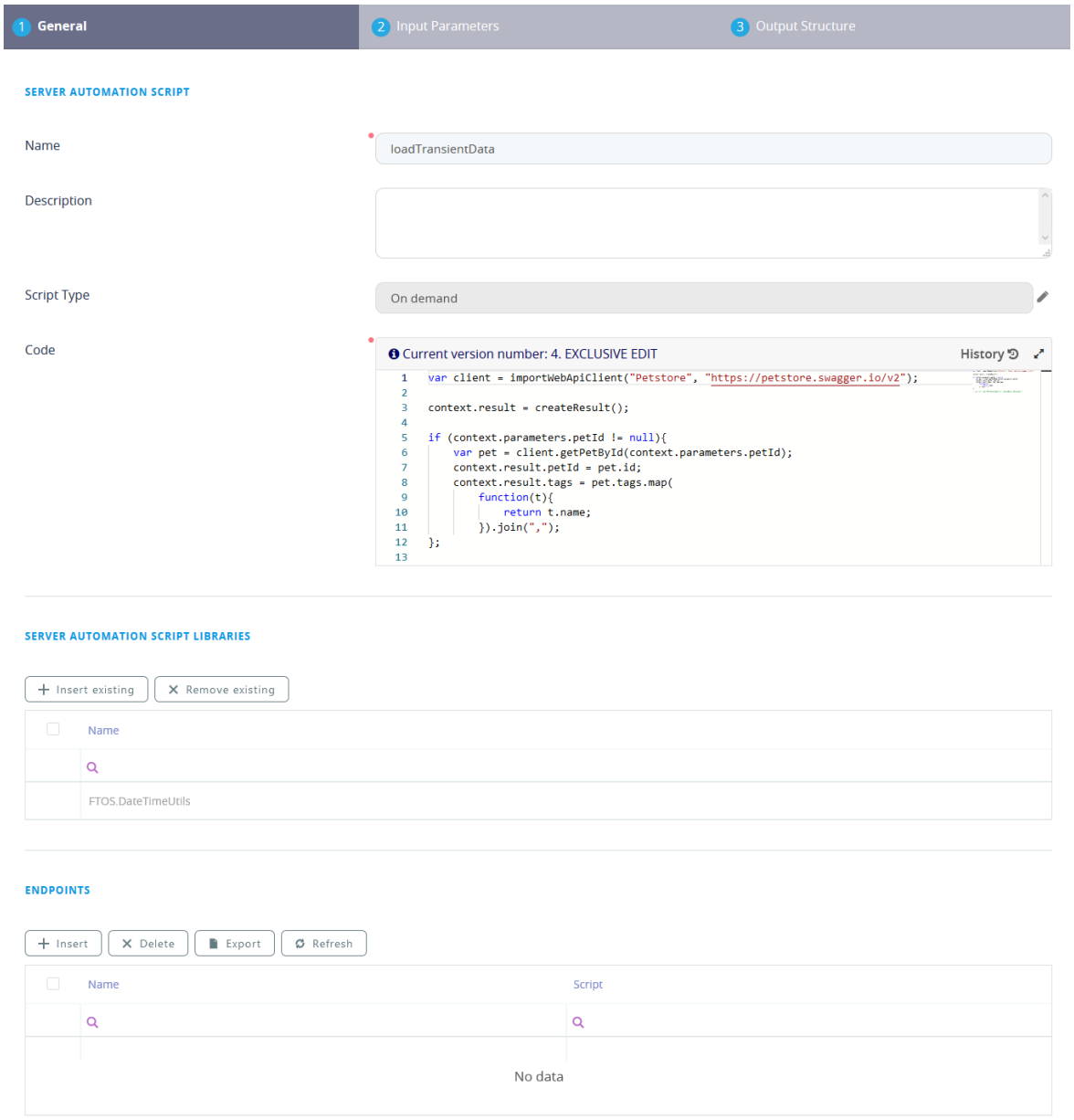
//returns data in UI (on callback of the callActionByName))
setData(acc);
```

To avoid calling the wrong methods and attributes, you can use the '$m' mechanism when writing the code. For more information, see "Code snippets for entities and attributes" on page 347.

NOTE If you want to use an automation script library (the one whose functions you appended in the **Code** field), after saving the automation script, go to Edit Automation Script page, from the List of Automation Script Libraries section, click the **Insert existing** button and select the desired script library by double-clicking it.

7. At the top-right corner of the page, click the **Save and reload** ( ) icon to save the automation script. This will enable the Server Automation Script Libraries and the Endpoints sections at the bottom of the page.
8. Use the **Server Automation Script Libraries** section to select any script libraries your automation script uses. For details about automation script libraries, see "Using Automation Script Libraries" on page 484.
9. Use the **Endpoints** section to define any endpoints that call the automation script. For more details about endpoints, see "Creating Endpoints" on page 504.

10. Clic **Save and reload** ( ).



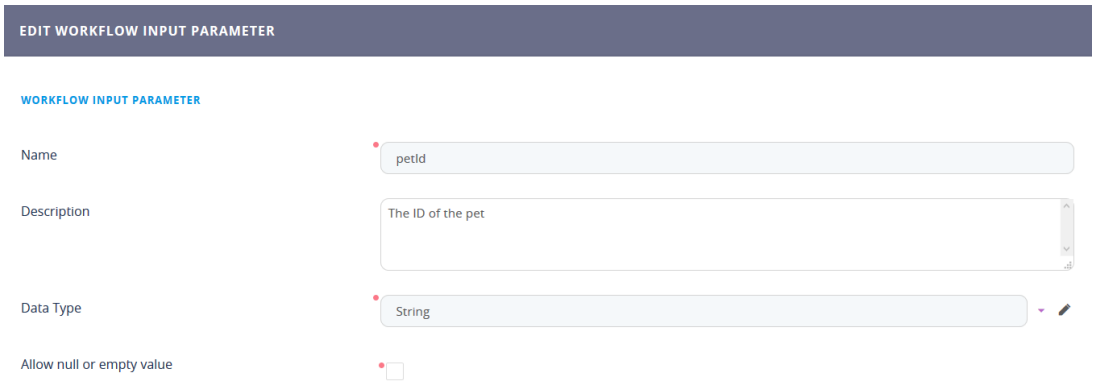
Customizing Input Parameters

You can define mandatory input parameters that must be passed to the script by the client process and you can enable intelligent code completion in the code editor for the script's input parameters. Input parameters are typically passed to the automation

script by a client side script using an ebs.callAction type of function, or by an API call using the CallAction endpoint.

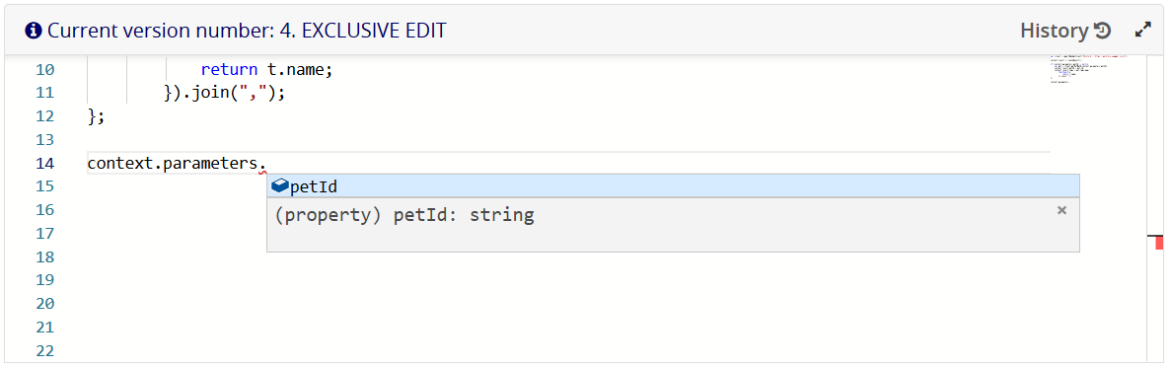
To define an input parameter:

1. Open the automation script in the editor and select the **Input Parameters** tab.
2. In the Workflow Input Parameters list, click the **Insert** button to add a new parameter to the list.
3. Fill in the input parameter's details.



   - **Name** - Enter a name for the input parameter. The name must match the incoming variable name provided by the client process.
   - **Description** - Optionally enter a description for the parameter.
   - **Data Type** - Select the parameter's data type. Currently supported data types are any, numeric, boolean, and string JavaScript object types.
   - **Allow null or empty value** - Leave empty to make the parameter mandatory for the automation script's execution. Selecting this checkbox makes the parameter optional.
4. Click the **Save and close** button ( ) at the top-right corner of the page.

Once defined as above, you can use the `context.parameters` property in the code editor to access the script's input parameters with intelligent code completion.

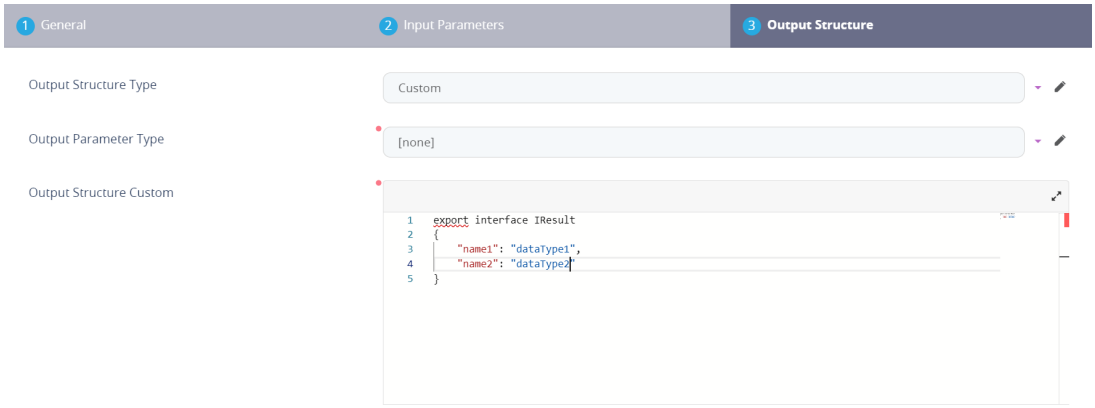


Customizing the Output Structure

You can customize the script's output structure to enable intelligent code completion for the result passed to the client-side callback function. You can map the output structure to an entity data model, or you can define you own custom structure. Also, you can specify if the output is in the form of a single object instance, or if it is a collection of objects each matching this output structure.

To define the script's output structure:

1. Open the automation script in the editor and select the **Output Structure** tab.



2. Select the **Output Structure Type**:

    - **Entity** - The output structure is based on an entity data model, matching the entity's attributes' names and types.
    - **Custom** - Select this option if you wish to define the script's output structure manually.

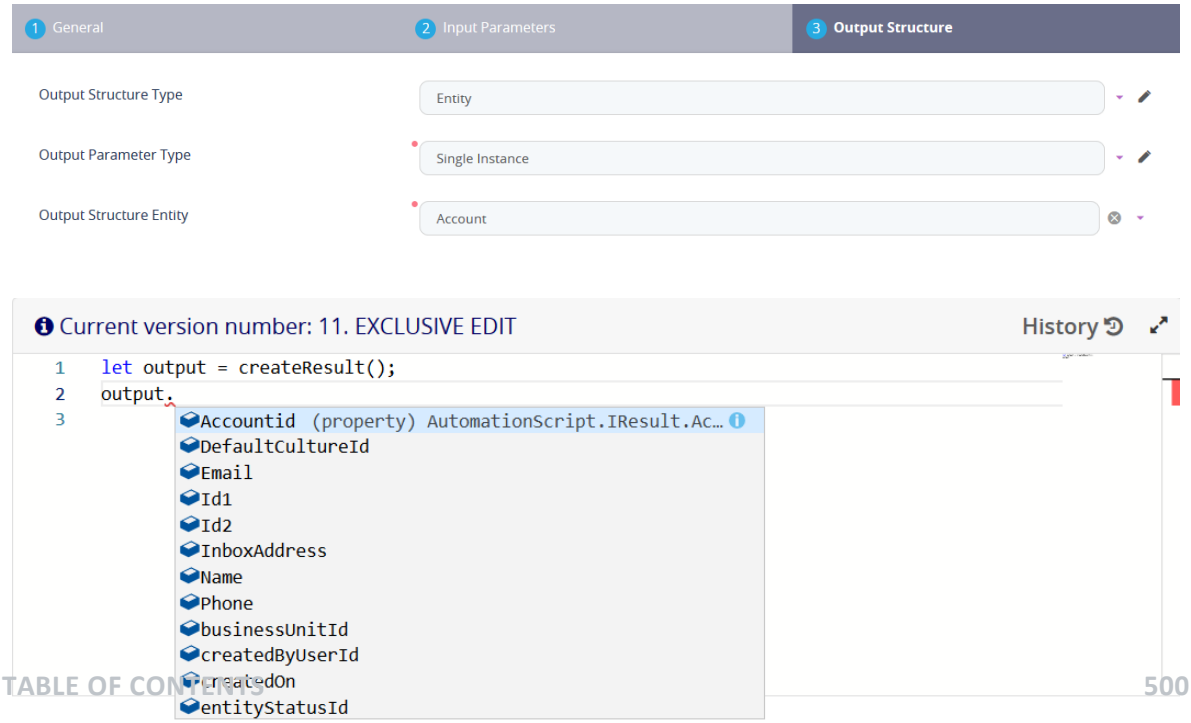
- **Boolean** - Select this option if the script returns either a *true* or *false* result, for instance if you wish to use it for validations.

3. Select the **Output Parameter Type**:
    - **Single Instance** - The script result is a single object instance.
    - **Collection** - The script result is a collection of objects.
4. If you selected an output structure type based on an entity, select the **Output Structure Entity**. This is the entity providing the data model for the output structure.
5. If you selected a custom output structure type, fill in the **Output Structure Custom** field in the following format:

```
export interface IResult
{
    "name1" : "dataType1",
    "name2" : "dataType2"
}
```

6. Click the **Save and close** button ( ) at the top-right corner of the page.

After you define the output structure as above, you can use the `createResult()` and `createResultItem()` constructors to create result objects with intelligent code completion in the code editor.



You can attach the result object(s) to the `context.result` property which is passed to the client-side callback function.

```
var res = createResult();
res.Name = 'test name';
res.Email = 'x@mail.com';

context.result = res;

//or

var item1 = createResultItem();
item1.Name = 'test name';
item1.Email = 'x@mail.com';

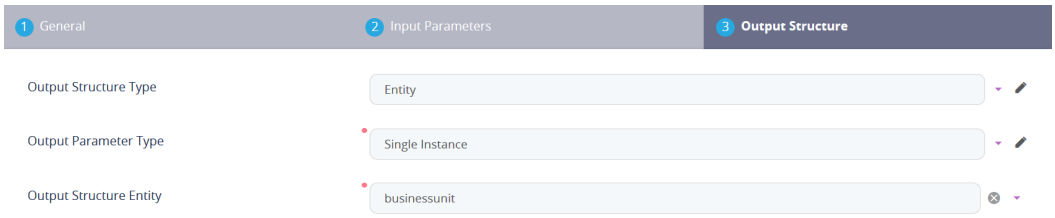

var item2 = createResultItem({
    Name : 'test name 2',
    Email : 'y@mail.com'
});

context.result.push(item1,item2);
```

Examples

Create an entity-based single instance output

In this example, we create a single instance output based on the **businessunit** entity, which has the following attributes: **businessunitid**, **name**, and **parentBusinessUnitId**.

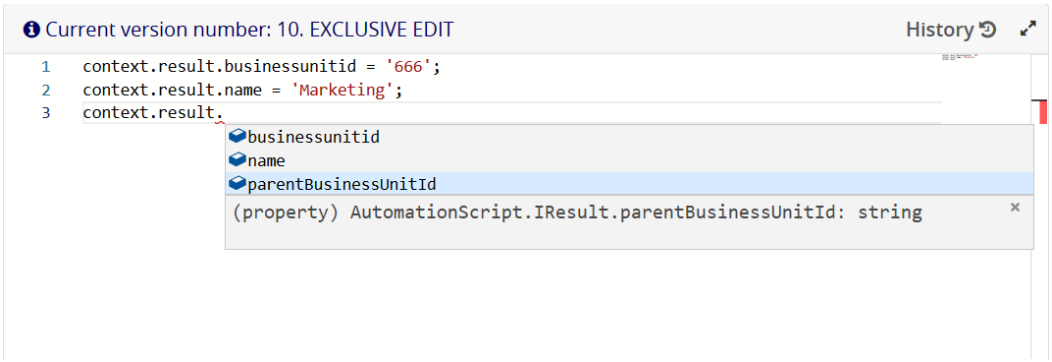


In the automation script code, we populate the result with the Marketing business unit details.

```
context.result = createResult();
```

```
context.result.businessunitid = '666';
context.result.name = 'Marketing';
context.result.parentBusinessUnitId = '001'
```

The customized output structure has provided intelligent code completion when setting the result.



Create a custom collection output

In this example we define a custom output collection with each result item storing a **name** and **age**:

```
export interface IResult
{
    name : string;
    age : number
}
```

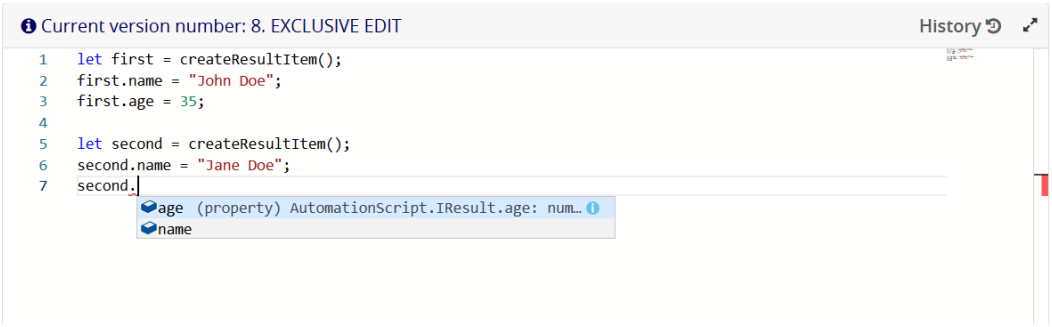
In the automation script code, we populate the output collection with two entries (**first** and **second**) corresponding to **John Doe** and **Jane Doe**, aged **35** and **40** respectively.

```
let first = createResultItem();
first.name = "John Doe";
first.age = 35;

let second = createResultItem();
second.name = "Jane Doe";
second.age = 40;
```

```
context.result.push(first, second)
```

The customized output structure has provided intelligent code completion when setting the result items.



Creating Endpoints

Endpoints specify the location from which FintechOS APIs can access the resources they need to carry out their function. They play a key role in guaranteeing the correct functioning of the software that interacts with it.

You can add an endpoint, attach security roles, attach an automation script to it, then on digital journeys or data forms call action to it.

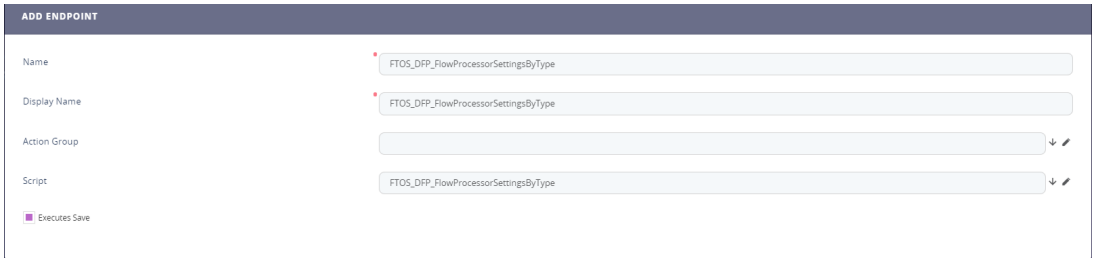
You can add endpoints either from the from Endpoints List page, or from the Edit Server Automation Script page, Endpoints section by clicking the **Insert** button. The **Script** field will be prefilled with the name of the automation script for which you create the endpoint.

1 Create an endpoint

This section describes how to add an endpoint from the Endpoints page:

1. From the menu, click **Advanced** > **Endpoints**. The Endpoints List page appears.
2. At the top-right corner of the page, click the **Insert** icon. The Add Endpoint page appears.

3. In the Name field type the name of the endpoint which will be used by the system.
4. In the Display Name field, type the name of the endpoint which will be displayed on the button in the user interface.
5. (Optionally)If you are working with actions groups, from the Action Group field, select the desired action group you have previously created. For more information on action groups, see Defining Action Groups.
6. From the Script drop-down, select the on-demand automation script you previously created.



7. At the top-right corner of the page, click the Save and close icon to save the endpoint or Save and reload to attach a security role to the endpoint.

You can now go to the digital journey or data form and call action to the server automation script.

2 Attach a security role to an endpoint

For higher security, you can now choose which security roles have the privileges to call actions on endpoints.

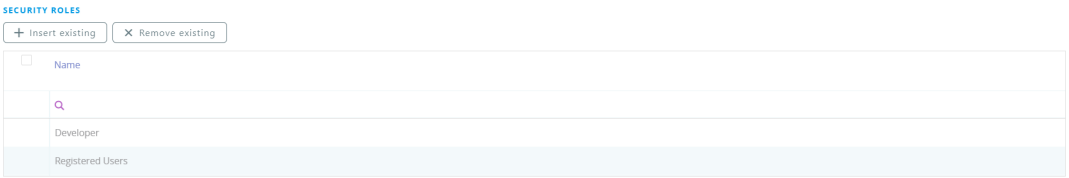
When calling actions on endpoints which have no security roles attached, errors will occur and the actions will not be performed.

NOTE
For backwards compatibility, the security role “Registered Users“ is automatically added to all endpoints created in previous versions of Innovation Studio.

> ⚠ **IMPORTANT!**
> The security role “Registered Users“ that is automatically added to all endpoints created in previous versions of Innovation Studio does not ensure the backwards compatibility for unauthenticated portals. In this case, you need to manually configure or import the security roles assignment.

To attach a security role to an endpoint:

1. Go to the endpoint configuration page (Edit Endpoint page), scroll down to the SECURITY ROLES section and click the Insert existing button. A window appears which lists all defined security roles.
2. Select one from the list by double-clicking on it or add a new security role (click the Insert button and provide all details required to add a new security role). The selected security role displays in the SECURITY ROLES section.



3. At the top-right corner of the page, click the Save and close icon to save the endpoint or Save and reload to attach a security role to the endpoint.

Calling Actions

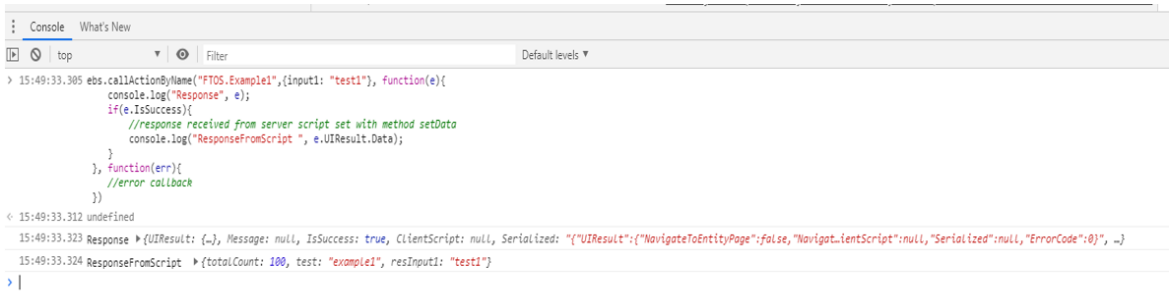
Prerequisite: To append a callAction to execute a server automation script on demand, you need to have the automation script that you want to execute and an endpoint for that script.

On the digital journey / data form where you want the automation script to be executed on demand, in the JavaScript fields (Advanced tab), append the callActionByName method.

```
ebs.callActionByName("FTOS.Example1", { input1: "test1" }, function
(e) {
```

```
        console.log("Response", e);
        if (e.IsSuccess) {
            //response received from server script set with method
setData
            console.log("ResponseFromScript ", e.UIResult.Data);
        }
}, function (err) {
        //error callback
});
```

The action button will be displayed in the user interface on the digital journey / data form. When the user clicks the button, the automation script associated with it will be executed in the browser's console:



NOTEcallAction connects an automation script with the front end (UI); therefore, you cannot append an action from another action.

If you have specific functionalities (e.g. functions1) in an automation script and other functionalities (e.g. functions2) in another automation script, you have two ways to execute them on demand:

- On forms / user journeys, in aftergenerateJs fields append both actions.

  ```
  Ebs.callActionByName(“action1”,{param1:”asd”} callback(e){
       Ebs.callActionByName(“action2”,{param2:”aaa”} callback
  (e){
                  });
  });
  ```

  where:

- action1 is the name of the endpoint to the automation script which contains functions1
- action2 is the endpoint to the automation script which contains functions2

- Group functions1 and functions 2 in an automation script library. For both automation scripts add reference to the same automation script library while in the Code field of the scripts you append only the functions specific for that script,

Scheduling Server Automation Scripts

In Innovation Studio, engineers can plan an automation script to run periodically by using the Scheduling feature.

A scheduled job can be set up to run at a fixed interval of time starting with Start Time (Pool Time) or using a Cron Expression.

To schedule a server automation script, follow these steps

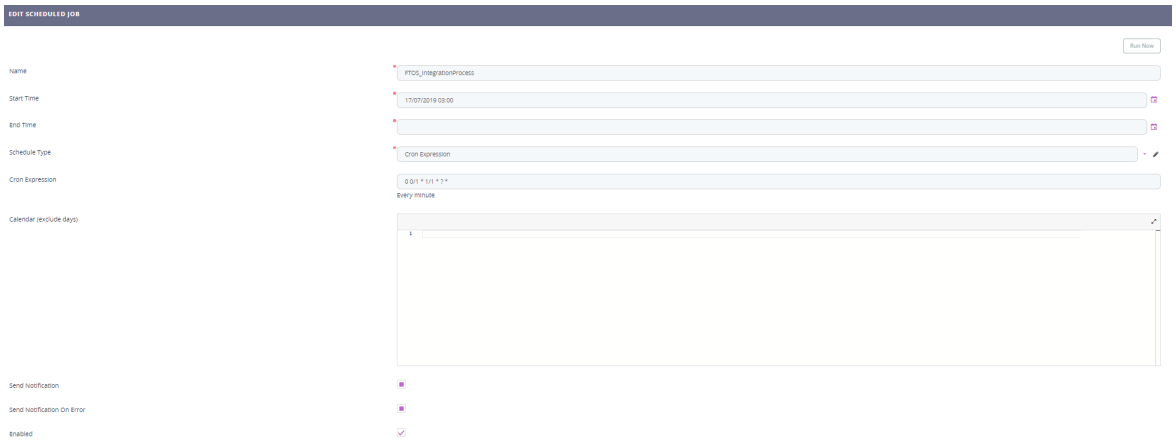
STEP 1. Add schedule job

To add a scheduled job, follow these steps:

1. From the menu, click Business Automation > Scheduled Jobs. The **Scheduled Jobs List** page appears.
2. At the top-right corner of the page, click the Insert icon. The **Add Scheduled Job** page appears.
3. Fill in the mandatory fields (marked with a red asterix).
4. (Optionally) If you want to set up specific days when the job will not execute, in the Calendar (exclude days) field , provide the code to do so. The types of calendar exclusion can be: ANNUAL (dd.MM), MONTHLY (dd), WEEKLY (days in week: Monday, Tuesday, .etc), HOLIDAY (dd.MM.yyyy).

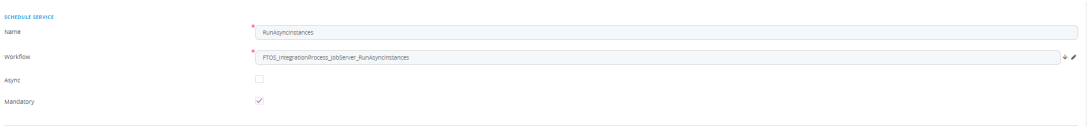
5. Tick the Enabled checkbox to enable the scheduled job, otherwise it will not run.
6. At the top-right corner of the page, click the Save and reload icon. The record is saved in the system and the Edit Scheduled Job page appears.

Now you can add schedule services.



STEP 2. Add schedule services

1. In the Edit Scheduled Job page, scroll-down to the Schedule Services section and click the Insert button. The Add Schedule Service page appears.
2. In the Name field, provide a name for the service.
3. Click the down arrow next to the Workflow field. A pop-up appears listing all existing server automation scripts.
4. Select the server automation script you want to schedule by double-clicking on it.



5. At the top-right corner of the page, click the Save and close icon to save the record. Add as schedule services as automation scripts you want to schedule for execution.

If you have more than one scheduled service in the Schedule Services list, you can set their execution order.

STEP 3. Set the execution order

If you have several automation scripts scheduled and need them to be run in a specific order, drag and drop records in the Schedule Services section in their execution order (whereas the first record in the section is the first one to be executed).

If you choose that one of the service is mandatory and it fails, all the following services (scripts) will no longer execute. Also, if you will choose to run script async (when adding /editing a schedule service by selecting the **Async** checkbox), the order of services will be disregarded and all automation scripts will run in parallel.

```
[{
        name:"calAnual",
        type:"ANNUAL",
        excludeDays:["03.11","17.01"]
      },
    {
        name:"calMonthly",
        type:"MONTHLY",
        excludeDays:["25","10"]
    },
    {
        name:"calWeekly",
        type:"WEEKLY",
        excludeDays:["Saturday","Sunday"]
    },
    {
        name:"calHoliday",
        type:"HOLIDAY",
        excludeDays:["25.12.2018","01.01.2019"]
    }
]
```

NOTEIn order for the cron jobs to trigger the automation script execution, the Job Server should be installed on deployment.

Job Server is aware of jobs changes (time, cron expression, reorder of services).

For all failed jobs, a Run now button is displayed which allows running the jobs again. The job data model has been updated to include a parent job id. When running a failed job (clicking the Run now button), a copy of the job is made having the parentJobId = original Job and it will be scheduled to run only once in 1 minute.

Using Plugin Assemblies

By using plugin assemblies, you can write custom C# code which can be triggered by on-demand or event-triggered scripts. Plugin assemblies is triggered similar to scripts on insert/update/delete.

This section walks you through the steps that you need to follow to use plugin assemblies.

STEP 1. Add Plugin Assembly

1. From the menu, click Advanced > Plugin Assemblies. The Plugin Assemblies List page appears.
2. At the top-right corner of the page, click the Insert icon. The Add Plugin Assembly page appears.
3. Click the Add file, browse for the plugin assembly (dll) file and select it.

   NOTEThe version of the dll file must be the same with the product version. When upgrading FintechOS, make sure that you manually upgrade the plugins.

4. In the Name field, enter the plugin name that will be used by the system.
5. At the top-right corner of the page, click the Save and reload icon. The Edit Plugin Assembly page appears.

Now you can add the plugin and the UI processor.

STEP 2. Add the IEbsPlugin Plugin

In the Edit Plugin Assembly page, go to the Plugins section and click the Insert button. The Add Plugin page appears. In the Name field, type **IEbsPlugin**. At the top-right corner of the page, click the Save and reload icon.

STEP 3. Add UI Processor

In the Edit Plugin Assembly page, go to the UIProcessors section and click the Insert button. The Add UIProcessor page appears. In the Name field, type **IEbsProcessor**. At the top-right corner of the page, click the Save and reload icon.

XML Support

XML support is available in server automation scripts and libraries, allowing you to create and parse XML.

This is helpful if you want to use data stored as formatted data source in XML. For example, you might want to use in FintechOS, product or order related information you have already stored in XML format, instead of creating new entities and attributes.

This section covers the following topics:

Load XML from String

Method

```
server.Xml.Load(string xml)
```

Loads the XML from the specified string.

Parameter

```
xml string
```

String containing the XML to load. The string is XML formatted text.

```
var xml = '<Order></Order>';
var doc = server.Xml.Load(xml);
```

Catch XML Load Error

When loading XML from a string, errors that might occur on XML load from a string are not automatically logged. To log any errors that might occur on XML schema load, use `catch(err)`.

```
var xml = '<Order />';
try
{
    var doc = server.Xml.Load(xml);
}
catch(err)
{
    log(err);
}
```

Run XPath Queries

You can perform XPath queries to navigate through nodes (elements, attributes) in an XML document.

Method

```
Query( xpath : string ) : Element[]
```

```
var xml = '<?xml version="1.0" encoding="utf-8" ?>
<Orders>
     <Order id="O1">

            <Product id="1" name="Nexus 5">
                <Price>400.00</Price>
                <Qty>1</Qty>
            </Product>
            <Product id="5" name="Wireless Charger">
                <Price>50.00</Price>
                <Qty>1</Qty>
            </Product>
     </Order>
     <Order id="O2">
            <Product id="2" name="IPhone">
```

```
                <Price>800.00</Price>
             <Qty>1</Qty>
            </Product>
            <Product id="5" name="Wireless Charger">
                <Price>50.00</Price>
                <Qty>1</Qty>
            </Product>
            </Order>
</Orders>';
var products = doc.Query("/Orders/Order/Product[@name='Wireless
Charger']");
log(products[0]['id']);
log(products[1]['id']);
```

Where:

- **<Orders>** is the root element
- **Order** is an element node
- **<Price>** and **<Qty>** are child elements of **<Product>**
- **Wireless Charger** is an attribute node

Run XPath Queries with Namespaces

You can perform XPath queries with namespaces to navigate through nodes (elements, attributes) in an XML document.

Method

```
Query( xpath : string, namespaces : { key : string, value : string
)
```

```
var xml = '<?xml version="1.0" encoding="utf-8" ?>
<x:Orders xmlns:x="http://myuri">
        <x:Order x:id="O1">
            <Product id="1" name="Nexus 5">
                <Price>400.00</Price>
                <Qty>1</Qty>
            </Product>
            <Product id="5" name="Wireless Charger">
                <Price>50.00</Price>
                <Qty>1</Qty>
```

```
            </Product>
        </x:Order>
        <x:Order x:id="O2">
            <Product id="2" name="IPhone">
               <Price>800.00</Price>
                <Qty>1</Qty>
            </Product>
            <Product id="5" name="Wireless Charger">
                <Price>50.00</Price>
                <Qty>1</Qty>
            </Product>
        </x:Order>
</x:Orders>
var doc = server.Xml.Load(xml);
var products = doc.Query("/x:Orders/x:Order/Product[@name='Wireless
Charger']", {'x' : 'http://myuri' });
log(products[0]['id']);
log(products[1]['id'])
```

Node API Calls

The table below lists the API calls you can do on the nodes within an XML document.

Property	Returns	Description
HasAttributes	Boolean	Gets a value indicating whether this element has at least one attribute. **Property Value**: true if the current node has attributes; otherwise, false.
HasElements	Boolean	Gets a value indicating whether this element has at least one child element. **Property Value**: true if the current node has child elements; otherwise, false.

Property	Returns	Description
AttributeCount	Number	Gets the number of attributes on the current node (element). **Property Value**: The number of attributes if the current node (element) has attributes; otherwise, null.
ElementCount	Number	Number Gets the number of elements on the current node. **Property Value**: The number of elements if the current node (element) has child elements; otherwise, null.
Elements() : Element[]	Array of strings	Return all child elements of a node element.
Elements(name : string) To specify namespace use following syntax for name “{http://myuri.org}name”	Array of strings	Return all child elements with the specified node element.
this[attributeName : string] To specify namespace for attribute use following syntax for name “{http://myuri.org}name”	String	Gets the value of the specified node attribute.

Debugging Automation Scripts

FintechOS offers several options for debugging automation scripts from the development and testing :environments:

Debugging Log

The Debugging Log adds information in the log file and continues the automation script execution.

If the script breaks due to a **throwException** or to an unexpected error, the log information will be written in the log file.

How to use log for debugging purposes:

```
log("log 1");
log(newQuoteValabilityStartDate);
log("offer number" + '' +quote.Name);
log("test function"+birthDay +" "+ gender);
log("quote = " + serialize(quote));
```

Throw Exceptions

Break the automation script execution and display the message as specified within the **throwException** statement.

When the throwException method is called in an automation script, the passed error message is now available to users.

Examples of how throwException can be used for debugging purposes:

```
throwException(serialize("test"));
throwException(serialize(result.TotalNetProfit));
throwException(serialize(context));
throwException(String(Topic[0].a_Topic));
throwException(JSON.stringify(result));
throwException(JSON.parse(result));
```

JavaScript Exceptions

When parsing invalid Xml in automation scripts, JavaScript Exceptions are now caught by `try catch` in JavaScript:

```
try
        {
        // load invalid XML
        var doc = server.Xml.Load( '<a> ... <' );
}
catch(err)
        {
        // handle error
 }
```

Console Debugging

Server-side errors are displayed within Developer Tools. During development and testing phases, engineers able to track errors raised on the server-side directly in the browser Developer Tools.

Browser developer tools to use for debugging automation scripts:

- JSON Parser
- Javascript beautifier

The error output displayed in the Console is particularly useful when raising issues. Include the error output in the issue description to provide a complete overview of the error and reduce the investigation time.

NOTEConsole debugging can be used ONLY on development machines and in testing environments.

On development machines

For IISExpress, on the development machine open an elevated Command Prompt and run the following command:

C:\work\EBSCore\current>SETX\ebs-development-mode 1

For IIS, make sure that you set the variable at the system level (not on the user level):

C:\work\EBSCore\current>SETX\ebs-development-mode 1 /m

On testing environments

To debug on testing environments, go to the **web.config** file and add the following section:

```
<appSettings>
     ...
    <add key="ebs-development-mode" value="1 "/>
</appSettings>
```

Code Blocks

The Code Blocks feature enables FintechOS developers to insert predefined blocks of code into attributes of type After generate JS.

Code blocks are designed to be extensible and configurable. You can define your own code blocks and configure them based on your needs.

> **NOTE**
> Do not confuse script libraries with code blocks. Unlike the client script libraries and server automation script libraries which you call directly in scripts, you need to do some changes in a code block to fulfill your needs (attributes, etc.). Within a code block you can call existing script libraries.

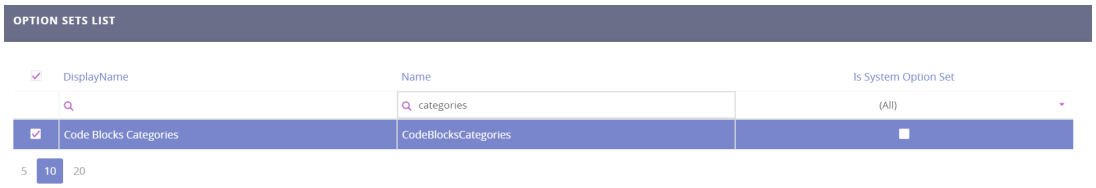
Code blocks are grouped per categories to help you easily spot the one that you want to use. To add code blocks, make sure that you have categories defined.

1 Add categories

Categories help you group code blocks.

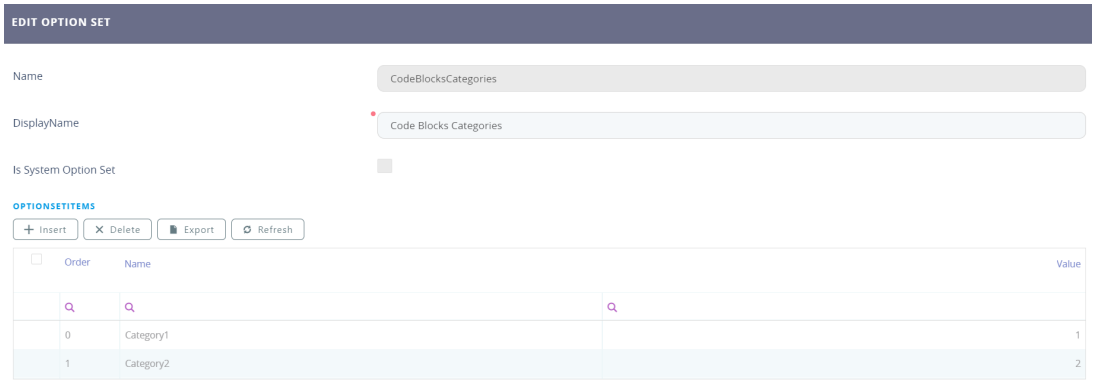
To add a code block category:

1. From the menu, click **ADMIN** > **Option Sets**. The Option Sets List page appears.
2. Search for the CodeBlocksCategories option set. You can do a partial Name or Display Name search by ''categories', or an exact Name match search by 'CodeBlocksCategories' or an exact Display Name match search by 'Code Blocks Categories'.



3. Double-click on search result. The Edit Option Set page appears.
4. Click the **Insert** button at the top of the OptionsetItems section. The Add OptionSetItem page appears.
5. In the **Name** field, type a name that will be used by the system and in the **Display Name** field, type the name that will be displayed in the user interface.

6. At the top -right corner of the page, click the **Save and close** icon to save the category. Add as many categories as best suit your needs. They are displayed in the OptionsetItems section.



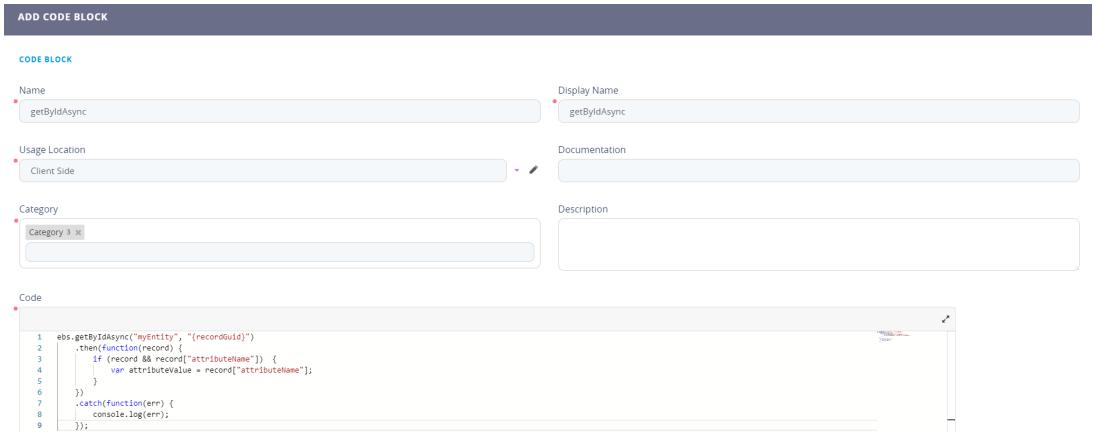
Now you can add code blocks.

2 Add code blocks

To add a code block:

1. From the main menu, click **Advanced** > **Code Blocks**. The Add Code Block page appears.
2. Fill-in the following fields:

Field	Description
Name	The category name used by the system. The field is mandatory
Display Name	The category name to be displayed in the user interface. The field is mandatory
Usage Location	The place where the code block will be available: **Client Side** or **Server Side**. The field is mandatory
Documentation	If you have the code block documented, provide the URL to the documentation.
Category	The category to which the code block will belong to. The field is mandatory.
Description	Provide a brief description of the code block to help others easily identify the scope of the code block.
Code	JavaScript code which will be inserted in After generate JS when using code blocks. The field is mandatory.



3. At the top-right corner of the page, click the **Save and close** icon to save the code block. Add as many code blocks as you need to increase efficiency.

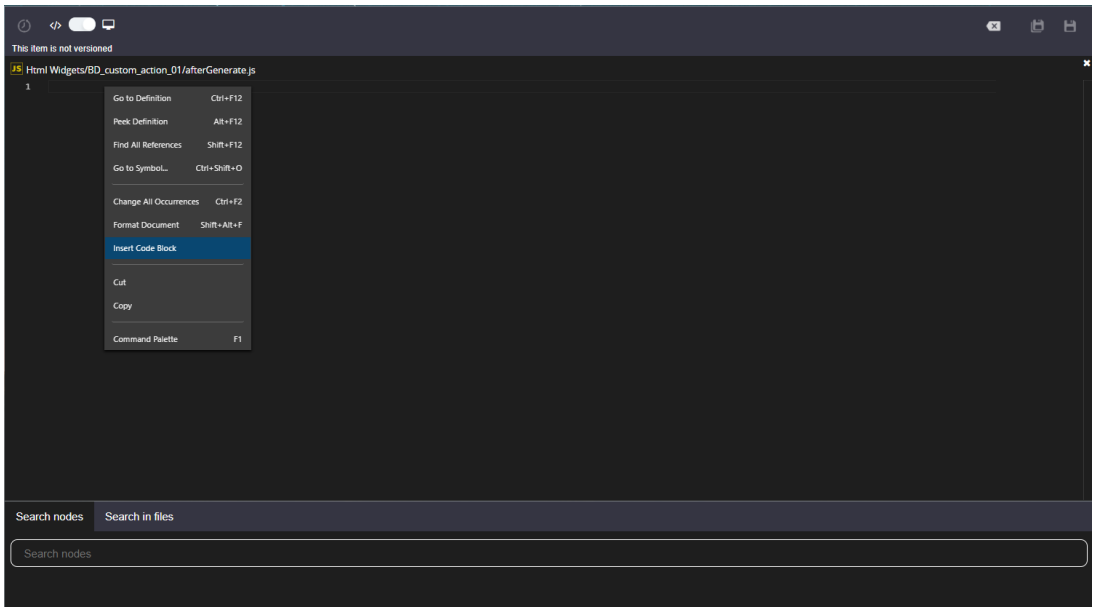
You can now start using code blocks

3 Use code blocks

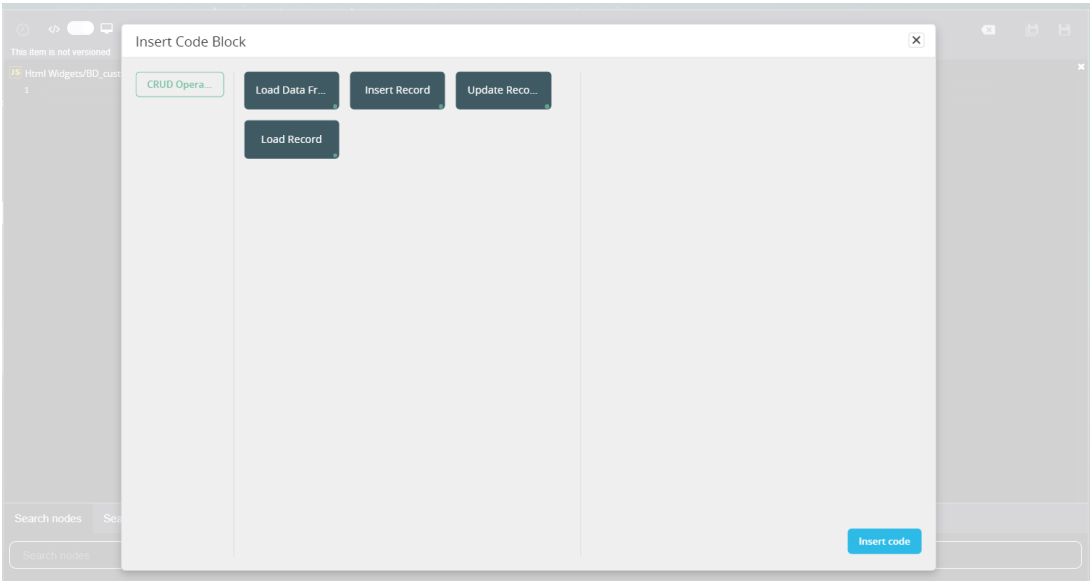
You can use code blocks in attributes of type After generate JS on forms (Advanced tab) or in the Advanced Code Editor.

To insert a code block in an After generate JS field:

1. Place the cursor inside the desired After generate JS, right click and from the contextual menu click **Insert Code Block**.



2. The Insert Code Block page appears.



3. Click on categories and which one contains the code block you want to use.
4. Double click on the desired code block or click on it then click the **Insert code** button. The code from the selected code block is added on a new line at the cursor position in the editor.
5. Modify the inserted code as best suit your needs.

Custom Client-side Functions

The use of custom functions is available within the following entities and JavaScript attributes:

Entity	Attributes
Entity Form	Before Events, After Events
Entity Form Section	After Events, After Section Save
Custom Form	After Generate Events
Entity View	After Generate Js
HtmlWidget	JavaScript

To use a custom function in a JavaScript field, import the client script library by using the `importClientScript` method in the following format:
`ebs.importClientScript("");`

Put the cursor between the two quotation marks (“|”) and press **CTRL+SPACE**. You will be suggested the available client script libraries.

Choose the client script library that defines the custom function that you want to use:

Example:

```
var myLib = ebs.importClientScript("myLibrary");
```

For usability purposes and to avoid calling the wrong methods and attributes, starting with 18.1.9, when creating script libraries and scripts you can use the ‘$m’ mechanism as follows:

- To transform text in entity_name string, type **$m.entity_name** and then press the **TAB** key.
- To transform text in in attribute_name string, type **$m.entity_name.attribute_name** and then press the **TAB** key.
- To transform text in relationship_name string, type **$m.entity_name.relationship_name** and then press the **TAB** key.

NOTEAll entities and their attributes and relationships are available regardless of the current entity.

The TypeScript definition of client script library declares a specific function; therefore, the Monaco editor will suggest the function name after you type the variable name

Example:

```
console.log(myLib.capitalize("aWord"));
```

If you’re using Chrome, open the entity where the code will be run and by using the **Developer tools** > **Console**, check that the custom function returns the expected result.

Example: Print the word “AWord”.

Code execution

When retrieved, the code is transformed using the module pattern:

INPUT

```
var c1 = 1;

function A()
{
    return c1;
}

var c2 = 2;
function B()
{
    var y = function C()
            {
            };
    return c2;
}
```

OUTPUT

```
//# sourceURL=clientScriptLib_myScript.js;

(function(){

    var $export = {};
        var c1 = 1;
         $export.A = function ()
        {
            return c1;
        };
         var c2 = 2;
         $export.B = function ()
        {
            var y = function C()
                {
                };
            return c2;
        };
     return $export;
}());
```

If the module pattern is detected in the INPUT, then no transformation occurs.

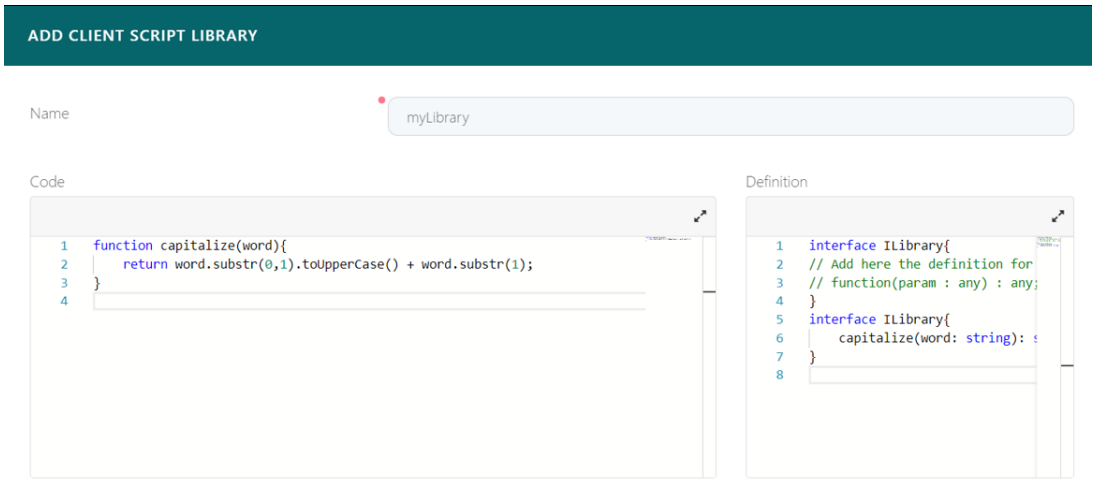
Defining Custom Functions (using Client Script Libraries)

To define a JavaScript function to be used across Innovation Studio, follow these steps:

1. On the menu, click Advanced > Client Script Libraries. The Client Script Libraries List page appears.
2. At the upper-right corner of the page, click the Insert icon. The Add Client Script Library page appears.
3. In the Name field, provide a unique name for the function.
4. In the Code field, provide the actual code that will be executed.

   Code example: capitalizing a word:

```
function capitalize(word){
    return word.substr(0,1).toUpperCase() + word.substr(1);
}
```



For usability purposes and to avoid calling the wrong methods and attributes, when creating script libraries and scripts you can use code snippets. For information on how to use code snippets, see Code Snippets Support for JavaScript.

5. In the Definition field, provide the TypeScript definition of the code you provided in the Code field. For more information on how to write TypeScript declarations, see the TypeScript documentation.

   TypeScript Definition: capitalizing a word

   ```
   interface ILibrary{
       capitalize(word: string): string;
   }
   ```

6. At the upper-right corner of the page, click the Save and close icon. The new function is added to the Client Script Library and it will be displayed in the Client Script Libraries List. For information on how to use custom functions, see How to Use Custom Functions.

You can also import a Client Script Library into another one but make sure that you avoid circular reference.

Fluent Queries

Fluent queries allow you to run database queries in your server automation scripts using an SQL-like fluent interface. Intelligent code completion is available in the code editor both for the query inputs and for the result sets.

IMPORTANT!

- Fluent queries can only be used in administrative context. See the Server SDK Reference Guide for information on how to temporarily change the transaction context.

- Result sets include only the columns specified in the select statements. Lookup fields are not automatically expanded.
- Data ownership is per organization.
- Date and datetime values are not returned as strings. The wrapper for datetime values (JsDateTime) stores the date as UTC and provides utility methods for data manipulation.

How to Execute a Fluent Query **528**
Working with Fluent Query Result Sets **535**

How to Execute a Fluent Query

To execute a fluent query in a server automation script:

1. Use the `server.query.getAlias` method to define an entity alias.
2. Use the `server.query.from` method to run the query on the desired entity.
3. Use the `.selectColumns` method to select the returned attributes.
4. Use the `.execute()` method to run the query.

```
var A = server.query.getAlias('Account');

var myFluentQuery = server.query.from('Account', A)
    .selectColumns (A.Name, A.Email)
    .execute();

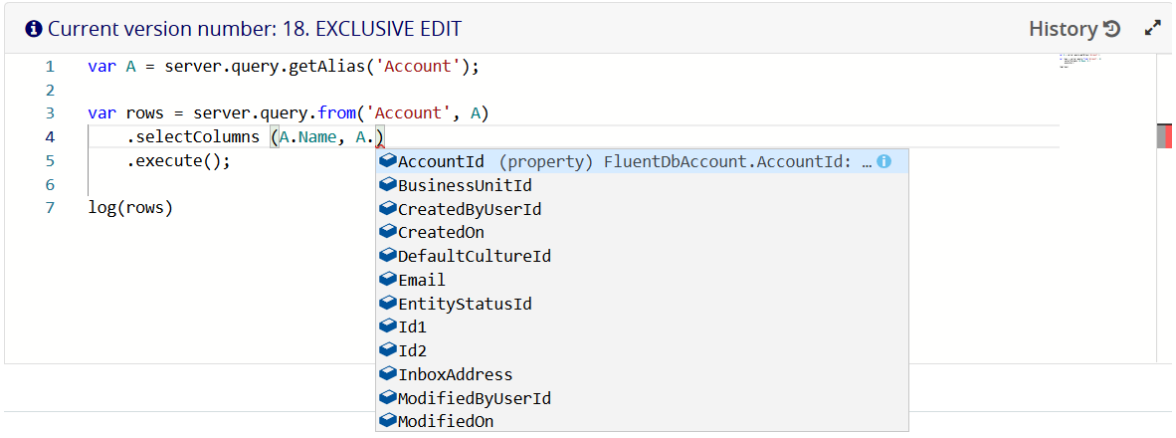
log(myFluentQuery)
```

The code above will log an output similar to the following in the `trace_roll.log` file:

```
-[START]----------------------------------------
```

```
Timestamp: 7/1/2020 5:01:51 PM
 Message: INFO [CID=200524bb-8d28-4079-80c3-a003a9ecad7d] [
    {
        "values": {
            "A_Name": "Jane Doe",
            "A_Email": "janedoe@fintechos.com"
        }
    },
    {
        "values": {
            "A_Name": "Andrew Jones",
            "A_Email": "adrew.jones@fintechos.com"
        }
    },
    {
        "values": {
            "A_Name": "John Doe",
            "A_Email": "john.doe@fintechos.com"
        }
    }
]
 Severity: Information
-[END]----------------------------------------
```

The entity alias definition enables intelligent code completion for the corresponding entity attributes:



Comparison Operators

The following comparison operators are supported in fluent queries: `equals` (`eq`), `notEquals` (`neq`), `greaterThan` (`gt`), `greaterThanOrEquals` (`gt`), `lessThan` (`lt`), and `lessThanOrEquals` (`lte`)

Logical Operators

The following logical operators are supported in fluent queries: `and`, `or`, `andNot`, and `orNot`.

Inner Joins

Use the `.innerJoin` and `.on` methods to define inner join clauses for the queried entities.

```
var O = server.query.getAlias('optionset');
var A = server.query.getAlias('attribute');


var myFluentQuery = server.query.from('attribute', A)
    .innerJoin('optionset', O)
        .on(A.OptionSetId.eq(O.OptionSetId))
    .top(5)
    .orderBy(A.Name)
    .execute()
```

Left Joins

Use the `.leftJoin` and `.on` methods to define left outer join clauses for the queried entities.

```
var O = server.query.getAlias('optionset');
var A = server.query.getAlias('attribute');

var myFluentQuery = server.query.from('attribute', A)
    .leftJoin('optionset', O)
        .on(A.OptionSetId.eq(O.OptionSetId))
    .top(5)
    .selectColumns(A.Name, O.DisplayName)
    .orderBy(A.Name)
    .execute();
```

Attribute Aliases (Projections)

To define aliases for the queried attributes, use the `.selectProjection` method instead of `.selectColumns` and the `.executeAndMap` method instead of `.execute`.

Projections are useful to customize the result set field names when different entities have similar attribute names or when attribute names are not expressive (see Working with Fluent Query Result Sets for more details about fluent query outputs).

Example: Return customized field names in a fluent query result set

In this example, we return information about 5 random entity-attribute pairs.

```
var E = server.query.getAlias('entity');
var A = server.query.getAlias('attribute');

var P; //projection alias

var rows = server.query.from('entity', E)
    .innerJoin('attribute', A)
        .on(E.EntityId.eq(A.EntityId))
    .top(5)
    .selectProjection( P =
    {
        EntityName : E.Name,
        AttributeName : A.Name,
        Type : A.AttributeType
    })
    .executeAndMap(P);

log(rows)
```

The result set for the above fluent query will include the customized **EntityName**, **AttributeName**, and **Type** field names.

```
-[START]----------------------------------------
Timestamp: 7/2/2020 2:00:55 PM
 Message: INFO [CID=200524bb-8d28-4079-80c3-a003a9ecad7d] [
    {
        "EntityName": "Account",
        "AttributeName": "Phone",
        "Type": "0a39db15-7634-4af3-8bd8-004fcf27e8a6"
    },
    {
        "EntityName": "Account",
        "AttributeName": "Accountid",
        "Type": "ddce8347-794d-4a8d-b9d0-42437f653ae4"
    },
```

```
    {
        "EntityName": "Account",
        "AttributeName": "Email",
        "Type": "0a39db15-7634-4af3-8bd8-004fcf27e8a6"
    },
    {
        "EntityName": "Account",
        "AttributeName": "modifiedOn",
        "Type": "2e33740e-5026-43b5-919c-cc2d422c280f"
    },
    {
        "EntityName": "Account",
        "AttributeName": "Id2",
        "Type": "0a39db15-7634-4af3-8bd8-004fcf27e8a6"
    }
]
 Severity: Information
-[END]--------------------------------------
```

Where Clauses

Where clauses are implemented using the `.where`, `.wherenot`, `.andWhere`, `.orWhere`, `.andWhereNot`, and `.orWhereNot` methods.

Example: Fluent query with multiple where clauses

```
var F = server.query.getAlias('FinChartItemValue');

var rows = server.query.from('FinChartItemValue', F)
    .top(5)
    .where(F.Percent.gte(100).and(F.Percent.lte(200)))
    .orWhere(F.Percent.gte(300).and(F.Percent.lte(400)))
    .execute();
```

Example: Fluent query with comparison operators

```
var F = server.query.getAlias('FinChartItemValue');

var rows = server.query.from('FinChartItemValue', F)
    .top(5)
    .where(F.Percent.isNull)
```

```
    .execute();

//equivalent query

var rows = server.query.from('FinChartItemValue', F)
    .top(5)
    .where(F.Percent.equals(null))
    .execute();
```

Aggregate Functions

To define aggregate functions on a set of values from the result set, use the `.getCountAlias`, `.getSumAlias`, `.getMaxAlias`, and `.getMinAlias` methods of the `server.query` property.

Example: Get attribute aggregates for 5 random entities.

In this example, we get the number of attributes, maximum attribute length, minimum attribute length, and total length of all attributes for 5 random entities.

```
var E = server.query.getAlias('entity');
var A = server.query.getAlias('attribute');

var Count = server.query.getCountAlias();
var SumLength = server.query.getSumAlias(A.Length);
var MaxLength = server.query.getMaxAlias(A.Length);
var MinLength = server.query.getMinAlias(A.Length);

var P; //projection alias

var rows = server.query.from('entity', E)
    .innerJoin('attribute', A)
        .on(E.EntityId.eq(A.EntityId))
    .top(5)
    .where(E.Name.startsWith('a'))
    .selectProjection( P =
    {
        'Entity Name' : E.Name,
        'Number of attributes' : Count,
```

```
        'Total attributes\' lengths': SumLength,
        'Maximum attribute length' : MaxLength,
        'Minimum attribute length' : MinLength
    })
    .executeAndMap(P);

log(rows);
```

The code above will log an output similar to the following in the `trace_roll.log` file:

```
-[START]----------------------------------------
Timestamp: 7/2/2020 11:21:28 AM
 Message: INFO [CID=200524bb-8d28-4079-80c3-a003a9ecad7d] [
    {
        "Entity Name": "Account",
        "Number of attributes": 15,
        "Total attributes legth": 1638,
        "Maximum attribute length": 500,
        "Minimum attribute length": 64
    },
    {
        "Entity Name": "action",
        "Number of attributes": 9,
        "Total attributes legth": 3200,
        "Maximum attribute length": 3000,
        "Minimum attribute length": 0
    },
    {
        "Entity Name": "actiongroup",
        "Number of attributes": 6,
        "Total attributes legth": 200,
        "Maximum attribute length": 200,
        "Minimum attribute length": 0
    },
    {
        "Entity Name": "ActionXSecurityRole",
        "Number of attributes": 3,
        "Total attributes legth": null,
        "Maximum attribute length": null,
        "Minimum attribute length": null
    },
    {
        "Entity Name": "applicationLanguage",
```

```
        "Number of attributes": 11,
        "Total attributes legth": 213,
        "Maximum attribute length": 100,
        "Minimum attribute length": 1
    }
]
 Severity: Information
-[END]----------------------------------------
```

Working with Fluent Query Result Sets

Use the `.field()` method to get field values from a row in the result set.

```
var E = server.query.getAlias('entity');

var rows = server.query.from('entity', E)
    .top(5)
    .execute();

let output = '\n\nThe following entities were found:\n';

rows.map(function(r)
{
    output += (' ' + r.field(E.Name) + '\n')
});

log(output);
```

The code above will log an output similar to the following in the `trace_roll.log` file:

```
-[START]----------------------------------------
Timestamp: 7/2/2020 1:15:34 PM
 Message: INFO [CID=200524bb-8d28-4079-80c3-a003a9ecad7d]

The following entities were found:
 entityBWTransitionActionGroup
 SystemUserSecurityRole
 BW
 optionset
```

```
 lookupCorrelationAttribute

 Severity: Information
-[END]--------------------------------------
```

Map result sets to POCO objects

Use the `.executeAndMap()` method to map a fluent query result set to an entity alias. This creates a plain old CLR object (POCO) which allows you to access field values as object properties, instead of using the `.field()` method.

```
var E = server.query.getAlias('entity');

var rows = server.query.from('entity', E)
    .top(5)
    .executeAndMap(E);

let output = '\n\nThe following entities were found:\n';

rows.map(function(r)
{
    output += (' ' + r.Name) + '\n')
});

log(output);
```

To map a fluent query result set to multiple entity aliases, use the `.executeAndMapComplex()` method.

```
var E = server.query.getAlias('entity');
var A = server.query.getAlias('attribute');

var rows = server.query.from('entity', E)
    .innerJoin('attribute', A)
        .on(E.EntityId.eq(A.EntityId))
    .top(20)
    .selectColumns(
        E.Name,
        A.Name)
    .executeAndMapComplex({ entity : E,  attribute : A});

rows.map(function(r)
{
    var entityName = r.entity.Name;
```

```
    var attrName   = r.attribute.Name;
    // do
});
```

Sequencers

This feature makes it possible to add a sequence of numbering to a digital document.

To configure follow these steps,

1. Open the Innovation Studio, select the Advanced menu item and left-click on the Sequencers menu.
2. Click the ''Insert'' button to add a new configuration or open an existing one by double-clicking. To delete, select the sequencer and click on the ''Delete'' button on the right side of the screen.

3. Fill in the following:

Fields	Required	Data type	Description
Name	Yes	Text	Insert a name for the sequencer.
Code	Yes	Text	Insert a code of the sequencer.
Prefix	No	Text	Insert a prefix for the sequencer to take intro account. For example, set the prefix AAA, and the sequence will be AAA1, AAA2 and so on.
Padding	No	Text	This is the number of characters for a sequence. For example, set the padding nr. 4, the sequence will be 0001, then 0002 and so on.
RangeMin	No	Text	Insert the minimum range of characters to be set in the sequence.
RangeMax	No	Text	Insert the maximum range of characters to be set in the sequence.
Number	No	Text	This is the number of the sequence.
Start date	No	Date	Insert the start date since when the sequencer will be available.
End date	No	Date	Insert an end date for when the sequencer will not be available.
Filter JS	No	JS	

4. Click the ''Save and reload'' button.

How to add items to the sequencer

The term ''items'' means the actual character.

1. To add an item, fill in the following fields:

Fields	Data type	Description
isUsed	Bool	Tick if the bool is true, the item will be used in the sequencer. By leaving the bool empty, false, the system will skip the number/ name.
SequencerID	Text	
Number	Numeric	Insert the number.
Name	Text	Insert the name.

2. Click the ''Save and reload'' button. Repeat of as many times as needed.

How to call the Sequencer

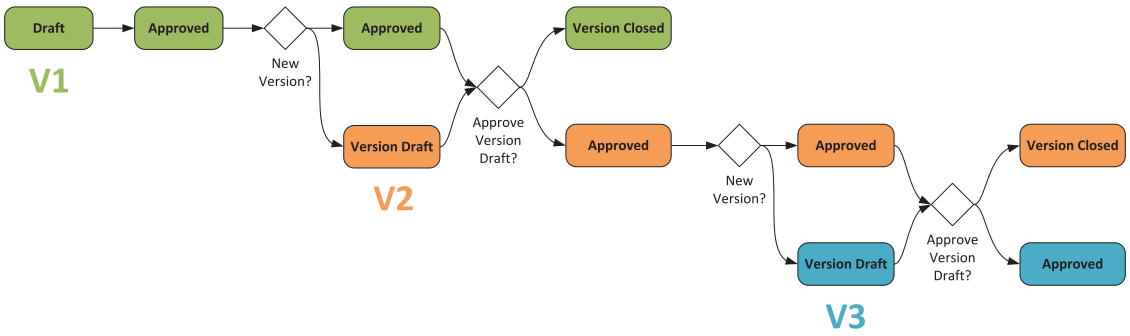
It is possible to call the sequencer using the automation script. For more information, see getSequenceNumber.

Entity versioning

Entity versioning allows you to implement a record update process that is certified and documented. The versioning process uses the Business Workflows Processor to assign different states to a record indicating if it is:

- a draft that hasn't been approved yet
- the currently active version of the record
- a discontinued prior version
- a substitute version that is a candidate to replace the currently active version.

The entity versioning process is illustrated below.



1. The original record starts out in a **Draft** state.
2. Once the draft is validated, the record goes into the **Approved** state, indicating that this it is the currently active version of the record.

   > **NOTE**
   >
   > Only one version of the record can be active (in the Approved) state at one time.

3. To modify an active record, you need to create a secondary version of it. The secondary version starts out in the **Version Draft** state, while the original record remains in the **Approved** state.
4. If the version draft is approved, then the original record transitions into the **Version Closed** state and the secondary version becomes the **Approved** currently active entity record.
5. The process can then be repeated from step 3 whenever a new version of the record is necessary.

Entity versioning is useful when you need a thorough validation and tracking of your record updates, such as when storing information about security roles or customer documents.

How to Set Up Versioning on an Entity

IMPORTANT!
Entity versioning requires the Digital Foundation deployment package. Make sure you have the Digital Foundation deployment package installed on your FintechOS instance before attempting to version an entity. The Digital Foundation deployment package is included in all Fintech System Packs.

1 Configure the versioned entity's attributes

On the entity that you wish to version, add the following attributes:

- **masterAttributeId** - lookup attribute that self-references the versioned entity. You need to also change the default relationship name to a custom value.
- **referencedAttributeId** - lookup attribute that self-references the versioned entity. You need to also change the default relationship name to a custom value.
- **attributeVersion** - Whole Number
- **attributeVersionDate** - Date Time
- **isLastVersion** - Boolean

2 Configure the versioned entity's data events

On the entity that you wish to version, create the following event-driven server side scripts:

Name	Event	Stage	Code
<VersionedEntity>_After_Insert	Insert	After	`var vh=importLibrary ("FTOS.VersioningHelper"); context = vh.VersioningHelper.RunAfterInsert(context);`

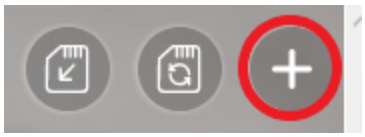
Name	Event	Stage	Code
<VersionedEntity>_After_Update	Update	After	`var vh=importLibrary ("FTOS.VersioningHelper"); context = vh.VersioningHelper.RunAfterUpdate(context);`
<VersionedEntity>_Before_Insert	Insert	Before	`var vh=importLibrary ("FTOS.VersioningHelper"); context = vh.VersioningHelper.RunBeforeInsert(context);`

3 Set Up the Versioning Button

On the form or step where you wish to be able to trigger the record versioning, add the following code in the **After Events** tab of the **Advanced** section:

```
var vh = ebs.importClientScript("FTOS_VersioningHelper");
vh.createVersioningContext(formScope);
```

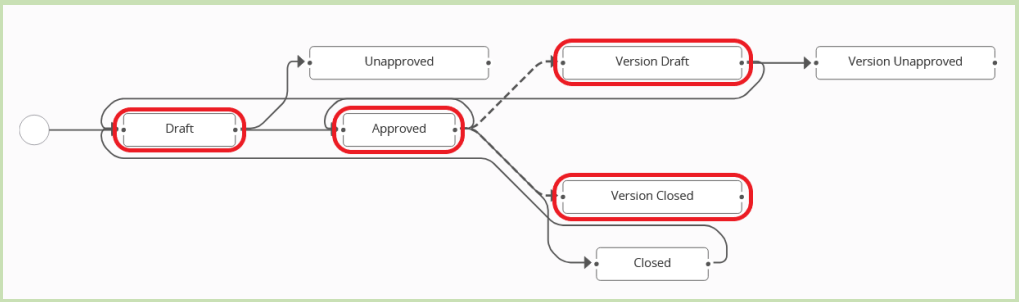
This will display the versioning button, which allows you to create a new version of the currently displayed record.



4 Set Up the Entity Workflow

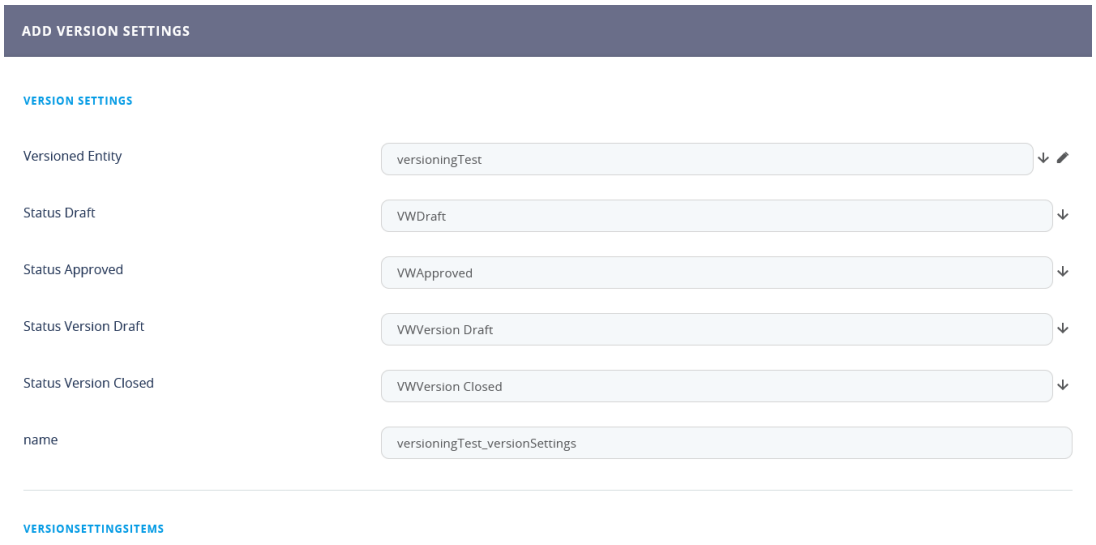
Assign a business workflow to the versioned entity. You may use any workflow that has at least 4 states which you can map to the Draft, Approved, Version Draft, and Version Closed statuses.

HINT
FTOS_VersioningWorkflow is an example of a preset business workflow that meets the above criteria.



5 Configure the Version Settings

1. Go to **Main Menu** > **Admin** > **Entity Versioning** > **Version Settings**.
2. Click the **Insert** button at the top right corner of the page to create a new set of version settings.
3. In the Add Version Settings window, enter the following information:
    - Versioned Entity - Name of the entity you are versioning.
    - Status Draft - State in the entity's business workflow you wish to map to the Draft status.
    - Status Approved - State in the entity's business workflow you wish to map to the Approved status.
    - Status Version Draft - State in the entity's business workflow you wish to map to the Version Draft status.
    - Status Version Closed - State in the entity's business workflow you wish to map to the Version Closed status.
    - Name - Assign a name to the set of version settings.

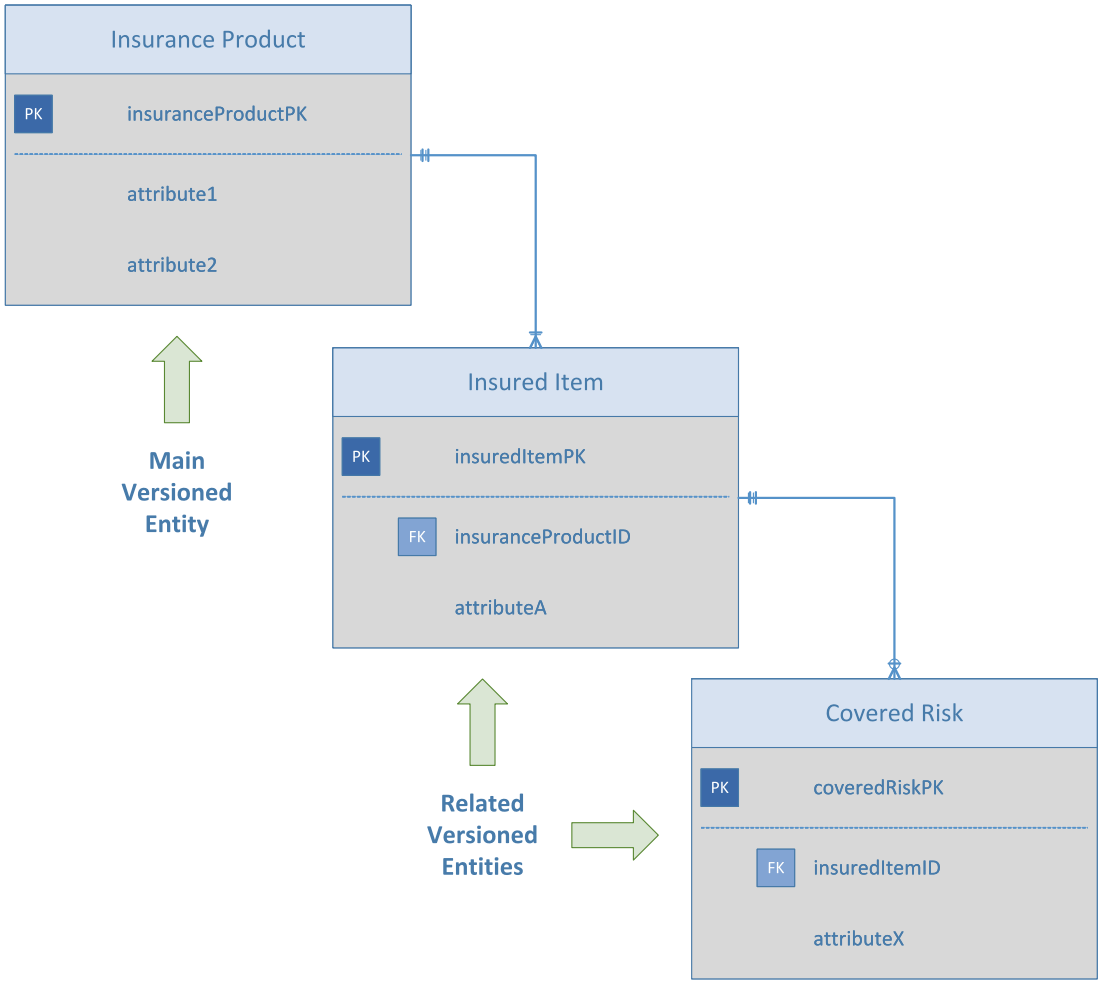


4. Click the **Save and Reload** button at the top right corner of the page.

(Optional) Set Up Related Versioned Entities

When you version an entity record, you may also want to automatically version some of its related entity records. These records belong to entities that have lookup attributes pointing to the versioned entity either directly or indirectly through a stream of intermediate related entities.

This allows you to propagate your versioning to any relevant dependencies. For instance, you may have an insurance product record with a number of related insured items records, each of which has multiple related covered risks records. When you create a new version of an insurance product, you will also be able to create new versions of all its insured items and all of their covered risks at the same time.

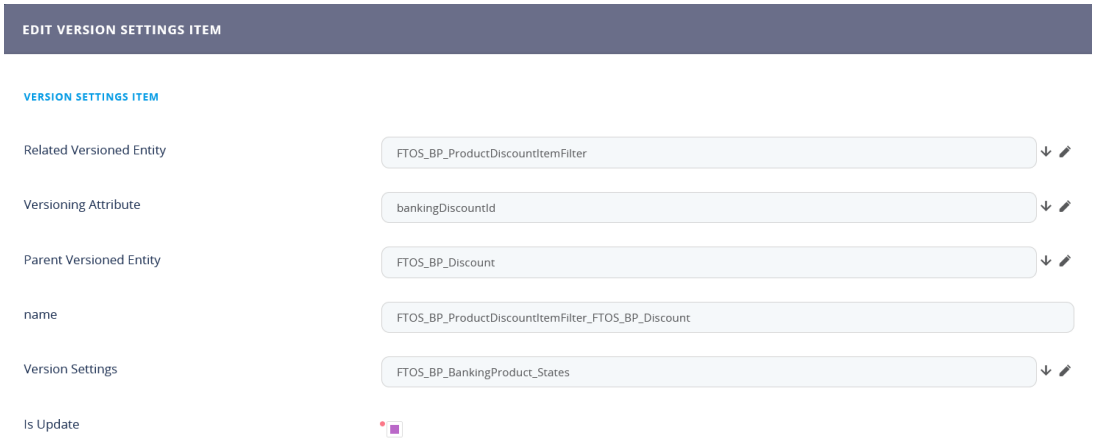


To add a related versioned entity, in the Version Settings window of your versioned entity:

1. Scroll down to the **Version Settings Items** grid and click the **Insert** button.
2. Fill in the Version Settings Item details as follows:
    - Related Version Entity - Name of the related versioned entity.
    - Versioning Attribute - Lookup attribute pointing to the main versioned entity (either directly or indirectly through a stream of intermediate related entities).

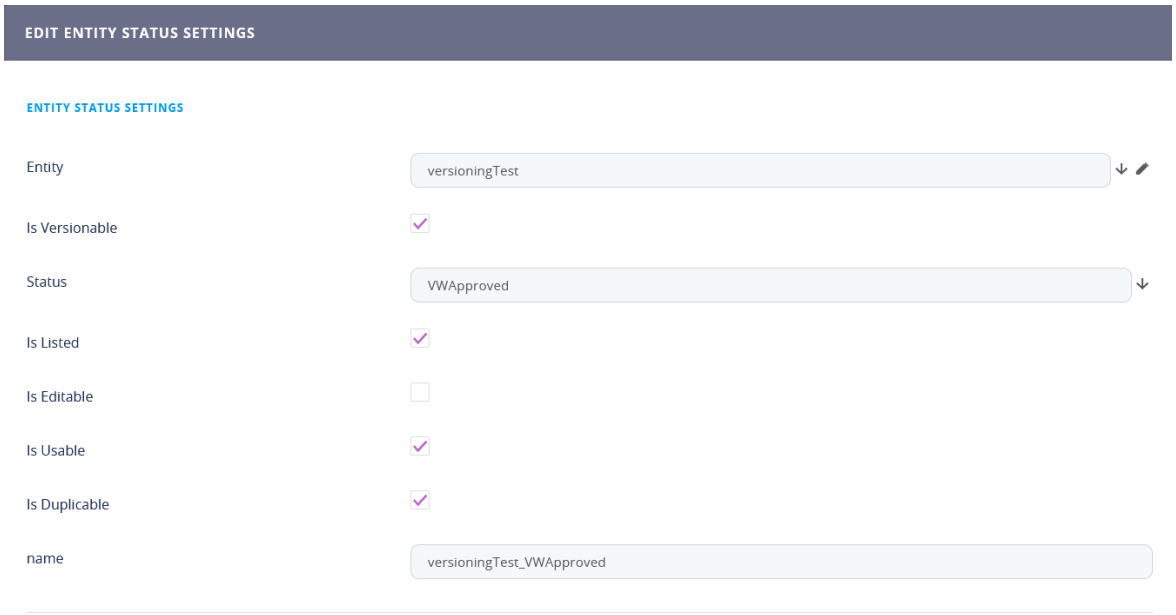
- Parent Versioned Entity - If the versioning attribute points to the main versioned entity (is linked directly), leave empty. Otherwise (if it is linked indirectly), select the entity the versioning attribute points to.
- Name - Enter a name for the version settings item.
- Version Settings - Select the name you assigned to the set of version settings.
- Is Update - Select if you want the related entity to be versioned together with the main entity.

3. Click the **Save and Reload** button at the top right corner of the screen.
4. Repeat for any additional related versioned entities.



6 Configure the Entity Status Settings

Go to **Main Menu** > **Admin** > **Entity Versioning** > **Entity Status Settings** and insert settings for each of the Draft, Approved, Version Draft, and Version Closed business states of the versioned entity:



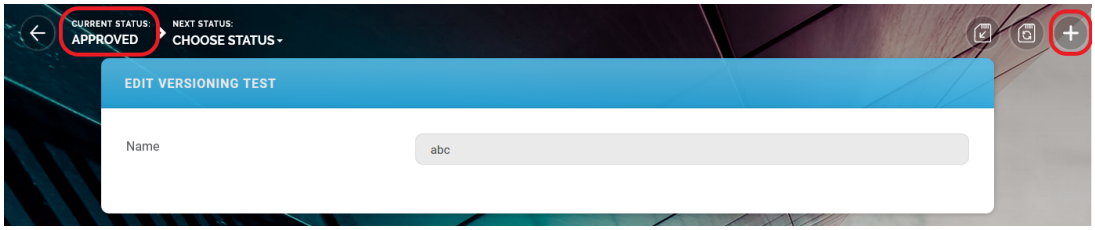
- Entity - Name of the versioned entity.
- Is Versionable - Indicates if you can trigger the versioning when a record is in this status (typically, it's the Approved status).
- Status - State in the versioned entity's business flow you mapped to either the Draft, Approved, Version Draft, and Version Closed statuses.
- Is Listed - Label you can use for customization purposes. Doesn't impact entity behavior.
- Is Editable - Indicates if you can edit the record when it is in the current status. For instance, you may want to be able to edit a record that is in Draft or Version Draft status, but avoid editing a record that has been archived (is in the Version Closed status).
- Is Usable - Label you can use for customization purposes. Doesn't impact entity behavior.

- Is Duplicable - Label you can use for customization purposes. Doesn't impact entity behavior.
- Name - Assign a name to the entity status settings.

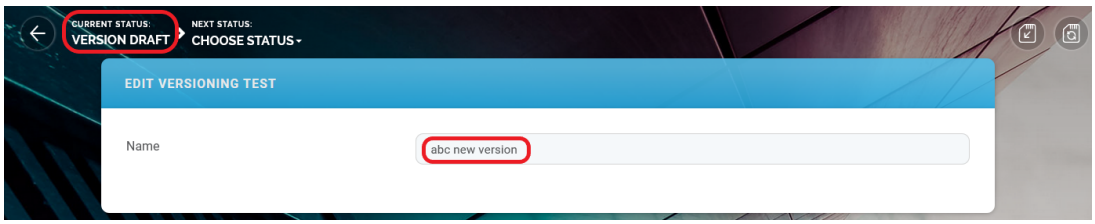
How to Version an Entity Record

After you have set up versioning on an entity, you can start versioning its individual records.

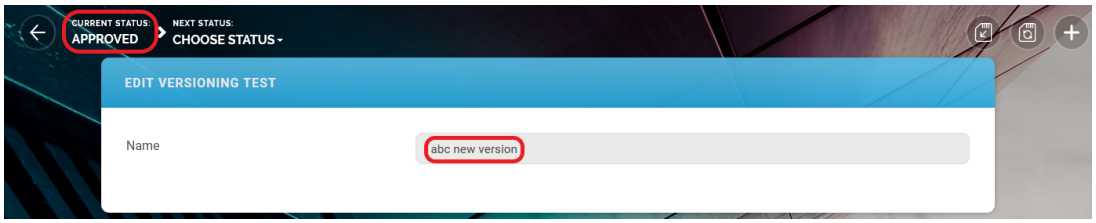
1. When a record reaches a business state that is versionable (see " 6 Configure the Entity Status Settings" on page 546), the versioning button will be displayed on the form or step you have set up accordingly (see " 3 Set Up the Versioning Button" on page 542).

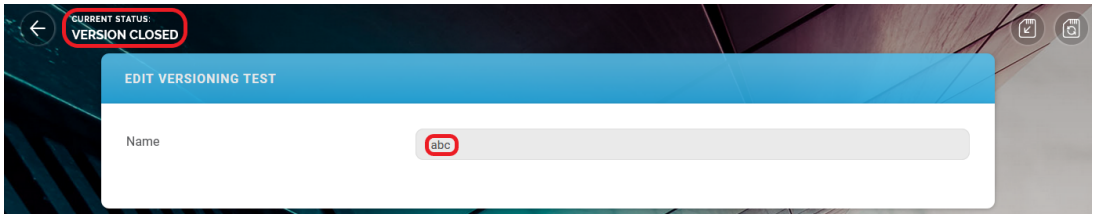


2. If you click the versioning button, a new record version will be created and opened in the Version Draft status. By default, the text "new version" will be appended to the record's primary attribute.



3. If you approve the version draft, the new version becomes the active record and the original record automatically transitions into the Version Closed status.

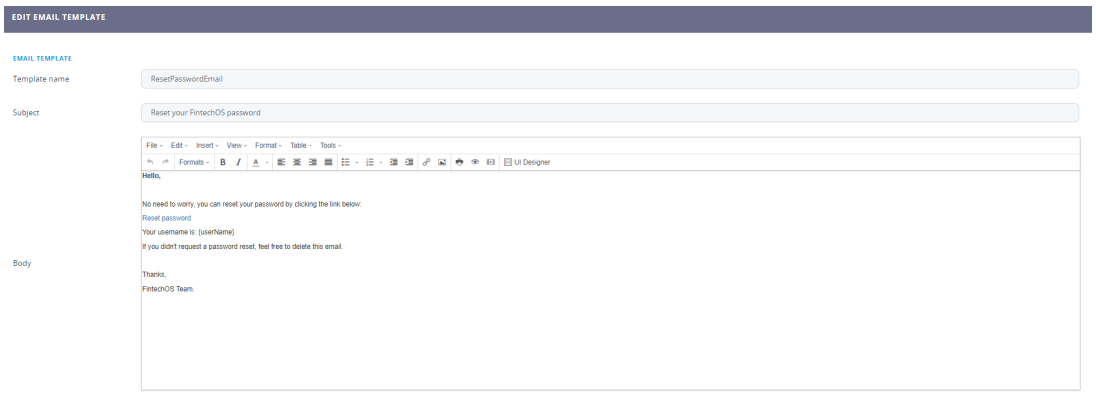




Email Templates

This feature helps the users to build creative emails for campaigns to promote or remind the clients about a new element of the company.

1. Open the main menu, select the **Admin** menu item and click on the **Email templates**.
2. Click the ''Insert'' button to add a new template, to delete, select the template and click on the ''Delete'' button on the right side of the screen.
3. To create a new one, fill in the following:



Field	Data type	Description
Template Name	Text	Type a name for the template.
Subject	Text	Type a subject to be used in the email.
Body		Design the body of the email. For more information, see "UI Designer" on page 297.

4. Click the ''Save and reload'' button.

Digital Frontends

Innovation Studio enables you to define every interaction that your business has with your internal team as well as with the customers. Broadly defined, digital frontends represent your user experience.

Properly defined digital frontends allow you to keep customers happy and loyal while ensuring higher efficiency within your organization.

This section covers the following topics:

Digital Experience Portals **552**

- Customizing the Login and Home Page 554
- Using Custom Theme 556
- Using Custom Icons 560
- Setting Sticky Header 564
- Grouping Entities in Menu Items 564
- Show Tooltips (for users) 566
- Creating HTML Widgets 567
- Creating Dashboards 568
- Editing Dashboards 578
- Using Portal Profiles 578
- Configuring the Digital Experience Portal 594
- Keyboard Shortcuts 603

Anonymous Frontends **605**

- Is it secure to expose digital journeys to unauthenticated users? 605
- Setting B2C Environment 606
- Overriding Default Save on Journeys 613
- Serving User Journeys in a Specific Language 615
- Manage Style Sheets for B2C User Journeys 616

Digital Experience Portals

Innovation Studio provides various ways for streamlining the experience of your business users by customizing the **Digital Experience Portals** in accordance to their needs.

The following customization features are available:

- Customize the login and home page
- Use a custom UI theme
- Use custom icons
- Visual branding
- Add digital journey sticky header
- Customize dashboards using widgets
- Show tooltips (if allowed by the Portal customization).

With the use of portal profiles, you can also customize **Digital Experience Portals** with specific elements like background image, menu items, dashboards, or specific values for system parameters. For more information on portal profiles, see "Using Portal Profiles" on page 578.

Theme Support

The portals come with multiple options for customizing the layout by uploading a background image, generating a color palette, or using a floating style, global dashboard, or shortcuts on the homepage.

Custom Theme Support

Custom theme support is useful for streamlining, automating, and merging deployments with FintechOS upgrades.

Enhanced Security

The portals facilitate adaptive user interface based on role (designer, portal), with available apps displayed accordingly, as well as dedicated data form for self-service user profile management.

Visual Branding Support

The portals come with the default color palette inherited from FintechOS brand. However, you can switch it over to custom colors and logo or pick one of the available color palettes.

Enriched Dashboards

There are extensive types of elements supported within user dashboards, such as KPIs, HTML widgets, charts and Fincharts, views, and Power Bi reports. Security wise, all such elements may be restricted to certain user roles.

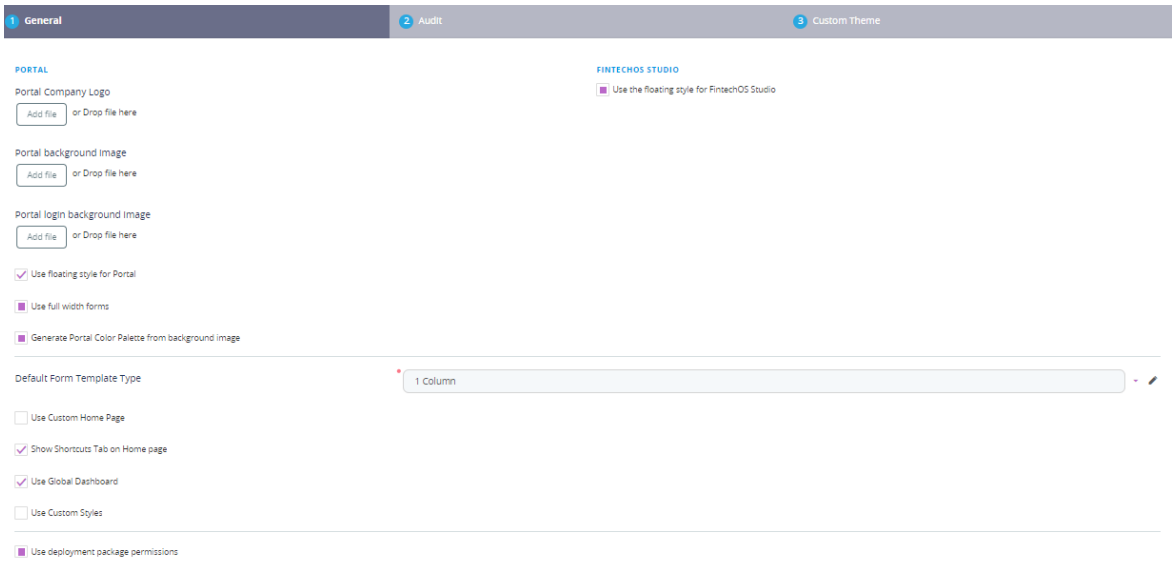
Native Analytics

The portals feature a powerful library of charts for displaying business information in a compelling and visual way, serving diverse and complex analytics needs.

The figure below shows an example of how a **Digital Experience Portal** may be customized:

Customizing the Login and Home Page

You can customize the **Digital Experience Portals** login and home page, from the menu, click **Admin** > **Settings**.



Upload your own company logo, background image, and login background image to be shown in the **Digital Experience Portal**.

To show the Portal UI container in full screen width ticki the **Use full width forms** checkbox.

You can also add custom actions (custom flows) on the home page by ticking the **Use Custom Home Page** checkbox and selecting the desired custom flow. A new custom flow can be inserted directly in the **Settings** page, by clicking the arrow next to the **Use Custom Home Page** field. The list of available custom flows will be displayed. Click the Insert icon and provide the custom flow information. For more information on custom flows, see Creating Form Driven Journeys.

Choose if the shortcuts are displayed as the first tab on the homepage by selecting the checkbox next to the **Show Shortcuts Tabs on Home page** label. Unselect the checkbox to disable the user shortcuts and not have them displayed as the first tab on home page, but available through a new icon on the top menu bar.

Multiple dashboards can be added to the Portal UI if the **Show on Home page** option is selected on the dashboards configuration page.

To force the default Dashboard, named Main Dashboard, to be displayed first on the user interface, select the **Use Global Dashboard** checkbox.

By default, the color palette of the Digital Experience Portal UI is inherited from the FintechOS brand. To apply your own brand, follow these steps:

1. Select the **Portal background image**.
2. Select the **Portal login background image**.
3. Set the transparent top menu bar and a floating feeling for all elements by ticking the **Use floating style for Portal** checkbox.

   NOTEThe global dashboard has higher priority than the floating dashboard; therefore, if both checkboxes are selected, only the global dashboard will be visible in the user interface.

4. Apply a color palette based on the background image by ticking the **Generate Portal Color Palette from background image** checkbox and thus restrict portal users changing the color palette.

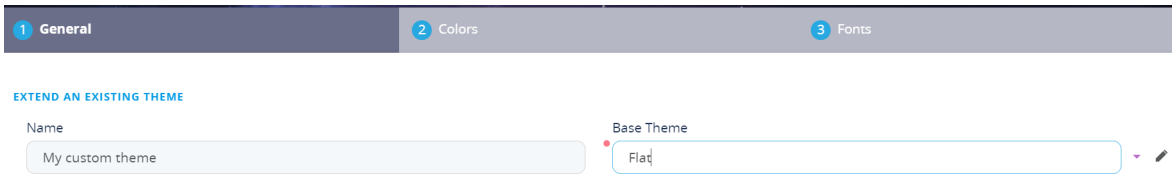
Using Custom Theme

NOTETo change the custom theme styles from a single place, put the stylesheet (CSS) file which contains the custom visual design and layout of the user interface on the server where the FintechOS installation package is located, within the **custom** folder.

This section walks you through the steps that you need to follow to use a custom theme for the Digital Experience Portal:

STEP 1. Create custom theme

To create a custom theme, from the menu, click **Admin** > **Settings** . The Portal UI configuration page is displayed on the **General** tab.

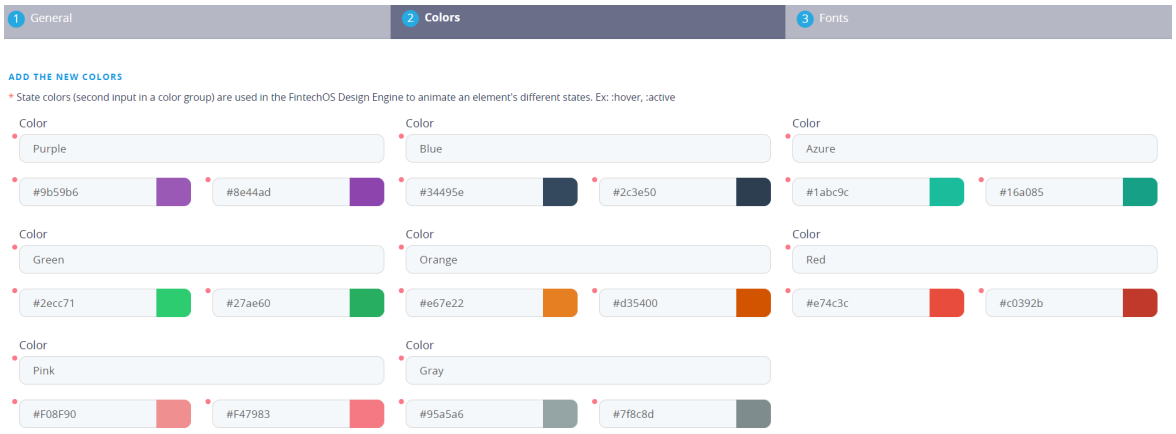


Click the **Custom Theme** tab. The list of custom themes you defined will be displayed. If you haven't created any custom themes, the list will be empty. Click the **Insert** button. The custom theme configuration page will be displayed on the **General** tab.

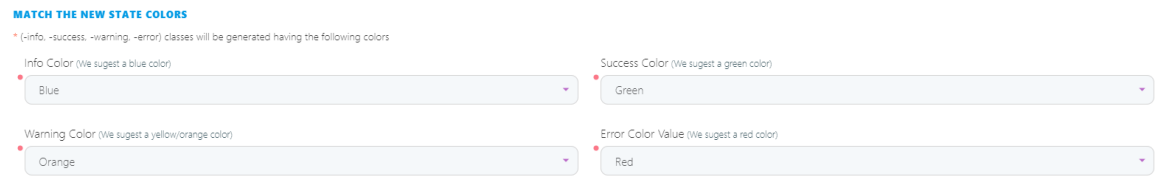
Provide a name for the custom theme and select the base theme.

Click the **Colors** tab to access extensive settings for your custom theme.

In the **Add the new colors** section, you can customize 16 colors from the base theme previously selected by providing the desired color hexa values.



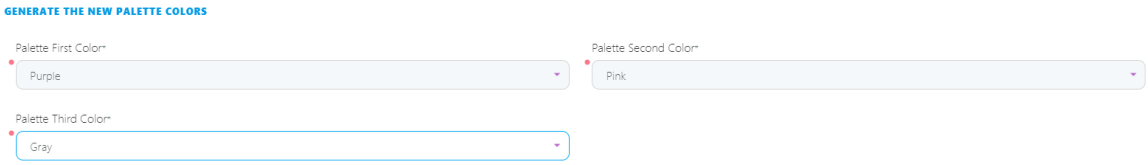
In the **Match the new state colors** section, set the matching colors for different system notification messages. Read the on screen information on the recommended colors.



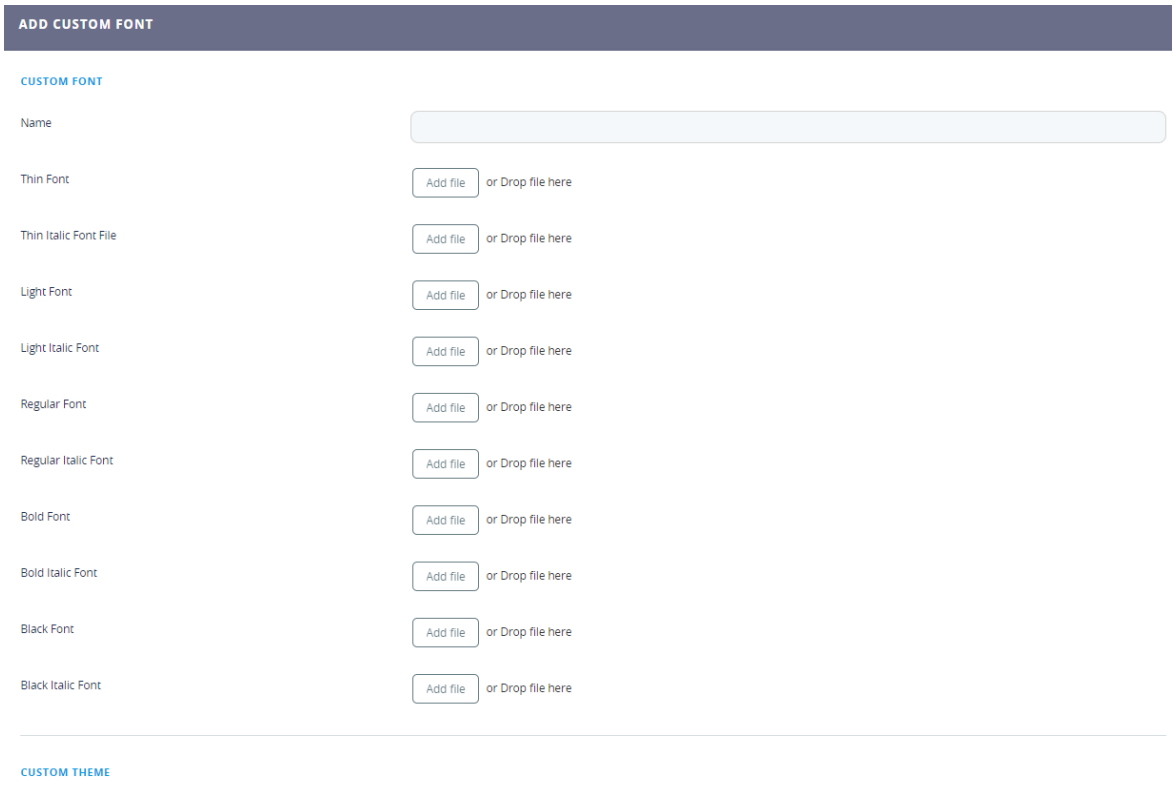
Once you select the matching colors for the different system notifications, a new section is displayed at the bottom of the configuration page and lists the new css color classes.

The colors listed in the **New css color classes** section are non-editable. If you want to modify them, you can do so from the **Match the new state colors** section.

In the **Generate the new palette colors** section, select three colors of the 16 colors customized in the **Add new colors** section. Based on these colors, a palette will be generated to the new custom theme.



You can also overwrite system fonts with your preferred font styles. To do so, click the **Fonts** tab. Click the drop-down arrow next to the **Custom Font** field. A pop-up will be displayed listing all custom fonts you have created. Double-click the desired custom font to be used. If no custom fonts have been defined, click the **Insert** button. The **Add Custom Font** page will be displayed.



In the **Name** field, provide a unique name for your custom font and based on the system fonts that you want to customize, drag and drop the new fonts as needed.

Should you use web safe fonts, we recommend you to choose Google Fonts. For UI consistency, use fonts from the same font family.

Save the new custom font and double-click it. Save the new custom theme by clicking one of the save icons at the top-right corner of the page. To use the new custom theme, make it default.

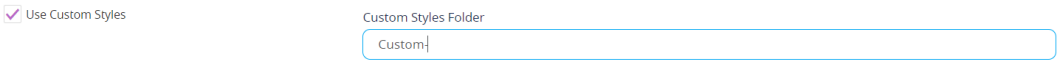
STEP 2. Set default custom theme

If you want to apply a custom theme to the Portal UI, on the server where you have installed FintechOS, open the **web.config** file and add the following setting:

```
<appSettings>
        <add key="feature.customTheme" value="CustomTheme" />
        ...
</appSettings>
```

STEP 3. Apply custom theme to the Portal UI

1. From the menu, click **Admin** > **Settings** . The Portal UI configuration page will be displayed on the **General** tab.
2. Select the **Use Custom Styles** checkbox and type the name of the folder.



NOTE
If you're using a custom theme for the Portal UI, in the top-right **Settings** menu, the options to choose theme and palette will no longer be available.

Using Custom Icons

Personalizing your **Digital Experience Portal** icons is a great way to identify the Portal with your company's brand, making it uniquely yours.

This topic covers everything you need to know in order to use custom icons.

What files do you need?

The following is a recommended best practice for structuring a file:

```
+-- customIcon
        |   +-- css
        |       +-- customIcon.css
        |   +-- font
        |       +-- customIcon.eot/ttf/woff/woff2/svg
        |   +-- customIcon.js
```

You will need your font files (e.g., .eot, .woff, .woff2, etc.). Once you have the font files, import them in your css file and make sure to add the icon classes.

IMPORTANT!FintechOS does not support spinning icons or any sort of animated icons if the animation is done with a separate class.

.css file

```
@font-face {
        font-family: 'customIcon';
        src: url('../font/customIcon.eot?48188603');
        src: url('../font/customIcon.eot?48188603#iefix') format
('embedded-opentype'),
        url('../font/customIcon.svg?48188603#fontello') format
('svg');
        font-weight: normal;
        font-style: normal;
}
[class^="customIcon-"]:before, [class*=" customIcon-"]:before {
        font-family: "customIcon";
        font-style: normal;
        font-weight: normal;
        speak: none;
        display: inline-block;
        text-decoration: inherit;
        width: 1em;
        margin-right: .2em;
        text-align: center;

        /* For safety - reset parent styles, that can break glyph
codes*/
        font-variant: normal;
        text-transform: none;

        /* fix buttons height, for twitter bootstrap */
        line-height: 1em;

        /* Animation center compensation - margins should be
symmetric */
        /* remove if not needed */
        margin-left: .2em;

        /* you can be more comfortable with increased icons size */
        /* font-size: 120%; */

        /* Uncomment for 3D effect */
        /* text-shadow: 1px 1px 1px rgba(127, 127, 127, 0.3); */
}

.customIcon-happy:before { content: '\e800'; } /* ' ' */
.customIcon-wink:before { content: '\e801'; } /* ' ' */
```

```
.customIcon-unhappy:before { content: '\e802'; } /* ' ' */
.customIcon-sleep:before { content: '\e803'; } /* ' ' */
.customIcon-thumbsup:before { content: '\e804'; } /* ' ' */
.customIcon-devil:before { content: '\e805'; } /* ' ' */
.customIcon-surprised:before { content: '\e806'; } /* ' ' */
.customIcon-tongue:before { content: '\e807'; } /* ' ' */
.customIcon-coffee:before { content: '\e808'; } /* ' ' */
.customIcon-sunglasses:before { content: '\e809'; } /* ' ' */
.customIcon-displeased:before { content: '\e80a'; } /* ' ' */
.customIcon-beer:before { content: '\e80b'; } /* ' ' */
.customIcon-grin:before { content: '\e80c'; } /* ' ' */
.customIcon-angry:before { content: '\e80d'; } /* ' ' */
.customIcon-saint:before { content: '\e80e'; } /* ' ' */
.customIcon-cry:before { content: '\e80f'; } /* ' ' */
.customIcon-shoot:before { content: '\e810'; } /* ' ' */
.customIcon-squint:before { content: '\e811'; } /* ' ' */
.customIcon-laugh:before { content: '\e812'; } /* ' ' */
.customIcon-wink2:before { content: '\e813'; } /* ' ' */
.customIcon-spin1:before { content: '\e830'; } /* '蚕 */
.customIcon-spin2:before { content: '\e831'; } /* '夷 */
.customIcon-spin3:before { content: '\e832'; } /* '尝 */
.customIcon-spin4:before { content: '\e834'; } /* '瞜 */
.customIcon-spin5:before { content: '\e838'; } /* '翳 */
.customIcon-spin6:before { content: '\e839'; } /* '羚 */
.customIcon-firefox:before { content: '\e840'; } /* '劳 */
.customIcon-chrome:before { content: '\e841'; } /* '褪 */
.customIcon-opera:before { content: '\e842'; } /* '褪 */
.customIcon-ie:before { content: '\e843'; } /* '类 */
.customIcon-crown:before { content: '\e844'; } /* '诉 */
.customIcon-crown-plus:before { content: '\e845'; } /* '谴 */
.customIcon-crown-minus:before { content: '\e846'; } /* '賰 */
```

There are tools you can use to generate custom icon fonts. We recommend you to use Fontello as it quickly builds everything you need to include vector images into your web pages and it also provides open-source artworks.

If you opt for using Fontello, after you select/upload the icons that you want to use, make sure that all the icons have the same prefix. You can accomplish this from the **Customize Names** tab.

Icons must the same "customIcon" prefix:

```
customIcon-happy
customIcon-wink
customIcon-unhappy
```

```
customIcon-sleep
```

Create the JavaScript file that will do the heavy lifting of delivering your icons into the platform:

```
! function ($) {
    if (!$.iconset_list || !$.iconset_list.length) {
            $.iconset_list = [];
    }
    /* Replace customIcon with your own icons family name */
    /* The way that the app adds your icon class to the icon
container is this: class="iconClass iconClassFix-icon"
    /* The iconClass is not mandatory but we have it because some
font families use it (ex. fontawesom). */
    $.iconset_customIcon = {
        iconFamily: "customIcon",
        iconClass: "",
        iconClassFix: "customIcon-",
        icons: ["happy", "wink", "unhappy", "sleep", "devil",
"surprised", "sunglasses", "chrome", "opera", "ie"]
    }
    $.iconset_list.push('customIcon');
}(jQuery);
```

Files Location

Add the files to the following location: *custom/customAssets/icons/customIconSet*. If you have another custom folder path set up, add the files to the following location *yourCustomFolder/customAssets/icons/customIconSet*, where customIconSet is the folder containing your files.

NOTEIf you add your files to a location other than the ones mentioned above, the icons will not be loaded.

Use custom icons

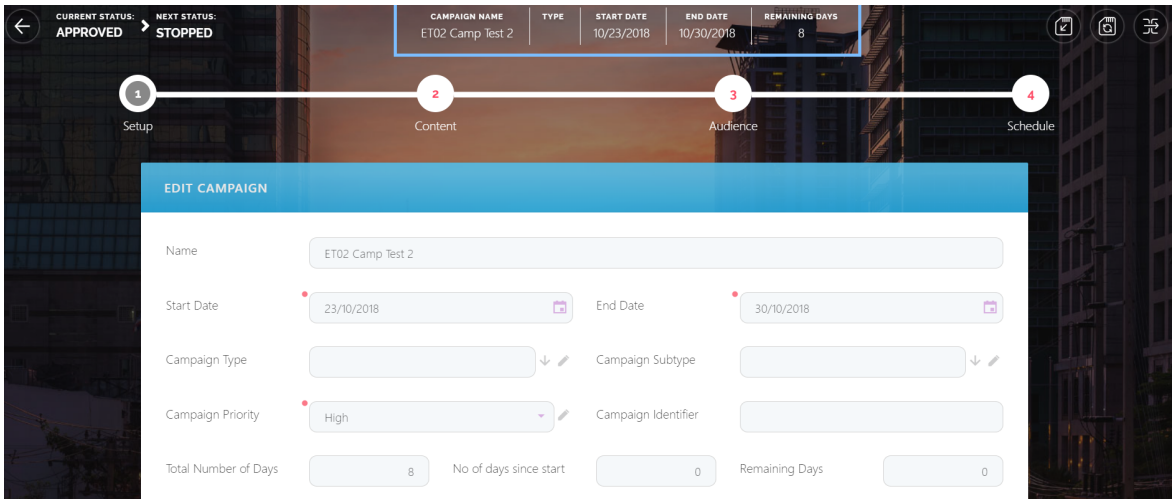
If you got all the needed files and you put them into the custom folder previously mentioned, in Innovation Studio, from the menu, click **Admin** > **Settings**. On the **General** tab, scroll-down, select **Use Custom Styles** and in the **Custom Styles Folder** field, enter the name of the folder where you have the custom files located, if it is other than "custom".

Setting Sticky Header

On data form driven flows, in the **Journey Configuration** page, **Header Items** tab, if you defined several header items, select the **Sticky Header Items** checkbox (it is selected by default) and on save, the journey header items will be displayed in the **Digital Experience Portals** on the menu bar.

> **NOTE**
> The Portal users might need to refresh the Portal in order for the header items to appear on the specific digital journey.

The figure below shows how the campaign header items are displayed in the **Digital Experience Portals**:



Grouping Entities in Menu Items

For usability purposes, you can group business related entities in menu items which will be displayed on the main menu.

To do so, follow these steps:

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Menu Items**. The **Menu Items List** page appears. It contains two records: **Portal** which allows you do define the entities to be shown in the Portal main menu, and **Studio** which allows you do define the entities to be shown in the Innovation Studio main menu.
2. Double-click on the **Portal** record. The **Edit Menu Item** page appears. In this page, you can only change the **Icon URL** and add, edit existing or remove menu items from the c.
3. At the top of the **Menu Item Children** section, click the **Insert** button. The **Add Menu Item** page appears.
4. Select the type of entity that you want to include in the menu. The following options are available:

   Entity - allows you to add a specific entity as item under the current menu.

   Custom Flow - allows you to add a specific custom flow as item under the current menu.

   Report - allows you to add a specific report as item under the current menu.

   Menu Section - allows you do define a sub-menu of the current menu. You will be able to add new menu items under the current menu.

   The content of the **Add Menu Item** page changes based on the menu item **Type** you selected.
5. Make sure that you fill-in the mandatory fields, the ones marked with a red dot.

   NOTEYou can select only entities which have in the entity form the **Portal** selected from the **View On** field.
6. At the top-right corner of the page, click the **Save and New** icon if you want save current menu item and add a new one, otherwise click the **Save and close** icon.
7. Reorder the menu items to best suit your needs by drag and drop records in the **Menu Item Children** section.

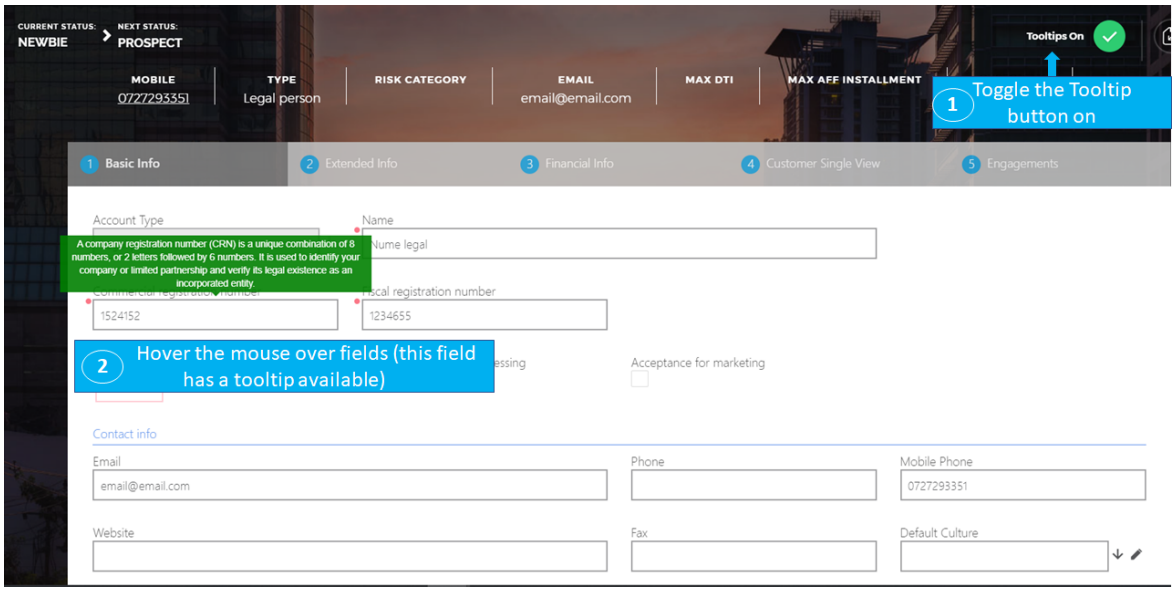
8. At the top-right corner of the page, click the **Save and close** icon.

Show Tooltips (for users)

Tooltips provide you useful information on what specific fields mean and help you understand what actions you should perform.

> **NOTE**
> You can see tooltips in the Portal UI only if your portal has been customized to show tooltips. Two customizations are available for showing tooltips: to always show tooltips when available and to give you the possibility to choose if you want to show tooltips or not. If the Portal has been customized not to show tooltips, none of the two options previously mentioned are available.

If you've been given the ability to decide whether you show tooltips or not, a **Tooltips** toggle button appears at the top-right corner of the UI. If the toggle is on, when hovering your mouse over a field that has a tooltip available, you will see the tooltip text:



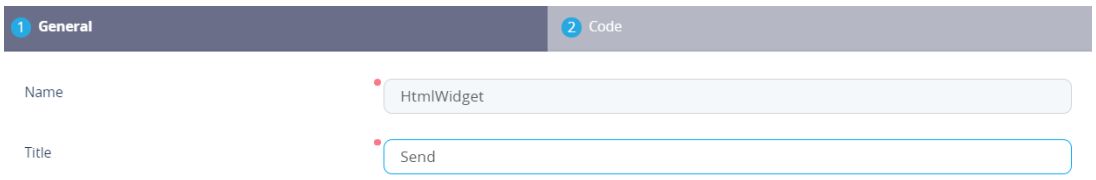
If the tooltips were customized to always be shown in the UI, you will always see tooltips when hovering over fields that have tooltips available. You will not have the option to turn them off.

Creating HTML Widgets

HTML widgets allow you to show specific actions to be performed by users in the UI, e.g. setting up the customer profile. In Innovation Studio, you can create HTML widgets using HTML and JavaScript code.

To create an HTML widget, follow these steps:

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Widgets**. The **Widgets List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The wizard configuration page appears by default on the **General** tab.
3. In the **Name** field, provide the name of the HTML widget to be used by the system.
4. In the **Title** field, provide the widget title to be displayed in the **Digital Experience Portals**.



5. Click the **Code** tab. You can provide the HTML code in the Html tab or provide the JavaScript code (JavaScript tab). You can create the widget template by either providing the HTML code in the Html field, or by using the UI Designer .
6. After providing the widget code, at the top-right corner of the page, click the **Save and close** icon.

The widget will be saved into the system and it will be displayed in the **HTML Widgets List** page.

Now you can add the HTML widget to a dashboard. For more information on how to add widgets to dashboards, see Adding Widgets to Dashboards.

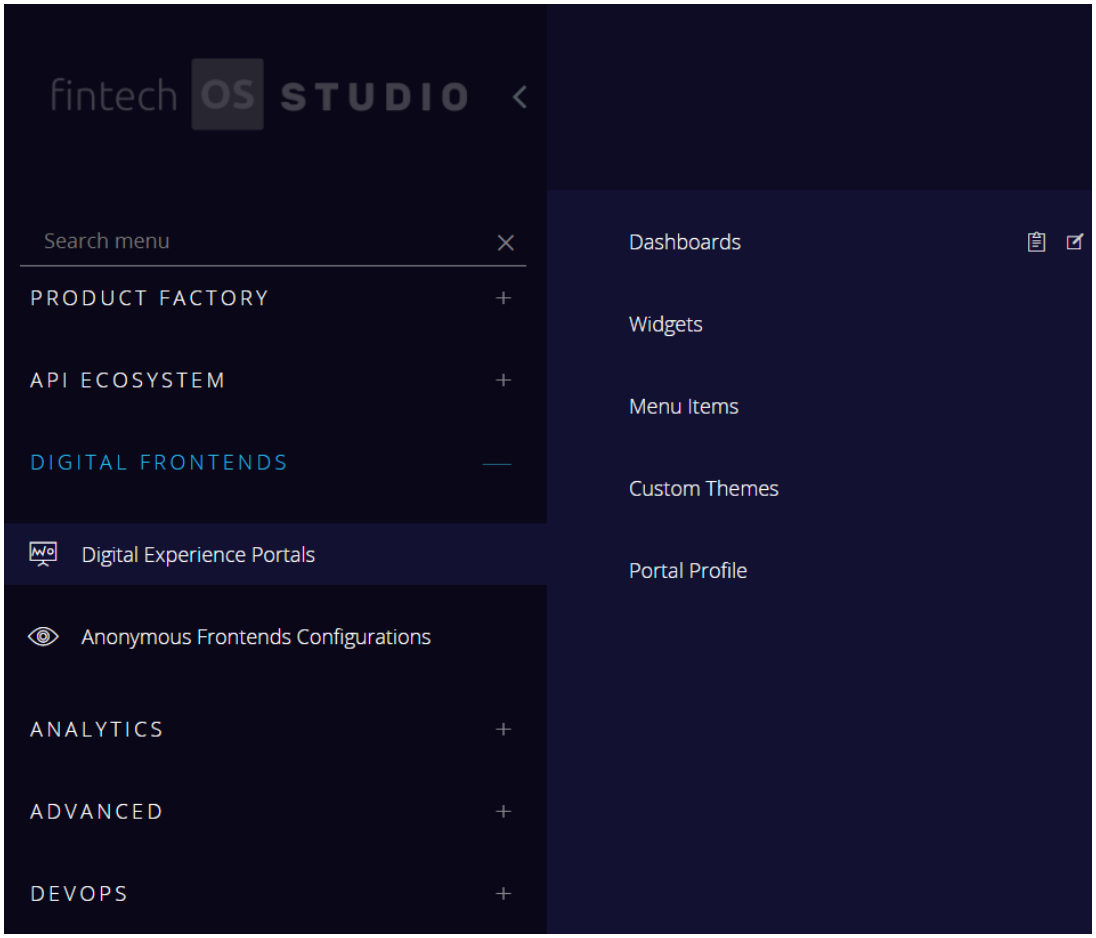
Creating Dashboards

Innovation Studio enables you to aggregate together lists fed from the open data model with the most relevant data (e.g., to-do lists ordered by SLA or severity) by using dashboards.

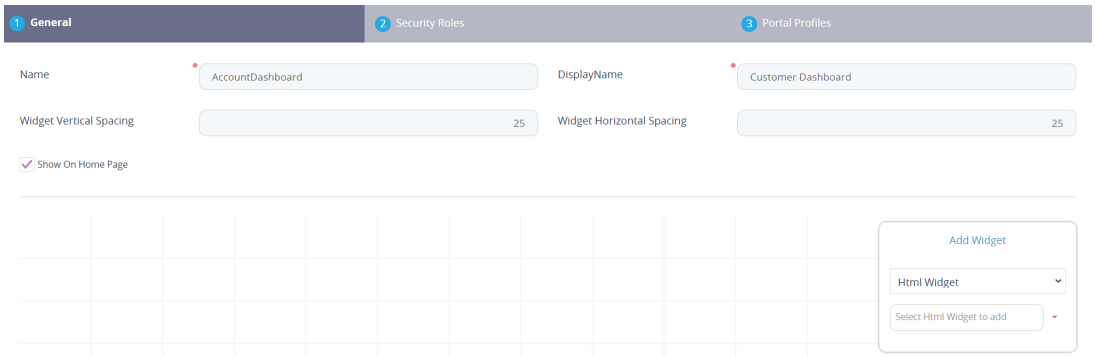
To create a dashboard, follow these steps:

Step 1. Add dashboard

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Dashboards**. The **Dashboards List** page opens.



2. At the top-right corner of the page, click the **Insert** icon. The dashboard configuration page appears by default on the **General** tab.
3. In the **Name** field, provide the name of the dashboard that will be used by the system.
4. In the **Display Name** field, provide the name of the dashboard that will be displayed in the **Digital Experience Portals**.
5. In the **Widget Vertical Spacing** field, provide the amount of vertical space to be add between the dashboard's elements. Default value is 20. The vertical spacer automatically adjusts with the screen size your page is viewed on.
6. In the **Widget Horizontal Spacing** field, provide the amount of horizontal space to be add between the dashboard's elements. Default value is 20. The horizontal spacer automatically adjusts with the screen size your page is viewed on.
7. Tick the **Show On Home Page** checkbox if you want the dashboard to be shown on the Portal home page. If the checkbox is false, then the dashboard and everything placed in its layout will not be rendered in the Portal.



8. At the top-right corner page, click the **Save and Reload** icon. The page refreshes and you can now add widgets to the dashboard. For more information on creating and adding widgets to dashboards, see .Adding Widgets to Dashboards.

You can also attach the security roles who have privileges to see the dashboard and add the dashboard to a portal profile. To do so, read the sections below.

Step 2. Attach security role to a dashboard

If your business case requires the dashboard to be available to designated roles within your organization, click the **Security Roles** tab and add the security roles that should have access to the dashboard. If no security roles are added here, all users will be able to view the dashboard in the **Digital Experience Portals**.

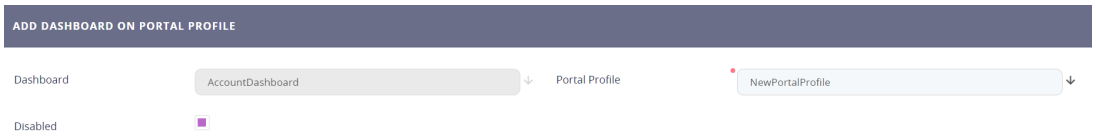
Once you finish customizing your dashboard, at the top-right corner of the page, click the **Save and close** icon to save the dashboard settings or **Save and reload** if you want to add the dashboard on a portal profile. For more information on how to add a dashboard to a portal profile, see the next section.

The dashboard is displayed in the **Dashboards List** page.

You can add as many dashboards as you need. The **Order Index** column specifies the order in which dashboards appear in the **Change Dashboard** drop-down menu on the Portal home page. You can change the order index of a record by using drag and drop.

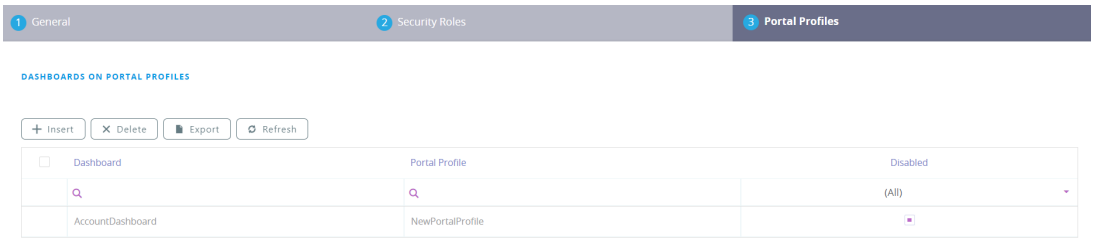
Step 3. Add dashboard on a portal profile

1. In the dashboard configuration page, click the **Portal Profiles** tab.
2. Click the **Insert** button at the top of the **Dashboards on Portal Profiles** section. The **Add Dashboard on Portal Profile** page appears.
3. Select **Portal Profile** on which the current dashboard will be added on.



4. At the top-right corner of the page, click the **Save and close** icon. The **Add Dashboard on Portal Profile** page closes and the record is displayed in the **Dashboards on Portal**

Profiles section.



The **Dashboards on Portal Profiles** section lists both dashboard added on the portal profile from the dashboard configuration page (**Portal Profiles** tab) and from the portal profile configuration page (if any).

Adding Widgets to Dashboards

> **NOTE**
> You can add the following widget types to a dashboard: PowerBI reports, HTML widgets, entity views. You should first create the widget and then add it to dashboards.

This section walks you through the steps that you need to follow to add widgets to dashboards and how to resize and customize the widgets.

Prerequisite

If you want to add HTML widgets to dashboards, you first need to create the HTML widget. For information on how to create HTML Widgets, see "Creating HTML Widgets" on page 567.

Add widgets to dashboards

To add a widget to a dashboard, follow these steps:

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Dashboards**. The **Dashboards List** page opens.
2. Double-click the desired dashboard from the list. The dashboard configuration page appears.

3. In the **Add Widget** area, from the widget type drop-down list, select the desired widget type.
4. Select the widget to be added.
5. Click the **Add** button. The selected widget is displayed on the left side of the page.
6. Optionally, customize the widget. For more information on how to customize widgets, see section below.
7. At the top-right corner of the page, click the **Save and close** icon.

Customize Widgets

There are various options for customizing widgets: resizing, adding widget details, and customizing the widget layout.

Resize Widgets

You can resize dashboard widgets by putting the cursor on the bottom-right corner of the widget that you want to resize. Click and simultaneously drag and drop to rezise as preferred.

Customize Widgets Layout

To customize the layout of a widget double click the widget, click the **Customize Widget** tab on the right side and make the desired layout settings.

> **NOTE**
> You cannot customize the layout of widgets that are entity views.

Click the **Add Widget** tab. You can choose to add a title by clicking the **Show Title** checkbox and providing the widget title to be displayed on the widget.

Click the **Save Widget** button and at the top-right corner of the page, click the **Save and close** icon to save the changes.

Adding Journeys to Dashboards

NOTE

- Create the form driven flow first and then add it to the desired dashboard by using entity dashboards.
- Do not confuse the entity shortcuts added to dashboards with shortcuts added to the Shortcuts by pinning entities on the main menu.

To add a form driven flow to a dashboard, add an entity shortcut to the desired dashboard by following these steps:

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Dashboards**. The **Dashboards List** page opens.
2. Double-click the desired dashboard from the list. The dashboard configuration page appears.
3. In the **Add Widget** area, from the widget type drop-down list, select **Shortcut**.
4. From the **Select the type of Shortcut** drop-down, select **Entity**.
5. Choose how you want to have the data form displayed on the dashboard:

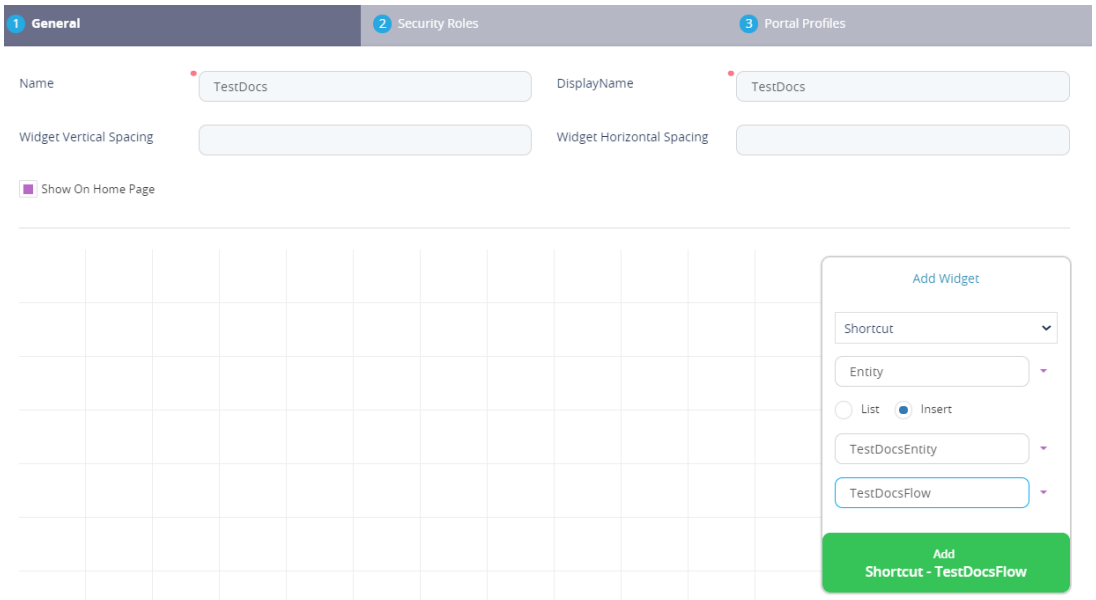
   Select **List** if you want the data form to be listed on the dashboard.

   Select **Insert** if you want the data form to be inserted on the dashboard. A new drop-down field will be displayed.

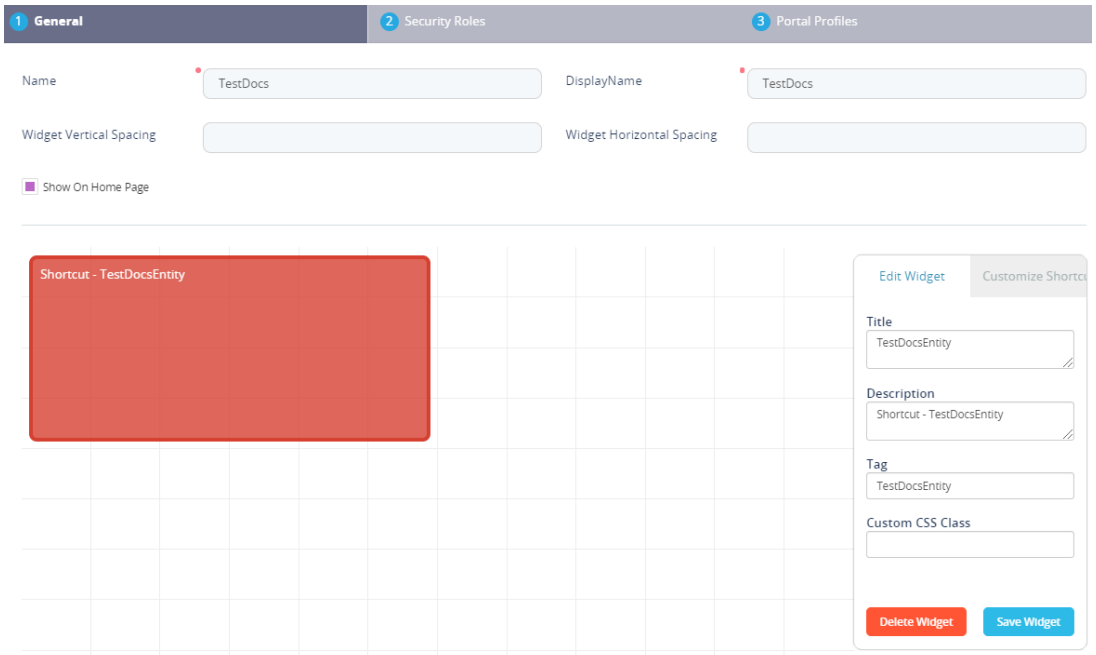
   HINT

   A user can add a form driven flow mockup for preview. Select the title of the mockup when inserting the shortcut.

6. Select the desired entity for which you want to have the default data form added to the dashboard.



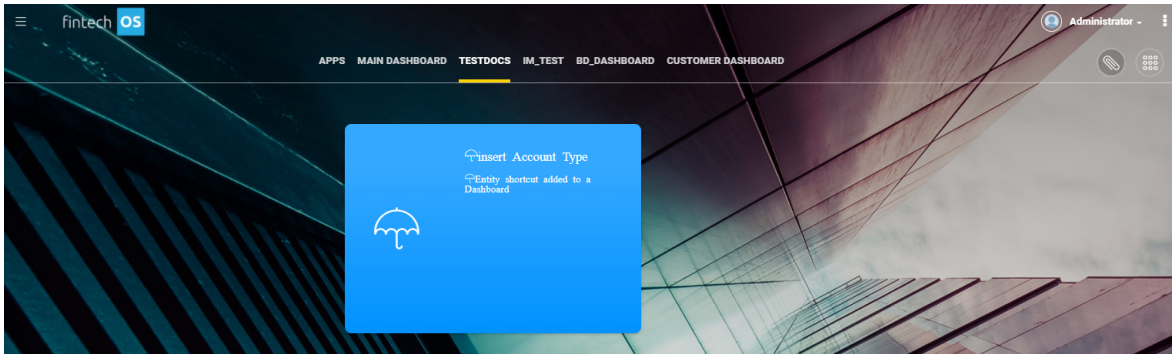
7. Click the **Add** button. The shortcut widget is displayed on the left side of the page. You can resize the widget by placing the cursor on the bottom-right corner of the widget and using drag and drop.
8. Optionally, customize the entity shortcut widget. To do so, on the left-side, double-click the widget. The add widget details displayed on the right-side are replaced by customizable widget details. You can choose to add a title, provide the widget title, description and tag.



Save the customization by clicking the **Save Widget** button.

9. At the top-right corner of the page, click the **Save and close** icon.

The figure below shows an example of an entity shortcut added to the dashboard.



When clicking on the desired entity shortcut on the dashboard, the entity default data form will be displayed in the format selected when adding the shortcut widget to the dashboard (List or Insert).

To add custom flows, see "Creating Custom Flows" on page 272.

Adding Charts to Dashboards

This section walks you through the steps for adding charts to dashboards, as well as for resizing and customizing charts.

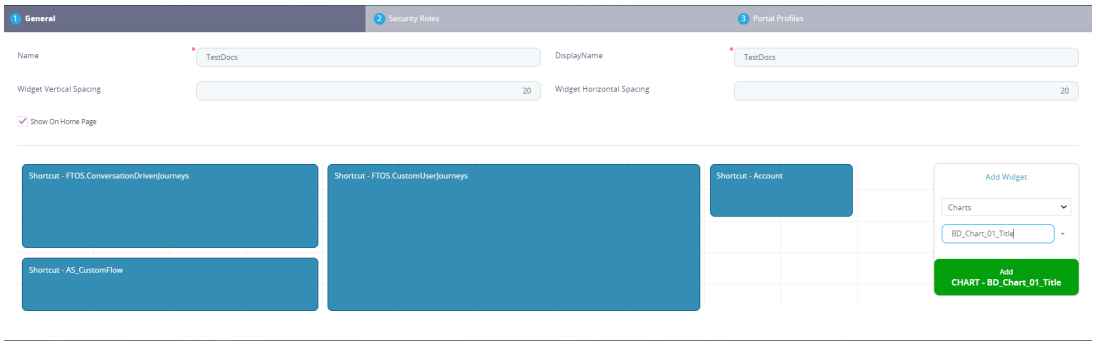
Prerequisite

If you want to add charts to dashboards, you first need to create the chart.

Add widgets to dashboards

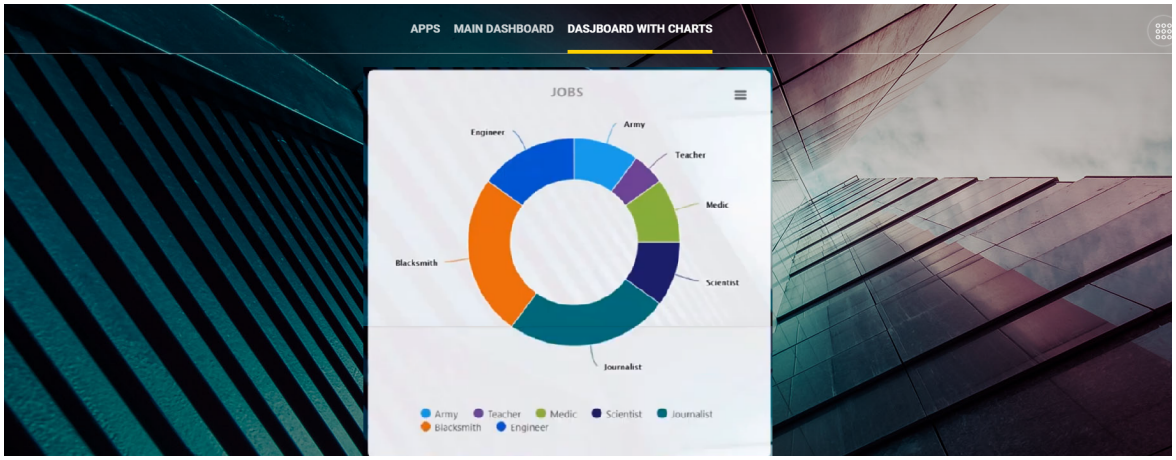
To add a chart to a dashboard, follow these steps:

1. From the menu, click **Digital Frontends** > **Digital Experience Portals** > **Dashboards**. The **Dashboards List** page opens.
2. Double-click the desired dashboard from the list. The dashboard configuration page appears.
3. In the **Add Widget** area, from the widget type drop-down list, select **Charts**. For more details, see "Charts" on page 463.
4. Select the chart to be added.
5. Click the **Add** button. The selected chart is displayed on the left side of the page.



6. Optionally, customize the widget. For more information on how to customize widgets, see section below.
7. At the top-right corner of the page, click the **Save and close** icon.

The figure below shows how a chart may look like on a dashboard in the Digital Experience Portal.



Customize Widgets

There are various options for customizing the way widgets of type charts are displayed in the **Digital Experience Portals**: resizing. One of them is by adding widget details and customizing the widget layout.

Resize Widgets

You can resize dashboard widgets by placing the cursor on the bottom-right corner of the widget that you want to resize. Click and simultaneously drag and drop to rezise as preferred.

Customize Widgets Layout

To customize the layout of a widget double click the chart widget, click the **Customize Widget** tab on the right side and make the desired layout settings.

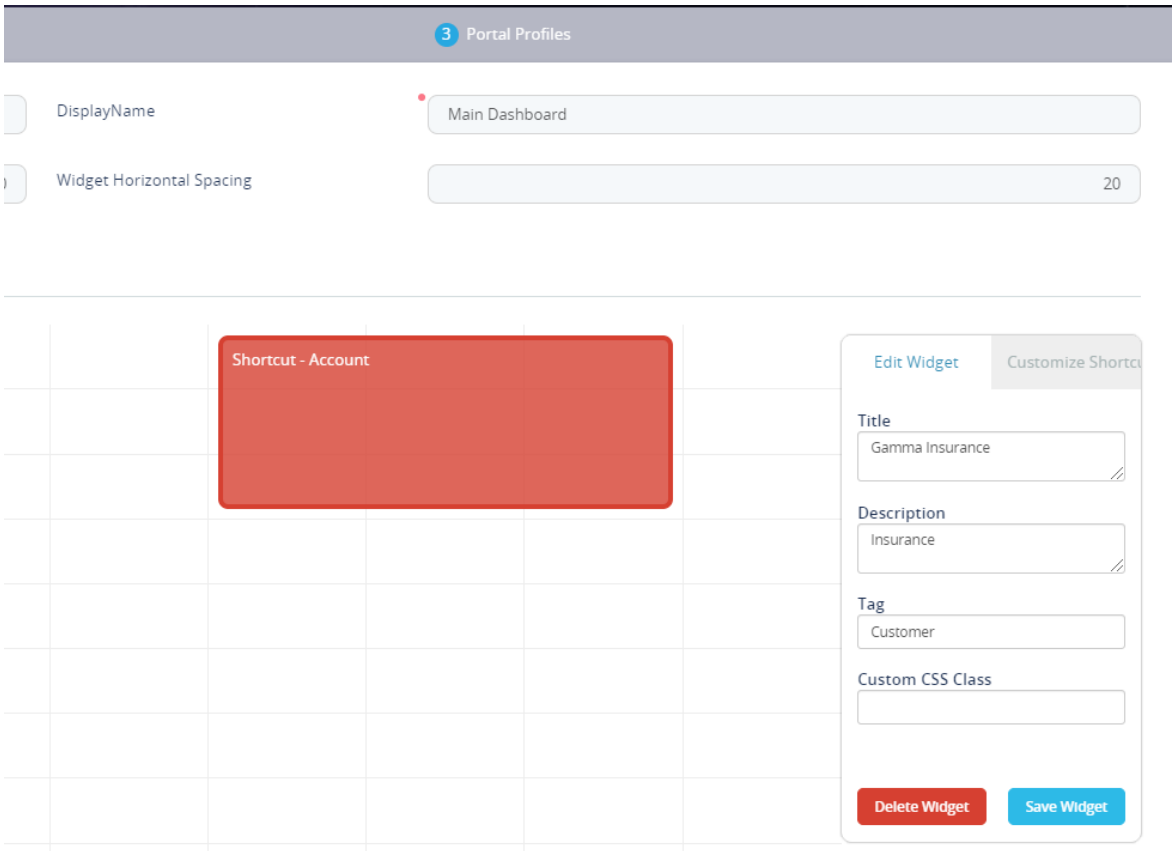
Click the **Add Widget** tab. You can choose to add a title by clicking the **Show Title** checkbox and providing the widget title to be displayed on the widget.

Click the **Save Widget** button and at the top-right corner of the page, click the **Save and close** icon to save the changes.

Editing Dashboards

You can edit dashboards by changing the dashboard details, adding new widgets, editing or removing them.

To remove a widget from a dashboard, go to the dashboard customization page, double-click the desired widget and on the right-side click the **Delete Widget** button. Then, Click the **Save and close** button.



Using Portal Profiles

Innovation Studio provides you with the possibility to configure **Digital Experience Portals** using portal profiles.

Portal profiles enable the customization of each portal to fit to specific brand.

Using portal profiles, you are able to:

- choose the company logo;
- choose the background and login background images;
- set specific system parameters;
- attach menu items;
- add dashboards.

Follow the steps below to customize **Digital Experience Portals** using portal profiles:

1. Insert in the web.config file of the FintechOS Portal one key with the name of the Portal Profile you wish to have.

   HINT

   Do not insert multiple keys with the name of all your Portal profiles. The system will take the last key inserted and its configurations as default.

2. To create secondary Portal profiles to the same Innovation Studio and database, duplicate the Portal file that you have in the server or locally. Rename the file as the name of the secondary Portal profile and insert in the web.config file the key containing the name. Therefore, you will have two Portal files with two web.config files and each as inserted only one key with the name of the Portal Profile.
3. Create the portal profile in the Innovation Studio using no-code.
4. **IMPORTANT!**

   Make sure the name is exactly as the value in the key.
5. Configure the portal profile to be used.

6. In the **Internet Information Services Manager** create a new URL for the each portal Profile you have created. The only difference in the URL is the name of the environment. For example https://insertservername.com/Genie20v1.7Gold_ PortalProfileNr.2/Main.

> ⚠ **IMPORTANT!**
>
> Two Portal profiles cannot exist at the same link! Each portal profile has its URL.

For example, there are two users named George and Paul. Each of them is using a different portal profile on two different environments. George is an operator, therefore his portal shows the queue of calls he has to answer, and the company logo. Paul works in accounting, therefore menu items only available for his user show him the general ledger and associated transactions. On the dashboard, he has reports and charts displayed, as well as an accounting logo.

To achieve this, two separate portal profiles were created in the Studio through **Digital Frontends**. Menu items and dashboards were attached to each portal.

For the banking industry, for example, a portal profile could contain menu items that trigger banking processes such as applications for loans, mortgage, corporate investment, savings solutions etc.

For the insurance industry, a portal profile may contain menu items that trigger processes such as motor quote and bind, life insurance, registering a first notification of loss, collecting claims and others.

Step 1. Insert key

To customize a digital experience portal using a specific portal profile, go to the *web.config* file and add the following setting:

```
<add key="core-setting-portal-profile" value="PortalProfile"/>
```

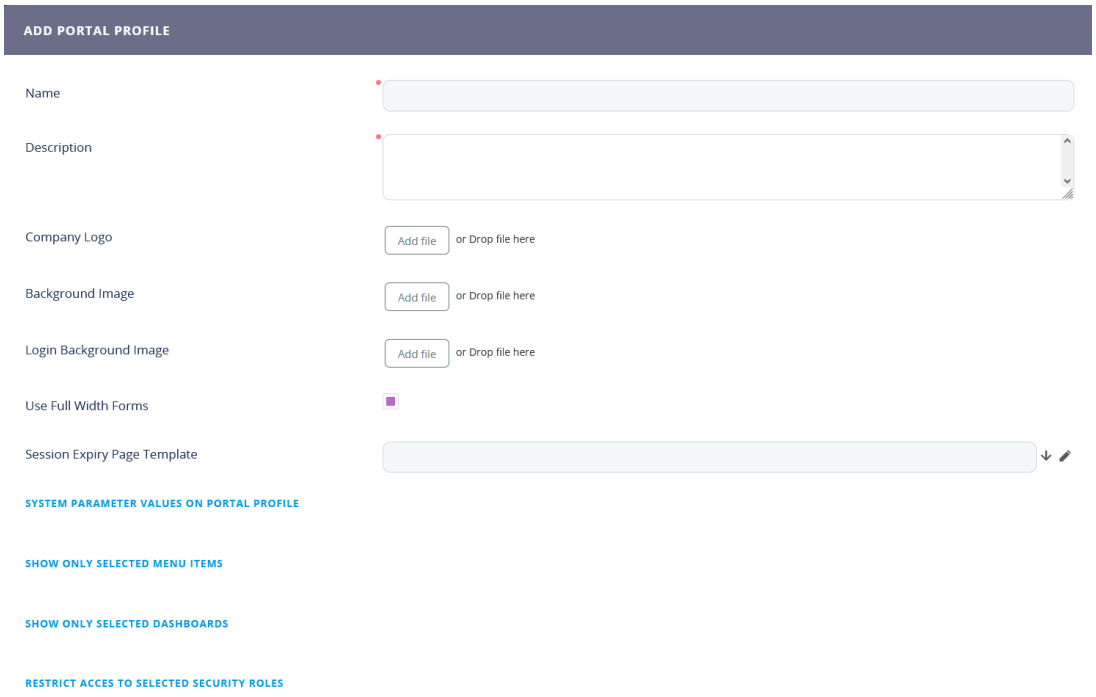
The <profile_name> value will filter the dashboards, system parameters and menu items in regard to this value.

Copy the Portal file that you have in the server or locally to create secondary Portal profiles to the same Innovation Studio and database. For example, the value is ''PortalProfile_Lighthouse''. Rename the file as the name of the secondary Portal profile as PortalProfile_Lighthouse and insert in the web.config file of the PortalProfile_Lighthouse the key containing the name PortalProfile_Lighthouse. If the core-setting-portal-profile app setting is missing or the value is empty or the profile name doesn't match any existing portal profiles, the functionality related to Portal Profiles will not be applied.

Step 2. Create portal profiles

To create a portal profile, follow these steps:

1. In Innovation Studio, from the Main Menu, click **Digital Frontends** > **Digital Experience Portals** > **Portal Profile**. The Portal Profiles List appears.
2. At the top-right corner of the page, click the **Insert** icon. The Add Portal Profile page appears.



3. Provide a **Name** and a **Description** for the portal profile.

4. You can display your own **Company Logo** on the portal pages by adding a custom logo file.
5. You can use your own **Background Image** and **Login Background Image** by adding the desired image file(s).
6. Check the **Use Full Width Forms** if you wish the forms displayed in the portal to cover the full width of the screen.
7. If you wish to customize the Session Expired page template, use this control to insert new "Custom Page Templates" on page 586 and/or to select one from the list. Otherwise, the default login page will be displayed when the user's session expires.
8. Save the portal profile by clicking the **Save and close** button at the top-right corner of the page or the **Save and reload** icon if you want to continue adding dashboards and menu items to the portal profile.

If you click the **Save and reload** icon, the **Add Portal Profile** page is replaced by the **Edit Portal Profile** page and all sections are unlocked.

See the links below for information on how you can the further customize your portal profile:

System Parameters on Portal Profiles

You can overwrite the values of a FintechOS system parameter with specific values per portal profile. You can also disable specific system parameters per portal profile.

NOTEThe default values of the system parameters will be overwritten by the values given in the portal profiles only when using the portal profile.

To overwrite a system parameter's value per portal profile, add the system parameter and its new value on the portal profile by following these steps:

1. In the portal profile configuration page (**Edit Portal Profile** page), scroll-down to the **System Parameters on Portal Profiles** section and at the top of the section, click the

Insert button. The **System Parameter on Profile** page opens.

2. Select the desired **System Parameter** and provide the **Parameter Value**.
3. Optional! You can **Disable** the system parameter if you don't need it in the digital experience portal. When you disable a system parameter in a portal profile, the corresponding global scope system parameter value will be used instead (set in **Main Menu** > **Admin** > **System Parameters**).
4. At the top-right corner of the page, click the **Save and close** icon to add the system parameter value on the portal profile.

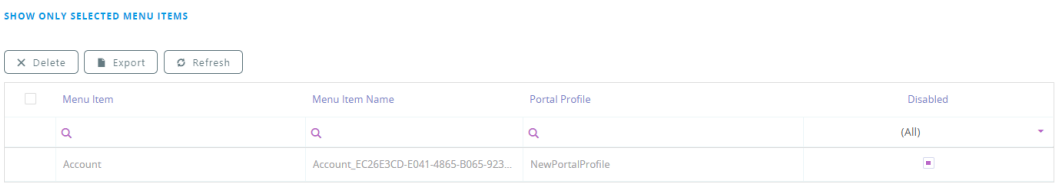
You can add as many system parameters values as you need on the portal profile by following the procedure above.

Attach Menu Items on Portal Profile

Prerequisite: In order to add menu items to a portal profile., you need to create the menu items first.

To attach a menu item to a portal profile:

1. In the portal profile configuration page (**Edit Portal Profile** page), scroll-down to the **Show Only Selected Menu Items** section and at the bottom of the section, click the **Attach Menu Item** button. A window appears listing the menu items you have defined.
2. Double-click the desired record from the list. A confirmation dialog appears.
3. Click **OK**. The selected record is displayed in the **Show Only Selected Menu Items** section.



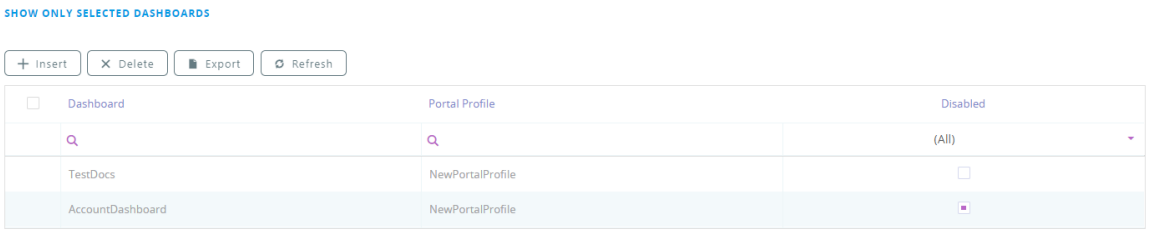
You can attach as many menu items as you need by following the procedure above.

Add Dashboards on Portal Profiles

Prerequisite: In order to add dashboards to a portal profile, you need to create the dashboards first; otherwise you will be able to add only the Main Dashboard.

To attach a menu item to a portal profile:

1. In the portal profile configuration page (**Edit Portal Profile** page), scroll-down to the **Show Only Selected Dashboards** section and at the top of the section, click the **Insert** button. The Add Dashboard on Portal Profile page appears.
2. Select the **Dashboard** by clicking the down arrow and double-clicking the desired dashboard in the window that appears.
3. At the top-right corner of the page, click the **Save and close** icon. The selected record is displayed in the **Show Only Selected Dashboards** section.



You can add as many dashboards as you need by following the procedure above.

Access to Portal Profiles based on Security Roles

Access to a specific Portal Profile can be restricted to users with a specific security role. Access to a profile can be full or restricted.

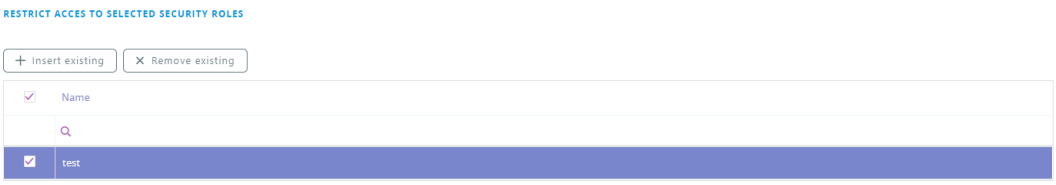
> **NOTE**
> In the web.config file make sure the name of the Portal Profile is set.

Setting security roles

This configuration enables users with a security role to see the Portal Profile. However, it can also exclude items from their viewing access. Depending on your business needs, you can add a security role to access the Profile or set restrictions to a security role or combinations of security roles with different viewing/ editing rights.

After having created the Portal Profile, follow the steps:

1. Open Innovation Studio, select the **Digital Frontends**, and click the menu item **Digital Experience Portals**.
2. Select the **Portal Profile** sub-item. Open the individual profile you wish to edit.
3. Scroll down to the **Restrict Access to Selected Security Roles** grid.
4. Click **Insert existing** to insert a new security role or **Remove existing** to delete access of a role.



5. Select the role you wish to add to the Portal Profile. Repeat as many times as needed to grant access to security roles.

Restrictions

Here, it is also possible to exclude items from specific security roles.

1. After inserting the role, double- clicking on it.
2. In the security item grid, click **Insert** and a new window will open. Fill in the fields.
3. To add a security item, from the **Security Items** section, click the **Insert** button. The **Add Security Item** page is displayed. In the **Entity** field, type the entity name or click the down-arrow and select it from the list. In the **Security Operation** field, type the record-level privilege (CRUD operation) or click the down-arrow and select it from the list. You can choose one of the following:

Read	Allows users to view records.
Update	Allows users to update records.
Insert	Allows users to add new records.

Delete	Allows users to delete records.

In the **Security Scope** field, type the level of access or click the down-arrow and select it from the list:

User	Privileges to the records owned by the user or assigned to the user.
Parental	Privileges to all records owned in the business unit to which the user belongs to, including privileges to the records owned in the child business units.
Business Unit	Privileges to all records owned in the business unit to which the user belongs to.
Organization	Privileges to all records in the organization regardless of their owner.

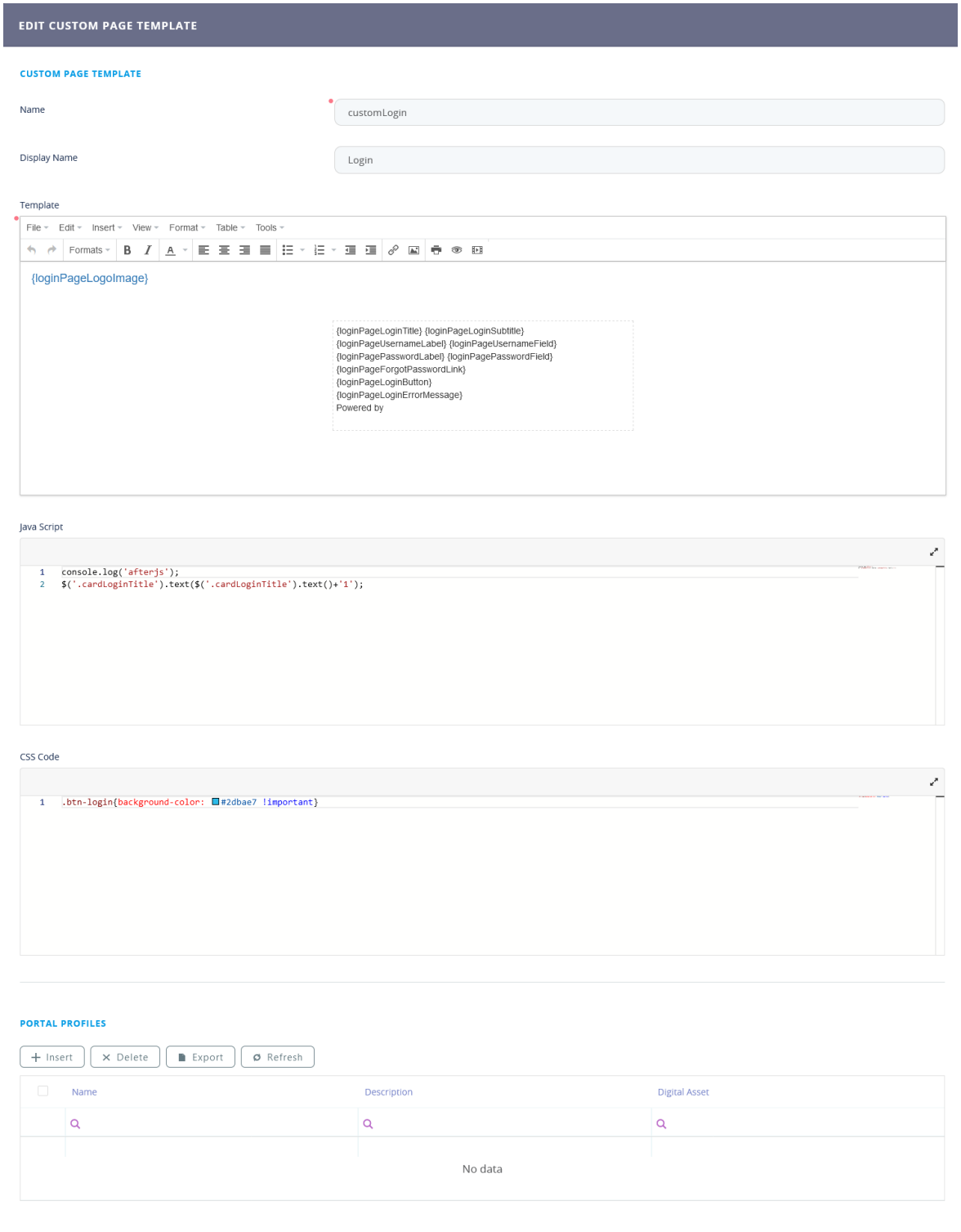
You can add as many items as you need by clicking the **Save and reload** icon and providing the new security item details. For more information, see "Creating Security Roles" on page 622.

4. Click the **Save and close** button in order to close and return to the Portal Profile editing page. Repeat for as many security roles as needed.

Custom Page Templates

Custom page templates allow you to customize specific FintechOS Portal components, such as session expiration pages. You can specify the HTML, CSS, and JavaScript code of the page and also use predefined tokens to configure the look-and-feel of the page logo, translations, or error messages.

NOTE
Currently, the custom page templates only allow customization of the session expiration pages displayed by different portal profiles. In the future, additional uses may become available, such as custom login pages.
At this moment, you can configure custom page templates only from the Session Expiry Page Template control on the portal profile editor. For more information, see "Step 2. Create portal profiles" on page 581.



To configure a custom page template, fill in the following fields:

- **Name** - Internal identifier of the custom page template.
- **Display Name** - The page name displayed in the user interface.
- **Template** - The page content. You can use the rich text editor or, from the Tools menu, you can switch to the Source Code mode to insert the HTML code directly. The rich text editor also supports "Tokens" below, which allow you to insert placeholders for predefined content.
- **JavaScript** - You can use this section to set up JavaScript code to run on the page, such as event handlers.
- **CSS Code** - Insert the code for the styling of the page.
- **Portal Profiles** - You can use this section to associate the template as a Session Expiry Page Template for various portal profiles. For more information, see "Step 2. Create portal profiles" on page 581.

Tokens

You can use the following tokens in the Template section of the custom page.

Token	Generated HTML Code	Observations
{loginPageLogoImage}	`<div class='logoImg' style='background-image: url(""/LoginService/DownloadCompanyLogo"");'></div>`	Inserts either the default logo or the logo set in the portal profile.
{loginPageLoginTitle}	`<div class='cardLoginTitle'>{ Strings.CustomLogin_LoginTitle }</div>`	The translations key can be updated from the Localization page.

Token	Generated HTML Code	Observations
{loginPageLoginSubtitle}	`<div><p data-key='Logon_Product_Description' data-resource='EBS.Core.Web.MVC.LocalizationResources.Strings'>`	The translations key can be updated from the Localization page.
{loginPageUsernameLabel}	`< label for = 'usernameField' class='labelAnimated active'>{ Strings.CustomLogin_ UserNameLabel }</label>`	The translations key can be updated from the Localization page.
{loginPageUsernameField}	`<input autocomplete = 'off' class = 'valid' id = 'usernameField' name='UserName' placeholder='{ Strings.CustomLogin_ UserNameField }' value=' {userNameValue}' type='text' aria-required='true' aria-invalid='false' aria-describedby='usernameField-error'/> `	

Token	Generated HTML Code	Observations
{PasswordLabel}	`<label for='passwordField' class='labelAnimated active'>{ Strings.CustomLogin_ PasswordLabel }</label>`	The translations key can be updated from the Localization page.
{loginPagePasswordField}	`<input autocomplete='new-password' class='valid' id='passwordField' name='Password' placeholder='{ Strings.CustomLogin_ PasswordField }' type='password' aria-required='true' aria-invalid='false' aria-describedby='passwordField-error'/> `	
{loginPageLoginButton}	`<input type='submit' class='btn btn-block btn-lg btn-fill btn-login' value='{ Strings.CustomLogin_ LoginButtonText }'/>`	The translations key can be updated from the Localization page.

Token	Generated HTML Code	Observations
{loginPageForgotPasswordLink}	`<a href='/Account/ForgotPassword'>{ Strings.CustomLogin_ HaveYouForgottenYourPassword }</a>`	This is available only if the reset-password feature is enabled. Otherwise the no HTML is generated.
{loginPageLoginErrorMessage}	`<div class='login-error-message'></div>` or `<div class='login-error-message'> <div class='validation-summary-errors'><span> {errorMessage}</span></div> {passwordResetText}</div>`	In case there is no error, the first HTML is generated. If there is an error, the second one is used. If the reset-password feature is enabled and the password expired, the following code is added: `Click <a href='{0}' > here </a> to reset your password.`

Default Behaviors for Custom Login Pages

- Background picture - If no picture is set in the portal profile, the default login background is inserted.

- Logo - If no logo is set in the portal profile, the default logo is inserted.
- Username and password - Both fields are set as required and validation errors are displayed if no value is provided.

Examples

HTML Section

This example simulates the default login screen using all the tokens:

```
<nav class="navbar navbar-transparent block-height-
7" role="navigation-demo">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="/" style="width:
195.209px;">{loginPageLogoImage}</a>
        </div>
    </div>
</nav>
<div class="content-wrapper">
    <div class="login-container">
        <div class="login-blur-mask"></div>
        <div class="container-fluid">
            <div class="row">
                <div class="col-md-4 col-md-offset-4 col-sm-
8 col-sm-offset-2 col-xs-12 type-6">
                    <div id="card-login-colors" class="card
card-login card-azure" data-color="pink" data-color-
active="false">
                        <div class="preview">
                            {loginPageLoginTitle}
                            {loginPageLoginSubtitle}
                        </div>
                        <div class="content" id="inp-login-
colors">
                            <div>
                                <div class="inp inp-azure
user-wrapper editor-field">
                                    {loginPageUsernameLabel}
                                    {loginPageUsernameField}
                                </div>
                                <div class="inp inp-azure
pass-wrapper editor-field">
```

```
                                    {loginPagePasswordLabel}
                                    {loginPagePasswordField}
                                </div>
                                <div>

{loginPageForgotPasswordLink}
                                </div>
                            </div>
                            <div id="btn-login-
colors" class="btn-login-wrapper">
                                {loginPageLoginButton}
                            </div>
                            {loginPageLoginErrorMessage}
                        </div>
                        <div class="circle white-green
heading-title">
                            <div class="left">
                                <span data-key="Layout_
Footer_PoweredBy" data-
resource
="EBS.Core.Web.MVC.LocalizationResources.Strings">Powered
by</span>
                            </div>
                            <div class="right">
                                <div class="logo-card-
login"></div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

JavaScript Section

This example illustrates the JavaSCript code matching the "HTML Section" on the previous page above.

```
console.log('afterjs');
$('.cardLoginTitle').text($('.cardLoginTitle').text()+'1');
```

CSS Section

This example illustrates the CSS code matching the "HTML Section" on page 592 above.

```
.btn-login{background-color: #2dbae7 !important}
```

Configuring the Digital Experience Portal

FintechOS provides you with extensive options to configure various parameters (UI elements) of the **Digital Experience Portals** directly in the **web.config** file, under the <appSettings> tag:

Use Stripped Theme

The stripped theme features a clean, minimalistic styling and design for all basic page elements:

- home page
- menu
- list of an entity records
- forms (regular, with header items(sticky/non sticky), with bullets sections, with tabs sections, wizard mode, with business status, with action groups, all combinations of the previous)
- dashboards
- shortcuts (on home page/ on header)
- reports
- charts
- custom actions

Using the stripped theme will load only the necessary style sheets needed for the Portal application to run normally. It serves as baseline for creating a Portal theme that best suits your brand and corporate needs.

To use the stripped theme, add the following setting in the **web.config** file:

```
<configuration>
       <appSettings>
            ...
            <add key="feature-minimal-css" value="<b>1</b>"/>
       </appSettings>
...
</configuration>
```

NOTE

- Generating the color palette from the background, choosing the theme and palette are not available.
- This feature is not available in Designer.

Load Custom Style Sheets

Style sheets allow you to define your own styles for forms and digital journeys for better accessibility and improved usability for your own comfort. Using style sheets, you can apply your own text style, text color, padding, etc.

NOTEPrior to loading specific style sheets, you need to create the style sheets. To do so, follow the procedure described in section "Using Your Own Style Sheets" on page 321

To load custom style sheets, add the following setting in the **web.config** file:

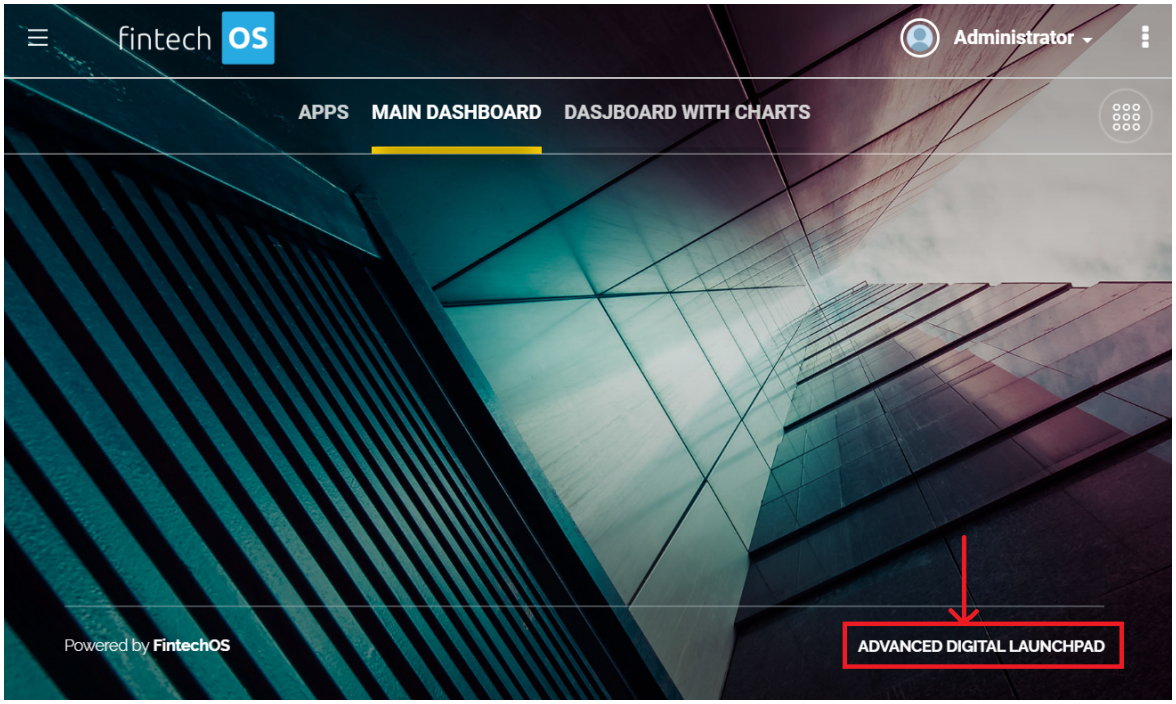
```
<configuration>
    <appSettings>
        ...
        <add key="feature-load-custom-style-
sheet" value="StyleSheet1, StyleSheet2"/>
    </appSettings>
    ...
</configuration>
```

NOTEThe custom style sheets will not apply to the login page.

IMPORTANT!The style sheets will load in the order they are specified in the **web.config** file; therefore, make sure that you properly define the styles. If you have specific styles defined in both custom style sheets, the ones from the second style sheet will overwrite the styles from the first style sheet.

Configure the footer text per language

If you have the Digital Experience Portal serving various languages, you can also set the text shown at the right-side of the footer to be shown per language.



To do so, go to the **web.config** file: and add the following setting :

```
<configuration>
    <appSettings>
        ...
        <add key="feature-right-footer-text" value ="[{'en-
GB':'ADVANCED DIGITAL LAUNCHPAD'},{'ro-RO': 'ADVANCED DIGITAL
LAUNCHPAD'}]"/>
    </appSettings>
```

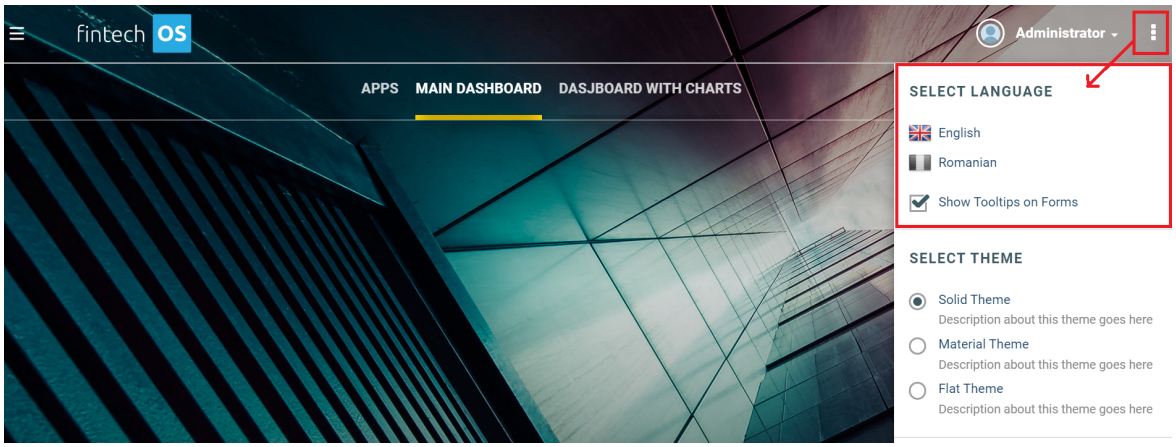
```
    ...
</configuration>
```

The value of the configuration is a json like array with an element for each supported language.

> **NOTE** If the value is empty or the entire xml tag is missing from the **web.config** file, the default text "ADVANCED DIGITAL LAUNCHPAD" is shown.

Move language selection to the user profile panel

The default **Digital Experience Portals** configuration comes with the language selection available in the **User Settings** menu:



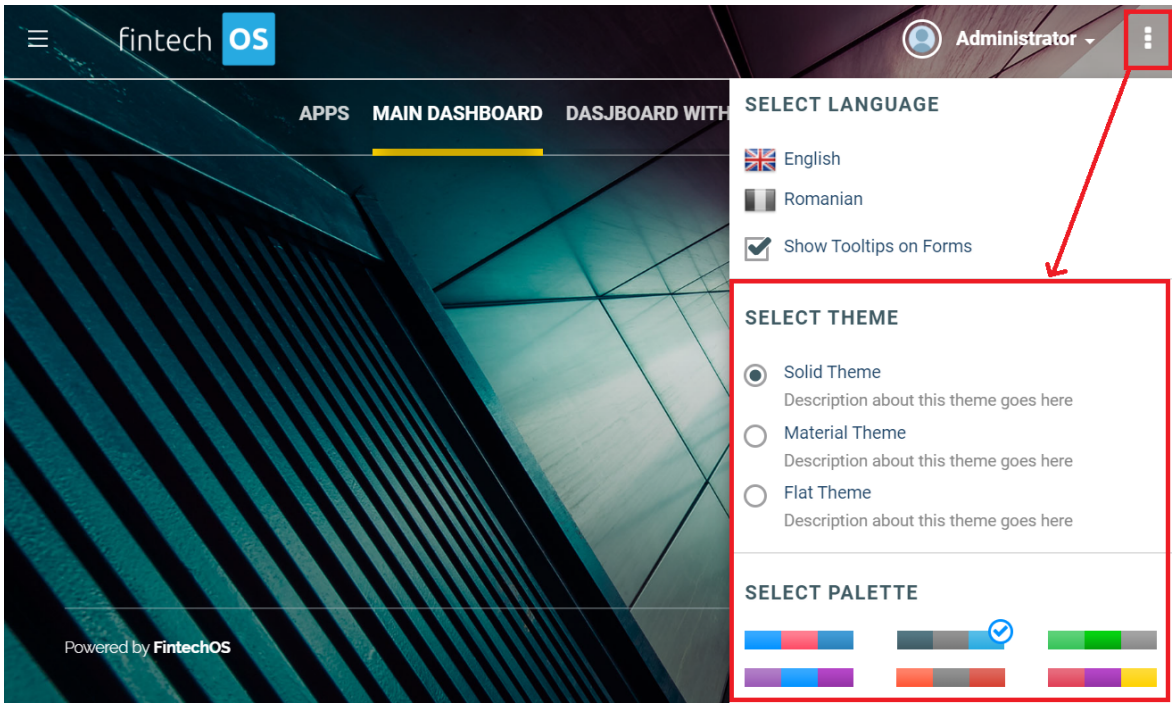
To move the selected language and the language selection to the User Profile panel,, go to the **web.config** file: and add the following setting :

```
<configuration>
    <appSettings>
            ...
        <add key="feature-move-language-to-user-
profile" value="<b>1</b>"/>;
    </appSettings>
    ...
</configuration>
```

If the value is **1** or **true**, the selected language and language selection will be moved to the User Profile panel.

Hide Select Theme and Select Palette settings

The default **Digital Experience Portals** configuration comes with the options to **Select Theme** and **Select Palette** available in the **User Settings** menu:



To hide the options to **Select Theme** and **Select Palette** from the **User Settings** menu, go to the **web.config** file: and add the following setting :

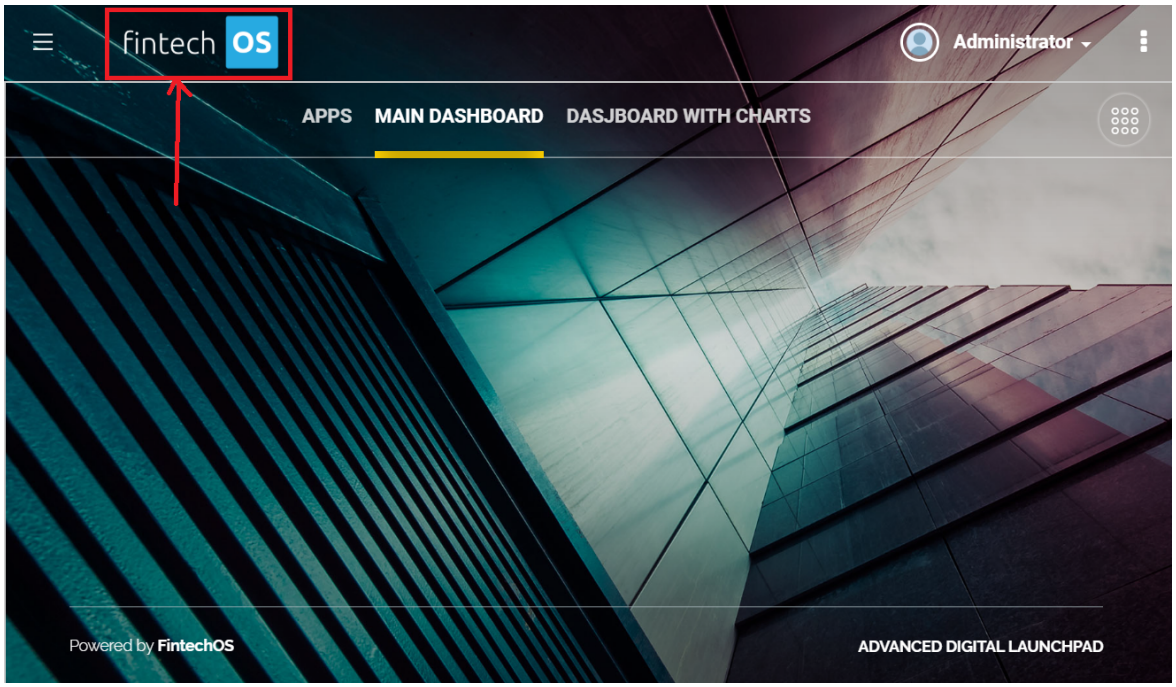
```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-user-settings" value="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

If the value is **1** or **true**, the options to select theme and palette from the **User Settings** menu will no longer be shown.

NOTEIf you move the language selection to the user profile panel and hide the options for selecting theme and palette from the User Settings menu, the menu will be hidden as well.

Hide Company Logo

The default **Digital Experience Portals** configuration comes with the company logo displayed at the top-left corner of the page.



If for any reason, you want to hide the company logo, go to the **web.config** file and add the following setting:

```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-company-logo" value="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

If the value is **1** or **true**, the Company Logo will be hidden.

Hide the Main Menu

The default **Digital Experience Portals** configuration comes with the Main Menu displayed at the top-left corner of the page.

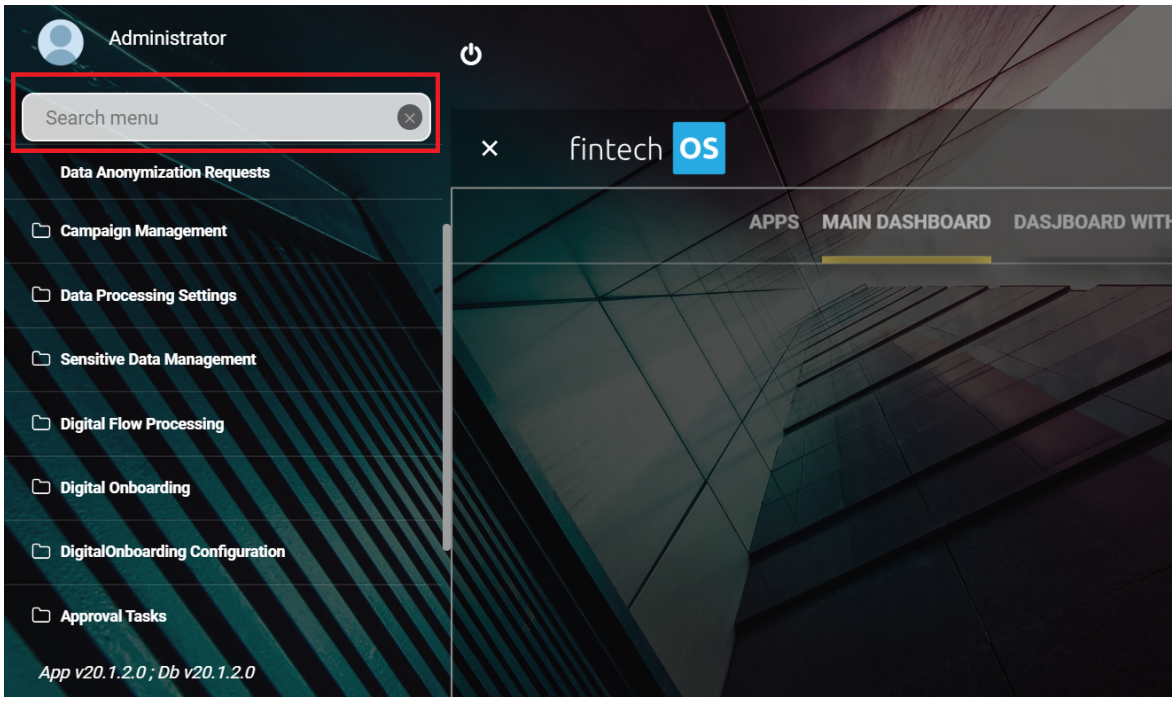
To hide the main menu, go to the **web.config** file and add the following setting:

```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-menu" value="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

If the value is **1** or **true**, the Main Menu will be hidden in the Digital Experience Portal.

Hide search in the Main Menu

The default **Digital Experience Portals** configuration comes with a search available at the top of the Main Menu (when the Main Menu is expanded).



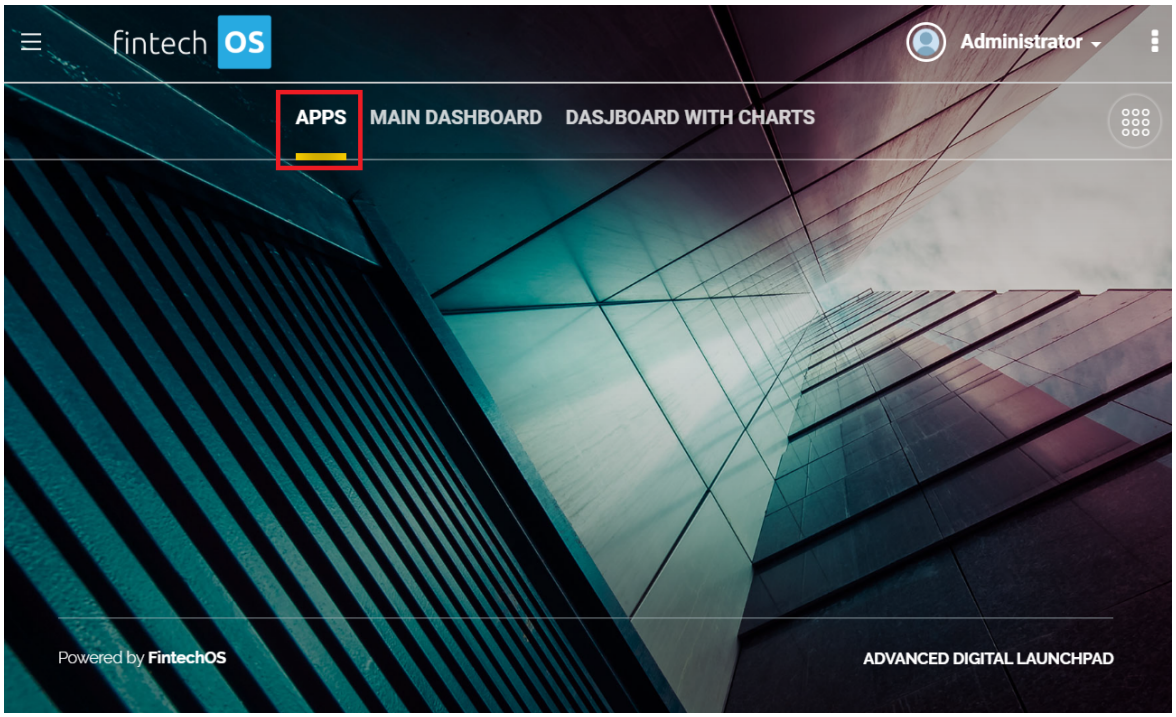
To hide the main menu, go to the **web.config** file and add the following setting:

```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-search-in-menu" value="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

If the value is 1 or true, the Search in the Main Menu is hidden in the **Digital Experience Portals**.

Hide APPS dashboard

The default **Digital Experience Portals** configuration comes with the APPS dashboard available on the home page.



To hide the APPS dashboard, go to the **web.config** file and add the following setting:

```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-app-dashboard" value ="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

NOTEThis setting also hides the TV icon if shortcut links are deactivated (in Innovation Studio, **Admin** > **Settings**, the **Show Shortcuts Tab on Home page** checkbox is not ticked.

Hide My Profile Link from the User Profile panel

The default **Digital Experience Portals** configuration comes with the My Profile link available in the **User Profile** panel. If clicked, the My Profile page appears.

To hide the My Profile link from the **User Profile** panel, go to the **web.config** file and add the following setting:

```
<configuration>
    <appSettings>
        ...
        <add key="feature-hide-my-profile-link" value="<b>1</b>"/>
    </appSettings>
    ...
</configuration>
```

Keyboard Shortcuts

FintechOS provides key combinations you can use from the keyboard as an alternative way to do something that you'd typically do with a mouse.

General keyboard shortcuts

Press this key	To do this
Ctrl + S	Save record changes on forms, sections, wizard mode, when inline editing views (data form, row, batch), when inline editing a data form with children. If the child is on edit mode, when pressing **CTRL+S**, the system saves both the child entity and the parent entity.
Tab	Move forward through entity records and fields.
Shift + Tab	Move back through entity records and fields.

Press this key	To do this
Alt + Left arrow	Go back. If you want to go back after editing records, either by clicking back or pressing the shortcut, a pop-up will be displayed informing you that changes were made and asking if you want to go back without saving the changes. Clicking **Yes** will go back without saving the changes . Clicking **No** will close the pop-up and you'll stay on the page.
Alt + Right arrow	Go forward.

Shortcuts for date/ date time fields

Press this key	To do this
Alt + Down arrow	Display calendar
Alt + Up arrow	Hide calendar.
Shift + Page Up	Move to the previous month in the calendar.
Shift + Page Down	Move to the next month in the calendar.
Enter	Select the date where the focus is.
Esc (Escape)	Close the date picker without selection.

Shortcuts for drop-down fields

Press this key	To do this
Up arrow	Scroll up values in a drop-down without opening the drop-down.
Down arrow	Scroll down values in a drop-down without opening the drop-down.
Alt + Down arrow	Open the drop-down.
Up / Down arrows	Scroll the values of an open drop-down in the direction specified.
Alt + Up arrow	Hide the drop-down.
Enter	Select value.

Shortcuts for radio buttons

Press this key	To do this
Left / Right arrow	Deselect or select the radio button if the active option is a radio button.

Shortcuts for check boxes

Press this key	To do this
Spacebar	Select or clear the check box if the active option is a check box.

Anonymous Frontends

Banking and financial institutions might want to provide their consumers with unauthenticated access to specific contracts and agreements with the click of a button (widget) on their website. Innovation Studio makes this possible by exposing data from form driven flows to unauthenticated users.

NOTEOnly the wizard like data form driven flows (the ones that have the **Wizard mode** checkbox ticked in the journey configuration page) can be exposed to unauthenticated users.

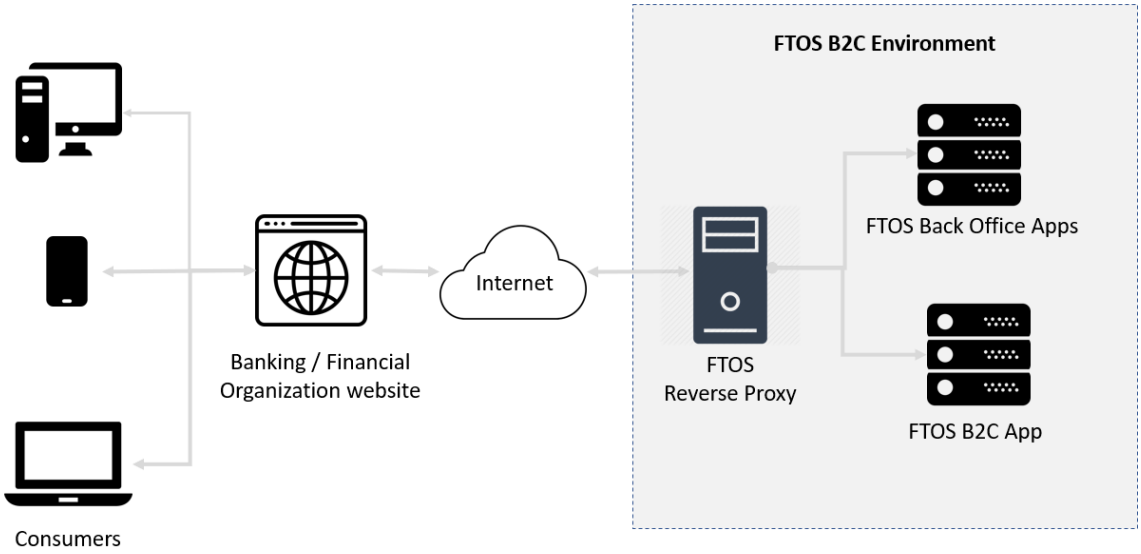
Is it secure to expose digital journeys to unauthenticated users?

An anonymous frontend environment with a secure architecture has been designed to allow exposing journeys to unauthenticated users (consumers).

The desired wizard-like form driven flow is exposed on the organization's website via an iFrame. As iFrame is vulnerable to hacker attacks, a reverse proxy sits between the internet and FintechOS apps that are placed in a non-public subnet.

The reverse proxy ensures a single point of authentication for all HTTP requests, forwarding the requests to the FintechOS B2C App (the one that contains the digital journey to be exposed). It also handles requests to the FintechOS Back Office apps (Innovation Studio and the Digital Experience Portals).

Here's a simplified diagram of the traffic flow.



In order to expose digital journeys, you need to setup a B2C environment. For information on how to do that, see B2C Environment Setup.

Setting B2C Environment

The B2C environment offers a secure architecture for exposing digital journeys to unauthenticated users. The B2C environment is comprised of the following components:

- FintechOS Back Office applications (Portal and Innovation Studio)
- FintechOS B2C Application. It is a copy of the back-office Portal with the following appSettings in web.config

```
<configuration>
    <appSettings>
```

```
        ...
        <add key="feature-b2c-userjourneys" value="1" />
        <add key="feature-b2c-userjourneys-impersonated-
user" value="the username" />
        <add key="feature-b2c-userjourneys-api-
key" value="feature-b2c-userjourneys-api-key" />
     </appSettings>
     ...
</configuration>
```

- **IMPORTANT!**

  - The impersonated user (`feature-b2c-userjourneys-impersonated-user`) cannot be **host** or **admin**.
  - The "Security Roles" on page 621 for the impersonated user must include only read access to the related dictionary entities (entities referenced by lookup attributes used in the journey).
  - Since the impersonated user has limited access to the journey's data model, make sure that you do not use data manipulation functions (such as ebs.getByID, ebs.getByIdAsync.htm, ebs.getByQuery.htm, etc.) in the journey's client-side code. Instead, use server-side scripts which you can access through functions such as ebs.callActionByNameAsync.htm or ebs.callActionByName.htm.

NOTE To expose multiple user journeys to a B2C portal, the user journeys API keys in the **web.config** file, separated by comma:

```
<configuration>
    <appSettings>
    ...
```

```
    <add key="feature-b2c-userjourneys-api-
key" value="feature-b2c-userjourneys-api-key,
feature-b2c-userjourneys-api-key" />
    </appSettings>
    ...
</configuration>
```

- FTOS Reverse Proxy application which ensure secure routing of HTTP requests from internet to FintechOS apps.

How to set up the B2C environment

This section walks you through the steps that for setting up a B2C environment:

STEP 1. Install the FTOS Reverse Proxy

1. Install .NET Core Sdk.
2. Install .NET Core 2.2 Runtime and Hosting Bundle for Windows.
3. Copy the FTOS Reverse Proxy files in a folder of your choice.
4. Create application in IIS:
    1. Create the application pool (.NET CLR=No Managed Code | Managed pipeline mode=Intergated | Identity=NetworkService)
    2. Create a web application in IIS using the Application Pool you have just created and having the Physical path the folder where you copied the FTOS Reverse Proxy files.

STEP 2. Configure the FTOS Reverse Proxy

In the FTOS Reverse Proxy installation folder, find the **proxy/proxy.config.js** file and make the necessary changes:

```
//the output refers to the __FintechOS B2C Application__
output.scheme = "http";
output.port = 80;
```

```
output.host = "192.168.15.15";
output.application = "FintechOS_B2C";// if there is no application
(FintechOS B2C runs as a root site in IIS) use output.application =
null
```

Configure your routes following the example from the file.

Make sure that you set the B2C header with the value from the **web.config** file of the FintechOS B2C application, appSetting **feature-b2c-userjourneys-api-key**:

```
output.requestHeaders.set("B2C", "feature-b2c-userjourneys-api-
key");
```

FTOS Reverse Proxy Configuration

```
filter.setup(function (input, output)
 {
    /* LOCAL TEST */
    output.scheme = "http";
    output.port = 60130;
    output.host = "localhost";
    output.application = null;
    /* ----------------------------------------
    // mandatory for Fintech, mapping # to Main#
    // ----------------------------------------
    */
    if (output.path.match(/^#/)) {
                output.path = "Main" + output.path;
                return output.redirect();
                }
    if (output.path === "/rca")
                {
                if (output.query.match(/\bsessionId=/i))
                    {
                    output.path = "/Main";
                    output.query +=
"#/entity/claimNotification/edit/newEntry/data
form/b2cUJ/pageno/1";
                    }
                else
                    {
                    output.path = "/Main";
                    output.query +=
"#/userjourney/claimNotification/insert/data form/b2cUJ";
                    }

       return output.redirect();
```

```
        }

        output.requestHeaders.set("B2C", "feature-b2c-userjourneys-
api-key");

        return output.go();
});
```

When configuring routes use the following templates:

- Insert Link: *#/userjourney/{entityName}/insert/data form/_{formName}_*
- Edit Link: *#/entity/{entityName}/edit/old/data form/{formName}/pageno/{pageNo}*

STEP 3. Enable journey to be accessible through the reverse proxy

Prerequisite: Make sure that the digital journey that you want to expose to unauthenticated users has the wizard mode active.

To enable a specific digital journey to be accessible through the reverse proxy, In Innovation Studio follow these steps:

1. From the menu, click **Digital Frontends** > **Anonymous Frontends Configurations**. The **Anonymous Frontends Configurations** List page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add Anonymous Frontends Configuration** page appears.
3. In the **Name** field, fill-in the name of the B2C domain and in the API Key field, provide the **feature-b2c-userjourneys-api-key** value:
4. At the top-right corner of the page, click the **Save and reload** icon. The page refreshes the **Edit Anonymous Frontends Configuration** page appears. The **Published Form Driven Flows** and **Published Custom Flows** sections will be unlocked. You can expose both data form driven and custom flows to unauthenticated users, as follows:

   Exposing data form driven flows

   Prerequisite: In order to expose a form driven flow to unauthenticated users, you should have created the journey. For information on how to create a form driven flow, see "Creating Form Driven Flows" on page 204.

1. From the **Published Form Driven Flows** section, click the **Insert** existing button. A pop-up appears listing all existing standard user journeys.
2. Double-click on the desired form driven flow that you'd like to expose to unauthenticated users. The pop-up closes and the selected record will be displayed in the **Published Form Driven Flows** section.
3. Save the settings by clicking the **Save and close** icon.

Exposing custom flows

Prerequisite: In order to expose a custom flow to unauthenticated users, you should have created the custom flow. For information on how to create a custom flow, see "Creating Custom Flows" on page 272.

1. From the **Published Custom Flows** section, click the **Insert** existing button. A pop-up appears listing all existing custom flows.
2. Double-click on the desired custom flow that you'd like to expose to unauthenticated users. The pop-up closes and the selected record will be displayed in the **Published Custom Flows** section.

   If you want to expose more custom flows to the current B2C frontend domain, add them in this section; otherwise, they will not be visible to unauthenticated users.
3. Save the settings by clicking the **Save and close** icon.

Once unauthenticated users will complete a digital journey, records are logged in. To see the audit logs of anonymous frontends, from the menu, click **Security** > **Anonymous Journey Access Logs**. The **Anonymous Journey Access Logs List** page appears. To see the B2C frontend domain from where the digital journey has been exposed to unauthenticated users (automatically displayed in the **B2C FrontEnd Domain** non-editable field) and also change the external process status, double-click on the desired record in the list.

STEP 4. Override default Save on the journey with an endpoint

Now that you've set up the B2C environment and you exposed journeys to unauthenticated users, you need to override the default save on the digital journey. For information on how to do that, see How to Override Save With an Endpoint.

STEP 5. Create and use your own styles sheets (optional)

The B2C journeys have by default poor styles, so you might want to create and use your own styles sheets. For more information, see Manage Style Sheets for B2C User Journeys.

STEP 6. Set anonymous frontends to serve in a specific language (optional)

An anonymous frontend can serve in a specific language. For information on how to set anonymous frontends to serve in a specific language, see Serving User Journeys in a Specific Language.

Step 7. Reset an anonymous frontend session

Resetting an anonymous frontend session is useful when running multiple instances of an anonymous frontend to ensure that if customers go back to a previous step, they don’t lose the data that has already been saved in previous steps.

Example

> There is an anonymous frontend for loan application comprised of various steps. In one of the steps, the customer is asked to provide the monthly income based on which the loan value is calculated in a different step.
>
> To register the customer loan application without altering the data based on which the loan amount was calculated, reset the anonymous frontend session by using the ftos.core.resetB2CSession() function in the last step (**Advanced** tab> **After Section Save** tab) of the loan application digital journey.

To reset an anonymous frontend session:

1. In Innovation Studio, log in the **Developer** mode.
2. Go to the configuration page of the digital journey exposed to anonymous users.

3. Click the **Steps** tab and in the **Entity Form Steps** section, double-click the last step of the digital journey (the step with the highest order index).
4. In the step configuration page, click the **Advanced** tab.
5. Click the **After Section Save** tab and in the JavaScript field, type the function `ftos.core.resetB2CSession();`
6. At the top right corner of the page, click the **Save and close** icon to save the step.

Overriding Default Save on Journeys

The default save method on forms will save into the database the data inputted by the user when completing the data form.

If you want to manipulate the data before saving it into the database, you can do so by overriding the default save method.

Overriding the default save method on data form is also useful if the current user does not have privileges on that entity. You can achieve this by using an endpoint (action) in three simple steps:

STEP 1. Create an on-demand automation script

This script will be executed instead of the default one. For information on how to create on-demand automation scripts, see Create On-demand Automation Scripts.

If you want to override the save on forms exposed to unauthenticated users, use this minimal working code in the on-demand script

```
setAdminMode(true);
//log(context);
var activeStatus = getOptionSetItemId("B2CProcessStatus",
"Active");
var entityIdByName = getEntityIdByName(context.EntityName);
var dateName = new Date().toString();
var sessId = server.B2C.SessionId;
var saveData = getEndpointSaveData();
saveData.EntityValues["formName"] = saveData.FormName;
saveData.EntityValues["sectionIndex"] = saveData.SectionIndex;
saveData.EntityValues["sectionName"] = saveData.SectionName;
```

```
if(saveData.OperationType == "edit"){
        update(saveData.EntityName, saveData.Id,
saveData.EntityValues);
}
else if(saveData.OperationType == "insert"){
    var generatedId = insert(saveData.EntityName,
saveData.EntityValues);
    setData({"Id": generatedId});

    var epVals = {
        "sessionId": sessId,
        "name": dateName,
        "entityId": entityIdByName,
        "recordId": generatedId,
        "status": activeStatus
    }
    var B2CExternalProcessId = insert("B2CExternalProcess",
epVals);
}
function getEntityIdByName(name){
    var a = getByQuery({
            entity: {name: "entity", alias: "a"},
            where: {
                type: "and",
                conditionlist: [{
                        type: "equals",
                        first: "a.name",
                        second: "val(" + context.EntityName + ")"
                }]
            }
    });
    return a[0]["a_entityid"];
}
```

Where:

- **saveData.EntityValues** contains the values that will normally be passed to the default save function.
- FintechOS relies on the **sessionId** parameter to retrieve the desired record through the B2C External Process entity.

STEP 2. Create an endpoint and attach the script to it

Create an endpoint (action) and add the automation script created at step 1 to it.

For information on how to create and endpoint and attach an automation script to it, see Create an Endpoint.

STEP 3. Call the endpoint on the form driven flow

In the Before Events of the form driven flow, call the **setSaveEndpoint** function of the formData object, with the name of the endpoint created at step 1:

```
formData.setSaveEndpoint("endpointName");
```

Function getEndpointSaveData()

When overriding the default save functionality using an endpoint, the function retrieves the entity save data and other save context information.

Returns: IEndpointSaveData

```
interface IEndpointSaveData
{
    Id: string;
    EntityName: string;
    OperationType: "edit" | "insert";
    FormName: string;
    SectionIndex: number;
    SectionName: string;
    EntityValues: any;
}
```

Serving User Journeys in a Specific Language

The platform enables you to serve a B2C digital journey in a specific language, that is, the digital journey can be started in a specific language.

Prerequisite

For a B2C journey to be started on a specific language, the language should exist in the system and the resources should be localized in that language. For more information on how to add a language and details on localization resources, see Localization.

Set a journey to serve in a specific language

You have two options for setting up a digital journey to be started in a specific language:

- Launch the digital journey in a specific language by adding the culture query parameter to the proxy URL

  Launching a digital journey in Romanian: *http://proxyurl?culture=ro-RO*

- Configure the FTOS Reverse Proxy by setting the culture. To do so, In the FTOS Reverse Proxy installation folder, find the **proxy/proxy.config.js** file and add the **output.query** property.

```
//the output refers to the __FintechOS B2C Application__
...
output.query += "?culture=ro-
RO#/userjourney/claimNotification/insert/data form/b2cUJ";
```

Once unauthenticated users will launch the digital digital journey, records will be automatically added in Innovation Studio to the B2C External Process entity (**Operations** menu > **B2C External Processes**) and FintechOS engineers will be able to see the culture in which the digital journey has been started.

Manage Style Sheets for B2C User Journeys

When FintechOS runs in B2C User Journey mode, the default style sheets used are fewer and themes have no effect; therefore, we recommend you to create and apply your own style sheets.

To manage the style sheets for digital journeys which are exposed to unauthenticated users via the B2C environment, follow the basic procedure for style sheets management.

NOTEThere are specific style particularities for B2C user journeys, described in the table below.

The following css elements are available for B2C user journeys:

Parameter	Default value	Description
--defaultFontSize	14px	
--defaultTextColor	#333	
--linkColor	#337ab7	
--linkHoverColor	#23527c	
--controlTextColor	#333	The color of the text displayed in controls. E.g.: OptionSet control.
--controlIconColor	#333	
--controlBorderColor	#ddd	
--controlActiveBorderColor	#337ab7	
--errorColor	#d64031	The background color of error toast message.
--infoColor	#2980b9	
--warningColor	#feb332	
--successColor	#049F0C	

Security

Innovation Studio provides a framework to reflect the security profile of your organization. Security design is essential to accomplishing the following:

- Protect information from being mishandled by users.
- Ensure that users have access to information based on business need to know.

To set up the organizational structure, you need to create the business units, security roles, and assign users the appropriate security roles to map the job-related responsibilities with the required level of access privileges within the platform.

The security of our technology rests on four major poles **data encryption, authentication, authorization, data governance and data audit**. FintechOS uses several method for authentication and a role-based access control, data ownership is given by security roles while sensitive data is being protected because it can become anonymous.

For example, to build an internal security mechanism, a user will have to build an organizational chart that includes business units, security roles and users. Create business units beside the root one, configure the security roles that will be given to the users using CRUD privileges, assign users to each role allowing each to see what they need by associating the security role to the digital journey/analytics.

This section covers the following topics:

Business Units **619**
Creating Business Units 619
Managing Business Units 620
Security Roles **621**
Default Security Roles 621
Creating Security Roles 622
Editing Security Roles 625
Users **626**
Adding Users 627
Editing Users 628

Recovering Password (for users) 629

Business Units

Business units are the foundation of the security structure in FintechOS. Each user must to be part of a business unit. When FintechOS is installed, a business unit is created by default, the root business unit. You can rename it, but you cannot remove or disable it.

NOTERoot is an important business unit that comes by default with Fintech. A user configured under the root business unit can see all the records of the entities based on granted access rights.

To define the organization structure, super users can add as many business units as necessary to fulfill the need of configuring several levels of access to information for specific to groups within the organization.

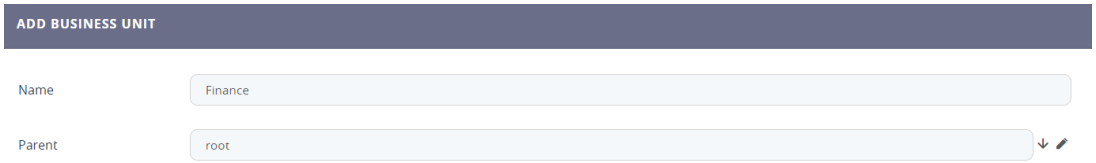
Creating Business Units

Depending on your organizational needs, there will always be only one root business unit and at least one business unit. The root business unit acts as the top level of the organizational hierarchy and all other business units are its children. The root business unit is the company while the child business units are subsidiaries or departments within the company.

To create a business unit, follow these steps:

1. From the menu, click **Security** > **Business Units**. The **Business Units List** page appears.

2. At the top-right corner of the page, click the **Insert** icon. The **Add Business Unit** page appears.
3. In the **Name** field, enter the name of the business unit.
4. From the **Parent** drop-down, click the down-arrow. A pop-up appears listing all existing business units.
5. If this is the first business unit you add, then select root otherwise, select the parent business unit so that it reflects your organization structure. The figure below shows an example on how to add the first business unit.



6. At the top-right corner of the page, click one of the save options. When you finish adding business units, click the **Save and close** icon. The page closes and the business unit will be displayed in the **Business Units List** page.

Managing Business Units

Business units can be edited even after they were created, or removed if there's no more use for them. However, keep in mind that business units containing users cannot be removed.

Editing Business Units

You can edit a business unit from the **Business Units List** page (from the menu, click **Security** > **Business Units**). Double-click the desired business unit, change the name or the parent business unit, then at the top-right corner of the page click one of the save icons. The updates take effect immediately.

Removing Business Units

> **NOTE** You cannot remove business units that contain users. If you try deleting such business units, you will get an error message informing you that the operation has failed.

To remove business units, from the menu, click **Security** > **Business Units**. The **Business Units List** page appears. Select the ones that you want to remove and at the top-right corner of the page, click the **Delete** icon. The updates take effect immediately.

Security Roles

A security role is a set of privileges and the level of access to various actions/functions within the platform. Security roles allow you to configure the security items, that is, the access privileges on CRUD operations for entities who belong to the open data model.

Users with elevated privileges (admin users) can control data access by setting up the organizational structure to protect sensitive data and configuring various organization layers to allow communication, collaboration or reporting.

You can grant even more granular access privileges in FintechOS, by associating security roles to digital journeys, workflows, analytics and Portal Profiles. Such security roles are then associated to a user, hence the user will be able to see those digital journeys, workflows, analytics or Portal Profiles. Such an example is given in "Access to Portal Profiles based on Security Roles" on page 584. The data is automatically filtered based on the privileges and level of access defined within the security role via the security items.

The lowest level of access privileges you can grant to users in FintechOS is on attribute level.

Default Security Roles

The following table describes the access rights level of the default security roles:

Security Role	Description
Debugger Users	This is a development role: it is used by the implementation team to debug issues on the Portal using the Debugger in the kit.
Developer	This is a development role: it is used by the implementation team to create users that access a restrictive part of the designer, not admin user.
Guest	This is a role inherited by the platform; it doesn't have any special platform access meaning.
JobServer	This role is used by the JobServer service to execute scripts from the platform with a specific schedule (see Schedule Jobs).
Registered Users	Users with this security role have access rights to edit their account from My Account and to access a minimum list of entities in order to log in without errors in the application.
User Management	Users with this role can manage the application users without having elevated privileges. **NOTE**System users who have been granted the User Management security role **cannot** manage existing Administrator users.
Integration Users	A role designed for integration with other systems. It is not an actual user, but rather a process that authenticates and calls various functions exposed inside the platform.
Widget	This is a role inherited by the platform; it doesn't have any special platform access meaning.

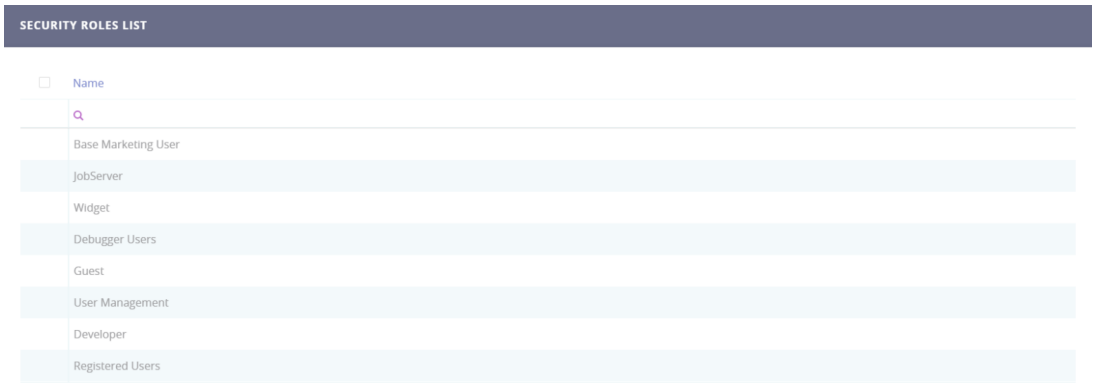
New security roles can be added to the list depending on your business needs and to each give security items with specific CRUD operations. Lastly, attach the security role created to the element you wish to give access to e.g. form driven flow or report.

Creating Security Roles

Configuring a security role is a two-step procedure:

1 Add the security role

1. From the menu, click **Security** > **Security Roles**. The **Security Roles List** page opens.



2. At the top-right corner of the page, click the **Insert** icon. The **Add Security Role** page opens.
3. In the **Name** field, type a name for the new security role.
4. At the top-right corner of the page click the **Save and reload** icon. The **Edit Security Role** page opens.

Now you can start adding security items .

2 Assign security items to the security role

Security items specify the entities and the privileges (CRUD operations on those entities) that the security role grants access to.

Users assigned with a security role will be able to perform only the CRUD operations on entity records assigned in the security items.

To add a security item, follow these steps:

1. From the **Security Items** section, click the **Insert** button. The **Add Security Role Item** page is displayed.
2. In the **Entity** field, type the entity name or click the down-arrow and select it from the list.

3. In the **Security Operation** field, type the record-level privilege (CRUD operation) or click the down-arrow and select it from the list. You can choose one of the following:

CRUD operation	Details
Read	Allows users to view records.
Update	Allows users to update records.
Insert	Allows users to add new records.
Delete	Allows users to delete records.

4. In the **Security Scope** field, type the level of access or click the down-arrow and select it from the list:

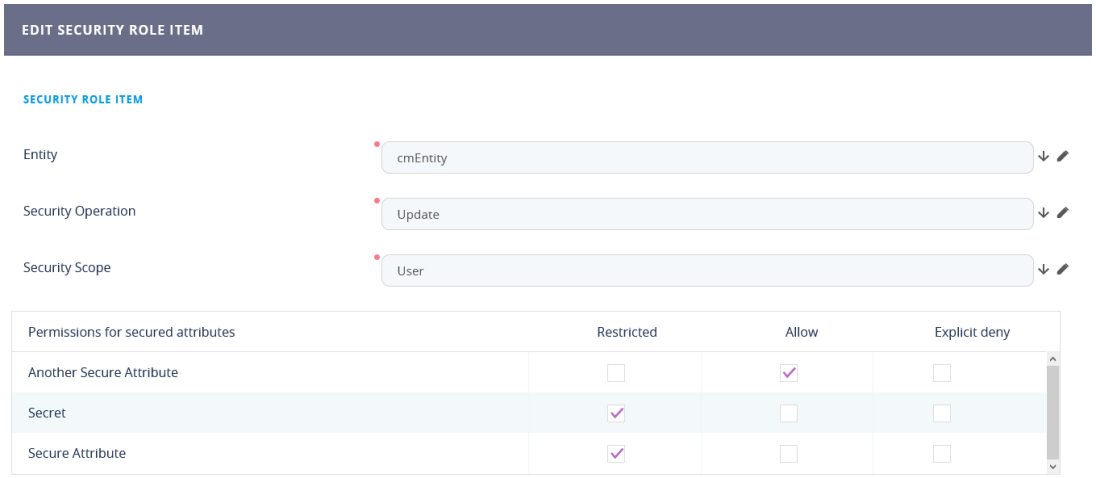
Scope	Details
User	Privileges to the records owned by the user or assigned to the user.
Parental	Privileges to all records owned in the business unit to which the user belongs to, including privileges to the records owned in the child business units.
Business Unit	Privileges to all records owned in the business unit to which the user belongs to.
Organization	Privileges to all records in the organization regardless of their owner.

5. Click the **Save and reload** icon at the top right corner of the screen.

6. If your security role item is an entity with attributes tagged as "isSecurable" on page 71, you can use the **Permissions for secured attributes** table at the bottom of the page to set the following permissions for the entity's secured attributes:

Permission	Details
Restricted (default)	Prevents access to the attribute.

Permission	Details
Allow	Allows access to the attribute.
Explicit deny	Prevents access to the attribute even if another security role allows access.

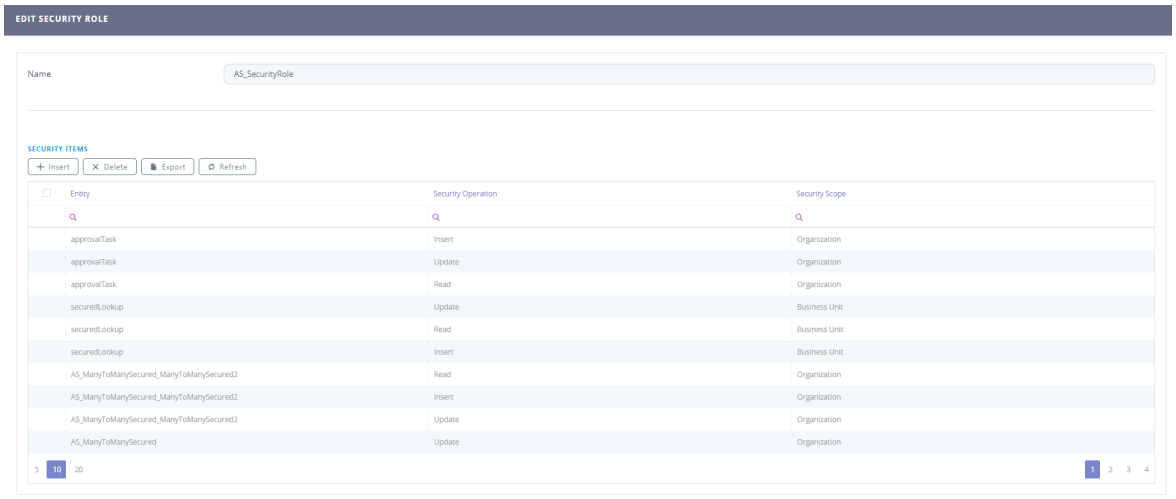


7. Click **Save and close** at the top right corner of the page.

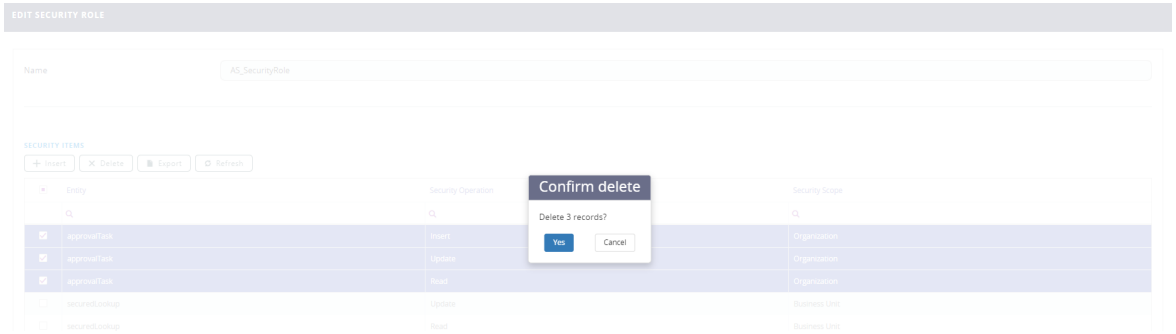
Repeat the process for additional security items you wish to assign to the security role.

Editing Security Roles

To edit a security role, from the menu, click **Security** > **Security Roles**. The **Security Roles List** page appears. Double-click the security role that you want to edit. The **Edit Security Role** page appears. You can edit security roles by changing their name or by adding or removing security role items.



To remove a security item, scroll-down to the **Security Items** section, select he item that you want to remove and at the top of the Security Items section, click the **Delete** button. A confirmation dialog appears. Click **Yes** and the selected security item will be removed from the system.



Make the desired security role changes and at the top-right corner of the page, click the **Save and close** icon to save the changes.

Users

One or more security roles can be associated at user level, enabling a simple process for promoting or revoking rights. You can grant access to the platform to both users within your organization and to persons outside your organization who use company data to make decisions (external users).

Users have access to the platform functionality based on the security role assignment. By default, all users are able to view and manage their account data in the **My Account** section.

NOTEOnly users with elevated privileges (admin account) can manage users.

In FintechOS, user type is a grouping for the users based on platform high level access (to not be confused with system roles).

The platform distinguishes three user types:

User Type	Description
Back Office	The user type that all users have when created. It does not have a special access scope, it is just a category.
Guest	It is just a category, it does not have any special platform access meaning. It has only a validation, there can't be two users with this type.
DNN Portal	Used in implementation if there is a DNN Portal created that gets data data form Fintech.

Adding Users

To add a new user, follow these steps:

1. From the menu, click **Security** > **System Users**. The **System Users List** page appears.
2. At the top-right corner of the page, click the **Insert** icon. The **Add System User** page appears.
3. In the appropriate fields, provide the user credentials the user will use to log into the platform (Username, Password, Confirm Password).
4. From the **Business Unit** drop-down, select the business unit to which the user belongs to.

> **NOTE** Root is an important business unit that comes by default with Fintech. A user configured under the root business unit can see all the records of the entities based on granted access rights.

5. Select the **System User Type** by selecting from existing user types or insert new ones based on your needs.
6. If you want the user to have full access privileges within the platform (Admin user), tick the **Is Administrator** checkbox.
7. Activate the user by ticking the **Is Authorized** checkbox.
8. At the top-right corner of the page, click the **Save and reload** icon. The **Edit System User** page appears.
9. From the **Security Roles Role** section, define the role assignment by clicking the **Insert existing** button and selecting the desired security role. A user can have multiple role assignments. The user will have the access privileges and the level of access as defined by the security items within the selected security roles.
10. At the top-right corner of the page, click the **Save and close** icon to save the user updates.

You will have to pass the credentials to users and recommend them to change the password at first login.

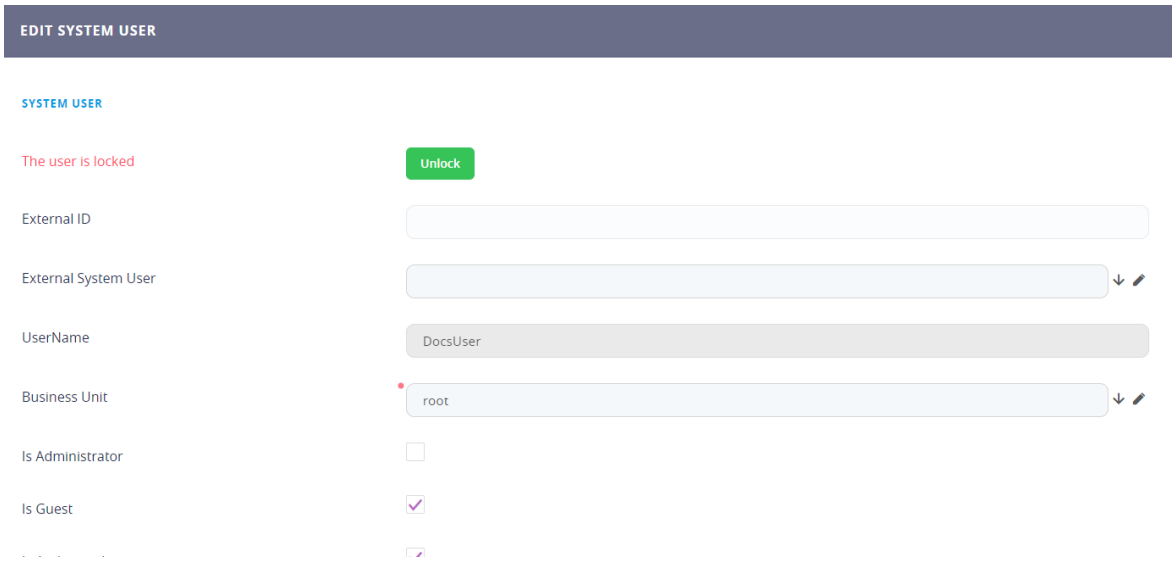
Editing Users

You can edit users by changing their details, by adding new security roles, editing existing ones or removing security roles.

To edit a user, from the from the menu, click **Security** > **System Users**. The **System Users List** page appears. Double-click the user you want whose details you want to edit. The **Edit System User** page opens. Make the desired changes and click the **Save and close** icon.

NOTEThe username field is read-only, you cannot edit it.

Unlock user account

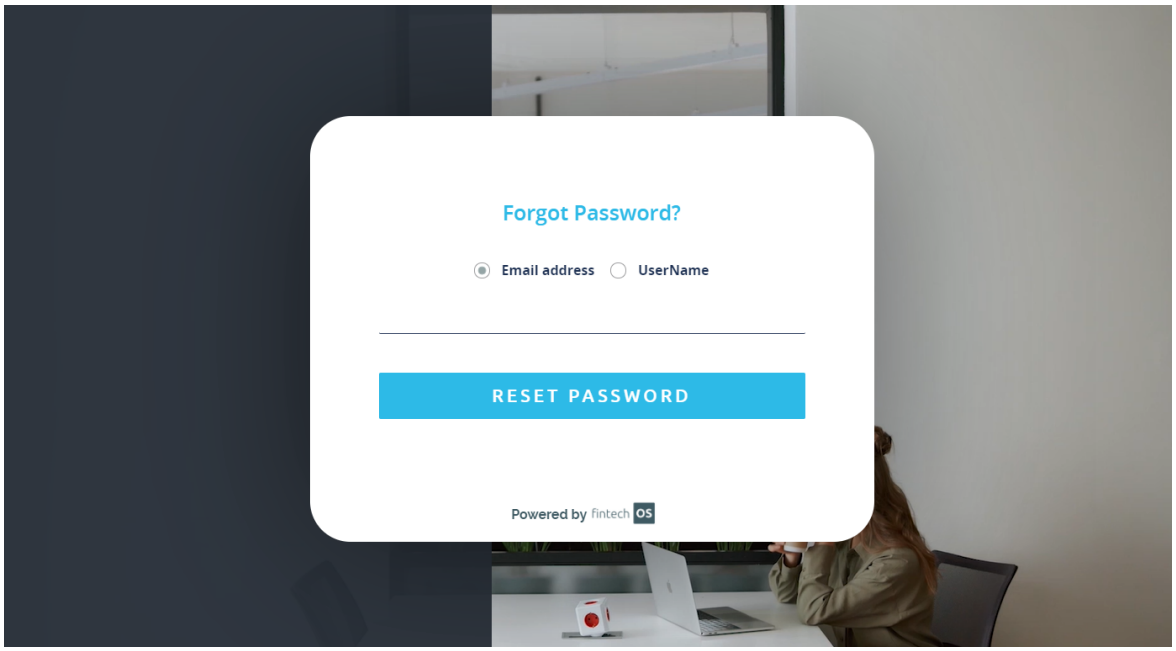


If users have locked their account by entering a wrong password many times, reaching the maximum number of password retries, users with elevated privileges (admin account) can unlock their account from the system user's configuration page, by clicking the **Unlock** button.

Recovering Password (for users)

NOTEIn order for users to recover their FintechOS password, the forgot password feature should be activated. If you are not able to reset your password, please contact your system administrator or the person who manages you FintechOS app.

To reset your password, on the login page click the Forgot password? link. You will be prompted to enter your e-mail address or your username. In the Recover password screen, enter the email address or the username associated with your FintechOS account.



Click **Send**.

You will receive an email with instructions on how to change your password.

If multi-factor authentication has been activated, following the instructions received by email, when providing a new password and confirming it, you will have to enter the security pass code received via SMS on your mobile phone when requesting for a new password.

NOTEThe forgot password functionality does not work for LDAP or Azure AD authentication. If you forgot your domain password, contact your system administrator to reset it.

If you haven't received an email for password reset,

- Check your email spam/junk folder.
- Make sure that the email address: noreply@fintech.com is not blocked or make sure that all emails from this email are always delivered.
- If the above do not work, try contacting your email service provider. They are most likely blocking emails from FintechOS from being delivered.

DevOps

Deployment packages allow users with elevated privileges (admin users) to export metadata, entity data, and report templates from an environment and import them into another environment. Packages are text-based so they can be version controlled to have their history inspectable with text-diff tools.

This section covers the following topics:

Exporting a Deployment Package **632**
1 Create deployment package 632
2 Add Components to the deployment package 633
3 Export deployment package 641
Importing a Deployment Package **642**
Prerequisites: 642
Creating Enhanced Deployment Packages **644**
Deployment Package File 644
Deployment Package Folder 644
ReportTemplates Subfolder 646
Viewing Deployment Package Logs **647**
Sorting Criteria used for XML Sibling Elements **648**
Configuration Data Definitions **651**
Configuration Data Deployment Package **657**
Import data using DevOps **659**
1 Deployment package 659
2 Configuration Data Deployment package 660

Exporting a Deployment Package

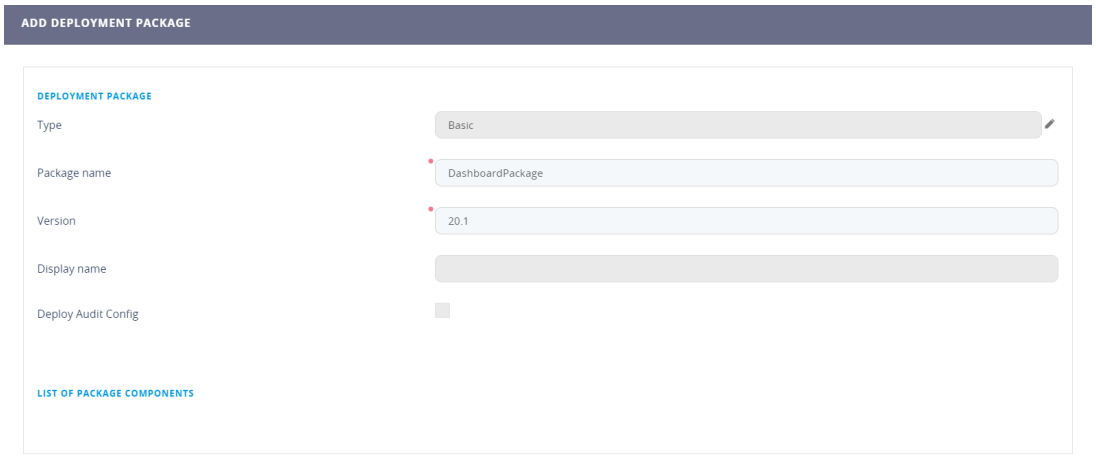
The deployment package menu entry exports the metadata from a source environment (and its database) to a destination environment (and a different database). This specific menu entry imports the package as well when accessed on the destination environment.

To export a deployment package from the **Deployment Packages** section, follow these steps:

> ⚠ **IMPORTANT!**
> While using the Deployment Package feature, you can only export what is implemented in FintechOS using the user interface.

1 Create deployment package

1. From the main menu, click **DevOps** > **Deployment Packages**. The **Deployment Packages List** page opens.
2. At the top-right corner of the list, click the **Insert** icon. The **Add Deployment Package** page opens.
3. Fill in the **Package Name** and **Version** fields.



4. At the top-right corner of the page, click the **Save and reload** icon. The **Edit Deployment Package** page opens. The other remaining fields will be automatically auto-filled by the system.

Now you can add the metadata to be included in the deployment package.

2 Add Components to the deployment package

1. From the **List of Package Components** section, click the **Insert** button. The **Deployment Package Component** page opens.
2. From the **Type** drop-down list, select the metadata type:

Field	Observations
Action group	n/a
Attribute	

Field	Observations
Business Entity	When exporting an entity, which has a lookup attribute, the user must include the entity the attribute is linked to (if the entity is not found on the destination environment). If it is not included, the import will not be successful. If the entity has an attribute type option set, it will extract the list of options as well. It will extract as well the views, data events (after and before), the extended model. ⚠ **IMPORTANT!** To deploy the extended model of an entity you need to export that entity as well. It will not extract the business workflow attached to the entity it has to be added from the option below.
Business workflow	

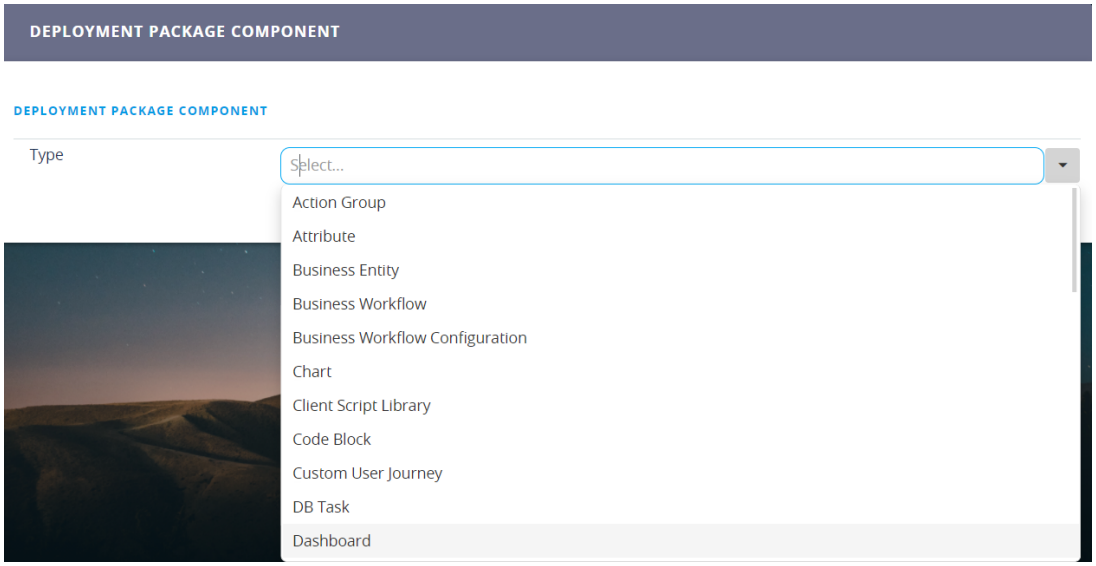
Field	Observations
Business workflow configurations	Unlike the previous component, here you can add just the statuses of a workflow.
Chart	
Client script library	
Code block	
Custom Page Template	Templates used to customize various user interface components, such as session expiration pages or login screens. For more information, see "Custom Page Templates" on page 586.
Custom user journey	
DB Task	
Dashboard	
Data Config definition	
Data Import Template	
Data maping	
Data model	
Data set	This is a data set used in "Data Sets" on page 425 for a business formula.

Field	Observations
Digital journey	With this type, it exports the form driven flow as well. Therefore, to export a form driven flow, attach it to a digital journey. When it will be imported, it will impoert the flow and the journey.
Email template	
Endpoint	This deploys the server-side scrip attached to the endpoint as well. There is no need to attach the script separately.
Entity form	
Entity menu section	
Entity Status	
Entity view	
Formula	
HTML widget	
Menu item	
Option set	
Portal profile	
PowerBI report	
Report	
Report document	
Script library	
Sequencer	

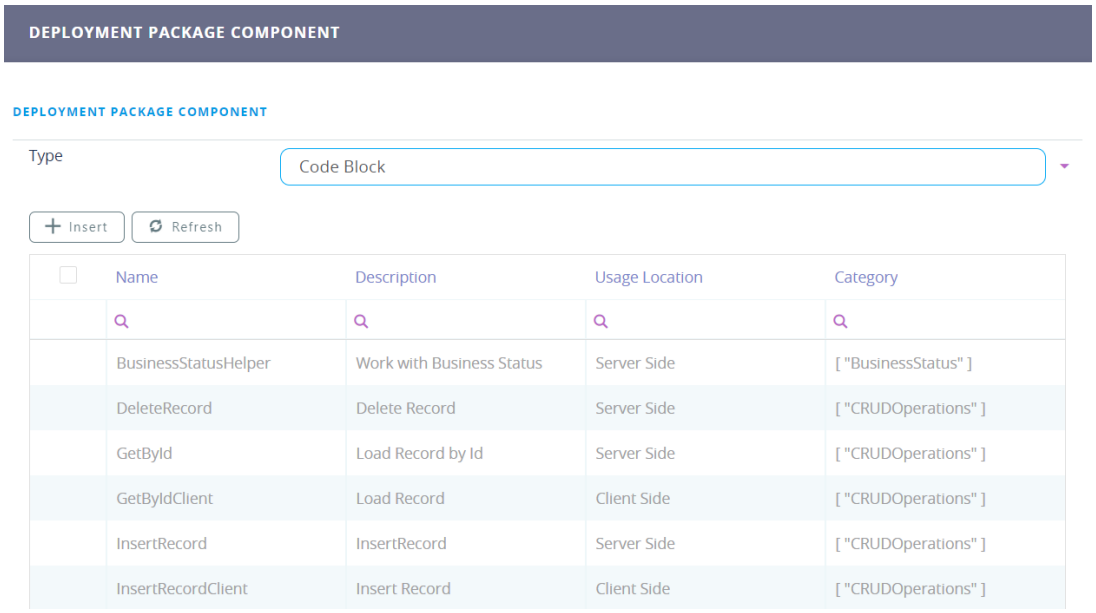
Field	Observations
Server script	
Style sheets	
System parameter	
User competence	
Web API client library	

HINT

The package will contain the configurations created in FintechOS, not the records inserted in the FintechOS Portal. To export the records use, "Configuration Data Definitions" on page 651 and "Configuration Data Deployment Package" on page 657.



3. Below the **Type** field, the list of metadata matching the selected type is displayed. The figure below shows an example of **Code Blocks** records.



4. From the list displayed, select only the components you want to add to the deployment package. You can add multiple components to the package by selecting the desired records from the components list and clicking the **Insert** button.
5. Click the **Insert** button again. The selected metadata is displayed in the **List of Package Components** section.
6. Click the **Save and reload** button. Now you can export the deployment package.

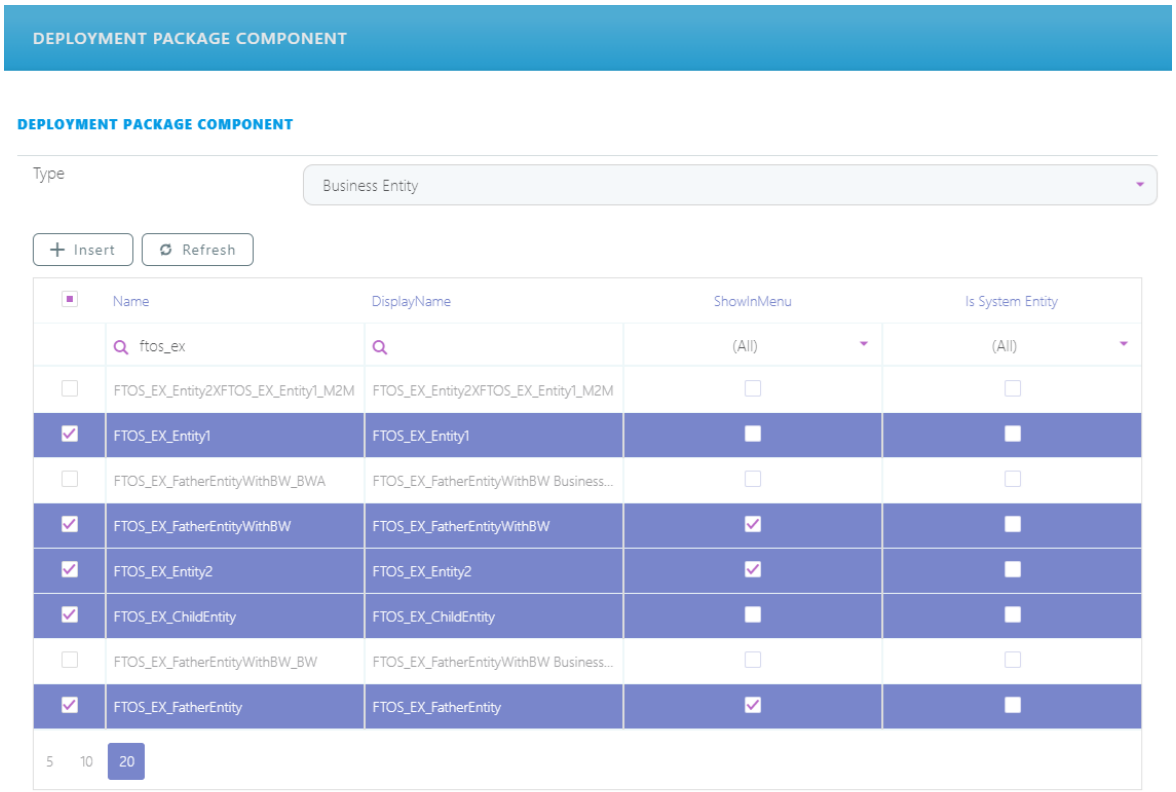
If you need to add more components to the package repeat the steps provided above.

NOTE

- You can add a component to a deployment package only once.
- When adding a metadata of type **Entity** to a deployment package, on export, the system exports all the metadata related to that entity (entity forms, entity views, attributes, option sets related to the attributes created and all the referenced relations); therefore, you do not have to add the entity related metadata from the components list.

- If you want to export entity related metadata (entity forms, entity forms or attributes) without exporting the entity on which the metadata was created, make sure that on the destination environment that entity exists.
- When creating a deployment package, all the referencing metadata found in lookup relationships must be included in the package or must exist on the destination environment.
- When adding a metadata of type **Server Script** to a deployment package, on export, the system exports all the script libraries related to that server script.
- The system does not export the **Audit check**; therefore, you will have to manually configure it on the destination environment.
- The system does not export the system user roles associated to metadata (e.g., roles on custom flows, dashboards etc.)
- **DO NOT** add to a deployment package entities that have been automatically created by the system (e.g., `_ADT tables`, `_BW` and `_BWA tables`), otherwise issues might occur.

  When exporting **Report Documents**, the template is not exported, so make sure that you attach the template file again in the destination environment in order to save it on the server or that you use an enhanced deployment package (see Creating Enhanced Deployment Packages for reference).
- Validations are done only on deployment package import.



HINTMany-to-many Relationships Need to Know

- **DO NOT** add to a deployment package entities that were automatically created by the system when defining a many-to-many relationship, otherwise issues might occur.
- When exporting entities that have a many-to-many relationship defined, make sure that all the referencing metadata found in this type of relationship exist in the deployment package created.

HINT Business Workflows Need to Know

- **Always add** the business workflows associated on the entities that were added to the package, otherwise, although allowed on import, only the `businessStatusId` and `PreviousBSId` attributes will be imported and the system will not create the business workflow entities on the destination environment.
- If an entity has business workflows associated, **DO NOT** add to a deployment package the business workflow entities that were automatically created by the system (_BW, _BWA tables), otherwise issues might occur.

3 Export deployment package

At the top-right corner of the **Deployment Packages List** page, select the deployment package you want to export and at the top-right corner of the page, click the **Export baisc deployment package**.

The deployment package is saved as an Excel file `.xlxs` type, locally in the default download folder on your computer/device.

The package also contains the audit information of entities, as follows:

- The `IsAudited` property of audited entities (the entities which have the **Audit Enabled** checkbox ticked).
- Information of entities storing audit logs (`Entity_ADT`).

The Excel file `.xlxs` type contains the names of the two attributes (**dataSourceType** and **language**) instead of their GUIDs and the attributes on the `ReportDocument` entity are provided in the `DocumentReport` tag instead of being separate tags:

```
<DocumentReport name="Rap No Format
Fetch"

dataSourceType
="Entity" language="English" AllowEdit="false" AllowDelete="false">
```

Importing a Deployment Package

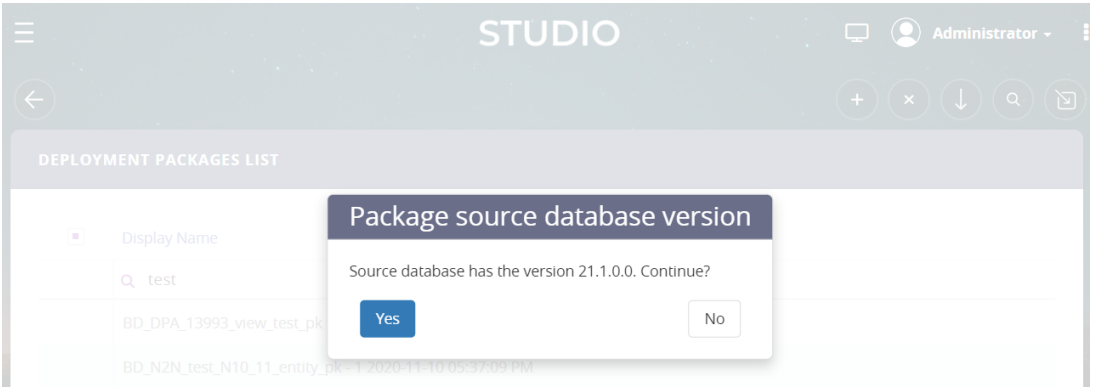
Once you import the deployment package into another environment, you can extend/enrich the imported metadata without altering the data already imported. If the metadata imported is modified by any means, update issues might occur if another user imports the same deployment package again or another version of it. The deployment package is used to update or insert metadata; it does not delete metadata.

Prerequisites:

- Make sure that the language set in FintechOS on the browser is English.
- If you want to import deployment packages in a FintechOS version which has XSS prevention enabled, make sure that the deployment packages and the new metadata do not contain the characters ”<”, “</”, “>”, “<” or “>”next to text, otherwise, on import, you will get an error message and you will not be able to import the packages.

To import the deployment package in another environment, on that destination environment, in Innovation Studio follow these steps:

1. From the main menu, click **DevOps** > **Deployment Packages**. The **Deployment Packages List** page opens.
2. At the top-right corner of the page, click the **Import deployment package** icon. The **Open** pop-up appears.
3. Browse for the deployment package (`xml` file) or enhanced deployment package (`zip` file) that you want to import, select it and click **Open**. A confirmation dialog appears that indicates the version of the database used by the source environment (from where the deployment has been exported):



4. Click **Yes** to confirm the import.
5. You will also be asked if you want to deploy the audit configurations.

The system checks the deployment package metadata and if no errors occur, the metadata is successfully added into the database.

If errors occur during the import, the system will throw two types of errors:

- **Blocking errors**: The data is not imported and an error message highlighted in red will be displayed informing you on what the problem is.
- **Warning errors**: Warnings are logged in the **Deployment Package** log, if any. To see the warnings, consult the log of the selected deployment package. For more information, see"Viewing Deployment Package Logs" on page 647.

If on the last confirmation dialog, you have chosen to deploy the audit configurations:

- on insert, the entity will be inserted with **IsAudited** value from the package (or false if the field does not exist in the package).
- on update, **IsAudited** field will be updated with the **IsAudited** value from the package.

If on the last confirmation dialog, you have chosen not to deploy the audit configurations:

- on entity insert, the **IsAudited** field will be always set to **False**.
- on entity update, the **IsAudited** field remains unchanged.

Creating Enhanced Deployment Packages

Such packages are useful when importing xml files in Innovation Studio. If, in addition to metadata, you wish to include entity data and report templates in a single deployment package, you need to create an enhanced deployment package. An enhanced deployment package is a `.zip` archive that bundles together metadata, entity data, and report templates in a standardized format.

The enhanced deployment package `.zip` archive must have the following structure:

```
Deployment_Package_Name.xml
Deployment_Package_Name
    DataConfigImport
        DataConfigImport.xml
        Data_Import_File.xlsx
        ...
        Another_Data_Import_File.xlsx
    ReportTemplates
        Report_Template_File.docx
        ...
        Another_Report_Template_File.docx
```

Deployment Package File

The deployment package file (`Deployment_Package_Name.xml` in the example above), contains the metadata exported from the source environment. For details, see "Exporting a Deployment Package" on page 632.

Deployment Package Folder

The deployment package folder name must match the deployment package file name without the extension (`Deployment_Package_Name` in the example above).

DataConfigImport Subfolder

The `DataConfigImport` subfolder contains the import data set files (`Data_Import_File.xlsx` and `Another_Data_Import_File.xlsx` in the example above) and a file called `DataConfigImport.xml` that maps the import data set files to the data import templates and entities on the destination environment.

For details on how to export the data set files (`Data_Import_File.xlsx` and `Another_Data_Import_File.xlsx` in the example above) from the source environment, see Data Exports.

DataConfigImport.xml File

The `DataConfigImport.xml` file maps the import data set files to the data import templates and entities on the destination environment:

```
<DataConfigImportSet xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" name="DataConfigImportName" version="1.0.1">
    <DataConfigImport dataImportTemplateName="AA01_
DIT"  entityName="AA01_Ent" fileName="Data_Import_
File.xlsx"       rollbackOnError="true"/>
    <DataConfigImport dataImportTemplateName="AA01_
DIT"  entityName="AA01_Ent" fileName="Another_Data_Import_
File.xlsx" rollbackOnError="true"/>
    <DataConfigImport dataImportTemplateName="AA01_
DIT2" entityName="AA01_Ent" fileName="Data_Import_File_
3.xlsx"       rollbackOnError="true"/>
    <DataConfigImport dataImportTemplateName="AA01_
DIT2" entityName="AA01_Ent" fileName="Data_Import_File_
4.xlsx"       rollbackOnError="true"/>
    <DataConfigImport dataImportTemplateName="AA01_
DIT2" entityName="AA01_Ent" fileName="Data_Import_File_
5.xlsx"       rollbackOnError="true"/>
</DataConfigImportSet>
```

If a file mentioned in the `DataConfigImport.xml` file is not found in the container folder and its **rollbackOnError** is set to true, the import is stopped. Files are uploaded in the order defined in the `DataConfigImport.xml` file.

ReportTemplates Subfolder

The `ReportTemplates` subfolder may contain report template files that are uploaded if they appear in a deployment package:

```
<Reports>
    <Report name="AA01_Report" displayName="AA01
Report" typeName="Document" scopeName="Entity" entityName="AA01_
Ent" destinationFileName="AA01_ReportFile" destinationField="AA01_
Ent_AttrFile" outputMethodName="Attach to
entity"

documentReportType
="PDF"

alwaysReturnFile
="false"

orderIndex
="1" showInMenu="false" AllowEdit="false" AllowDelete="false"><!
[CDATA[        ]]>
    <ReportParameters />
    <ReportItems>
        <ReportItem isDefault="true" name="AA01_Report : 1/3/2020 -
3/3/2020" reportName="AA01_Report" reportEntityName="AA01_
Ent" startDate="2020-02-29T22:00:00.0000000" endDate="2020-03-
02T22:00:00.0000000" AllowEdit="false" AllowDelete="false">
        <DocumentReport name="AA01_
ReportDoc"

dataSourceType
="entity" language="English" AllowEdit="false" AllowDelete="false">
            <Template>
[{"Name":"DocTemplate.docx","RealName":"DocTemplate_aac79c6c-61bf-
4515-8328-5d38a9f387ab.docx"}]</Template>
            <fetchCollection />
        </DocumentReport>
        </ReportItem>
    </ReportItems>
    </Report>
</Reports>
<DocumentReports>
```

```
    <DocumentReport name="AA01_
ReportDoc"

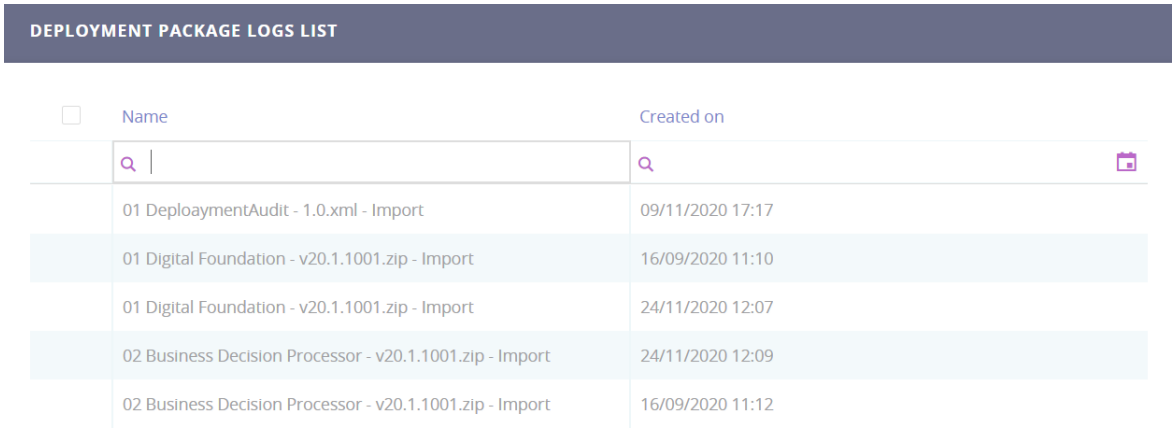
dataSourceType
="entity" language="English" AllowEdit="false" AllowDelete="false">
    <Template>[{"Name":"DocTemplate.docx","RealName":"DocTemplate_
aac79c6c-61bf-4515-8328-5d38a9f387ab.docx"}]</Template>
    <fetchCollection />
    </DocumentReport>
</DocumentReports>
```

If a report template file is mentioned in the deployment package, it is searched in the `ReportTemplates` folder and, if found, uploaded before the **Document Report** import. The template is imported with the real name created upon that upload. This is because files are uploaded/stored with a modified name called real name obtained by adding a Guid like suffix which ensures its uniqueness.

Viewing Deployment Package Logs

When importing a deployment package into an environment, the output of the checks performed during the import are saved into a log specific to the deployment package.

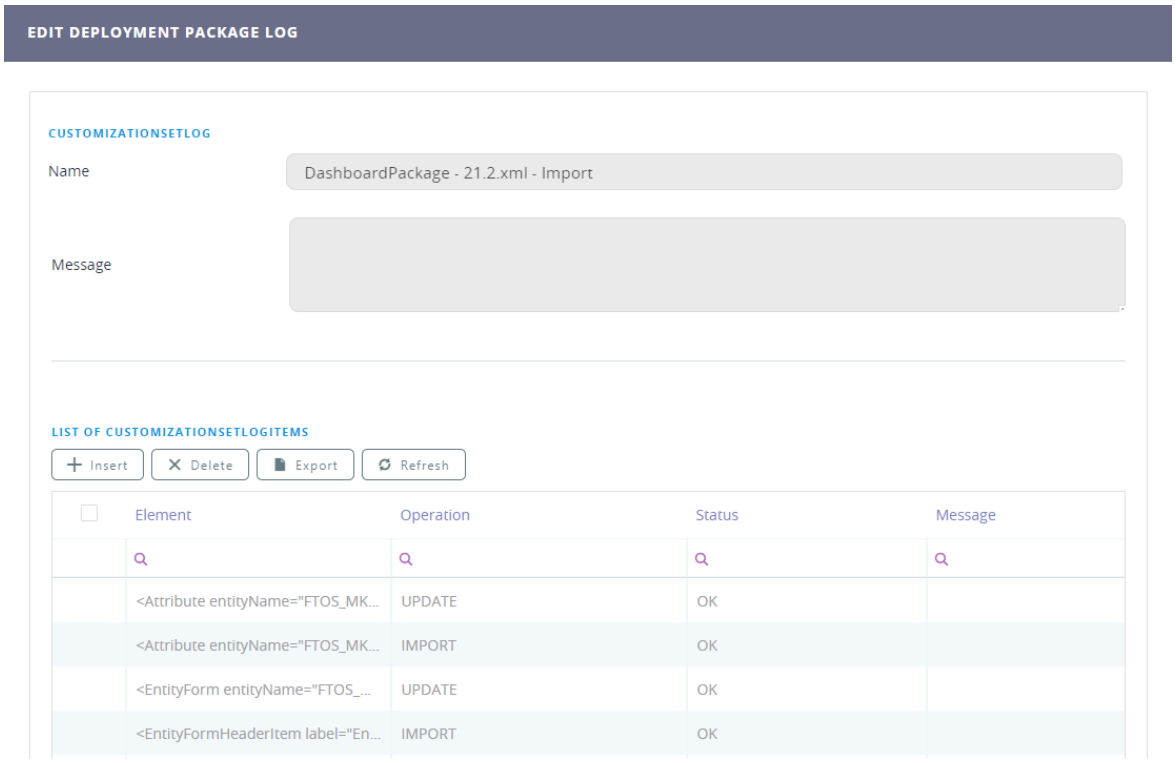
To view the log of a deployment package, from the menu, click **DevOps** > **Deployment Package Logs**. The **Deployment Package Logs List** page opens.



The **Deployment Package Logs** list contains a table with two columns: the name of the deployment package (the name of the `xml` file), and the date when it was imported/or the date when the user tried importing the file.

Double-click on the desired deployment package name. The **Edit Deployment Package Log** page opens.

> **NOTE** You cannot edit fields in this page.



The **List of CustomizationSetLogItems** section displays the status of all the deployment package components being imported. If the import for a component has failed, the reason for the import failure is displayed in the **Message** column.

Sorting Criteria used for XML Sibling Elements

The definition of well-formedness as per the XML 1.0 specifications states that attributes are unordered, but it does not guarantee the order of elements.

The order of XML elements is important especially when you have sibling XML elements (or elements with child elements).

To overcome the risk of improperly ordered elements when importing and exporting XML packages, XML elements are ordered by element name and primary attribute.

The table below lists the criteria used for sorting XML elements that are siblings:

XML element path	Criteria
/EmailTemplates/EmailTemplate	TemplateName
/CustomActions/CustomAction	Name
/OptionSets/OptionSet	Name
/OptionSets/OptionSet/OptionSetItems/OptionSetItem	Name
/EntityStatuses/EntityStatus	Name
/Actions/Action	Name
/ActionGroups/ActionGroup/Actions/Action	Name
/ActionGroups/ActionGroup	Name#EntityName#EntityFormName
/EntityMenuSections/EntityMenuSection	Label
/DataImportTemplates/DataImport	Name#EntityName
/DataImportTemplates/DataImport/DataImportAttributes/DataImportAttribute	ColumnName#AttributeName
/SystemPramaters/SystemParameter	Name
/Entities/Entity	Name
/Forms/EntityForm and all other places where EntityForm appears in a list	Name#EntityName
/Forms/EntityForm/Sections/EntityFormSection	Name
/Forms/EntityForm/HeaderItems/EntityFormHeaderItem	Label
/Forms/EntityForm/FooterItems/EntityFormFooterItem	Label
/Forms/EntityForm/FilteredFields/EntityFormFilteredField	AttributeToFilter#AttributeToFilterBy#AttributeToFilterReference
/Forms/EntityForm/OptionFields/EntityFormOptionField	AttributeName

XML element path	Criteria
/Views/EntityView and all other places where EntityView appears in a list	Name#EntityName
/Views/EntityView/EntityViewColumns/EntityViewColumn	Label
/Attributes/Attribute and all other places where Attribute appears in a list	Name#EntityName
/Charts/Chart	Name
/Dashboards/Dashboard	Name
/Reports/Report	Name
/Reports/Report/ReportItems/ReportItem	Name
/Reports/Report/ReportItems/ReportParameter	Name
/Workflows/Workflow	Name
/WorkflowLibraries/WorkflowLibrary	Name
/Workflows/Workflow/Libraries/WorkflowLibrary	Name
/DocumentReports/DocumentReport	Name
/BusinessWorkflows/BusinessWorkflow	Name
/BusinessWorkflows/BusinessWorkflow/Statuses/Status	Name
/BusinessWorkflows/BusinessWorkflow/Rules/Rule	Name
/Relationships/Relationship	Name#ReferencedEntityName#ReferencedAttributeName #ReferencingEntityName#ReferencingAttributeName #RelationshipType#RelationshipConstraint
/Entities/Entity/Relationships/Relationship	Name#ReferencedEntityName#ReferencedAttributeName #ReferencingEntityName#ReferencingAttributeName #RelationshipType#RelationshipConstraint

XML element path	Criteria
/CoreSettings/CoreSetting	Name
/ClientScriptLibraries/ClientScriptLibrary	Name
/HtmlWidgets/HtmlWidget	Name

Configuration Data Definitions

Configuration Data Definitions allow advanced data exports, by defining the relevant entity attributes along with rules to mitigate duplication, referential integrity, versioning, business statuses, etc. on the destination environment at import. Thus, the data can be shared between different environments quickly and easily.

After creating the configuration data definition, you can include it in a "Configuration Data Deployment Package" on page 657 where you can select the individual records that match your configuration data definition which you wish to export.

To create a configuration data definition:

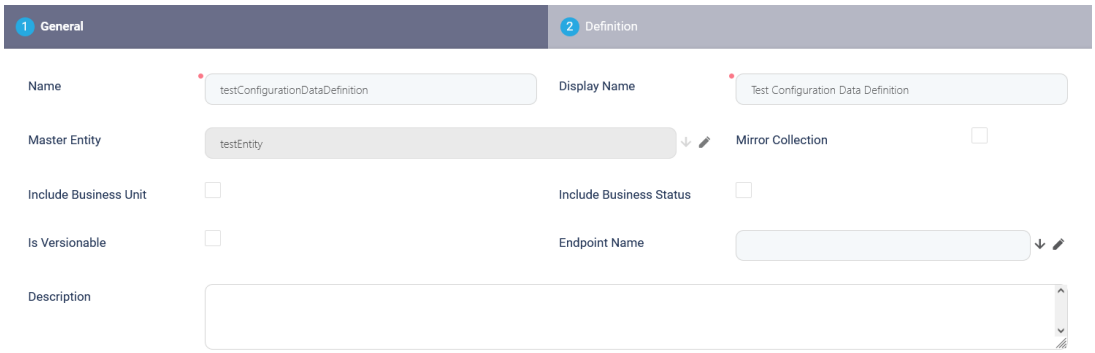
1. Open Innovation Studio and select **DevOps** > **Configuration Data Definitions**. The **Configuration Data Definitions List** opens.
2. Click **Insert** at the top-right corner of the screen. The **General** configuration page opens.
3. Fill in the following fields:

Field	Required	Description
Name	Yes	Name of the Configuration Data Definitions settings.
Display Name	Yes	Name shown in the user interface.
Master Entity	No	The entity on which the data export is built. This entity must have a unique constraint set.

Field	Required	Description
Mirror Collection	No	This option enables replacing the same records with the same name and values. It updates the data on import. For example, for a N-to-N relationship, if the bool is not ticked, and in the source entity there is an attribute cash and in the target entity there is an attribute card, after import the target entity will have both cash and card. If the bool is ticked, the cash attribute will stay and the system will delete the card reference.
Include Business Unit	No	This includes the security elements set for the data.
Include Business Status	No	When a business workflow is attached to the entity, the status will be exported in the file. Each row has a status. For example the statuses can be: • Previous Status • Current status • Next Status (for a predefined workflow). **NOTE** This setting is mutually exclusive with **Is Versionable**.

Field	Required	Description
Is Versionable	No	If the master entity is versionable, you can mark this in the configuration data definition (see "Entity versioning" on page 539 for details). If checked, when entity records are imported on the destination environment, depending on the matching records found on the destination environment (records with the same unique constraints), the following apply: • If no matching record is found, a new record in the Draft state is created. • If a matching record in the Draft or Version Draft state is found, the import will fail. • If a matching record in the Approved state is found, a new record in the Version Draft state is created. **NOTE** This setting is mutually exclusive with **Include Business Status**.

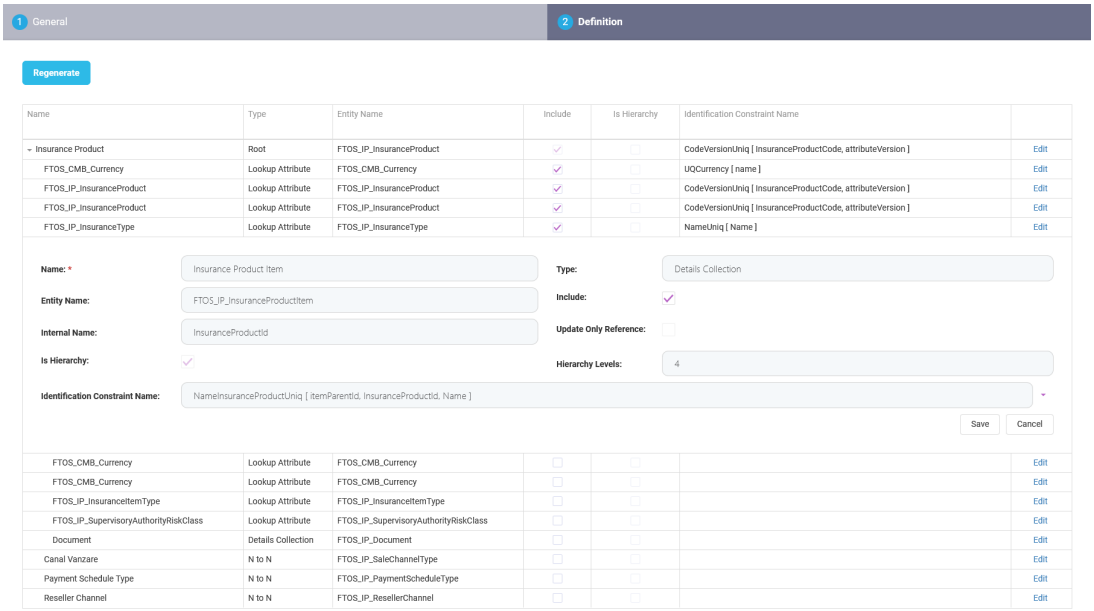
Field	Required	Description
Endpoint Name	No	Allows you to trigger a server side script with the specified endpoint name on the destination environment on import. The IDs of the inserted master entity records will be passed through the `context.Data.packageVersionIds` object, for example: `var packageVersionIds = context.Data.packageVersionIds; for (var i = 0; i < packageVersionIds.Count; i++) { var id = packageVersionIds[i]; }`
Description	No	Insert details about the configuration data definition here.



4. Click **Save and Reload** and open the **Definition** tab to configure the master entity's relationships.
5. If the master entity's data model has been updated since the last time you edited the configuration data definition, click **Regenerate** to update the grid.
6. Click the **Edit** button at the right side of each attribute, and fill in the following information:

Field	Required	Description
Name	Yes	This is the name of the attribute.
Type	No	This is the type of attribute: • Root - Indicates the master entity. • Lookup Attribute - Indicates a lookup attribute with a Dictionary lookup relationship type. • Details Collection - Indicates a lookup attribute with a IsChildOf lookup relationship type. • N to N - For many-to-many relationships. • Parent - Indicates an entity that references the master entity.
Entity name	No	Name of the referenced entity (including the master entity).
Include	No	Tick if you wish to include the attribute in the export.
Internal name	No	It is made up of the Name + "v"+Version + exportDate.
Update only reference	No	This field is mandatory, for parent-child relationship between entities.
Is Hierarchy	No	When referencing an entity with an IsChildOf relationship type, if the referenced entity includes a self-referencing lookup attribute, this box is automatically checked.

Field	Required	Description
Hierarchy Levels	No	When the Is Hierarchy box is checked, use this field to set up how many levels of self-referencing are allowed for the referenced entity. Default value: 4.
Identification constraint name	No	Select the constraint created earlier. For more information, see "Entity Unique Constraints" on page 78.



NOTE

If the entity A has a lookup attribute to another entity B or entity A is a child to entity B that in turn is parent to A, those attributes will be shown in the file, but if they are not included, the records will be empty i.e. will reference as none.

7. Click **Save** to add the attribute or click **Cancel** to cancel the process.

   Repeat for as many attributes as needed.

To export the package follow the steps from the "Configuration Data Deployment Package" below page.

Configuration Data Deployment Package

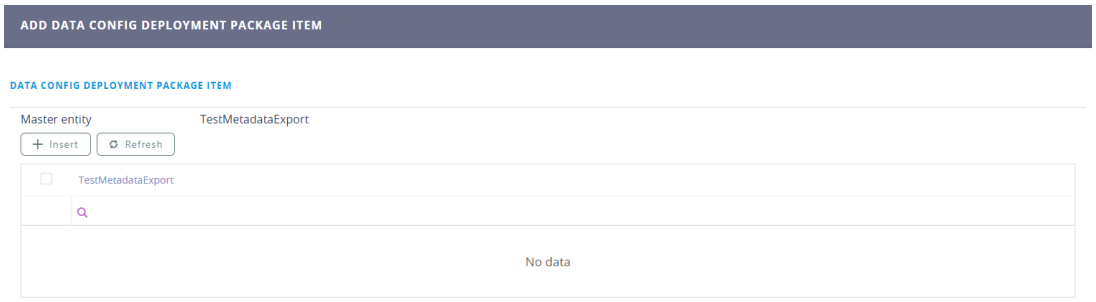
This type of package exports and imports records inserted into the platform. The advantage of this feature is that you can effortlessly export data and then import it into a different database or on the same database, but on a different environment. To export the package from a Innovation Studio, follow these steps and watch the video.

1. Open **Innovation Studio** and select **DevOps** > **Configuration Data Deployment Packages**. The **Configuration Data Deployment Packages** List opens.
2. Click the **Insert** icon. The **Add Configuration Data Deployment Package** page opens.
3. Fill in the following fields:

Field	Required	Data type	Description
Name	No	Text	Name of the package.
Version	Yes	Text	Name + "v"+Version + exportDate.
Display Name	Yes	Text	Name shown in the Portal.
Configuration Data Definition	No	Lookup	Once it is saved, it cannot be edited.
Was imported	No	Bool	This marks the confirmation of import.
Description	No	Text area	Insert the details here.

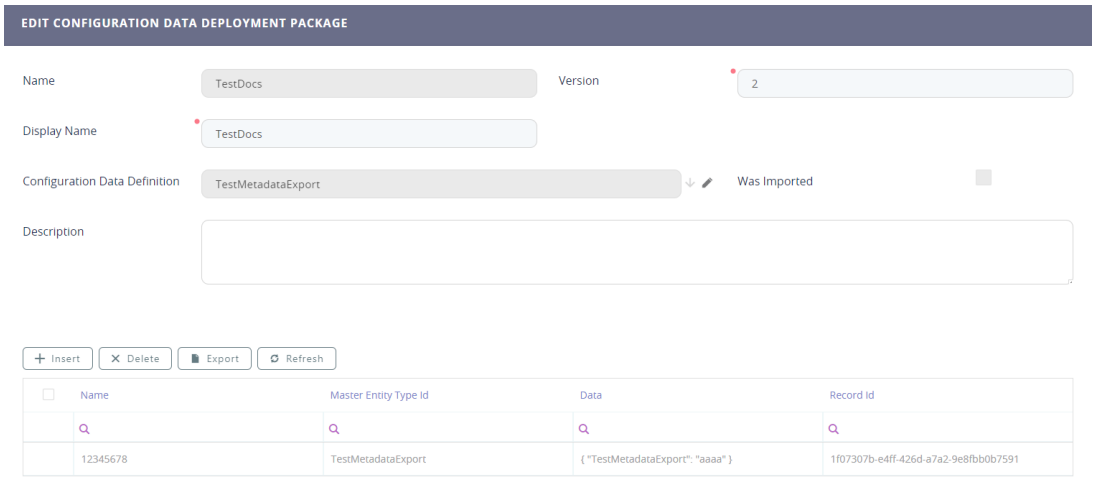
4. Click the **Save and reload** button.

5. Click the **Insert** button on the grid twice, and select the unique identifier or click **Insert**.



6. Fill in the following:

Field	Data type	Description
Name	Text	It is the name of the item.
Record ID	Option set	
Master Entity Type Id	Text	This is the entity exported.
Data		This is where the unique constraint will show.



7. Click on **Export Data Config Data** to export the package.

Import data using DevOps

There are two ways in which you can import an .xml file in Innovation Studio. Depending on what you are importing, there are several steps to follow and dedicated spaces to use:

- to import a deployment package, follow the path Innovation Studio > DevOps >Deployment package.
- to import a configuration data deployment package, follow the path Innovation Studio > DevOps > Configuration Data Deployment package.

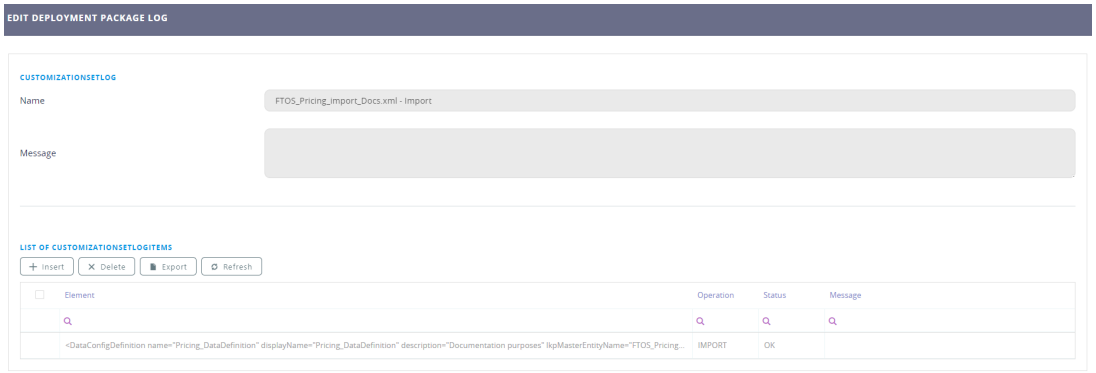
1 Deployment package

To import the package:

1. Follow the path Innovation Studio > DevOps >Deployment package.
2. Click on the **Import deployment package** icon found on the top right-side part of the screen.
3. The local Files menu of the device opens. Select the file to be imported.
4. The system will ask the user **''Source database has the version 21.1.1.0. Continue?''**. The answers options are **Yes** and **No**. Select **Yes** to continue.
5. The system will ask the user **''Do you want to deploy audit configurations (you should normally choose 'Yes')?''**. The answers options are **Yes** and **No**. Select **Yes** if you wish to deploy it. If you do not wish to deploy it, click **No**.
6. The package has been successfully imported if the package meets the requirements. These requirements are needed to be met when the package is created, i.e. exported. For details, see Creating Enhanced Deployment Packages.

To see the package:

1. Follow the path Innovation Studio > DevOps >Deployment package Logs.
2. Search the package using the name you have given it.
3. Double- click on the name. The **Edit Deployment Package Log** opens. The fields are read-only.



4. In the grid named List of customization log items, double-click on the item to see it.

   The page **Edit Deployment Package Logs Component** opens. The fields are read- only.

2 Configuration Data Deployment package

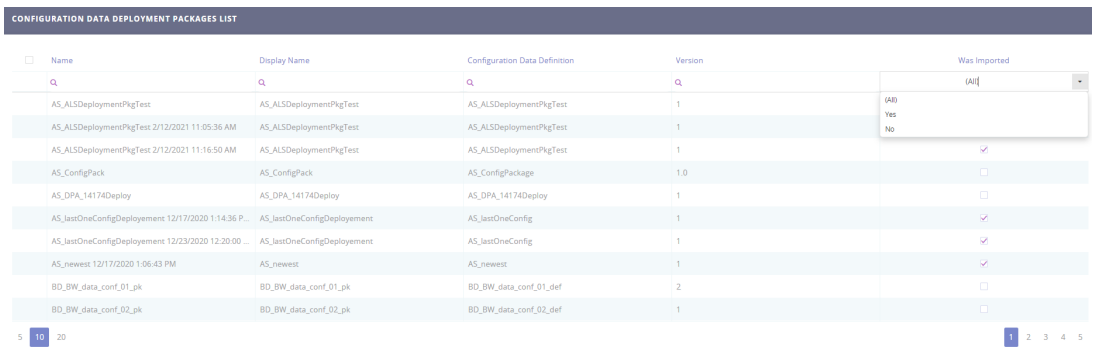
To import the package:

> ⚠ **IMPORTANT!**
> This package only contains the records, not the metadata, such records are inserted into the fields in a FintechOS Portal.

1. Follow the path Innovation Studio > DevOps > Configuration Data Deployment package.
2. Click on the **Import Data Config Data** icon found on the top right-side part of the screen.
3. The local Files menu of the device opens. Select the file to be imported.
4. The system will ask the user **''Source database has the version 21.1.1.0. Continue?''**. The answers options are **Yes** and **No**. Select **Yes** to continue.
5. The system will ask the user **''Do you want to deploy audit configurations (you should normally choose 'Yes')?''**. The answers options are **Yes** and **No**. Select **Yes** if you wish to deploy it. If you do not wish to deploy it, click **No**.
6. The package has been successfully imported if the package meets the requirements. These requirements are needed to be met when the package is created, i.e. exported.

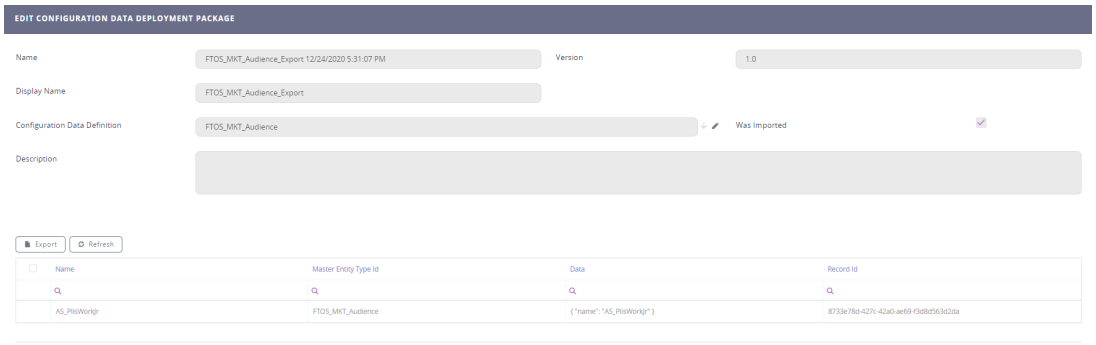
To see the package:

1. Search for it by its name or to see only the imported packages tick the boolean **was imported** yes.



2. To open the package, double-click on the name. The fields are read-only. If the boolean

Was imported is true, i.e. ticked, then the package was imported successfully.



Fintech Mobile Launcher - Capacitor

The Fintech Mobile Launcher allows you to connect your mobile device to a FintechOS Portal or digital journey.

Setup

- Install the requirements (nodejs, AndroidStudio, Xcode for iOS).
- Download the project from https://github.com/FintechOS/MobileLauncherCapacitor git clone https://github.com/FintechOS/MobileLauncherCapacitor --recursive
- npm install in the root of this project.
- npm install in the /capacitor folder (in order to fix a redirect issue, a fork was made on the capacitor repo - reason why this exists)
- npm run build in the /capacitor folder
- npx cap sync in the root of the project

If you experience issues with npx cap, please run "npm install -g @capacitor/cli"

npx command examples

npx is a new utility available in npm 5 or above that executes local binaries/scripts to avoid global installs

Adding a platform: - npx cap add android - npx cap add ios

Open builder based on platform: - npx cap open ios (requires xcode) - npx cap open android (requires android studio)

Configuration

The main config file is called capacitor.config.json and is one of the few items that needs to be changed before deploying a new mobile app.

```
{
  "appId": "com.fintechos.core.n.demo", //id which you use to
register your application
  "appName": "FintechOS Native Demo",
  "bundledWebRuntime": true,
  "npmClient": "npm",
  "webDir": "www",
  "appendUserAgent": "FtosNative",//mandatory in order to let the
platform know it has been accessed from a mobile device
  "server": {
    "url":
"https://training0003.westeurope.cloudapp.azure.com/FTOS_
Portal/Main", //link to your portal or user journey - this is the
page that will be loaded when launching the app
    "allowNavigation": [
      "training0003.westeurope.cloudapp.azure.com"
    ]
  },

  //below you can specify which plugins to be added to the app and
their settings
  "plugins": {
    "SplashScreen": {
      "launchShowDuration": 0
    },
    "PushNotifications": {
      "presentationOptions": ["badge", "sound", "alert"]
    }
  }
}
```

Google Services

In case that the bundle name or id changes, or you configure a different FCM project for notifications, the MobileLauncherCapacitor\android\app\google-services.json needs to be updated and the build process to be executed. Note that the changes won't propagate until the build process is called.

The google-services.json needs to contain the bundleid specified in the appId field of the capacitor.config.json (i.e. com.fintechos.core.n.demo)

iOS Build

If asked for CocoaPods install: - run pod install or pod update inside the ios folder

FTOS sample widget code to register for notifications

Window.Capacitor methods will be changed in the future with a more user friendly wrapper.

```
  //we determine the portal was launched from a mobile device by
adding UserAgent "FtosNative"
  //isFtosNative checks if there's a user agent with that value
  if (ebs.isFtosNative) {

      window.Capacitor.Plugins.PushNotifications.requestPermission
().then( result => {
        if (result.granted) {
          // Register with Apple / Google to receive push via
APNS/FCM
          window.Capacitor.Plugins.PushNotifications.register();
        } else {
          // Show some error
        }
      });

      // On success, we should be able to receive notifications
```

```
      window.Capacitor.Plugins.PushNotifications.addListener
('registration',
        (token) => {
            console.log(token);
          alert('Push registration success, token: ' +
token.value);
          //registering to Azure Notifications Hub, for more
details on this please check
          //http://docs.fintech-
os.local/docs/Content/articles/server/AutomationScripts/server_
side_sendMobileNotifications.html
          ebs.callEbsService({
              //possible values:fcm or apns - for this example it
is hardcoded to fcm
              Platform: "fcm",
              Handle: token.value,
              AppName: "testapp"
          }, "./NotificationHub/CreateOrUpdateRegistration",
console.log, console.error);
        }
      );
  }
```