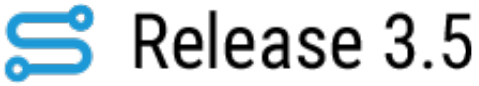
Simplifier Academy

Courses & Documentation

PDF generated April 30, 2019

Table Of Contents

New Features Release 3.5



The Documentation refers to the latest version of the Simplifier. If you have an On-Premise installation and need help with an older version, please contact us via support@itizzimo.com, we are glad to support you.

New Features

Client-side Business Objects

Business objects can now also be executed directly on the browser or on the respective mobile device. They are created centrally and can therefore be reused several times. Read more »

Tags of connectors and business objects for easier search / categorization

Each connector/business object, widget and data type can now be tagged. For example, project-related configuration objects can be given a project name, which makes them easier to find again.

Automatic reloading of the preview when saving in the UI Designer

Every click on the “Deploy” button automatically updates the preview in the browser.

New Business Object API incl. autocompletion

Within a business object, the most common functions can be accessed using a uniform API and auto-completion. Read more »

Logging & Monitoring for Connectors and Business Objects

Via the Logs & Monitoring you can now see who did the last changes in a connector or business object.

Mobile Log

Errors in the Simplifier Mobile Client can now be evaluated centrally.

Application configuration level

You can now check how much programming effort is avoid through the low-code approach, the degree of configuration is now displayed in percent in the application overview. Read more »

Interactive API documentation

The documentation of the Simplifier REST API is now directly accessible via an HTML page and can be tested directly.

New Mobile Action

For the detection of counter readings we now support the technology of our partner Pixolus via configuration.

The background mode can be activated for notifications / push messages so that the user is informed even without active use of the Simplifier Mobile Client.

Updated Features

Conflict management during import of transports

When importing Simplifier transports, the user will be made aware of potential conflicts in the future. This creates transparency for shared connectors, data types or business objects. Read more »

Usage List for Connectors and Business Objects

For efficient management of dependencies between applications and data from backend systems, we now offer an overview that shows which data is used by which application or which interfaces are integrated in which applications.

Extended configuration options in the Process Designer

We have expanded the configuration options in the Process Designer:

- Busy indicator for access to connectors or business objects can now be switched on and off
- Validation of mandatory parameters for mappings for business objects and connectors
- New toolbar incl. copy & paste function
- Validation of the configuration when saving the user story
- External links in the user stories allow linking to an external project management tool
- Double-click on configuration artifacts opens the configuration directly
- The navigation activity can be linked for follow-up activities and transition effects can be set
- Search for widgets
- Code in a script activity can be embellished by pretty print

Read more on Process Designer and the following pages.

Extended authorization structure and security measures

The authorizations have been refined so that the application or interface can only be read, created or changed for each user. Read more »

Security measures

Brute force attacks can now be prevented via the security policies.

Time-controlled jobs

Simplifier Jobs can now be started automatically in several variants (daily, weekly, monthly). Read more »

Mark Widgets as deprecated

Widgets that are no longer required can be marked as deprecated and are no longer available for use in the UI Designer. Read more »

Simplifier Universal Client for Android

We have reworked our Simplifier Universal Client for Android to include the same features as in the iOS Universal Client. In addition to the function that you can work offline with the Simplifier client or the autostart function for already installed business applications, we now also support the fingerprint sensor for login with Android. Furthermore, the Android client now appears in a new layout in the native Android material design.

Moreover, we offer an optimized menu navigation on data glasses, depending on the hardware complete with voice control. Support information is also automatically transmitted to the server on request.
And for the developers, we now offer a developer mode that displays errors from the console directly on the device.

Getting Started

Using state-of-the-art technologies, we accelerate your application creation. We have designed and built our platform in terms of logic and usability to accommodate the modern, agile development processes within companies. Due to the low-code approach, applications no longer need to be elaborately programmed but can be easily configured and integrated into any system. Thus, applications can be mapped process-oriented.

Basically, creating an application with our Simplifier can be divided into the following 5 steps:

1. Connect systems

In the future, for each application, access the data that is needed contextually to make the process more efficient. Standardized connectors enable you to quickly connect to any back-end system and various data sources.

2. Create user interface

Easily, quickly and intuitively create the user-friendly interface for all your applications. Use the pre-designed elements designed for this purpose and create a uniform look and feel for an improved user experience.

3. Configure processes

Configure the process logic and processes of each application using the Process Designer. Based on individual user stories, reusable processes are defined within the Process Dashboard. Each user story can be stored individually, so that you can work with several people on different stories at the same time and thus be able to map a process logic quickly and collaboratively. When the application is deployed, all stories were merged into one.

4. Test application

Test your application at any time in the Simplifier Mobile client or in the browser. Intermediate testing allows for faster detection and elimination at any point in time. The Simplifier Mobile Client supports the testing process by ensuring that applications can be used across devices and that the correct display of the application on each end device can be ensured.

5. Publish application

The Simplifier transport system allows you to transfer the application to your productive system quickly and easily. And all without compilation or complex deployment processes. Quickly create a transport file of the finished application – all the components contained in it are fully integrated into the file – download them and import them directly into your production instance. Finished! The application is now available to any authorized user.

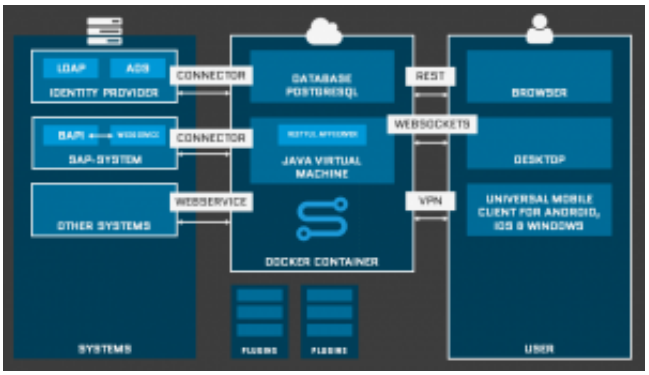
What is the Simplifier?

The Simplifier is a low code platform for mapping business processes in integrated business and IoT applications and to interconnect internal and external IT infrastructures. Applications only need to be configured once to be available on any mobile device and operating system.

Main Features:

- Web-based Development Environment to configure integrated Mobile, Wearable and Browser applications
- Customization of Applications with UI Designer, Visual Scripting for SAPUI5 and AngularJS based Applications
- Customization of Backend Interfaces through standardized Connectors (SOAP, REST, XML, IDoc, MQTT, OPC UA, API, CSV)
- Rapid Deployment and OTA-Updates
- Contextual Technologies (Augmented Reality / Realtime Communication, Scanning, Sensors)
- Multi-Device (Browser, Phones, Tablets, Glasses, Watches)
- Multi Platform Universal Client (Android, iOS, Windows)

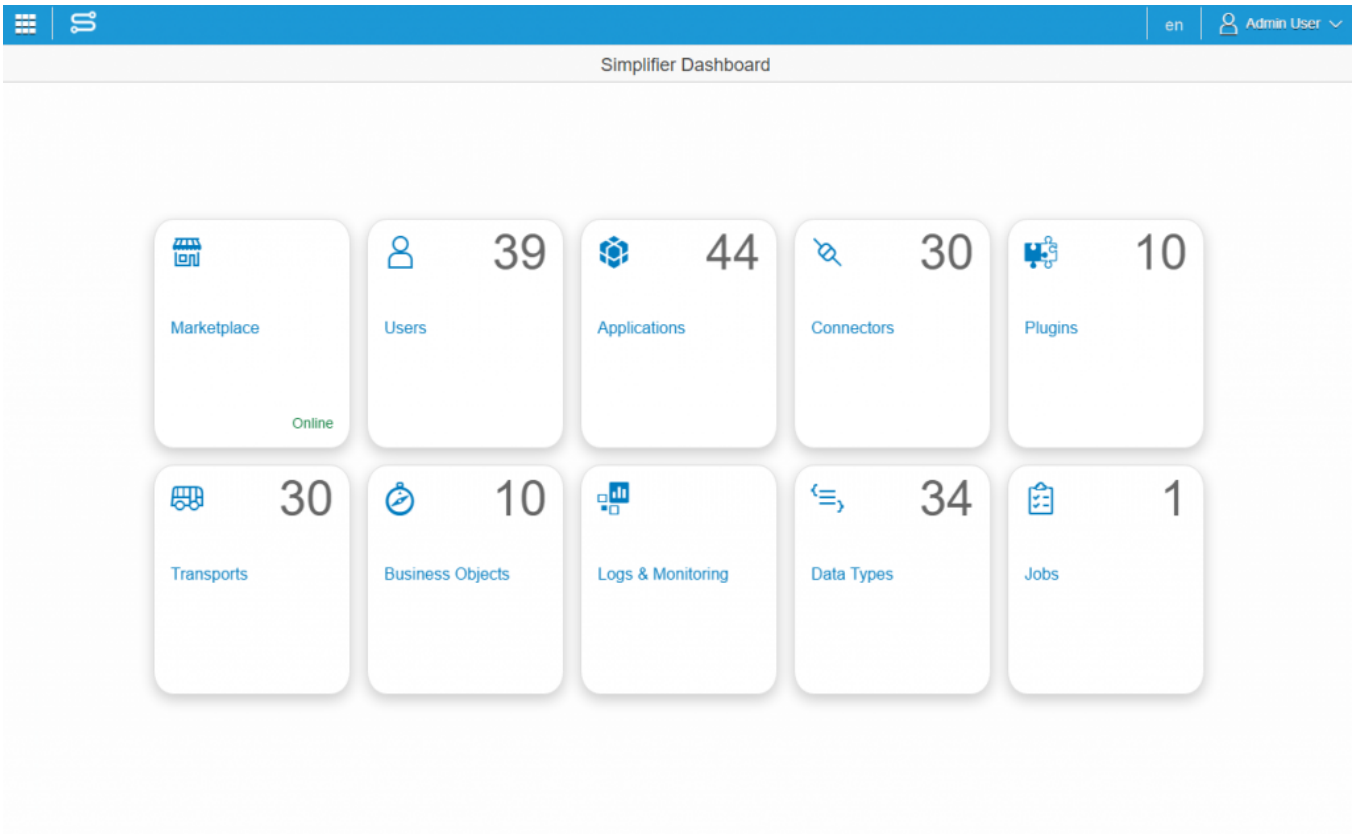
Simplifier and Mobile Devices



Simplifier Architecture – Cloud or On-Premise

State of the art Web Technologies

The Dashboard



The Dashboard is the place for you to start. It gives you a clear overview about your Simplifier instance. Here you find information about:

- Marketplace
- Users and their roles
- Applications
- Connectors
- Plugins
- Transports
- Business Objects
- Logs and Monitoring
- Data Types
- Jobs

Start Simplifier for free

The Simplifier Freemium (free + premium) account contains a test environment where you can see and test selected features. The Freemium account comes with predefined application templates and example applications for different use cases.

Apply for an account by filling out the registration form on simplifier.io

You can test the Simplifier extensively and completely free of charge for 90 days. After 90 days the Simplifier is of course still for free use, only the server hosting has to be paid for.

You have the possibilities to create as many applications as you like and create and configure connectors to integrate all your systems.

Besides that, you have full administration access including user administration and monitoring and you have unlimited permissions to Simplifier items like widgets, libraries, data types, plugins, jobs and business objects.

Prerequisites

[vc_row row_type=”row” stretch_row_type=”no”][vc_column][vc_column_text]

1. The Admin Interface supports only Google Chrome Browser latest Version -1 & Firefox latest Version -1 for Simplifier Configuration & App Development, please download from https://www.google.de/chrome/browser/desktop/
2. Use Google Developer Tools for Mobile Preview, Console, Debugging and DOM Inspection
3. Install and use UI5 Inspector to inspect and debug your UI5 Application
4. For testing on mobile devices use Android (>4.4) , iOS (>9) and Windows 10 devices with Simplifier Mobile Client available for iOS and Android
5. Aps created with Simplifier support the second to last Version (n – 1) of the following Browser:
   – Chrome
   – Firefox
   – Safari
   And the Browser:
   – Internet Explorer (Version 10-11)
   – Microsoft Edge

[/vc_column_text][/vc_column][/vc_row]

Features and supported operating systems

- Remote Debugging for Mobile Devices
- Debug View within Simplifier Mobile Client (Debug Version)
- Auto-Login Functionality
- App Autostart
- Call Settings

See the full featured list of supported operating systems:

	Browser	Android > 4.4+	iOS > 10.x+	Windows 10	Hololens
Capture/Play Audio/Video This plugin provides access to the device's audio, image, and video capture capabilities.	**?**	•	•	•	**?**
Get Device Information This plugin defines a global device object, which describes the device's hardware and software. Although the object is in the global scope, it is not available until after the deviceready event.	**?**	•	•	•	**?**
Access Accelerometer This plugin provides access to the device's accelerometer. The accelerometer is a motion sensor that detects the change (delta) in movement relative to the current device orientation, in three dimensions along the x, y, and z axis.	**?**	•	•	•	**?**
Access Device Orientation Provides information about the physical orientation and motion of the device.	•	• 5.0+	•	•	**?**
Access Compass This plugin provides access to the device's compass. The compass is a sensor that detects the direction or heading that the device is pointed, typically from the top of the device. It measures the heading in degrees from 0 to 359.99, where 0 is north.	•	•	•	•	•
Access file system This plugin implements a File API allowing read/write access to files residing on the device.	•	•	•	•	•
Up-/Download Content This plugin allows you to upload and download files.	•	•	•	•	**?**
Access GPS This plugin provides information about the device's location, such as latitude and longitude.	•	•	•	•	**?**
Get Network Information	•	•	•	•	**?**

This plugin provides an implementation of an old version of the Network Information API. It provides information about the device's cellular and wifi connection, and whether the device has an internet connection.					
Native Dialogues This plugin provides access to some native dialog UI elements	•	•	•	•	•
Trigger / Handle Notifications The essential purpose of local notifications is to enable an application to inform its users that it has something for them — for example, a message or an upcoming appointment — when the application isn't running in the foreground.	**?**	•	•	•	**?**
Access/Configure Statusbar This plugin enables the user to make changes to the status bar of a mobile device	**?**	•	•	**?**	**?**
Access Device Vibrator This plugin provides a way to vibrate the device.	Chrome Firefox	•	•	•	**?**
			Ignores the specified time and vibrates for a pre-set amount of time.	Max time is 5000ms (5s) and min time is 1ms	
Scan Barcodes Scan many different kinds of barcodes.	•	•	•	•	**?**
Beacon Scanning This plugin provides the functionality to use beacons with the iBeacon protocol.	**?**	•	•	**?**	**?**
Augmented Reality This plugin provides augmented reality functionality by Wikitude.	**?**	•	•	**?**	**?**
Bluetooth This plugin enables communication between a phone and Bluetooth Low Energy (BLE) peripherals.	**?**	•	•	•	**?**
PDF Viewer View PDF files.	•	•	•	•	**?**
Toggle Flash Toggle the flash light of the device.	•	•	•	**?**	**?**
Text to Speech Text to Speech functionality.	•	•	•	•	**?**
Speech recognition Speech to text.	**?**	•	•	**?**	**?**
Share with This plugin allows you to use the native sharing window of your mobile device.	**?**	•	•	•	**?**
SQLite	**?**	•	•	•	**?**

Native interface to sqlite.					
Fullscreen Set your app fullscreen.	**?**	•	•	**?**	**?**
Battery Status Get battery status and information.	**?**	•	•	•	**?**
Offline Work Login and run your applications offline.	**?**	•	•	**?**	**?**
Remote Call Audio/Video communication.	•	•	•	**?**	•

separate appli
required.

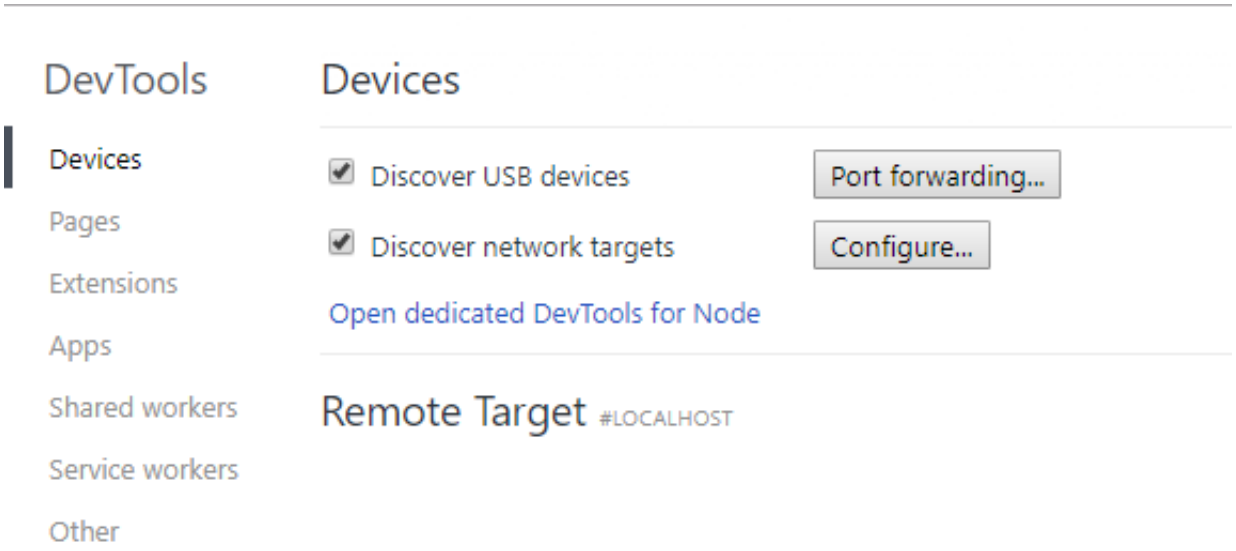
Live Debugging with Chrome DevTools

You can debug your applications on the Simplifier in real time on mobile devices.

Enter **chrome://inspect** into your Chrome browser.

Connect your mobile device via USB and enable “USB Debugging” within the device settings. Chrome will autodiscovering your mobile device and integrating it within the developer tools options.

When connected, you can begin screencasting by clicking the appropriate button in the developer tools window. Then your mobile device can be supplied directly with input from your desktop screen. Any changes are instantly displayed on your mobile device.

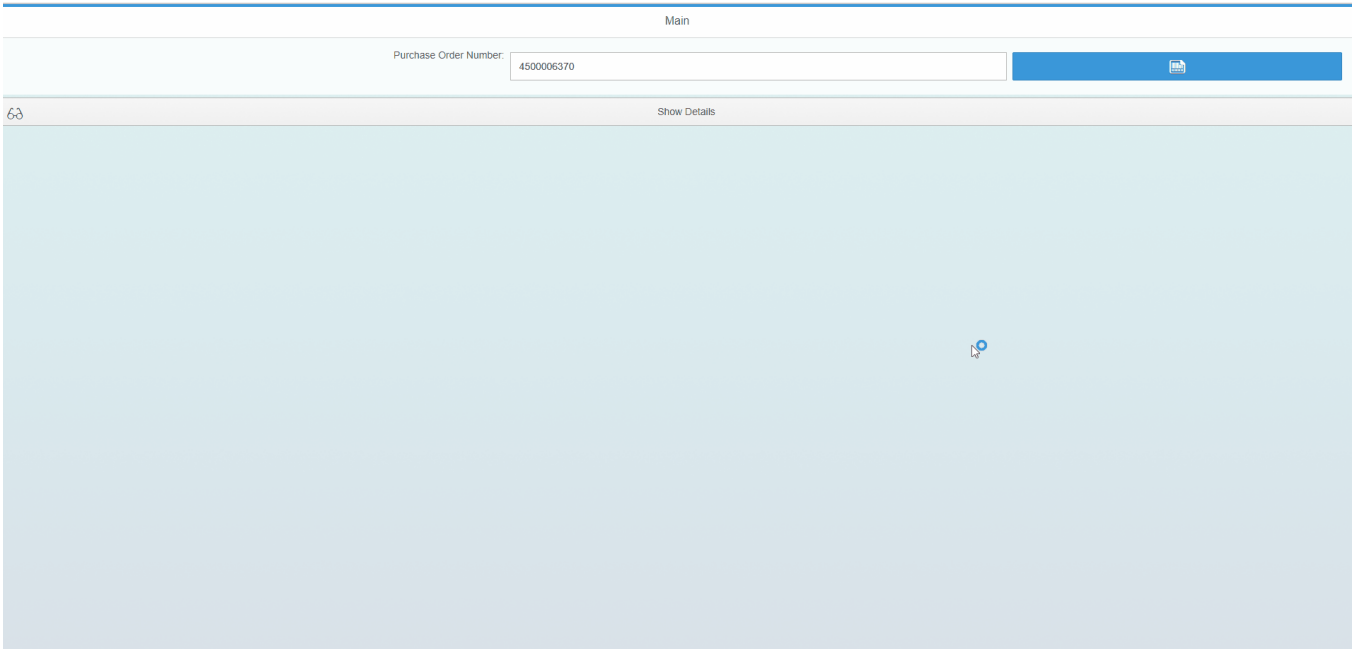


By the way, the “Port forwarding” button above provides some flexibility for those who don’t have their screen and developer machine on the same network. In this scenario, you can create a TCP port on your mobile device and assign it to a specific TCP port on your development machine. All forwarded traffic is transferred via USB, bypassing the mobile device’s network configuration.

Mobile Imitation

By clicking the “Toggle device toolbar” button, the current page is reduced to the imitated screen parameters according to the set properties. This imitated display can be operated with the mouse in the same way as with the finger on a physical device.

In this view, you can simulate pinch zooming, finger scrolling and even multi-touch events.


Main
Purchase Order Number:
4500006370
Show Details

Supported Barcode formats

Barcode Type	Android	iOS	Windows
QR_CODE	?	?	?
DATA_MATRIX	?	?	?
UPC_A	?	?	?
UPC_E	?	?	?
EAN_8	?	?	?
EAN_13	?	?	?
CODE_39	?	?	?
CODE_93	?	?	?
CODE_128	?	?	?
CODABAR	?	?	?
ITF	?	?	?
RSS14	?	?	?
PDF417	?	?	?
RSS_EXPANDED	?	?	?
MSI	?	?	?
AZTEC	?	?	?

Simplifier Mobile Client

Download the Simplifier Mobile Client in the appropriate stores, depending on your mobile device.

Descriptions of how the Simplifier Mobile Client works can be found in the subpages.

---

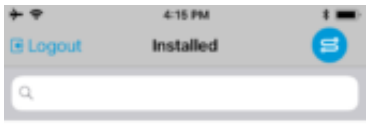
iOS Client

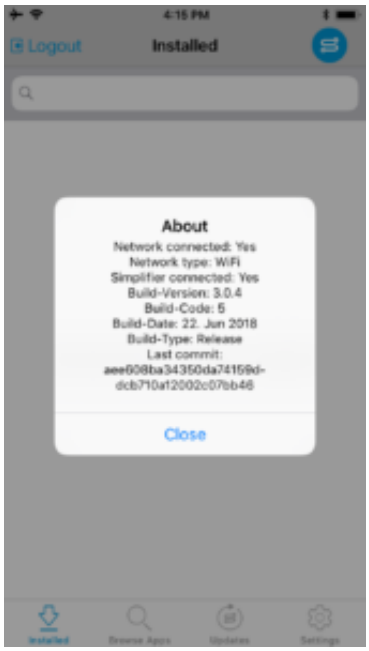
Below is a description of the Simplifier Mobile Client for iOS. After you have downloaded the Simplifier Mobile Client from the App Store, start it on your mobile device.

First you have to authenticate yourself on the login screen with your Simplifier username and password. Enter the instance you want to access. If the device has a fingerprint reader, you can choose to restore your password with it. You can save your login so you don’t have to re-enter it every time.

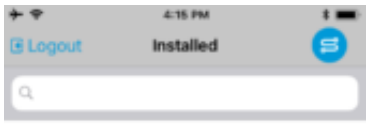
Since it is uncomfortably to type the instance, you can also use the QR code login. Read here how to create a corresponding QR code in the Simplifier.

Once you have successfully authenticated yourself, you are in the overview of installed applications. At the beginning, this overview is empty. At any time, you can log out by clicking on the logout button in the top left corner. At the top right, on the Simplifier icon, various information will be displayed.

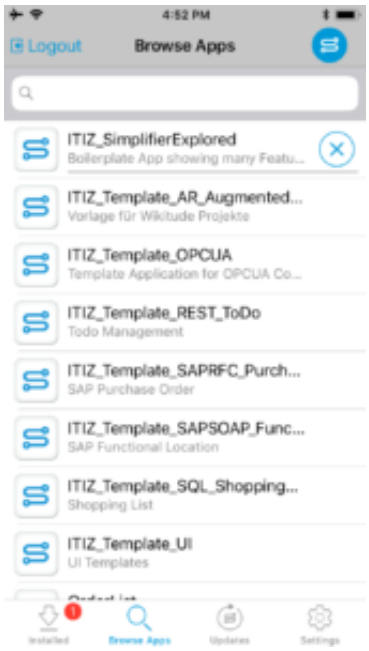


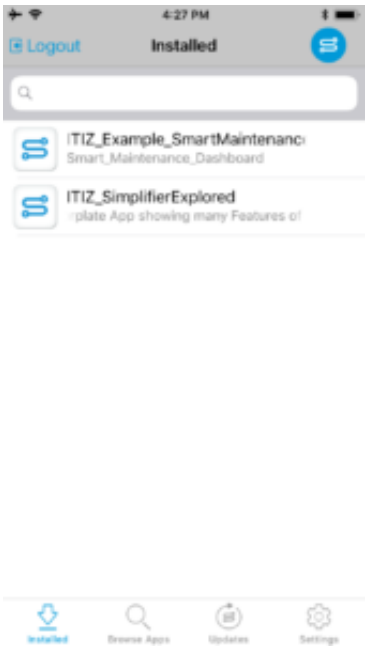


To now use apps on your mobile device, switch to the screen **Browse Apps**. You now see an overview of all applications that are on the specified instance. To install apps, simply click on them. When the apps are downloaded, **Installed** will display a notification with the number of newly installed apps. You can delete installed application by swiping to the left.



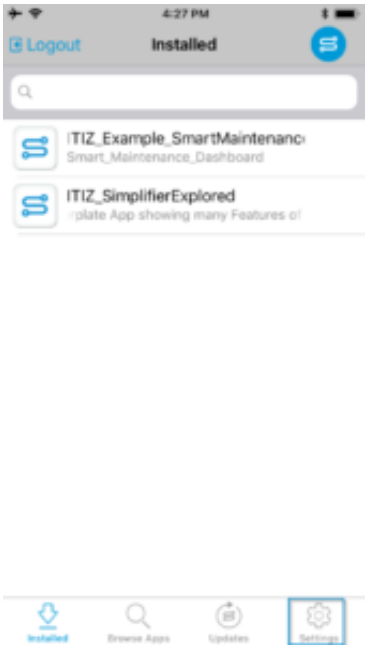
No installed applications available

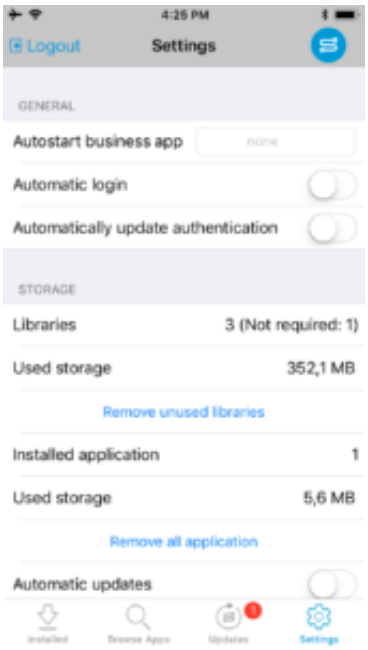


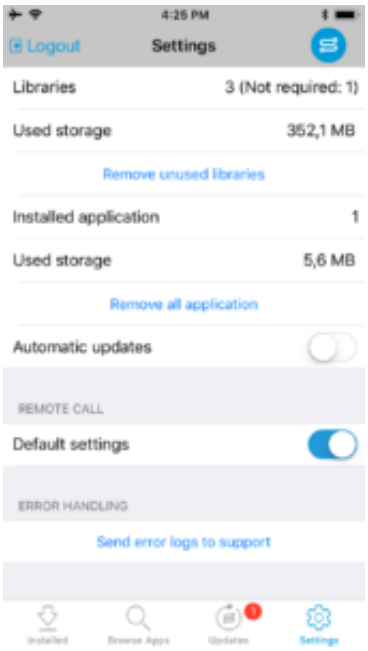


If an app, that you have already installed, has been newly deployed on the instance, you will be informed about updates of the application.

At the bottom right, you will find an overview of all settings.




4:25 PM
Logout
Settings
GENERAL
Autostart business app
none
Automatic login
Automatically update authentication
STORAGE
Libraries
3 (Not required: 1)
Used storage
352,1 MB
Remove unused libraries
Installed application
1
Used storage
5,6 MB
Remove all application
Automatic updates
Installed
Browse Apps
Updates
Settings


4:25 PM
Logout
Settings
Libraries
3 (Not required: 1)
Used storage
352,1 MB
Remove unused libraries
Installed application
1
Used storage
5,6 MB
Remove all application
Automatic updates
REMOTE CALL
Default settings
ERROR HANDLING
Send error logs to support
Installed
Browse Apps
Updates
Settings

Client installation on Windows

If the Cordova Simplifier client and -server are running in a closed network, and the SSL certificates are self-signed, it is necessary to install a special version of the Simplifier mobile client on Windows. This version contains certain permissions that are not compatible with the Windows Store.

You need 2 files for the installation:

- <name>.appxbundle – Contains the client for x86, x64 and ARM processors.
- <name>.cer – Contains the developer certificate of the app.

There are 2 ways to install the client. In either case, the “developer mode” must be enabled in Windows to allow applications to be “cross-loaded”. This process is equal to Android and the menu item “Unknown origin”.

If a “certificate error” is displayed in one of the steps, please refer to the section “Installing the certificate” below.

ATTENTION

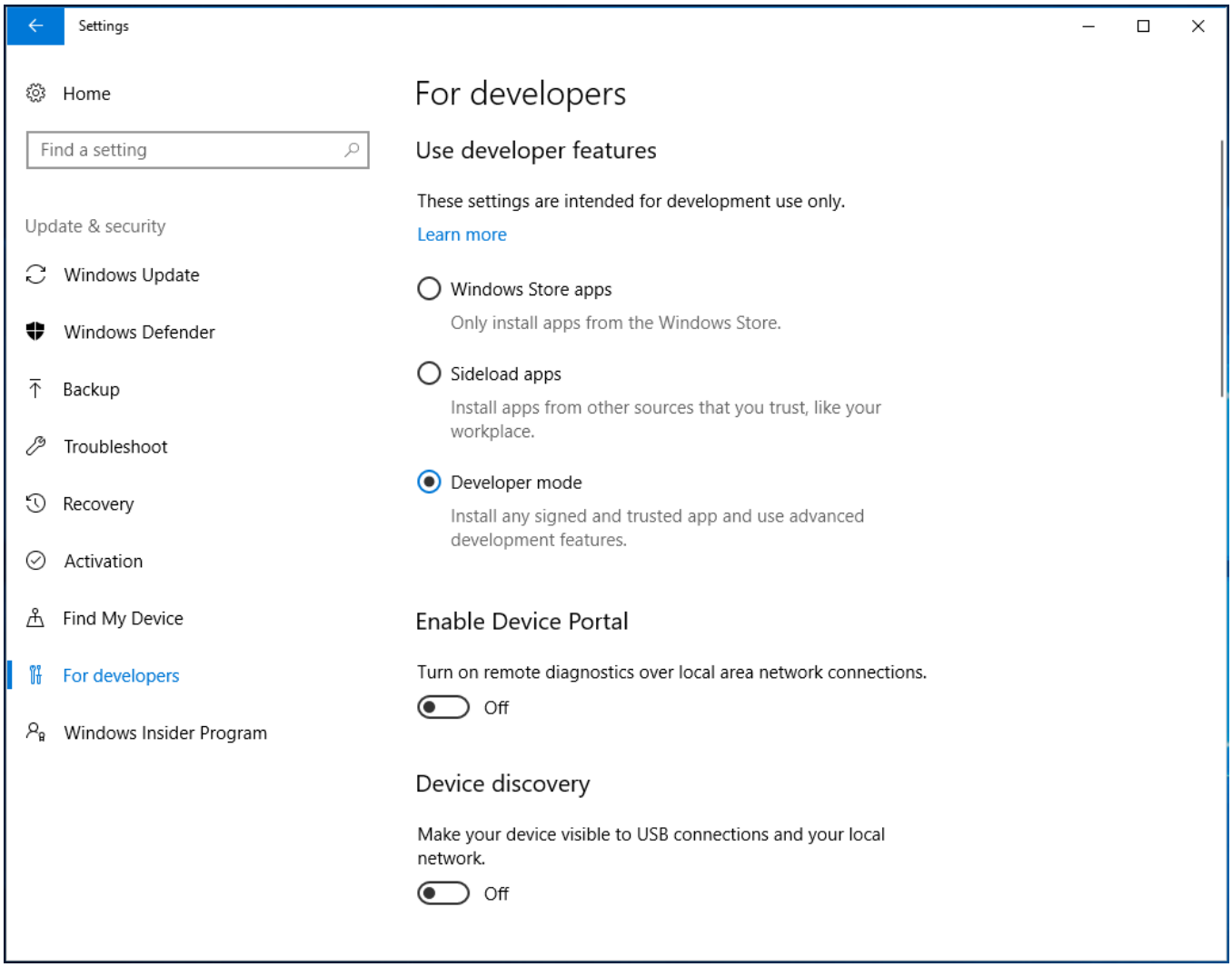
It is mandatory to be in the local domain. This requires at least Windows 10 Pro or Enterprise!

Prerequisites

Developer-Mode

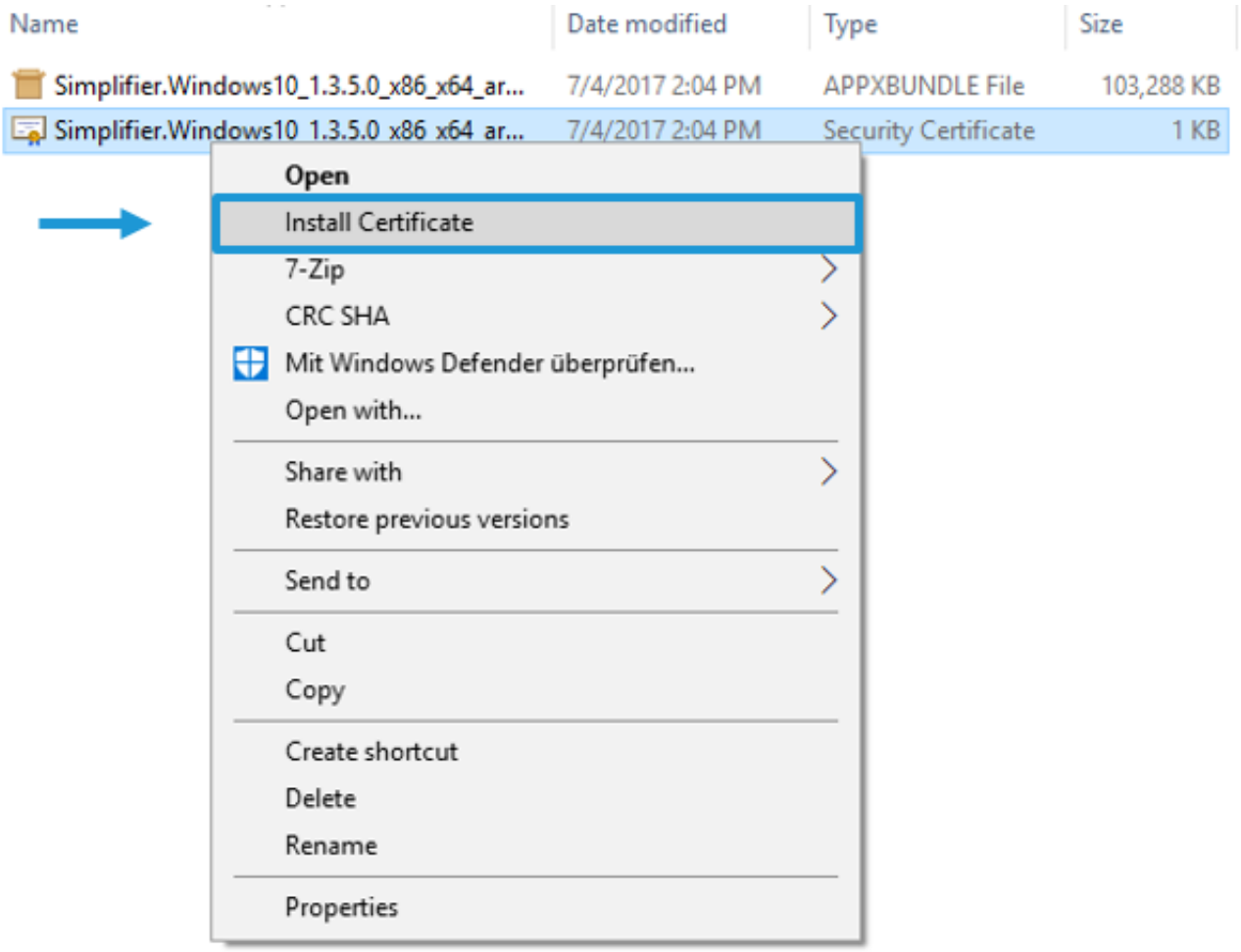
Please follow these steps:

- Settings ? Update and Security ? For Developers
- Enable the developer mode and install the equivalent packages
- finished



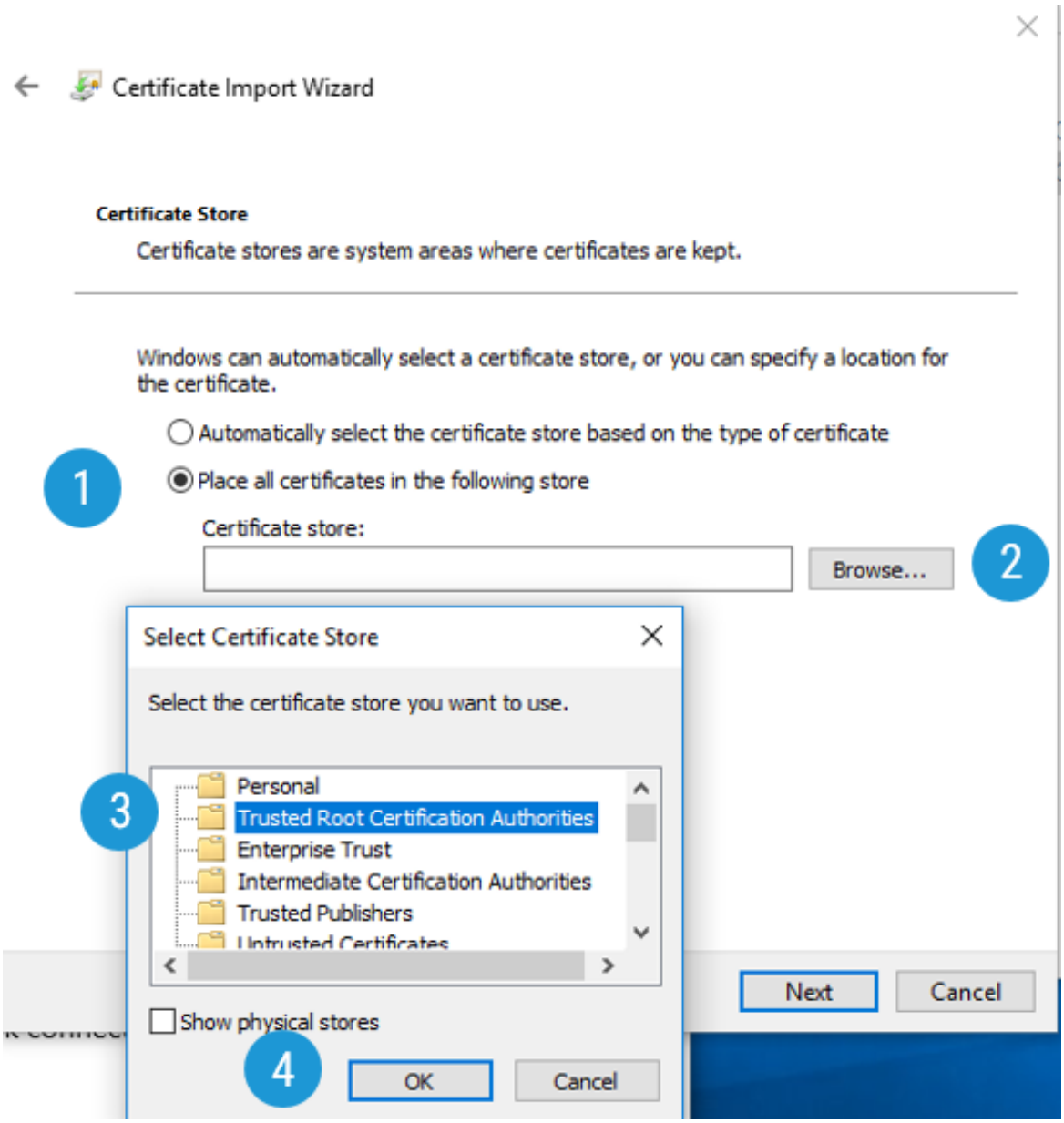
Install the certificate

At first, the developer certificate must be installed:



A wizard opens where you should follow the steps below:

- In the welcome screen, select the second item “Local computer”
- Continue
- The Import Wizard opens
- We need again the second point “Save all certificates in the following memory”
- Click “Browse”
- Select “Trusted Root Certification Authorities”



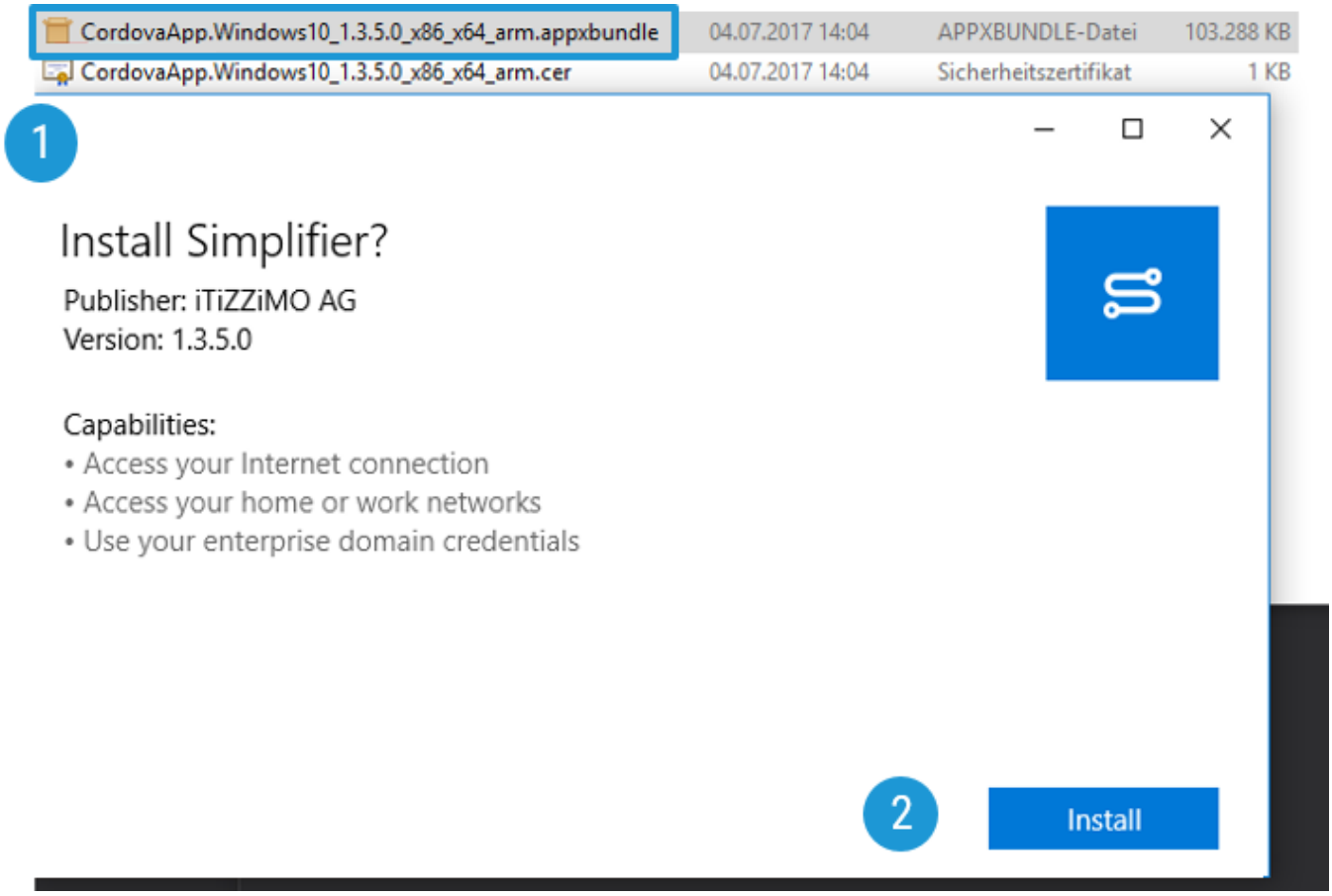
- Click “OK”
- Continue
- A summary of the previous steps is now displayed
- A click on “Finish” completes the operation and the certificate is installed

Option 1: App-Installer

The following app must be installed on the Windows 10

computer: https://www.microsoft.com/en-us/store/p/app-installer/9nblggh4nns1 Store link.

This will associate .appxbundle files and you can start the installation process with a double-click on the file.



This screen will also check the required app permissions. The keyword is “Access to home or workplace networks” to head to the local Simplifier instance and “Use company domain credentials”, which is required for the certificate to the instance.

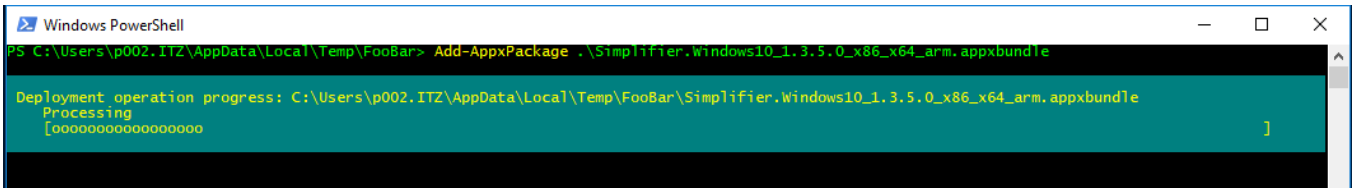
Option 2: Powershell

Switch with the Powershell to the folder with the Simplifier-Files (.appxbundle, .cer). Please note that this is a local location. Network addresses will fail.

Enter the command “Add-AppxPackage <filename> .appxbundle”. The file name may vary.

Example:

The installation process is started.



In the end, the Simplifier app can be started from the Start menu or the Tile menu.

If you are interested in the official Microsoft explanation click on the links below for more information.

- Information about side loading: https://docs.microsoft.com/de-de/windows/application-management/sideload-apps-in-windows-10
- Information about the capabilities: https://docs.microsoft.com/en-gb/windows/uwp/packaging/app-capability-declarations

In our Mobile Client, we use the following capability scenarios:

- Access your Internet connection
- Access your home or work networks (Required for On Premise only)
- Use your enterprise domain credentials (Required for On Premise only)

Client Schema

You have the possibility to launch your business application via URL.

Schema:

```
simplifierclient:///<action>/<value>?<param>=<value>[&<paramN>=<valueN>]
```

For explanation:

simplifierclient:// – The url type, on that the simplifier client is registered. All uris with this link opens the client by default. If parameters or path components are missing, at least the client is always started.

/<action> – The action to take. For now only “**appDirect**” is available.

/<value> – The value for the action.

?<param>=<value> – The URL arguments are beeing passed to webview so business app can access them. So on the client the local href would be something like
file:///some_very_long_ios_path/www/businessapps/Simplifier_Explored?foo=1&bar=2

Example:

```
simplifierclient:///appDirect/Simplifier_Explored?foo=1&bar=2
```

The example above launches the simplifier client if installed and runs app “Simplifier_Explored”.

Restrictions:

- Simplifier Client needs to be installed
- if client is not running, client will be startet and user has to login
- shows popup with countdown when a link was clicked
- url-launch is higher prioritized than automatic app-launch
- if client is already running with a business app, nothing will happen to prevent misbehavoir in app lifecyle
- shows warning if desired app is not installed
- if autoupdate before launch is enabled, the business app will be updated before launch
- broken or non valid links are not beeing processed

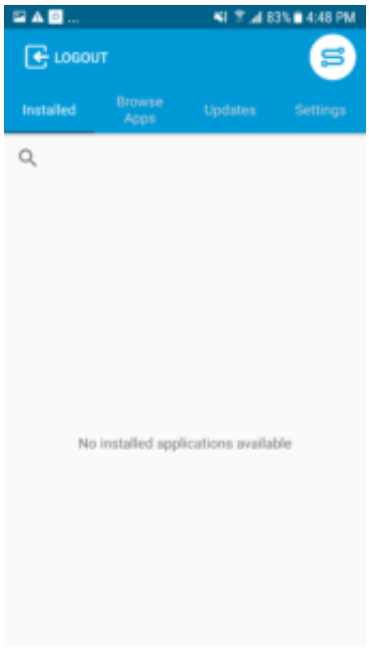
Android Client

Below is a description of the Simplifier Mobile Client for Android. After you have downloaded the Simplifier Mobile Client from the Play Store, start it on your mobile device.

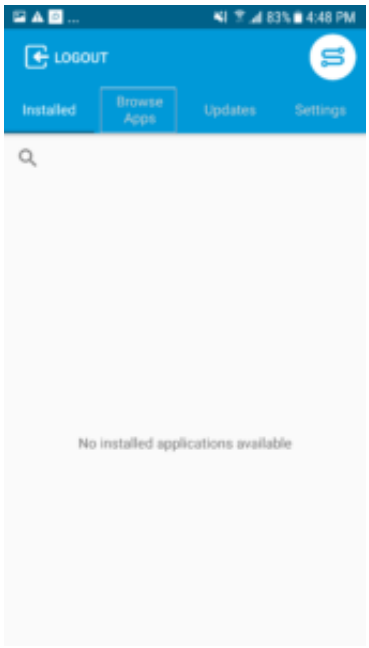
First you have to authenticate yourself on the login screen with your Simplifier username and password. Enter the instance you want to access. If the device has a fingerprint reader, you can choose to restore your password with it. You can save your login so you don’t have to re-enter it every time.

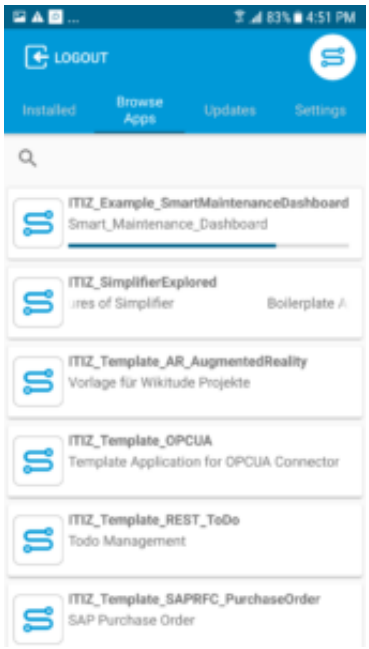
Since it is uncomfortably to type the instance, you can also use the QR code login. Read **here** how to create a corresponding QR code in the Simplifier.

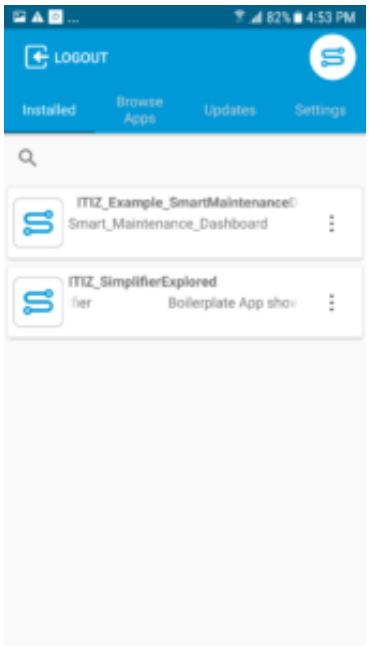
Once you have successfully authenticated yourself, you are in the overview of installed applications. At the beginning, this overview is empty. At any time, you can log out by clicking on the logout button in the top left corner. At the top right, on the Simplifier icon, various information will be displayed.



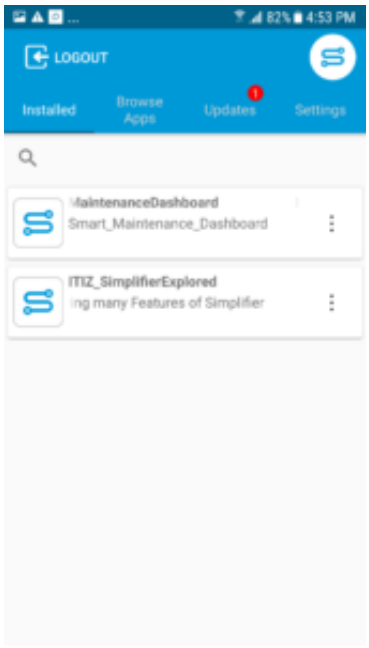
To now use apps on your mobile device, switch to the screen **Browse Apps**. You now see an overview of all applications that are on the specified instance. To install apps, simply click on them. When the apps are downloaded, **Installed** will display a notification with the number of newly installed apps. You can delete installed application by swiping to the left.


83% 4:48 PM
LOGOUT
Installed
Browse Apps
Updates
Settings
No installed applications available

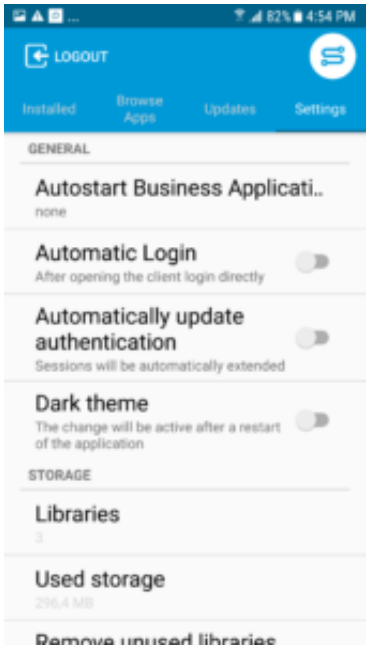

83% 4:51 PM
LOGOUT
Installed
Browse Apps
Updates
Settings
ITIZ_Example_SmartMaintenanceDashboard
Smart_Maintenance_Dashboard
ITIZ_SimplifierExplored
ITIZ_Template_AR_AugmentedReality
Vorlage für Wikitude Projekte
ITIZ_Template_OPCUA
Template Application for OPCUA Connector
ITIZ_Template_REST_ToDo
Todo Management
ITIZ_Template_SAPRFC_PurchaseOrder
SAP Purchase Order

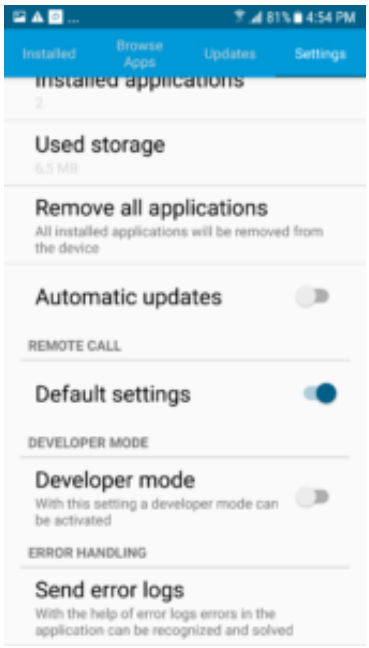


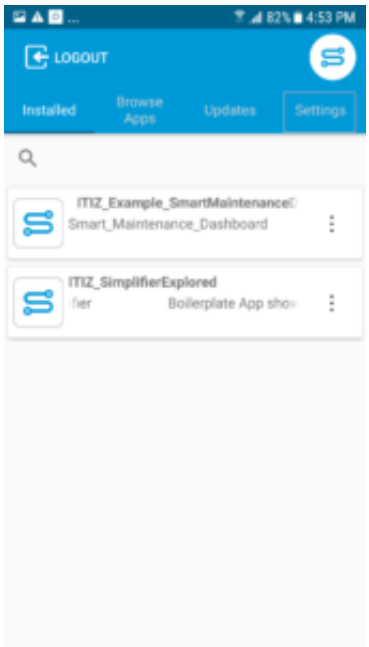
If an app, that you have already installed, has been newly deployed on the instance, you will be informed about updates of the application.



At the top right, you will find an overview of all settings.




81% 4:54 PM
Installed
Browse Apps
Updates
Settings
Used storage
Remove all applications
All installed applications will be removed from the device
Automatic updates
REMOTE CALL
Default settings
DEVELOPER MODE
Developer mode
With this setting a developer mode can be activated
ERROR HANDLING
Send error logs
With the help of error logs errors in the application can be recognized and solved


82% 4:53 PM
LOGOUT
Installed
Browse Apps
Updates
Settings
ITIZ_Example_SmartMaintenance
Smart_Maintenance_Dashboard
ITIZ_SimplifierExplored
Boilerplate App

Access Business Objects within Applications

To address Business Objects in your application, you do not have to call them in JavaScript code any longer. Simply use the activity “Data Objects” in the Process Designer. For more information, take a look at the documentation of the activity Business Object.

Access Connectors

You can access all connectors you've previously assigned to your business object inside your script template.

```
Simplifier.Connector.<ConnectorName>(payload?: string|object): object
```

Example:

```
var connectorResult = Simplifier.Connector.MySoap({bindingName: "Binding", "operation
Name": "MyOp", "soap": {"foo": "bar"}});
```

Each call to a connector in a script template follows this convention:

```
Simplifier.Connector.<ConnectorName>.<CallName>(payload?: string|object): object
```

Example:

```
var connectorCallResult = Simplifier.Connector.MySoap.myCall({"Foo": "bar"});
```

Access other Business Objects

You can call any other Business Objects in your current one in a similar way than calling a connector, using the following syntax:

```
Simplifier.BusinessObject.<BOName>.<MethodName>(payload?: string|object, parametrized
?: boolean = true): object

Simplifier.CurrentBusinessObject.<MethodName>(payload?: string|object, parametrized?:
 boolean = true): object
```

The payload has to be a stringified JSON object. The result of the business object method is a string and therefore it has to be parsed.

Example:

```
var otherMethodResult = Simplifier.BusinessObject.OtherBO.someMethod({"foo": "bar"});

var unparamtetrizedResult = Simplifier.BusinessObject.OtherBO.someMethod("{\"foo\": \
"baz\"}", false);

var sameBoResult = Simplifier.CurrentBusinessObject.someMethodOnSameBO({"baz": "baz"}
);

var noArgsResult = Simplifier.CurrentBusinessObject.methodWithoutArgs();
```

Access Plugins

Call a plugin from your business object using this syntax:

```
Simplifier.Plugin.<PluginName>.<SlotName>(payload?: string|object): object
```

Example:

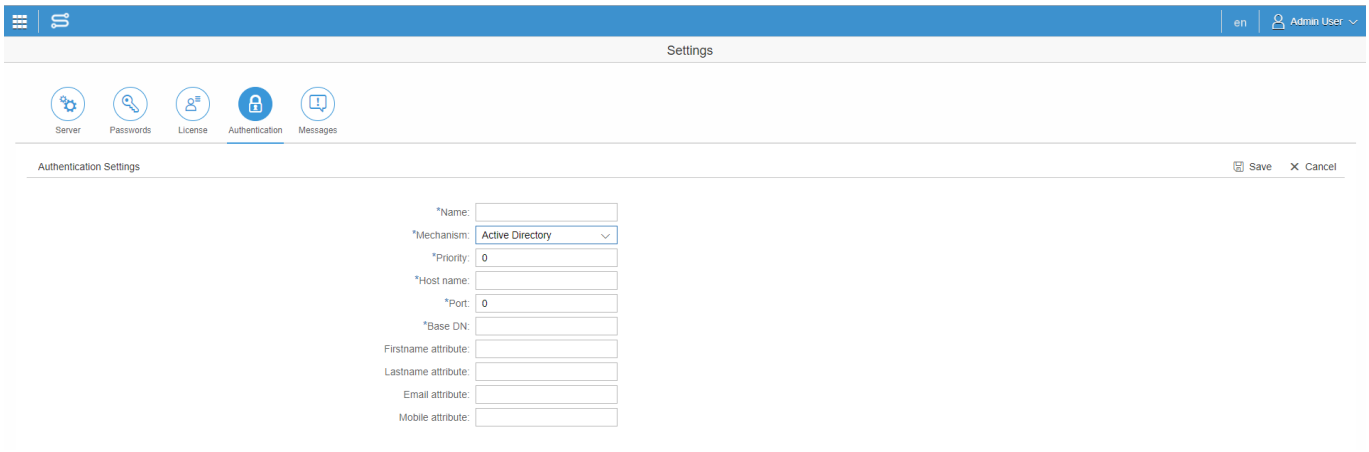
```
var repos = Simplifier.Plugin.contentRepoPlugin.listRepos();

var newRepo = Simplifier.Plugin.contentRepoPlugin.createRepo({name: "MyRepo"});

Simplifier.Plugin.contentRepoPlugin.updateRepo("{\"id\": 15}");
```

Action

Currently the Action shape is divided into Navigation, UI Action, Mobile Action and Server Action.

Active Directory

The Simplifier can sync users of Active Directories, like users from other LDAP sources.



Property	Description
Name	The name of the authentication mechanism
Mechanism	Active Directory
Priority	Priority in which the authentication system will be executed (descending)
Hose name	The address of the domain controller
Port	The port of the domain controller (default 389)
Base DN	Base distinguished name, the entry point (i.E. OU=area51,DC=simplifier,DC=io)
Firstname attribute	The attribute in the user object for the first name
Lastname attribute	The attribute in the user object for the last name
Email attribute	The attribute in the user object for the email
Mobile attribute	The attribute in the user object for the mobile number

Add a connector call to an existing connector

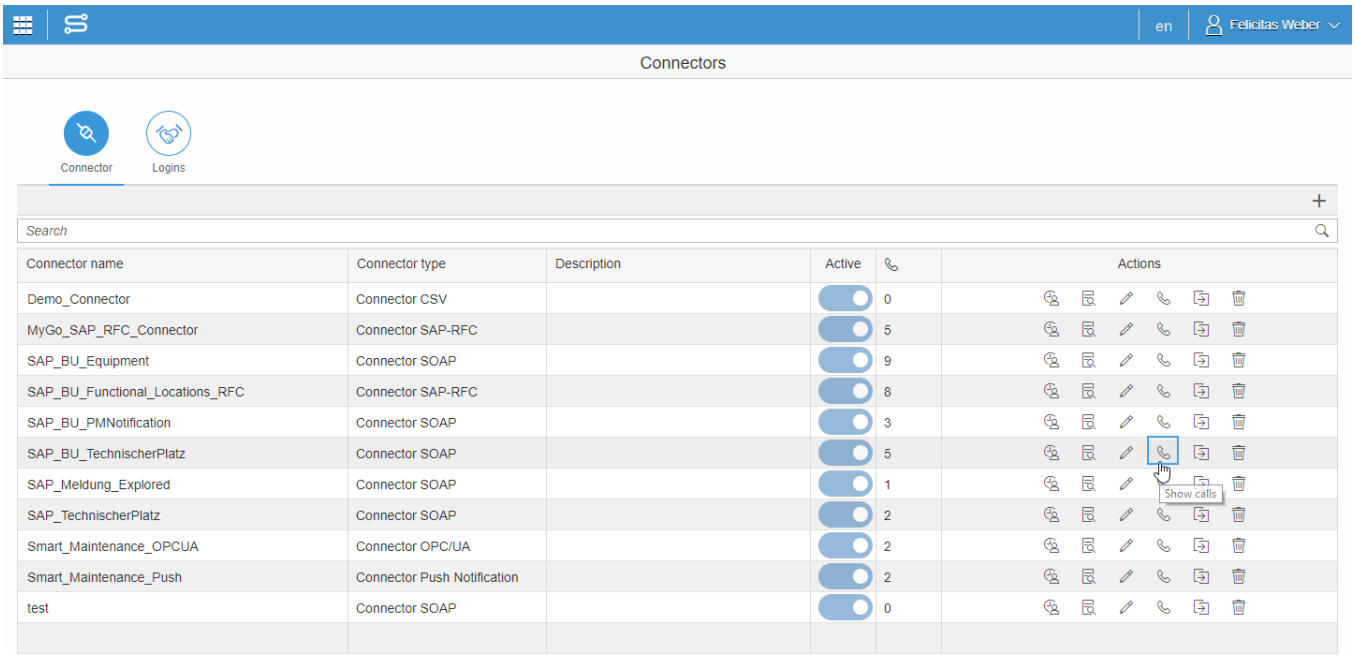
A connector call represents one (sub)-operation of a connector, for example a soap wsdl has one or many operations – here you can customize each one as a call.

Attention

To use a connector within an app, you have to configure a specific connector call and assign data types within the configuration! Otherwise the connector will not be visible in the Process Designer.

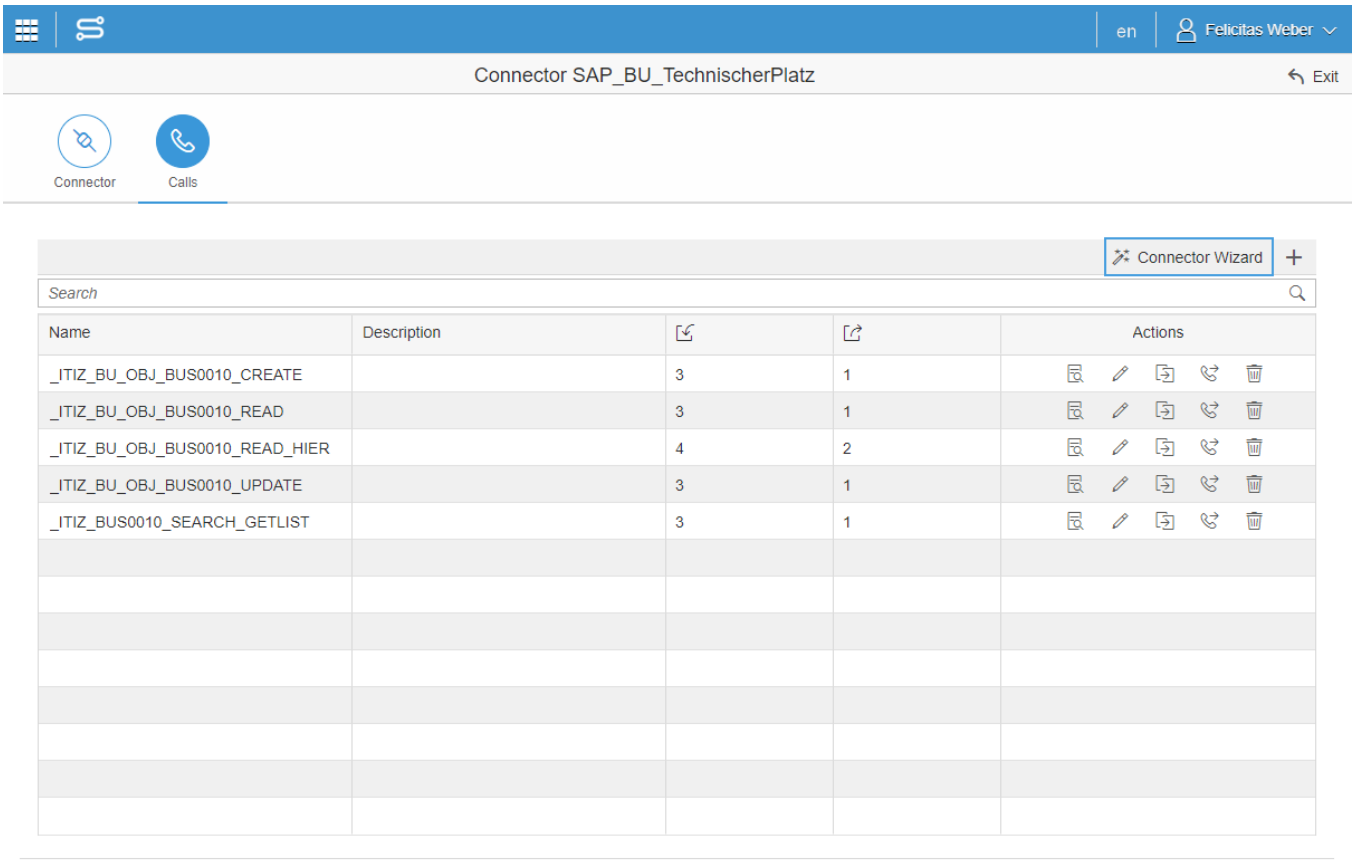
Step 1

Choose a connector from the overview and click the call icon. In the call overview add a new call for each connector operation.

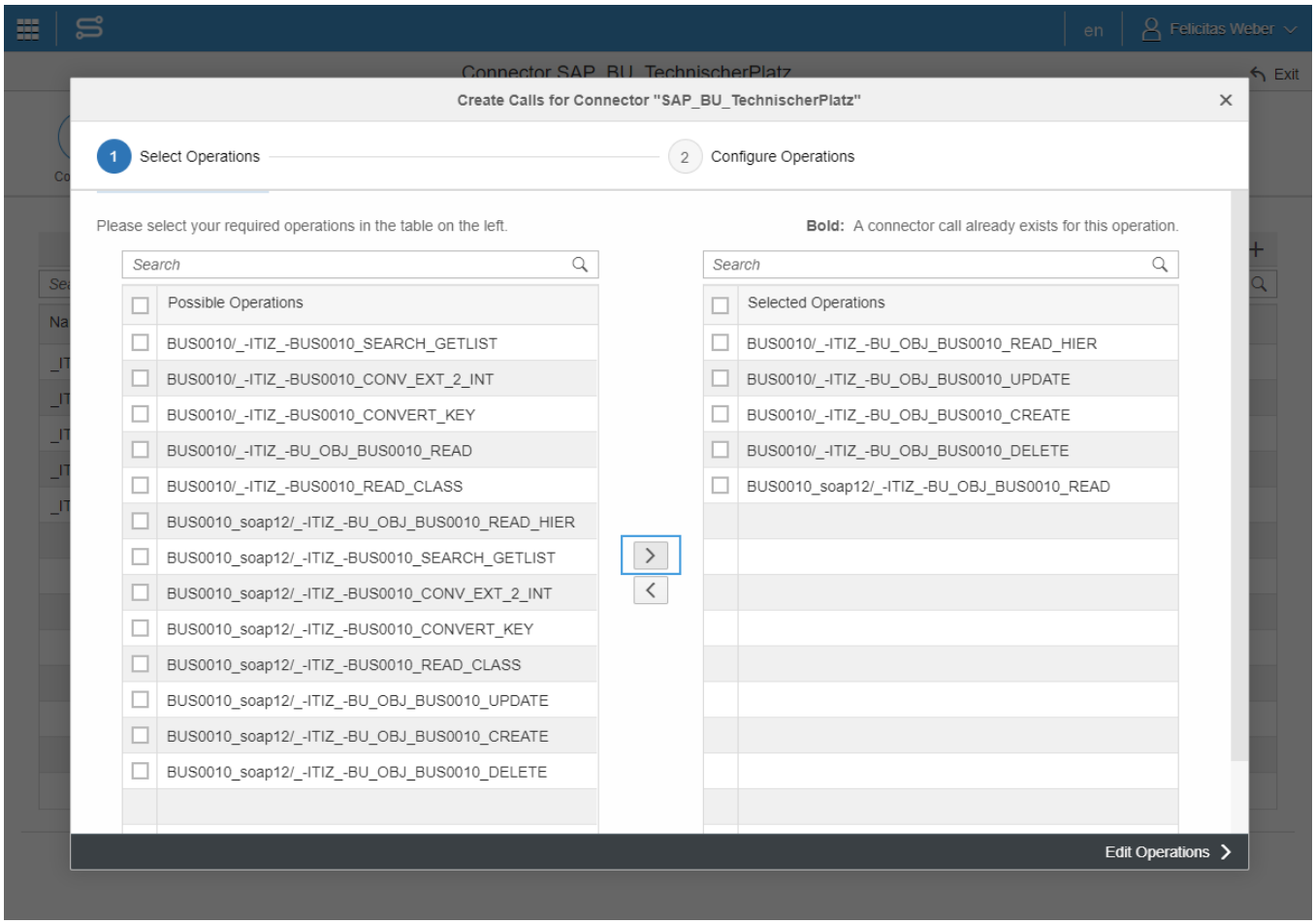


Step 2

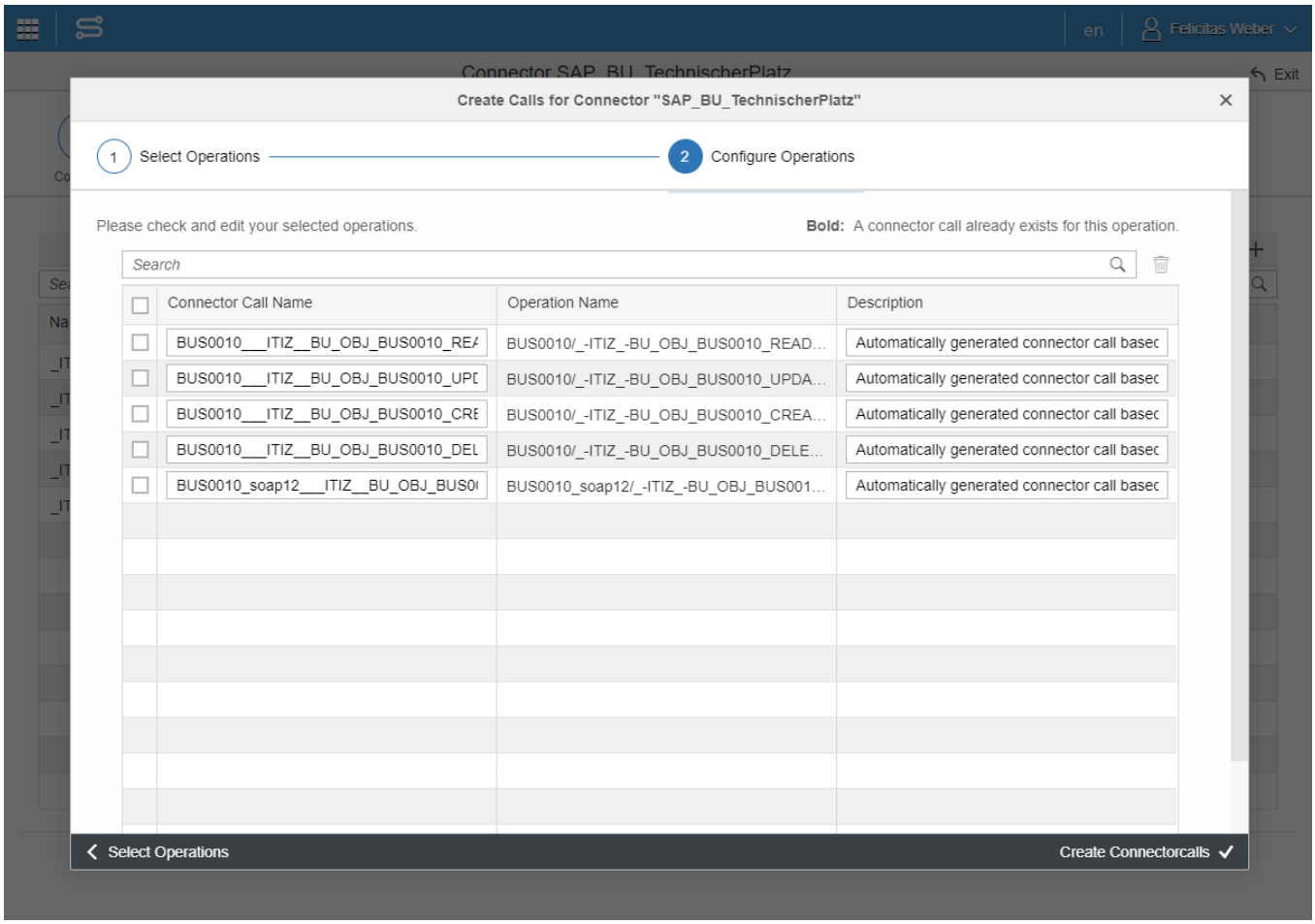
For SOAP and SQL connectors, you have the possibility to use the **Connector Wizard**. It helps you to create your connector calls much easier and faster. If you click on it, you can choose the ones that you need.



Click on the Connector Wizard.

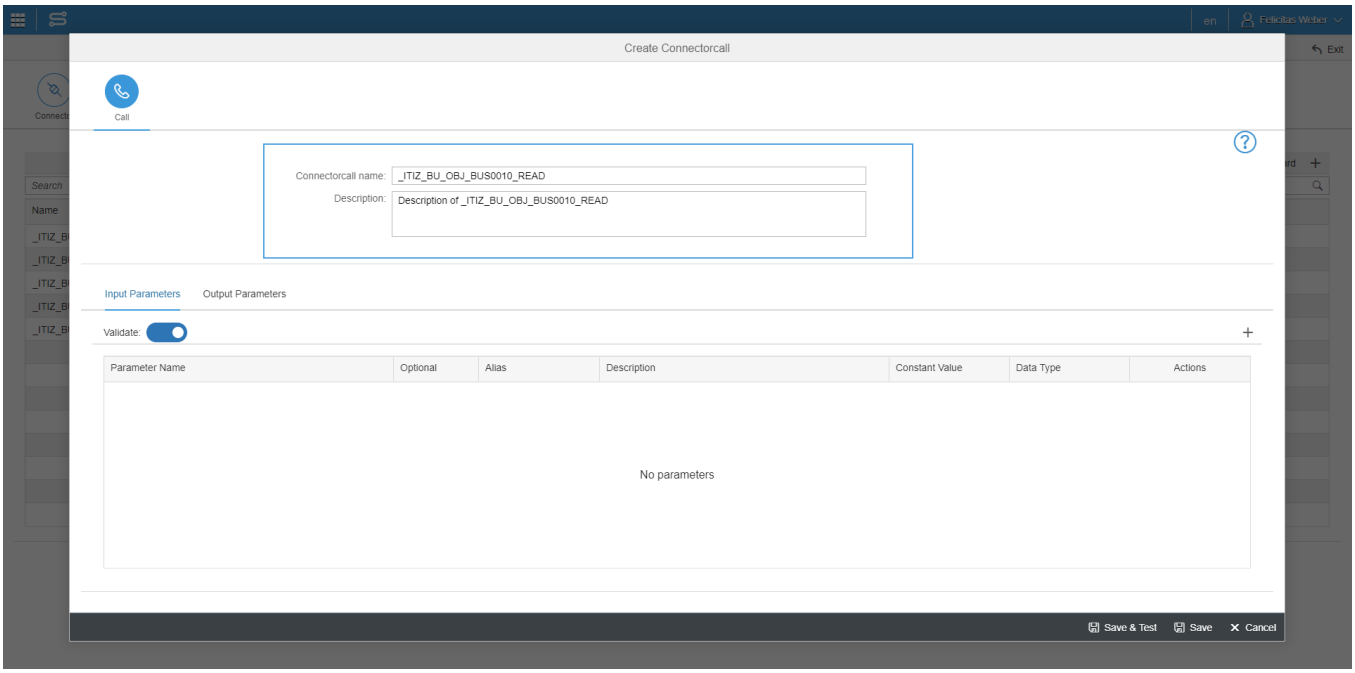


Choose the ones that you need on the left side and move them to the right with the upper arrow.



In the next step you can edit the automatically generated connector calls.

Otherwise click on the plus icon in the upper right and enter a unique call name that describes the operation (e.g. read, write, update, delete, search, …).



Create a connectorcall.

Connector Call name

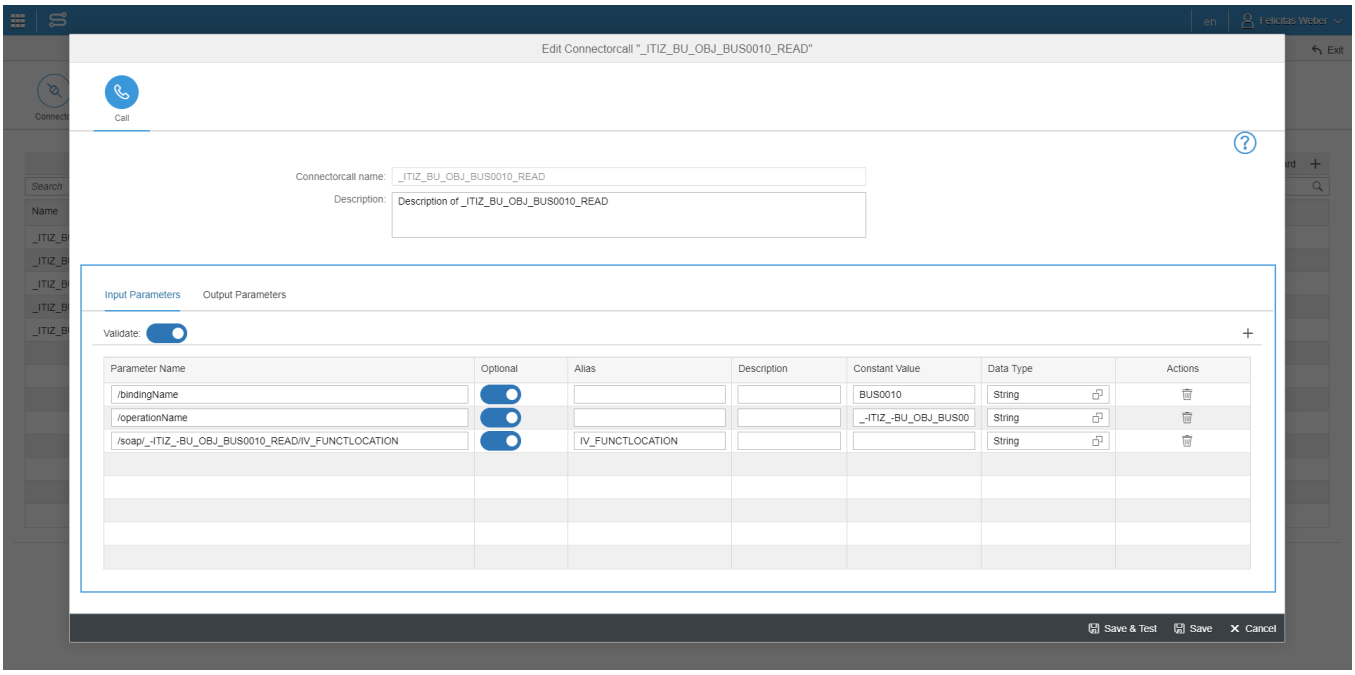
Unique name without spaces to describe the operation.

Description

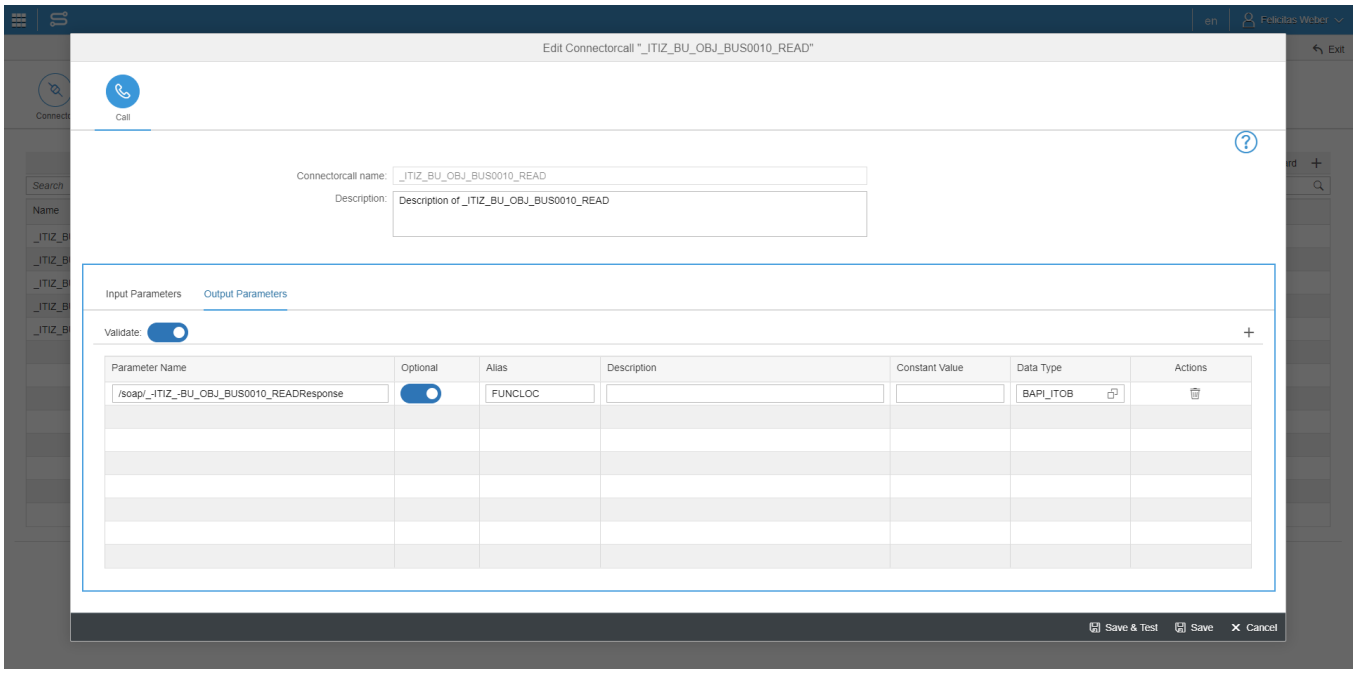
Description of the operation.

Step 3

For configuring a connector call, you have to specify input and output parameters in the following tables:
Each connector call has its own specific parameters.



Specify the input parameters.



Specify the output parameters.

Validate

You can validate the Input and Output parameter in the backend. It validates:

- Base type against type security
- Domain type against security and restrictions
- Structures against type security and underlying property types
- Collections against type security and the underlying types / property categories

If the validation is **not** successful, the client is notified of all failed validations and it's written to the Connector log or System log at the same time.

For every new Connector Call, this flag is set by default. Already existing Connector Calls **do not** have this checkbox flagged to guarantee the compatibility.

Parametername

The technical path or name within a rest api definition or web service description language or csv header column.

Alias

A meaningful non-technical description for the technical parameter. This wording is used in the edit mode for a user story (Process Designer) for mapping data with ui elements.

Description

Optional description of the parameter.

Constant Value

A constant value like SAP Client or company code that can’t be overwritten by any business apps. The value will be validated, so that it’s not possible to use a constant value with a wrong base type in Connector Calls and Buisness Objects.

Data Type

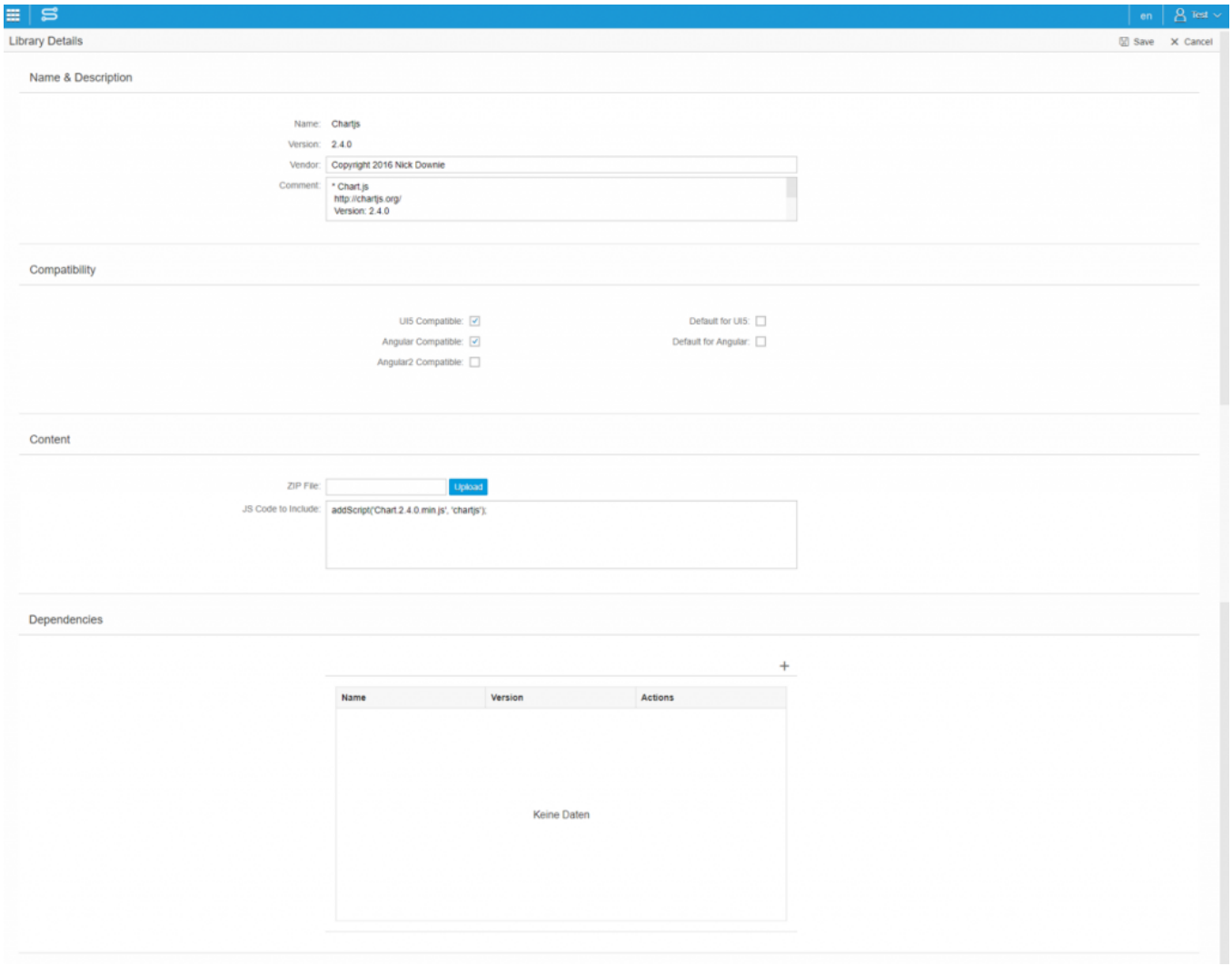
Assigned Simplifier data type for validating data before it gets back or from a backend system.

Step 4

After finishing the parameters, you can save the connector call settings.

Add a new Library

To add a new library click on the “+” in the right corner.



Now you can fill in the following parameter:

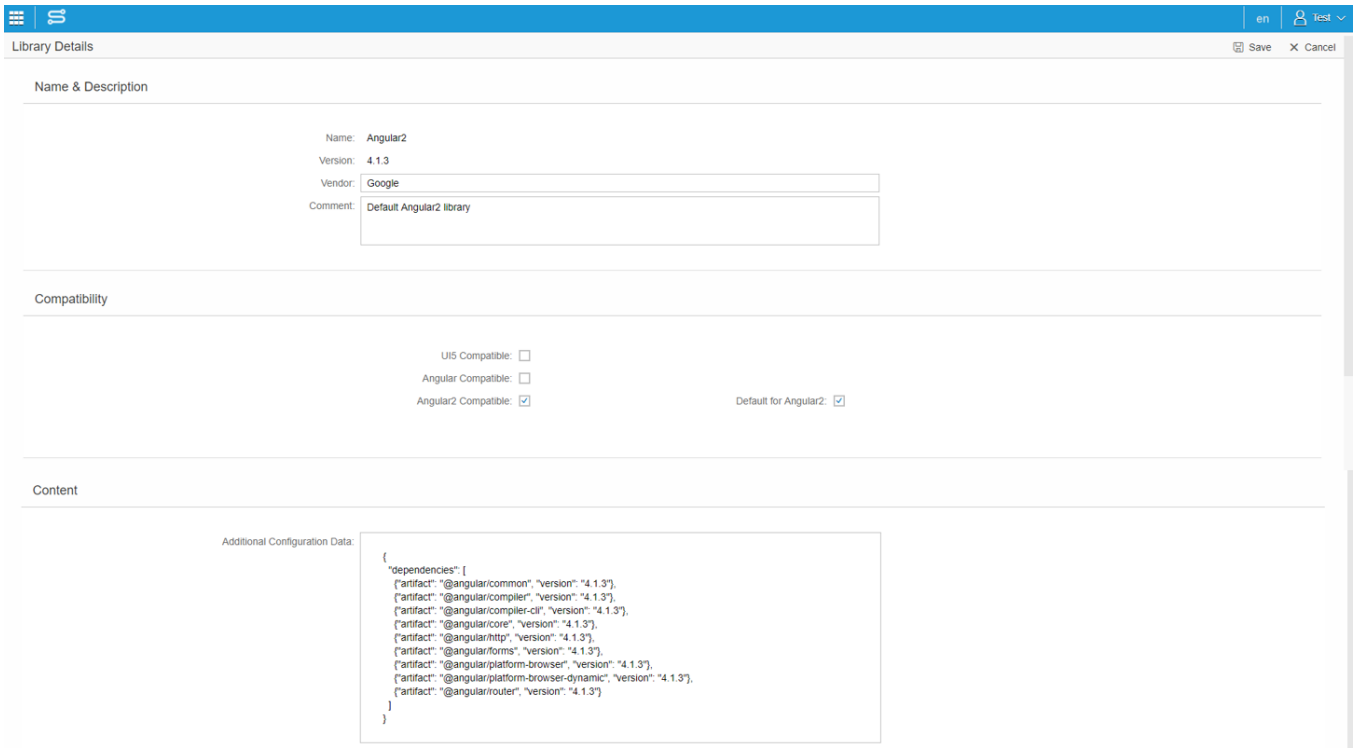
	Parameter	Description
Name & Description	Name	The name of the library. NOTE: The combination of name and version number must be unique!
	Version	The version of the library.
	Vendor	The vendor of the library.
	Comment	A description of the library, or e.g. licence information.
Compatability	UI5 compatible	Controls the assignment to UI5 Apps.

	Angular JS compatible	Controls the assignment to Angular JS Apps.
	Angular 2 compatible	Controls the assignment to Angular 2 Apps.
	Default for UI5	Assigns the library automatically when creating UI5 Apps.
	Default for Angular JS	Assigns the library automatically when creating Angular JS Apps.
	Default for Angular 2	Assigns the library automatically when creating Angular 2 Apps.
Content	ZIP file	The ZIP file, that contains the library.
	JS code to include	Code snippet to integrate the library into Apps.
Dependencies	Dependencies	Dependencies to other libraries.

Angular 2 libraries

If you mark a library as “Angular 2 compatible”, you have to fill in additional configuration data in the Content Area (you don’t need a file upload for Angular 2 libraries anymore).

In this additional configuration data field, you can configure npm dependencies of your Managed Library and import custom components. With this high level configuration it is possible to include any Angular 2 specific npm package and configure vanilla Angular 2 apps.


en
test
Library Details
Save
Cancel
Name & Description
Name: Angular2
Version: 4.1.3
Vendor: Google
Comment: Default Angular2 library
Compatibility
UI5 Compatible:
Angular Compatible:
Angular2 Compatible:
Default for Angular2:
Content
Additional Configuration Data:
{
"dependencies": [
{"artifact": "@angular/common", "version": "4.1.3"},
{"artifact": "@angular/compiler", "version": "4.1.3"},
{"artifact": "@angular/compiler-cli", "version": "4.1.3"},
{"artifact": "@angular/core", "version": "4.1.3"},
{"artifact": "@angular/http", "version": "4.1.3"},
{"artifact": "@angular/forms", "version": "4.1.3"},
{"artifact": "@angular/platform-browser", "version": "4.1.3"},
{"artifact": "@angular/platform-browser-dynamic", "version": "4.1.3"},
{"artifact": "@angular/router", "version": "4.1.3"}
]
}

Add a PDF Template

Add Template

To add a template, you need the following parameter:

URL	/client/1.0/PLUGIN/pdfPlugin/adminTemplateAdd	
Input-Parameter	**Name**	Template name
	Data	Template content (Base64-coded)
	Stylesheet	Content of the LESS Stylesheets (Base64-coded, optional)
	PreviewJson	Content of the sample data in JSON format (Base64-coded, optional)
Output-Parameter	None	

Example for a call:

```
{     "name": "templatename",     "data": "SGFsbG8gV2VsdA==\",     "stylesheet: "SGFs
bG8gV2VsdA==\",     "previewJson": "SGFsbG8gV2VsdA==\" }
```

Output example:

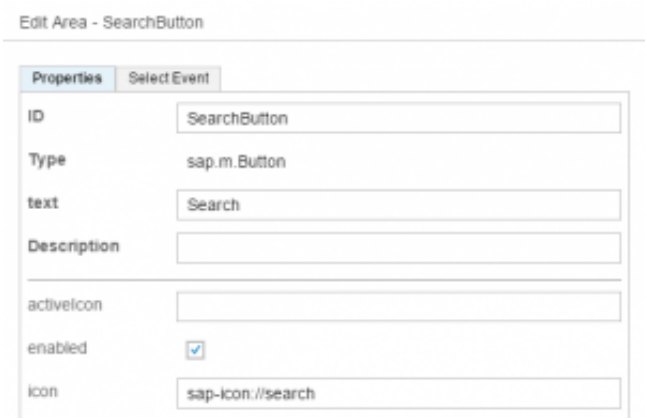
```
{     "success": true }
```

Add an Icon to a Widget

If you want to set an icon to a Widget, use the following syntax:

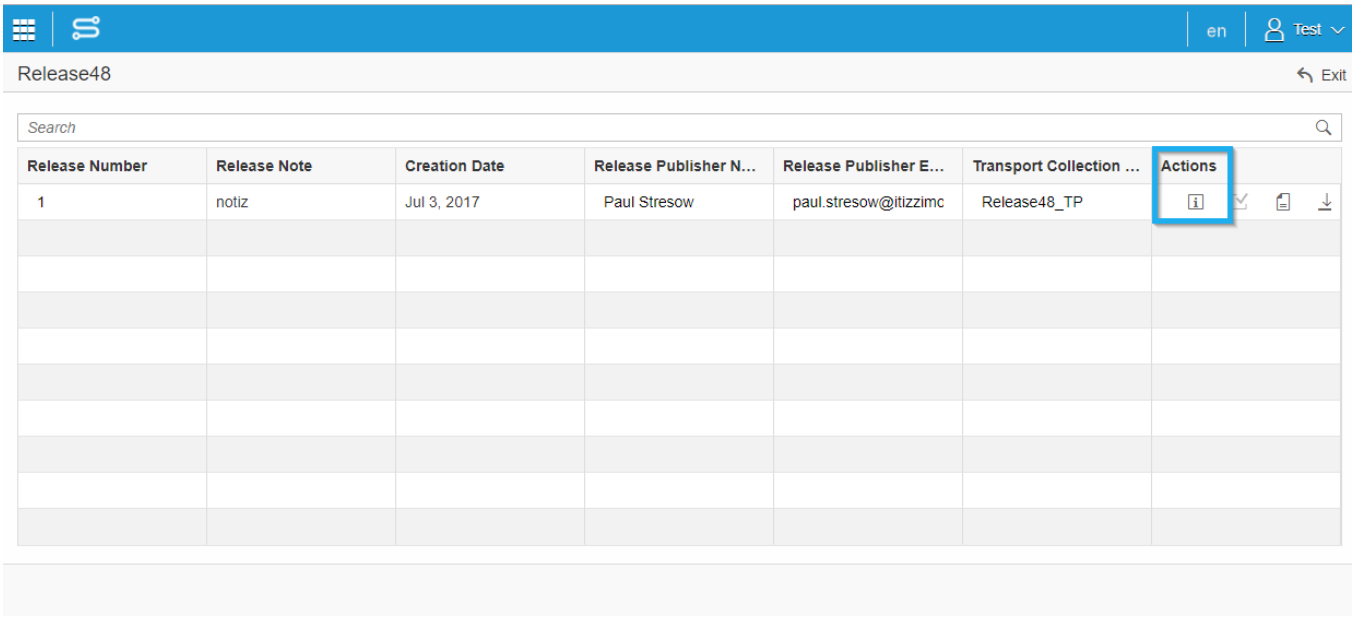
sap-icon://iconName

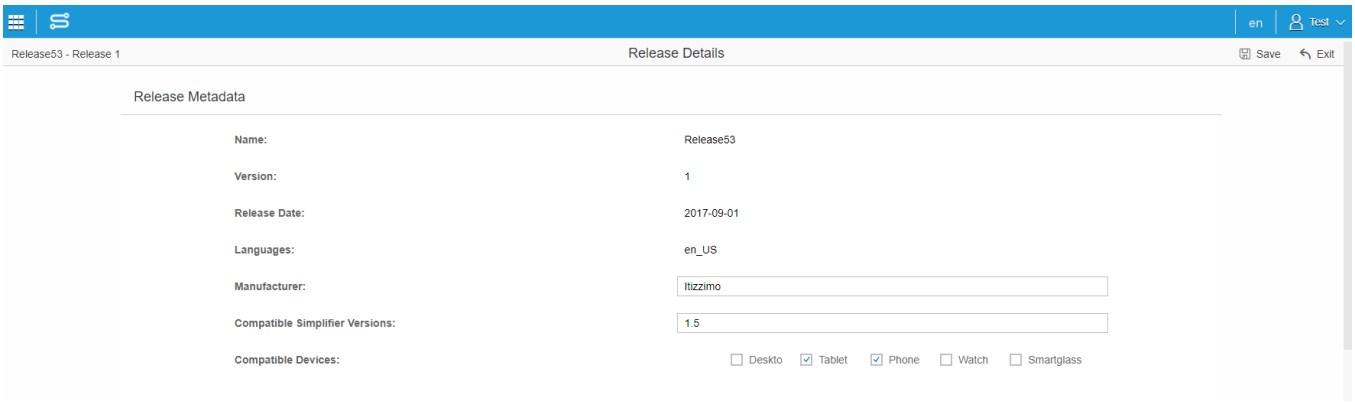
You can find all possible Icons with the OpenUI5 Icon Explorer.

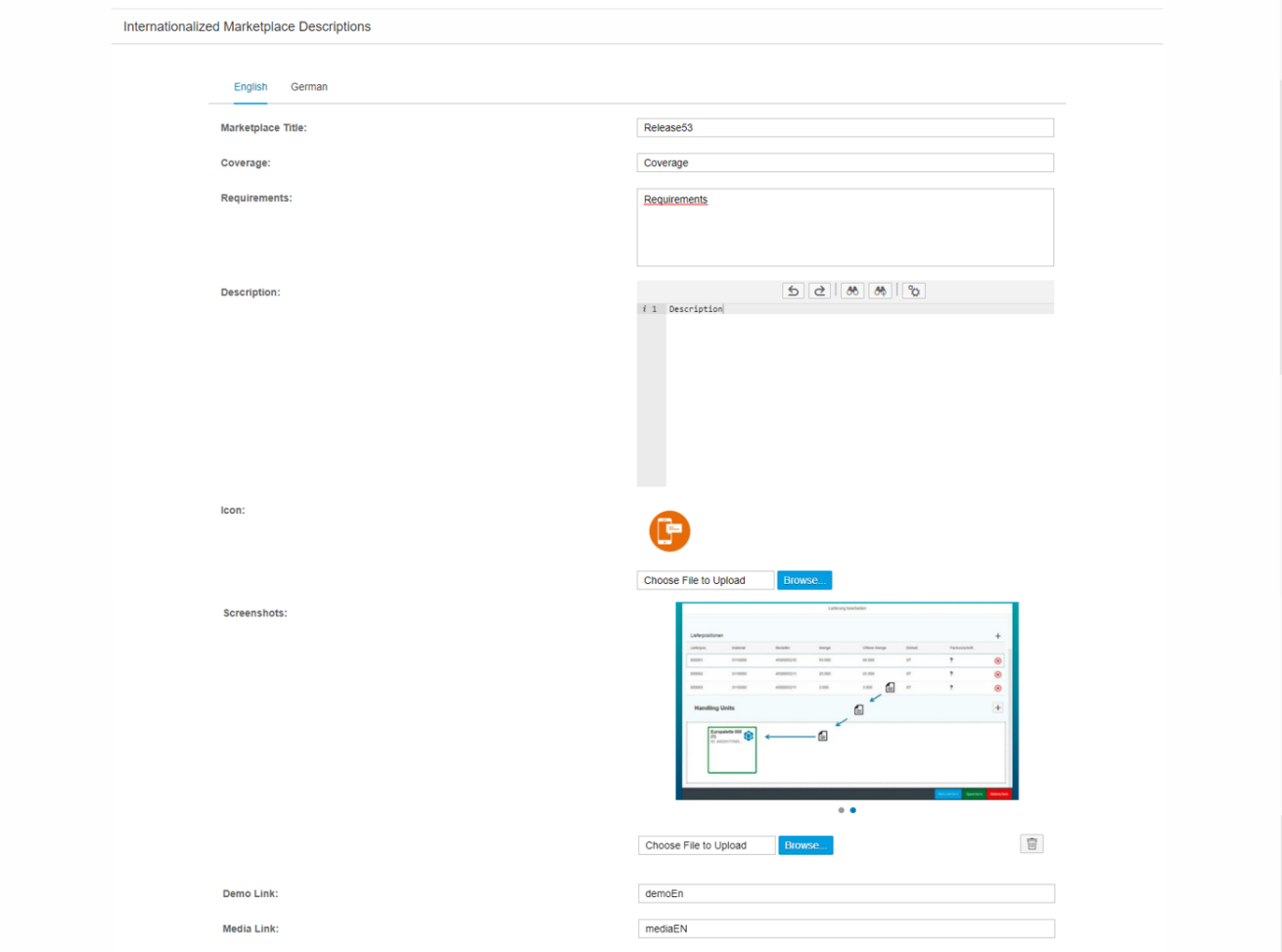


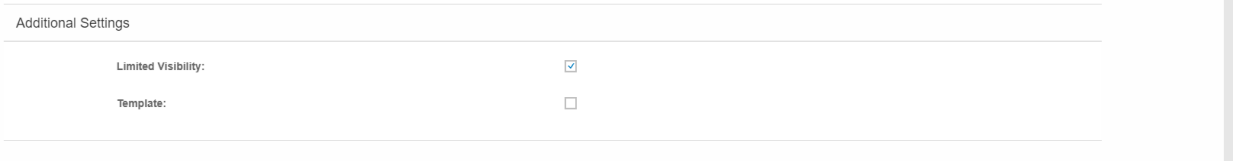
Add Metadata to the Release

With this Action Button in the Release Overview, you will be redirected to the metadata dialog.




en
Test
Release53 - Release 1
Release Details
Save
Exit
Release Metadata
Name:
Release53
Version:
1
Release Date:
2017-09-01
Languages:
en_US
Manufacturer:
Itizzimo
Compatible Simplifier Versions:
1.5
Compatible Devices:
Deskto
Tablet
Phone
Watch
Smartglass





Here you can insert additional information like requirements, compatible devices or add screenshots of the application.

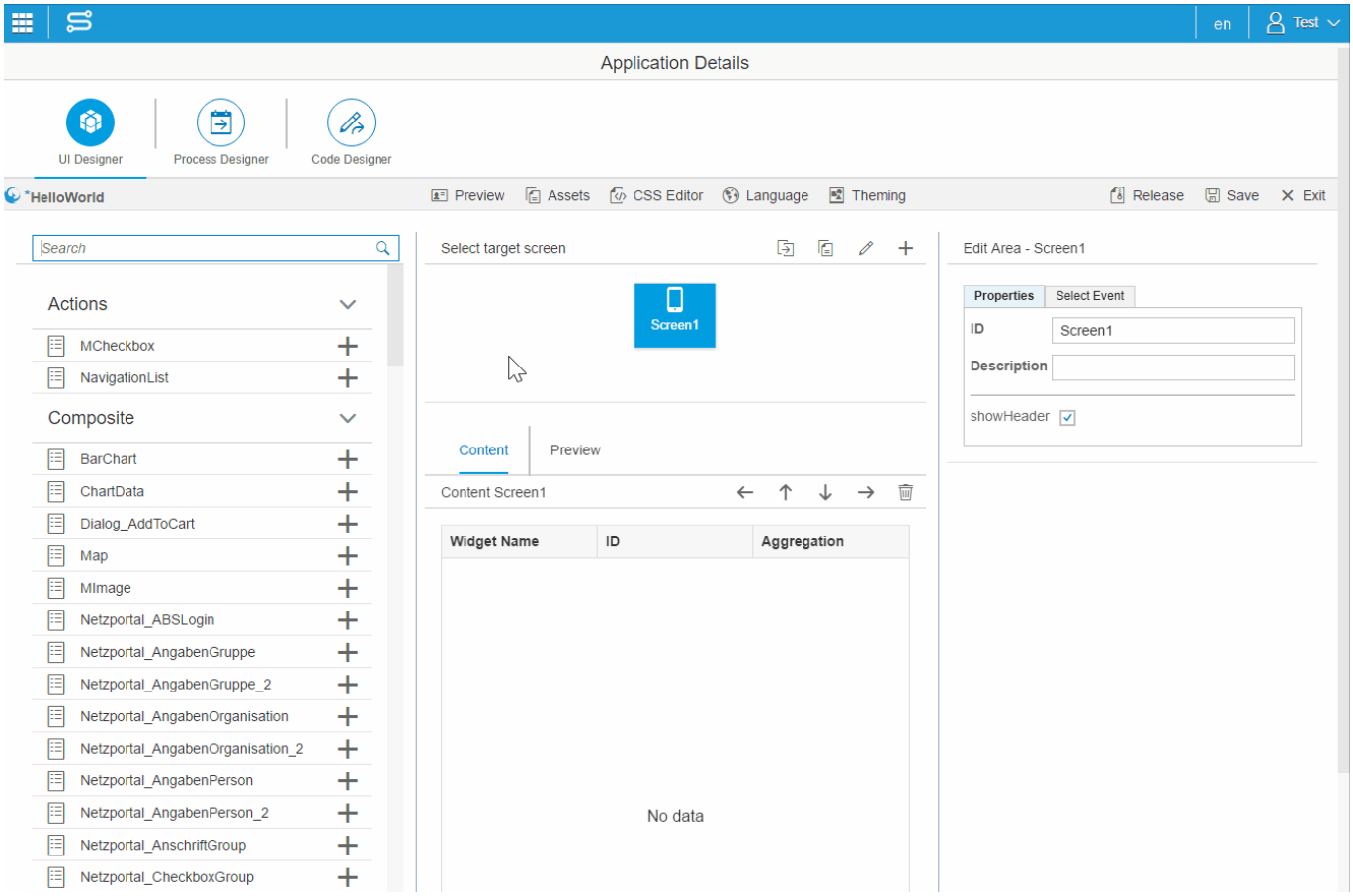
The metadata will be displayed in the Marketplace, if you upload the app. You can maintain the metadata in German or English.

	Data	Description
Release Metadata	Name	The Name of the App (will be set by the Simplifier).
	Version	The Version of the Release (will be set by the Simplifier).

	Release Date	The Release Date (will be set by the Simplifier).
	Languages	Languages defined in the App (will be set by the Simplifier).
	Manufacturer	The Manufacturer.
	Compatible Simplifier Versions	The Version of the Simplifier on which the App is running.
	Compatible Devices	Compatibility to Desktop/Mobile/Wearables.
Internationalized Marketplace Descriptions	Marketplace Title	The Title of the App that will be displayed in the Marketplace.
	Coverage	The scope of the App.
	Requirements	The System Requirements (like OPC/UA System).
	Description	The App Description.
	Icon	The Application Icon which will be displayed in the Marketplace.
	Screenshots	App Screenshots (will be displayed at the bottom).
	Demo-Link	Optional Demo Link to the App.
	Media-Link	Optional Media Link (for example an embedded YouTube link).
Additional Settings	Limited visibility	Triggers if the App is visible in the Marketplace or only available by a secret link.
	Template	

Add Widgets to the Screen

To fill your screen with UI elements like text fields, charts or action buttons, you can search and choose them from the widget panel on the left side:
First, search for a specific widget using the name or just browse the the list, then add the widget to the screen. The widget will appear in your content area.



Additional Requirements for Oracle Databases as Backend

On-Premise Installation

Oracle as a DB backend for the Simplifier requires some additional server settings, which are listed below. The Simplifier is currently running with MySQL 5.7 and Oracle 11g.

Database Settings within the Oracle Database:

Parameter	Recommended Value
OPEN_CURSORS	3000

Supported Oracle version:

Oracle Database 11g Release 11.2 – 64bit

Desired/recommended instance names (Productive and Test):

simplifierp and simplified

Required tablespaces:

simplifier 5G, Temp 1G, Undo 512 MB, Users 5MB

Oracle user and required roles and permissions:

simplifier, simplifier_np (in Prod and Test) permissions to run DDL

Database Characterset:

AL32UTF8

National Characterset:

UTF8

Default language:

German, Germany

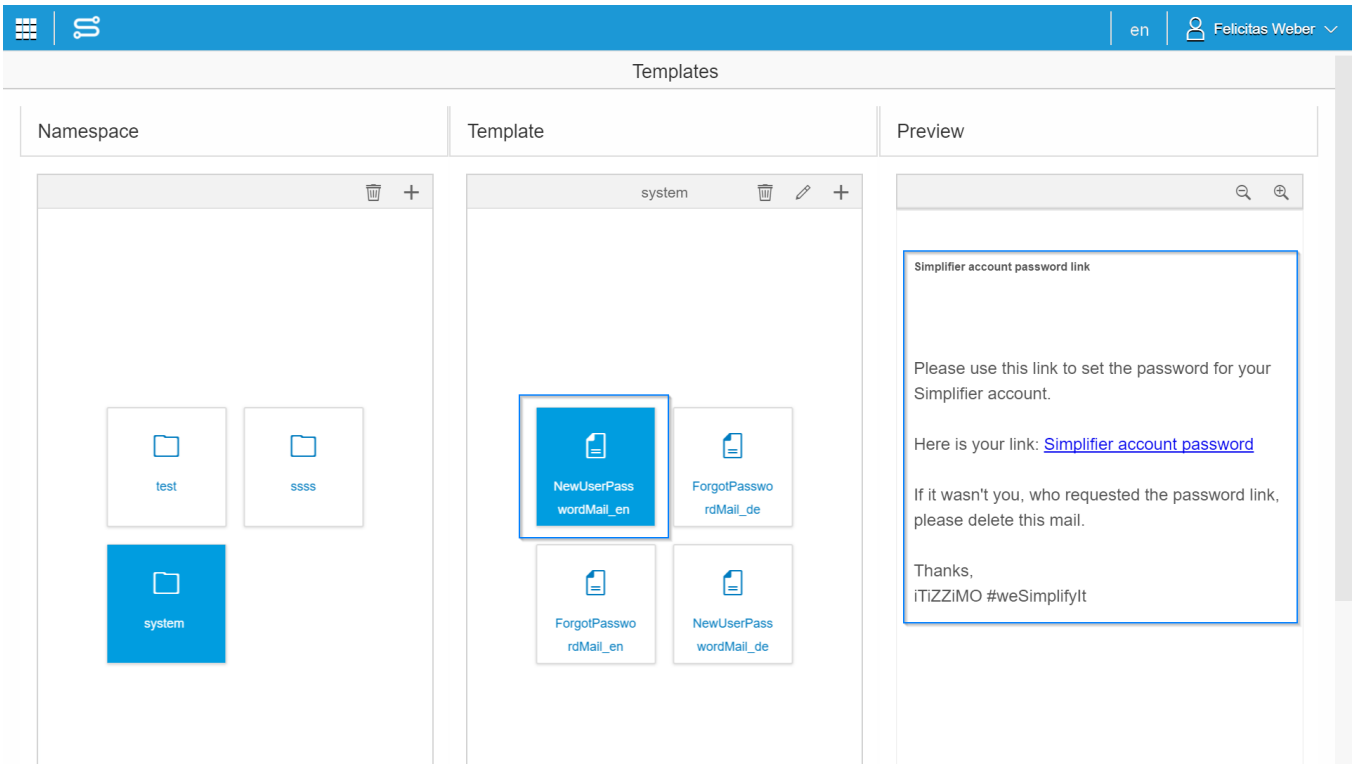
Processes and Sessions:

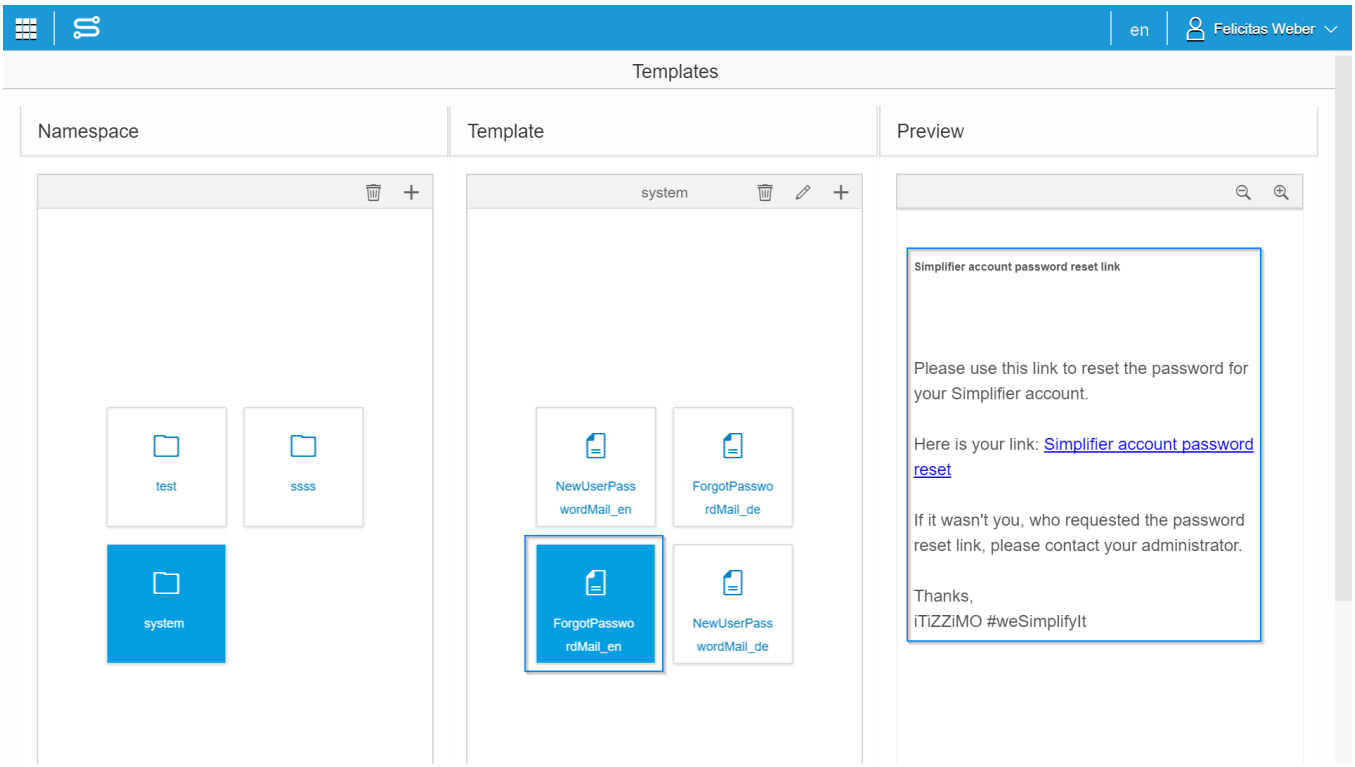
Value to 1000

Administrate Templates

Templates are HTML Templates which for example can be used to send emails. With Templates you can build patterns that you can use consistently.

In our example we use one Template for an e-mail that will be sent to a new user to get a password. We use another Template in case a user has forgotten his password, so that he has the opportunity to reset the password. The Templates are arranged in the Namespace ‘system’.

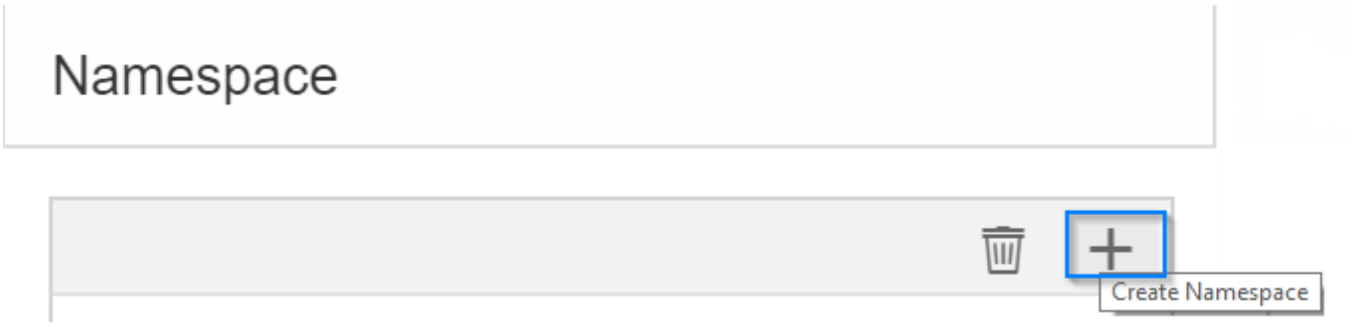

en
Felicitas Weber
Templates
Namespace
Template
Preview
system
test
ssss
system
NewUserPass wordMail_en
ForgotPasswo rdMail_de
ForgotPasswo rdMail_en
NewUserPass wordMail_de
Simplifier account password link
Please use this link to set the password for your Simplifier account.
Here is your link: Simplifier account password
If it wasn't you, who requested the password link, please delete this mail.
Thanks,
iTiZZiMO #weSimplifyIt



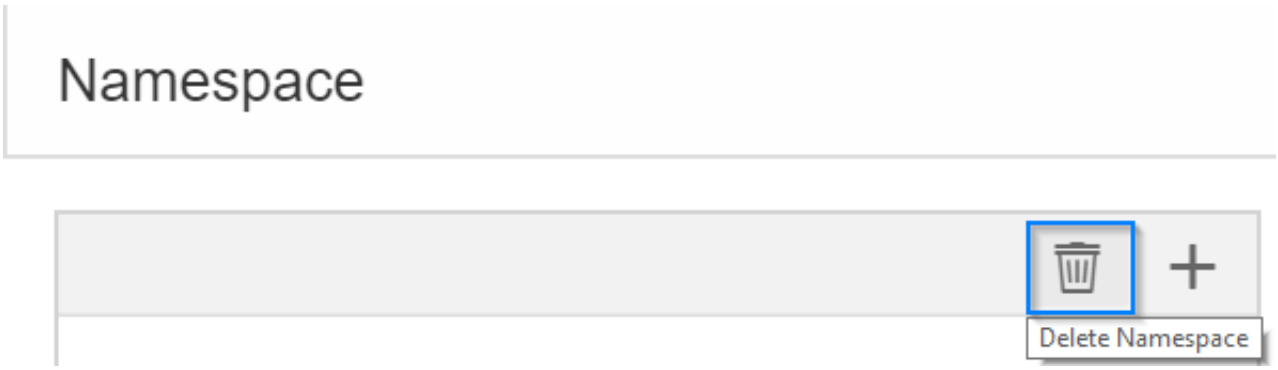
Namespace Section

Namespaces are logical groupings of Templates.

Function	**Description**
Add Button Delete Button	The Add Button allows you to create new empty Namespaces. The Delete Button will delete the Namespace and **all** underlying Templates.



Create Namespace



Delete Namespace

Template Section

You can create and customize the Templates in HTML. With the mustache notation you can add data dynamically at runtime. (For information about how to send the Templates go to: Business Object Script Template – Send Email).

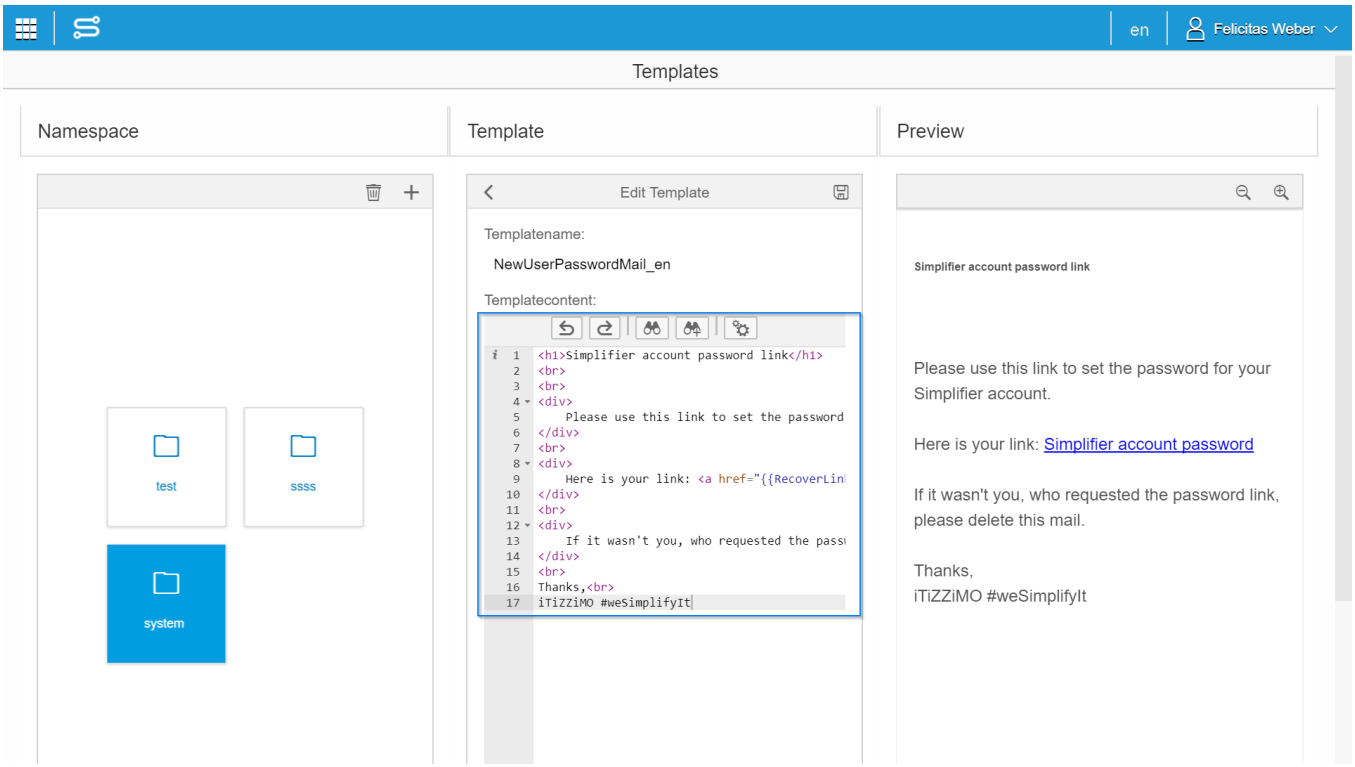
Function	Description
Add Button	Navigates to the detail page for Templates in order to create a new Template.
Edit Button	Navigates to the detail page for Templates in order to edit a Template.
Delete Button	Deletes the Template.

Create Template

Edit Template

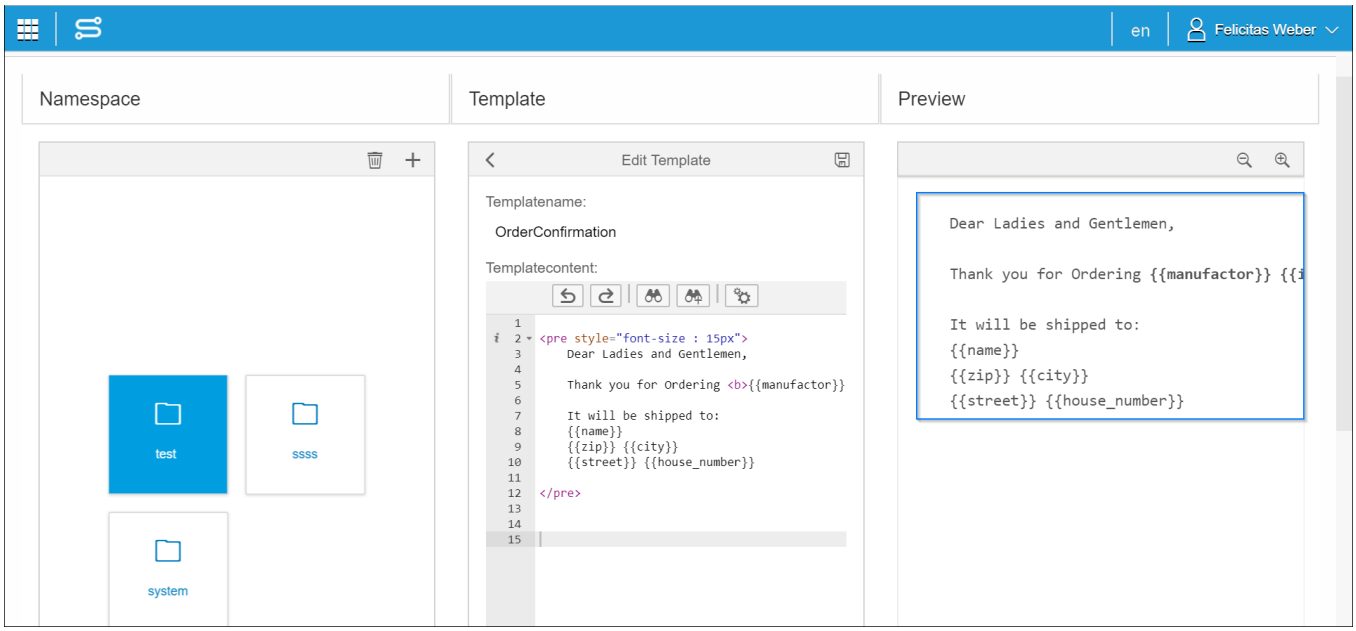
Delete Template

The HTML Templatecontent looks like this:



Preview Section

The Preview Section shows live updates while editing a Template or provides a quick view when selecting a Template.


en
Felicitas Weber
Namespace
Template
Preview
test
ssss
system
Edit Template
Templatename:
OrderConfirmation
Templatecontent:
<pre style="font-size : 15px">
Dear Ladies and Gentlemen,
Thank you for Ordering <b>{{manufactor}}
It will be shipped to:
{{name}}
{{zip}} {{city}}
{{street}} {{house_number}}
</pre>
Dear Ladies and Gentlemen,
Thank you for Ordering {{manufactor}} {{
It will be shipped to:
{{name}}
{{zip}} {{city}}
{{street}} {{house_number}}

Anonymous Profile for Plugins

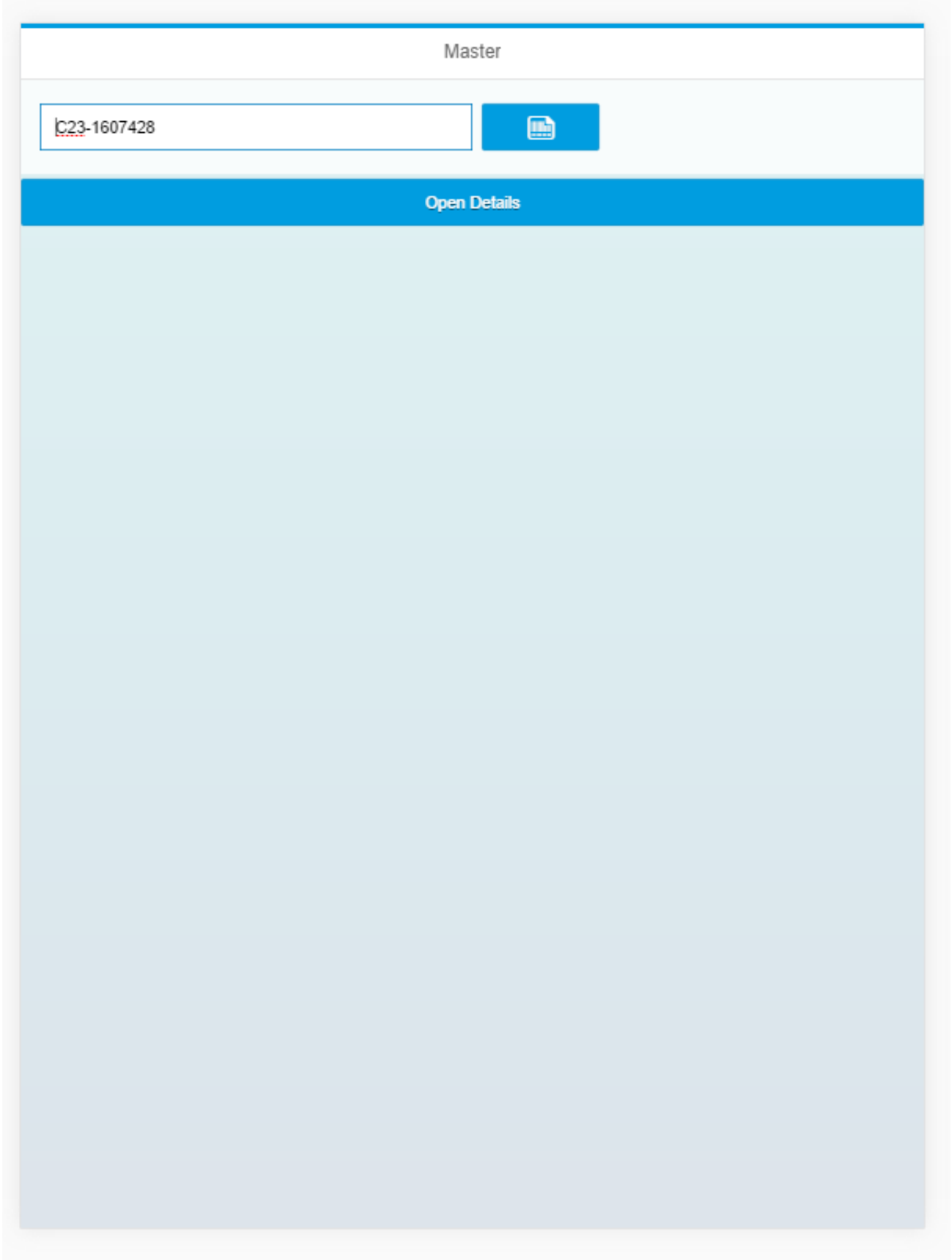
If you want to access plugins, you can work with anonymous users. Therefore the PluginAPI works with AnonymousAppProfile.

So only the assigned role to your Application needs the permission to use the Plugin.

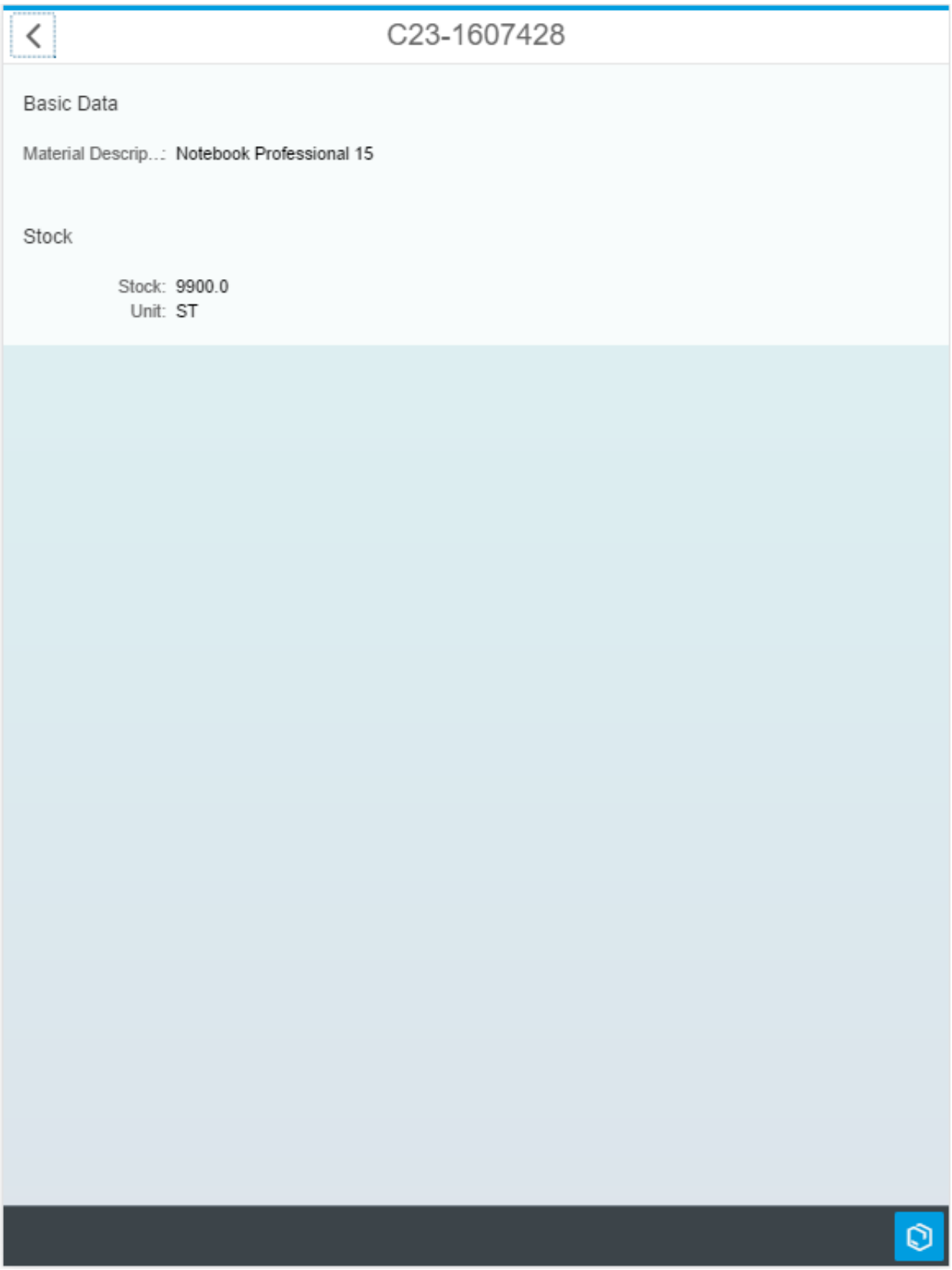
Read more about roles.

App SAP Material

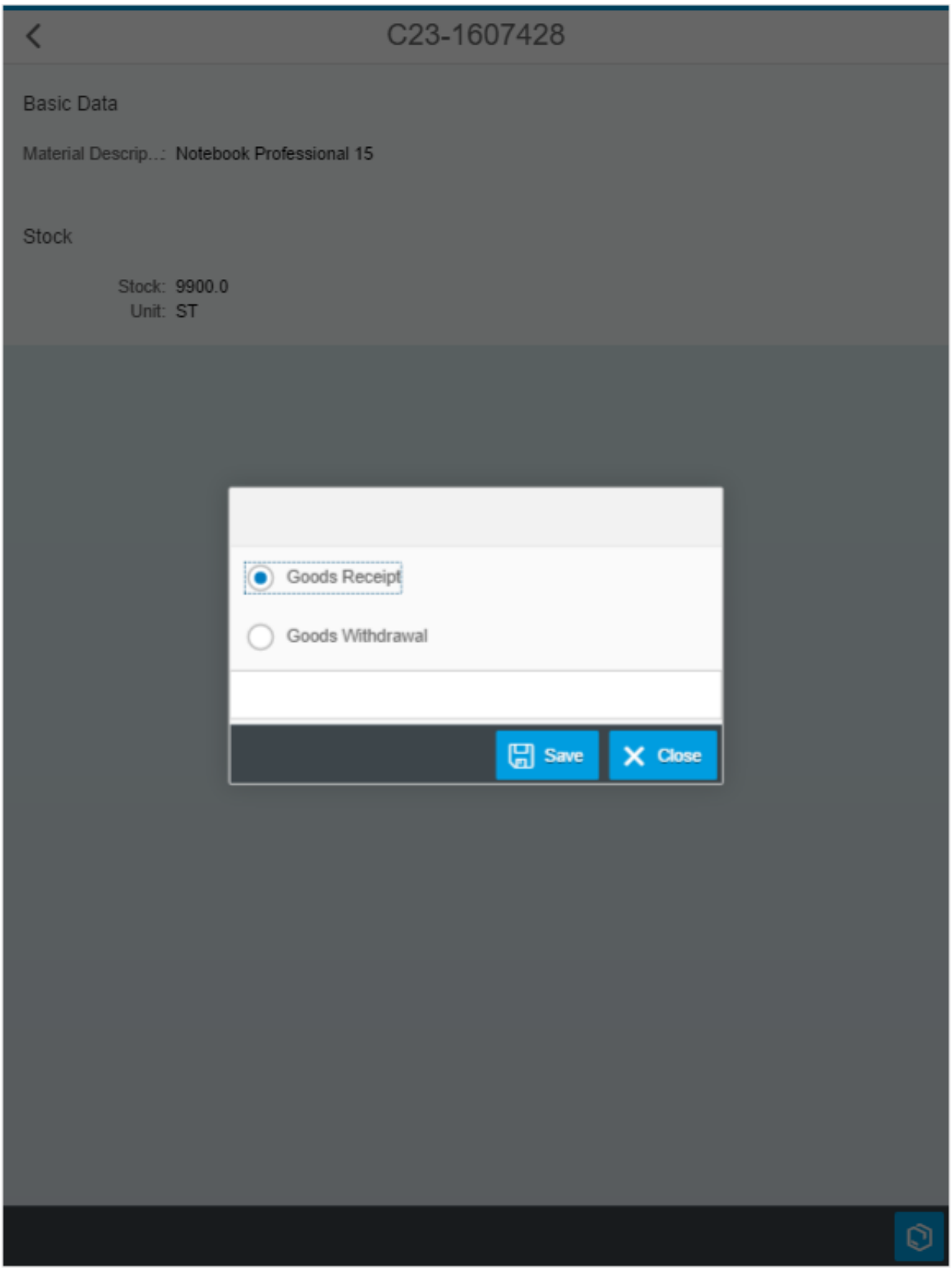
The App “SAP_Material_Academy” shows exemplarily how to search for SAP Material Numbers and display the Material Name and Stock Data. You can reduce the stock through “Goods Withdrawal” or reduce the quantity through “Goods Receipt”.



Screen 1


C23-1607428
Basic Data
Material Descrip...: Notebook Professional 15
Stock
Stock: 9900.0
Unit: ST

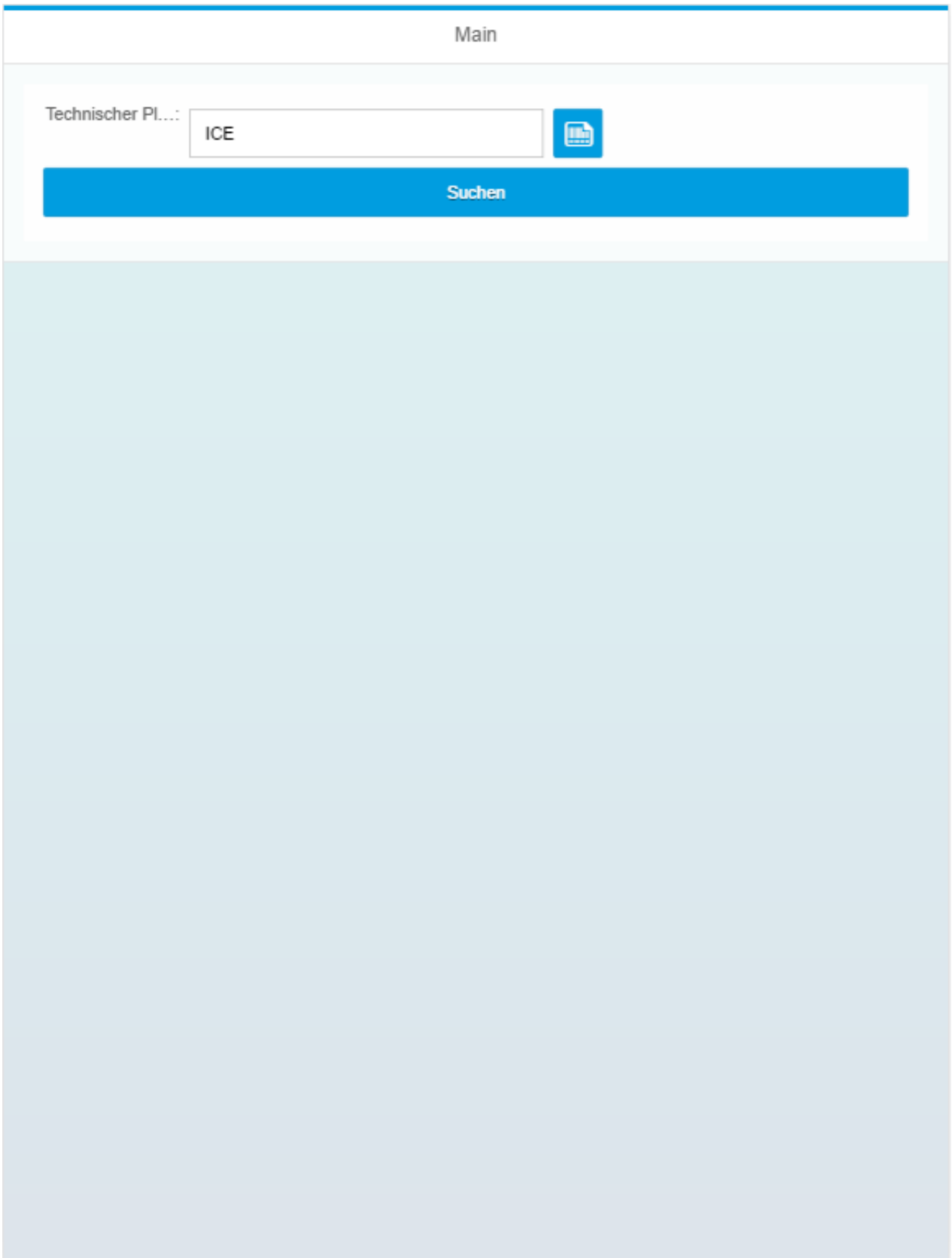
Screen 2


C23-1607428
Basic Data
Material Descrip...: Notebook Professional 15
Stock
Stock: 9900.0
Unit: ST
Goods Receipt
Goods Withdrawal
Save
Close

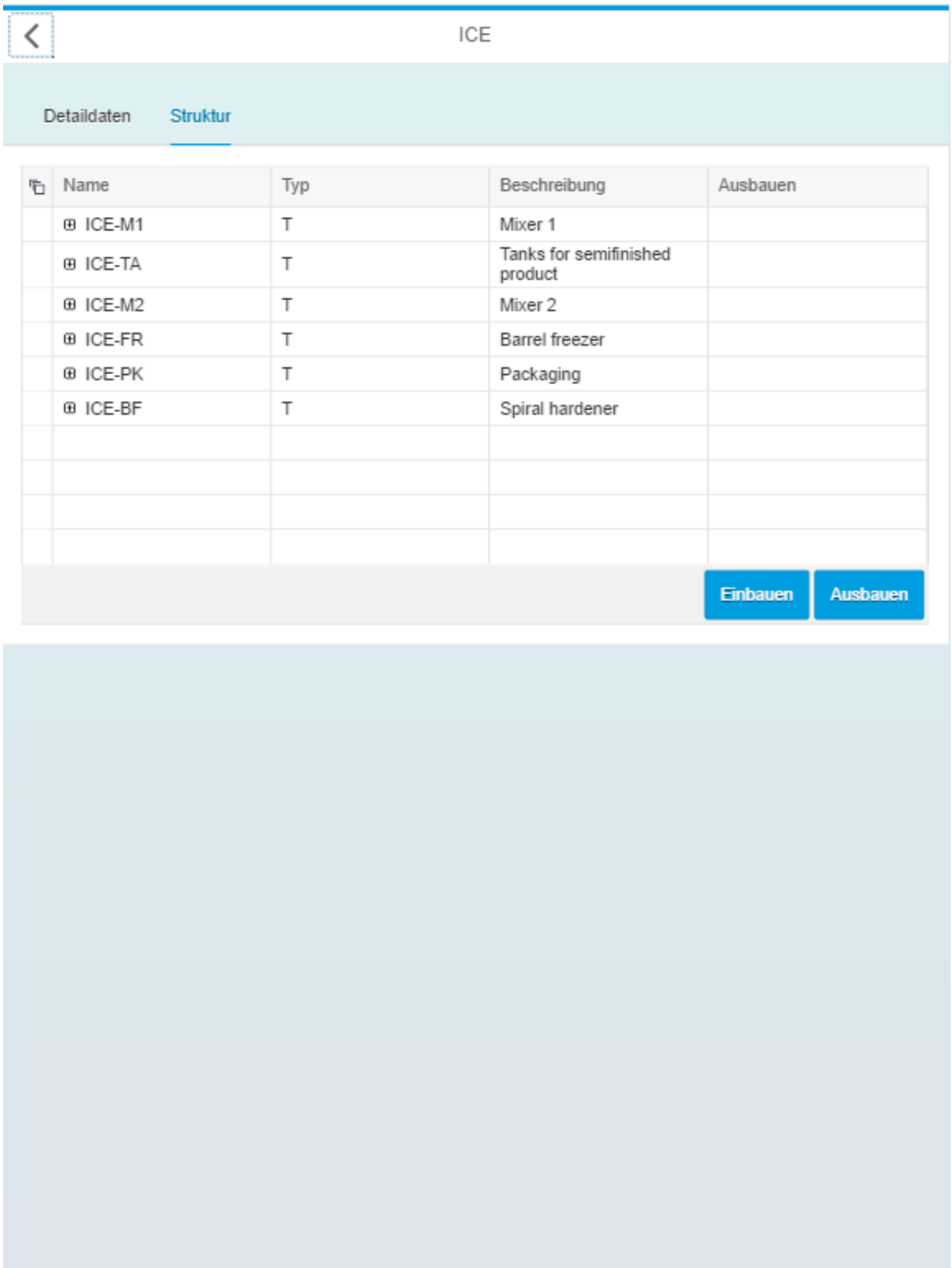
Dialog

App SAP Technischer Platz

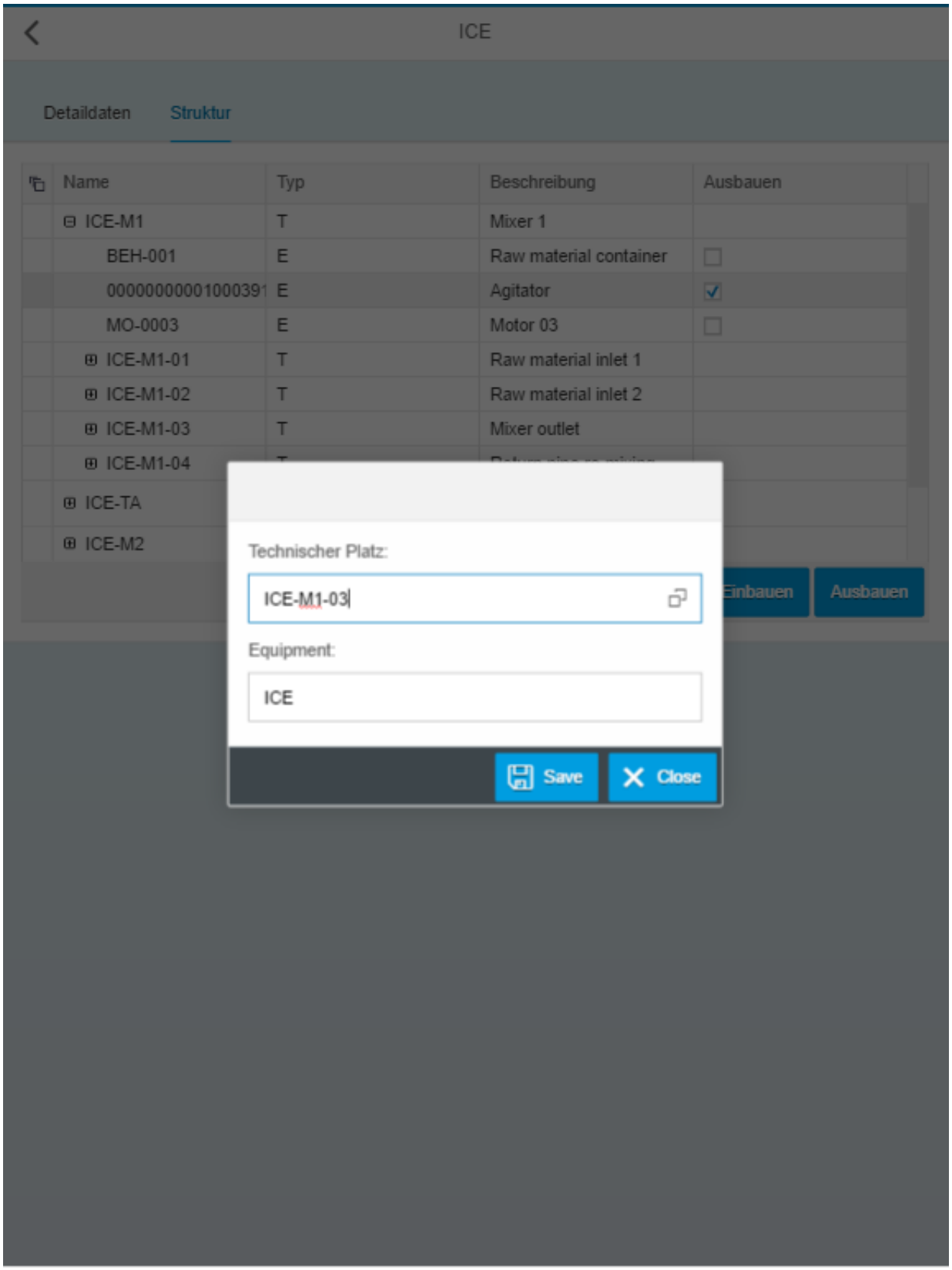
The App “SAP_TechnischerPlatz_Academy” enables you to search for SAP Technical Location and display Header Data with the structure of the Technical Location. You can install and remove structural Items of the Technical Location.



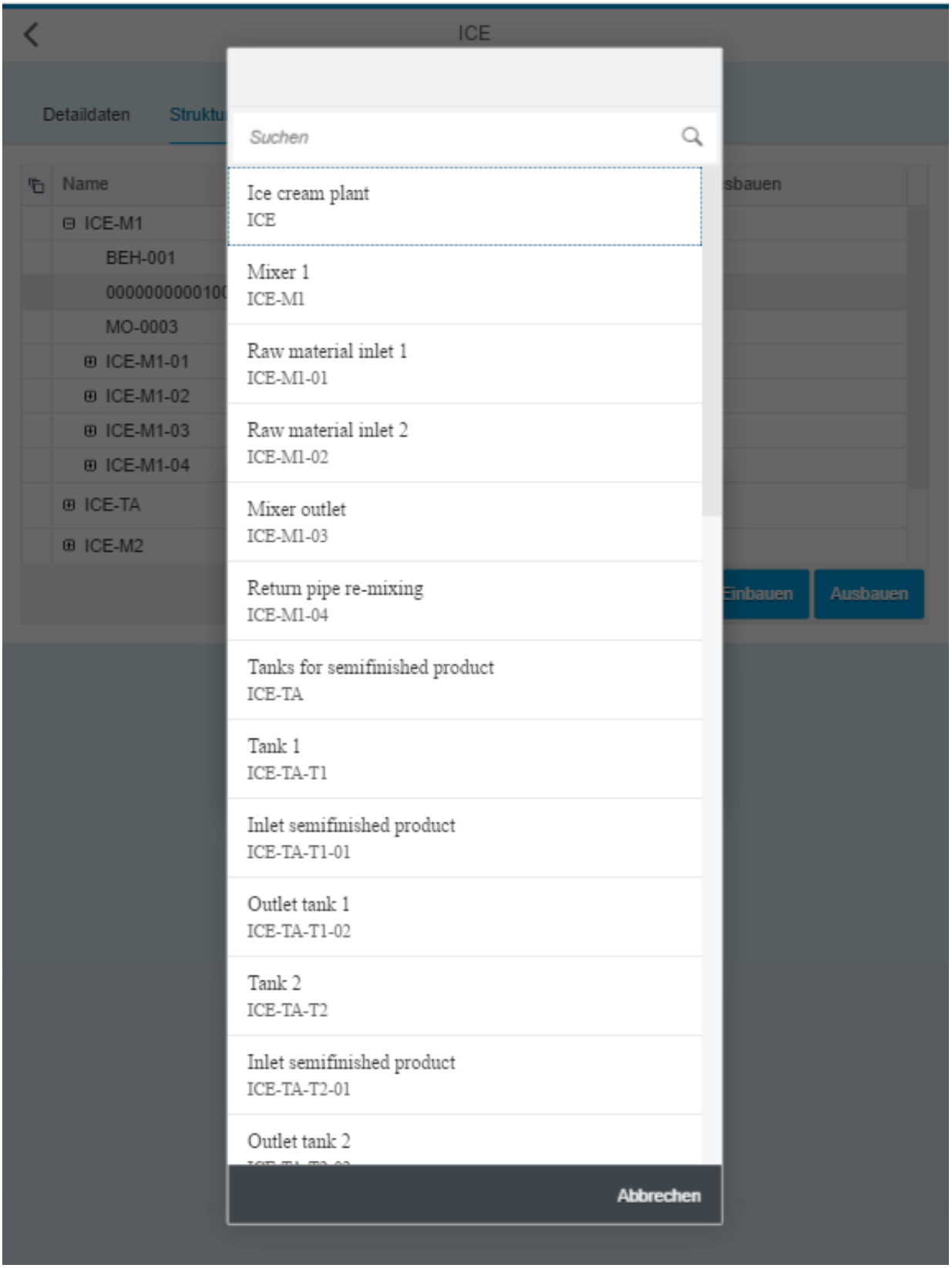
Screen 1


ICE
Detaildaten
Struktur
Name
Typ
Beschreibung
Ausbauen
ICE-M1
T
Mixer 1
ICE-TA
T
Tanks for semifinished product
ICE-M2
T
Mixer 2
ICE-FR
T
Barrel freezer
ICE-PK
T
Packaging
ICE-BF
T
Spiral hardener
Einbauen
Ausbauen

Screen 2


ICE
Detaildaten
Struktur
Name
Typ
Beschreibung
Ausbauen
ICE-M1
T
Mixer 1
BEH-001
E
Raw material container
00000000001000391
E
Agitator
MO-0003
E
Motor 03
ICE-M1-01
T
Raw material inlet 1
ICE-M1-02
T
Raw material inlet 2
ICE-M1-03
T
Mixer outlet
ICE-M1-04
ICE-TA
ICE-M2
Einbauen
Ausbauen
Technischer Platz:
ICE-M1-03
Equipment:
ICE
Save
Close

Dialog


ICE
Detaildaten
Struktu
Name
ICE-M1
BEH-001
MO-0003
ICE-M1-01
ICE-M1-02
ICE-M1-03
ICE-M1-04
ICE-TA
ICE-M2
sbauen
Einbauen
Ausbauen
Suchen
Ice cream plant
ICE
Mixer 1
ICE-M1
Raw material inlet 1
ICE-M1-01
Raw material inlet 2
ICE-M1-02
Mixer outlet
ICE-M1-03
Return pipe re-mixing
ICE-M1-04
Tanks for semifinished product
ICE-TA
Tank 1
ICE-TA-T1
Inlet semifinished product
ICE-TA-T1-01
Outlet tank 1
ICE-TA-T1-02
Tank 2
ICE-TA-T2
Inlet semifinished product
ICE-TA-T2-01
Outlet tank 2
Abbrechen

Dialog 2

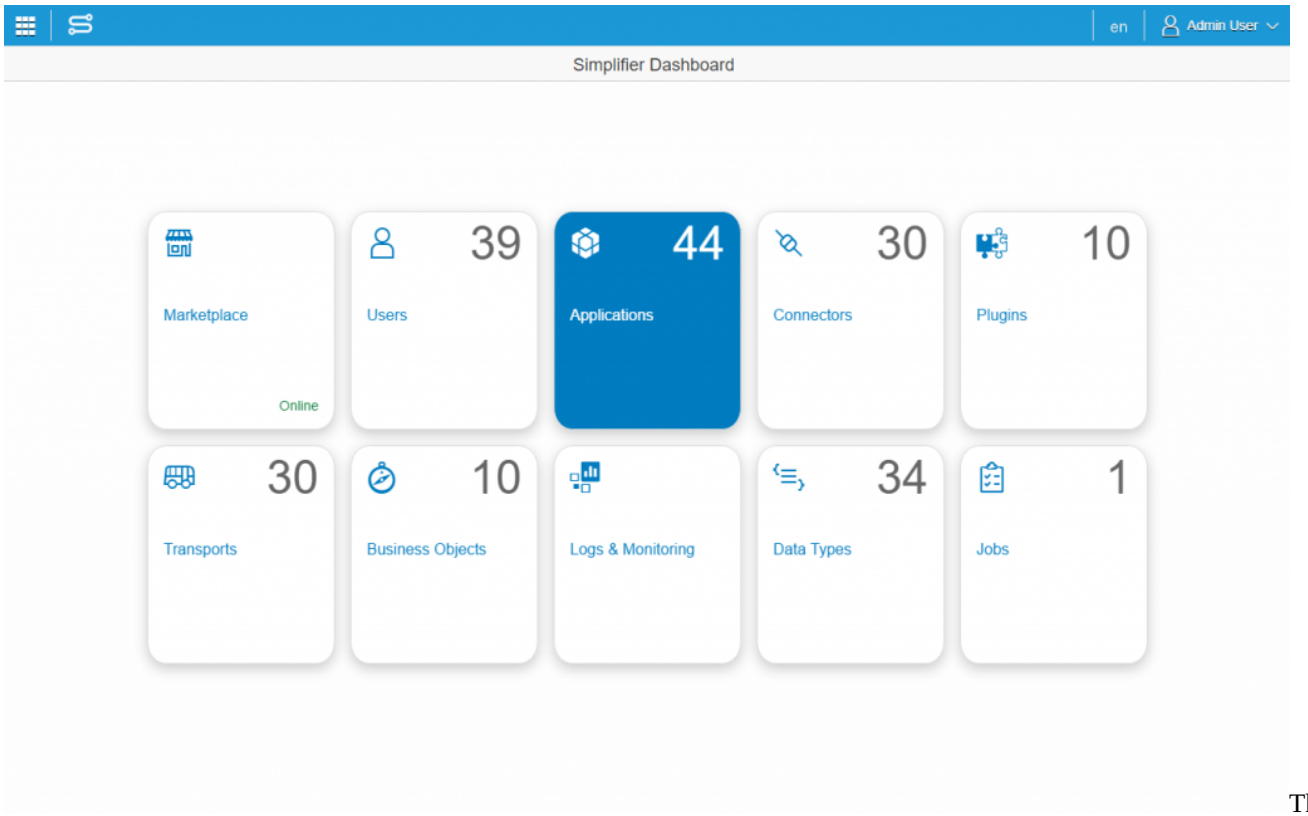
App Simplifier Explored

The App “Simplifier_Explored_Academy” is a feature-based mobile App, showing the native capabilities of the Simplifier running on mobile devices.

It is subdivided into individual User Stories:

Context Features	As a user I want to access context sensitive features of my mobile phone.
Communication Features	As a user I want to establish realtime audio and video communication.
Device Features	As a user I want to be able to see details of my mobile device on a screen within this app.
Data Aggregation	As a user I want to be able to load multiple data results from a Backend and display them in a table.
Device Controls	As a user I want to be able to access native functionality of my mobile device. Taking Pictures, Playing and Recording Audio and Video, Changing the device orientation, Toggle Light.
Media Features	As a user I want to take pictures, play and record audio, show PDF files etc..
Connector Features	As a user I want to read data from backend Systems and display it in my App.
Multi Device Features	As a user I want to use my app on different devices and I want to show different screens based on the device I am using.

Applications



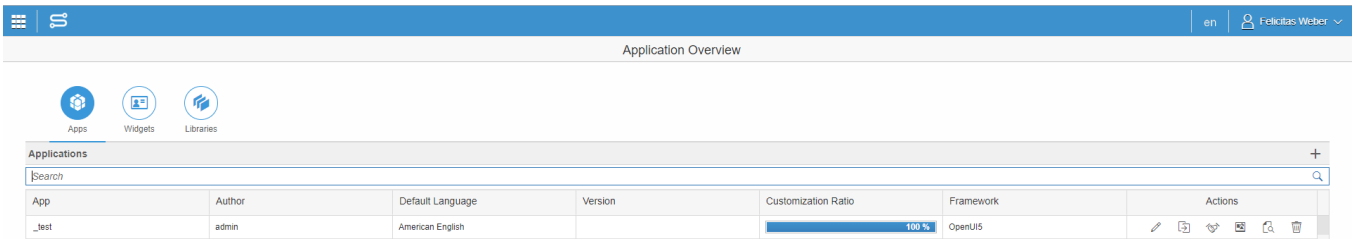
The Simplifier transforms your business process into a configured business application for

- Web Portals for Desktop-Browsers (Internet Explorer, Firefox, Chrome)
- Mobile Phones and Tablets
- Wearables Devices like Smartwatches and Glasses

We use the abbreviation “Apps” for all applications regardless of the device and user. An app should run on any device category because it is generated on common open standard technology.

Overview

By clicking on the Applications tile, you will be lead to the overview. There you will see a table with all created applications. Within this table you’ll get several information like the name of the apps, the author, default language, the version number, customization ratio, framework, and several actions.



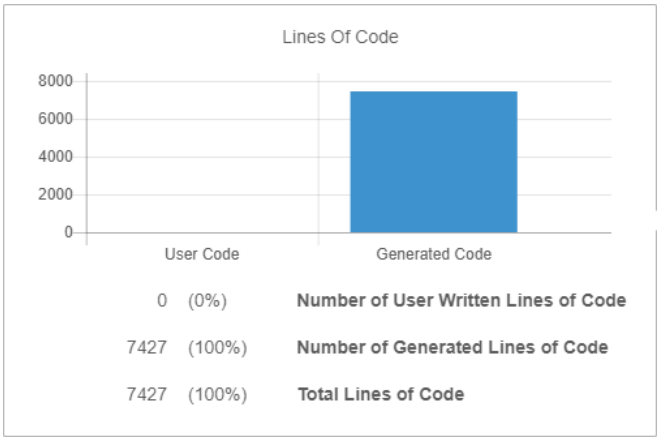
App
It’s the name of the application.

Author
The name of the author who have created the application.

Default Language
American English or German.

Version
It’s the version number of the releases.

Customization Ratio
After deploying an app, a code metric is available in the overview. It shows the generated lines of code and a comparision to the lines of code written by the customizer in events and script activities.



Framework
OpenUI5 or Angular2.

Actions
Edit Application, Copy Application, Show Releases, Edit meta resources, Preview and Delete Application.

Read how to create a new application.

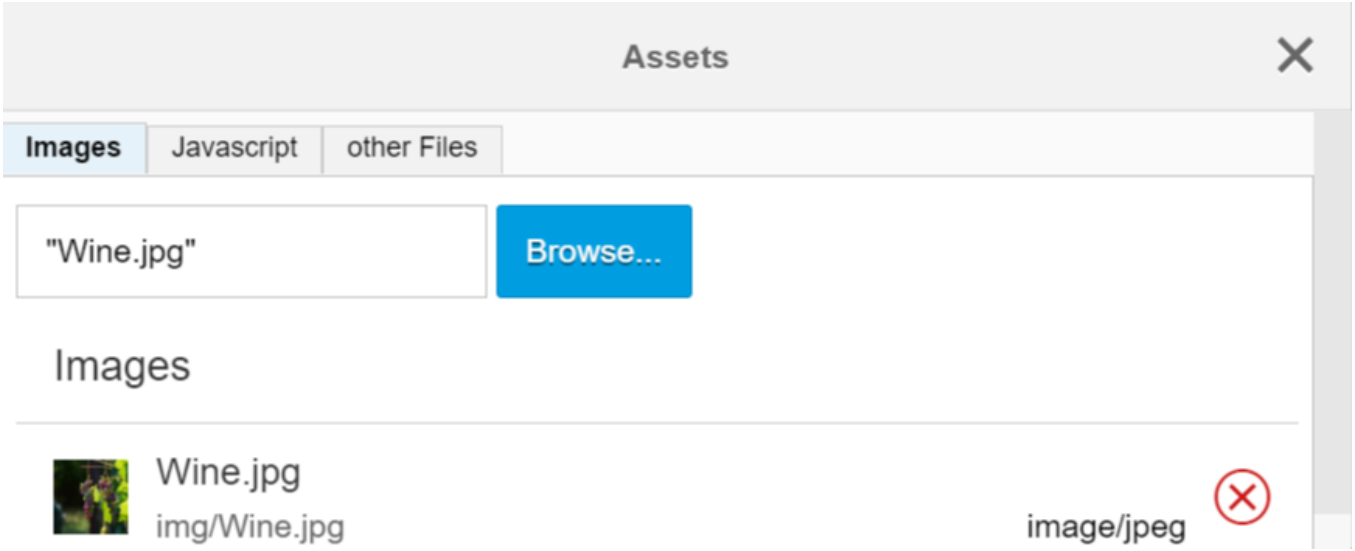
Assets

To upload files like documents, images, videos, 3D models or office documents to your application, click on the assets button.

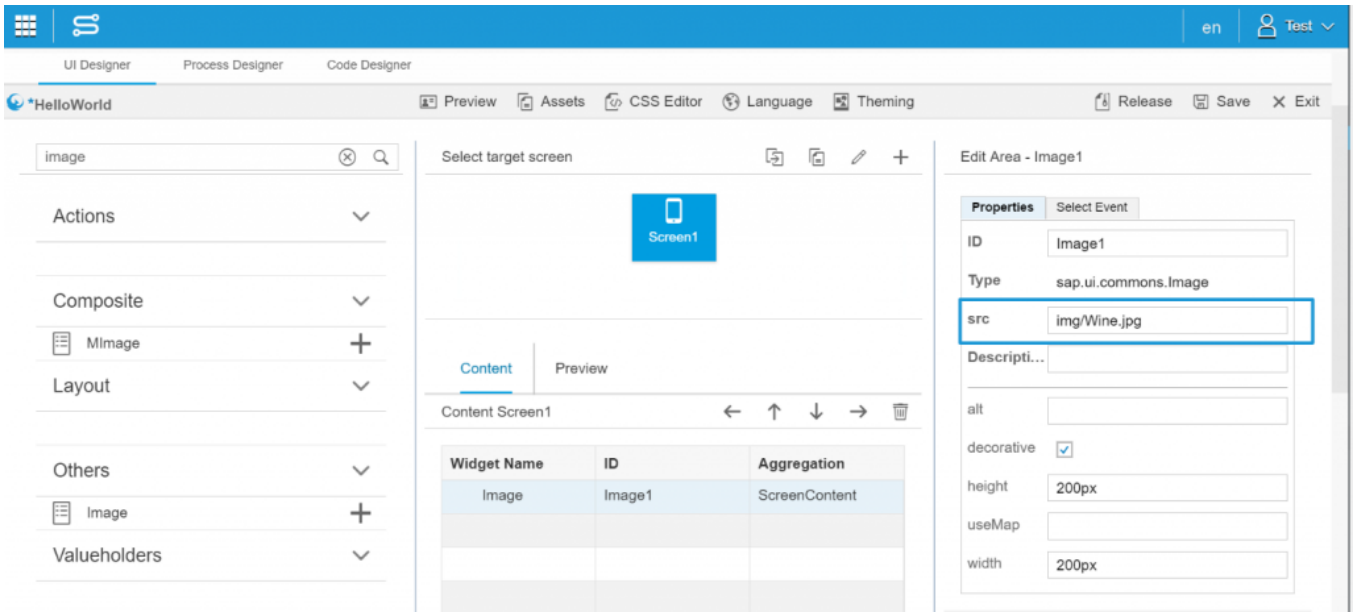
You can choose between three different options:

- Images like .PNG, .TIFF, .JPEG or .BMP Files
- Javascript for extending your application with other libraries
- Other files like .PDF Documents, 3D-Models or Media-Files (Audio, Video, etc)

To upload an image, choose it from your client via the upload button – a preview will be generated after uploading and also the path for referencing it later into an parameter of an image widget. In our screenshot the path is *img/Wine.jpg*.
By clicking on the red cross on the right side, you can delete the asset file.



To insert the assets into your user interface add, an image widget and write the path in the source field (src) in the Edit Area on the right.


en
Test
UI Designer
Process Designer
Code Designer
*HelloWorld
Preview
Assets
CSS Editor
Language
Theming
Release
Save
Exit
image
Actions
Composite
MImage
Layout
Others
Image
Valueholders
Select target screen
Screen1
Content
Preview
Content Screen1
Widget Name
ID
Aggregation
Image
Image1
ScreenContent
Edit Area - Image1
Properties
Select Event
ID
Image1
Type
sap.ui.commons.Image
src
img/Wine.jpg
Descripti...
alt
decorative
height
200px
useMap
width
200px

Assign Roles

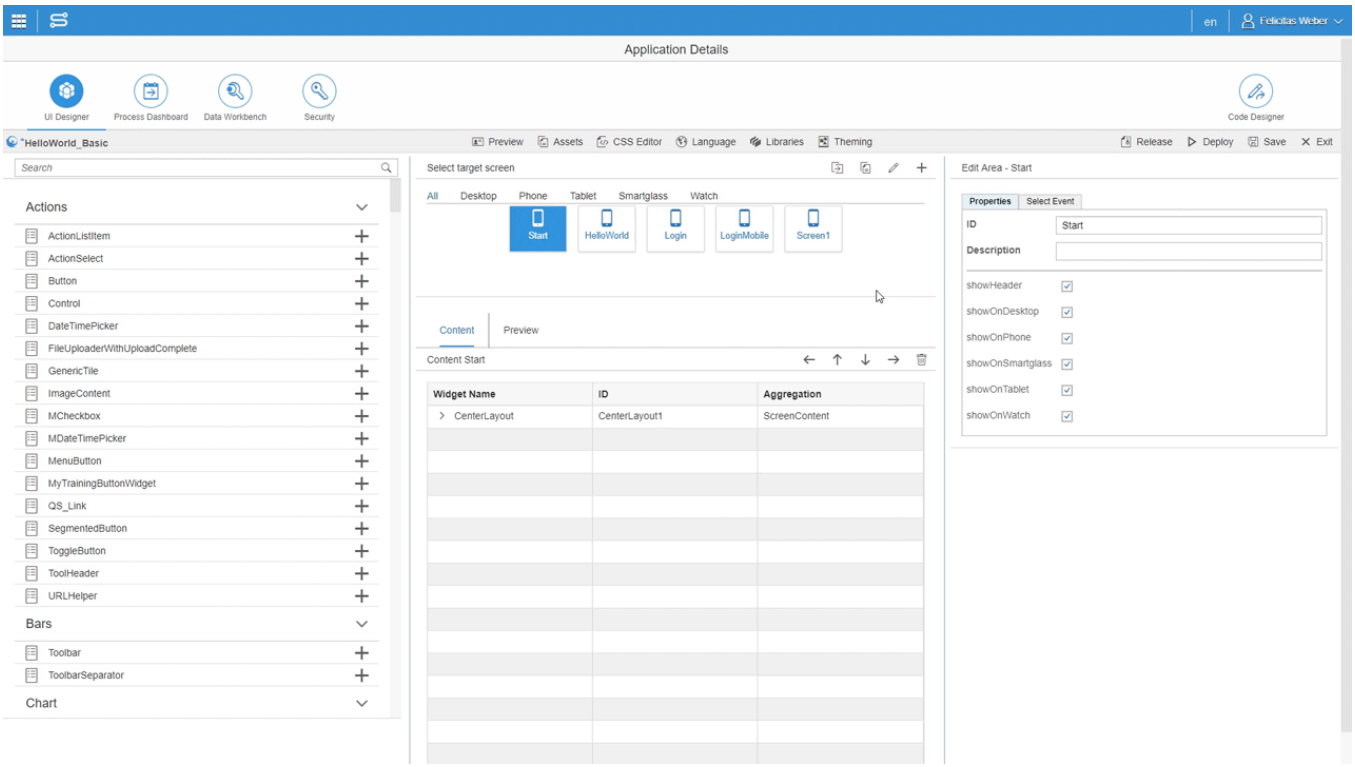
Roles control who receives the messages. Any user who has the corresponding role and uses the Simplifier client at the time of sending a message will receive the message.

Enter the required roles in JSON notation as follows:

```
{
"roles": [
"<Role-1>",
"<Role-2>",
...
"<Role-n>"
]
}
```

Assigning Libraries to Apps

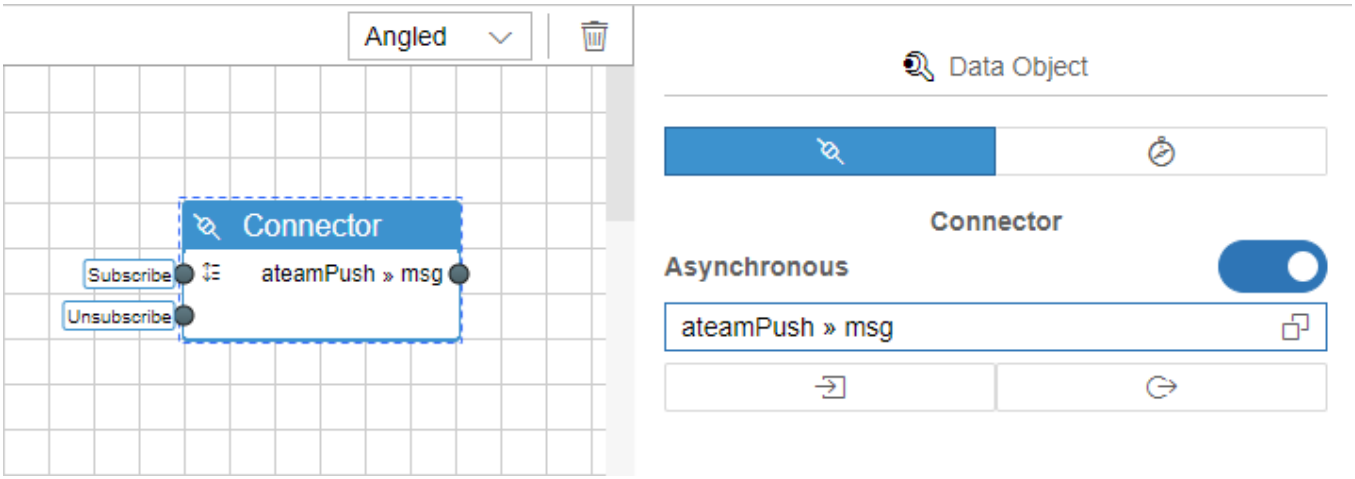
You can assign libraries to your application in the “UI Designer”.



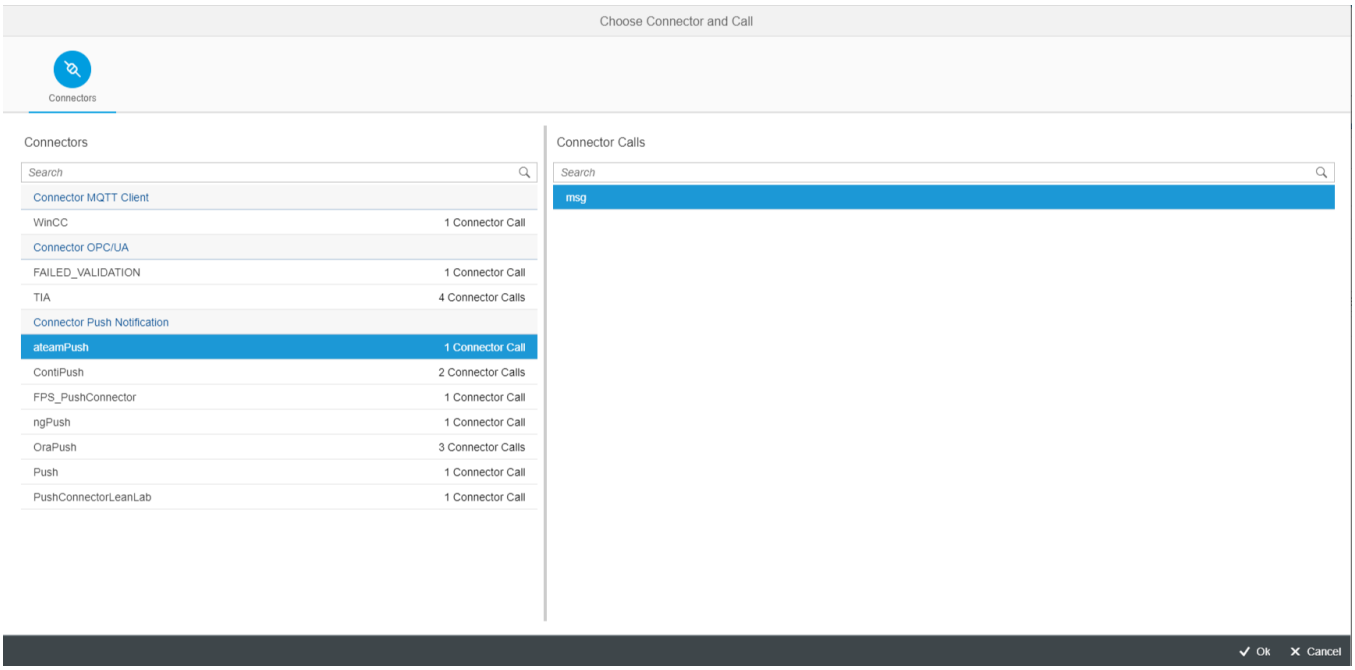
Click on “Libraries” in the upper menu bar to see an overview of all the libraries that are currently used in your app.
If you click “+” you can add a library.
The icon left-handed computes all dependend libraries (of the App itself and its Widgets as well as all transitive dependend libraries).

Asynchronous Connector

You can configure asynchronous connectors in the Process Designer. The configuration works as same as with the synchronous connector.



When setting a connector to asynchronous, the assistant only offers connectors with asynchronous interfaces (Push, MQTT and OPC/UA).



The activity has two inputs “subscribe” and “unsubscribe” and one output.

subscribe	can be triggered with any output
unsubscribe	can be triggered with any output
output	will be triggerd on any async. message the Data Object receives

Asynchronous Connector Request Json Examples

This section contains the required request data Jsons for different connectors and the description of each individual field.

The following Connectors are described with an example:

- OPC/UA Connector (Monitoring Requests)

Authentication Settings

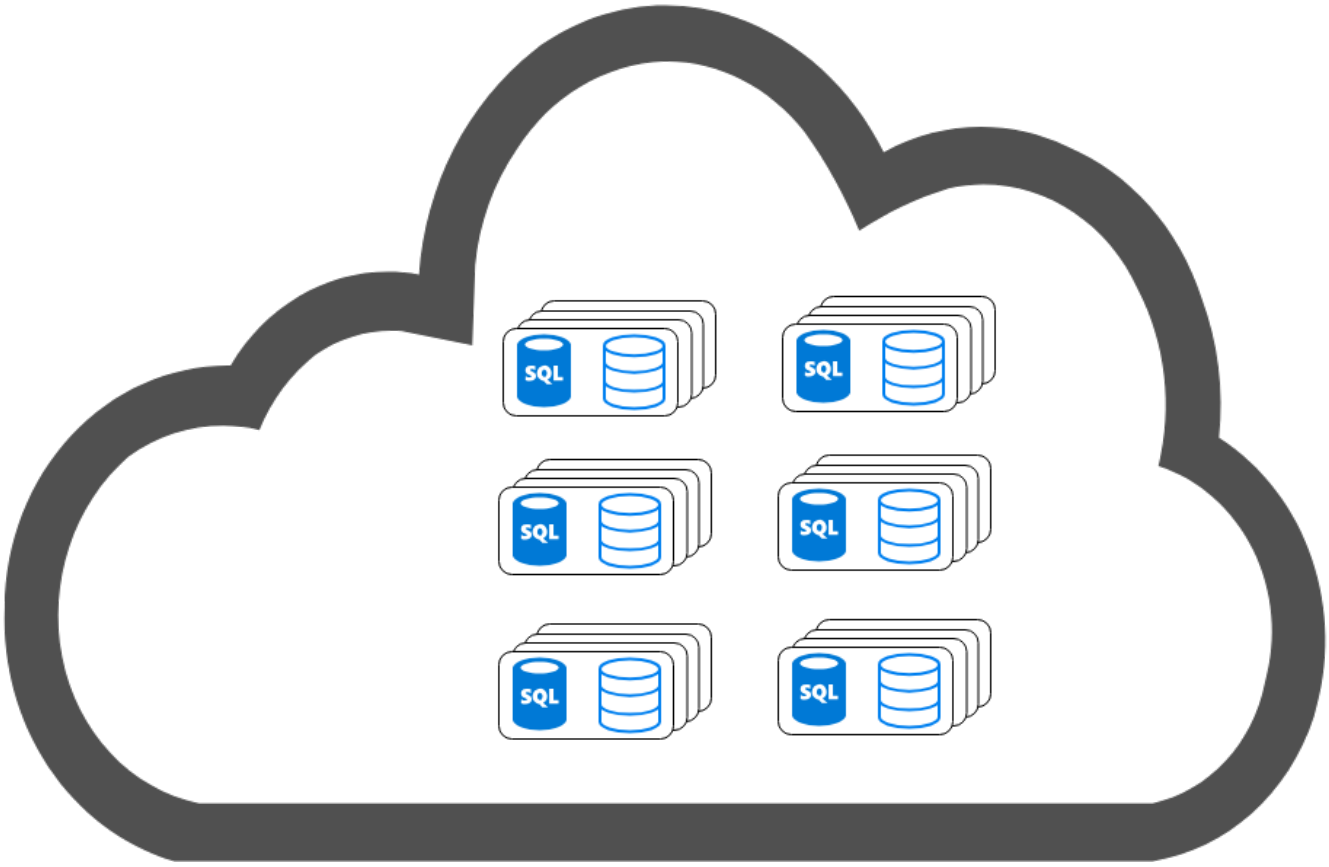
The **Authentication settings** allow you to establish a connection to a LDAP or SAP system in order to sync external user to the Simplifier.

Backups

Every instance is backed up daily in the Simplifier Cloud.

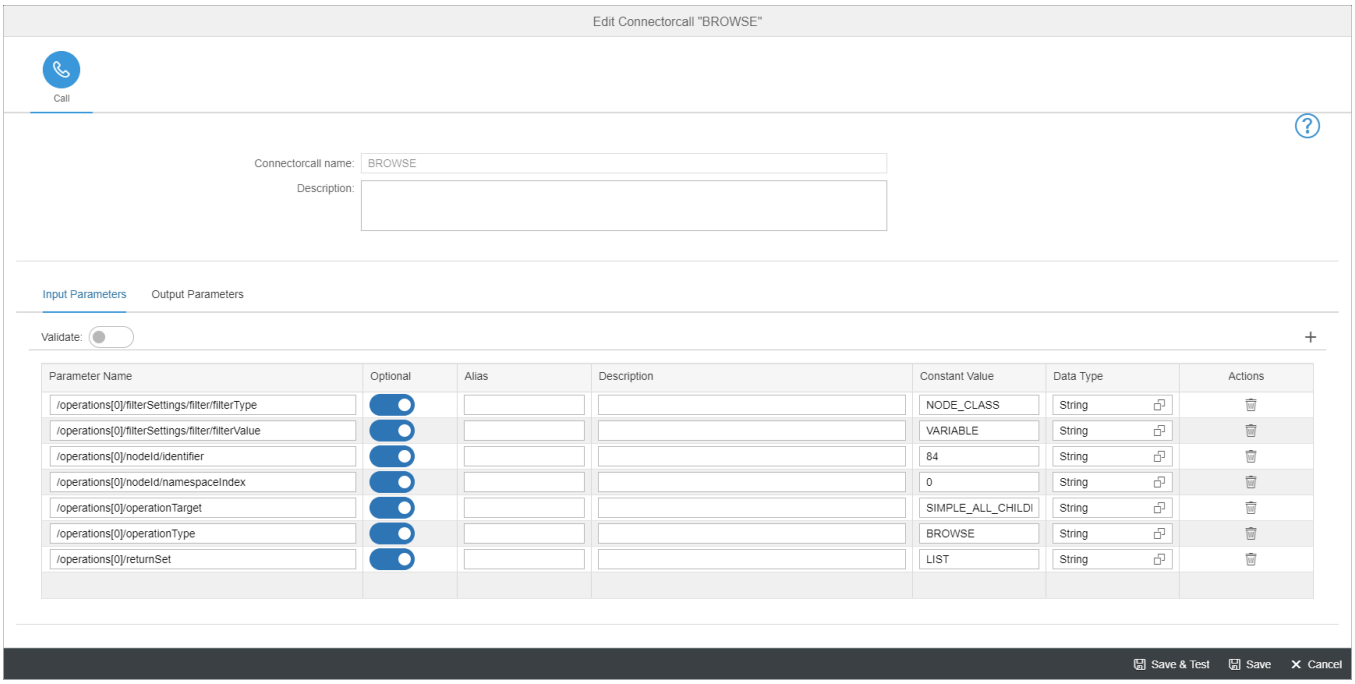
Both files and a logical database backup (dump) are stored directly on the machine. These are held locally for 4 weeks. This is very useful when restoring a single Simplifier instance.
Furthermore, the Simplifier Cloud is image-based backed up every day. These backups are held for 14 days.
Should the system fail completely, we can initiate a complete restore at any time.

Type of backup	Backup interval	How long are the backups stored?
tarball of files and a logical database backup	daily	4 weeks
image-based backup of the whole Simplifier Cloud	daily	2 weeks


SQL
SQL
SQL
SQL
SQL
SQL

BROWSE Call - OPC/UA Connector

Call for BROWSE operations (The name TIA_BROWSE_ALL_VARIABLES is the arbitrary chosen name for this call)



Input Parameter

For the Browse Connector Call, you need to configure the "**operationType** " and the "**nodeId**" (consisting of 2 parameter: **identifier** and **namespaceIndex**). Furthermore you need to define the **operationTarget**,
a **returnSet** and **filterSettings** (optional).

operationType: Defines which operation you want to execute, in this case "BROWSE".
Parameter Name: operations/arrayItem[0]/operationType
Constant Value: BROWSE
Data Type: String

nodeID: Defines the identification of the OPC/UA node. It is split in 2 Parameter:

- **Identifier**:
  Parameter Name: operations/arrayItem[0]/nodeId/identifier
  Data Type: String or Numeric
- **NamespaceIndex**:
  Parameter Name: operations/arrayItem[0]/nodeId/namespaceIndex
  Data Type: String
  In every namespace, each ID must be unique (it is possible to use the String "7617" and the Numeric 7167 together in one namespace)

operationTarget: You can browse references forward, backward or in both directions. Choose between the basic attributes (simple) or further ones, depending on the class (extended).
Parameter Name: operations/arrayItem[0]/operationTarget
Data Type: String
Constant Value: Choose between

- SIMPLE_ALL_CHILDREN
- SIMPLE_ALL_PARENTS
- SIMPLE_BOTH
- EXTENDED_ALL_CHILDREN
- EXTENDED_ALL_PARENTS
- EXTENDED_BOTH

returnSet:
Parameter Name: operations/arrayItem[0]/returnSet
Data Type: String
Constant Value: LIST

filterSettings (optional):

- **Type**:
  Parameter Name: operations/arrayItem[0]/filterSettings/filter/filterType
  Data Type: String
  Constant Value: NODE_CLASS
- **Value**:
  Parameter Name: operations/arrayItem[0]/filterSettings/filter/filterValue
  Data Type: String
  Constant Value: Choose between

    - DATA_TYPE
    - METHOD
    - OBJECT
    - OBJECT_TYPE
    - REFERENCE_TYPE
    - VARIABLE
    - VARIABLE_TYPE
    - VIEW
    - UNSPECIFIED

NOTE: The specific commands are NOT defined here!

Output parameters

You can return all Output Parameter like this:

Parameter Name: /
Data Type: String

If you want to get only selected Output Parameter, use the following syntax:

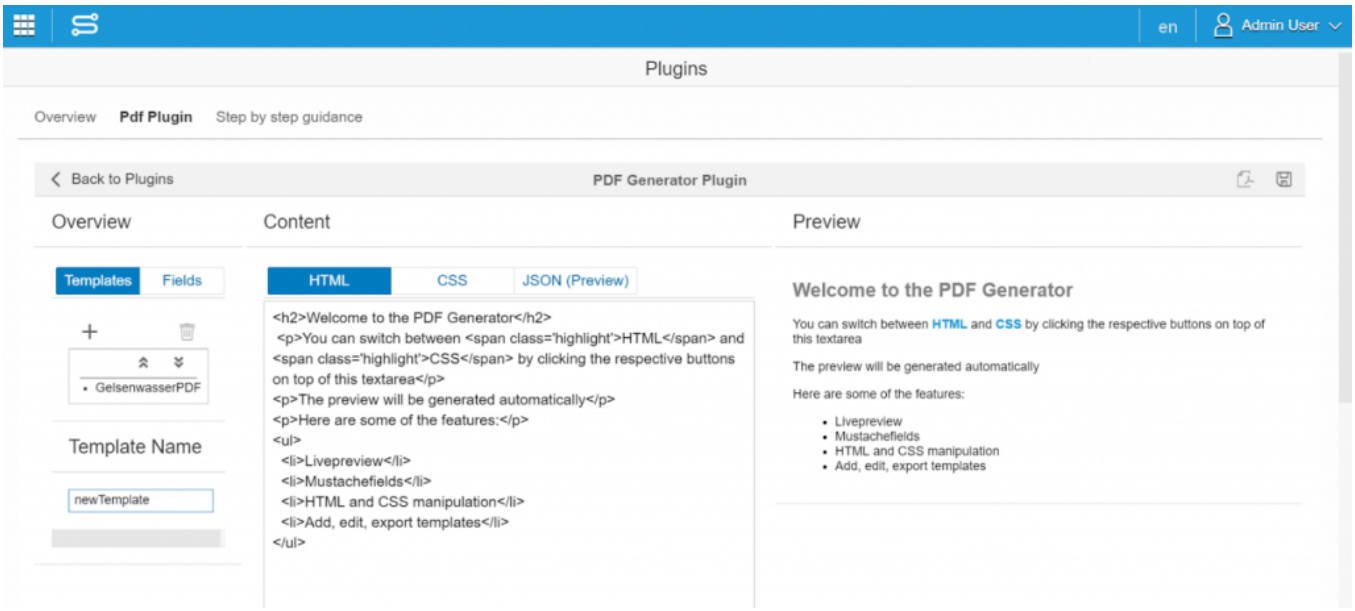
Parameter Name: operationsResult/[0]/browseResult/children/nodes/
Data Type: depends on the Parameter you want to be returned.

For now only the complete unformatted JSON will be returned.

Built a PDF Template

Administration

To generate a PDF and manage templates, the role “pdfPlugin” has to be assigned to your User.



Templates

You can build a PDF Template by using HTML, CSS and Mustache. A live preview is provided, so you can see changes in real time.

The rendering is executed with wkhtmltopdf, therfore every html format and feature that supports the QT Webkit render engine is working.

With every Template, a stylesheet in LESS format is generated and will be embedded automatically. You can maintain this stylesheet via the same interface as the HTML template.

The inclusion of graphics (**<img src=”…”>**) and additional stylesheets (**<link rel=”stylesheet” href=”…”>**) is also supported.
These external asserts are retrieved via the “assets” slot of the AppServer (they should be uploaded there in advance). You can refer to them in the template with a relative filename (no “http://” prefix, no path, etc.!).
Example: **<img src=”image.jpg”>** (if the file was uploaded as “image.jpg”).

Furthermore, you can add expressions in mustache format. These “variables” are later replaced by values from the update file to a session.

The dynamic data is retrieved as a JSON string in the key-value-store with the key: “**sessiondata/$session**“.
($session = the session ID that is specified for the generation)

Merging

You can combine your PDF document with other PDFs or images from the key-value store.

For this purpose you can call the list of all ressouces you want to merge with the key "**merge/$session**" in the key-value store. The list should correspond to a JSON-Array, where the entries of the JSON-Array are the keys of the resources to be merged. For example: **["dokument1.pdf", "document2.pdf", "bild.jpg"]**.

The binary data of the corresponding documents should be filed in the key-value store under the keys "dokument1.pdf", "document2.pdf" and "bild.jpg".

If the list of merge resources is not found for a session or if the list is empty, the merge is skipped.

Saving the generated PDF

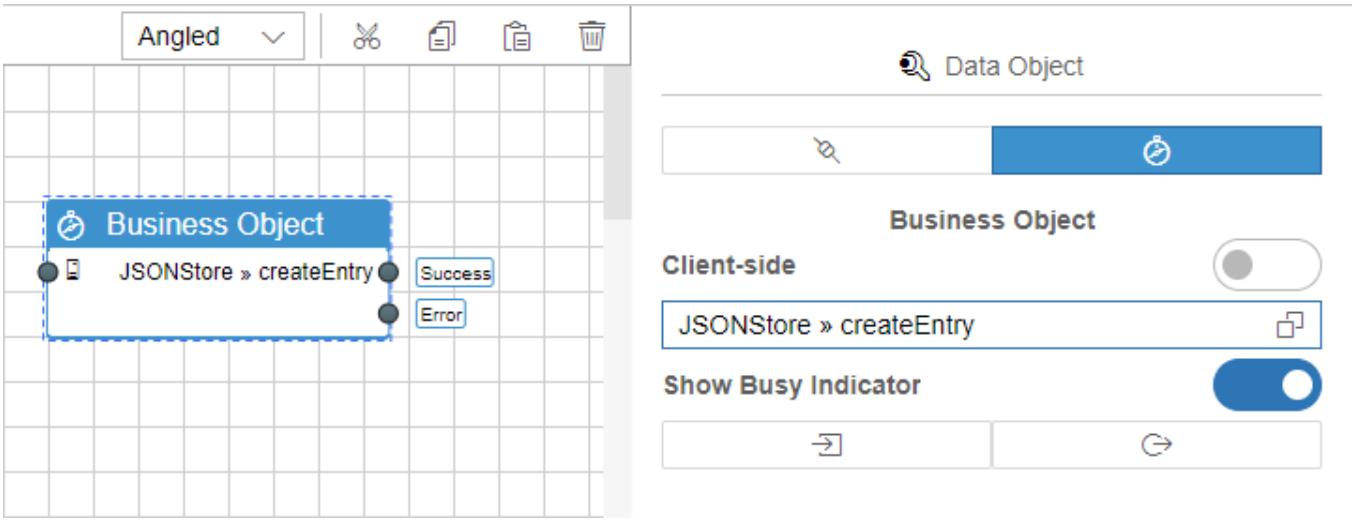
After a PDF has been successfully generated, the binary data is stored in the Key-Value Store under the key "**pdf/$jobid.pdf**". ($jobid = the job ID, that will be returned after the generation has started)

If the generation can not be executed successfully due to an error, a fault reporting is stored under the key "pdf/$jobid.log" in the key-value store.

Business Object

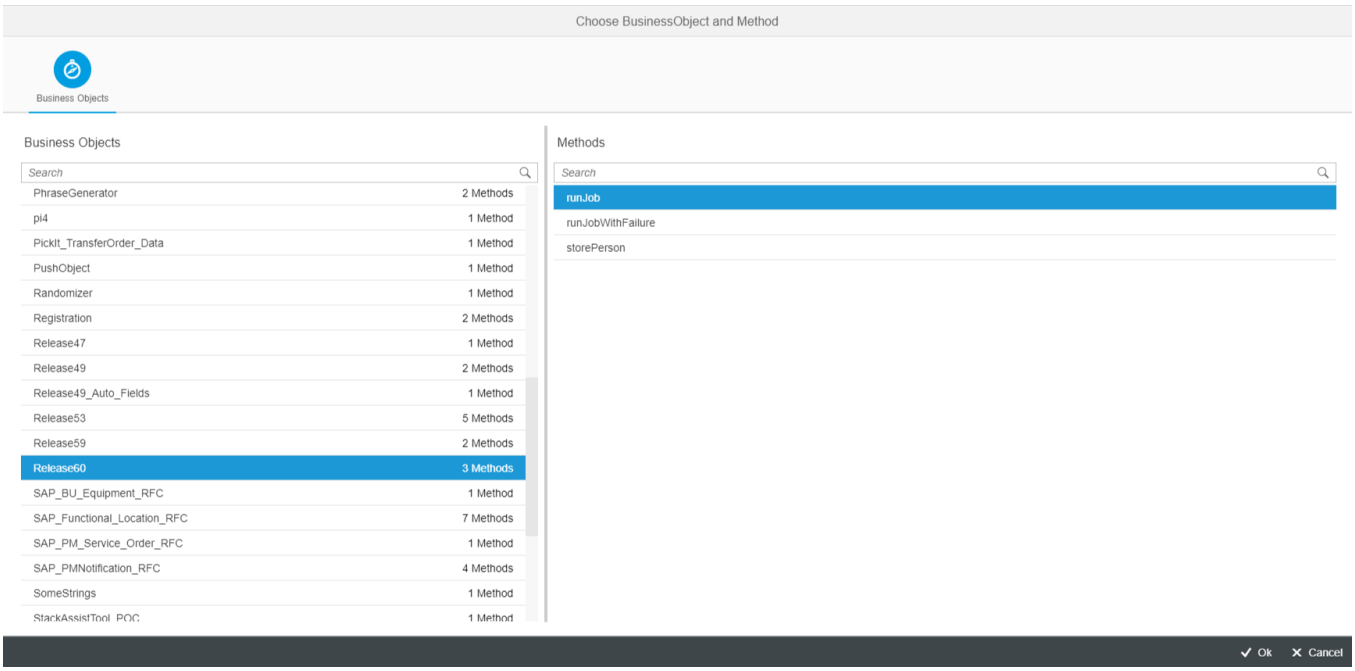
The input and output mapping of Business Objects works equivalent to the mapping of Connector Calls.

After selecting the right Business Object and the predefined script template, you can map the input and output parameters.



Function	Description
Client-side	You can choose between client-side and server-side business objects. Read more about client-side business objects.
Value Helper	By opening the value helper, an assistant opens that takes you to your Business Object.
Show Busy Indicator	You have the possibility to configure if the UI is blocked by the busy indicator, or can configure which element on your screen should be blocked by it.
Input Mapping	You can map variables, autofields, widget properties, and constants to the input parameter of your Business Object.
Output Mapping	You can map the output parameter of your Business Object to variables and widget properties.

The assistant offers all configured Business Objects that includes methods. You can search for the name of the Business Object. When selecting a Business Object, all methods are listed on the right-hand side.


Choose BusinessObject and Method
Business Objects
Business Objects
Search
PhraseGenerator 2 Methods
pi4 1 Method
PickIt_TransferOrder_Data 1 Method
PushObject 1 Method
Randomizer 1 Method
Registration 2 Methods
Release47 1 Method
Release49 2 Methods
Release49_Auto_Fields 1 Method
Release53 5 Methods
Release59 2 Methods
Release60 3 Methods
SAP_BU_Equipment_RFC 1 Method
SAP_Functional_Location_RFC 7 Methods
SAP_PM_Service_Order_RFC 1 Method
SAP_PMNotification_RFC 4 Methods
SomeStrings 1 Method
Methods
Search
runJob
runJobWithFailure
storePerson
Ok
Cancel

Business Object Script Template - Send Email

You can use this Template to send emails within Business Objects.
The Tools mapper interface:

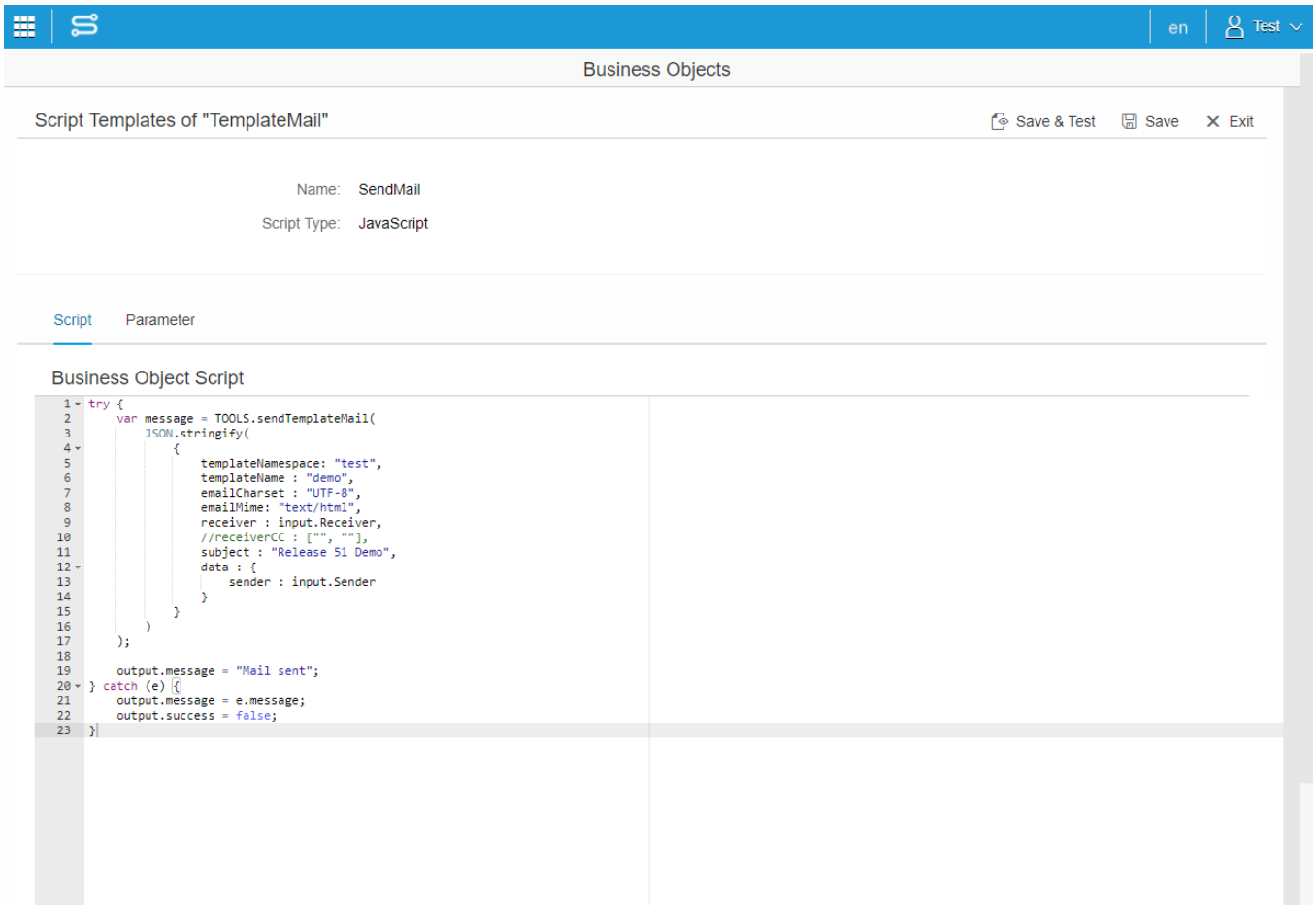
TOOLS.sendTemplateMail(json : String);

Parameter:

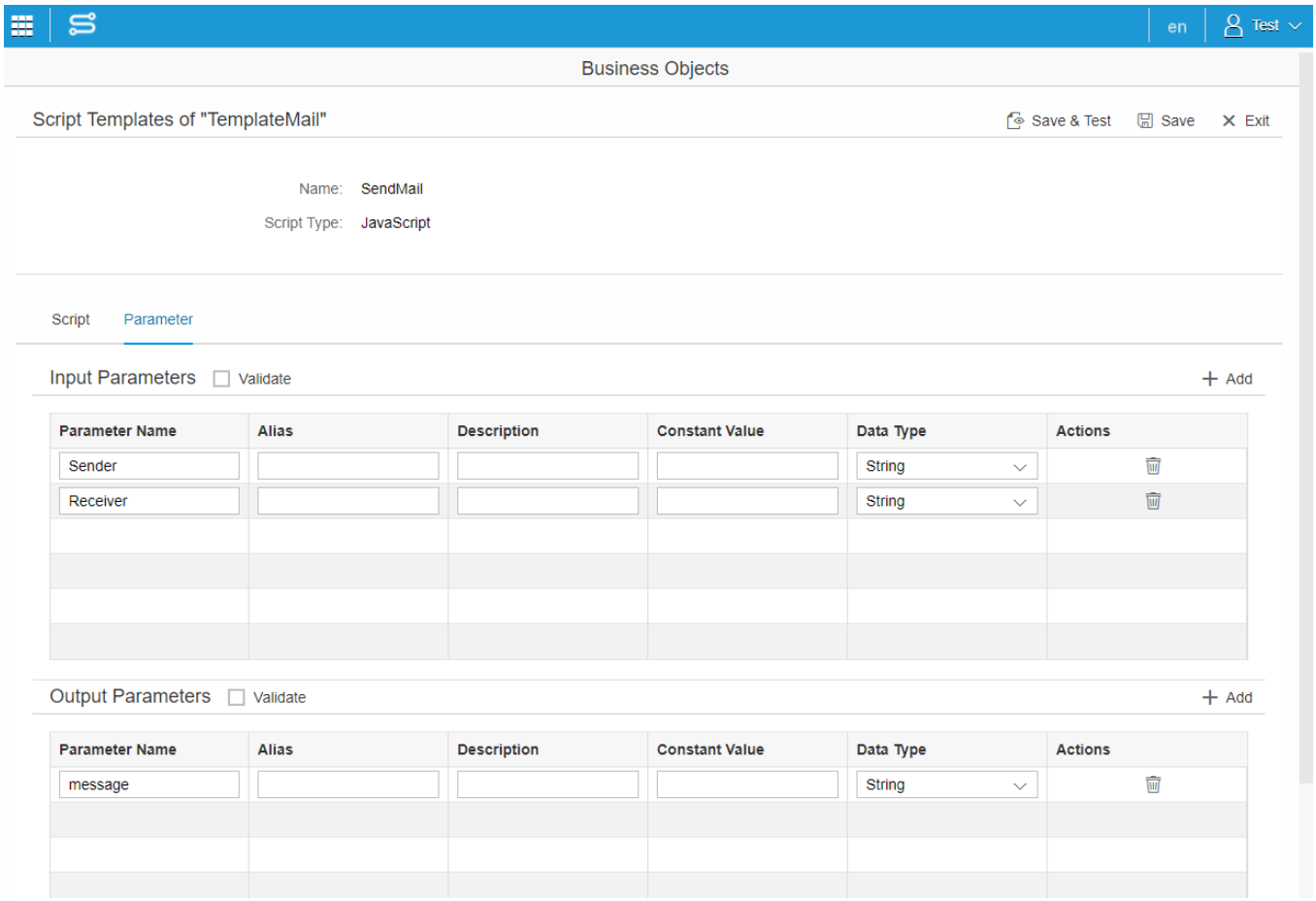
Parameter	Description	Type
templateNamespace	The namespace of the Template	String
templateName	The name of the Template	String
emailCharset	Charset of the email to send	String
receiver	The receiver email	String
receiverCC	Optional, multiple cc emails	Array[String]
subject	The subject of the email	
data	Optional, any data	JSON-Object

Example:

```
try
 {
     var message = TOOLS.sendTemplateMail(
         JSON.stringify(
             {
                 templateNamespace: "test",
                 templateName : "demo",
                 emailCharset : "UTF-8",
                 emailMime: "text/html",
                 receiver : input.Receiver,
                 //receiverCC : ["", ""],
                 subject : "Release 51 Demo",
                 data : {
                     Release : "51" // any data you want to use in the mustache templ
ate
                 }
             }
         )
     );

           output.message = "Mail sent";
 } catch (e) {
     output.message = e.message;
     output.success = false;
 }
```



Business Object Script Template



Business Object Template Parameter

Business Object via Script

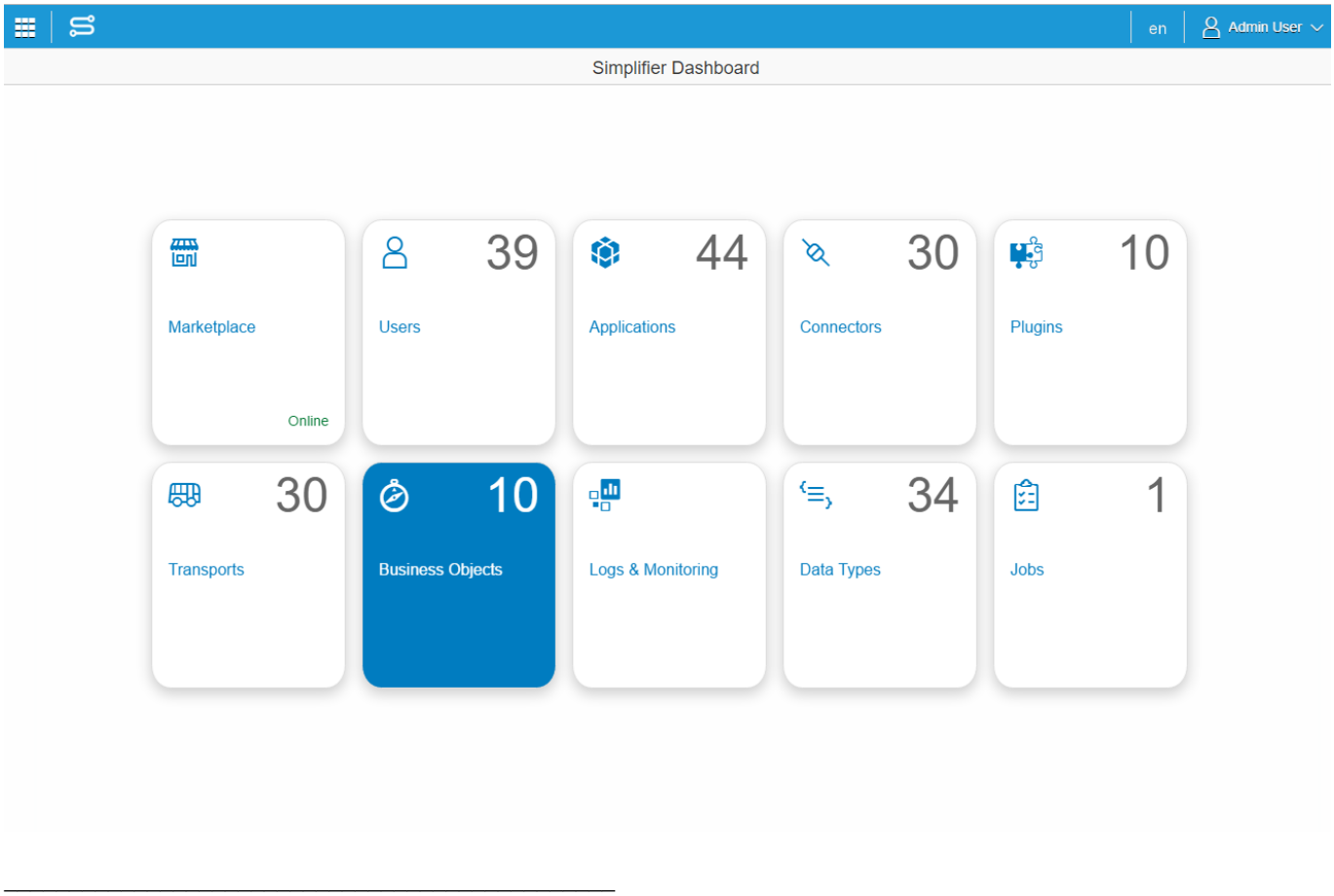
```
this.callBusinessObject(businessObjectName, method, payload, callback, showBusyIndica
tor, failOnError, failCallback, parametrized)
```

businessObjectName	the name of the Business Object
method	name of the script template to be called
payload	JSON object with parameters as required by the called script
callback	function, which is called after the successful execution of the connector
showBusyIndicator	boolean value that indicates whether the screen has to be blocked by a loading bar (true) or not (false)
failOnError	boolean value that indicates whether the connector should be called in case of an error of the function passed via “failCallback” (false) or not (true)
failCallback	function, which is called in case of an error in the connector, if false “failOnError” is passed
parametrized	boolean value that indicates whether the called parameters in the payload according to the rules in the script template are to be verified (true) or not (false)

Business Objects

The Simplifier allows you to create complex integrated applications up to a high degree solely through configuration. Nevertheless, at some point in time, advanced business logic might be required, which can't be implemented merely by configuration. This is when Business Objects come into play.

Business Objects are implemented via Javascript. This way they integrate seamlessly into Simplifier applications. They enable you to write arbitrary business logic and interact with other Simplifier artifacts like connectors, plugins or other Business Objects. They can also be shared among different applications. Although written in Javascript, Business Objects execute on the server side, keeping possibly complex and time-consuming operations away from the client.



Certificates

Our cloud infrastructure provider, T-Systems operates data networks for customers worldwide and processes data in its own data centers – data protection and data security have absolute priority. The ICT provider, therefore, has its services regularly audited by independent institutions and has itself certified that it complies with global standards and norms, for example through ISO certifications.

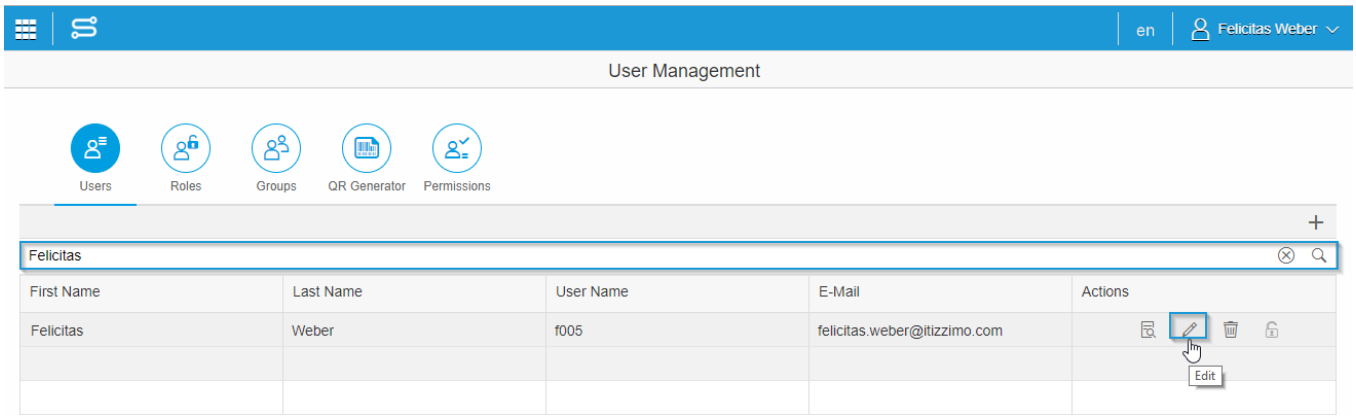
Here you will find an overview of the current certificates:

Certificate Description	Download
Quality management according to DIN EN ISO 9001	ISO 9001
Information security management according to ISO/IEC 27001	ISO/IEC 27001
Business continuity management according to ISO 22301	ISO 22301
Environmental management according to ISO 14001	ISO 14001
Occupational health and safety management according to OHSAS 18001	OHSAS 18001
Information security measures for cloud services according to ISO/IEC 27017	ISO/IEC 27017
Protection of personal data in public clouds according to ISO/IEC 27018	ISO/IEC 27018
Trusted cloud privacy profile for cloud services (TCDP)	Open Telekom Cloud
Security of cloud service providers according to CSA CLOUD CONTROL MATRIX	CSA STAR
Cloud service according to TÜV trust it requirements catalogue	Trusted Cloud Service
Zero outage as a certified service process	Zero Outage
Compliance management systems (CMS) according to IDW PS 980	auditor’s certificate

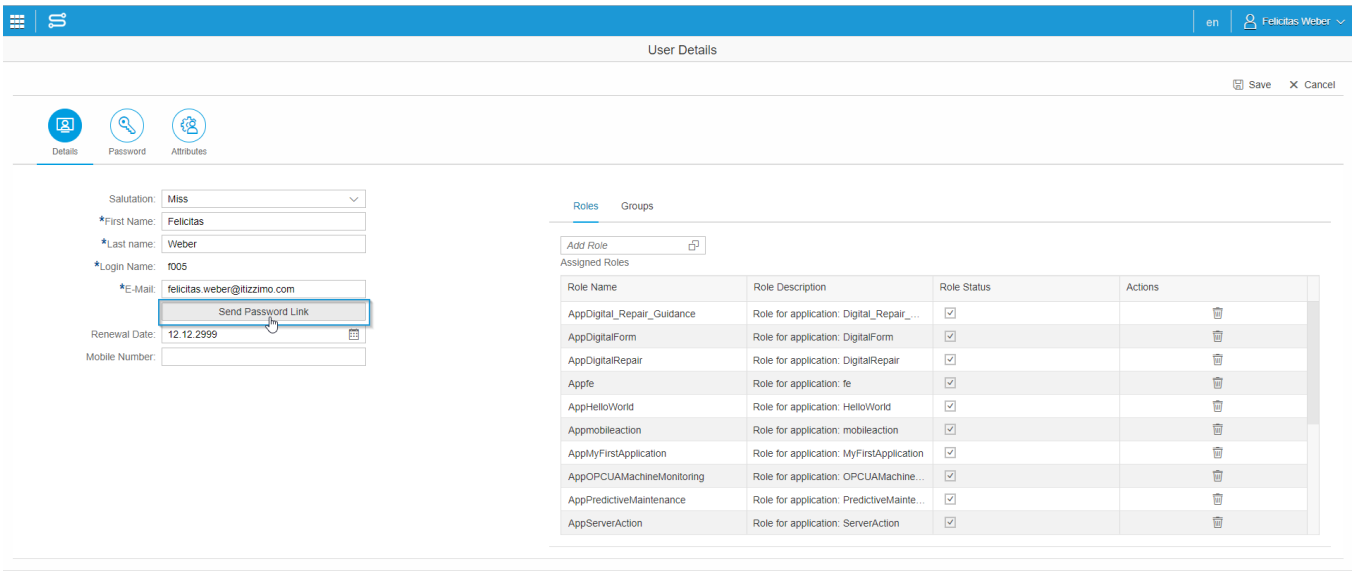
Change your Password

For security reasons, it is always a good idea to update your password regularly.

In order to change your password, you have to switch to the ‘**Users**‘ tile in the Simplifier dashboard. After that, search for the user whose password you want to change and click on the **edit** icon underneath ‚Actions‘.

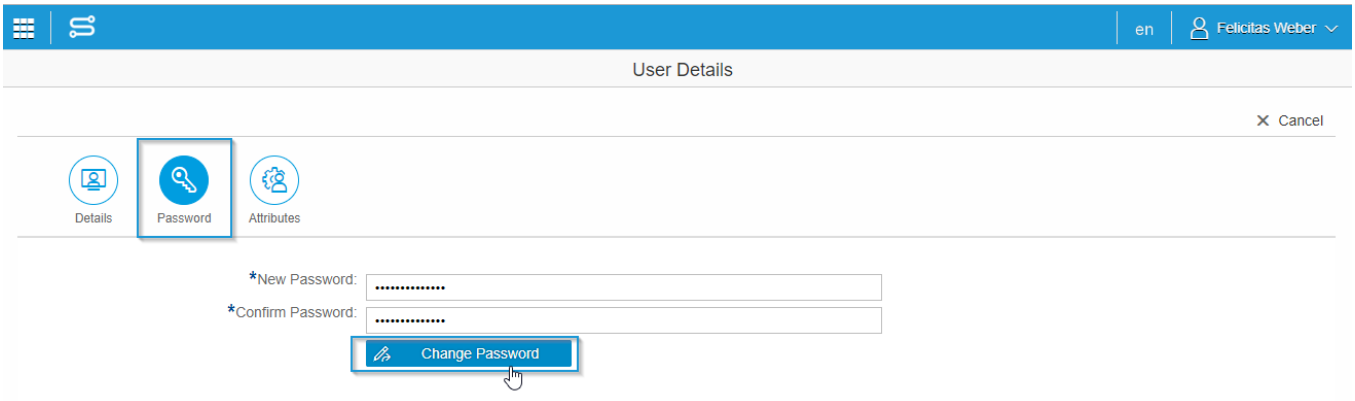


You will be lead to the user details of the selected user. Now click on the ‘**Send Password Link**‘. You will receive an email with a link to change your password.



If you need help, please contact an admin.

If you’re an admin and want to change someone’s password, click on the ‚**Password**‘ tab in the upper left corner. Now all you have to do is enter the new password, confirm it and finally click on ‚**Change Password**‘.

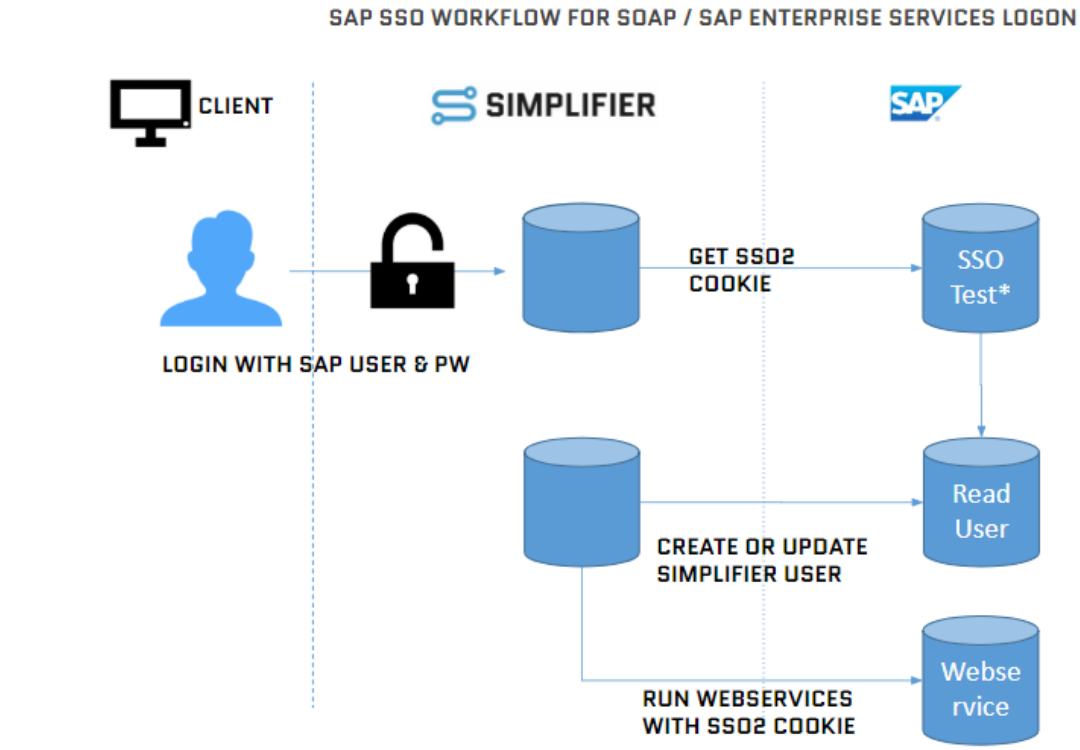


Checklist - Simplifier onPremise Installation

Here you will find a checklist for all On Premise installations. You can check off the points when you've finished them.
During this time, please do not reload this page.

Have a FQDN (Fully-Qualified Domain Name) for each instance of the D (Development) Q (QA /Test System) P (Productive) System

Checklist SAP SSO over SOAP



Check 1: SSO2 Check

1. Start Transaction SE80
2. Choose Type BSP Application
3. Choose SYSTEM_TEST/test_sso2.htm
4. Test/Run (F8)
   http://hostname.example.com:8000/sap(bD1kZSZjPTgwMA==)/bc/bsp/sap/system_test/test_sso2.htm
5. Check if Cookie ‚MYSAPSSO2=…' available

Check 2: SSO Parameter

1. Run transaction code RZ11(temporary) RZ10 (permanent)
2. Check if the following parameter has been set

login/accept_sso2_ticket	1
login/create_sso2_ticket	2 (without certificate)
icm/host_name_full	FQDN (e.g. hostname.example.com)

Check 3: SSO Login

1. Open transaction SA38
2. Choose report SEC_TRACE_ANALYZER

Check 4: Permissions

Every user needs the following permission object:

S_SERVICE

Attributes	**Values**
SRV_NAME	Name of Webservice
SRV_TYPE	Type of Webservice (HS)

Troubleshooting / Common Errors & Solutions

The following section documents the most common errors with possible solutions.

Q: What should I do when HTTPS/SSL is not available?

A: If you have problems with the connection set it from SSL to None

Q: What if the WSDL Consumer has problems parsing the WSDL?

A: Manually replace the string ws_policy in the WSDL with standard

Q: How can I monitor the error log of SAP Web services?

A: The error log can be viewed with transaction “srt_util”.

Q: How can I change the SAP web service login language?

A: The standard login language is also via SAP Webservices in English. Thus, all data determinations according to e.g.: Status texts, material text ect. always return in English language.

To be able to change it to German, the following prefix must be appended to the **SOAP Webservice operation URL**: “?sap-language=DE”
This does NOT mean the WSDL URL!

Q: How can I call the web service from another SAP client?

A: The **web service operation call** must be done with the parameter?sap-client=[client] so that the system can recognize the client.

Q: What if the Simplifier does not have access to the SAP system?

A: Check the following points:

Please make sure that there is a physical connection between the Simplifer (host) instance and the system. Firewall/Ports may need to be enabled to allow communication in both directions.

The Simplifier Docker or host system must be maintained with the correct network settings for on premise installations. This includes, for example, the setting for DNS servers.

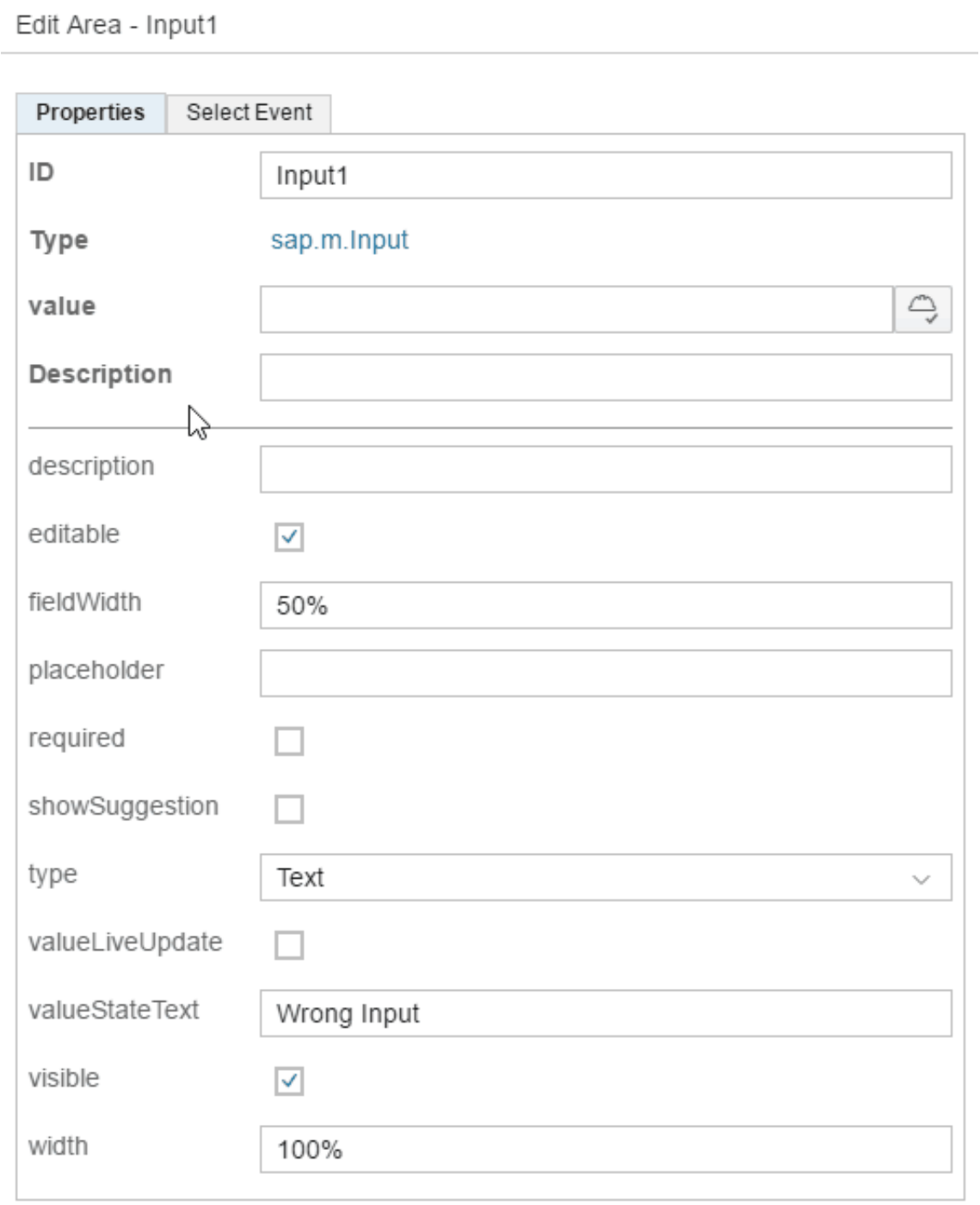
Client Validation

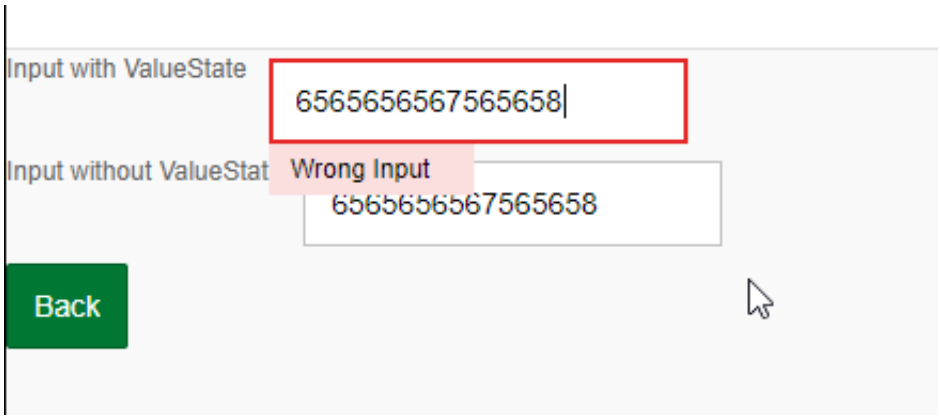
The Data Type of default properties in application items can now be overwritten, but only domain types with the same basic type can be used.

You can find the settings in the Property Panel of the UI Designer. The button will open a popover, which lets you define the Data Type and the validation event. If the validation is enabled (checkbox), an indicator will be displayed. You can change the valueState and valueStateText as result.

In this example, we wanted to make sure, that the Input field will be filled with the correct data, a ZIP Code. Therefore we added the validation for the predefined Datatype “ZIP” (a String with exactly 5 numbers) on the change Event.

Now, if you open the application in the preview and type in anything else than 5 numbers, the valueState of the Input field would change to “ERROR” and the valueState text “Wrong Input” would be displayed.


Edit Area - Input1
Properties
Select Event
ID
Input1
Type
sap.m.Input
value
Description
description
editable
fieldWidth
50%
placeholder
required
showSuggestion
type
Text
valueLiveUpdate
valueStateText
Wrong Input
visible
width
100%


Input with ValueState
6565656567565658
Input without ValueStat
Wrong Input
6565656567565658
Back

Client-Side Business Object API

You can access any methods of the Simplifier by using the Simplifier Object.

Connectors

```
Simplifier.Connector.<ConnectorName>(payload: object, successCallback: function, busy
Flag?: boolean, failOnError?: boolean, errorCallback?: function): void
Simplifier.Connector.<ConnectorName>.<CallName>(payload: object, successCallback: fun
ction, busyFlag?: boolean, failOnError?: boolean, errorCallback?: function): void
```

Example:

```
var payload = {bindingName: "Binding", operationName: "MyOp", soap: {foo: "bar"}};
function onSuccess (data) { resolve(data); };
Simplifier.Connector.MySoap(payload, onSuccess, true, true);
Simplifier.Connector.MySoap.myCall(payload, onSuccess, true, false, function () { con
sole.log("something went wrong"); });
```

Business Objects

```
 Simplifier.BusinessObject.<BOName>.<MethodName>(payload: object, successCallback: fu
nction, busyFlag?: boolean, failOnError?: boolean, errorCallback?: function, parametr
ized?: boolean = true): void
```

Example:

```
var payload = {leftOperand: 3, "operation": "add", rightOperand: 4};
function onSuccess (data) { resolve(data); };
Simplifier.BusinessObject.OtherBO.someMethod(payload, onSuccess, true, false, functio
n () { console.log("something went wrong"); }, true);
```

Client-side Business Objects

```
Simplifier.ClientsideBusinessObject.<CSBOName>.<FunctionName>(payload: object, succes
sCallback: function, busyFlag?: boolean, failOnError?: boolean, errorCallback?: funct
```

```
ion): void
Simplifier.CurrentClientsideBusinessObject.<FunctionName>(payload: object, successCal
lback: function, busyFlag?: boolean, failOnError?: boolean, errorCallback?: function)
: void
```

Example:

```
var payload = {leftOperand: 3, "operation": "add", rightOperand: 4};
function onSuccess (data) { resolve(data); };
Simplifier.ClientsideBusinessObject.OtherBO.someMethod(payload, onSuccess, true, fals
e, function () { console.log("something went wrong"); });
Simplifier.CurrentClientsideBusinessObject.someMethod(null, onSuccess, true, false, f
unction () { console.log("something went wrong"); });
```

Plugins

```
Simplifier.Plugin.<PluginName>.<SlotName>(payload: object, successCallback: function,
 busyFlag?: boolean, failOnError?: boolean, errorCallback?: function): void
```

Example:

```
var payload = {name: ""};
function onSuccess (data) { resolve(data); };
Simplifier.Plugin.contentRepoPlugin.listRepos(null, onSuccess);
Simplifier.Plugin.contentRepoPlugin.createRepo(payload, onSuccess);
```

CryptoJS

```
var sMySecretKey = "secret";
var oCrypted = CryptoJS.AES.encrypt("dontStealMyData", sMySecretKey);
output.result = CryptoJS.AES.decrypt(oCrypted, sMySecretKey).toString(CryptoJS.enc.Ut
f8)
```

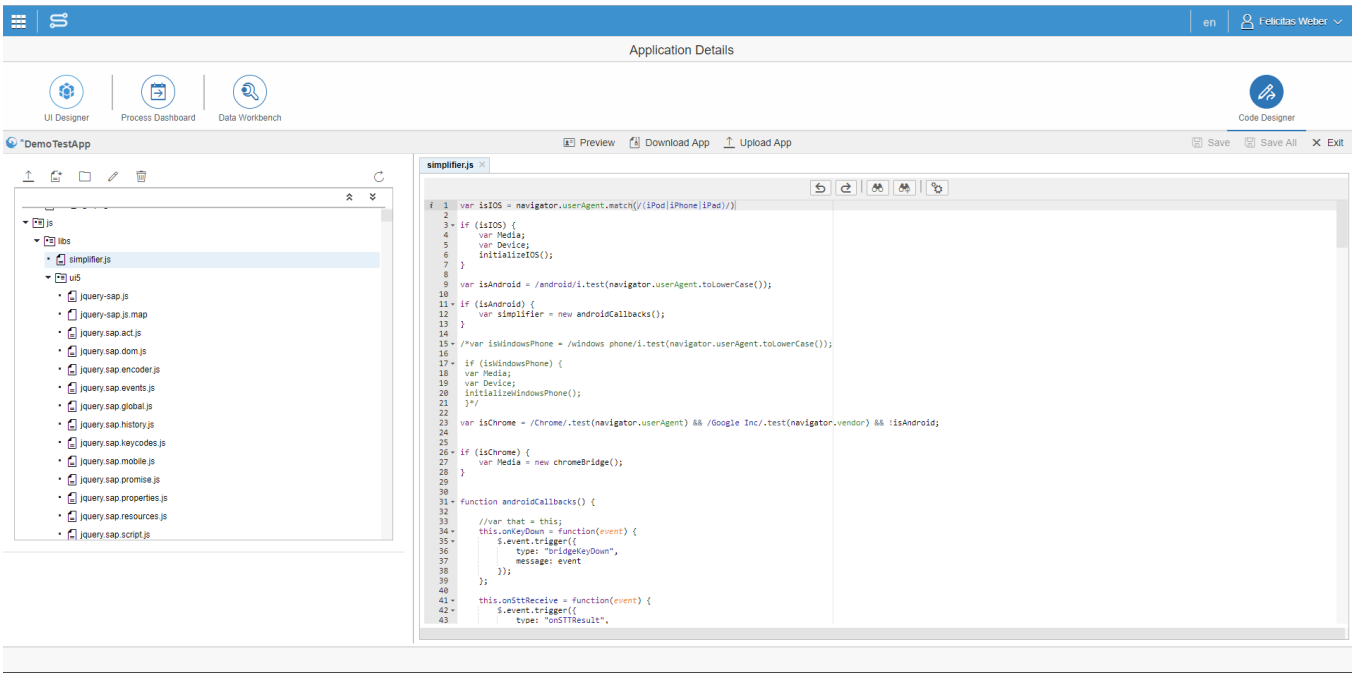
Take also a look at crypto-js.

Code Designer

The Code Designer is a web based development environment that can be used to write source code like in a normal IDE. Entered Code will be highlighted according to the programming language used (e.g. Javascript, HTML or CSS).

On the left side there is a Document tree view where all the files of an application are displayed in a structured hierarchy. New files can be created, uploaded or moved within the tree view.

All opened files are displayed in tabs and can thus be edited easily by switching tabs. The search function is used to search through code by keyword. Also change can be undone or redone with the Undo / Redo Buttons.

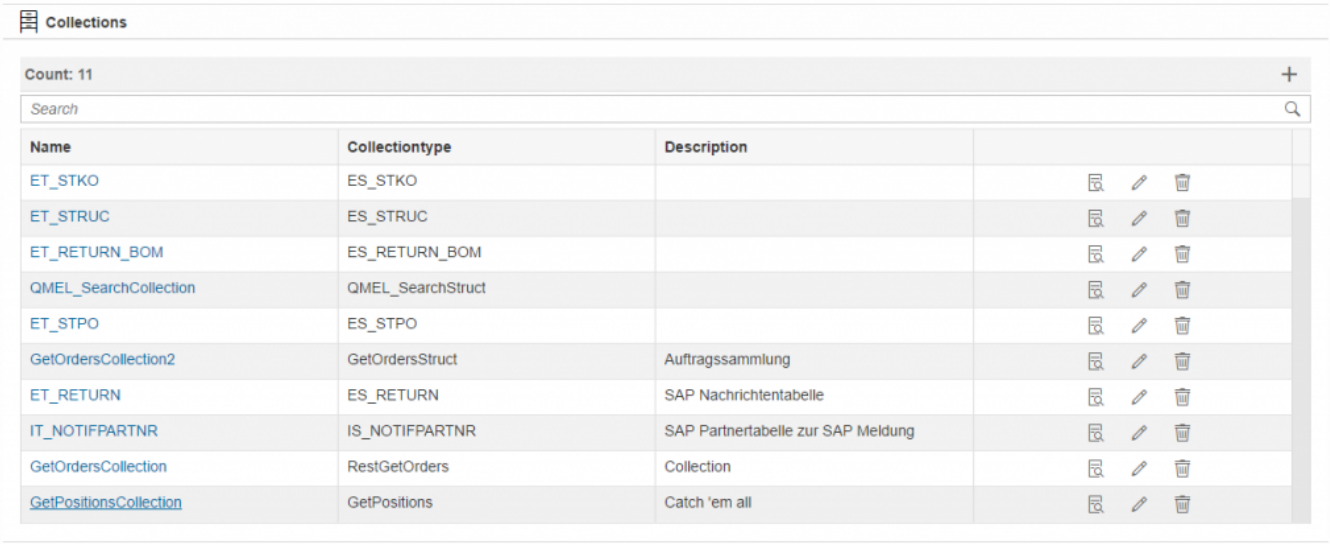


If you add or change code here, it can not be displayed in the edit mode of a user story (Process Designer). The connection is only unidirectional. Click on the “Download App” button to get the code file.

Collection Type

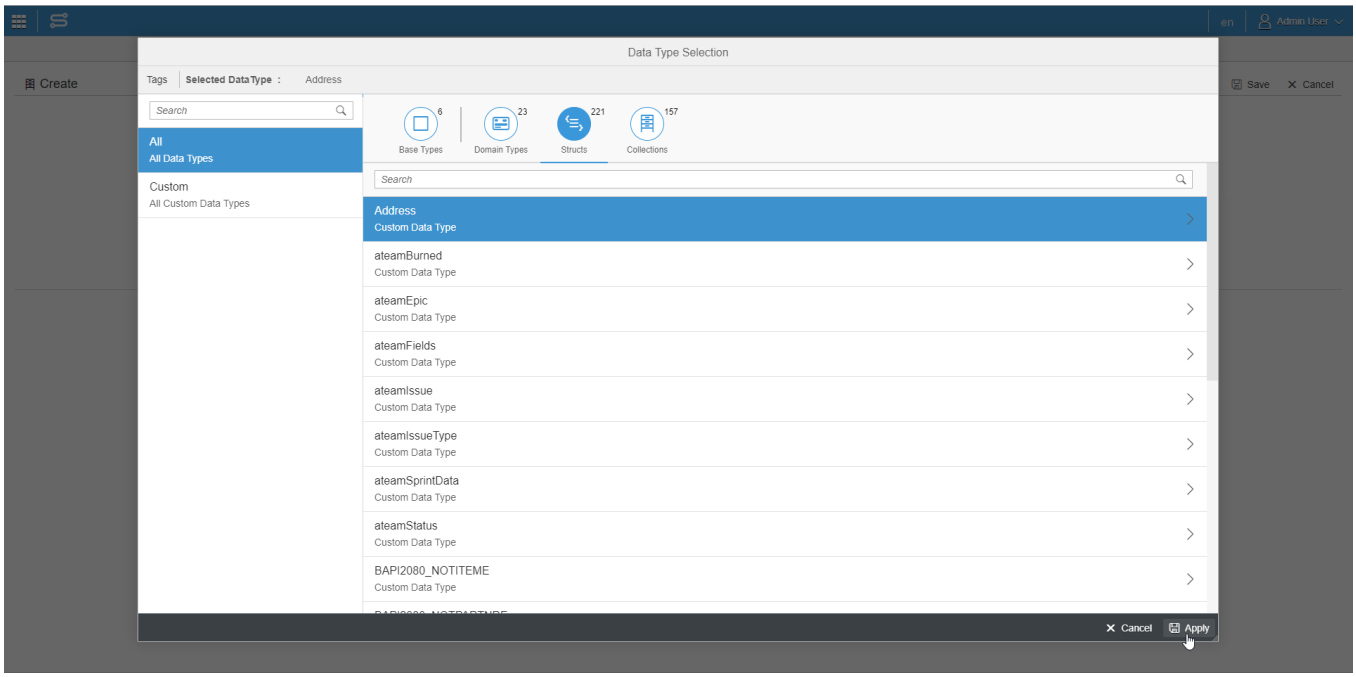
Collections represents multiple results of Structs. For example a databank request may deliver a list of addresses from numerous people.

To create a new Collection Type click on the “+” button.

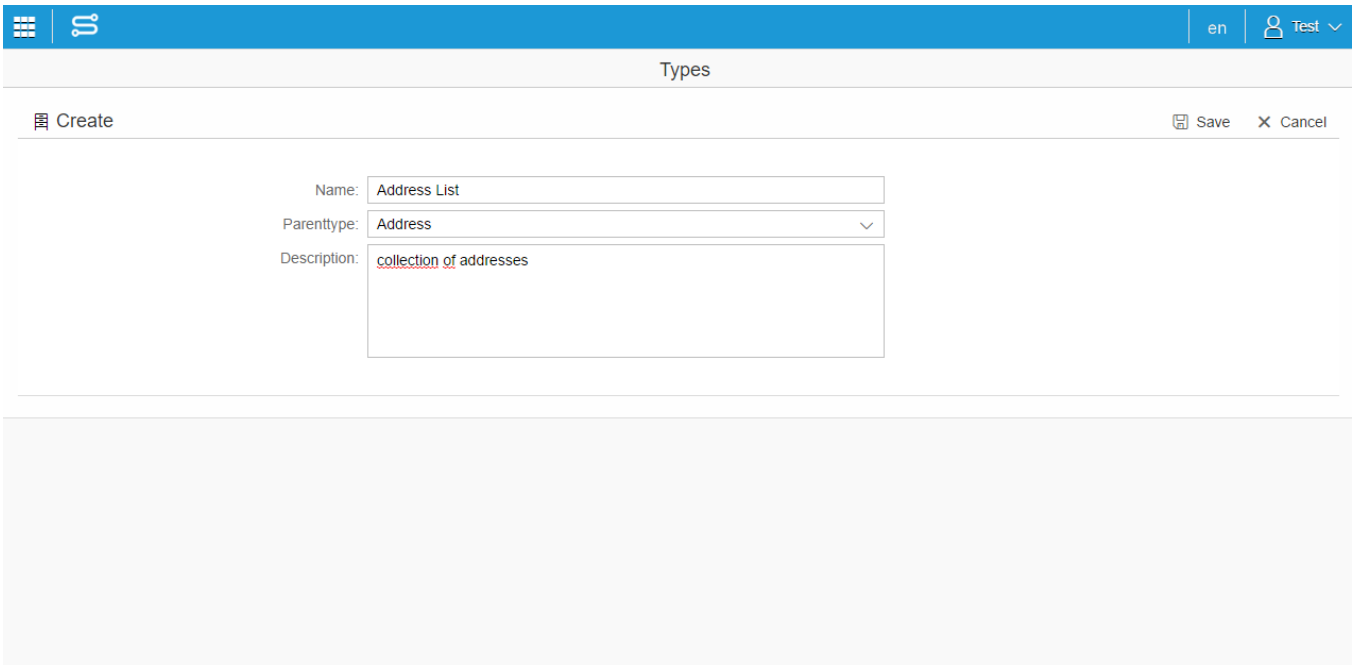


You can define a unique collection name and a description.

By clicking on “Parenttype” a new pop up opens, where you can choose the parenttype from.
NOTE: A Collection can only reference to a Struct or single Domain Types.



After you have clicked on apply, the parenttype will be used.


en
Test
Types
Create
Save
Cancel
Name:
Address List
Parenttype:
Address
Description:
collection of addresses

Components of Applications

A Simplifier application usually contains the following configurable components:

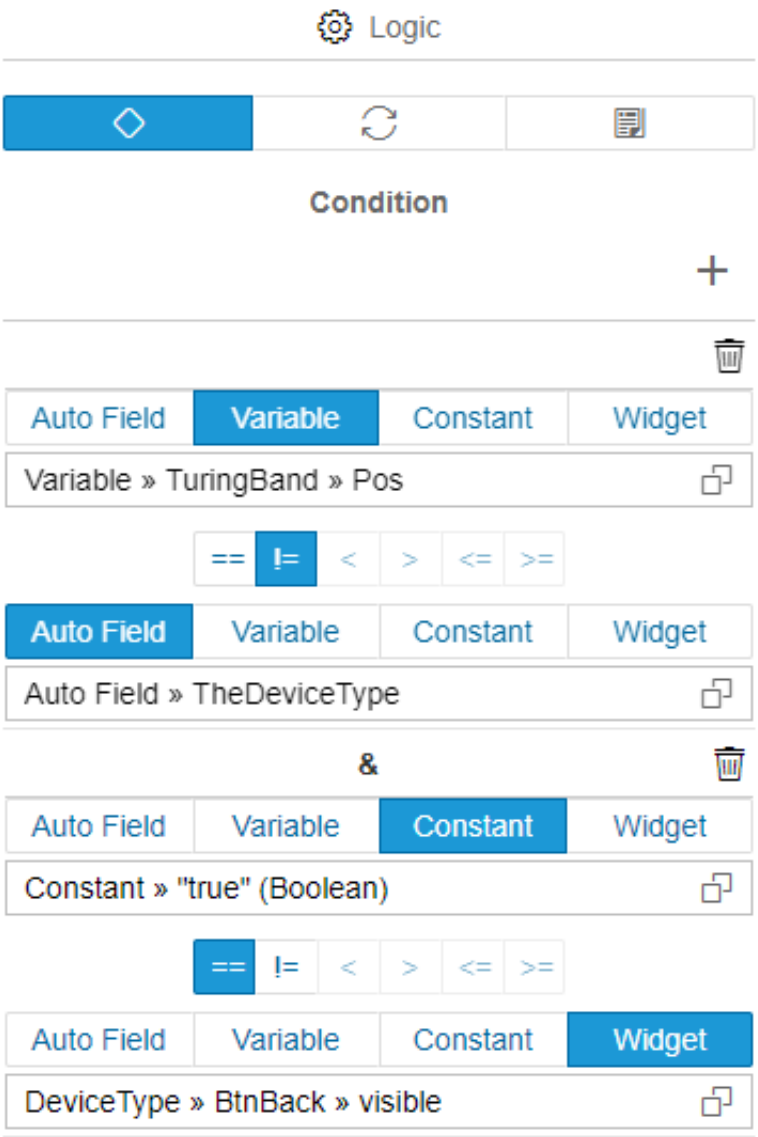
Component	Part of Application	Description
Widget	User Interface	A widget represents a specific element in the User Interface like a checkbox or login screen.
Screens	User Interface	A screen collects several widgets in a specific order to represent a certain process step or task.
CSS Editor	User Interface	The CSS editor can be used to create a unique corporate design for your business application.
Assets	User Interface	Assets can be used to add static content to your apps like images, videos or documents.
Process Dashboard	Process Logic	Visual Workflow Builder to define the process workflow and business logic.
Code Designer	Code	Add specific program code in multi-user WebIDE.

Condition

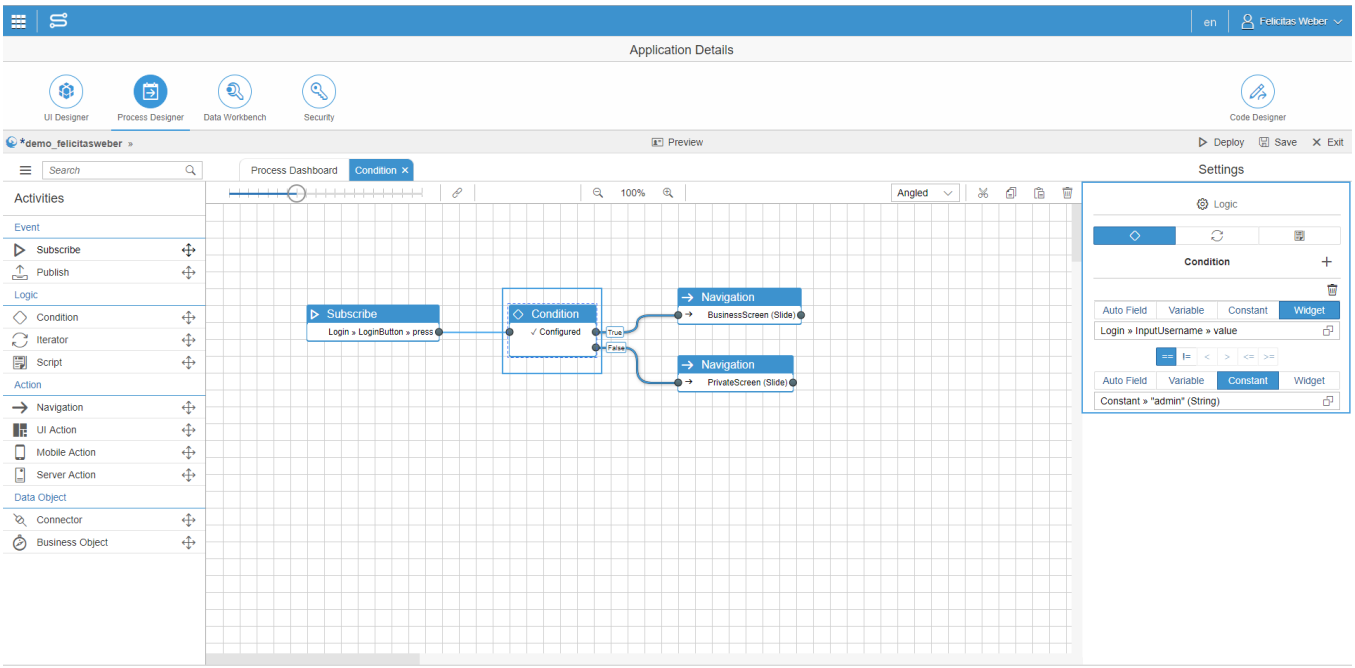
Conditions give you the possibility to make the process flow dependent on the evaluation of certain expressions.

In such an expression, you can compare Autofields, Variables, an actual value of a widget or a given constant with each other and decide upon the result, if the process continues one way or the other. Or you might decide to continue the process only, if one condition holds and omit the alternative, which in turn results in a process stop at this stage.

If you double click on the shape, it will add a new condition. Alternatively you can click on the plus icon on the right.



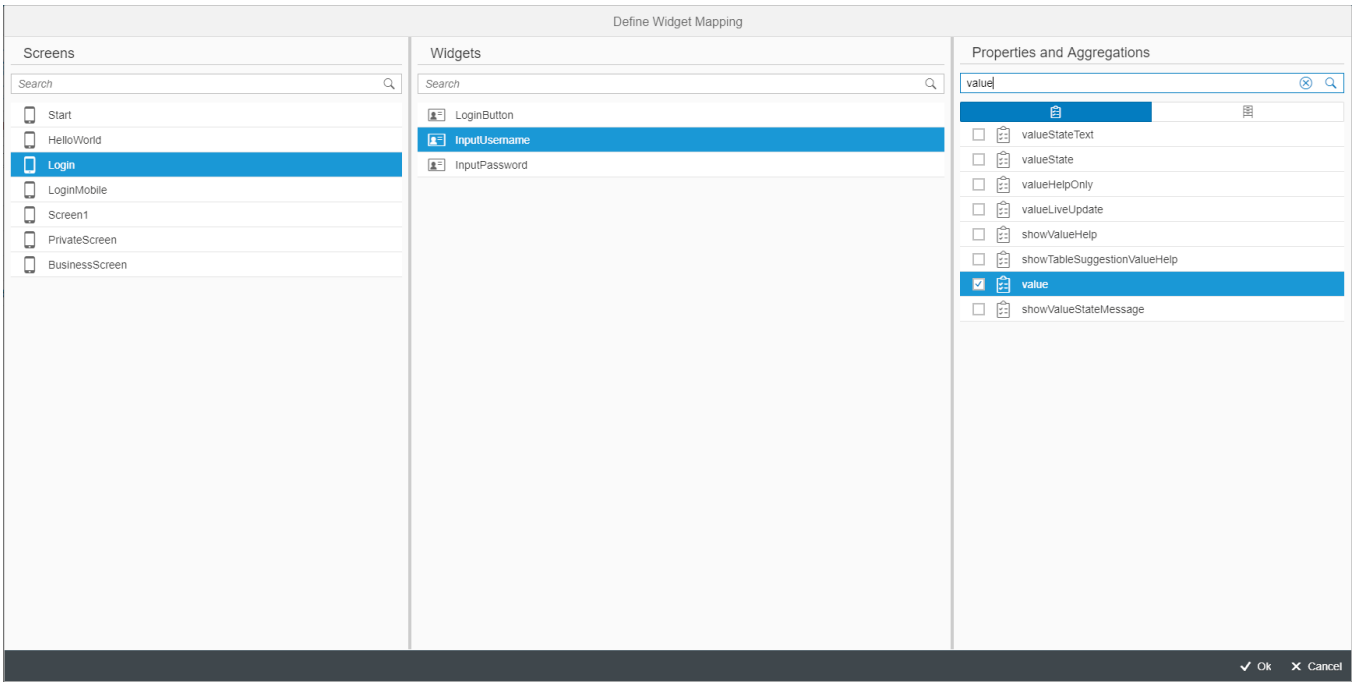
Let's see an example:



The process pane contains a condition, that’s executed after the login button got pressed. The condition compares a widget value (value property of widget InputUsername) with the fixed constant value “admin”. This means, that the expression evaluates to true, if the input value of field username equals “admin”. In this case, the process flow leads to a navigation to screen “BusinessScreen”. If the username does not equal “admin”, the user will be navigated to the Screen “PrivateScreen”.

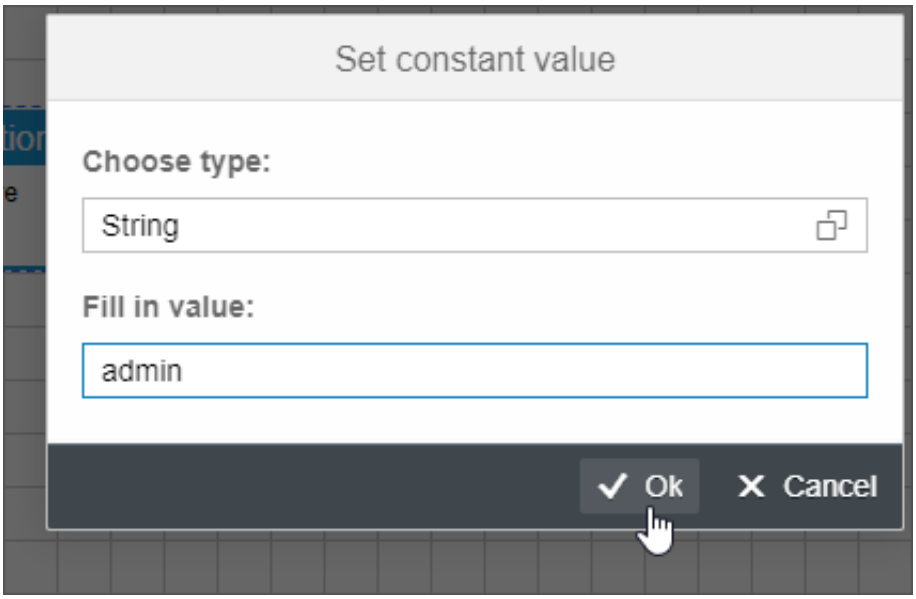
Comparing a Widget

Selecting a widget for comparison is guided via a three-step dialog. It asks you to select the screen in the first step, then one of the widgets on this screen and finally the property of the widget that you like to read for the comparison:



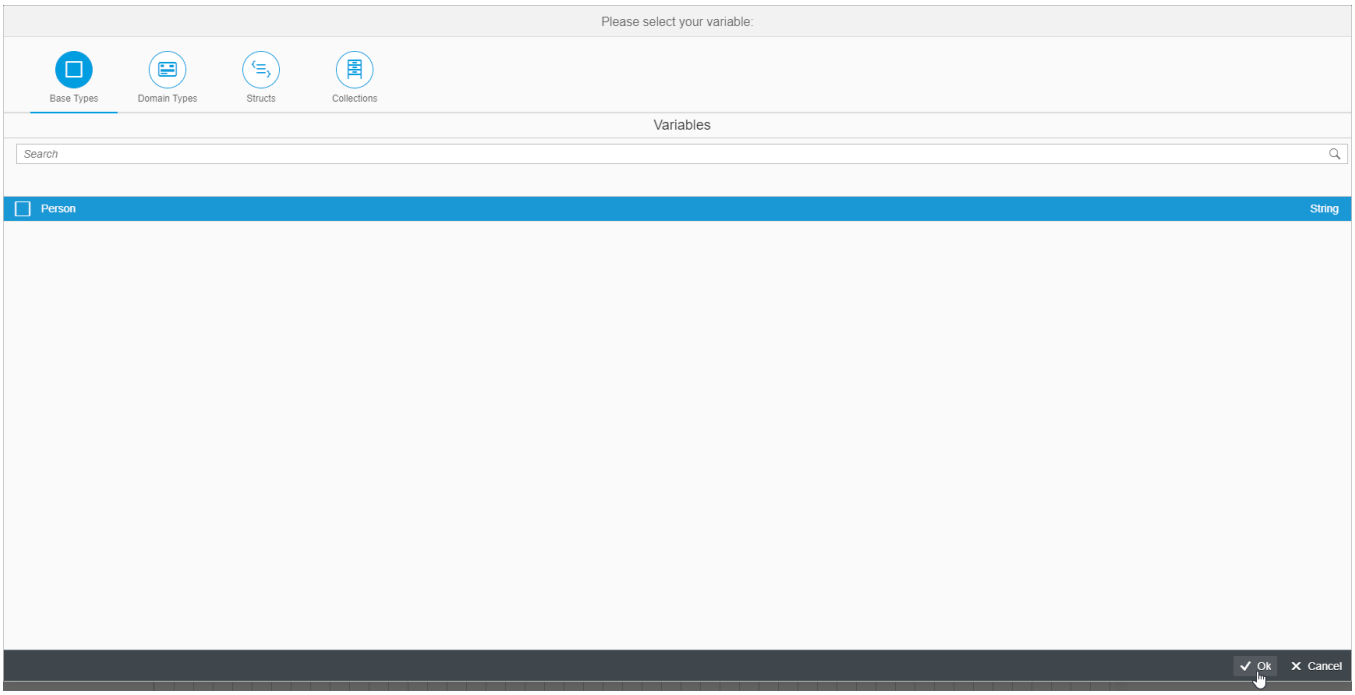
Comparing a Constant

Defining a constant value for comparison is even easier: you specify the data type via a selection in a dropdown box and set the value in the field below.



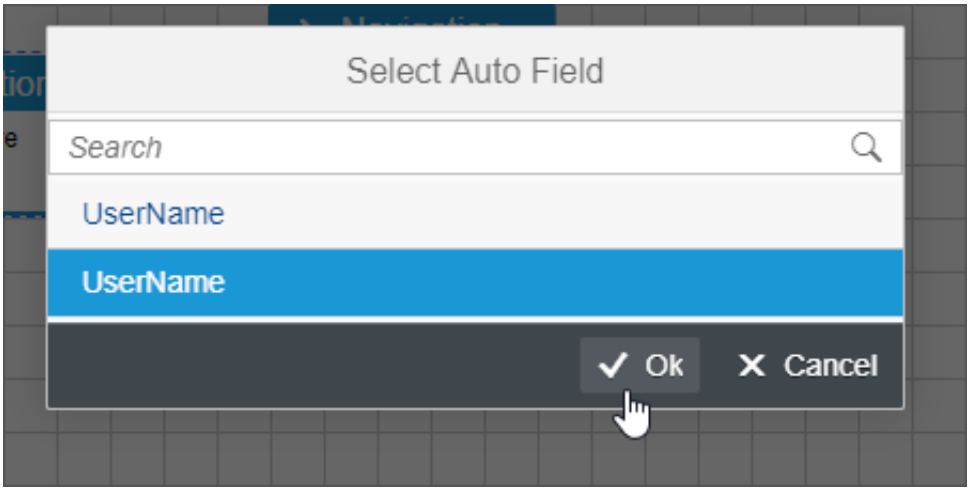
Comparing a Variable

If you want to compare a Variable, every existing one in the Data Workbench will be shown within the list. You can switch between the tabs Base Types, Domain Types, Structs an Collections.



Comparing an Autofield

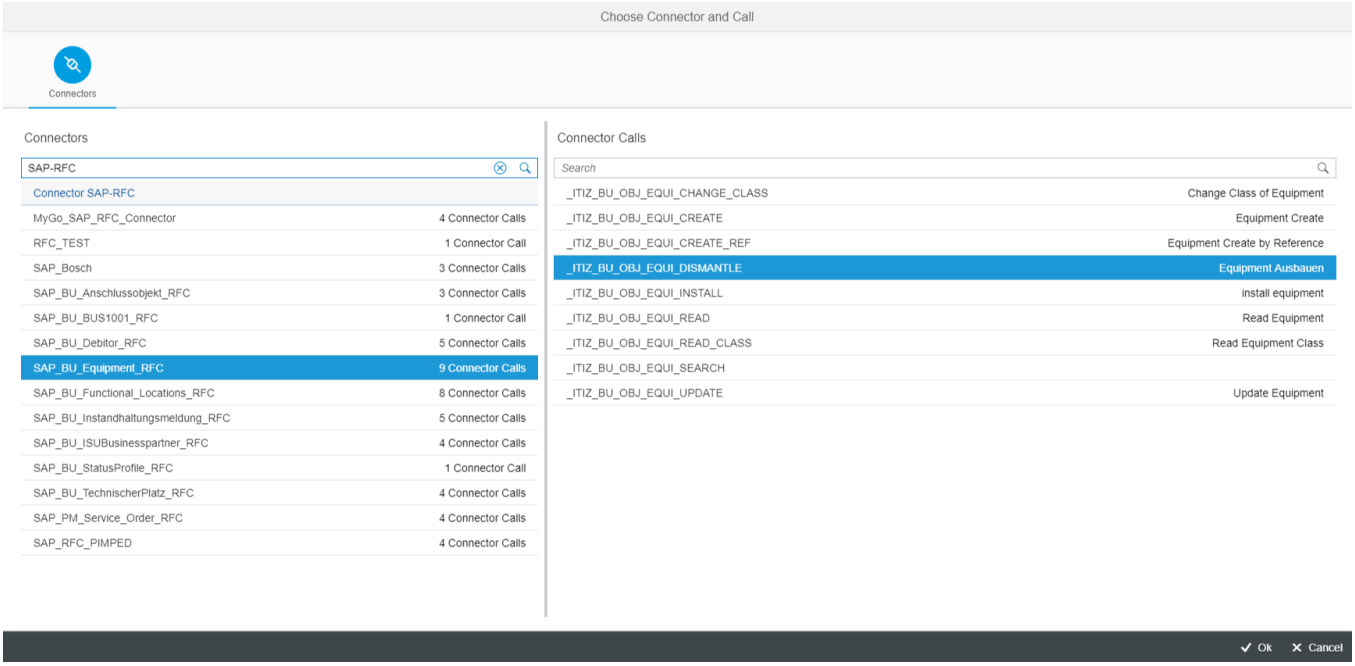
And you can select every generated Autofield for comparision as well.



Connector

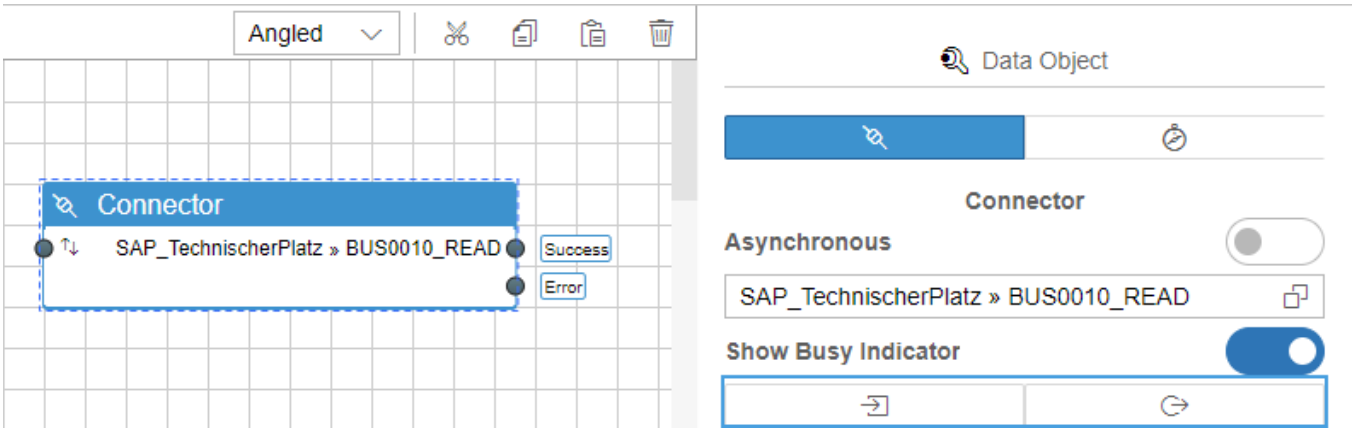
A typical example for a Data Object is a previously defined connector call. An assistant offers all configured connectors that owns connector calls. You can search for the connector name or even the connector type. If you select a connector, all calls are listed on the right side.

Choose the required Connector and the Call you want to execute.

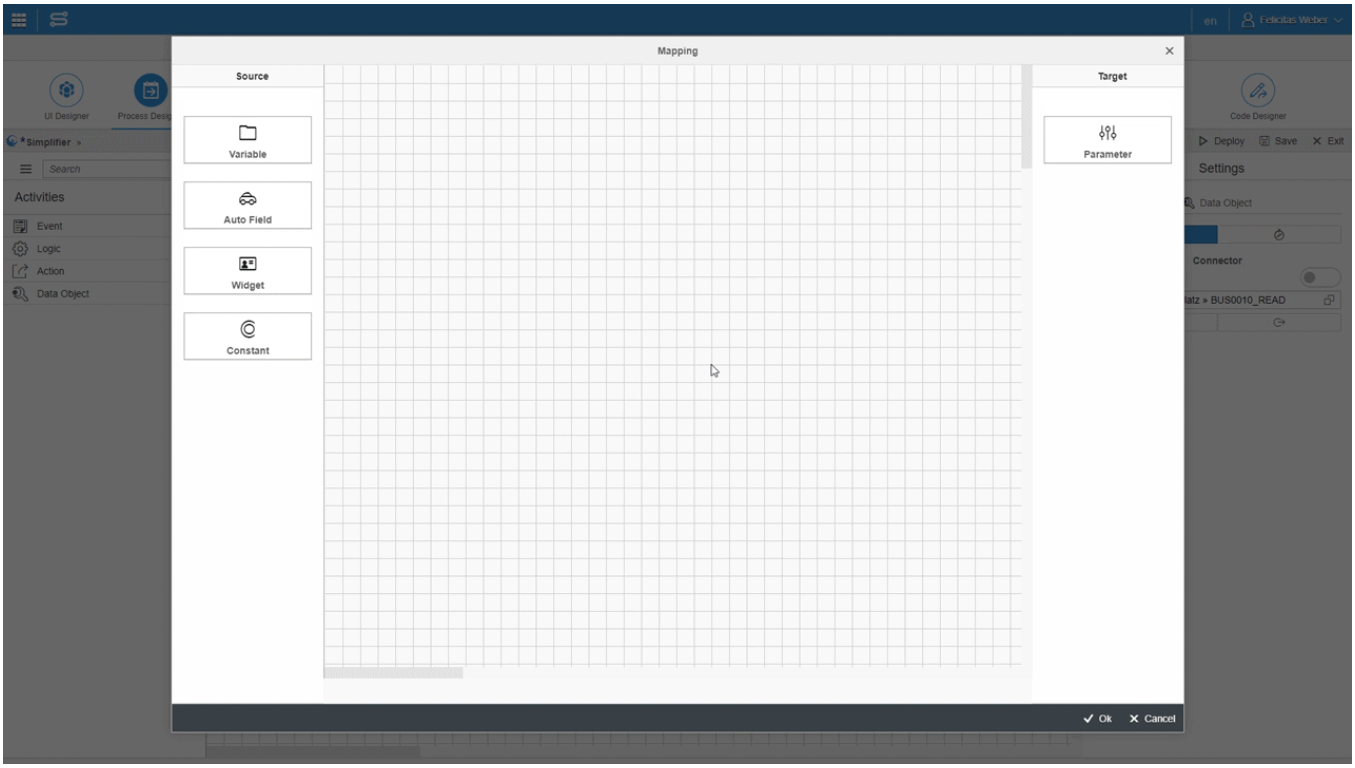


You have the possibility to configure if the UI is blocked by the **busy indicator**, or can configure which element on your screen should be blocked by it.

If you want to assign a connector call to a Data Object, you have to define the sources for the connector input (the request) and the destination for its output (the response). This is called the “Input Mapping” and “Output Mapping”:



Clicking one of these two buttons opens a new popup which allows you to connect widgets with the input or output attributes of a connector call. Independent of whether you're defining the input or output mapping, you'll always find the mapping sources in the left pane and the destinations on the right. If you define the input mapping, for example, you'll find the list of widgets in the left pane and the available connector call request parameters on the right. Within the output mapping dialog, the widgets are on the right pane, as they are the destination of the received data. Below is an example for the input mapping screen:



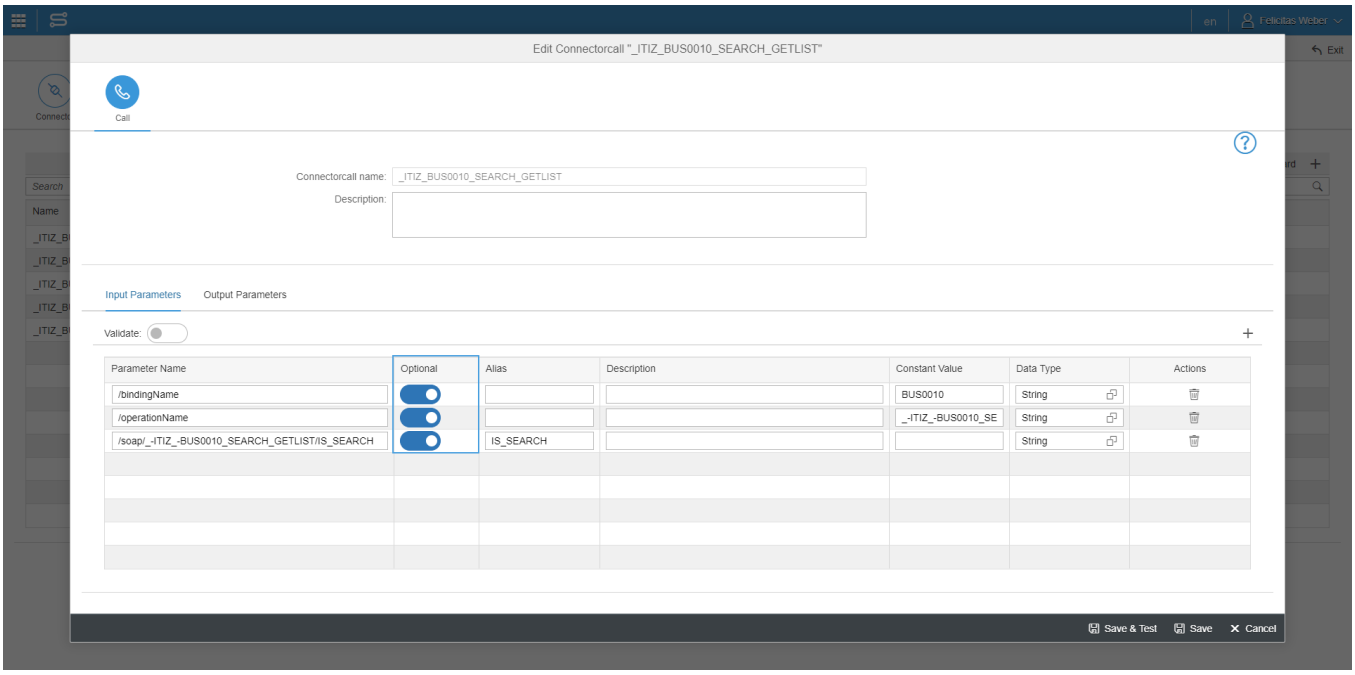
In this example, the request of the selected connector call provides only a single input parameter, an ID field named "TechnischerPlatzId". This input parameter is connected with the value of the widget "CustomerType". This means, that once the Data Object is triggered, e.g. through a button press event, the connector will be called with the actual value of the widget "CustomerType" as request parameter.

You can choose the exact source that you want to use for the input parameter by double-clicking on the widget. It opens a list of all properties so you can choose the one you need as input.

Connector Call Specific Parameters

The user interface for configuring a connector call is generic, thus it looks the same for all kinds of underlying types of connectors. Having the same interface for all kinds of connector calls is very convenient. But one drawback of this approach is, that some connectors require fixed parameters to be set, in order to work properly. This section tells you more about these details.

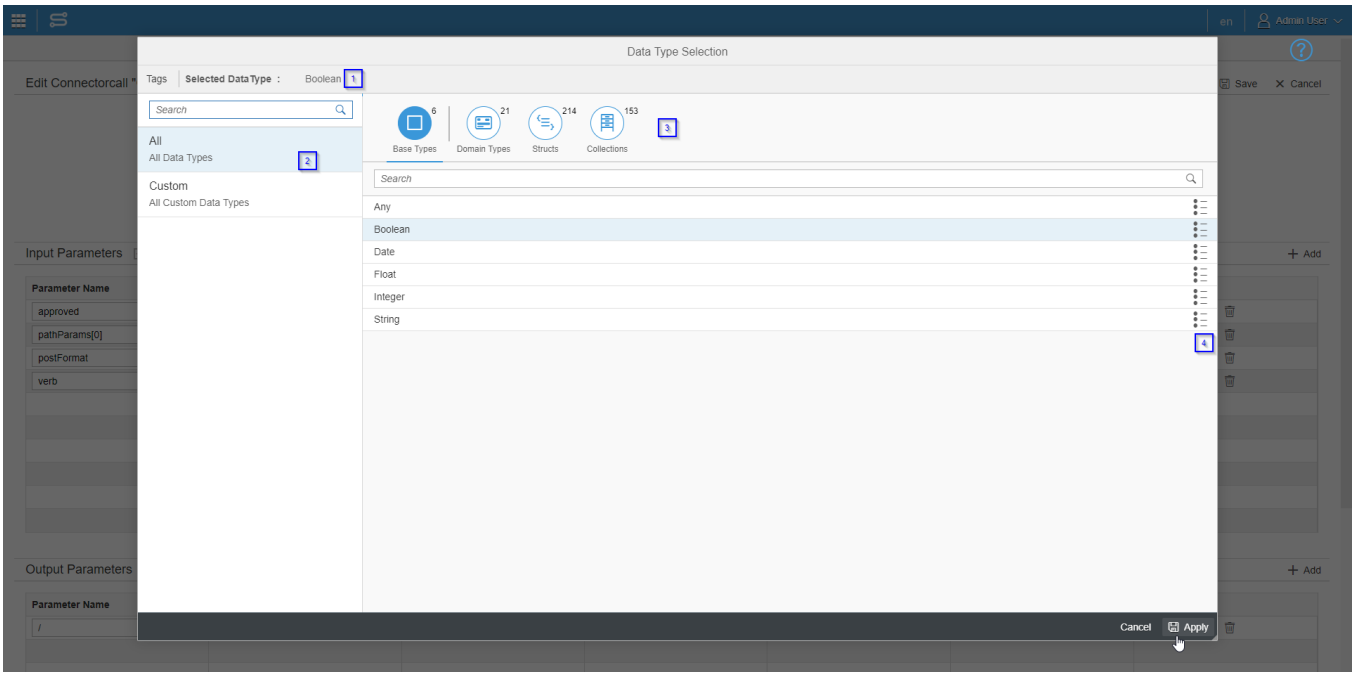
You can declare parameters of Connectors as optional.



When declaring a parameter as non optional, the validation of the call will fail if the parameter is not provided.

Data Type Selector

If you click on the data type field, a selector opens.



Number	Description
1	The currently selected data type.
2	In this filtering list of all data types, you can find manually and automatically built data types. Custom data types are only manual data types. When the dialog opens in an automatically generated connector call, the data types of the connector can also be selected.
3	Prefilter of base and domain types, structs and collections.
4	You can always step deeper in the structure to select a data type.

Connector Call via Script

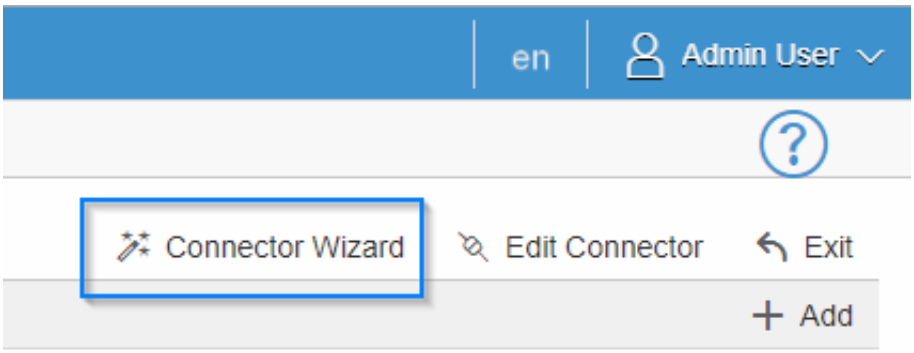
In order to execute a connector call please use this code snippet:

```
this.callConnectorCall(connectorName, connectorCallName, payload, callback, showBusyI
ndicator, failOnError, failCallback)
```

connectorName	the name of the connector
connectorCallName	the name of the connector call name
payload	a JSON object with the required parameters for the call
callback	function, which is called after the successful execution of the connector call
showBusyIndicator	boolean value that indicates whether the screen has to be blocked by a loading bar during the call (true) or not (false)
failOnError	boolean value that indicates whether the connector call should be called in case of an error of the function passed via “failCallback” (false) or not (true)
failCallback	function, which is called in case of an error in the connector call, if false for “failOnError” is passed

Connector Call Wizard

In the Connector Call overview you can find the assistant in the top right corner.

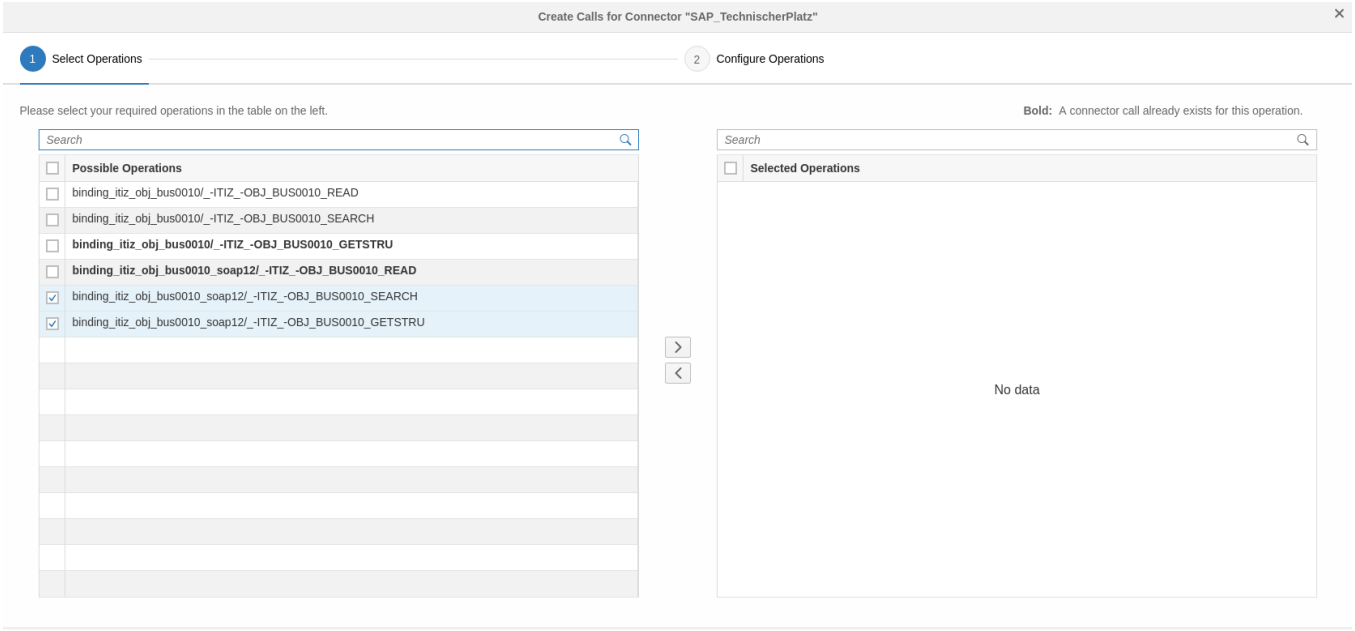


The assistant will open and helps you to create the Connector Call.

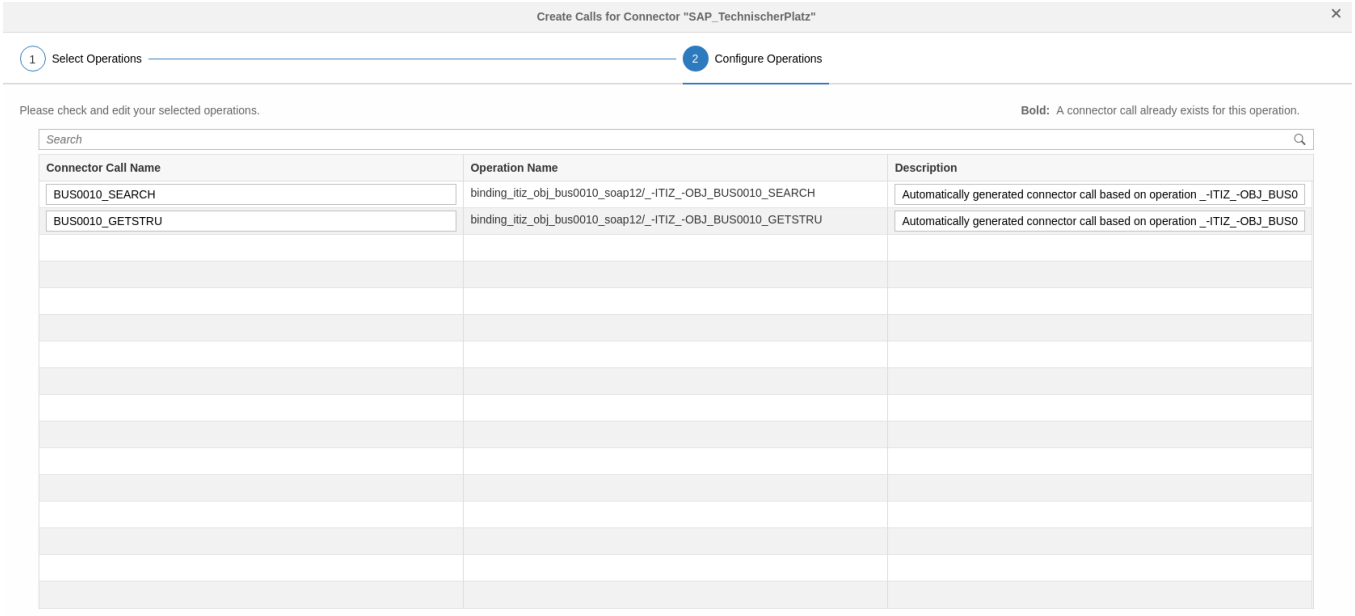
***Note:** At the moment the Wizard is only available for SOAP and SQL Connectors.

SOAP Connector Wizard

In the first step you can select the operation you want to execute.



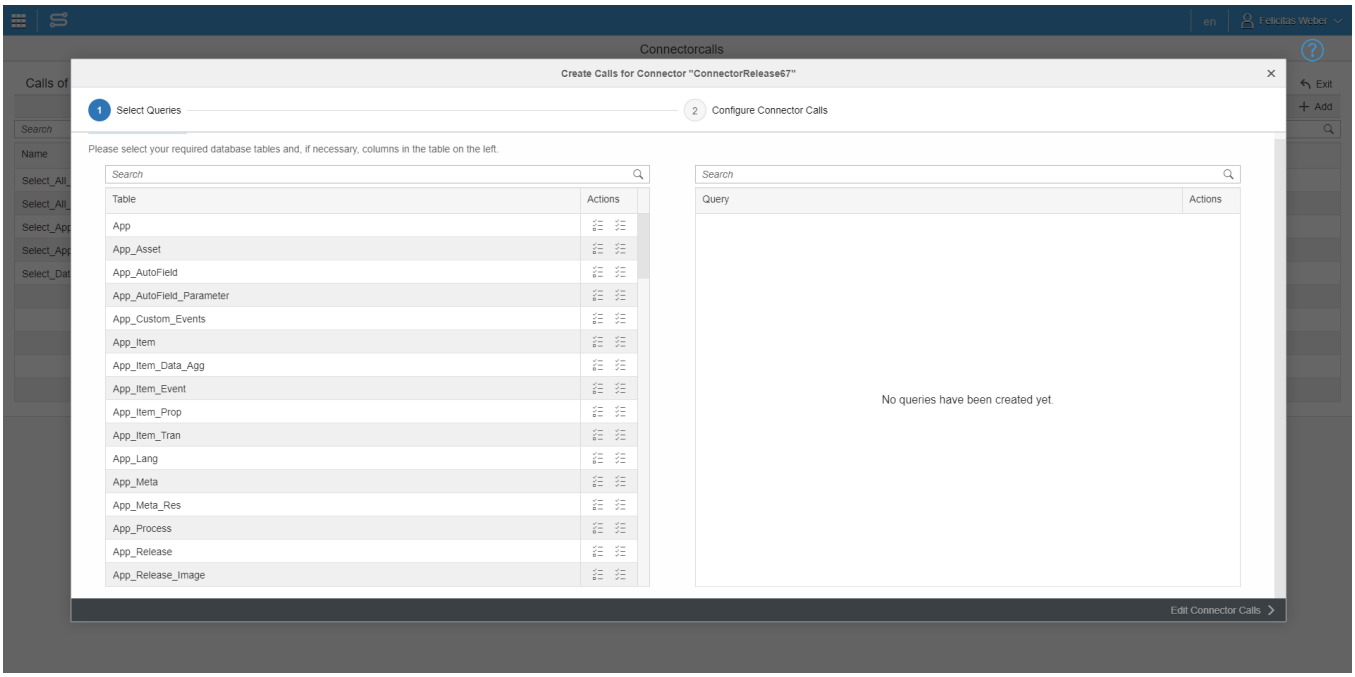
In the second step you can edit the name and description of the Connector Call.



SQL Connector Wizard

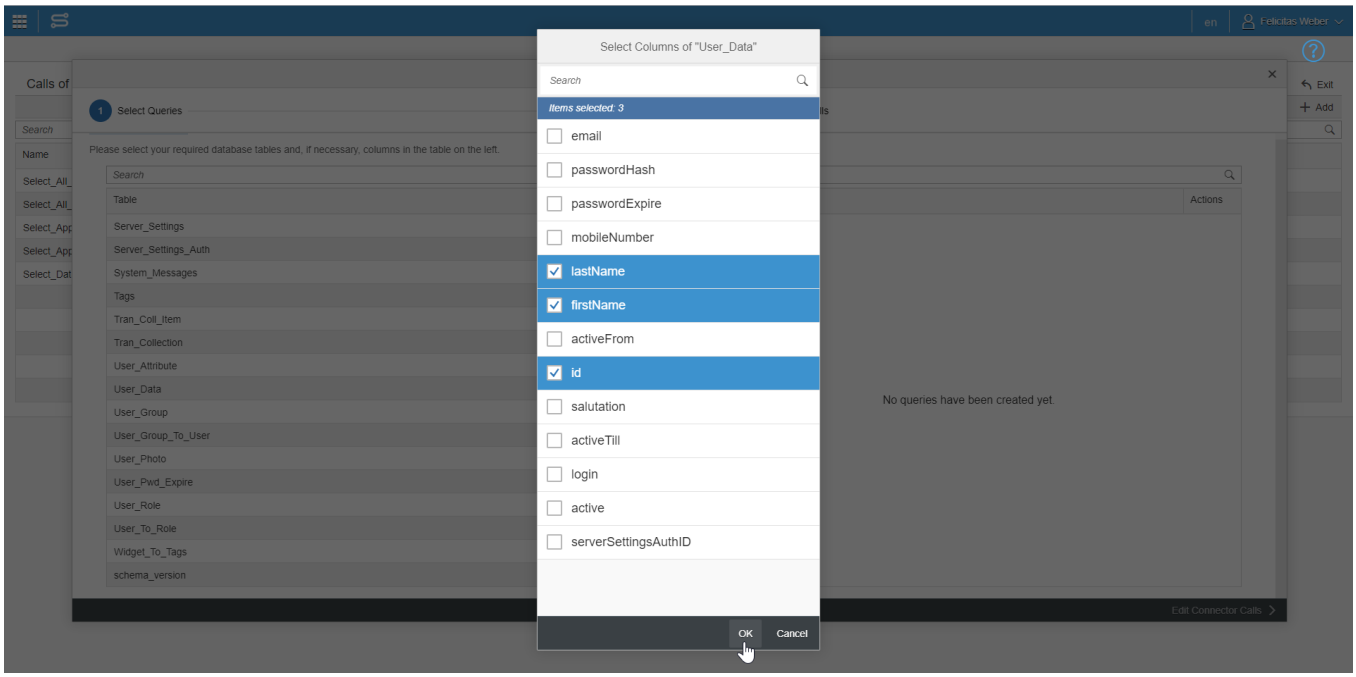
With the SQL Wizard it’s easy to configure new Connector Calls based on SQL Connectors for MySQL and Oracle.

When opening the Wizard, all tables available for the schema are displayed.

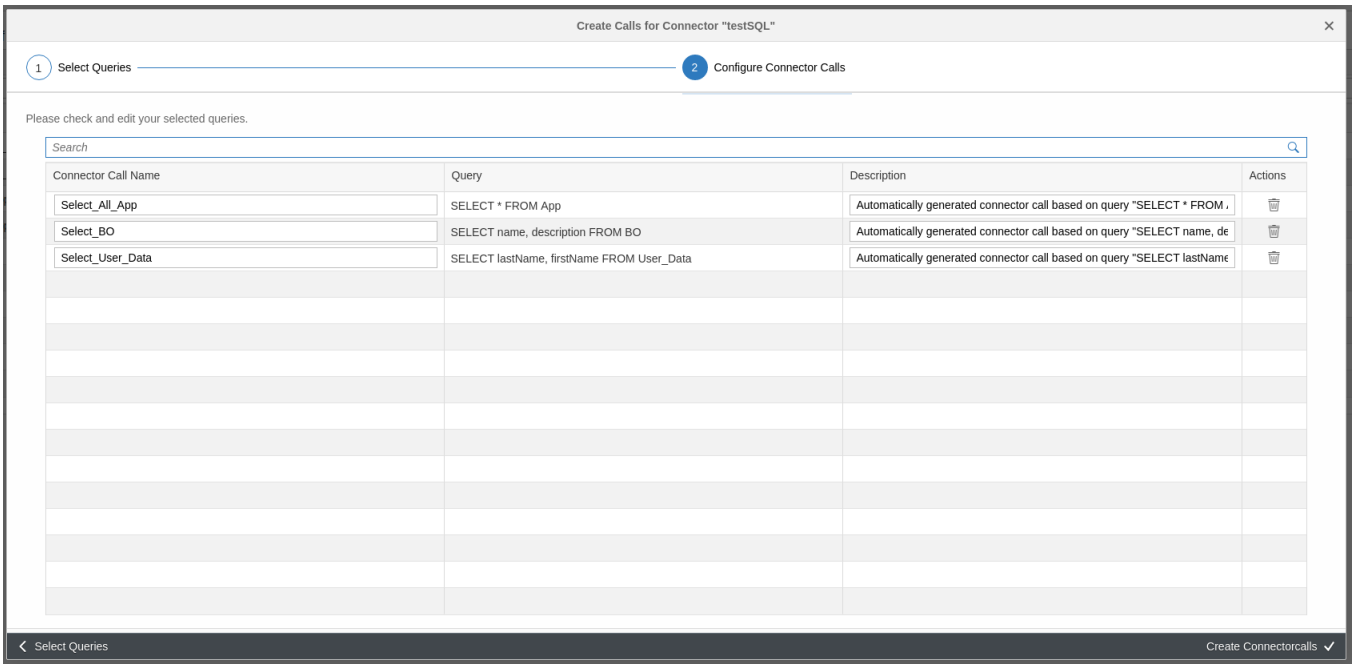


You can search for your Table, and then select under “Actions” whether you want to select all columns (right icon) or only certain ones (left icon).

If you only want to select certain ones, click on the left icon. A new popup opens in which you can select the desired columns. Then click OK.



After you have selected the desired tables and, if necessary, their columns, you can edit the connector calls by clicking on “Edit Connector Calls”.



When you have edited the calls, click on “Create connectorcalls”.

Connector Calls

Connectors define the connection (entry point) to an external system. Given such a connection, you might send several different requests to the connected system. We call one such concrete pair of request / response a “connector call”. In order to use a connector call in the edit mode of a user story (Process Designer), you must create at least one connector call for each connector.

Connector Details

In step three, you can add data to your Connector.
Each Connector has specific details that depend on the properties of the communication protocol.

Connector Types

You can choose between different Connector Types. A Connector Type represents the technical protocol.

Connector Type	Description
Push Notification	Sends Push Notifications over Websockets directly to Simplifier Clients or Simplifier Browser Apps without using Google or Apple's Cloud services to support data privacy and protection.
oData	oData Proxy for using oData v2 Services . Open Data Protocol (OData) is an open protocol which allows the creation and consumption of queryable and interoperable RESTful APIs in a simple and standard way.
SOAP	**S**imple **O**bject **A**ccess **P**rotocol based on HTTP and XML Format.
CSV	Read and/or write comma separated files on a local file store.
MQTT Client	Message Queue Telemetry Transport (MQTT) is a lightweight messaging protocol for small sensors and mobile devices, optimized for high-latency or unreliable networks. This Connector acts as a client and can publish or subscribe messages from a mqtt broker.
Logging Write	This Connector can transfer the Simplifier application logs to a central monitoring tool / logwatch server.
REST	Connector for HTTP REST Services. Representational State Transfer (REST) architecture uses standardized operations (GET, PUT, POST, DELETE) on web services.
SQL	Database Connector for executing SQL Statements on a database schema.
Email	Email Connector for sending Emails over SMTPS (Simple Mail Transfer Protocol) with or without SSL Encryption.
OPC-UA	Open Platform Communication – Unified Architecture Connector, it connects to an OPC-UA server and performs READ/WRITE/SUBSCRIBE operations. For now READ/WRITE operations are supported through connector calls, here and Live-Value Monitoring through websockets, here.
SAP RFC	Connector based on standard JCO SAP RFC Connector to call remote function blocks

Connector via Script

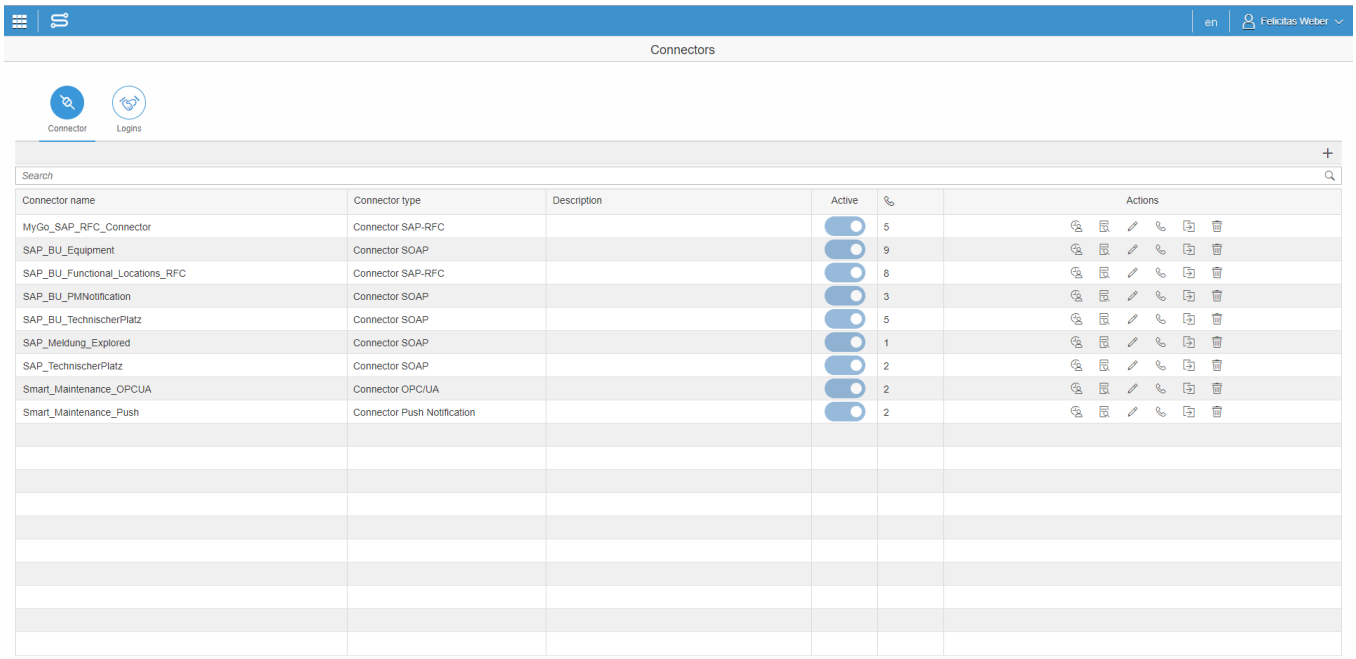
```
this.callConnector(connectorName, payload, callback, showBusyIndicator, failOnError, failCallback)
```

connectorName	the name of the connector
payload	a JSON object with the required parameters of the call
callback	function, which is called after the successful execution of the connector
showBusyIndicator	boolean value that indicates whether the screen has to be blocked by a loading bar during the call (true) or not (false)
failOnError	boolean value that indicates whether the connector should be called in case of an error of the function passed via “failCallback” (false) or not (true)
failCallback	function, which is called in case of an error in the connector, if false “failOnError” is passed

Connectors

Connectors are used to retrieve data from backend systems and third-party sources and write back changed data into them. They can be used by a unique name within configured business applications or business objects. It represents the systems part of the simplifier.

Connector	Adresses a specific backend systems like SAP, Salesforce or a Database
Connector Call	Brings a specific connector into action. It contains input and output parameters


en
Felicitas Weber
Connectors
Connector
Logins
Search
Connector name
Connector type
Description
Active
Actions
MyGo_SAP_RFC_Connector
Connector SAP-RFC
5
SAP_BU_Equipment
Connector SOAP
9
SAP_BU_Functional_Locations_RFC
Connector SAP-RFC
8
SAP_BU_PMNotification
Connector SOAP
3
SAP_BU_TechnischerPlatz
Connector SOAP
5
SAP_Meldung_Explored
Connector SOAP
1
SAP_TechnischerPlatz
Connector SOAP
2
Smart_Maintenance_OPCUA
Connector OPC/UA
2
Smart_Maintenance_Push
Connector Push Notification
2

Convert XML to/from JSON

You can use this template to convert XML to/from JSON.

Convert XML to JSON

```
Simplifier.Util.xml2json(xml: string): string
```

Parses an XML string, converts it to JSON and returns the representing JSON string.

```
try{
    var convertedJson = Simplifier.Util.xml2json("<xml></xml>");

    output.json = convertedJson;
    output.message = "Convertion successful";
    output.success = true;
} catch (e) {
    output.message = e.message;
    output.success = false;
}
```

Convert JSON to XML

```
Simplifier.Util.json2xml(json: string): string
```

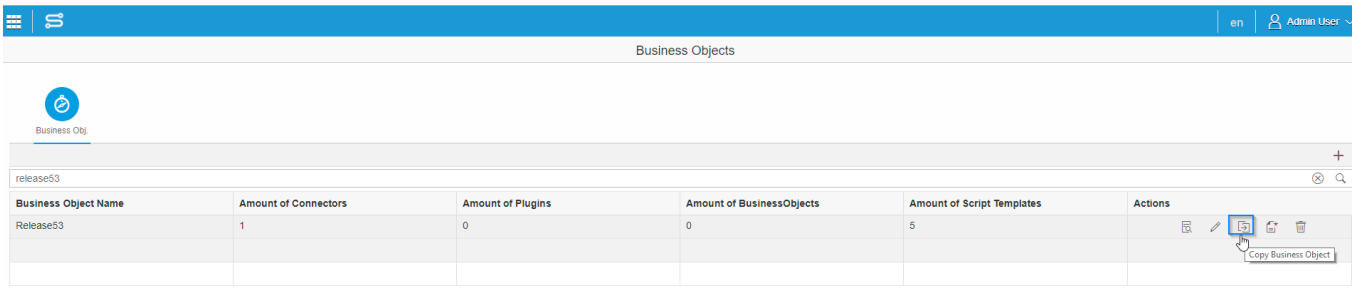
Parses a JSON string, converts it to XML and returns the representing XML string.

```
try{
    var convertedXML = Simplifier.Util.json2xml(JSON.stringify({attribute: "value"}))
;

    output.xml = convertedXML;
    output.message = "Convertion successful";
```

```
    output.success = true;
} catch (e) {
    output.message = e.message;
    output.success = false;
}
```

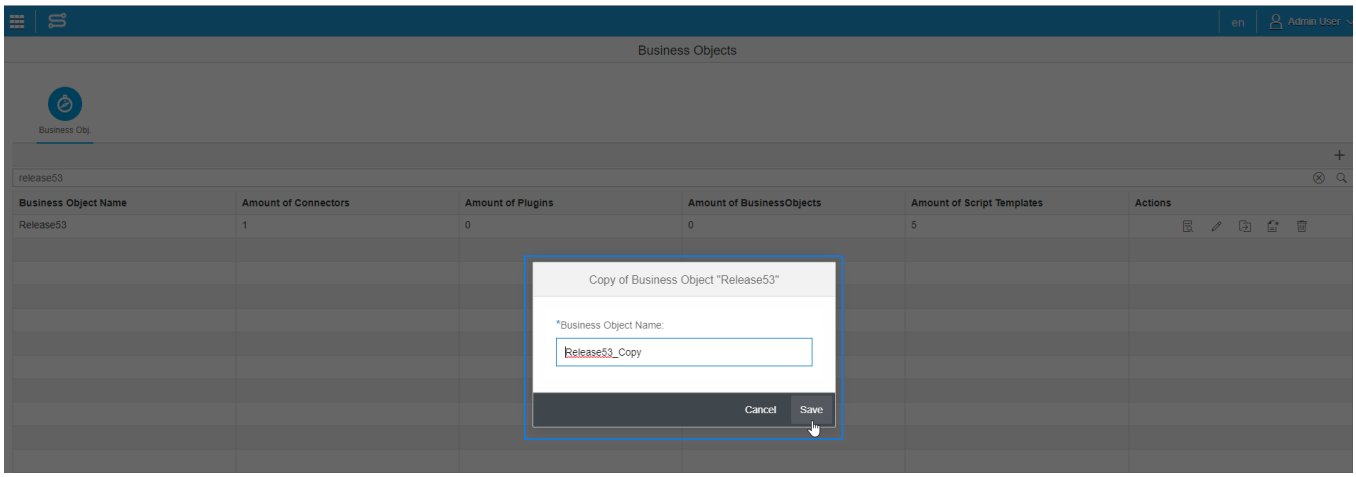
Copy Business Objects

To copy a Business Object, just click in the Overview of Business Objects on the ‘Copy Business Object’ Button.

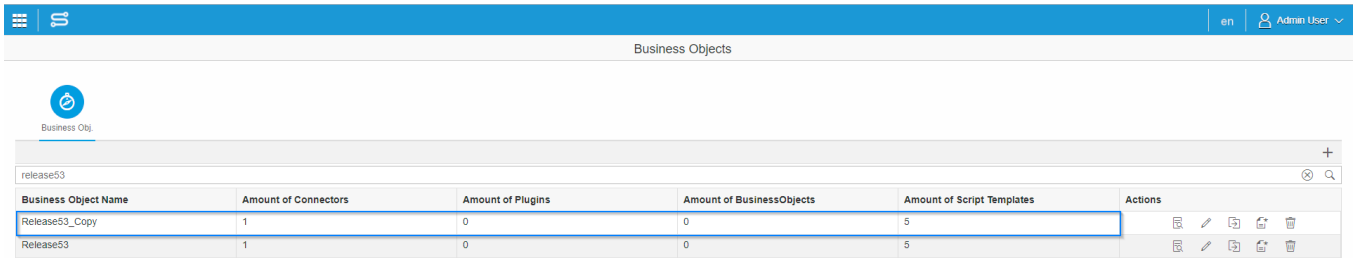


Copy Business Object

After you have clicked on it, a pop-up appears in which you can specify the new name of the Business Object. Then click on ‘Save’.

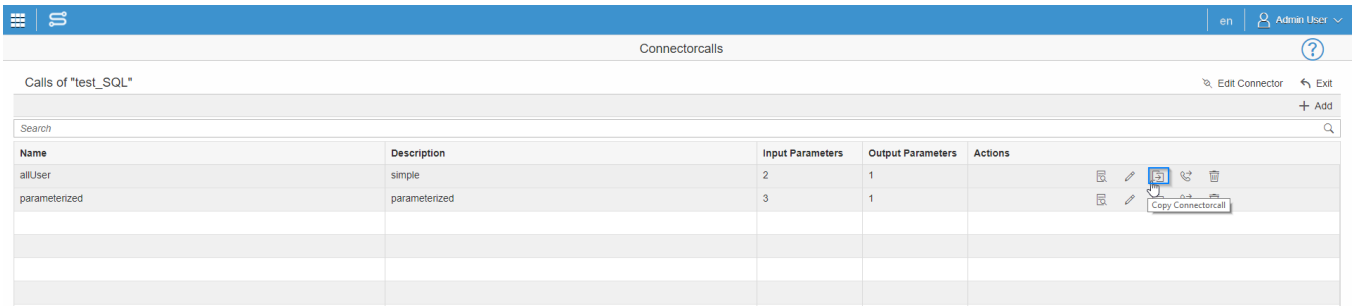


Now the Business Object has been copied. All included Connectors, Plugins and other Business Objects as well as the Script Templates are available in the copy.



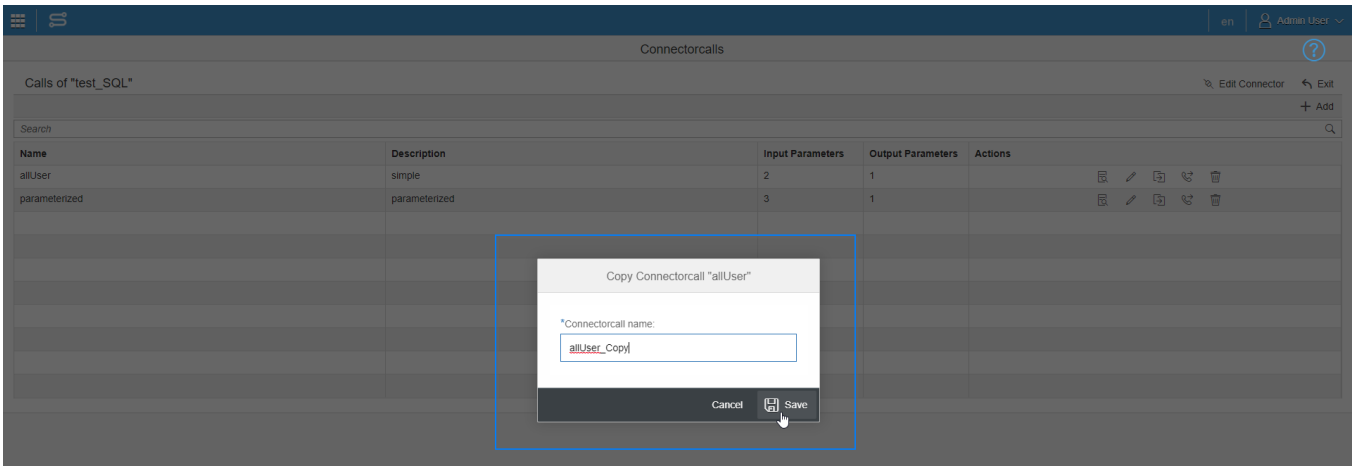
Copy Connector Calls

You can copy a connector call within a connector in the connector call overview by clicking the appropriate copy button.



copy a connector call

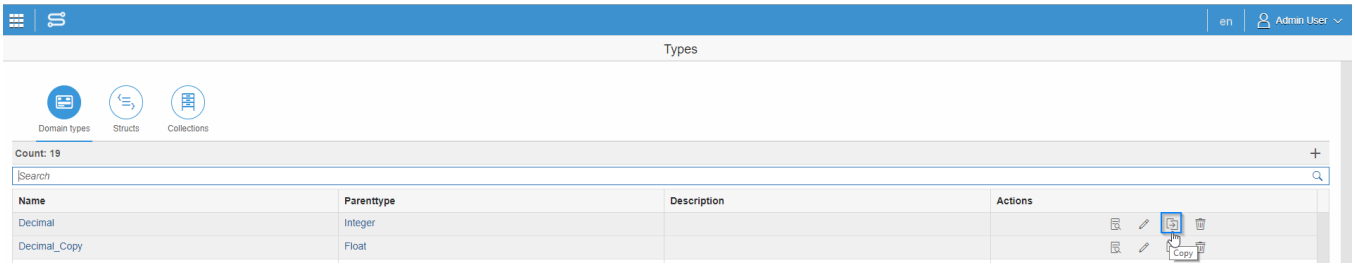
By clicking the button a new pop up opens in which you can specify the name of the copied connector call. The default value is the name of the copied connector call added _copy.



Once you have assigned a name, click on the save button. Your connector call has been copied with all input and output parameters.

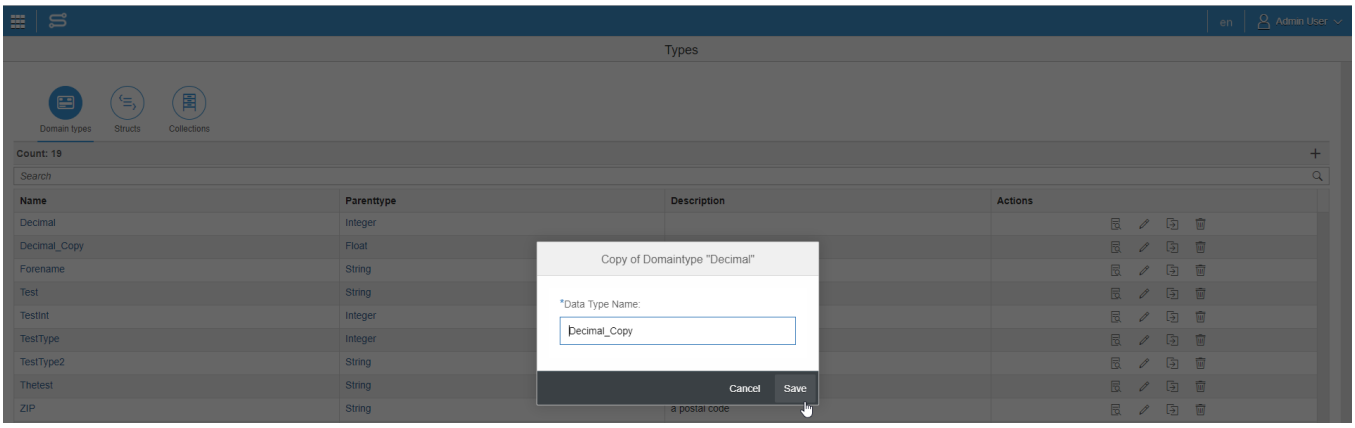
Copy Data Types

You can copy any Data Type of the Simplifier. The copy will have all attributes/fields and any tags given to the copied template.



Copy Data Type

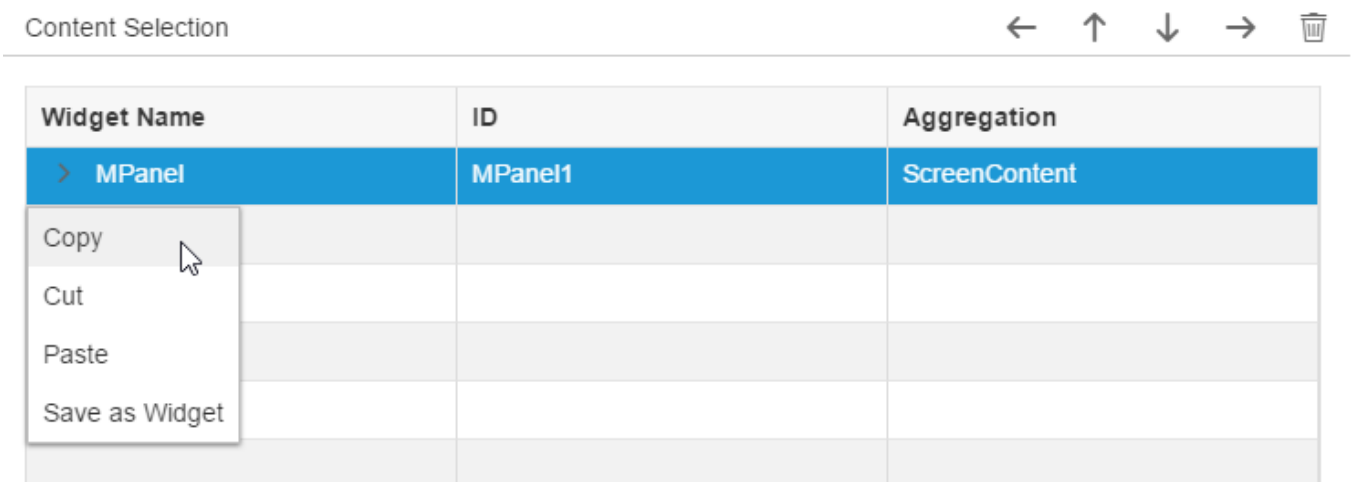
Click on the copy icon and a new pop-up will appear.



By default, “_Copy” is added to the current Data Type Name. However, you can also assign a new name. Click ‘Save’ and the copy has been created.

Copy your Widgets

You can copy Widgets and insert them on a different Screen or even another Application.

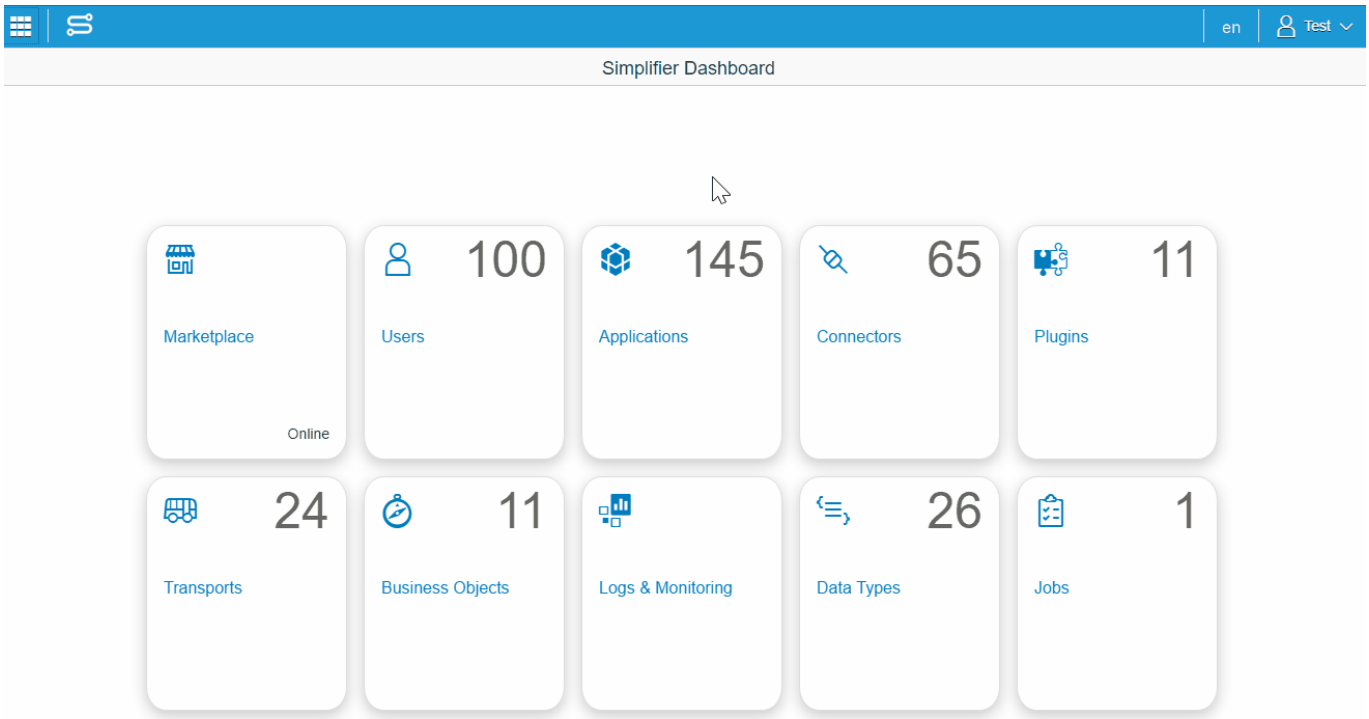


Create a New Application

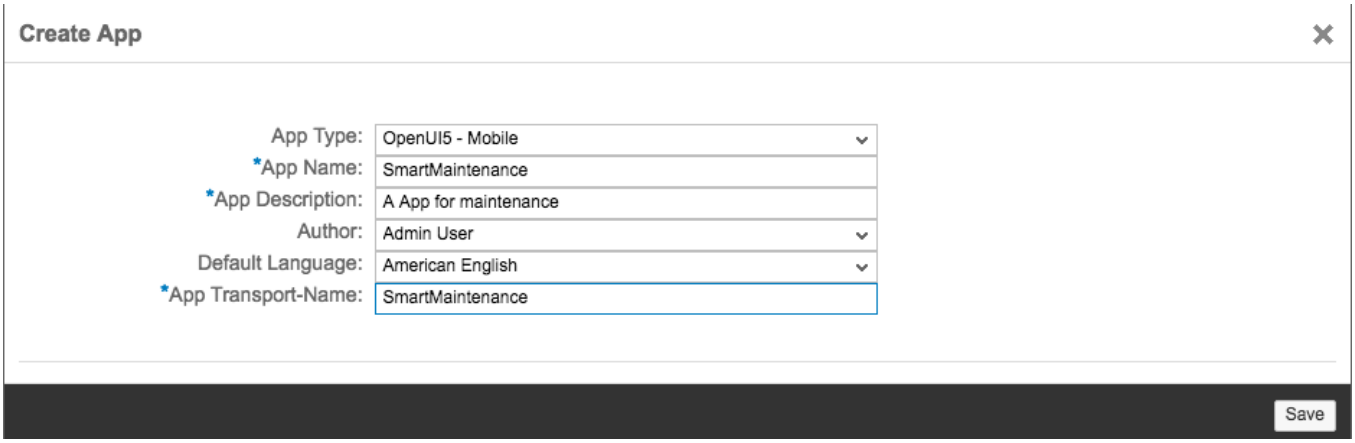
Please set up the following checkpoints before you create an app:

- All connector calls should be configured in case of using backend interfaces
- You should know the right technology for your use-case before you build app

To create a new application, click on the Applications module. Then click on the “+” button on the top right to create a new business app:



The following dialog will appear:



Fill out the necessary fields analog to the table below:

App-Type	Choose your Technology for your business app: • OpenUI5 – Mobile for mobile applications • OpenUI5 – Portal for complex web portals • Angular 2 For more information, see the chapter Technology
App-Name	Unique app name like “SmartMaintenance”
App-Description	A short description of your app
Author	The Simplifier username of the app author
Default Language	Your default language for configuration – you can translate this language later via the language translation feature
App-Transport-Name	The name of the transport for transferring your configuration to another simplifier instance like a quality assurance system

After the app creation you can configure it with the following tabs:

UI Designer

Configure the User-Interface and mark all necessary events to design your workflow.

Process Dashboard

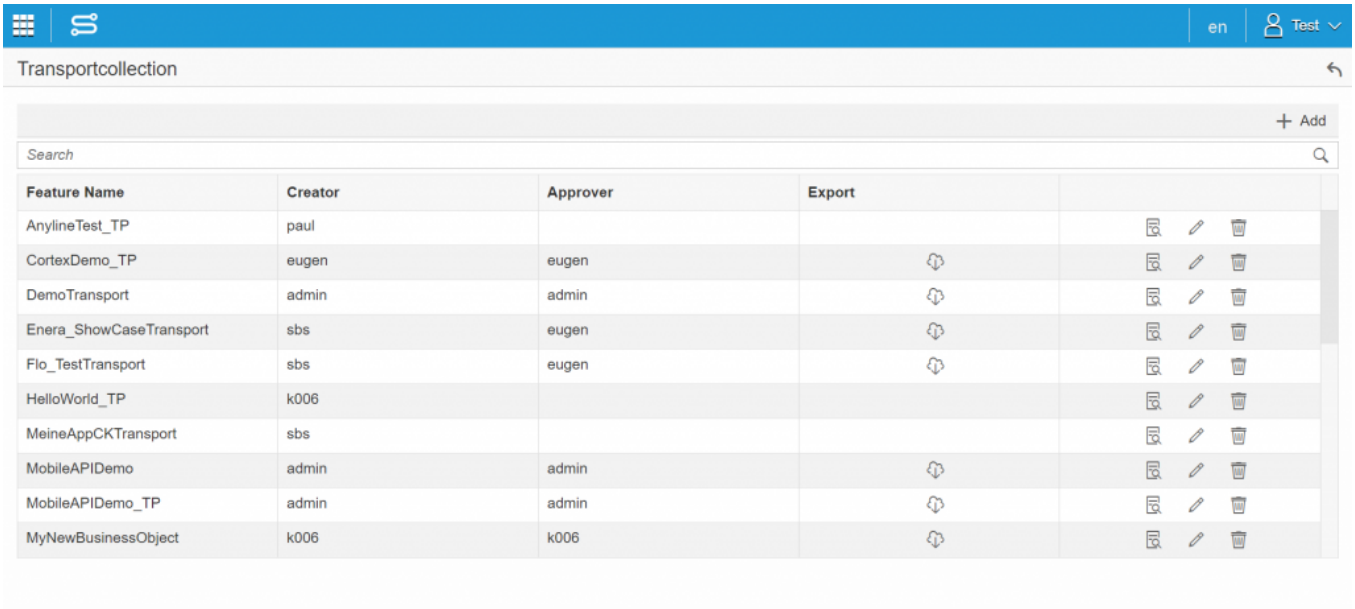
Define your workflow with simple visual events and actions.

Code Designer

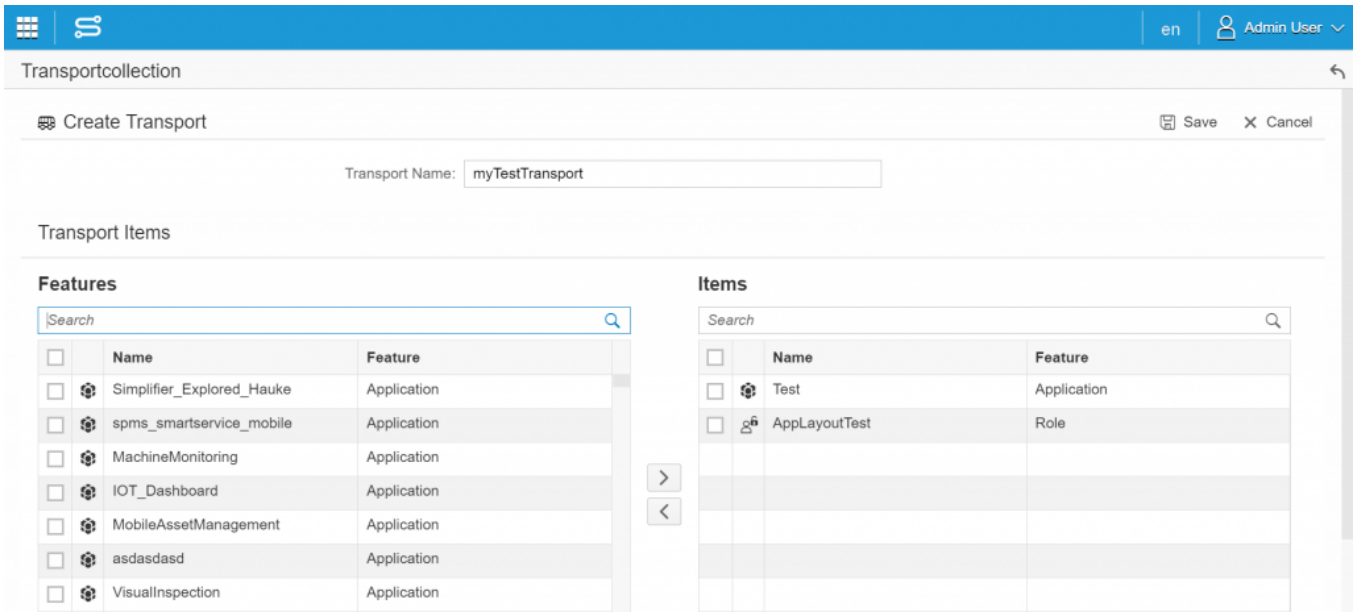
Add additional javascript files to your app and extend it in web-based integrated development environment.

Create an individual Transport

Enter tile “Transports” on the Simplifier Dashboard and click on Transportcollection to define a new Transport or edit an existing one. This leads you to the overview page, which lists all existing Transports:

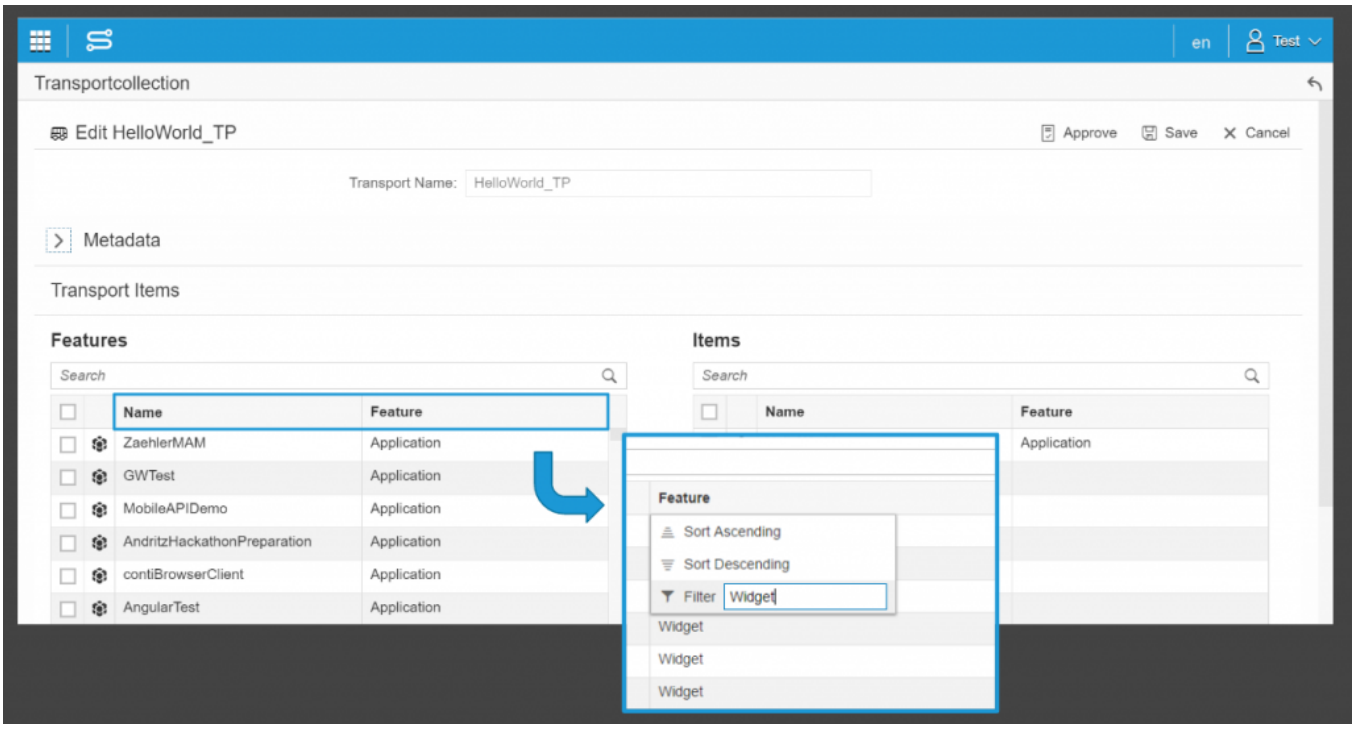


To create a new Transport, or edit an existing one, make sure you’ve got the “Transport Creation” permission associated to one of your roles. Click on “+” on in the upper right corner to create a new Transport from scratch or press the edit button beside one of your existing one to edit it. This leads you to a new page which allows you to define all Transport artifacts:



The upper part of the screen allows you to enter a name for your Transport.

The lower part of the screen is dedicated to the selection of the artifacts which shall be part of your Transport. The left pane lists all available features (Apps, Connectors, Roles, Business Objects, Datatypes, Widgets & Libraries) and allows you to search for specific ones. Ideally, the features of your app have an identical name part, so you can easily find all belonging features through the search bar. You can filter the list by clicking on “Name” or “Feature” and select e.g. only Widgets.



Select an artifact by checking the checkbox beside it. After clicking the right arrow, all selected artifacts are immediately listed in the right pane. (Attention: The “Application” feature represents only the cover of your app and its standard Widgets, not all your used features like e.g. customized Widgets, Business Objects or Connectors!

When you’re done, press the “Save” button in the upper right corner or “Cancel”, if you want to omit your changes.

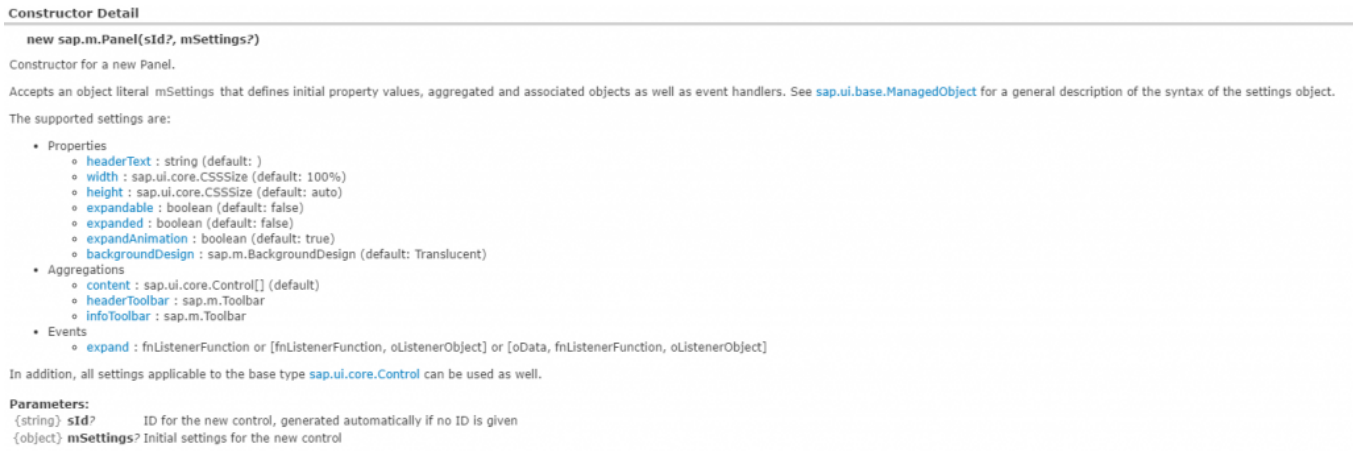
Create an OpenUI5 Widget

If you choose to create a new OpenUi5 Widget, you have to take a look at the constructor details in its API reference.

Here you can find the OpenUi5 API of all Widgets and a description when to use them.

Start

Let’s create a mobile version of a Panel. For starters search new sap.m.Panel in the API reference.
You will see the supported settings, in this case: Properties, Aggregations and Events.



Constructor Detail

new sap.m.Panel(sId?, mSettings?)

Constructor for a new Panel.

Accepts an object literal mSettings that defines initial property values, aggregated and associated objects as well as event handlers. See sap.ui.base.ManagedObject for a general description of the syntax of the settings object.

The supported settings are:

- Properties
  - headerText : string (default:)
  - width : sap.ui.core.CSSSize (default: 100%)
  - height : sap.ui.core.CSSSize (default: auto)
  - expandable : boolean (default: false)
  - expanded : boolean (default: false)
  - expandAnimation : boolean (default: true)
  - backgroundDesign : sap.m.BackgroundDesign (default: Translucent)
- Aggregations
  - content : sap.ui.core.Control[] (default)
  - headerToolbar : sap.m.Toolbar
  - infoToolbar : sap.m.Toolbar
- Events
  - expand : fnListenerFunction or [fnListenerFunction, oListenerObject] or [oData, fnListenerFunction, oListenerObject]

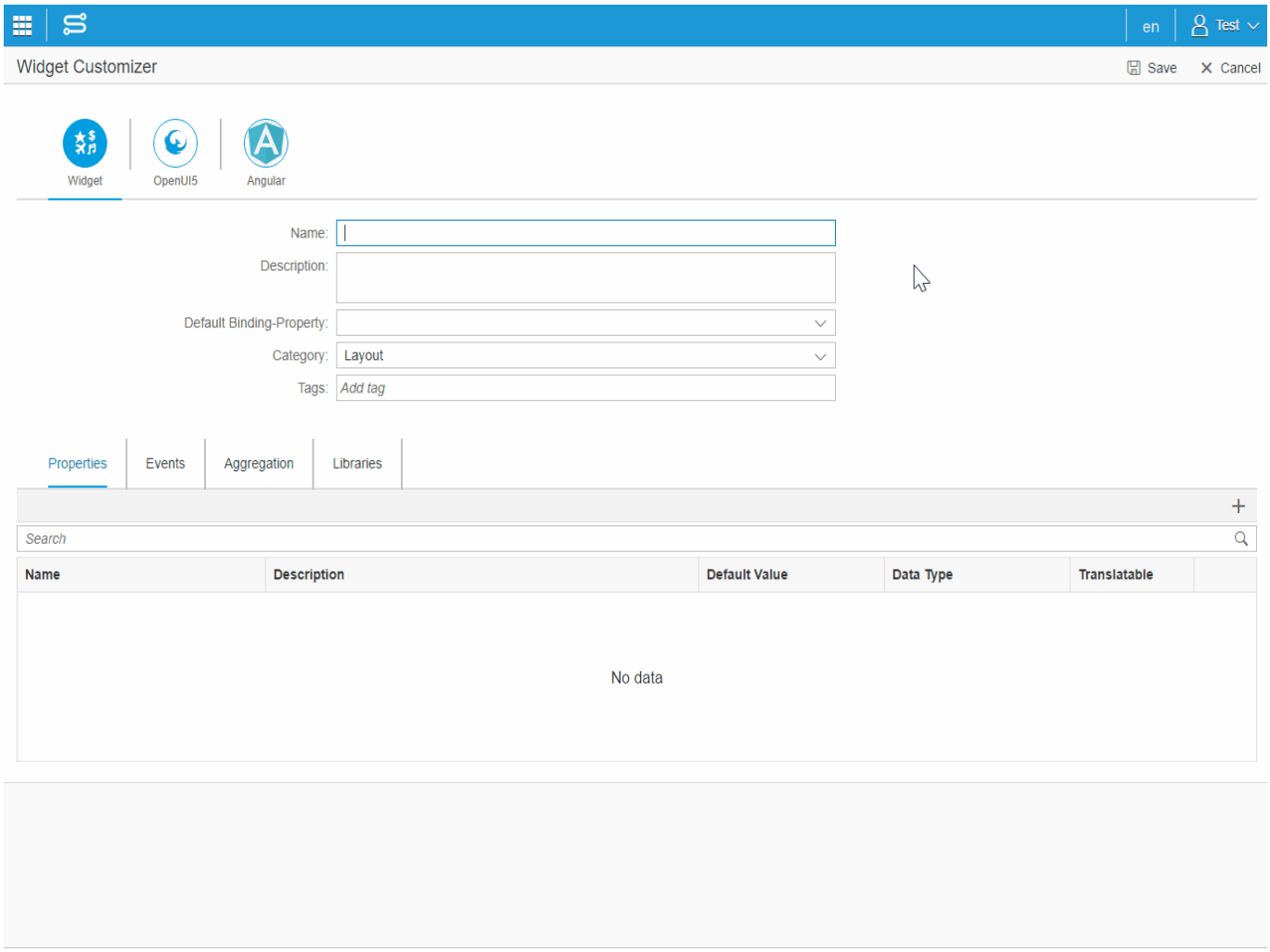
In addition, all settings applicable to the base type sap.ui.core.Control can be used as well.

Parameters:
{string} **sId**? ID for the new control, generated automatically if no ID is given
{object} **mSettings**? Initial settings for the new control

- Properties describe the different attributes of an element (e.g. width or height).
- Aggregations describe which other elements the Widget could contain (e.g. a panel consists of a header & info toolbar and content).
  Depending on the Control that is displayed in the API, you can use every or just specific Controls (e.g. sap.ui.core.Control vs sap.m.Toolbar)
- Events describe the possible direct interactions for the user (e.g. expand the panel).

Step 1

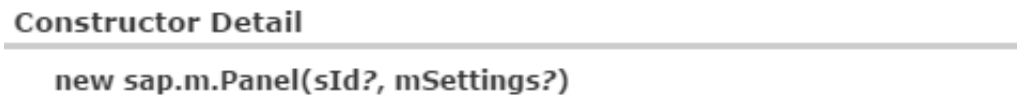
For the first step name your Widget, write a short description and choose a category in the Widget tab.



In addition you've got now the possibility to add custom tags to the widget. You can search and filter for widget tags in the search field of the widget overview list and in the widget search field of the UI designer.

Step 2

Click on the OpenUI5 tab to fill out the specific parameters.
The Widget Type has to be the same as the UI5 control name. In this case: *sap.m.Panel*



API reference

Simplifier Widget Type

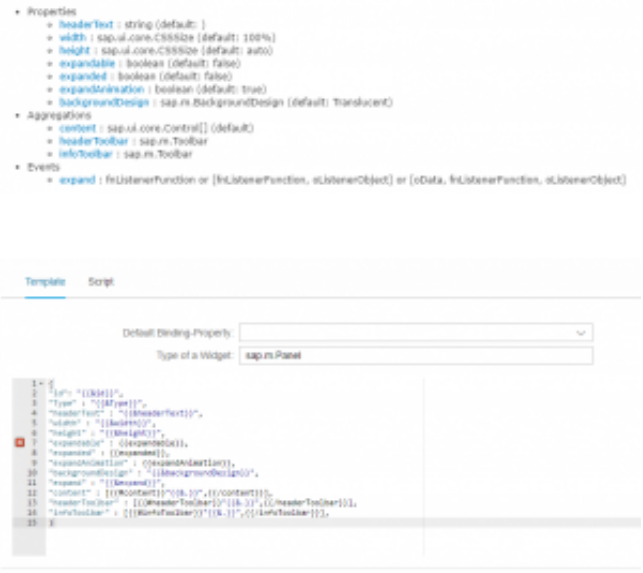
Step 3

Fill in the template for the Widget. It represents a blueprint of the Widget which can be used by the Simplifier.
The OpenUI5 templates are written in JSON with Mustache placeholder syntax.

The Simplifier supports three different types:

- String: “{{&placeholderName}}”
- Boolean: {{placeholderName}}
- Aggregation: [{{#placeholderName}}”{{&}}”,{{/placeholderName}}]

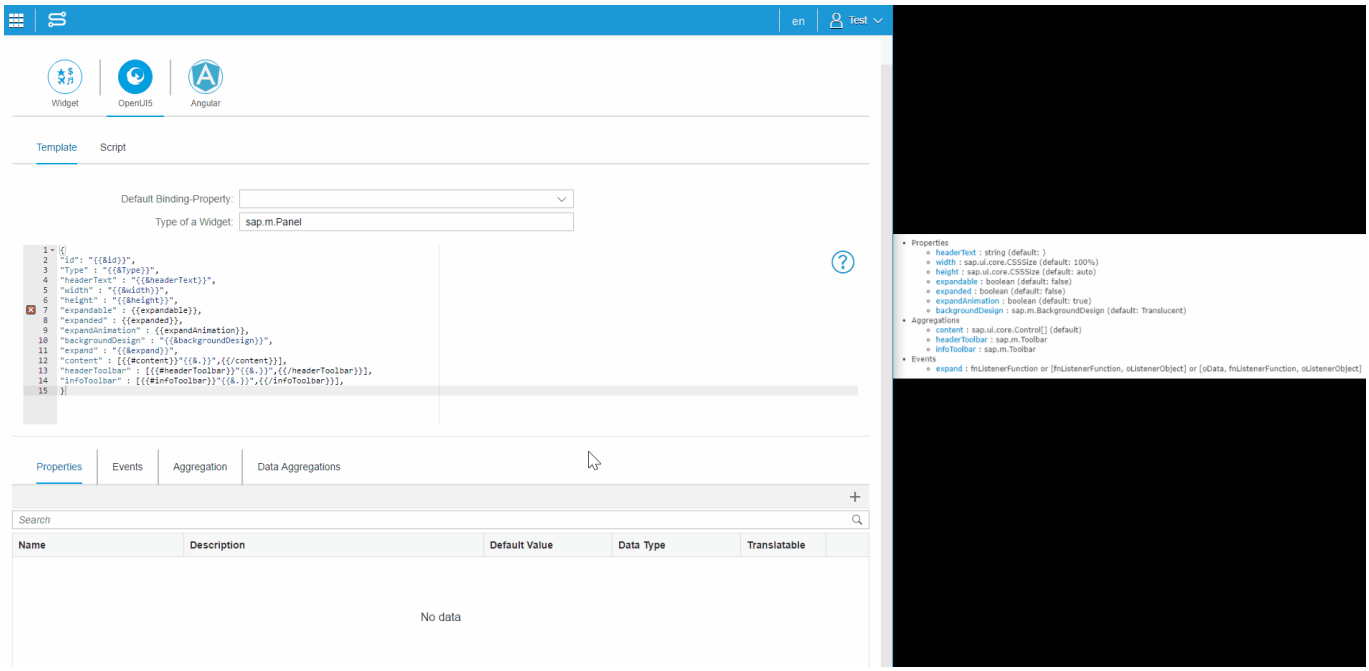
The placeholderName will be used to declare properties, events, etc.. It is advisable to use the OpenUI5 names.
For SAP Controls (e.g. sap.m.BackgroundDesign) you can use the „String“ Data Type.



Step 4

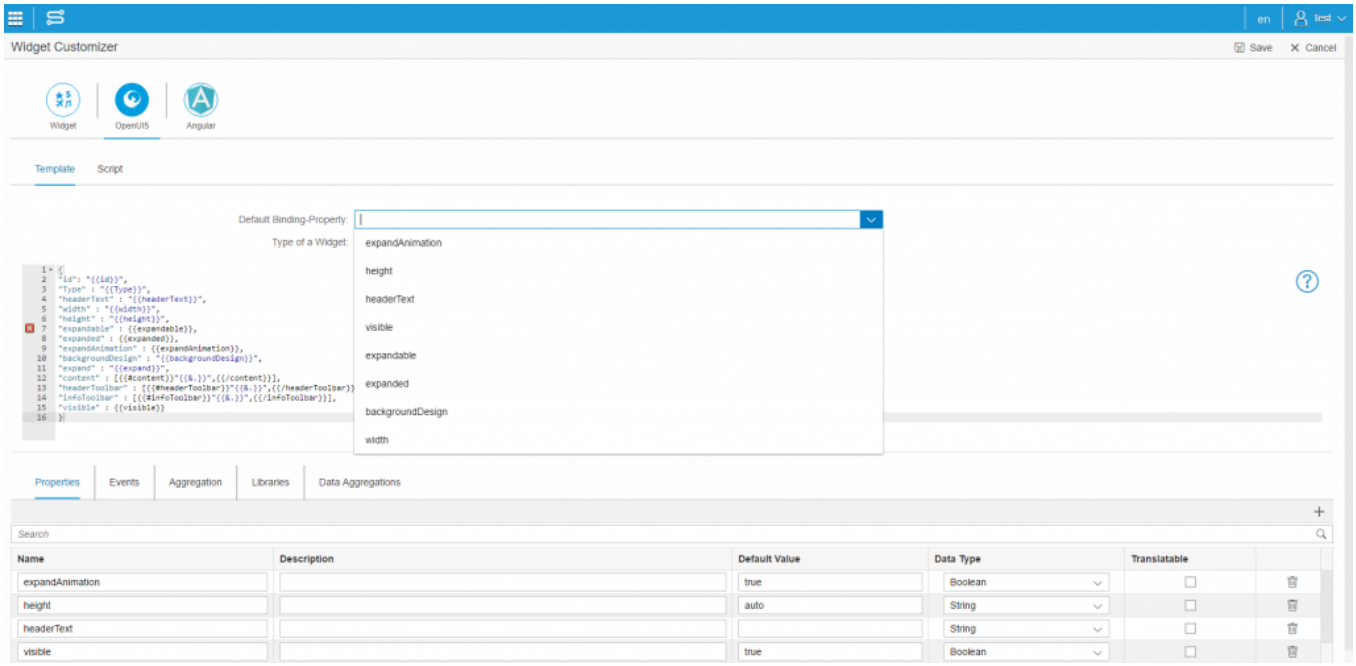
All attributes that are declared with a mustache value in the template, have to be declared in the properties / events / etc. area below as well, so you can work with them in the UI Designer later in the process. You can use constant values instead of mustache e.g. if you don‘t want a property to be editable.

- Properties:
  Fill in the name (your template placeholderName), a description (optional), the default value and the data type (as written in the API reference). If a property should be translatable, you have to check it here.
- Events:
  Simply put the name (again the template placeholderName) on the list.
- Aggregation:
  Transfer your template placeholderName and the content type (API reference).
  If your aggregation shall be able to contain more than just one control, check the “Multiple” checkbox.



Step 5

If all properties are listed, you can set the Default Binding-Property which is the prioritized widget property used in the edit mode of a user story (Process Designer).



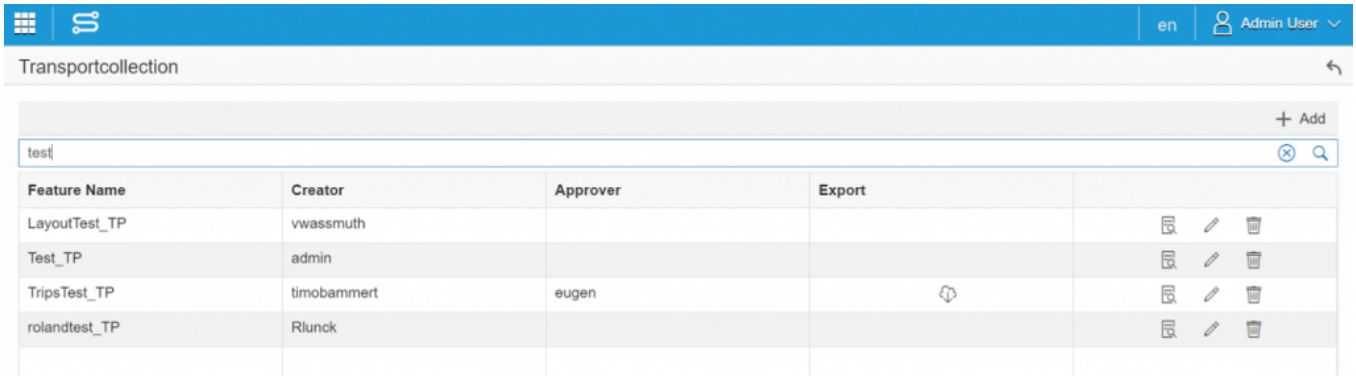
End

After hitting the “Save” button, you’ve successfully created a Panel Widget for your application.

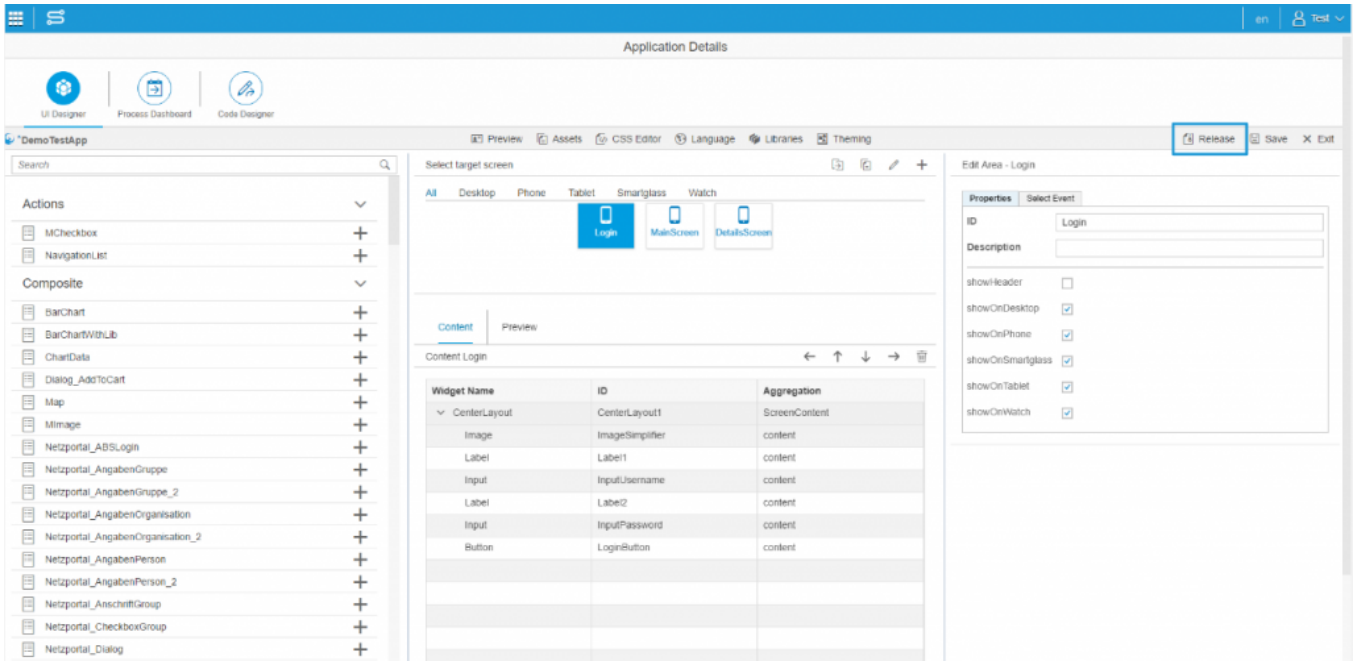
Create and Edit Transports

There are two different options to create a Transport:

- Adding an individual Transport via the “Transportcollection” tile



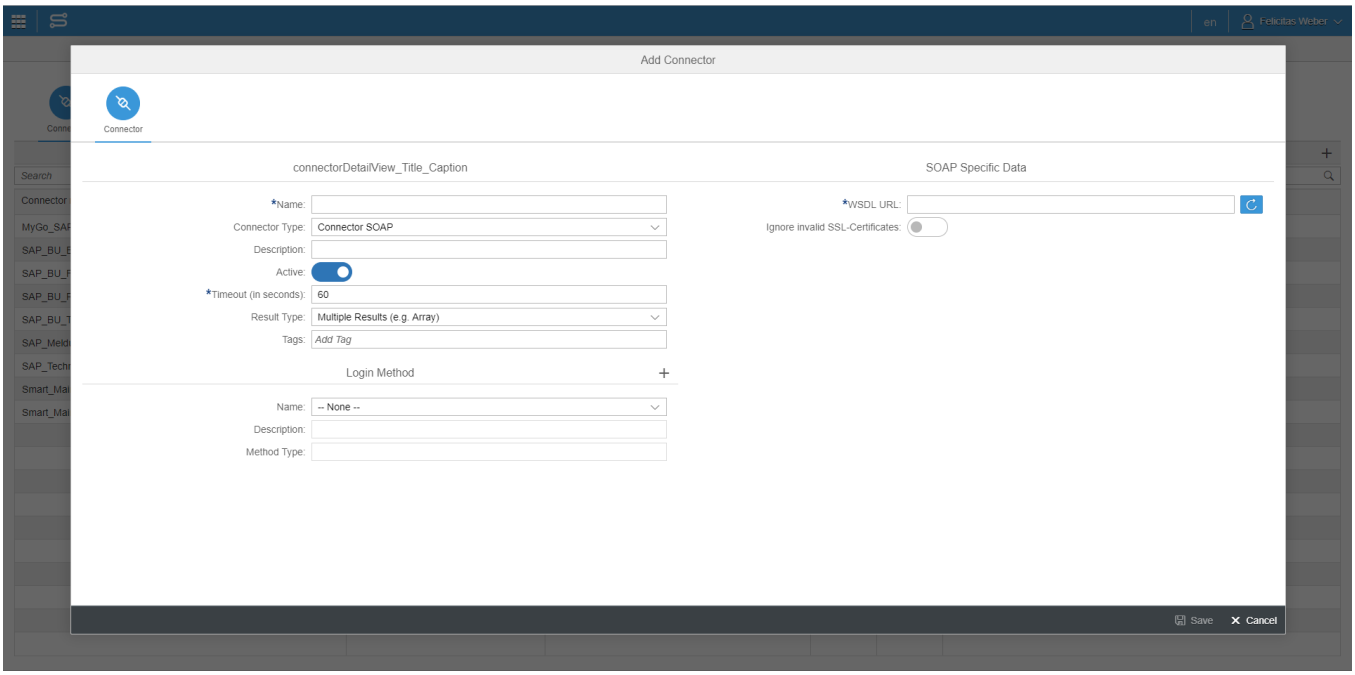
- Releasing your app in the UI Designer



Create and Manage Connectors

Create a Connector

To create a new connector, click on the plus icon on the upper right corner. It opens a new pop up where you can enter the required and optional information.



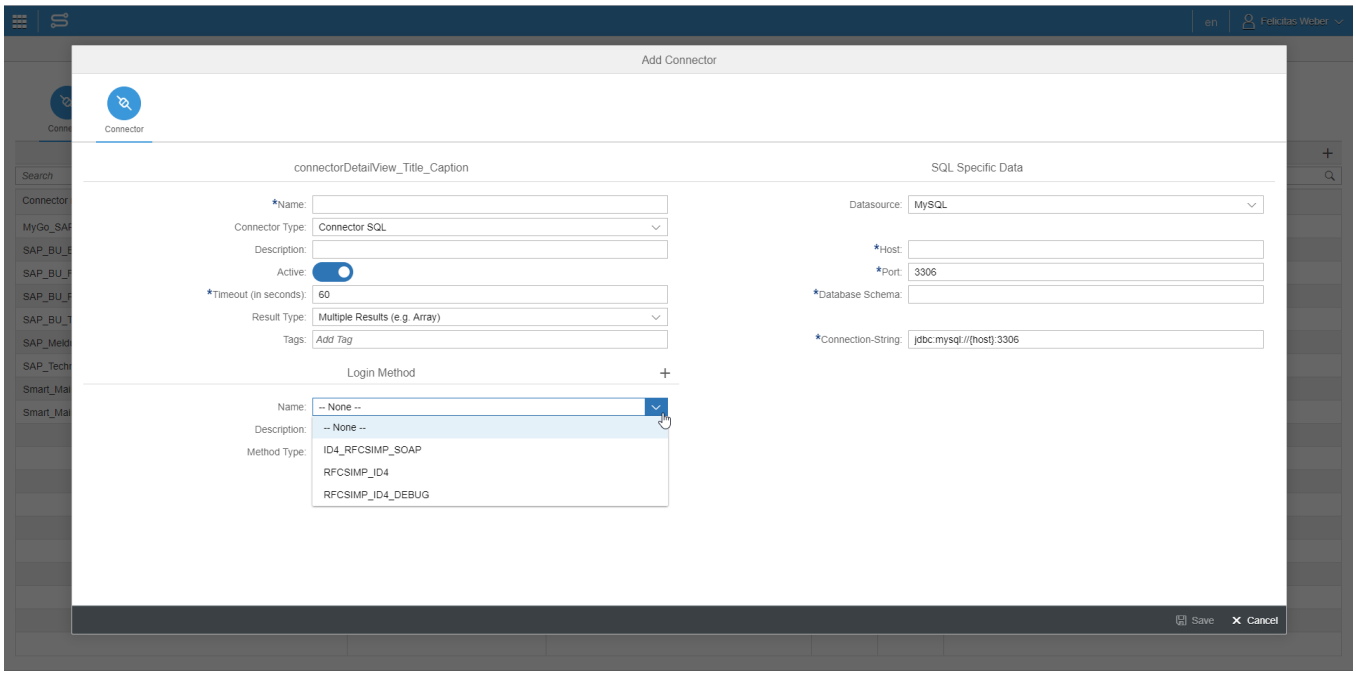
Connector Type

Name	The connector needs a unique name.
Connector Type	Set the technical protocol of the interface – see chapter Connector Types.
Description	Add a description.
Active	Set the connector active. You can see within the overview which connector is active.
Timeout time (in seconds)	Set the time in seconds until the connector request will run. After the set timeout, the request will be discontinued.
Result Type	Type of result from the connector request. It can either be a single result or multiple results (e.g. Array).
Tags	You can add tags to your connector (e.g. the name of a project).

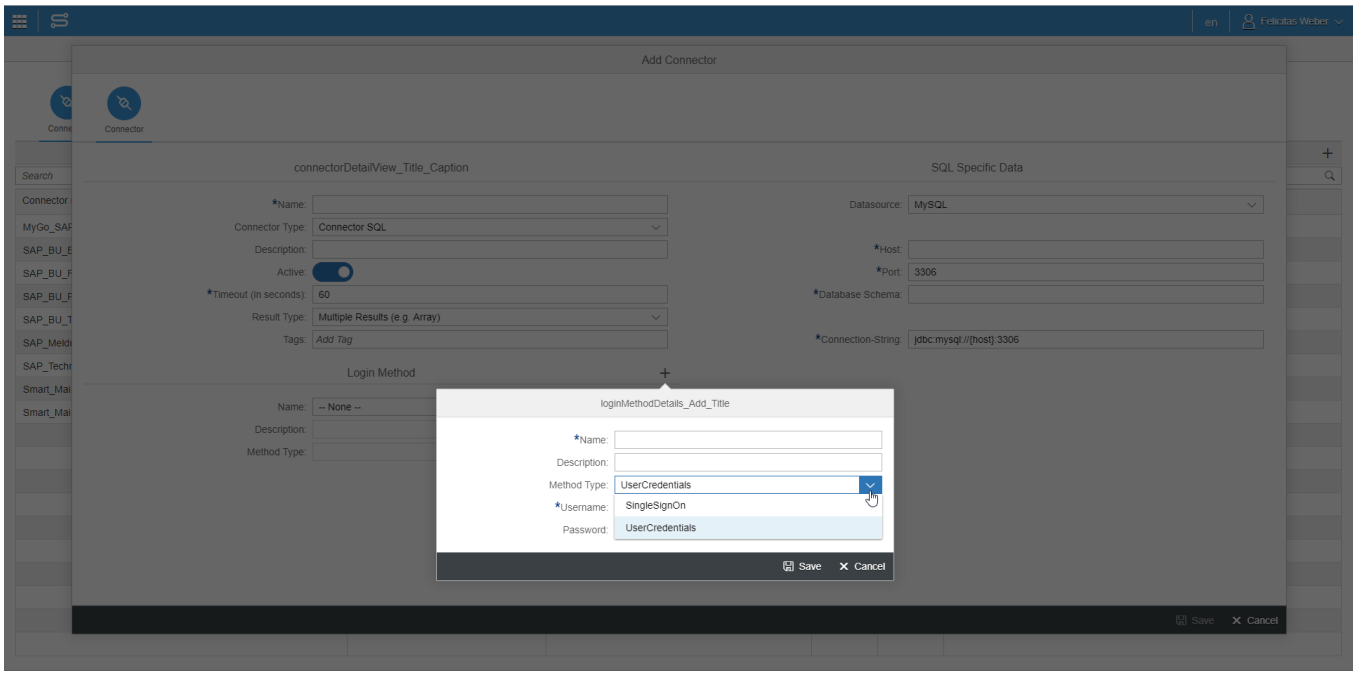
Login Method

You can add or select the login method for the specific backend systems. To select an existing login method, click on the corresponding field. It opens a drop down where you can select it.

If you want to create a new one, you can choose between using UserCredentials and SingleSignOn.

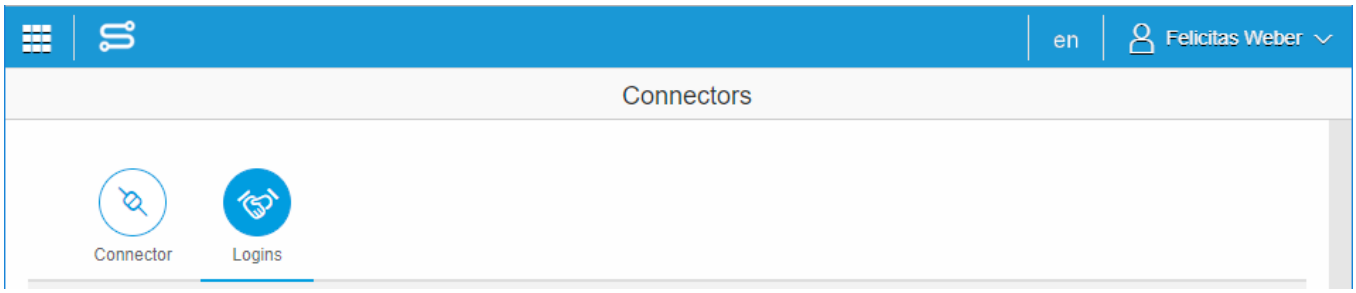


select an existing login method



create a new login method

If you'd like to get an overview on your existing login methods and manage them, click on the "Logins" tab in the connector overview.

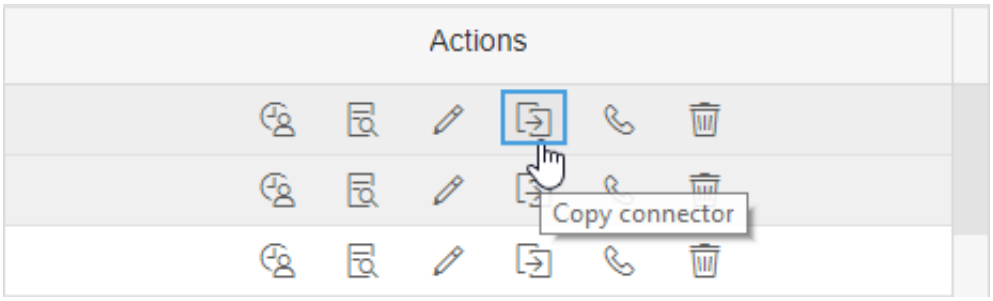


Connector Details

On the right side you can add data to your connector. Each connector has specific details that depend on the properties of the communication protocol. Read more on the following pages.

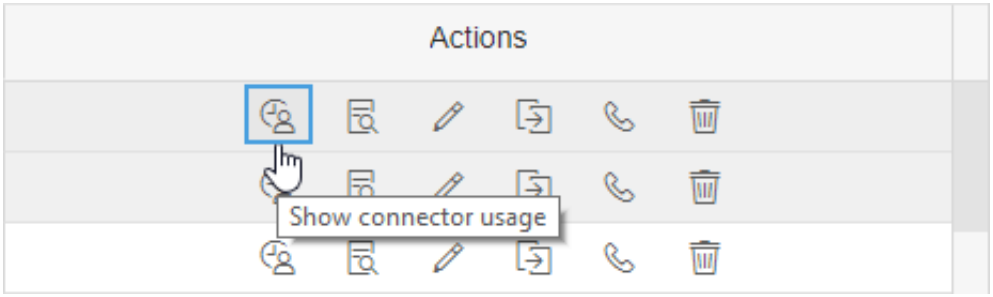
Copy a Connector

You can also copy an existing Connector. The complete configuration of the Connector, including its Connector Calls, is copied and created with the duplicate.

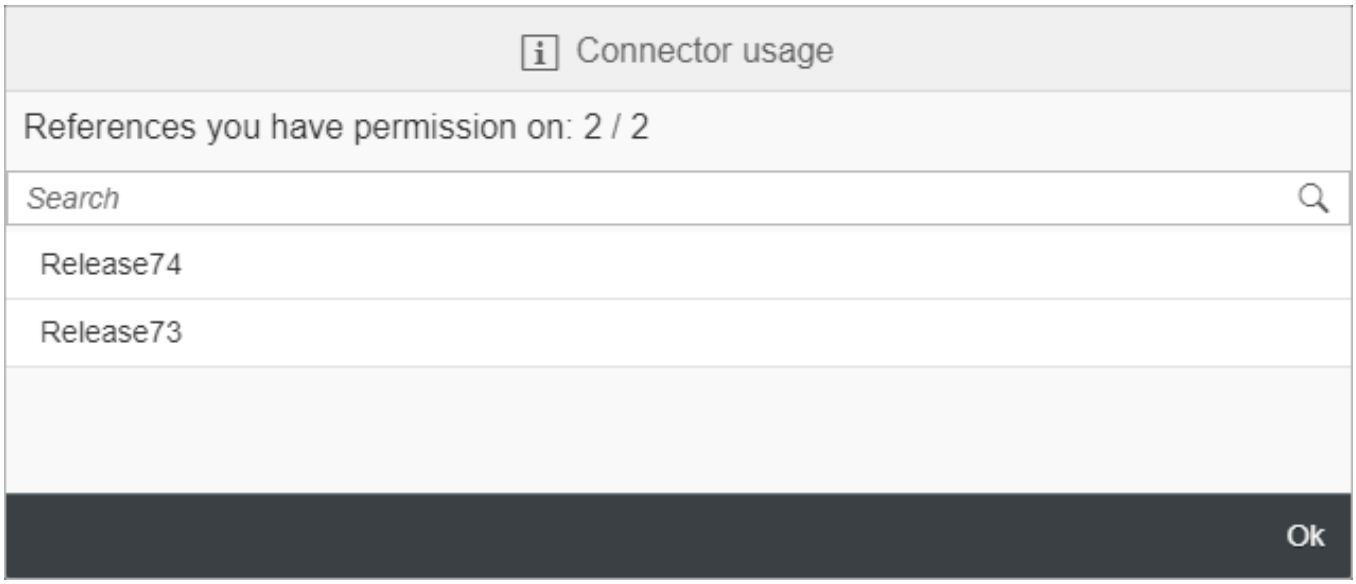


Usage of Connector

You can see which applications are using the Connector. For that click on the appropriate icon within the Connector overview underneath ‘Actions’.

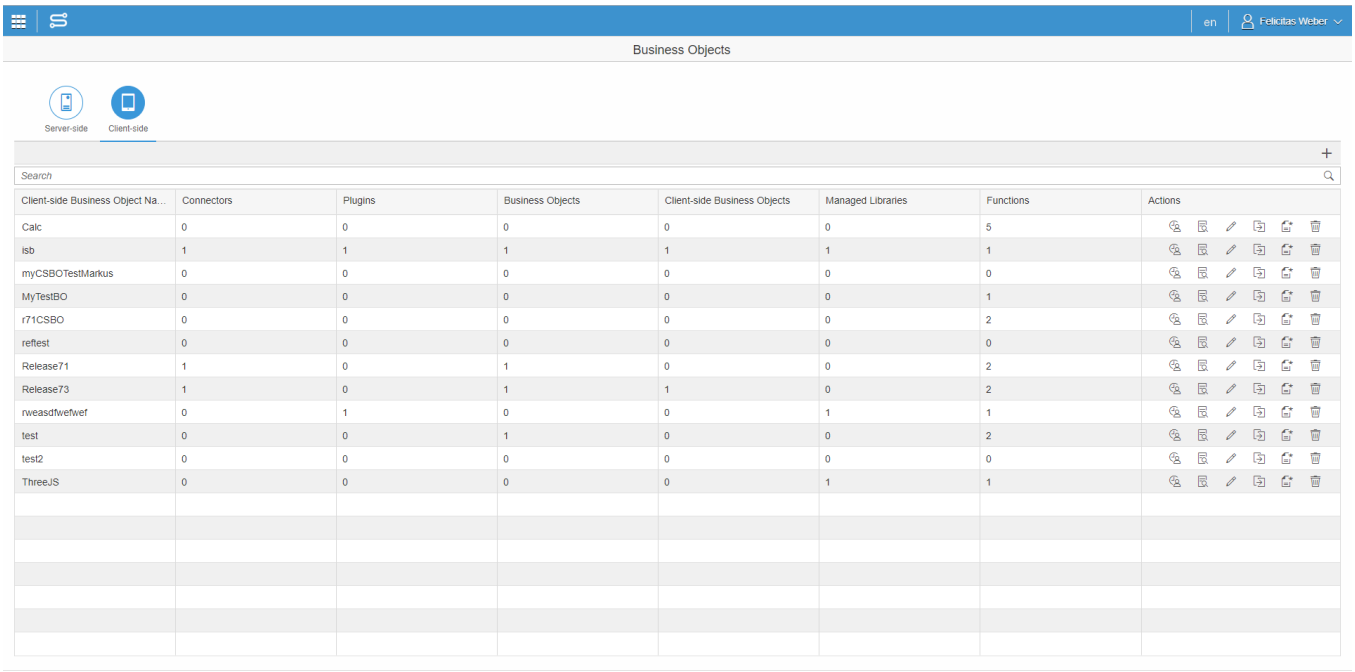


It opens a new popup that displays all applications that uses the connector:


Connector usage
References you have permission on: 2 / 2
Search
Release74
Release73
Ok

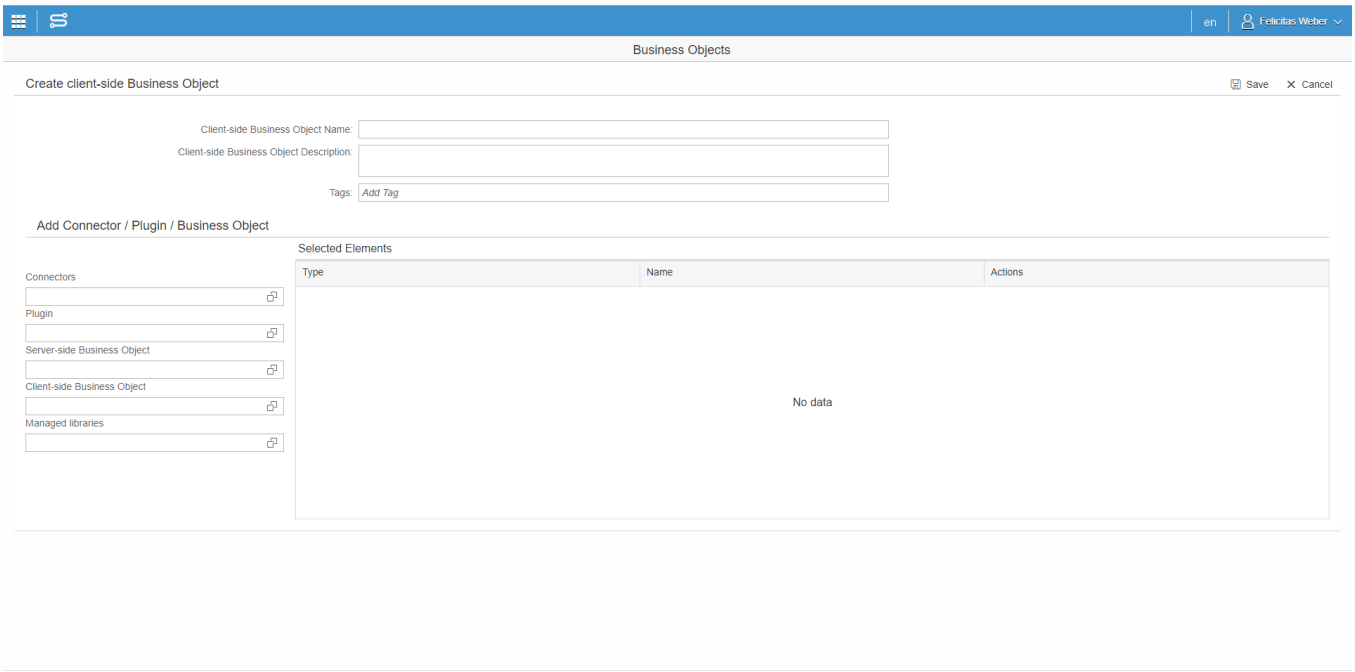
Create client-side Business Object

Business Objects are managed under the module “Business Objects”. The main screen lists all existing Business Objects in table form. On the top left you can switch between server-side and client-side Business Objects.



Client-side Business Object Na...	Connectors	Plugins	Business Objects	Client-side Business Objects	Managed Libraries	Functions	Actions
Calc	0	0	0	0	0	5	
isb	1	1	1	1	1	1	
myCSBOTestMarkus	0	0	0	0	0	0	
MyTestBO	0	0	0	0	0	1	
r71CSBO	0	0	0	0	0	2	
reftest	0	0	0	0	0	0	
Release71	1	0	1	0	0	2	
Release73	1	0	1	1	0	2	
rweasdfwefwef	0	1	0	0	1	1	
test	0	0	1	0	0	2	
test2	0	0	0	0	0	0	
ThreeJS	0	0	0	0	1	1	

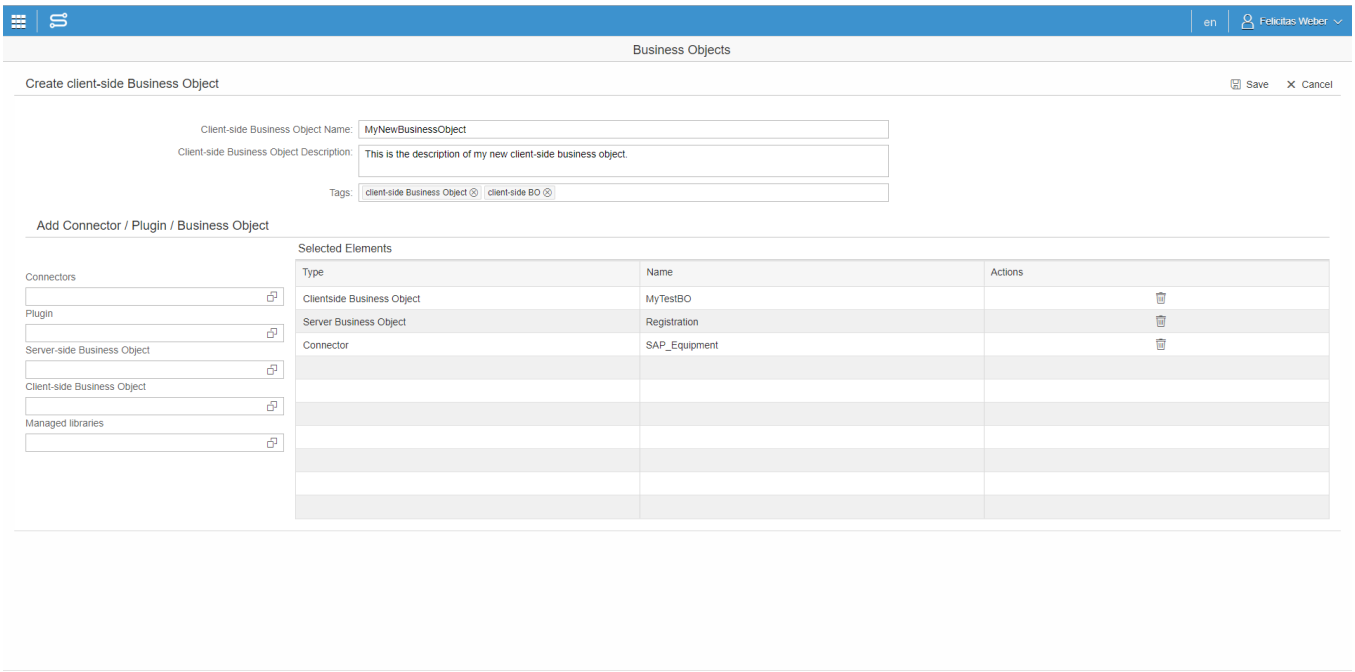
Press “+” in the upper right corner to create a new one from scratch. This fires up the following screen:



create client-side business object

First, choose a name for your client-side Business Object, e.g. “MyNewBusinessObject” and set a description (optional). Add some tags, so you can search in the overviews and the UI Dsigner by the tags.

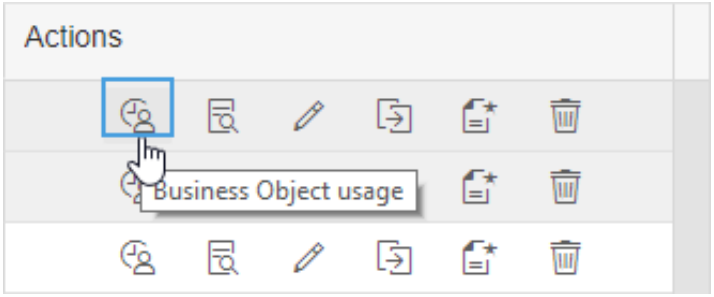
You may then select any Connector, Plugin, Server-side Business Object, Client-side Business Object or managed libraries on the left side. It opens a dialog where you can select it. Each selected item appears in the list below, from where you might also remove it again by clicking the delete icon underneath ‘Actions’.



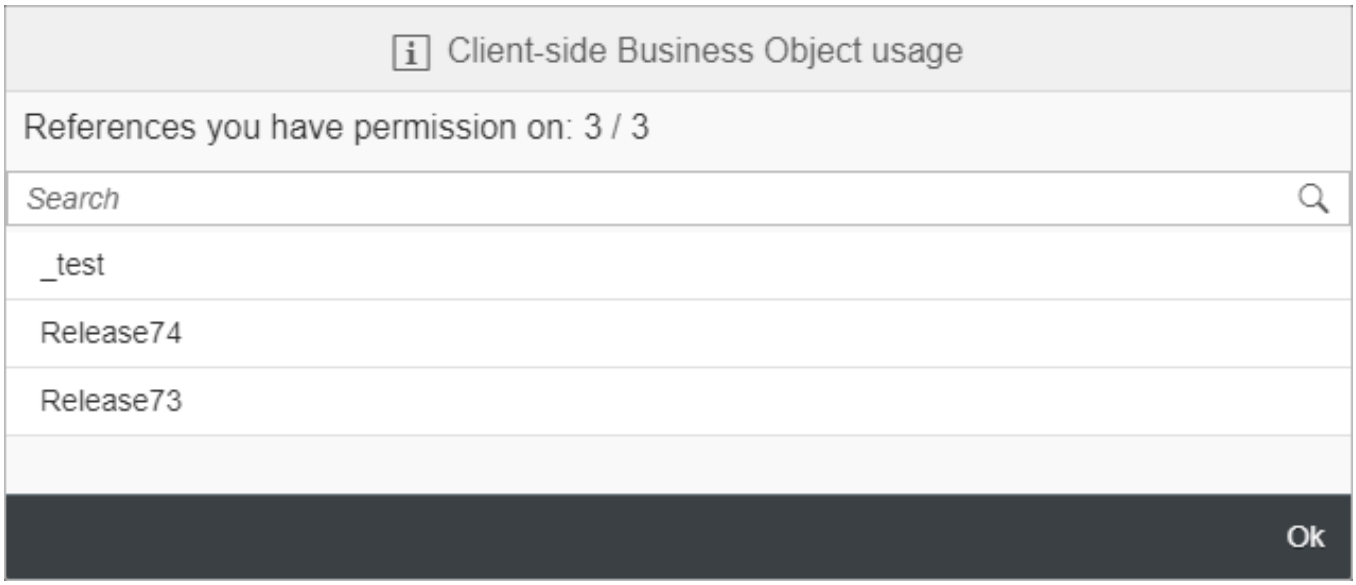
When you're done, leave the screen by hitting the "Save" button and return to the overview page. Your new Business Object appears in the table.

Usage of Business Objects

You can see which applications or interfaces are using the Business Object. For that click on the appropriate icon within the Business Object overview underneath 'Actions'.



It opens a new popup that displays all used applications:



Accessing input and output parameters of client-side Business Objects

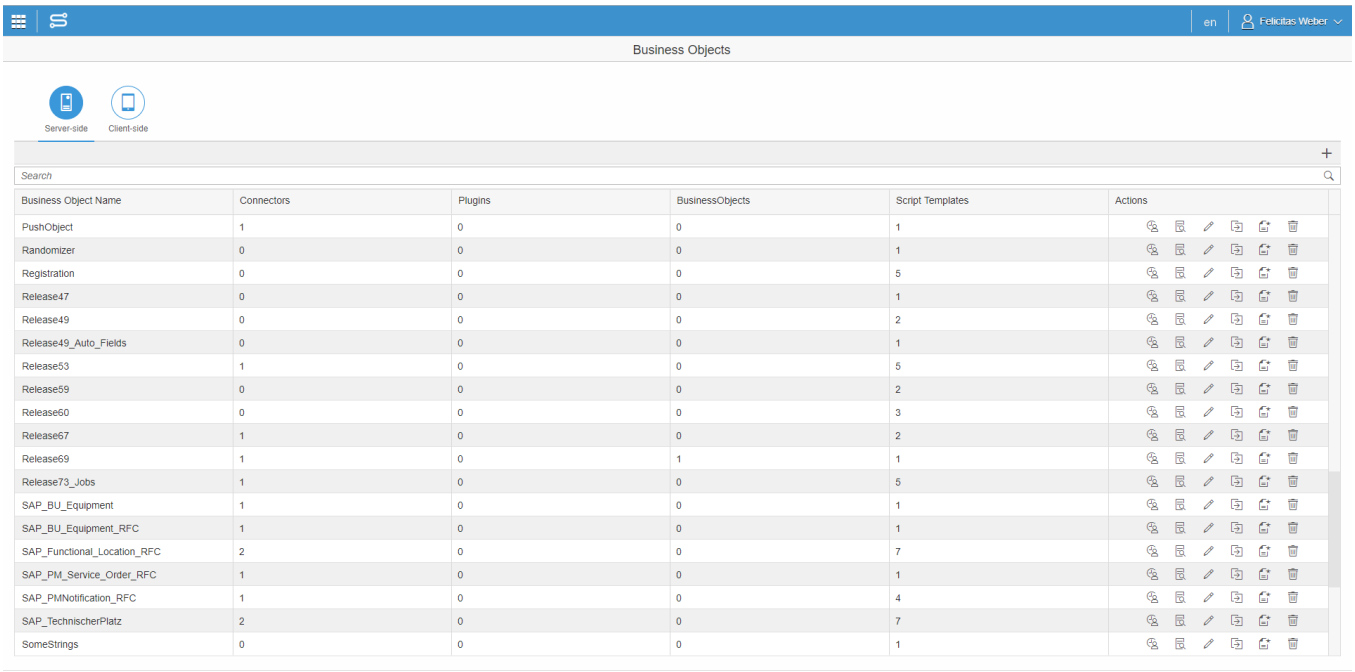
You can access your input parameters via oPayload.<myInputParameter>.

To use the output parameters you have to return an object that has your parameters as properties. E.g.

```
return {
      myOutputParameter : myOutputValue
}
```

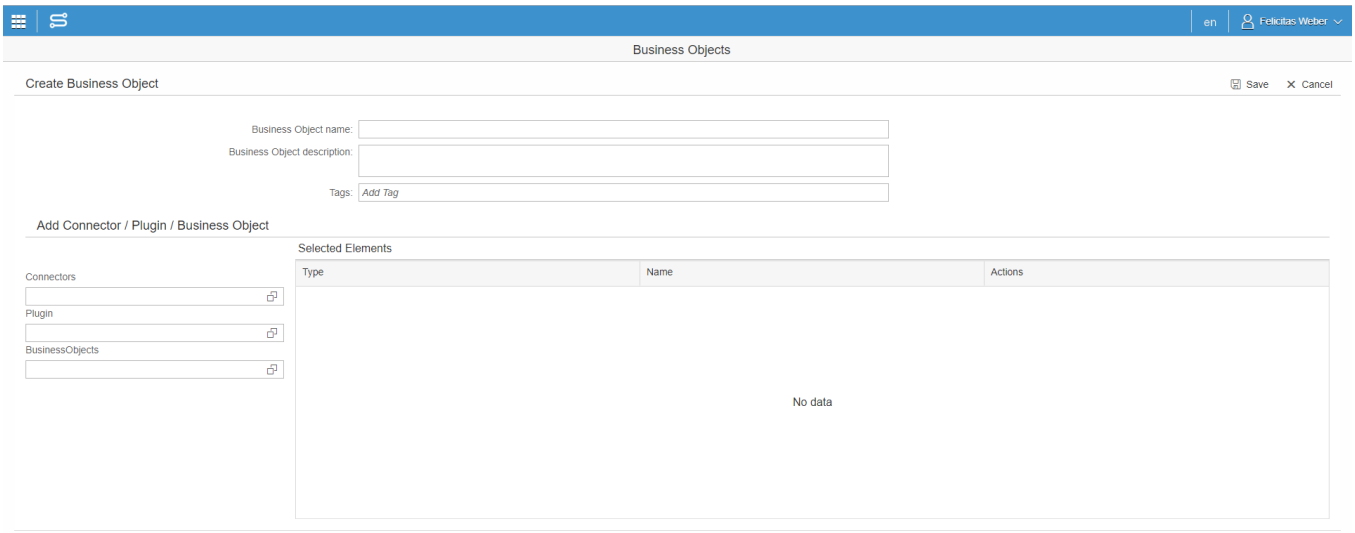
Create server-side Business Objects

Business Objects are managed under the module “Business Objects”. The main screen lists all existing Business Objects in table form. On the top left you can switch between server-side and client-side Business Objects.



Business Object Name	Connectors	Plugins	BusinessObjects	Script Templates	Actions
PushObject	1	0	0	1	
Randomizer	0	0	0	1	
Registration	0	0	0	5	
Release47	0	0	0	1	
Release49	0	0	0	2	
Release49_Auto_Fields	0	0	0	1	
Release53	1	0	0	5	
Release59	0	0	0	2	
Release60	0	0	0	3	
Release67	1	0	0	2	
Release69	1	0	1	1	
Release73_Jobs	1	0	0	5	
SAP_BU_Equipment	1	0	0	1	
SAP_BU_Equipment_RFC	1	0	0	1	
SAP_Functional_Location_RFC	2	0	0	7	
SAP_PM_Service_Order_RFC	1	0	0	1	
SAP_PMNotification_RFC	1	0	0	4	
SAP_TechnischerPlatz	2	0	0	7	
SomeStrings	0	0	0	1	

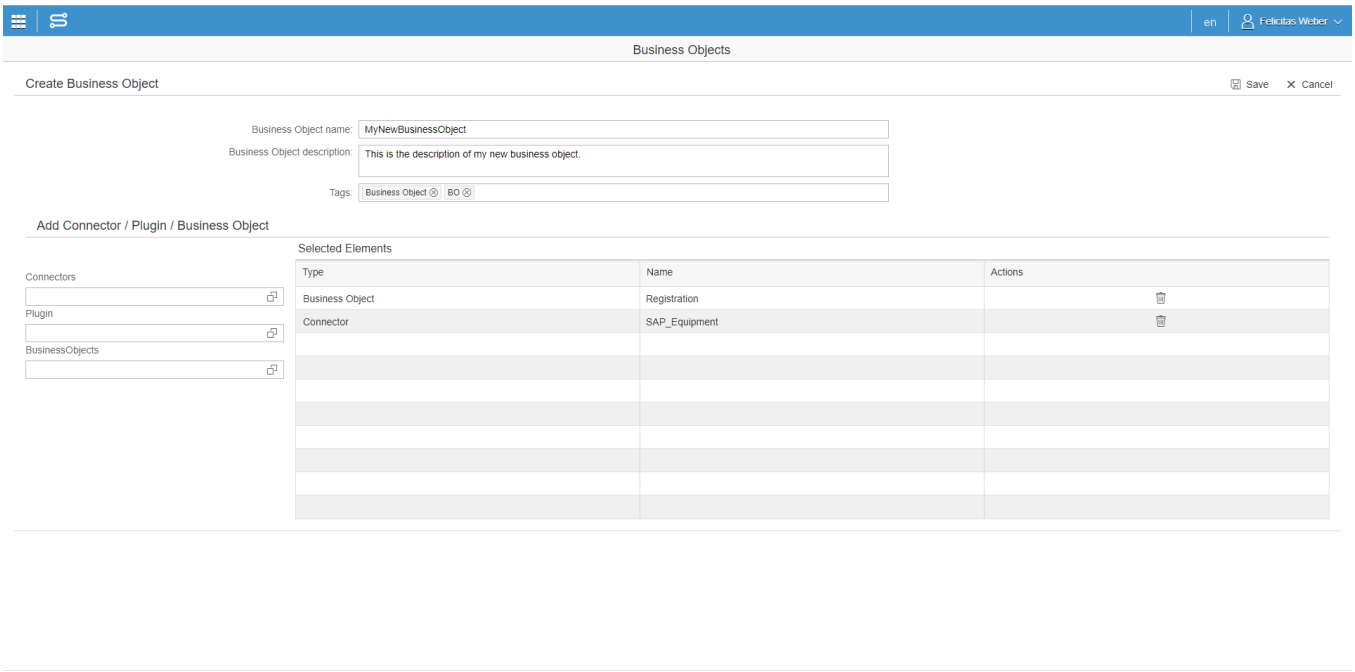
Press “+” in the upper right corner to create a new one from scratch. This fires up the following screen:



create server-side business object

First, choose a name for your Business Object, e.g. “MyNewBusinessObject” and set a description (optional). Add some tags, so you can search in the overviews and the UI Dsigner by the tags.

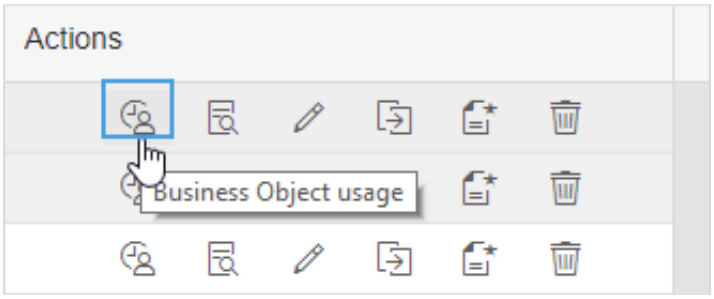
You may then select any Connector, Plugin or other Business Object you want to refer on the left side. It opens a dialog where you can select it. Each selected item appears in the list below, from where you might also remove it again by clicking the delete icon underneath ‚Actions‘.



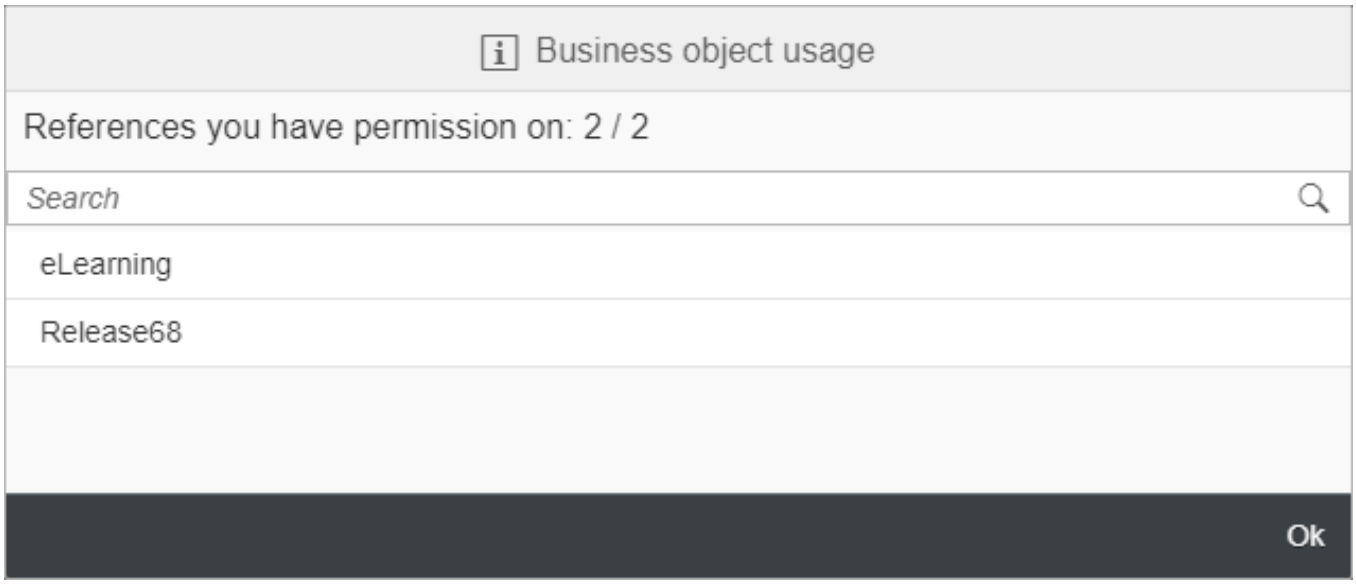
When you’re done, leave the screen by hitting the “Save” button and return to the overview page. Your new Business Object appears in the table.

Usage of Business Objects

You can see which applications or interfaces are using the Business Object. For that click on the appropriate icon within the Business Object overview underneath ‘Actions’.

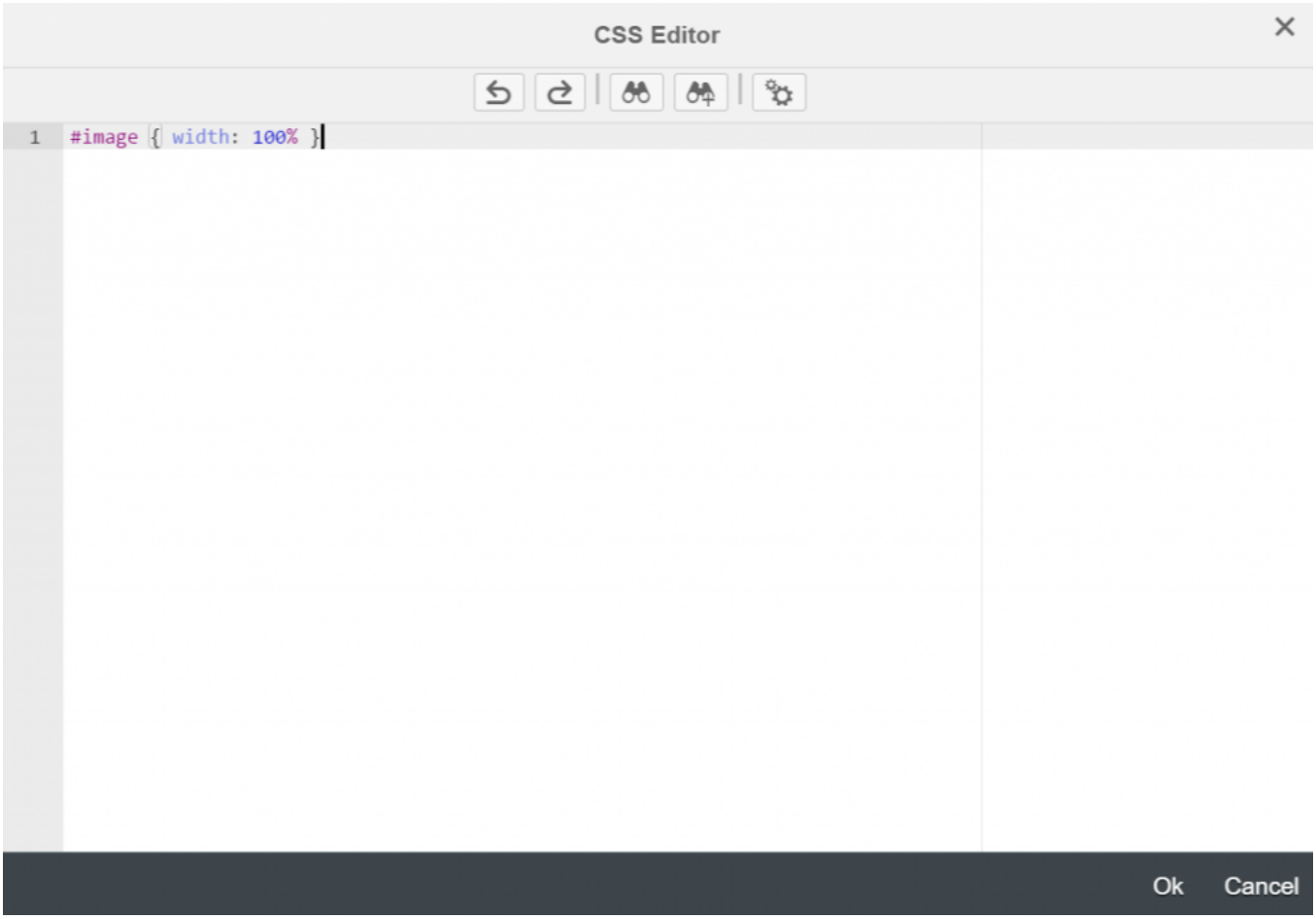


It opens a new popup that displays all used applications:


Business object usage
References you have permission on: 2 / 2
Search
eLearning
Release68
Ok

CSS Editor

To design your application, you can use CSS for styling with standard web cascading style sheets.



If you would like to learn more about CSS, take a look at the tutorials of the w3 schools.

After changing css properties you have to reload your application in the preview tab

To Do so please use the following shortcuts

Browser	**Keyboard Shortcut**
Chrome on Windows	Ctrl and press F5
Chrome on Mac	? Cmd and ? Shift and then press R.
Firefox on Windows	Ctrl and press F5
Firefox on Mac	? Cmd and ? Shift and then press R

CSV Connector Calls

Go to CSV Connector Details for more information about the CSV Connector.

A CSV Connector can be configured in 3 different modes:

***** *READ*: The connector can only read from the specified CSV file *path*, no write operations are permitted.

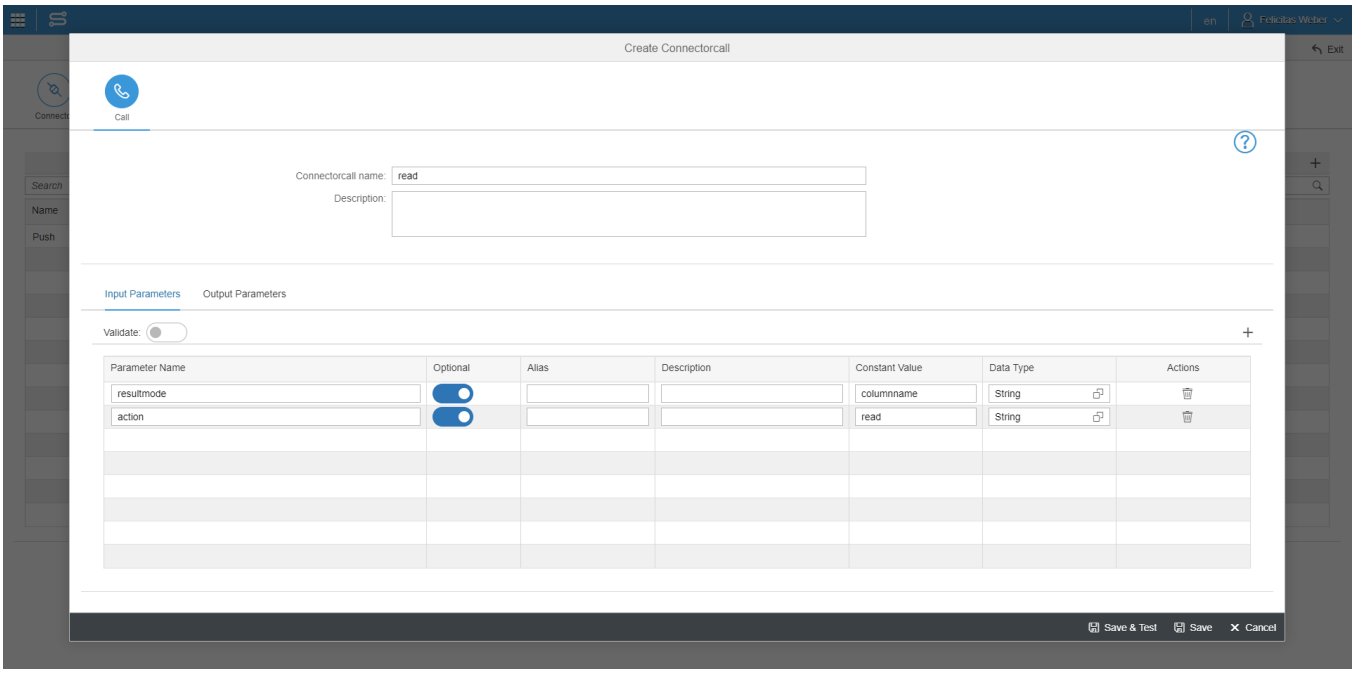
***** *WRITE*: The connector can only write to the CSV file, but not read from it.

***** *READ/WRITE:* The connector can read from the file and also write to it.

Currently, only the READ operation is implemented!

READ

The CSV Connector Call for a READ operation requires 2 Input parameters: “*action*” and “*resultmode*“.



To execute a read operation, call the Connector with the parameter *"**action**"* and the constant value *"**read**"*.

Reading Connectors get the result as JSON array of arrays by default. There can be definied two "***resultmode***"parameter:

- "***columnnumber***" returns an array of JSON objects, where the key is "col0", "col1", … "colX" for the column.
- "***columnname***" returns an array of JSON objects where the key is the String taken from the header row (only available if *`headerInFirstLine`* is true – see CSV Connector Details).

The Connector returns everything if you use "/" as Output parameter.

CSV Connector Details

Path

Filepath and Filename to local .CSV File that should be written, relative to the current working directory of the application server. It is recommended to give an absolute path, so it doesn't matter which directory is set as "Current Working Directory" from the appserver start script.

Delimiter

Delimiter of the columns that separates the values like comma or semicolon. This must be exactly one character, more than one character is not supported by the library.
******In order to use the tabulator character, the expression '*\t*' can be used in the Admin UI.****** If more than one character is specified, all but the first character will be discarded.

Charset

Character encoding used to read/write the file. If a charset is used that is unknown to the application server JVM, all read/write operations will fail.

Mode

Operation Mode of the Connector, either READ, WRITE or READ/WRITE – the CSV Connector can currently only read the referenced csv file.

Attention!

Please make sure that the Simplifier application Server has the right permissions to access and read the file on the operating system level.

Header

Activate the checkbox if the CSV File has a header in the first row.

Quote all Items

Activate the checkbox if all items should be quoted in terms of strings (“). Otherwise only values which contain the delimiter are put in quotes. This setting is irgnored when reading.

Go to CSV Connector Calls to configure the corresponding Calls.

Custom Extension of Simplifier SAP Connector

It’s also possible to extend our interface with implicit enhancements

- at the end of all the programs (Includes, Reports, Function pool, Module pool, etc.), after the last statement
- at the beginning and end of all FORM subroutines
- at the end of all Function Modules
- at the end of all visibility areas (public, protected and private) of local class

This feature is official documented on the sap wiki.

The second option is to use BADIS to enhance our Namespace.

Minimum Requirements for the Simplifier SAP Connector

To install the Simplifier SAP Connector, your SAP system should meet at least the following requirements:

- SAP ERP 6.0 EHP 7
- SAP ABAP 7.4+
- SAP Base 7.40
- SAP IS-U Release 6.17 EHP 7 (optional only for IS-U interface)

Data Centers of Simplifier Cloud

The Simplifier Cloud is hosted in data centers of T-Systems Germany.

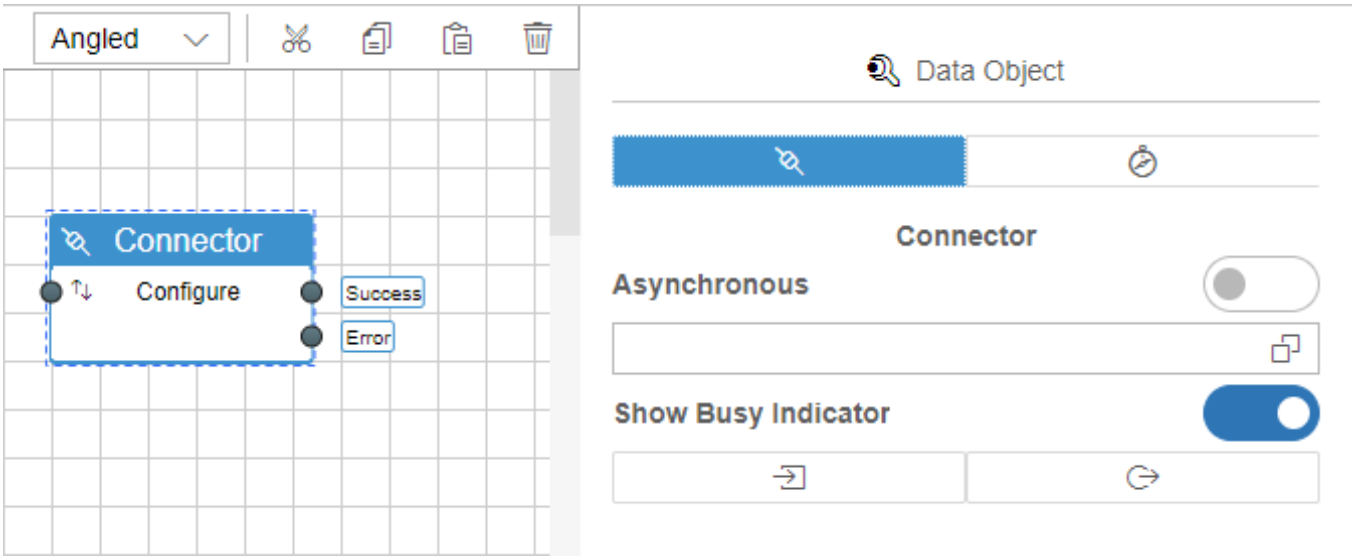
The locations of the twin-core data centers are:

Lübecker Str. 2 39124 Magdeburg Germany	Am Schiens 10-11 39221 Bördel and/Biere Germany

[vc_gmaps link="#E-8_JTNDaWZyYW1lJTIwc3JjJTNEJTIyaHR0cHMlM0ElMkYlMkZ3d3cuZ29vZ2xlLmNvbSUyRm1hcHMlMkZkJTJGdSUyRjAlMkZlbWJlZCUzRm1pZCUzRDEwYWxiVVlCVTh1S0szRUhYa3lDb1JyeXltZDNXenRtdiUyMiUyMHdpZHRoJTNEJTIyMzgwJTIyJTIwaGVpZ2h0JTNEJTIyNDgwJTIyJTNFJTNDJTJGaWZyYW1lJTNF"]

Data Object

Data Objects represent data sources and destinations, which can be triggered for execution. You can choose either a predefined Connector (you can activate it to asychronous) or Business Object.

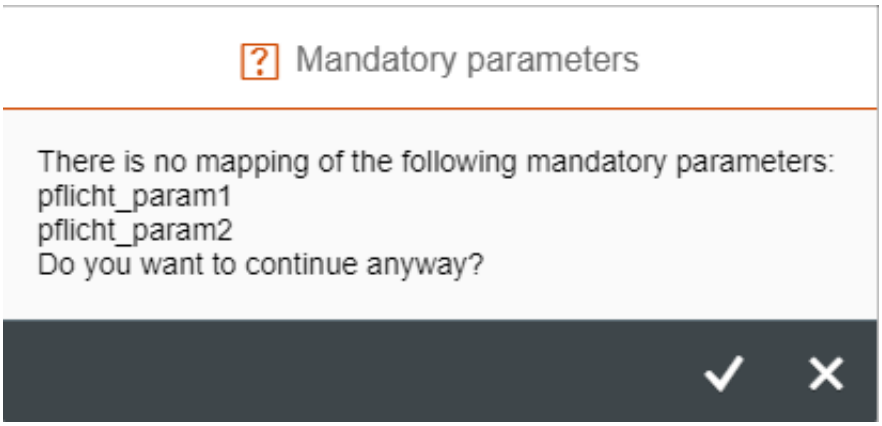


Function	Description
Asynchronous	If you select this option, the value helper assistant only offers asynchronous connectors.
Value Helper	If you open the value helper, an assistant opens that guide you to your connector.
Show Busy Indicator	You have the possibility to configure if the UI is blocked by the busy indicator, or can configure which element on your screen should be blocked by it.
Input Mapping	You can map variables, autofields, widget properties and constants to the input parameter of your connector.
Output Mapping	You can map the output parameter of your connector to variables and properties.

Validation of mandatory Parameters

All configured parameters has an indicator if it’s mandatory.

If you haven’t map all input parameters of a data object that has been marked as mandatory, the following warning will appear:


Mandatory parameters
There is no mapping of the following mandatory parameters:
pflicht_param1
pflicht_param2
Do you want to continue anyway?

Data Types

The module “Data Types” is the central way to define different types of data, structures and collection of Data Types and their validation rules.

Data Types are a way to ensure data are sent and received in the right type format to and from the backend systems.
With this feature, you can define data definitions to validate your data with client and server-side validation to prevent security issues and backend saving problems due to wrong data formats or hacker attacks.

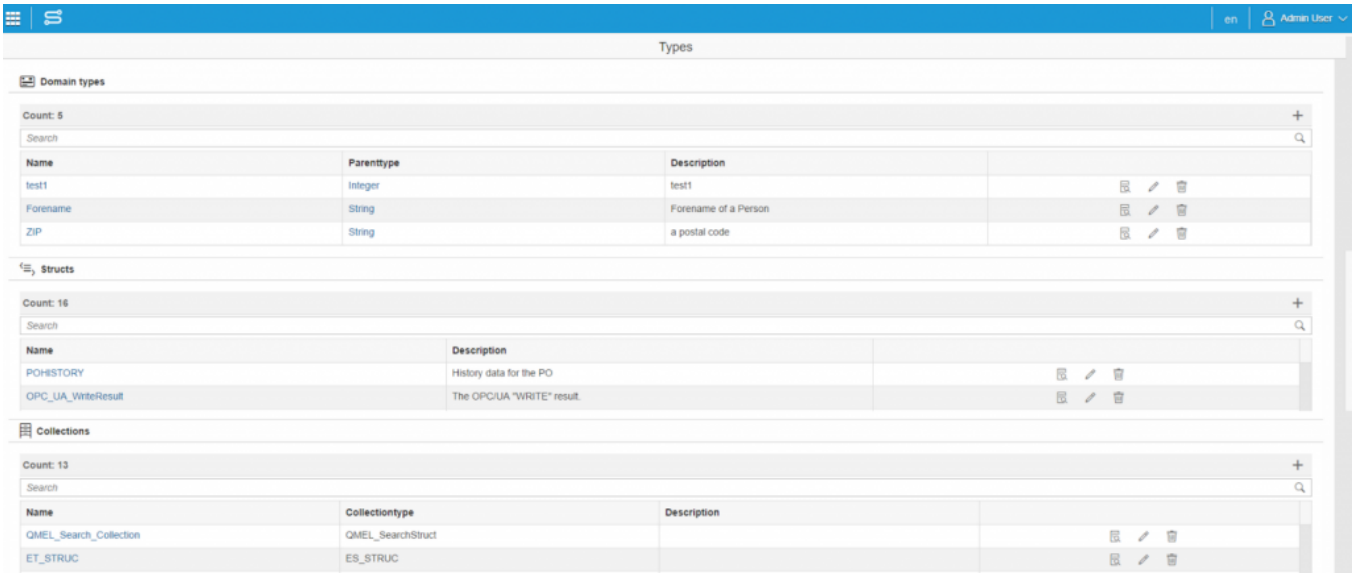
Data Types can be assigned to widgets and connector calls to validate input and output data.

There are six **Base Data Types** defined in the Simplifier:

Date	Date Format
String	Characters, numbers and any other symbols from the Unicode Character Set
Boolean	True or False
Integer	Positive and negative numbers like -2, -1, 0, 1, 2 …
Float	Numbers with precisions like 2,503
Any	Accept all kind of Data Types even heterogeneous Arrays.

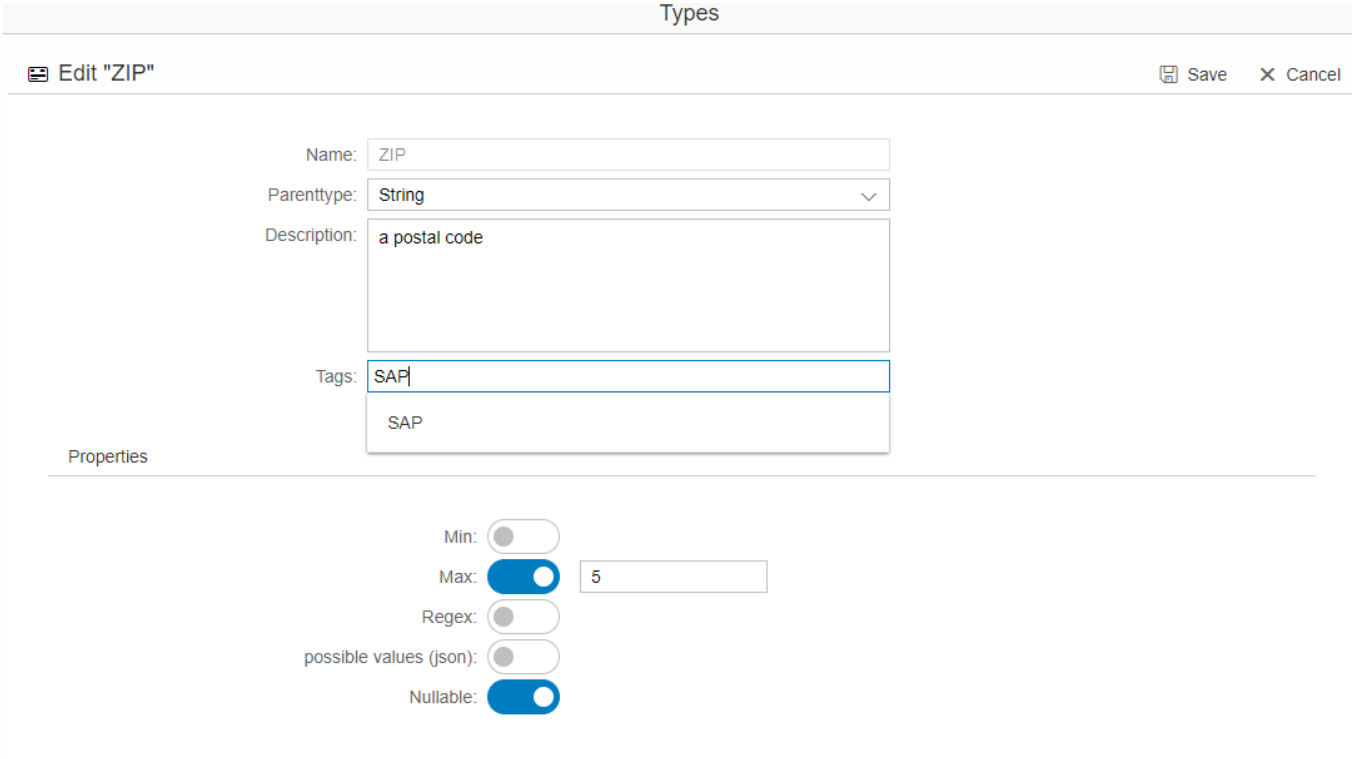
With the “Data Types” tile, you are able to enhance the Base Types and define your own logic.
The new Data Types are split into three different types:

- Domain types
- Structs
- Collections



You can assign tags to Data Types to find them easily in the searchbar. Already existing tags are suggested including [tags of

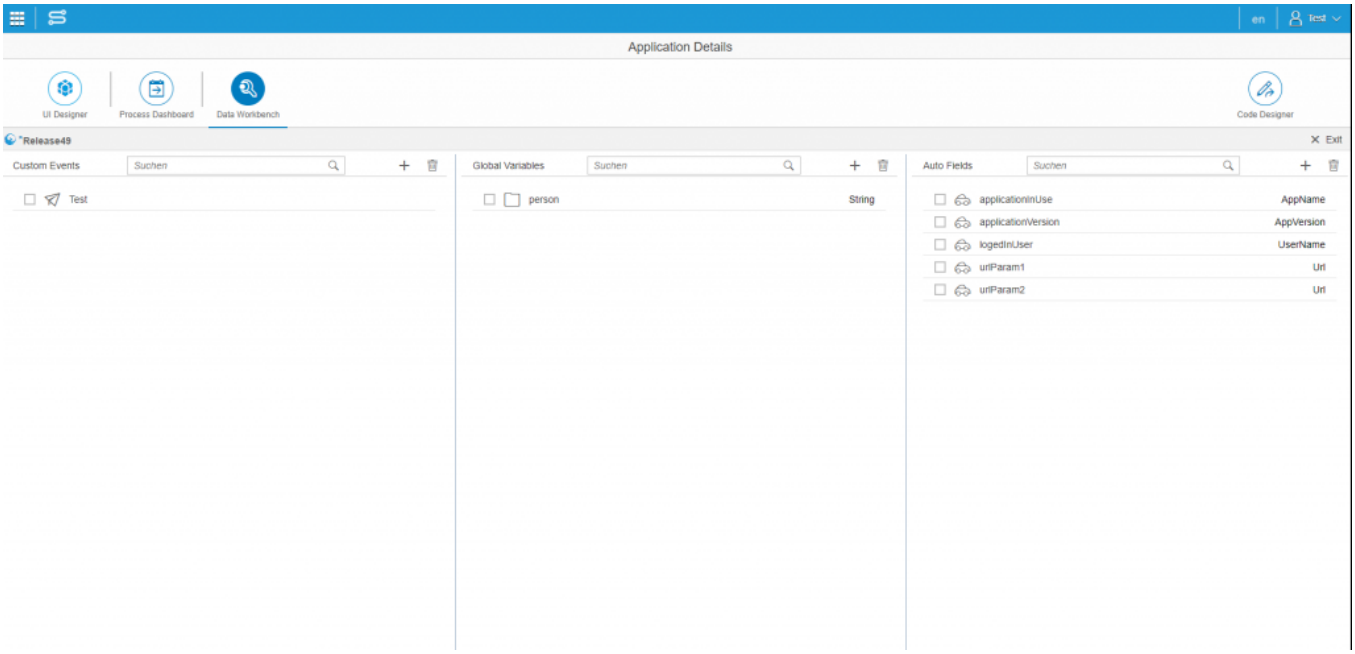
widgets.



Tags for Data Types

Data Workbench

Within the Data Workbench, you can administer global Events, Variables and Auto Fields that you want to use cross-functional in the User Stories.



PLEASE NOTE

The Data Workbench is no longer available for applications using AngularJS.
If you build your application with Angular 2, there are no limitations.

Delete a PDF Template

Delete Template

To delete a template, you need the following parameter:

URL		/client/1.0/PLUGIN/pdfPlugin/adminTemplateDelete
Input-Parameter	**Name**	Template name
Output-Parameter	None	

Example input::

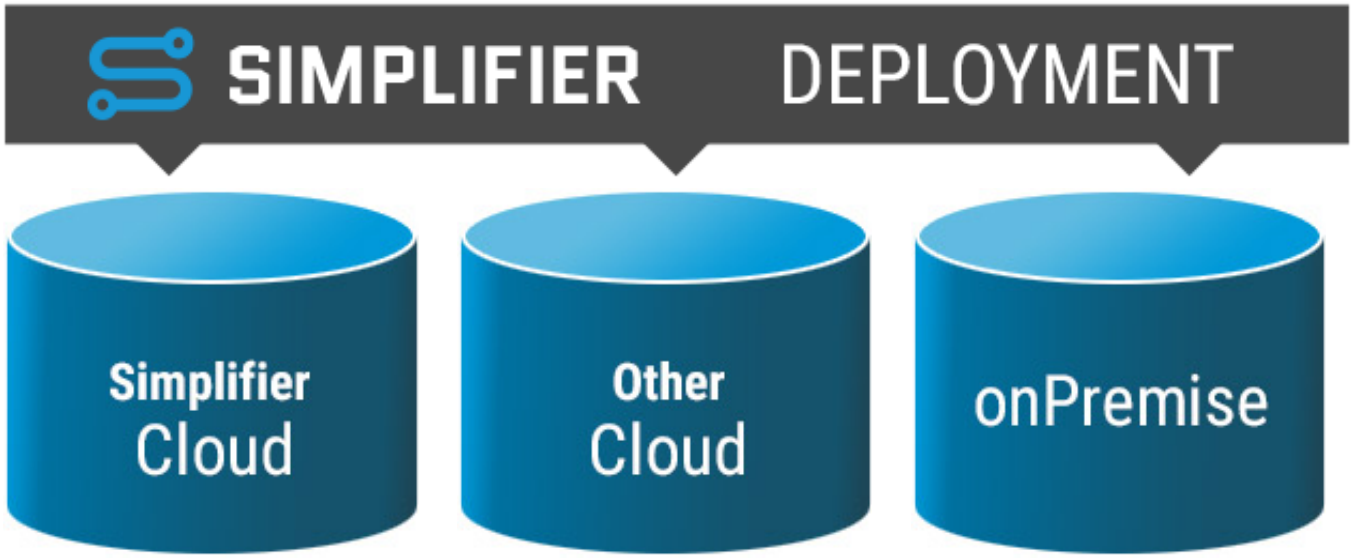
```
{     "name": "templatename" }
```

Example output:

```
{     "success": true }
```

Deployment & Installation Instructions

A Simplifier application can be deployed in different ways. You can deploy to your local machine for development and testing, you can deploy to the Simplifier cloud, Cloud Foundry-based platforms, Azure, AWS, SAP Cloud, or a server you configured yourself.



Cloud installations are hosted and maintained by iTiZZiMO. Each instance is reachable via a unique DNS name:

https://<instance-name>.simplifier.io

On-premise installations are hosted by our customers, on their own infrastructure. This scheme is especially useful if the Simplifier shall be integrated into a closed network infrastructure.

Checklist for Installation

For Installation of the Simplifier, the following persons and things are required:

- IT Security Officer
- Systemowner regarding Backend Interfacing
- Firewall Administrators
- Reverse Proxy Administrator
- IT Administrators
- SSL Certficates for HTTPS

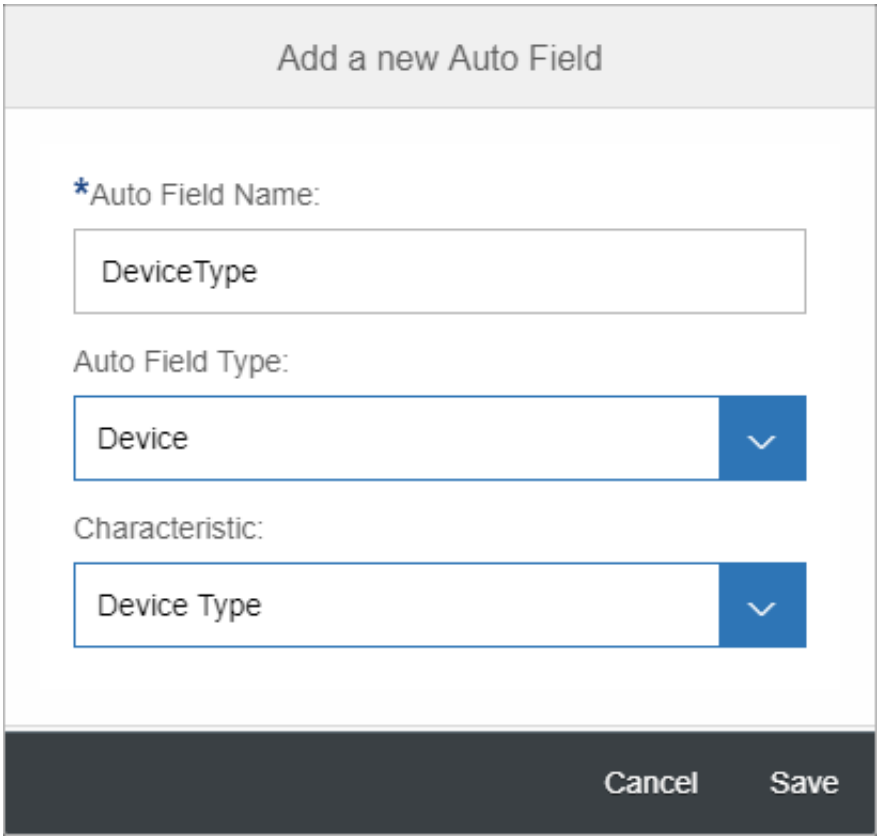
Use the following checkpoints for an successful installation

Checkpoint	Description
☑	System-Requirements are clear
☑	Domain-Name for Development, QA and Productive Systems are clear
☑	SSL Certifcates for all 3 Instances is available
☑	Firewallports 443 and 587 are open
	Backendsystem is reachable via Supported Protocols

Device Condition

The Simplifier is able to recognize the device type that is using the application, so you can assign different functionalities to it, e.g. different designed login screens for mobile or wearable devices.

At first you need to create a new autofield type with the autofield type “Device” and the characteristic “Device Type”. How to create autofields, you can see here.

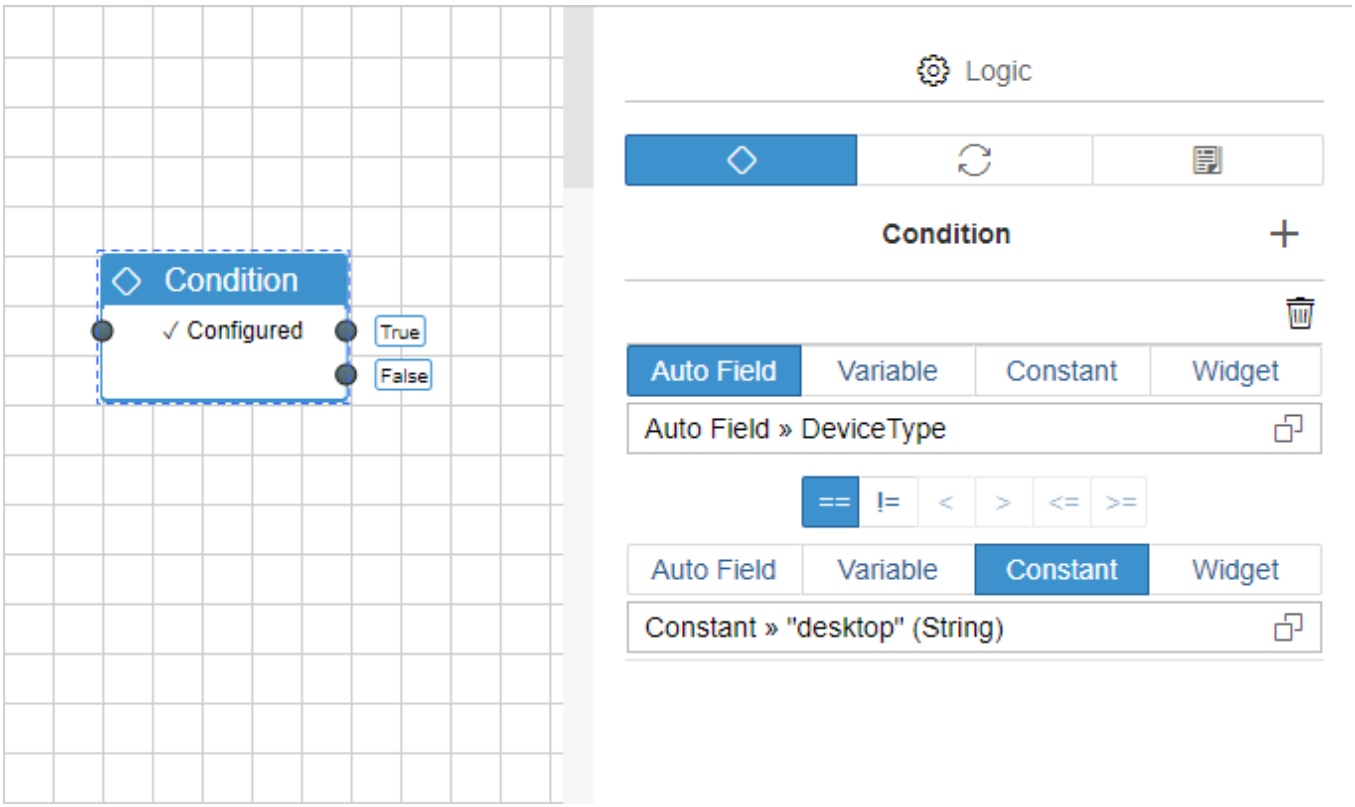


Within the Process Designer, you can refer to this autofield using a condition. Select the corresponding auto field and assign it to a constant (string).

Choose between:

- desktop
- phone
- smartglass
- tablet
- watch

Please pay attention to lower case!


Condition
✓ Configured
True
False
Logic
Condition
Auto Field
Variable
Constant
Widget
Auto Field » DeviceType
==
!=
<
>
<=
>=
Auto Field
Variable
Constant
Widget
Constant » "desktop" (String)

Docker Hub

The Simplifier is also available on Docker Hub.

Short Instructions

Create the directory which will host all external user-specific data:
$ mkdir -p /home/simplifier/data
$ export SIMPLIFIER_DIR=”/home/simplifier/data”

Install SSL certificates:
$ mkdir -p $SIMPLIFIER_DIR/certs
$ cp <certificate.pem> SIMPLIFIER_DIR/certs/default.crt
$ cp <keyfile.pem> SIMPLIFIER_DIR/certs/default.key

Run docker:
Alternative 1: with SSL/Certificates
$ docker run -d -v $SIMPLIFIER_DIR:/opt/simplifier/data \
-p 80:80 -p 443:443 -p 8090:8090 \
—name=simplifier itizzimo/simplifier

Alternative 2: without SSL/Certificates
$ docker run -d -v $SIMPLIFIER_DIR:/opt/simplifier/data \
-p 80:8080 -p 8090:8091 \
—name=simplifier itizzimo/simplifier

Docker Installation

Get Docker CE

Referenced to the official Docker instructions.

Note: This installation instructions is based on the example of the operating system Ubuntu 16.04 LTS.

SET UP THE REPOSITORY

Update the apt package index:

```
$ sudo apt-get update
```

Install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get install \

    apt-transport-https \

    ca-certificates \

    curl \

    software-properties-common
```

Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

```
$ sudo apt-key fingerprint 0EBFCD88

pub   4096R/0EBFCD88 2017-02-22

      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88

uid                  Docker Release (CE deb) <docker@docker.com>

sub   4096R/F273FCD8 2017-02-22
```

Use the following command to set up the stable repository. You always need the stable repository, even if you want to install builds from the edge or test repositories as well. To add the edge or test repository, add the word edge or test (or both) after the word stable in the commands below.

Note: The lsb_release -cs sub-command below returns the name of your Ubuntu distribution, such as xenial. Sometimes, in a

distribution like Linux Mint, you might have to change $(lsb_release -cs) to your parent Ubuntu distribution. For example, if you are using Linux Mint Rafaela, you could use trusty.

---

amd64:

```
$ sudo add-apt-repository \

   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

   $(lsb_release -cs) \

   stable"
```

INSTALL DOCKER CE

Update the apt package index:

```
$ sudo apt-get update
```

Install the latest version of Docker CE:

```
$ sudo apt-get install docker-ce
```

Docker on Mac

Install Docker for Mac

Docker for Mac is the Community Edition (CE) of Docker for MacOS. To download Docker for Mac, head to Docker Store.

Download from Docker Store

What to know before you install

README FIRST for Docker Toolbox and Docker Machine users

If you are already running Docker on your machine, first read Docker for Mac vs. Docker Toolbox to understand the impact of this installation on your existing setup, how to set your environment for Docker for Mac, and how the two products can coexist.

- **Relationship to Docker Machine**: Installing Docker for Mac does not affect machines you created with Docker Machine. You have the option to copy containers and images from your local default machine (if one exists) to the new Docker for Mac HyperKitVM. When you are running Docker for Mac, you do not need Docker Machine nodes running at all locally (or anywhere else). With Docker for Mac, you have a new, native virtualization system running (HyperKit) which takes the place of the VirtualBox system. To learn more, see Docker for Mac vs. Docker Toolbox.
    - Mac hardware must be a 2010 or newer model, with Intel's hardware support for memory management unit (MMU) virtualization, including Extended Page Tables (EPT) and Unrestricted Mode. You can check to see if your machine has this support by running the following command in a terminal: sysctl kern.hv_support
    - macOS El Capitan 10.11 and newer macOS releases are supported. We recommend upgrading to the latest version of macOS.
    - At least 4GB of RAM
    - VirtualBox prior to version 4.3.30 must NOT be installed (it is incompatible with Docker for Mac). If you have a newer version of VirtualBox installed, it's fine.**System Requirements**: Docker for Mac launches only if all of these requirements are met.

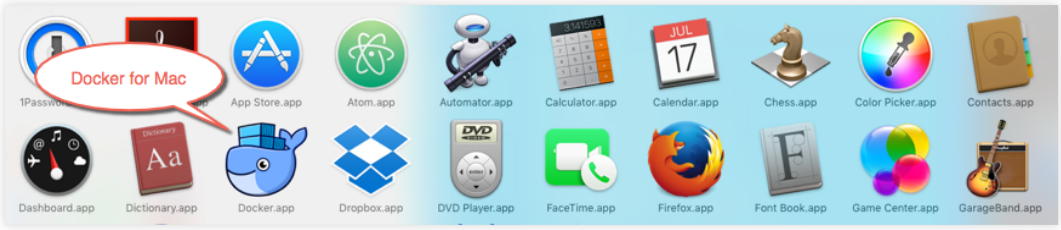
Note: If your system does not satisfy these requirements, you can install Docker Toolbox, which uses Oracle VirtualBox instead of HyperKit.

- **What the install includes**: The installation provides Docker Engine, Docker CLI client, Docker Compose, Docker Machine, and Kitematic.

Install and run Docker for Mac

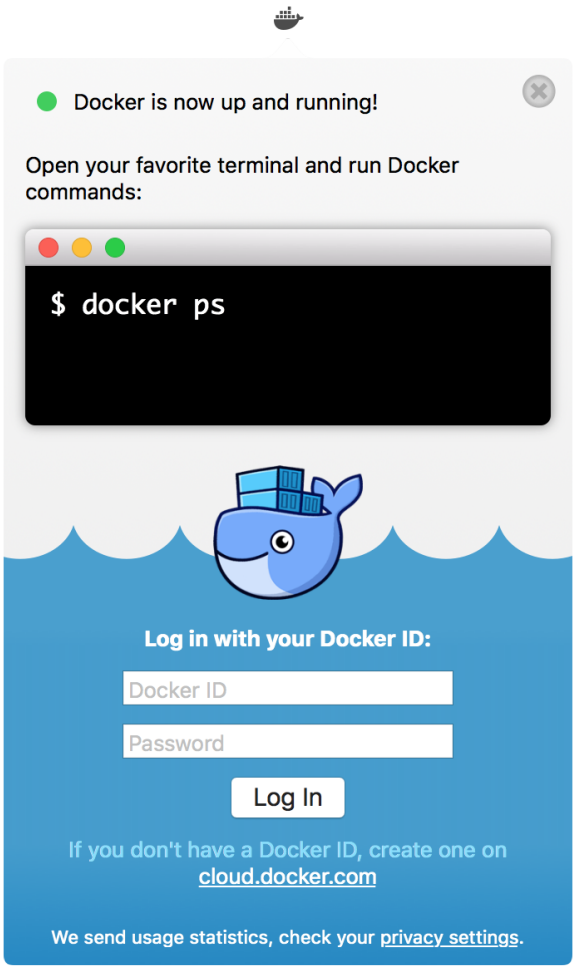
1. Double-click Docker.dmg to open the installer, then drag Moby the whale to the Applications folder.

2. Double-click Docker.app in the Applications folder to start Docker. (In the example below, the Applications folder is in “grid” view mode.)



You are prompted to authorize Docker.app with your system password after you launch it. Privileged access is needed to install networking components and links to the Docker apps.The whale in the top status bar indicates that Docker is running, and accessible from a terminal.

If you just installed the app, you also get a success message with suggested next steps and a link to this documentation. Click the whale ( ) in the status bar to dismiss this popup.



3. Click the whale ( ) to get Preferences and other options.
4. Select **About Docker** to verify that you have the latest version.

Congratulations! You are up and running with Docker for Mac.

Docker on Ubuntu / Debian

Get Docker CE

Referenced to the official Docker instructions.

Note: This installation instructions is based on the example of the operating system Ubuntu 16.04 LTS.

SET UP THE REPOSITORY

Update the apt package index:

```
$ sudo apt-get update
```

Install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get install \

    apt-transport-https \

    ca-certificates \

    curl \

    software-properties-common
```

Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

```
$ sudo apt-key fingerprint 0EBFCD88

pub   4096R/0EBFCD88 2017-02-22

      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88

uid                  Docker Release (CE deb) <docker@docker.com>

sub   4096R/F273FCD8 2017-02-22
```

Use the following command to set up the stable repository. You always need the stable repository, even if you want to install builds from the edge or test repositories as well. To add the edge or test repository, add the word edge or test (or both) after the word stable in the commands below.

Note: The lsb_release -cs sub-command below returns the name of your Ubuntu distribution, such as xenial. Sometimes, in a

distribution like Linux Mint, you might have to change $(lsb_release -cs) to your parent Ubuntu distribution. For example, if you are using Linux Mint Rafaela, you could use trusty.

---

amd64:

```
$ sudo add-apt-repository \

   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

   $(lsb_release -cs) \

   stable"
```

INSTALL DOCKER CE

Update the apt package index:

```
$ sudo apt-get update
```

Install the latest version of Docker CE:

```
$ sudo apt-get install docker-ce
```

Docker on Windows 10

Install Docker for Windows

Docker for Windows is the Community Edition (CE) of Docker for Microsoft Windows. To download Docker for Windows, head to Docker Store.

Download from Docker Store

What to know before you install

If your system does not meet the requirements to run Docker for Windows, you can install Docker Toolbox, which uses Oracle Virtual Box instead of Hyper-V.

- **README FIRST for Docker Toolbox and Docker Machine users**: Docker for Windows requires Microsoft Hyper-V to run. The Docker for Windows installer enables Hyper-V for you, if needed, and restart your machine. After Hyper-V is enabled, VirtualBox no longer works, but any VirtualBox VM images remain. VirtualBox VMs created with docker-machine (including the defaultone typically created during Toolbox install) no longer start. These VMs cannot be used side-by-side with Docker for Windows. However, you can still use docker-machine to manage remote VMs.
- Virtualization must be enabled in BIOS and CPU SLAT-capable. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see Virtualization must be enabled in Troubleshooting.
- The current version of Docker for Windows runs on 64bit Windows 10 Pro, Enterprise and Education (1607 Anniversary Update, Build 14393 or later).
- Containers and images created with Docker for Windows are shared between all user accounts on machines where it is installed. This is because all Windows accounts use the same VM to build and run containers.
- Nested virtualization scenarios, such as running Docker for Windows on a VMWare or Parallels instance, might work, but come with no guarantees. For more information, see Running Docker for Windows in nested virtualization scenarios
- **What the Docker for Windows install includes**: The installation provides Docker Engine, Docker CLI client, Docker Compose, Docker Machine, and Kitematic.

About Windows containers

Looking for information on using Windows containers?

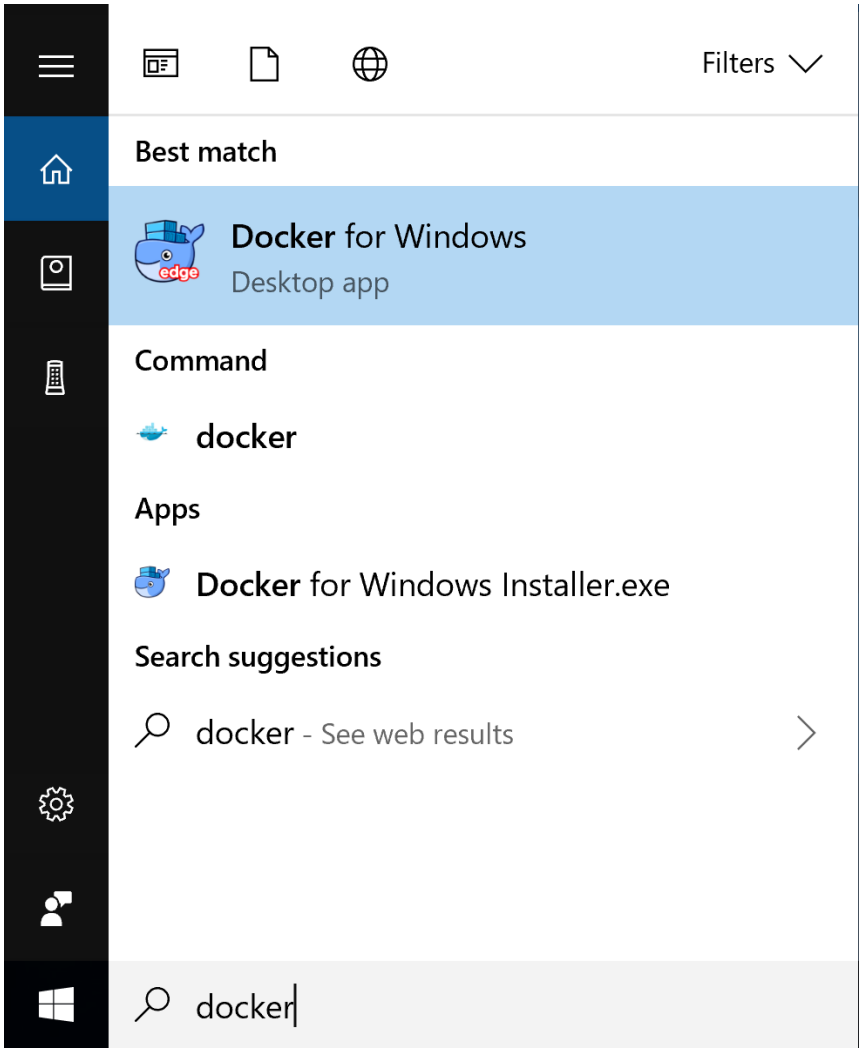
- Switch between Windows and Linux containers describes the Linux / Windows containers toggle in Docker for Windows and points you to the tutorial mentioned above.
- Getting Started with Windows Containers (Lab) provides a tutorial on how to set up and run Windows containers on Windows 10 or with Windows Server 2016. It shows you how to use a MusicStore application with Windows containers.
- Docker Container Platform for Windows Server 2016 articles and blog posts on the Docker website

Install Docker for Windows desktop app

1. Double-click **Docker for Windows Installer.exe** to run the installer.If you haven't already downloaded the installer (Docker for Windows Installer.exe), you can get it from **download.docker.com**. It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.
2. Follow the install wizard to accept the license, authorize the installer, and proceed with the install.You are asked to authorize Docker.app with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
3. Click **Finish** on the setup complete dialog to launch Docker.

Start Docker for Windows

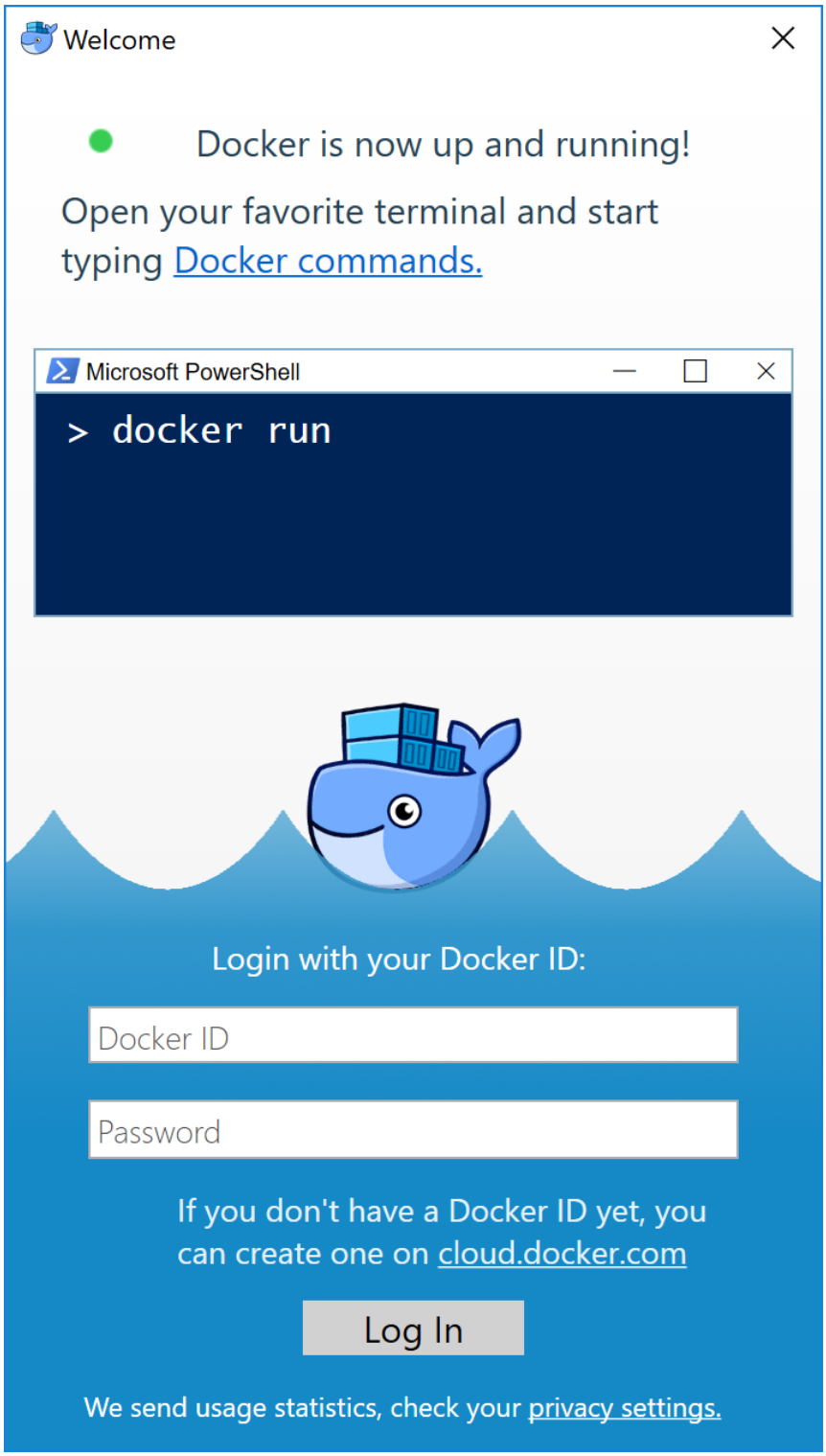
Docker does not start automatically after installation. To start it, search for Docker, select **Docker for Windows** in the search results, and click it (or hit Enter).



When the whale in the status bar stays steady, Docker is up-and-running, and accessible from any terminal window.

If the whale is hidden in the Notifications area, click the up arrow on the taskbar to show it. To learn more, see Docker Settings .

If you just installed the app, you also get a popup success message with suggested next steps, and a link to this documentation.



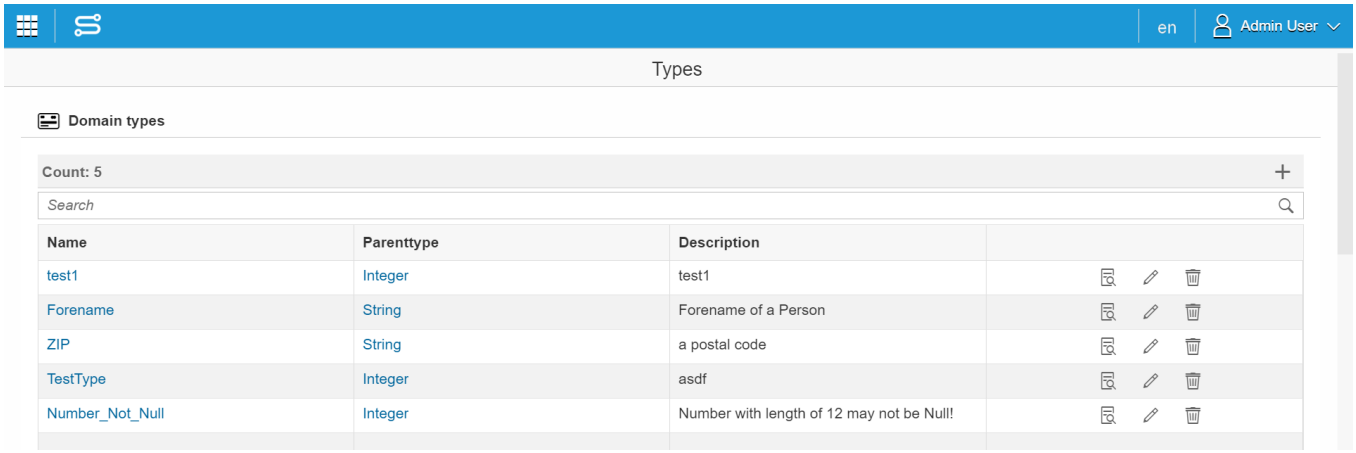
When initialization is complete, select **About Docker** from the notification area icon to verify that you have the latest version.

Congratulations! You are up and running with Docker for Windows.

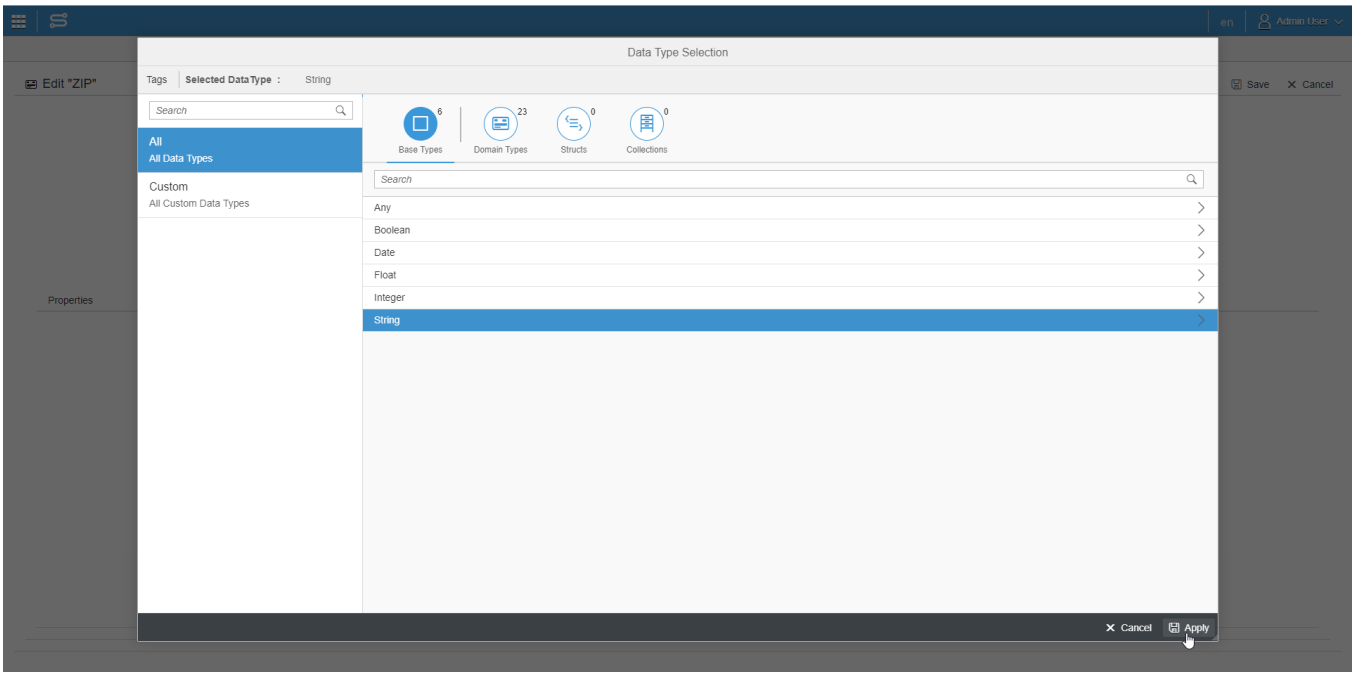
Domain Type

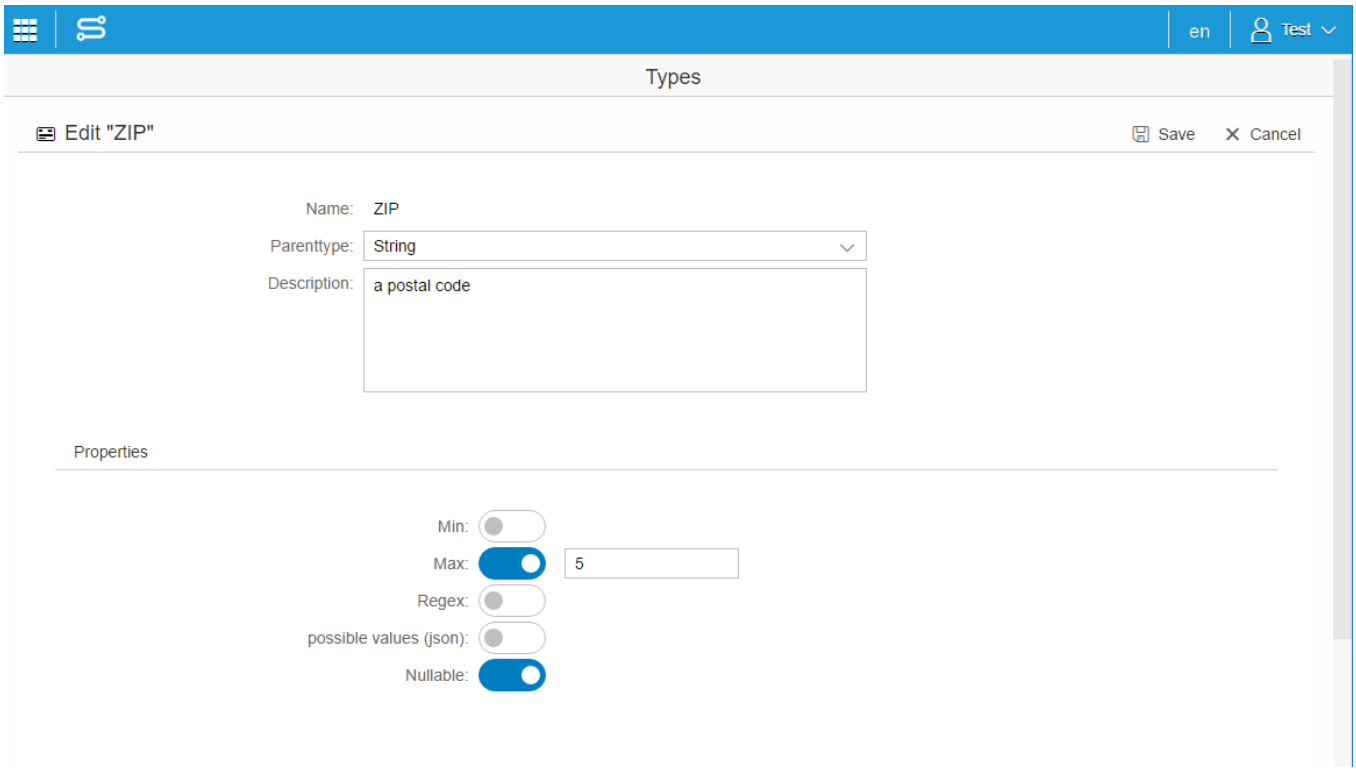
A Domain type represents a single Data Type that inherits from a Base Type like String, Integer, Float, Date etc. and can include different properties.
For example a ZIP Code is an inherited type of string with the property of a maximum of 5 chars length.

To create a new Domain Type click on the “+” button.



Enter a unique Name for the Domain Type and an optional Description.
Also select a Parenttype to inherit from by clicking on the appropriate field.


en
Admin User
Edit "ZIP"
Save
Cancel
Properties
Data Type Selection
Tags
Selected DataType :
String
Search
All
All Data Types
Custom
All Custom Data Types
6
Base Types
23
Domain Types
0
Structs
0
Collections
Search
Any
Boolean
Date
Float
Integer
String
Cancel
Apply



In the area below you can set the properties of your Data Type:

Min

Minimum length of a String, minimum number of a range, earliest date of a date range.

Max

Maximum length of a String, maximum number range, latest date of a date range.

RegEx

Regular expression to validate this Domain Type – for more information see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp.

Possible Values

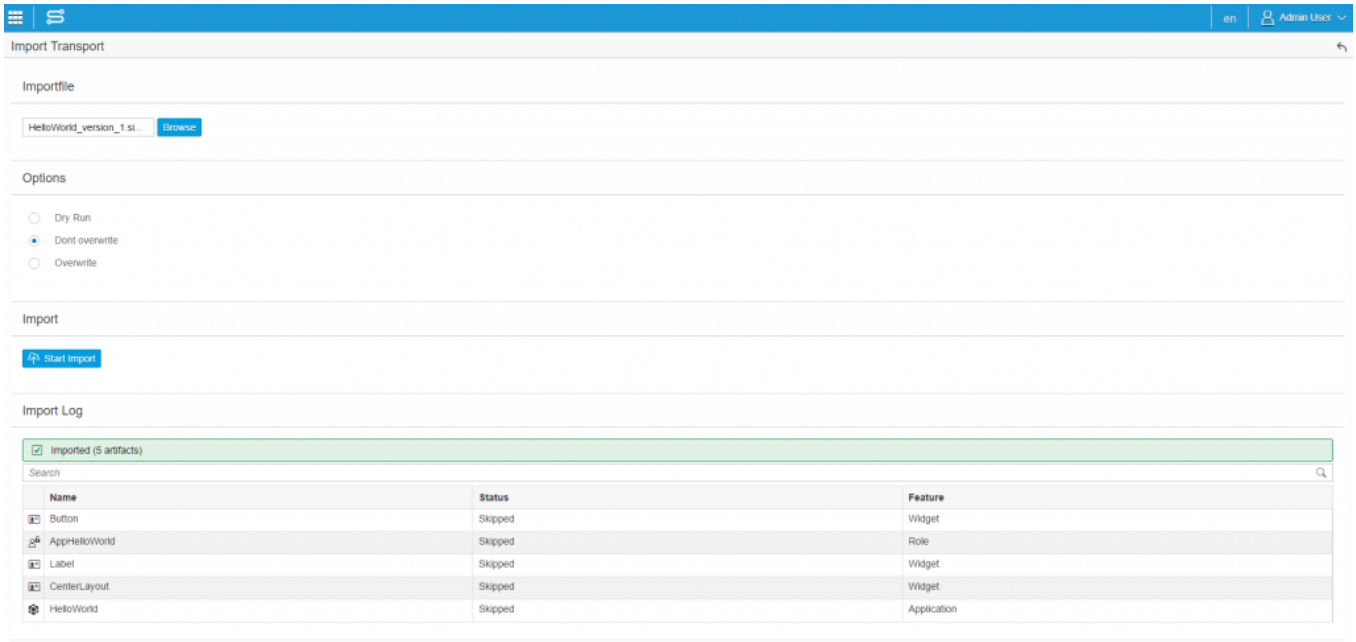
Simple JSON Array of Strings repesenting literals of chosen Base Type.

Nullable

If activated, this value can be empty (null).

Don't overwrite data

The options „Don’t overwrite“ and „Overwrite“ are going to import the data to your system.
By choosing “Don’t overwrite”, only new features are uploaded. Every feature with the same name like an existing one is skipped.



Downloads

This section offers various Downloads as examples on how to use different features of the Simplifier.

If you have a problem building your App, you can import a fitting Transport to your Simplifier instance and reproduce the steps.

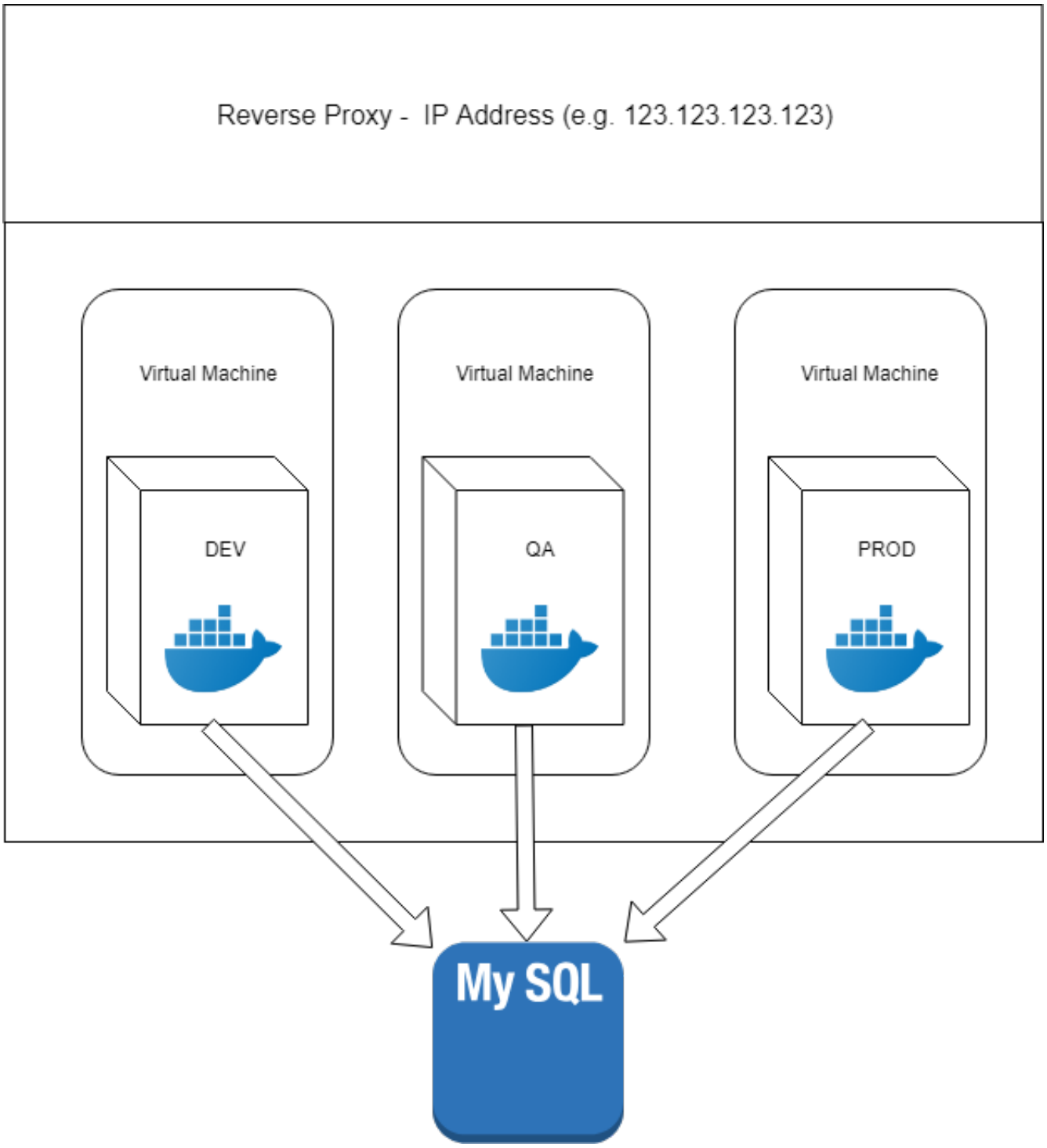
_____

DQP System

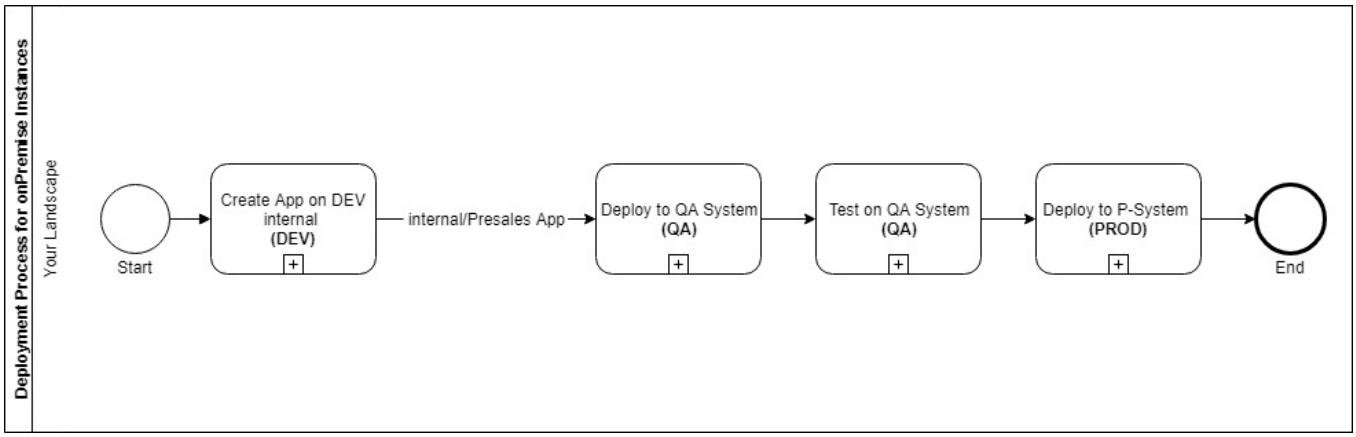
For each step in the development and update process of the Simplifier, an independent instance should be used.

- Development
- Quality Assurance
- Productive

Here we would like to give you an example for deploying a DQP system on three virtual machines.



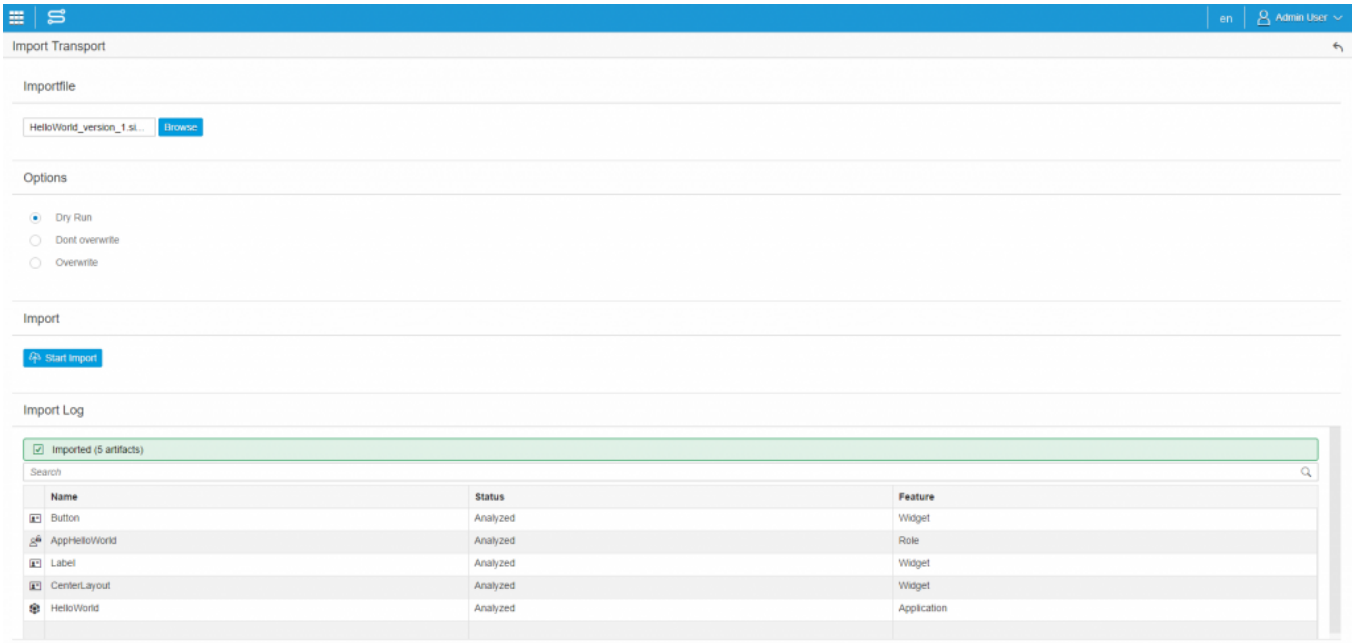
Deployment and Integration Workflow D-Q-P



- Development of an app on the DEV instance
- Transport to the QA instance
- Testing the app on the QA instance
- Transport to the Productive instance

Dry run

A “Dry run” analyzes the content of the file and displays a list of all features.
It does not import the data so you can simply test how the transport would work out.



Edit a PDF Template

Edit Template

To edit a PDF template, you need the following parameter:

URL

Input-Parameter

Data

Stylesheet

PreviewJson

Output-Parameter

Example for a call:

```
{     "name": "templatename",     "data": "SGFsbG8gV2VsdA==\",     "stylesheet: "SGFs
bG8gV2VsdA==\",     "previewJson": "SGFsbG8gV2VsdA==\" }
```

Output example:

```
{     "success": true }
```

Email Connector Call

The Email Connector Call requires 3 input parameters to be defined:

receiver	The email address of the receiver.
subject	The subject of your message.
msg	The message itself.

All Data Types should be set as String.

There are 3 optional parameters that can be configured as well:

receiverCC	The email address of the copy receiver.	String
receiverBCC	The email address of the blind copy receiver.	String
attachments	A list of all attachements.	List[ByteAttachment]

ByteAttachment

{

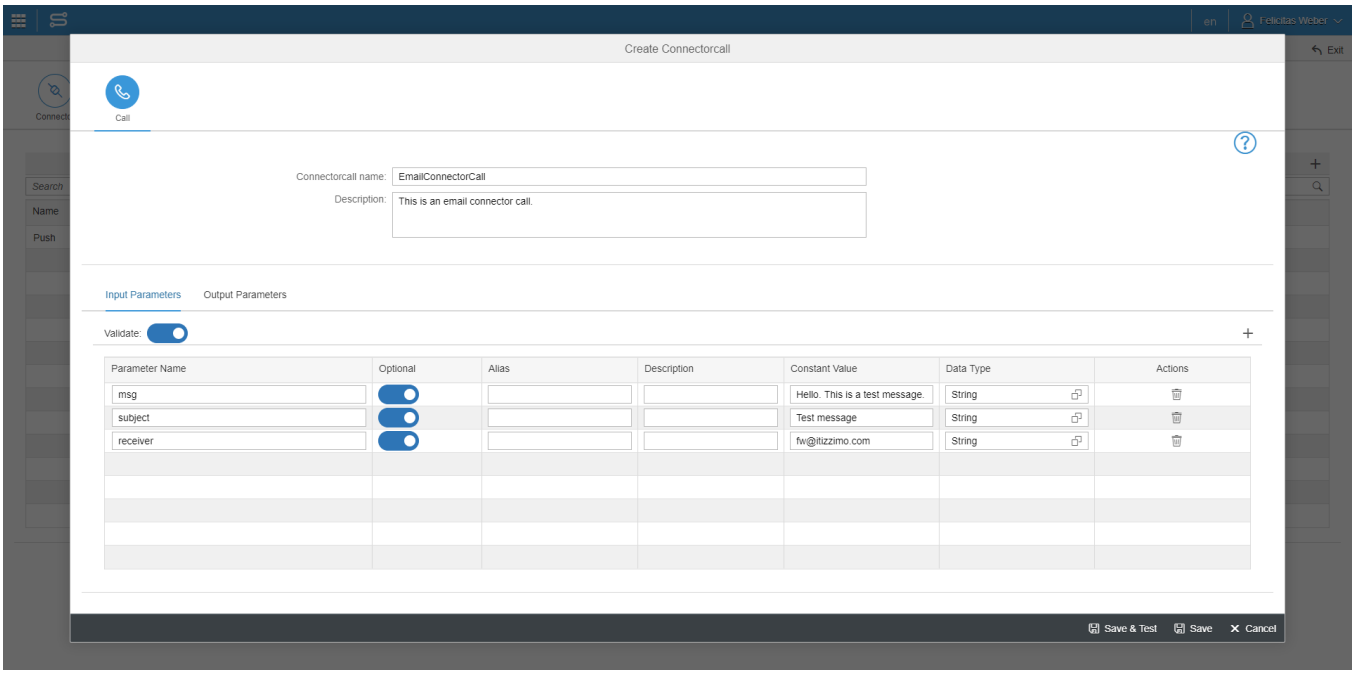
“session”: String,

“fileName”: String,

“attachmentMimeType”: String

}

Example:


Create Connectorcall
Call
Connectorcall name:
EmailConnectorCall
Description:
This is an email connector call.
Input Parameters
Output Parameters
Validate:
Parameter Name
Optional
Alias
Description
Constant Value
Data Type
Actions
msg
Hello. This is a test message.
String
subject
Test message
String
receiver
fw@itizzimo.com
String
Save & Test
Save
Cancel

Email Connector Details

SMTP host

Hostname of SMTP Server

SMPT port

Port of SMTP server

SMTP authentication

If enabled, the client attempt to authenticate the user using the AUTH command. Default to false.

SMTP StartTLS

If activated, it enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands. Note that an appropriate trust store must configured, so that the client will trust the server’s certificate. Default to false.

Encode/Decode Base64

Encode Base64

Encodes a string to a base64 string.

```
Simplifier.Util.encodeBase64(string: string): string
```

Example:

```
try{
    var encoded = Simplifier.Util.encodeBase64("String to encode");

    output.b64 = encodedD64;
    output.message = "Encoding successful";
    output.success = true;
} catch (e) {
    output.message = e.message;
    output.success = false;
}
```

Decode Base64

Decodes a base64 string.

```
Simplifier.Util.decodeBase64(json: string): string
```

Example:

```
try{
    var decoded = Simplifier.Util.decodeBase64(encoded);
```

```
    output.b64 = decodedBase64;

    output.message = "Decoding successful";

    output.success = true;

} catch (e) {

    output.message = e.message;

    output.success = false;

}
```

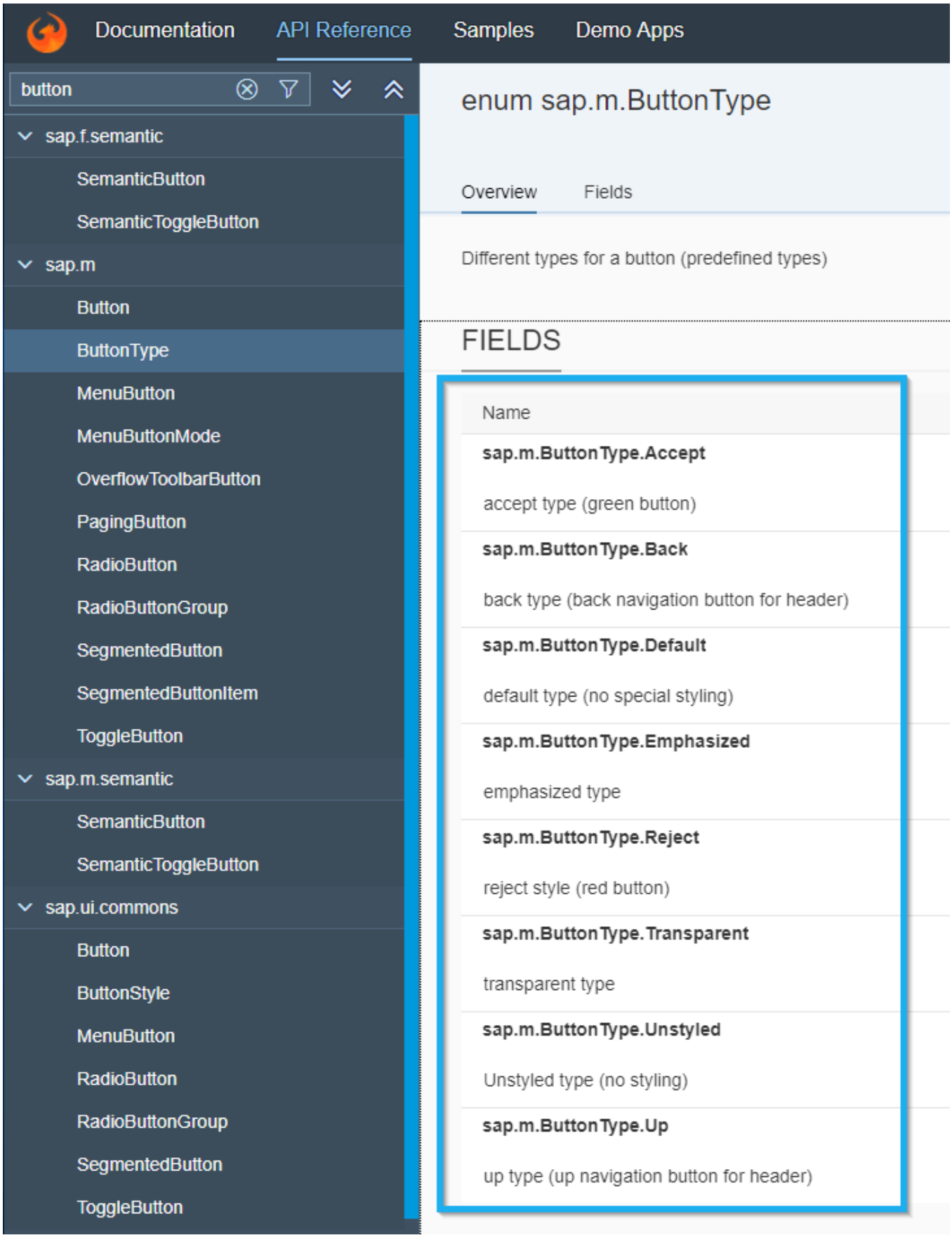
Enumeration in Widget Properties

You can maintain enumeration for Widget properties. Enumerated properties can only hold the defined values and will be displayed as selection in the UI Designer.

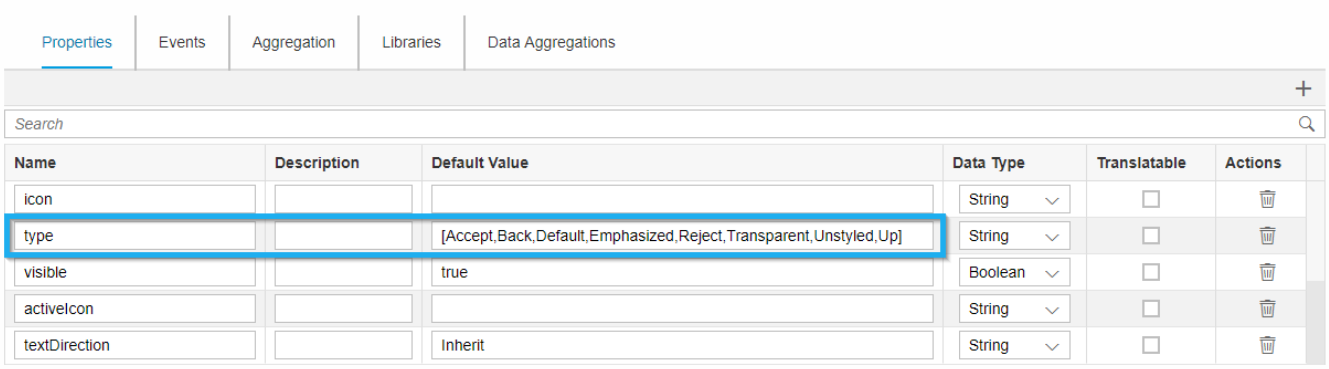
In order to define the enumerations you have to add them in an Array notation to the default value of the property. It is not necessary to set the values in quotes.

Example

There are different predefined types of buttons in OpenUI5. The properties can be maintained as a list in the widget mask. In our example, we look at all different button types in the OpenUi5 API Reference and transfer them into the widget edit mask with the appropriate syntax: [typ1, typ2].

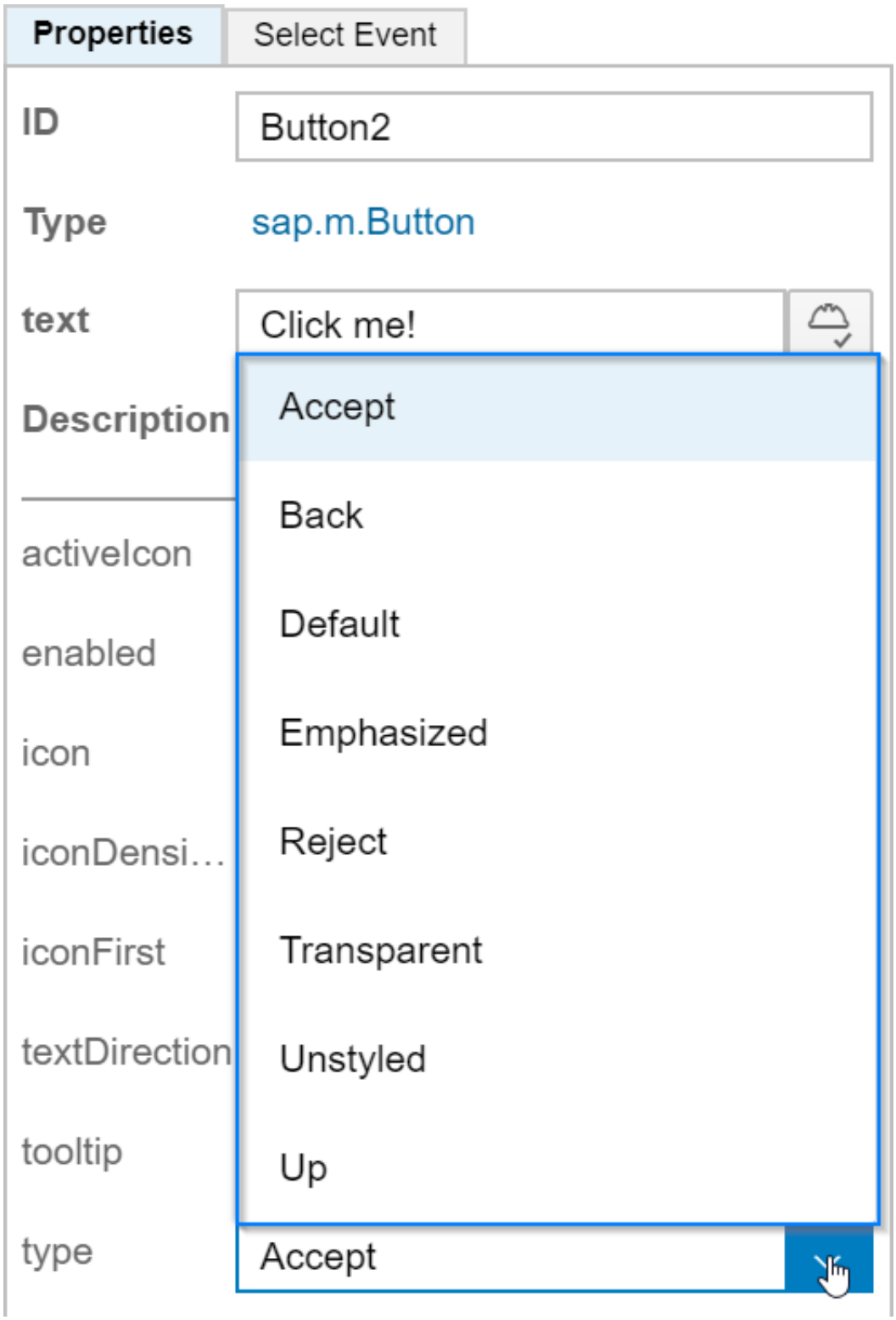


OpenUI5 API Reference



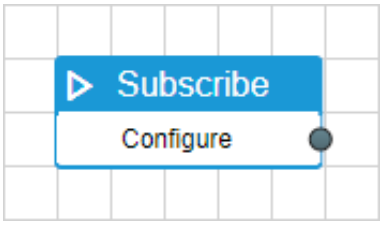
simplifier Widget edit mask

Result in the UI Designer


Properties
Select Event
ID
Button2
Type
sap.m.Button
text
Click me!
Description
activeIcon
enabled
icon
iconDensi…
iconFirst
textDirection
tooltip
type
Accept
Accept
Back
Default
Emphasized
Reject
Transparent
Unstyled
Up

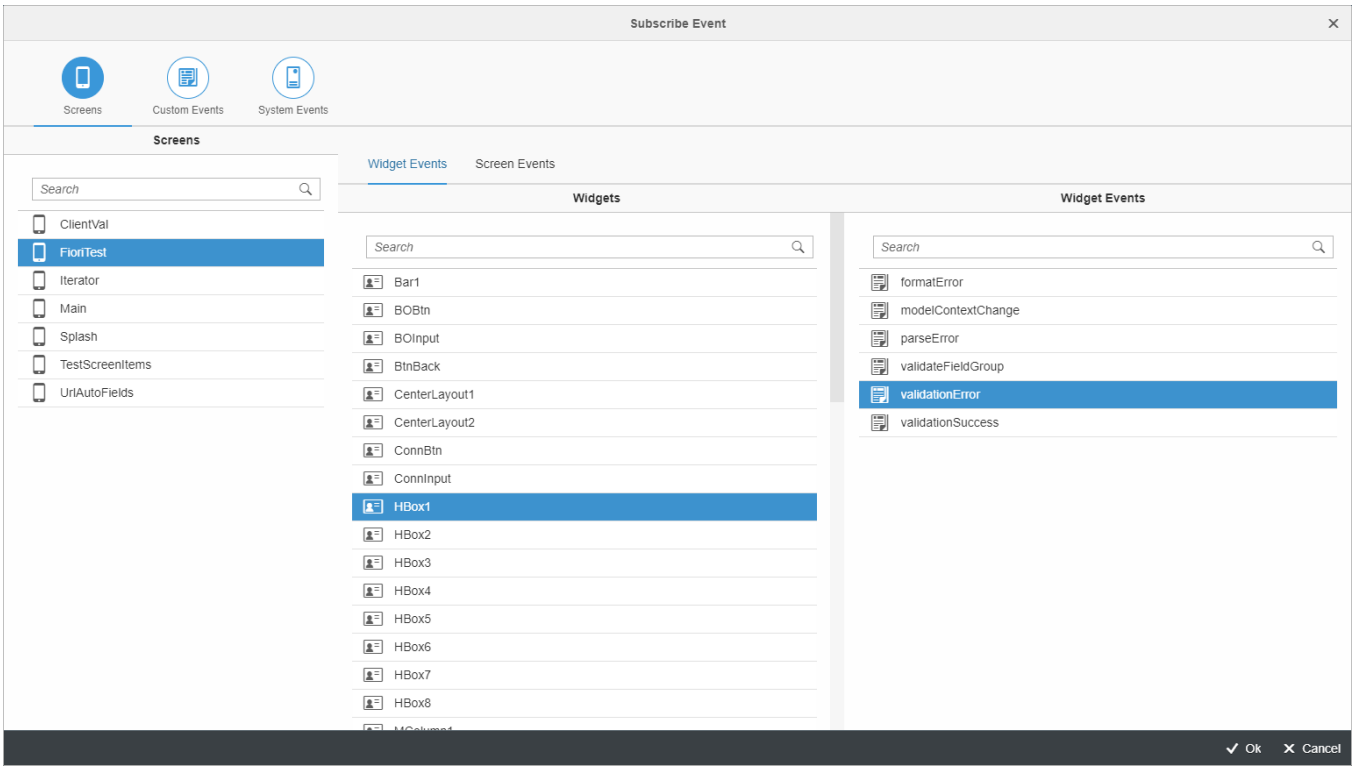
Event

Each workflow in the Process Designer starts with an event. Click on the Event element under “Activities” on the left pane and drag & drop it into the drawing area.

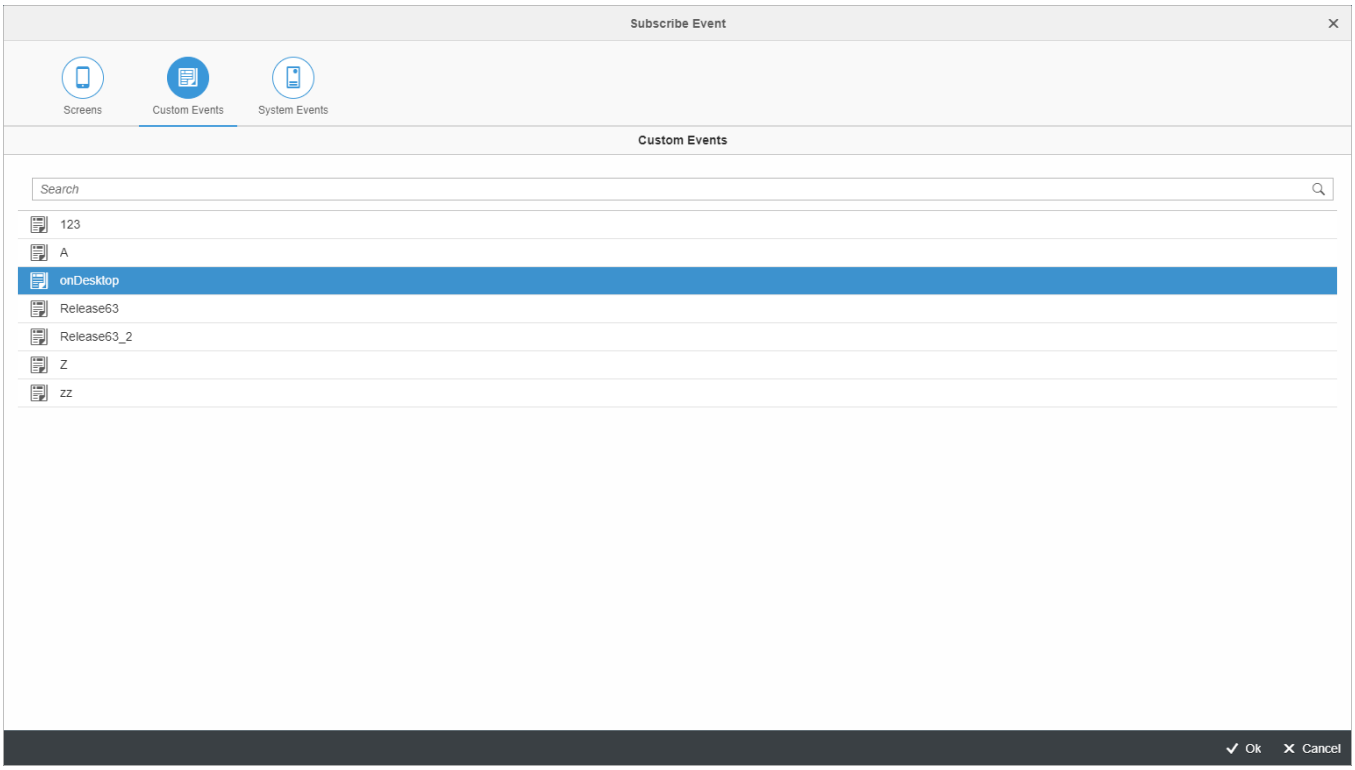


If you double click on the shape of the activity, the event selection assistant opens. You can also open it by clicking right underneath “Subscribe event” / “Publish event”, an Event selection assistant will be open. It guides you to your required Event. When subscribing an Event, you can select Events from three categories: Screens (Widget Events/Screen Events), Custom Events and System Events. Each category is sorted and searchable.

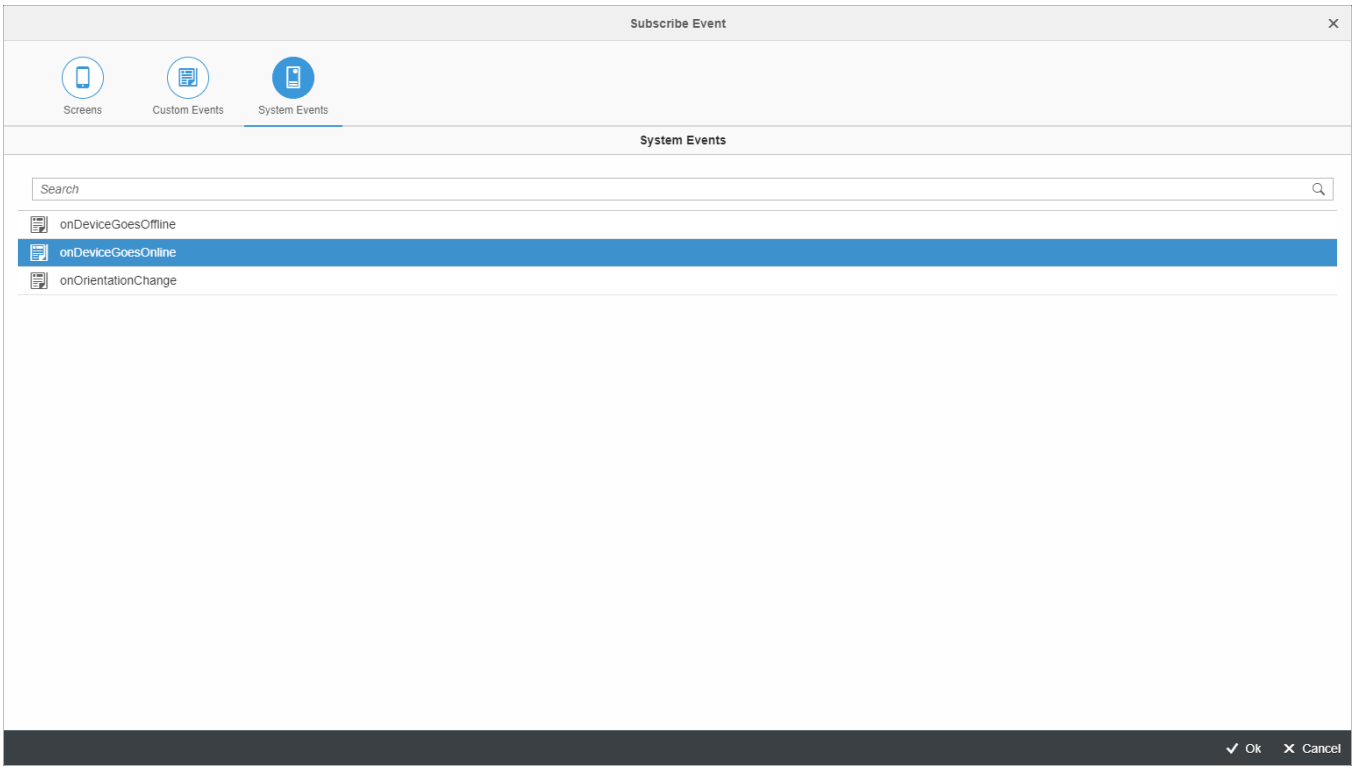
If you want to use your Event in the User Story you are currently working in, select one of the Events under the tab “Subscribe Events”.


Subscribe Event
Screens
Custom Events
System Events
Screens
Search
ClientVal
FioriTest
Iterator
Main
Splash
TestScreenItems
UrlAutoFields
Widget Events
Screen Events
Widgets
Search
Bar1
BOBtn
BOInput
BtnBack
CenterLayout1
CenterLayout2
ConnBtn
ConnInput
HBox1
HBox2
HBox3
HBox4
HBox5
HBox6
HBox7
HBox8
Widget Events
Search
formatError
modelContextChange
parseError
validateFieldGroup
validationError
validationSuccess
Ok
Cancel

Screens



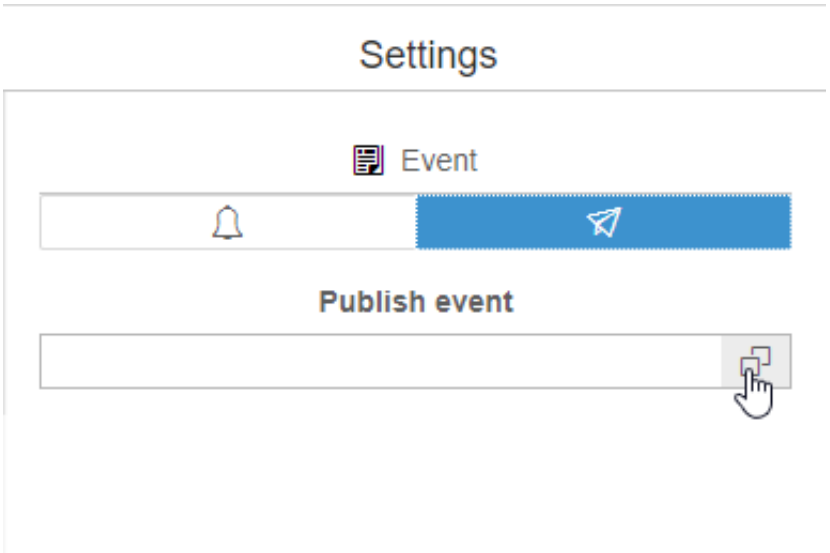
Custom Events



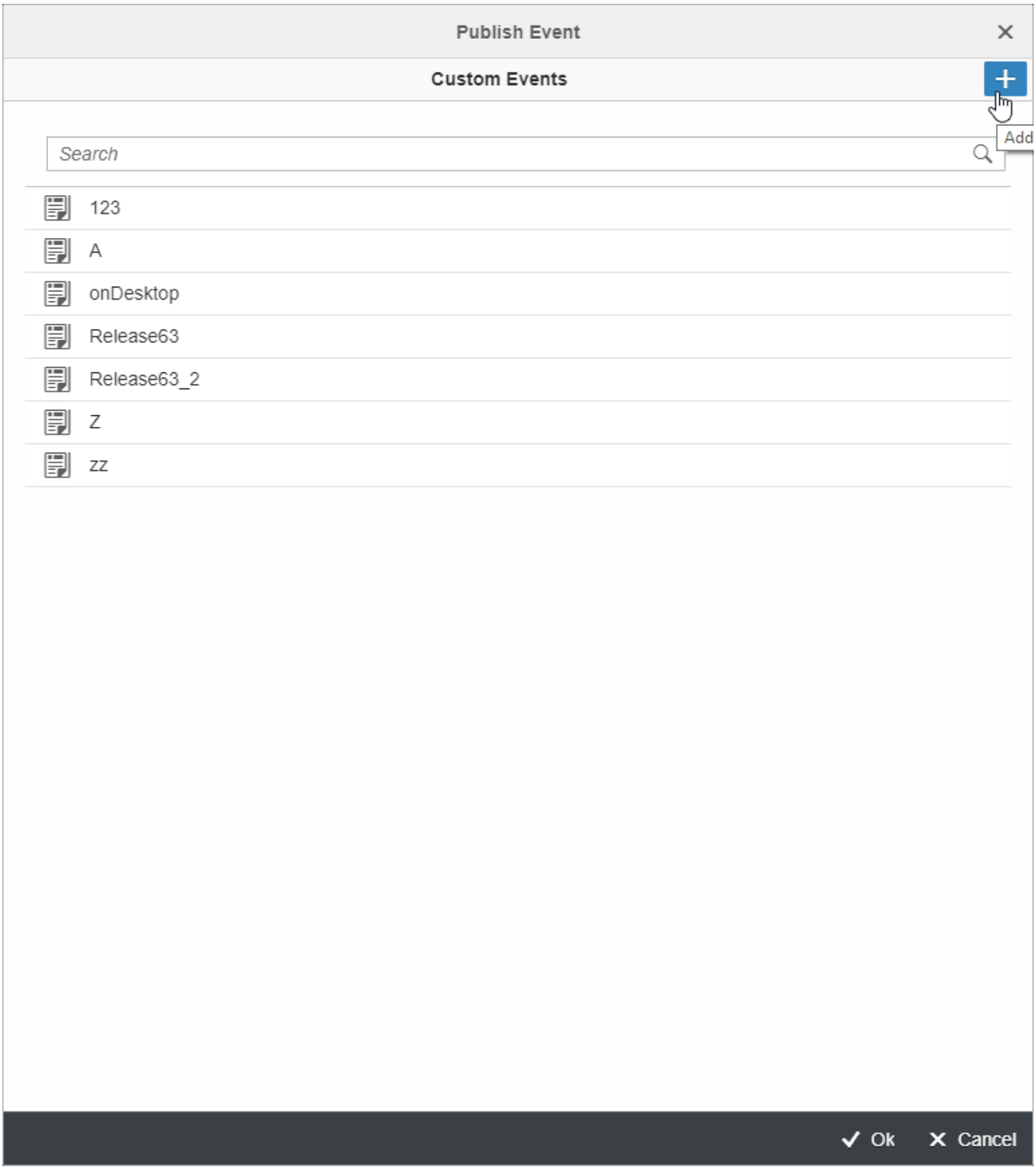
System Events

The assistant considers every Event managed by the application.

It is only possible to publish Custom Events. The Assistant is aware of this and only displays this category when you define your publishing activity.



If you want to make an Event usable in another Stories, create a new Custom Event under the tab “Publish Event”. Then you can subscribe to it in another User Story. A shorcut for creating Custom Events can be found in the top right-hand corner.



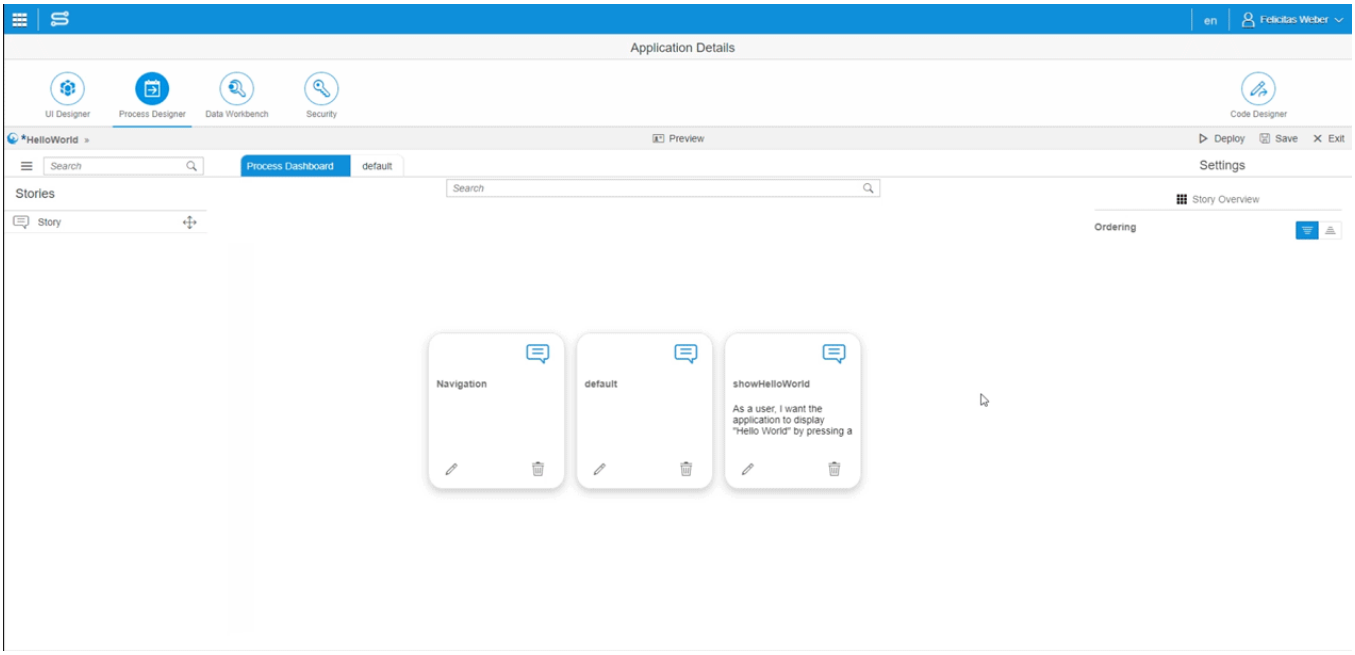
Example:

You have a process in User Story 1 that contains a condition to check if an Input field is filled out after clicking on a “Login” Button. Afterwards, the User should be navigated forward to the next Screen.

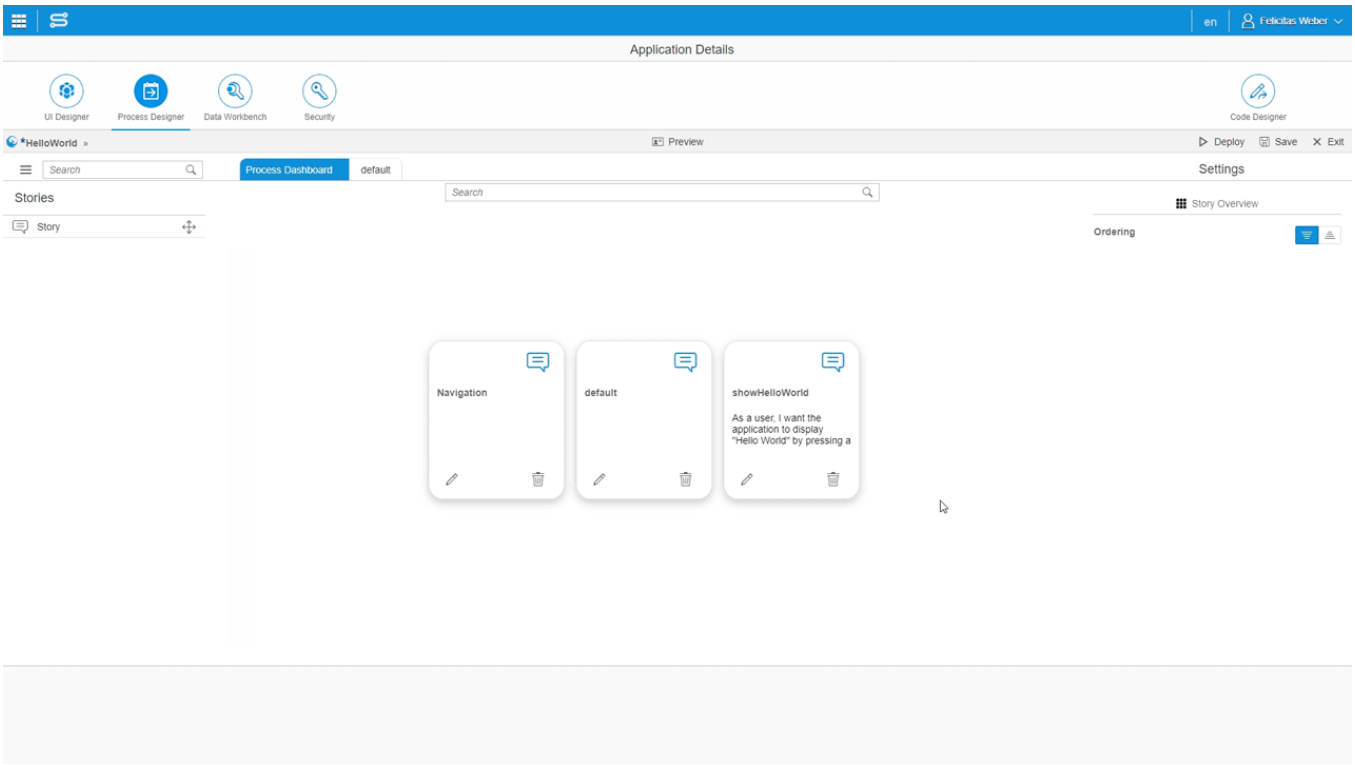
Imagine you have an extra User Story exclusively for the whole Navigation of your App. So naturally you want the end of the Event from User Story 1 (the Navigation) displayed in User Story 2.

Therefore we published the new custom Event “LoginButton” in User Story 1 and subscribed to it in User Story 2.

User Story 1:



User Story 2:



Published Events can be maintained in the Data Workbench.

Example Cordova Plugin for Mobile Action

As an example, we would like to show a pdf document as a native mobile action after clicking on a button in a business application.

Step 1

Add a button to your screen in the UI Designer.

Step 2

Edit your user story and add the activity “Event” and “Script” to your working area. Assign your button (press) to the Event and click on “Script” to edit the JavaScript code.

Step 3

To view a pdf document, you need the Cordova Plugin “File Transfer” and “Document Viewer“. Open their documentation and head down to “3. Using the Plugin”. Here you can find the necessary JavaScript Code. Add it to your Script.

In this example:

```
var options = {
 title: "Title",
 documentView : {
 closeLabel : "close"
 },
 navigationView : {
 closeLabel : "close"
 },
 email : {
 enabled : false
 },
 print : {
 enabled : false
 },
 openWith : {
 enabled : false
 },
 bookmarks : {
 enabled : false
 },
 search : {
 enabled : false
 },
 autoClose: {
 onPause : false
 }
}

function onPossible(){
 window.console.log('document can be shown');
 //e.g. track document usage
}

function onMissingApp(appId, installer)
```

```
{
 if(confirm("Do you want to install the free PDF Viewer App "
 + appId + " for Android?"))
 {
 installer();
 }
}

function onImpossible(){
 window.console.log('document cannot be shown');
 //e.g. track document usage
}

function onError(error){
 window.console.log(error);
 alert("Sorry! Cannot show document.");
}

function onShow(){
 window.console.log('document shown');
 //e.g. track document usage

 //reset baseurl
 document.getElementsByTagName('base')[0].href = temp;
}

function onClose(){
 window.console.log('document closed');
 //e.g. remove temp files
}

function onMissingApp(id, installer)
{
 if(confirm("Do you want to install the free PDF Viewer App "
 + appId + " for Android?"))
 {
 console.log("trigger installer");
 }
}

var fileTransfer = new FileTransfer();
var uri = encodeURI(
"https://projects.simplifier.io/appDirect/Xenon_OPC_UA_Consolidated/data/Parametrieru
ng_Dosiereinheit.pdf");
var fileURL = cordova.file.cacheDirectory + "asdf.pdf"

//save current base
var temp = document.getElementsByTagName('base')[0].href;

//download file
fileTransfer.download(
 uri,
 fileURL,
 function(entry) {
 console.log("download complete: " + entry.toURL());
```

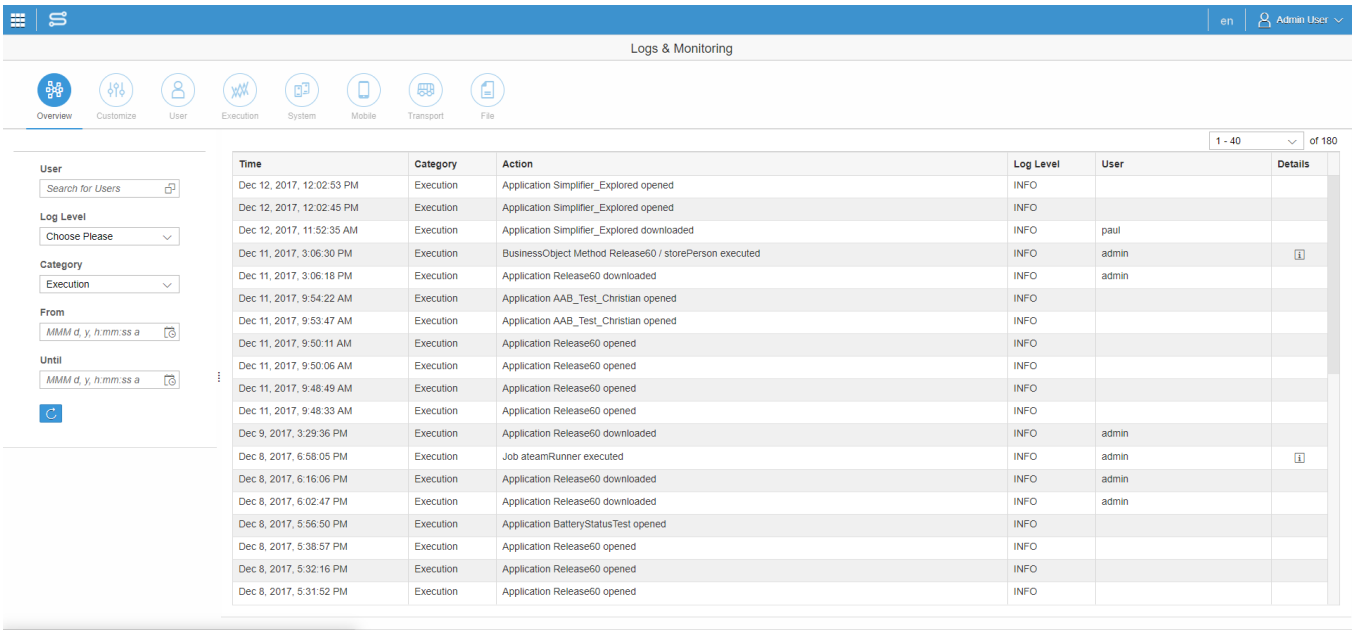
```
 //set base to local to make it work
 console.log("resetting base");
 document.getElementsByTagName('base')[0].href = "";

 cordova.plugins.SitewaertsDocumentViewer.canViewDocument(
 entry.toURL(), 'application/pdf'
, options, onPossible, onMissingApp, onImpossible, onError);

 cordova.plugins.SitewaertsDocumentViewer.viewDocument(
 entry.toURL(), 'application/pdf', options, onShow, onClose, onMissingApp, onError);

 },
 function(error) {
 console.log("download error source " + error.source);
 console.log("download error target " + error.target);
 console.log("download error code" + error.code);
 },
 false
);
```

Step 4

Save and deploy the app.

Execution Log

You can use the execution log to trace the execution of e.g. connectors.



The following type of entries are logged:

Type	**Description**
Open App	When the direct path to an app is opened (appDirect).
Download App	When downloading the app to the client (user context is provided).
Connector Execution	When using a connector directly, the execution and payload will be logged.
Connector Call Execution	When a connector call is invoked. All parameters, even the constant parameters are logged.
Business Object Execution	When using a business object, the payload and parameters are logged.
Plugins	Plugins which are called by the old akka interface.
Asynchronous Connectors	When subscribing and unsubscribing a connector.
Job Execution	Every execution of the job.
Any above	Any exception by executing an artefact above.

Fetch a PDF Template

Fetch Template

To fetch a PDF template, you need the following parameter:

URL	/client/1.0/PLUGIN/pdfPlugin/adminTemplateFetch		
Input-Parameter	**Name**		Template name
Output-Parameter	**Value**	**Template**	HTML Template Content (Base64-coded)
		Stylesheet	Content of the LESS Stylesheets ((Base64-coded, optional)
		PreviewJson	Content of the sample data in JSON format (Base64-coded, optional)

Example for a call:

```
{     "name": "templatename" }
```

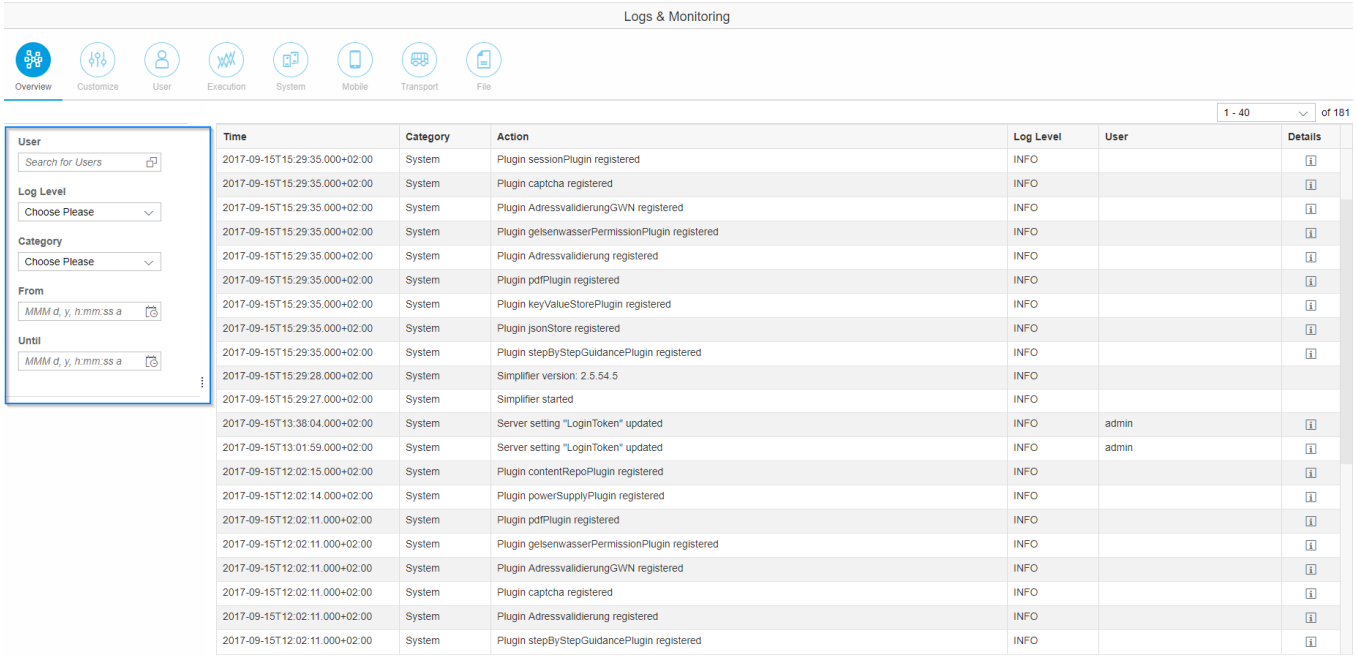
Output example:

```
{     "success": true,
 "value": { "template": "SGFsbG8gV2VsdA==\", "stylesheet: "SGFsbG8gV2VsdA==\" "previe
wJson": "SGFsbG8gV2VsdA==\" } }
```

Filter

The Logs & Monitoring Tile uses all search features of the backend (i.e. pagination or filtering).

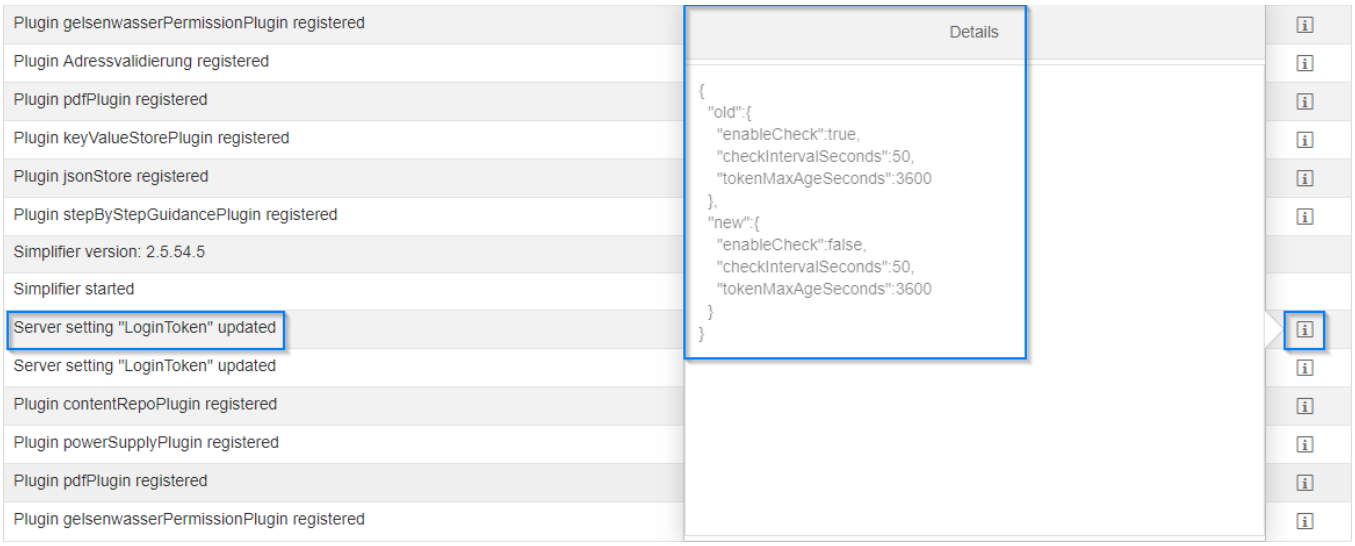
On the left-hand side you can set filters.



You can choose between the following filters.

Filter	**Function**
User	Filter for specific user actions.
Log Level	Filters based on the severity of the message: INFO, WARNING, ERROR.
Category	The Categories as shown in the tabs: Customize, User, Execution, System, Mobile, Transport, File.
From	From Date.
Until	Until Date.

In the logs on the right-hand side you can click a detail button, which will open further information.


Plugin gelsenwasserPermissionPlugin registered
Plugin Adressvalidierung registered
Plugin pdfPlugin registered
Plugin keyValueStorePlugin registered
Plugin jsonStore registered
Plugin stepByStepGuidancePlugin registered
Simplifier version: 2.5.54.5
Simplifier started
Server setting "LoginToken" updated
Server setting "LoginToken" updated
Plugin contentRepoPlugin registered
Plugin powerSupplyPlugin registered
Plugin pdfPlugin registered
Plugin gelsenwasserPermissionPlugin registered
Details
{
"old":{
"enableCheck":true,
"checkIntervalSeconds":50,
"tokenMaxAgeSeconds":3600
},
"new":{
"enableCheck":false,
"checkIntervalSeconds":50,
"tokenMaxAgeSeconds":3600
}
}

FQDN

A fully qualified domain name (FQDN) is sometimes also referred to as an absolute domain name.

Example on our Simplifier cloud:

Development	dev-yourcompany.simplifier.io
Quality Assurance	qa-yourcompany.simplifier.io
Productive	yourcompany.simplifier.io

Example for onpremise installation:

Development	dev-simplifier.yourcompany.com
Quality Assurance	qa-simplifier.yourcompany.com
Productive	simplifier.yourcompany.com

Frontend Exercise - Mobile SAP Purchase Orders

This Transport provides all components you need to do the Frontend Exercise course.

It contains:

- the Connector “MyGo_SAP_RFC_Connector”
- the Data Types:
  - POITEM
  - ES_RETURN_BOM
  - POItemList

General Instructions

Here you will find general instructions about Simplifier deployment:

General Requirements for On-Premise-Installations

We support you with on-premise installations of the Simplifier on one of his own instances. To do that, we deliver a prepared Docker image to you. The image comes pre-configured and contains all the required components, including the Simplifier server in its most recent version.

The target instance must fulfill the following requirements:

- At least 8 GB RAM minimum, 12 GB recommended
- 64 BIT x86, server processor with 4 cores and at least 2 GHz per core
- At least 20 GB of free hard disk space
- Open ports: 80 (TCP), 443 (TCP) , 8090 (TCP)
- SMTP relay (port 587)
- SSL Certificate for encrypted connection
- Operating system:
    - Linux
        - In general, the Docker engine can run on all Linux versions with kernel version> = 3.10, but for the versions below, there are “official” releases.
        - If you are uncertain about the compatibility go to the Docker website.
          Tested Distributions:
            - Ubuntu
            - CentOS
            - Debian
            - Fedora
            - RHEL (Redhat Enterprise Linux) und SUSE Enterprise are officially supported only by paid docker variants
    - Mac
        - Install Docker for Mac
          Note: Our docker containers, respectively the database server, require a file system which can be case-sensitive under MacOS. Therefore, it may be necessary to create a seperate volume for the user data which is configured with the option “-v” when the container is started.

          The Simplifier MacOS Deployment is not recommend for production use, because of the limited support for container

    - Windows
      Install Docker for Windows
        - Windows 10 Professional
        - Windows Server 2016 (Please download the “Docker Edge” version on the website.)
          The runtime is given, but not as a Windows Service. The Docker Containers only stay up and running in a locked user session.

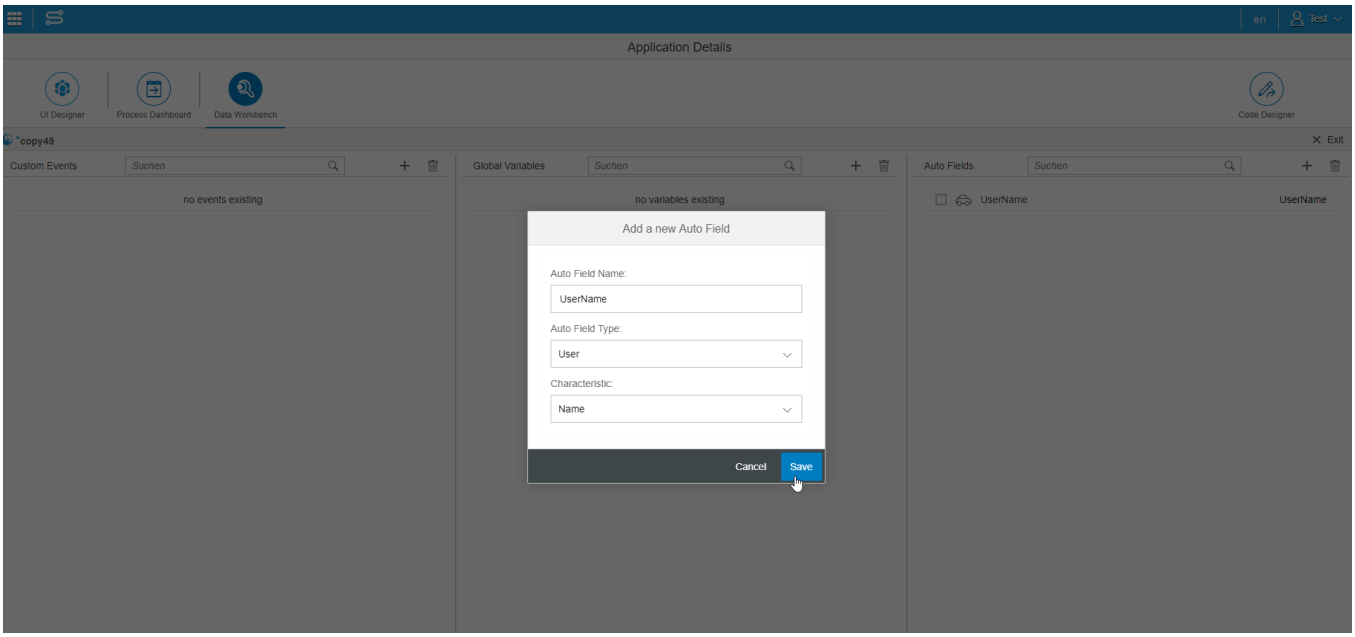
          The Simplifier Windows Deployment is not recommend for production use, because of the limited support for container

Depending on what kind of maintenance service level has been defined, we might need access to your machine in order to install updates or analyze logs. In this case, the target machine should fulfill the following additional requirements:

- Your server is connected to the Internet
- SSH daemon is installed and running (preferable on port 22)

Global Auto Fields

Auto Fields are automatically computed / filled fields. You can use them e.g. if you want to greet the user who is logged in with his/ her actual name or load the version number.



You can declare Autofields from five categories

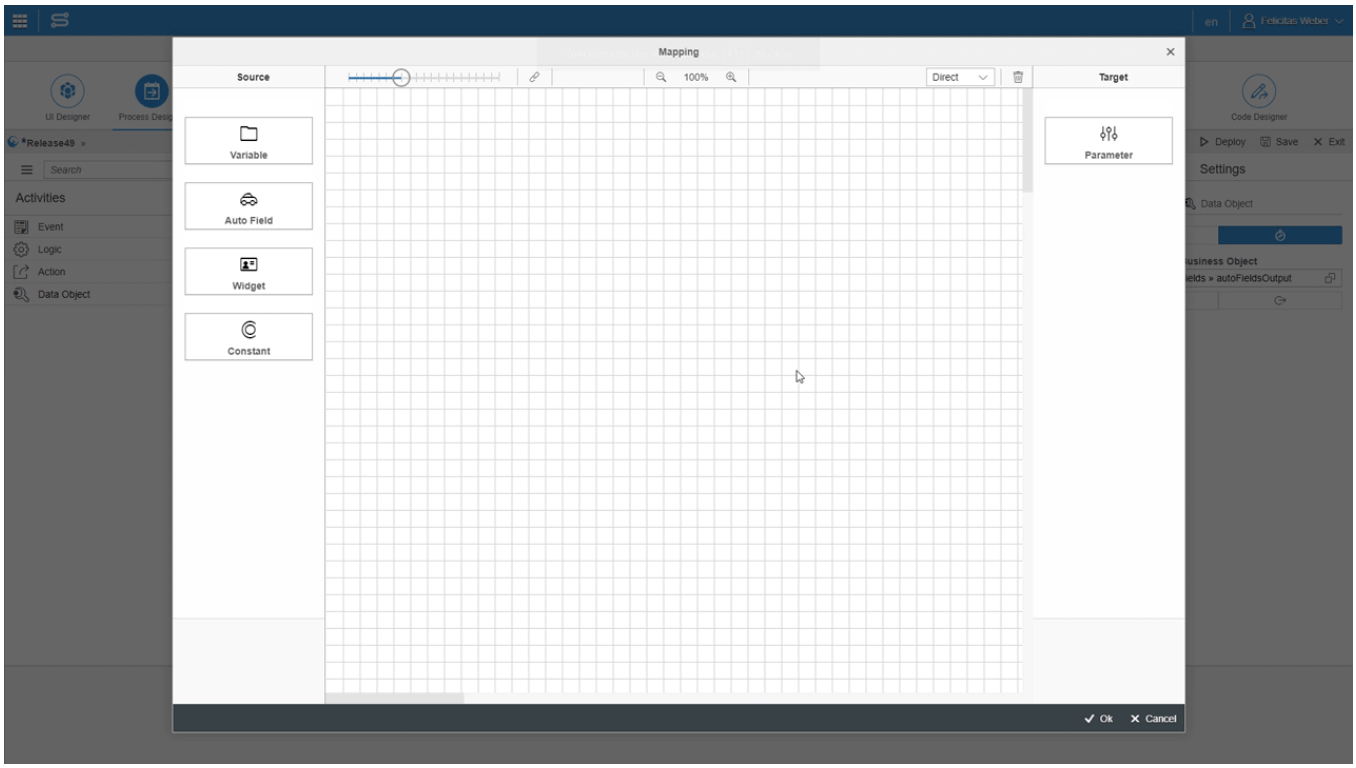
- Application
- User
- URL
- Geolocation
- Device

Category	Charactersistic	Description	Parameters
Application	Name	The name of the Application.	–
	Version	The current version of the Application. If the App is not yet released, it is stated as “n/a”	–
User	Name	The currently logged in user name.	–
URL	Query Parameter	A freely definable URL Query Parameter which will be provided in the App.	Name of the Parameter
	Host	The domain/hostname can be read out in a configuration.	–
Device	Online	Holds the current connection state.	–
	Mobile Client	Tells you if your Application is running on the Cordova Client.	–

Screen orientation	The screen orientation is set.	–
Device Type	The Autofield can hold this values: desktop, phone, tablet, watch and smartglass.	–

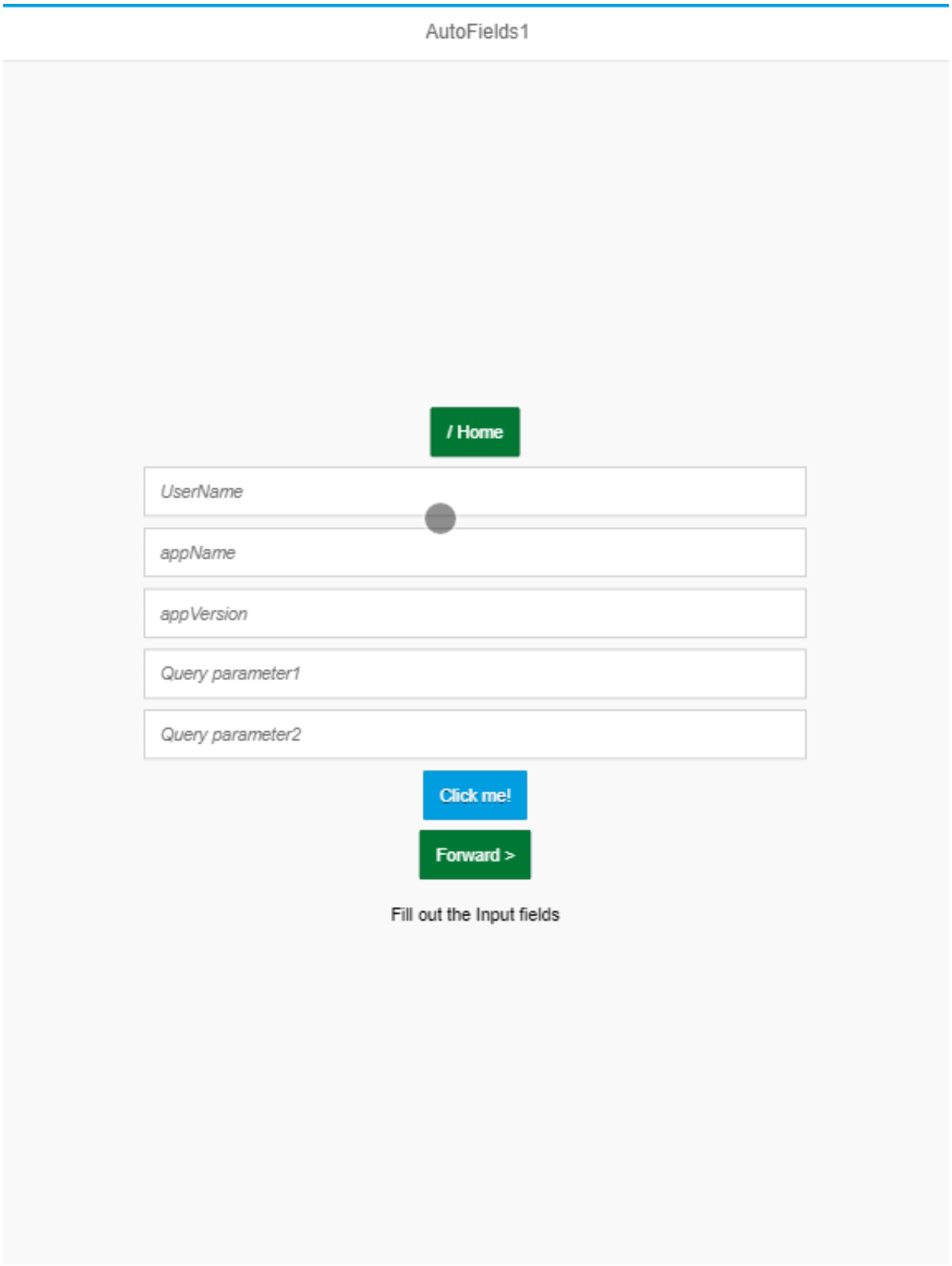
How to use Auto Fields

You can use Auto Fields within a Data Object. In this example, we created an Auto Field that automatically gets your user name. Then we mapped it with the Input Parameter “userName” of a preconfigured Business Object.



We configured some more Auto Fields like “applicationVersion” or 2 URL Parameter. The result is shown below:

At first, you can see the former way of typing in the information by hand. On the second Screen, we used the Auto Fields and you get the information automatically, simply by clicking on the button.


AutoFields1
/ Home
UserName
appName
appVersion
Query parameter1
Query parameter2
Click me!
Forward >
Fill out the Input fields

Global Events

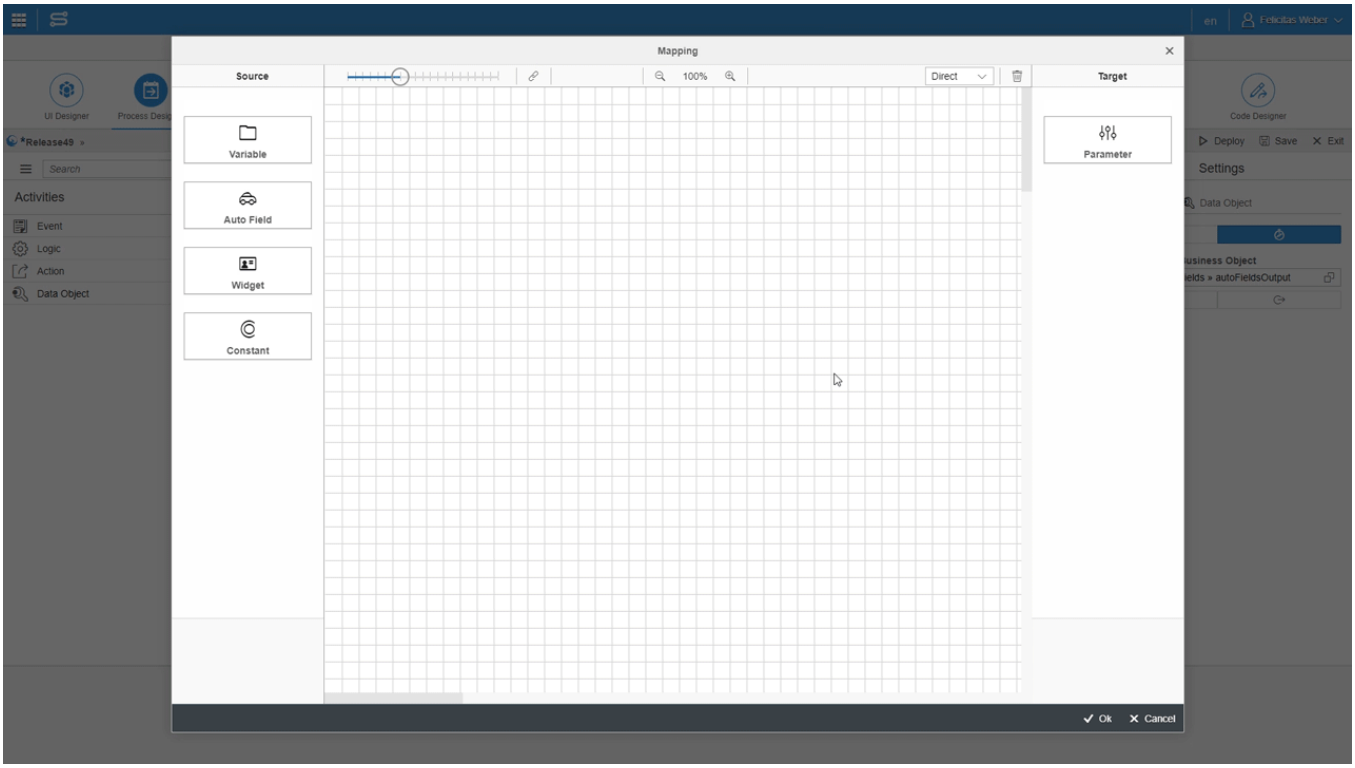
Global Events can be imagined as tunnels which connect different User Stories or processes. If you publish an Event, you open the entrance to the tunnel and by subscribing to this particular Event, you create the exit of the tunnel. This enables you to jump between the Stories and helps you to have a clearly laid out working space.

Go to Events for an example use of global Events.

Global Variables

You can use global Variables as a container to buffer data. E.g. if a Connector returns a lot of data and you would like to use some of it later in your work process, you can save the parameter as Variables and map them later.

You can reference Variables in Data Objects as In- or Output parameter. To do so, drag a Variable (that you've created previously in the Data Workbench) from the toolbar in the mapping dialog.

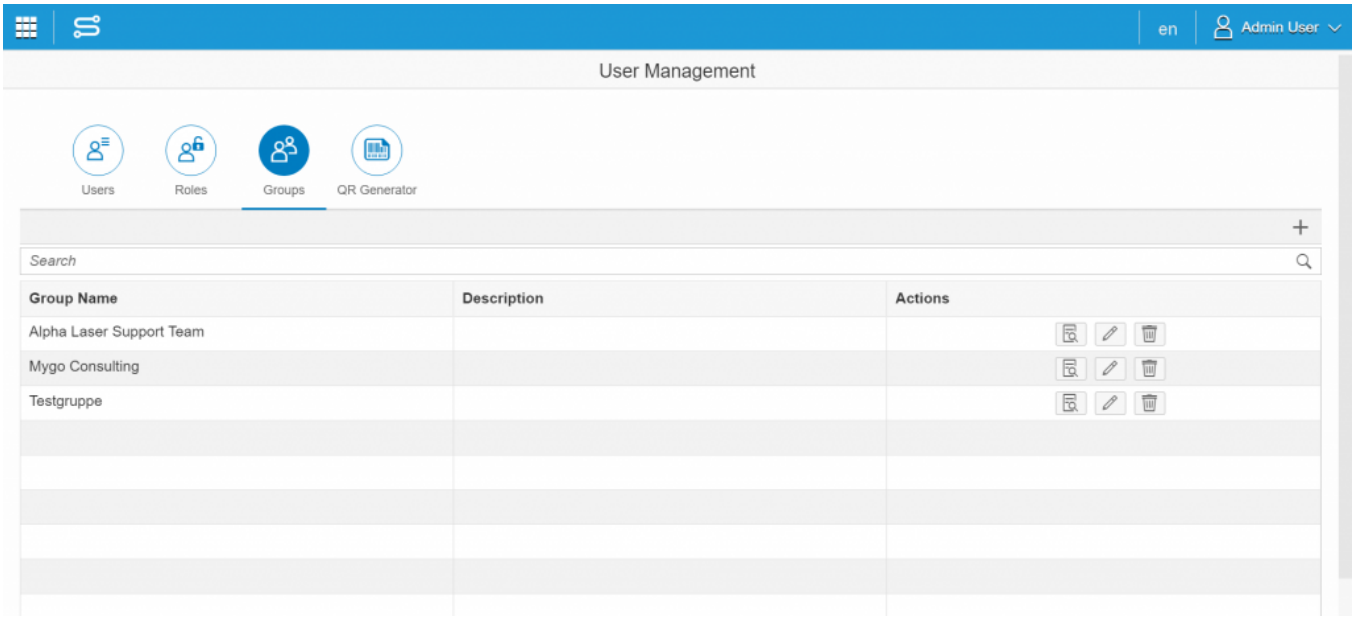


Glossar

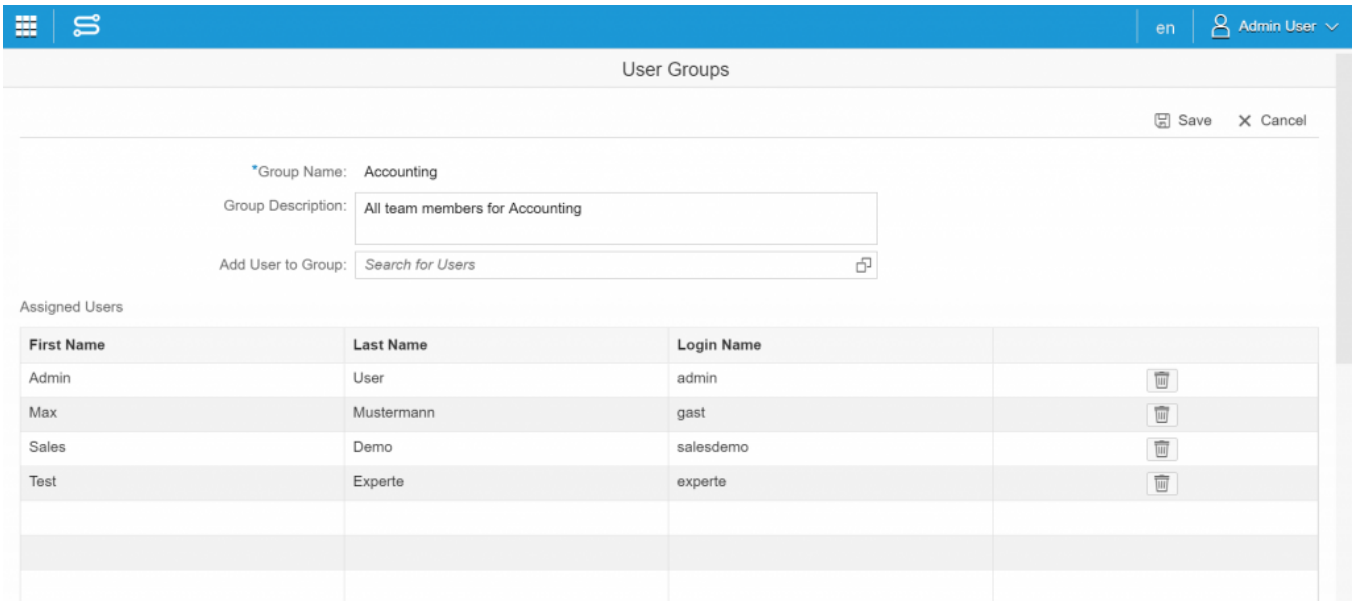
Here you will find general and Simplifier specific abbreviations, technical terms and their meaning.

Group Overview

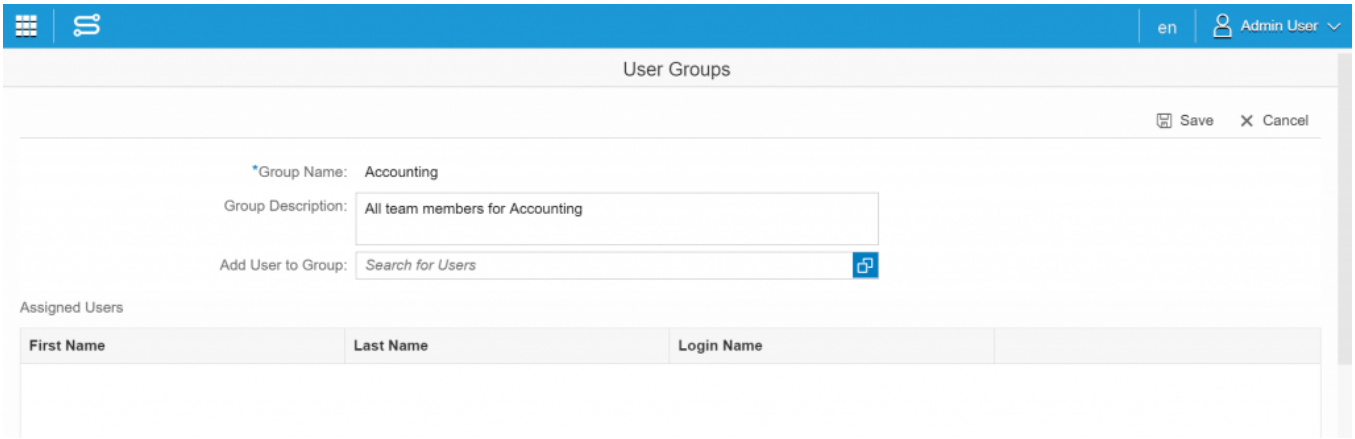
A group contains several users and could be used for workflow logic in business apps like informing a team via email or push notification about a certain event or task.



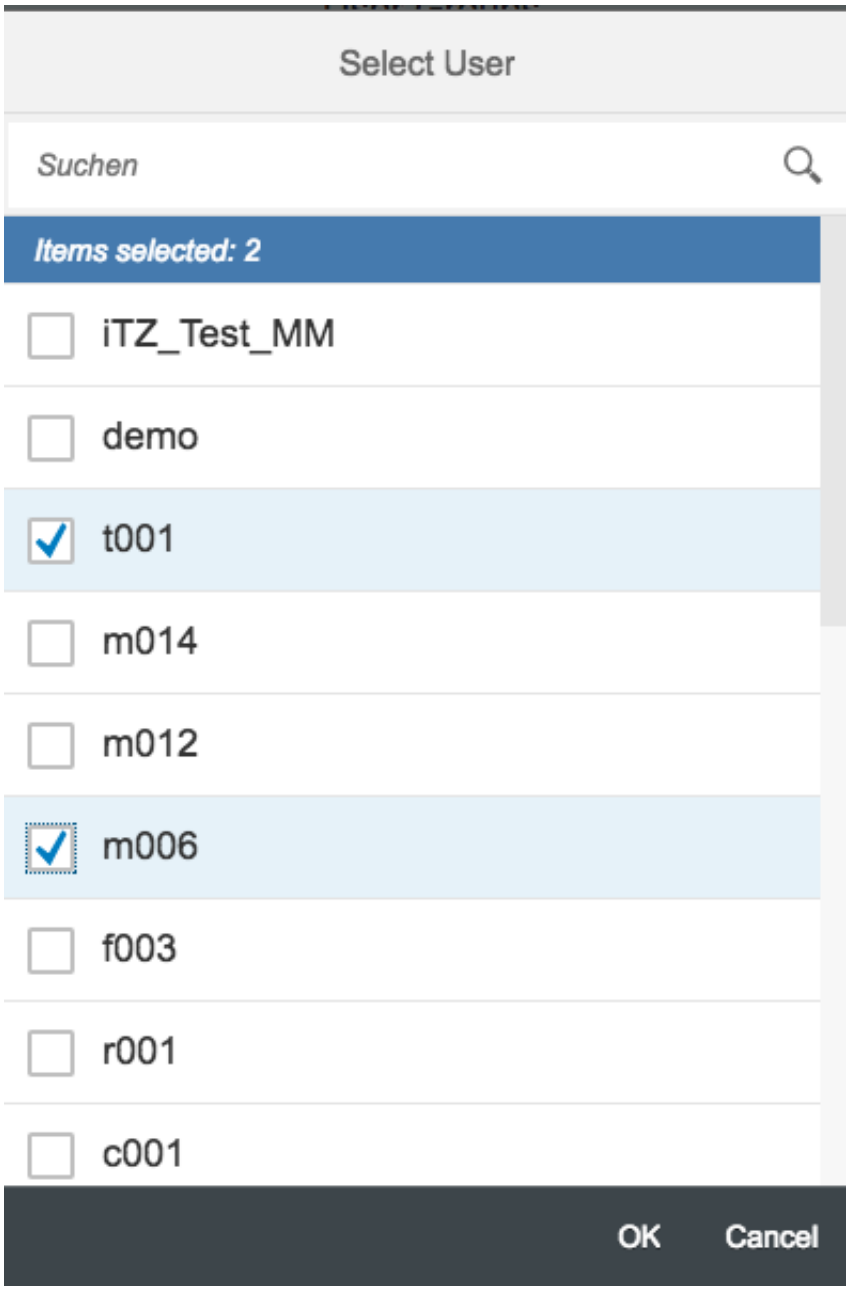
Details View of a Group



To create a new group you have to specify a unique name and an optional description e.g. for a team or special task force group.



To add users, click into the “Add User to Group” Field and search for specific usernames. Mark several users and click OK to add them.


Select User
Suchen
Items selected: 2
iTZ_Test_MM
demo
t001
m014
m012
m006
f003
r001
c001
OK
Cancel

Handling & Updating an On-Premise Installation

Docker Basic Commands

Start the Simplifier container:

```
$ docker start simplifier
```

Stop the Simplifier container:

```
$ docker stop simplifier
```

Restart the Simplifier container:

```
$ docker restart simplifier
```

Create a backup of the complete Simplifier data directory, e.g.

```
$ docker stop simplifier
```

```
$ tar cvzf simplifier_backup.tar.gz /home/simplifier
```

Updating

In case of updates, we will prepare a new docker image for you, preserving your personal settings. Please download the image to a temporary directory of your choice (e.g. /tmp) and change into the directory. Finally unpack and load it, as described in steps 3-4 in the installation instruction.

To perform the update, proceed as follows:

1. Stop the container and remove it from Docker. Take care NOT to remove your data directory /home/simplifier/data !
2. Perform the following commands in order:

```
$ docker stop simplifier
```

```
$ docker rm simplifier
```

```
$ docker run ––name simplifier {additional options as in step 6 before}
```

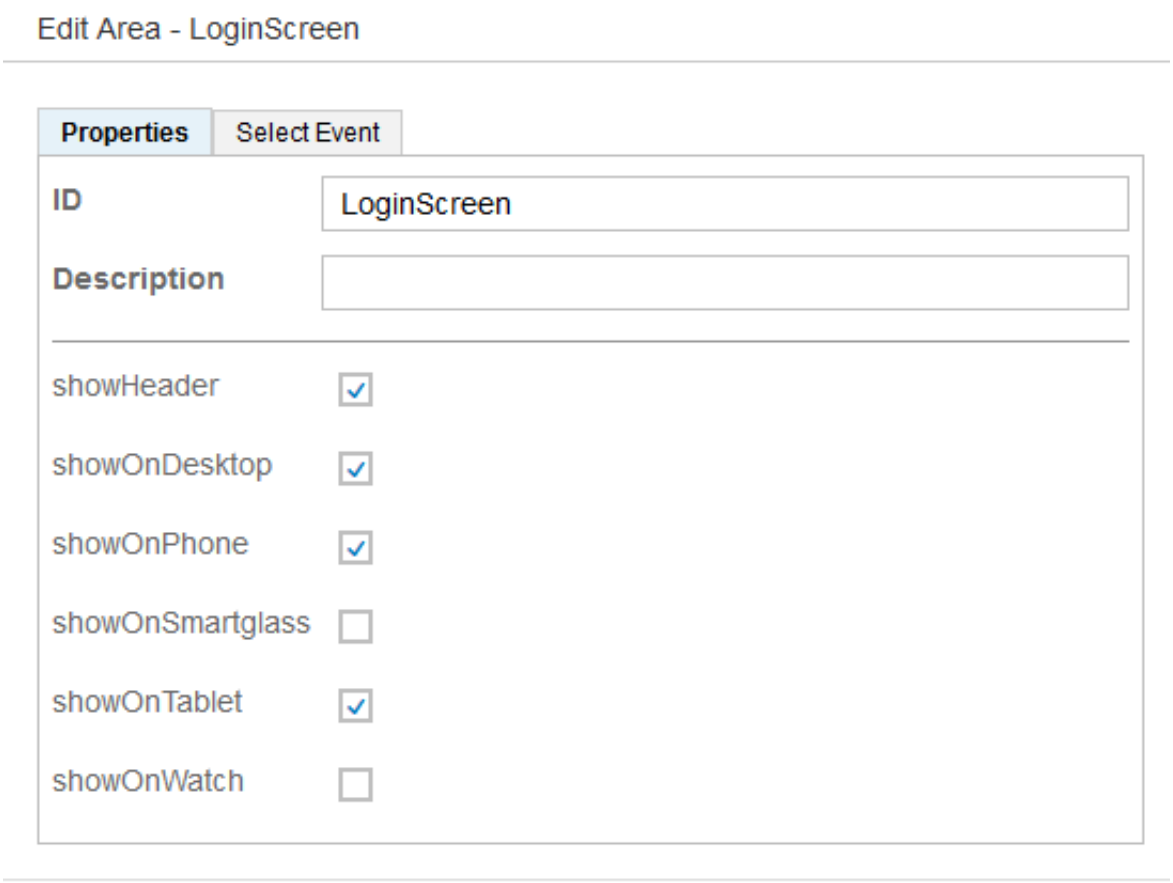
Hardware

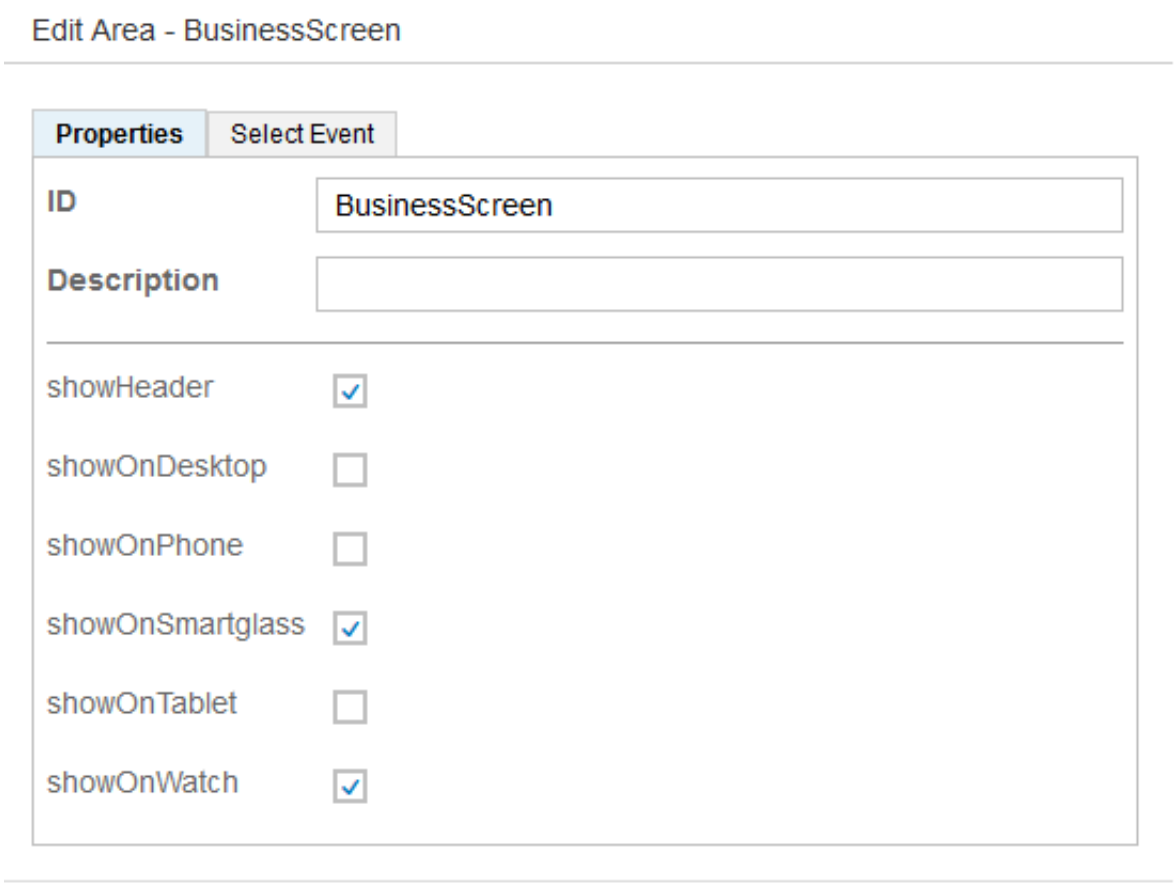
OEM	Hardware Description	H/W Revision	Connector Type
HARTING	HARTING IIC MICA (Modular Industry Computing Architecture) makes it possible to temporarily save, evaluate and process data in the immediate vicinity of machinery and equipment. With its modular open platform, the HARTING IIC MICA can be customised with custom hardware, software and interfaces – to suit your individual requirements for Integrated Industry.	All Revisions	MQTT Client
Sartorius	Cubis Individual Balance	All Revisions	HTTP REST

How to Customize Screens for Different Devices

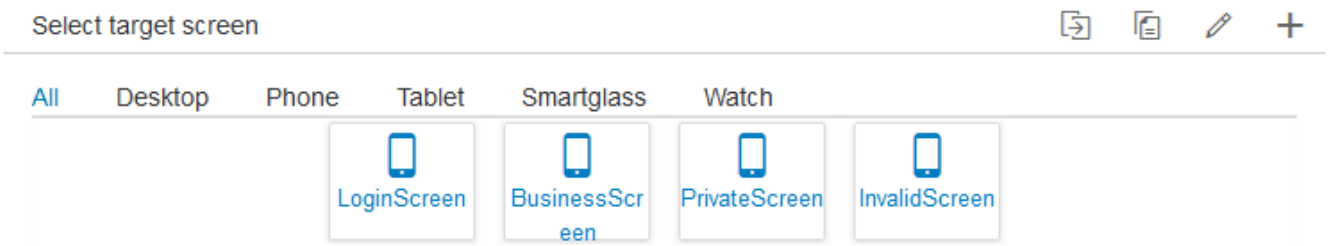
You can design your screens individually for different devices, e.g. a login screen that is optimized for mobile phones and one for smartwatches.

Simply select as screen property the devices you want to address and e.g. create a second screen for the alternatives.



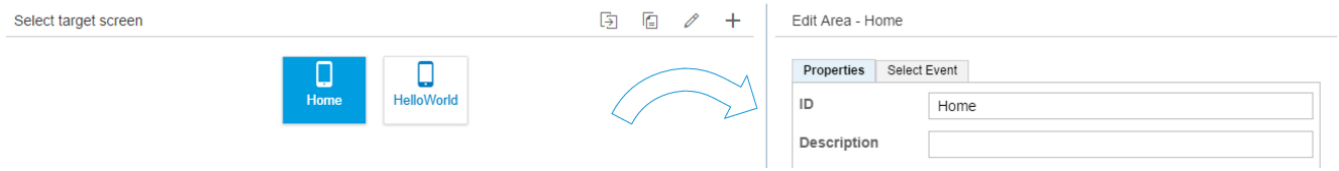


Now you can filter your screens and work precisely on one at a time.



How to rename your Screens

To rename screens, select it and change the ID within the properties section.



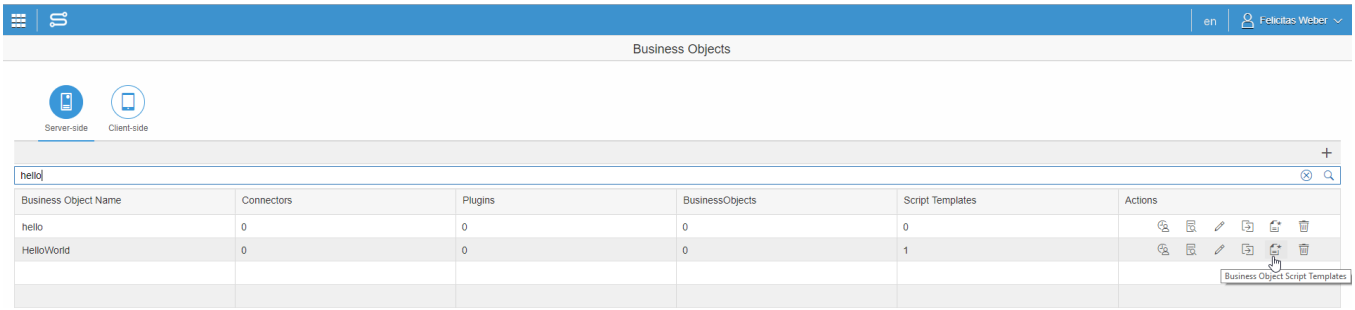
Use the Description field to comment and annotate the screen. This helps to identify the purpose of the screen later on.

How to use Widgets in the UI Designer

In the follwing sub-sites, you can get some information on how to use Widgets in the UI Designer, e.g. adding an Icon or changing the order of the Widgets.

Implement logic to your Business Object

The logic of a Business Object is implemented via Script Templates. Each Business Object can hold as many Script Templates as wanted. Click on the “Script” Icon to add some Templates.

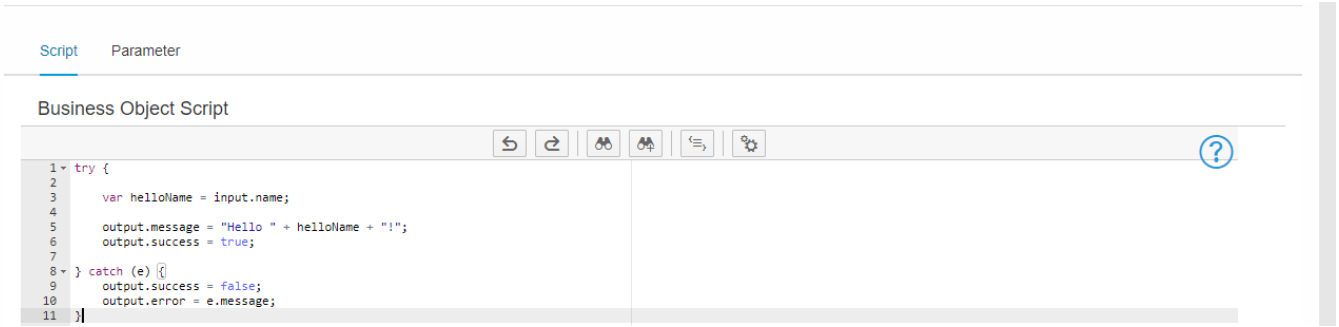


Creating a Script Template involves writing a method via JavaScript and editing parameters. In the Script Editor you can code logic that can be called in the edit mode of a user story (Process Designer).

The content corresponds to the inner body of a Javascript function. In other words: your main code block must not be wrapped into a separate function definition, but rests on the top level context. It’s nevertheless possible to define sub-functions on top of your main code block.

It’s best practice to wrap your main code block in a try-catch-block to handle possible errors.

Let’s have a typical hello world example:



In the toolbar above you have several possibilities: undo, redo, search, search and replace, format code and settings.

This example reads a name which is provided to the Business Object as an input parameter, compiles it into a greeting message and writes the result to the output.

Take a look at the try-catch-block surrounding the main code section:
If no error occurs, the upper part of the code inside the “try”-section will execute and return the greeting message and output.success = true. However, if any error occurs, the function will jump down into the “catch” section and return output.success = false and assign any details of the failure to attribute output.error.

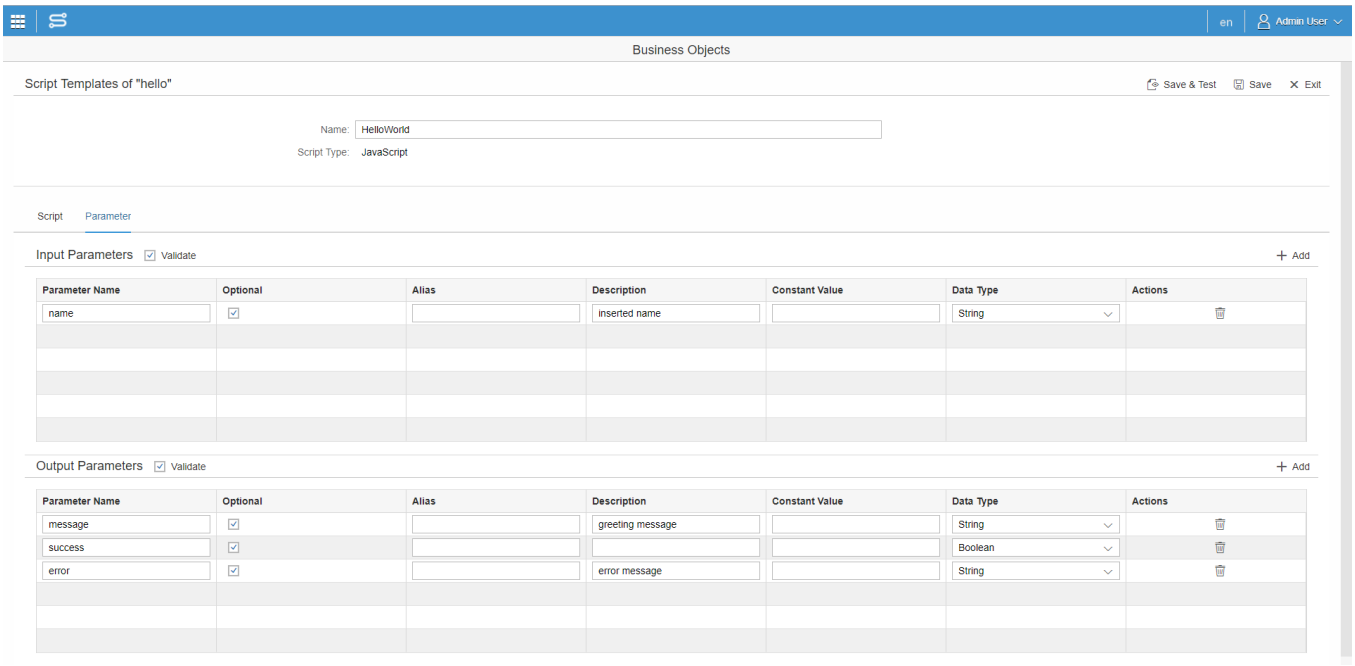
In order to use the Script Template correctly, you have to add the same Input and Output parameters you used in the payload. Those parameters will be shown in the mapping dialogs in the edit mode of a user story (Process Designer). Please note the correspondent handling of in- and output through the (JSON) objects “input” and “output”. Both of them may carry arbitrary attributes.

In this example the object “input” carries the attribute:

- *input.name* to read the inserted name.

The “output” object carries the attributes:

- *output.message* to send a greeting message.
- *output.success (true/false)* to indicate whether the Script Template executed successfully.
- *output.error* to hold the root cause of an error in case of failure.



Validate

You can validate the Input and Output parameter in the backend. It validates:

- Base type against type security
- Domain type against security and restrictions
- Structures against type security and underlying property types
- Collections against type security and the underlying types / property categories

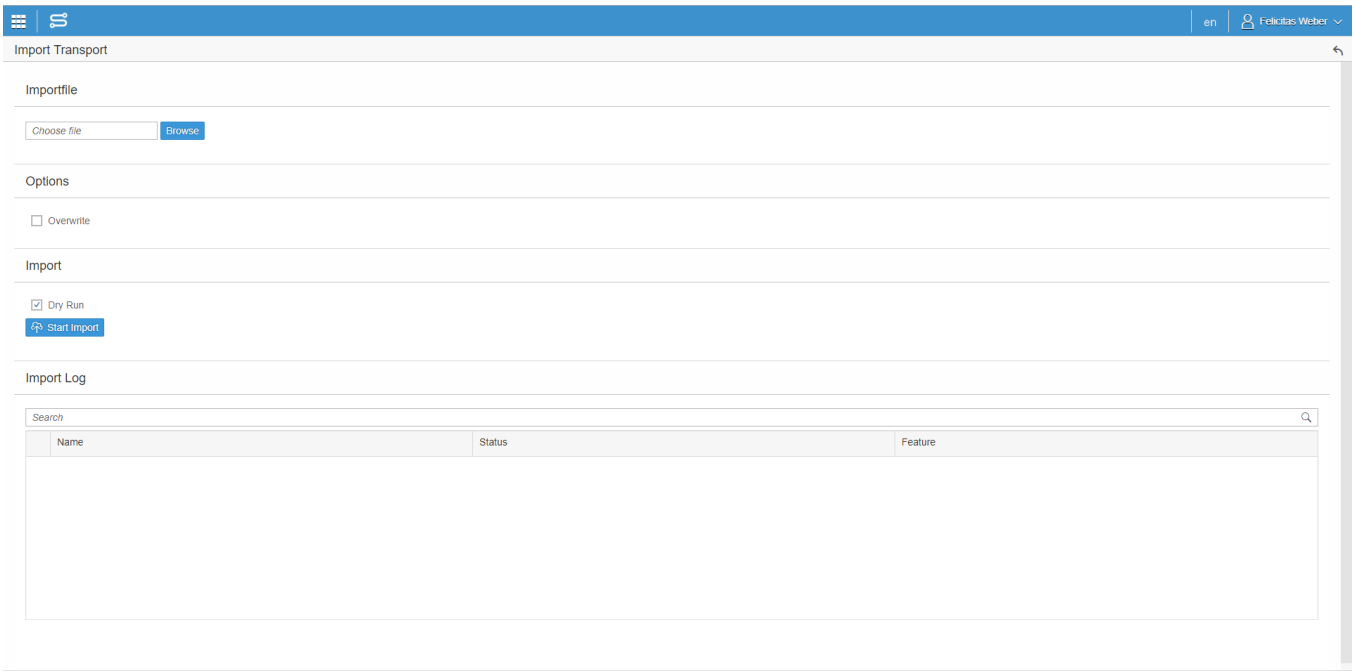
If the validation is **not** successful, the client is notified of all failed validations and it's written to the Business Object log or System log at the same time.

For every new Business Object, this flag is set by default. Already existing Business Objects **do not** have this checkbox flagged to guarantee the compatibility.

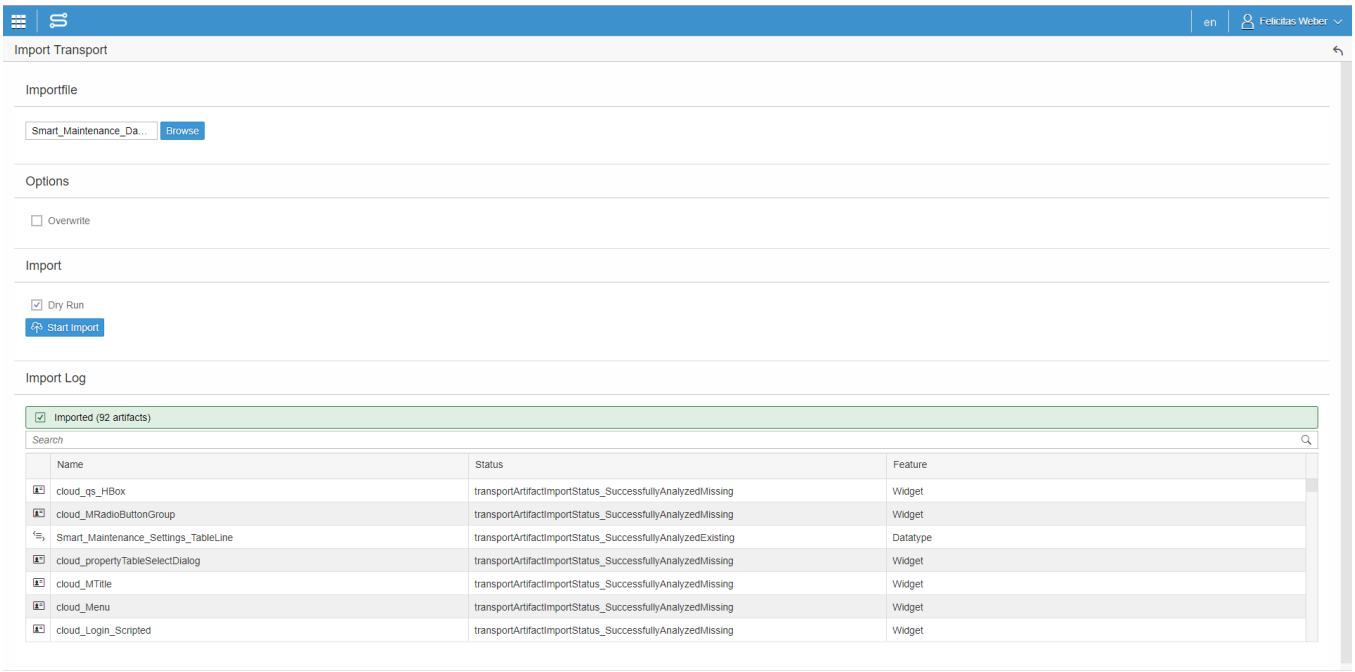
NOTE You have the possibility to declare parameters as optional. When declaring a parameter as non optional, the validation will fail if the parameter is not provided.

Import Transport

To import a file to your Simplifier instance click on the “Import Transport” tile. You will be forwarded to an import overview page.



Choose your file you want to import. If you select ‘Dry Run’, it simulates the import.



If the transport works correctly, you can import it with the option ‘Overwrite’. When certain artifacts already exist and you do not select ‘Overwrite’, only the new artifacts are transported.

Installation PDF Plugin

Configuration

To use the pdf Plugin, you have to configure it first.
Copy the file “settings.conf.dist” from the directory “plugins/pdfPlugin/src/main/resources”, save it as “settings.conf” and adjust it as follows:

In order to start the conversion, you need to install the program wkhtmltopdf on your operation system. The path to the wkhtmltopdf executable must be stated in the “settings.conf” file. Furthermore you need two folders, one to file your template and the other for the temporary data during the conversion. You can either use relative or absolute paths for the folders.

For example:

settings.conf

```
pdfPlugin {
     storageDir = "templates"
     tempDir = "tmp"
     wkhtmltopdf = "C:/Program Files/wkhtmltopdf/bin/wkhtmltopdf.exe"
}

....
```

NOTE:

If you use wkhtmltopdf on a Linux without the X11 Server, the error “**wkhtmltopdf: cannot connect to X server**” may occur.

In this case you need to install the program “xvfb” via the package manager to simulate the X11 server.

Create a wrapper (e.g. /usr/local/bin/wkhtmltopdf-xvfb) for the “wkhtmltopdf” program and write the path in the PdfPlugin Config.

wkhtmltopdf-xvfb

```
<#!/bin/bash>
xvfb-run --server-args="-screen 0, 1024x768x24"/usr/bin/wkhtmltopdf$*
```

Plugin Execution

The Plugin is located in the directory: plugins/pdfPlugins. It can be activated with the SBT/Activator via a “run” command. The STDIN command “stop” ends the Plugin execution.

You can adapt the logback-configuration file “plugins/pdfPlugin/src/main/resources/logback.xml” to configure the log output or display it in another file.

Installing an On-Premise Image

We always prepare an all-in-one Docker image for our customers which contains all required components.

Given a target machine that matches the requirements described in the previous chapter, the installation is quite easy:

1. Create the directory which will host all external user-specific data:

```
$ mkdir -p /opt/simplifier/data

$ export SIMPLIFIER_DIR="/opt/simplifier/data"
```

2. Copy the downloaded file with ending .tar.gz to a temporary directory on the target machine, e.g. /tmp and cd to this directory.

```
$ wget -O <filename>.tar.gz
```

3. Unpack the file in place:

```
$ tar xzvf <filename>.tar.gz
```

You will get two files: one readme.txt and the docker image with the ending .tar.

4. Inside the directory which contains the unpacked file, run the following commands as root- (super-) user:

```
$ docker load -i <imagefile.tar>
```

5. Install SSL certificates:

```
$ mkdir -p $SIMPLIFIER_DIR/certs

$ cp <certificate.crt> $SIMPLIFIER_DIR/certs/default.crt

$ cp <keyfile.key> $SIMPLIFIER_DIR/certs/default.key
```

6. Run docker image:

Alternative 1: with SSL/Certificates

```
$ docker run ––name simplifier -v $SIMPLIFIER_DIR:/opt/simplifier/data \

-p 80:80 -p 443:443 -p 8090:8090 \

-d docker.itizzimo.com/simplifier-onpremise
```

Alternative 2: without SSL/Certificates

```
$ docker run ––name simplifier -v $SIMPLIFIER_DIR:/opt/simplifier/data \

-p 80:8080 -p 8090:8091 \

-d docker.itizzimo.com/simplifier-onpremise
```

7. In case you want to install from DockerHub, use itizzimo/simplifier as image suffix. The docker engine automatically downloads the image from DockerHub and you can skip step 2., 3. and 4.

Example without SSL/Certificates:

```
$ docker run ––name simplifier -v $SIMPLIFIER_DIR:/opt/simplifier/data \

-p 80:8080 -p 8090:8091 \

-d itizzimo/simplifier
```

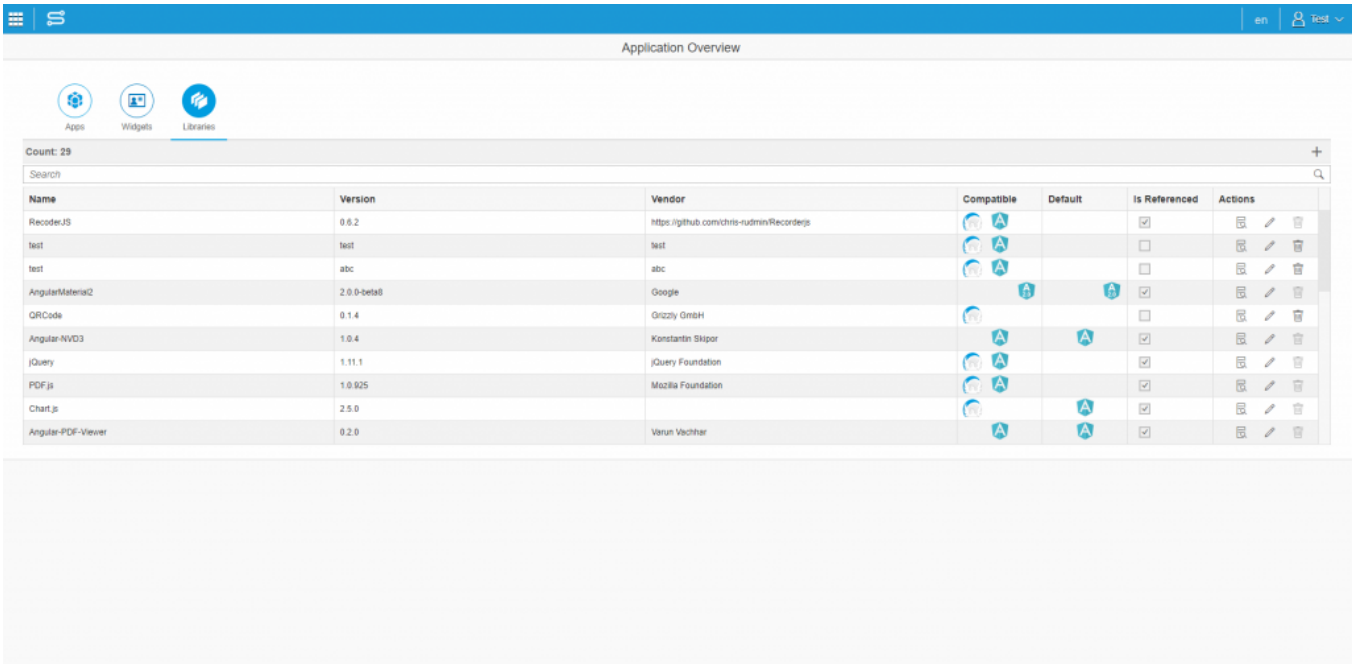
8. Open your browser

Now use your browser at your Client Computer to access http(s)://<IP> or <FQDN>/UserInterface. The Simplifier will prompt a license dialog. After pasting that license
you can start configuring Apps in the AdminUI.

Integration of external Libraries

Sometimes it is necessary to add an extra library to your app, e.g. if you want to display some special charts. You can upload and manage those external libraries under the “Libraries” tab in the Application tile.

If you want to know how to implement them into your application, go to “Assigning Libraries To Apps”



In the list overview you get information about:

- the name
- the version info
- the vendor
- if it is compatible to Angular JS, Angular 2 or UI5
- if it was set as a default for Angular JS, Angular 2 or UI5
- if the library is referenced in any application

Standard Equipment

The Simplifier provides the following libraries by default:

App Technology	Library	Version	Type
UI5	OpenUI5	1.38.2	Direct
AngularJS	AngularJS	1.4.7	Direct
Angular	Angular Material	1.0.0-rc4	Direct
Angular	Angular-Fittext	3.3.3	Direct
Angular	Angular-PDF-Viewer	0.2.0	Direct
Angular	Angular-NVD3	1.0.4	Direct

Angular	jQuery	1.11.1	Dependency
Angular	PDF.js	1.0.925	Dependency
Angular	NVD3	1.8.1	Dependency
Angular 2	Angular2	4.1.3	Direct
Angular 2	Ionic	3.4.2	Dependency

Integration of Libraries - addAfterInitHandler

addAfterInitHandler

Parameter	Type	Description
Handler	Function	Callback function, which is called after all scripts have been loaded completely.

Integration of Libraries - addBeforeInitHandler

addBeforeInitHandler

Parameter	Type	Description
Handler	Function	Callback function, which is called immediately before the loading of the script begins.

Integration of Libraries - addScript

To integrate the library with a js code snippet, use the following parameter:

JS code to include:

```
addScript('js/d3.min.js','d3');
addScript('js/nv.d3.min.js','nvd3',['d3']);
```

addScript(ScriptPath, Name, Dependencies)

Parameter	Type	Description
ScriptPath	String	Relative path in the uploaded ZIP structure to the .js file you want to include (e.g. src/js/includedScript.js).
Name	String	Name of the library you can use to access the .js file (e.g. includedScript). By using “includedScript” in your script code you can now use all methods of your integrated library.
Dependencies	Array<String>	Dependent scripts (refers to the parameter “name” of “addScript”.) It guarantees, that all dependencies are loaded beforehand. Use this if your library needs other libraries to work properly.

It is important to ensure that all scripts specified under “dependencies” are either integrated into the same library, or a dependency is set on the library in which the script is integrated.

Integration of Libraries - addStyle

To integrate the library with a js code snippet, use the following parameter:

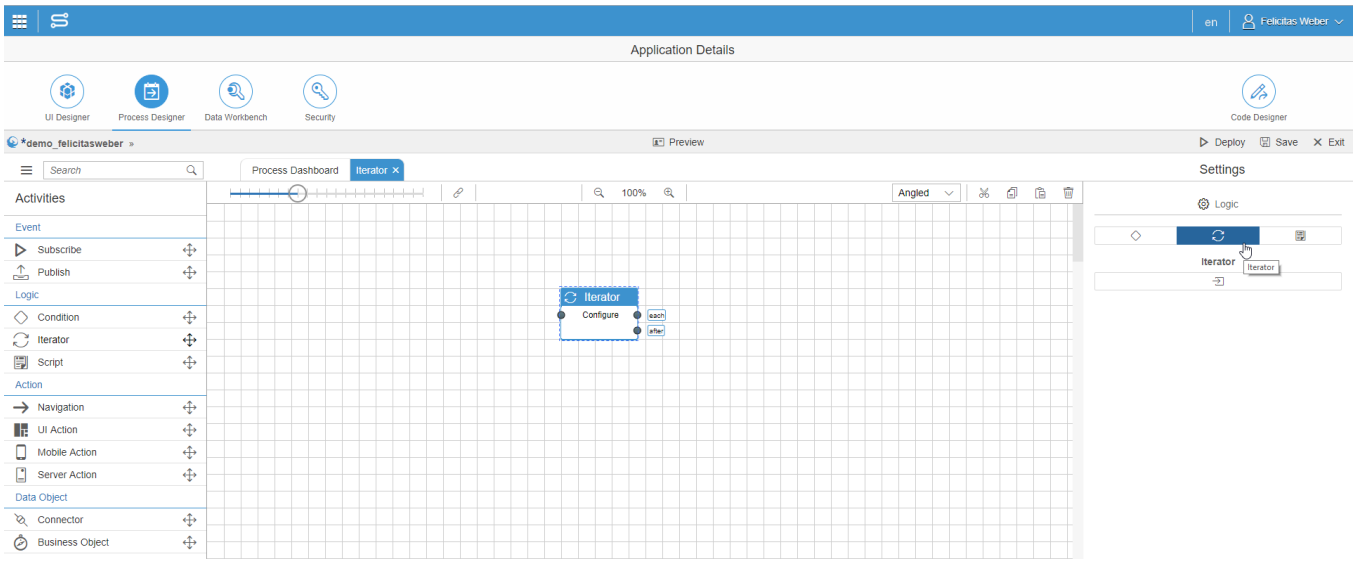
addSyle

Parameter	Type	Description
StyleURL	String	Relative path to the uploaded ZIP structure of the library.
Name	String	Style name (optional).

Iterator

An iterator is an object that sequentially passes data structures. It respectively returns the following element and determines if there are another elements following.

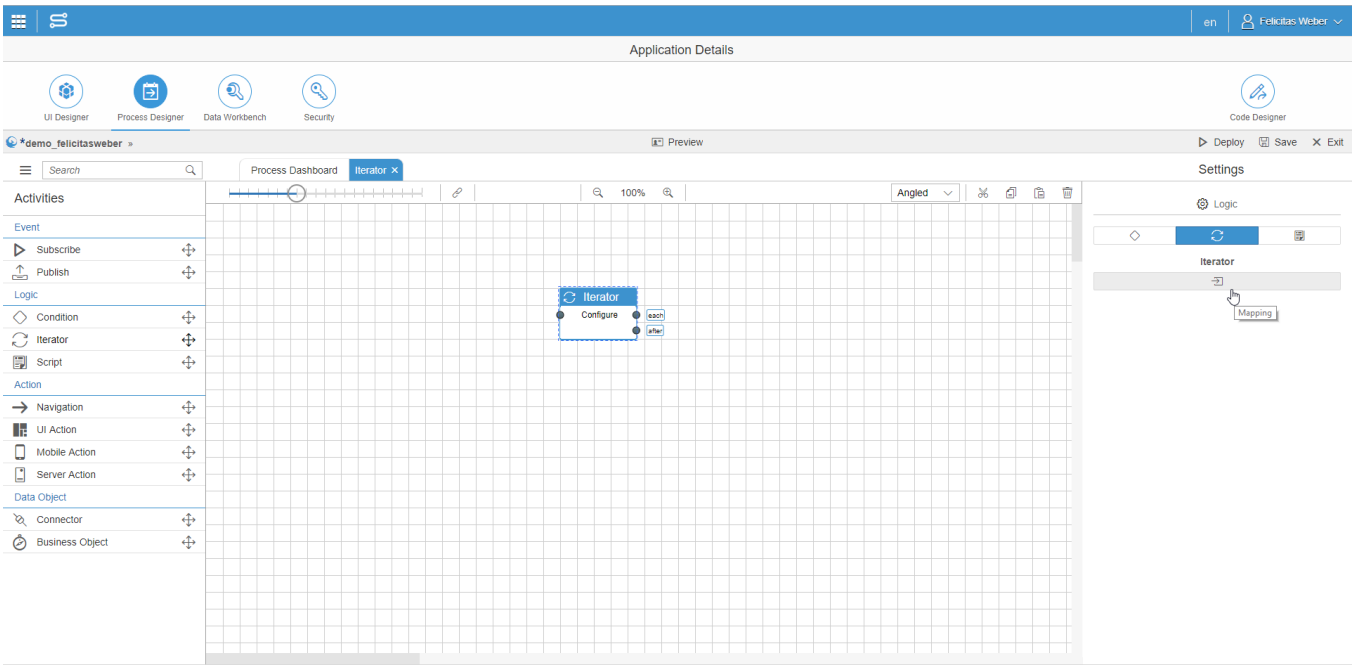
With the Iterator Activity you iterate over a variable. It has two output ports: each and after.

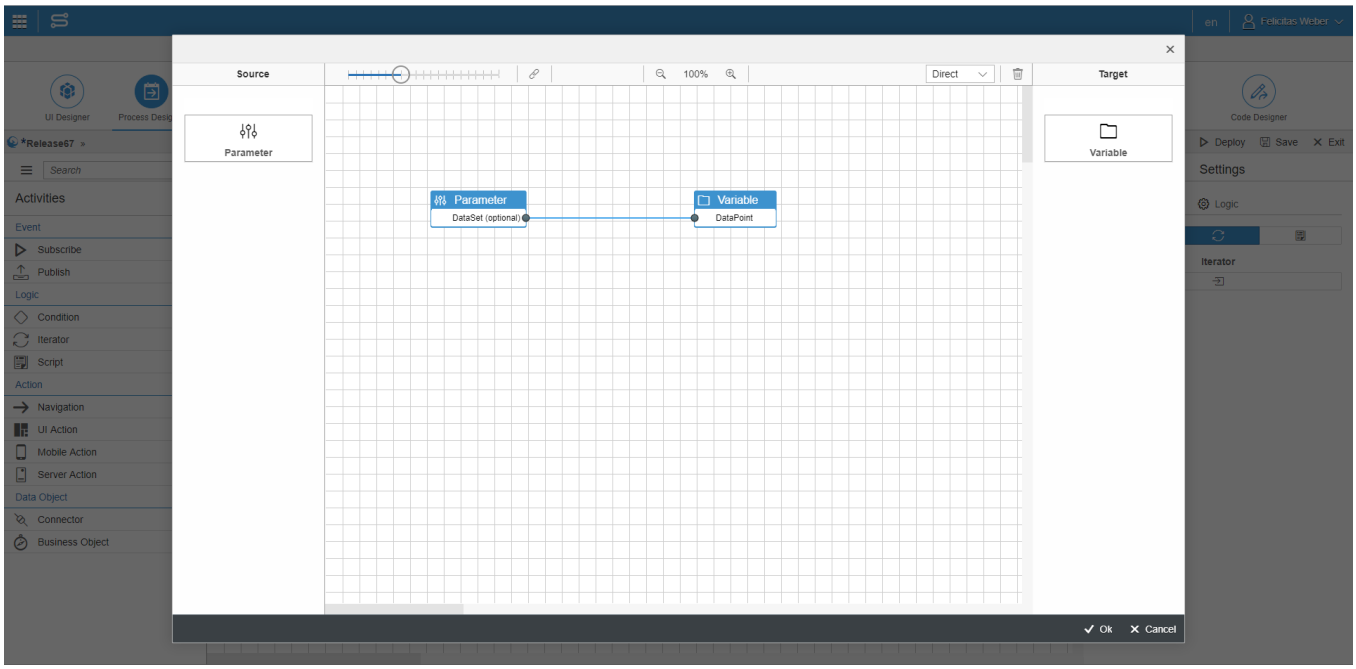


Port	Description
each	For every Iteration the following process will be executed.
after	After all Iterations ended the following process will be executed.

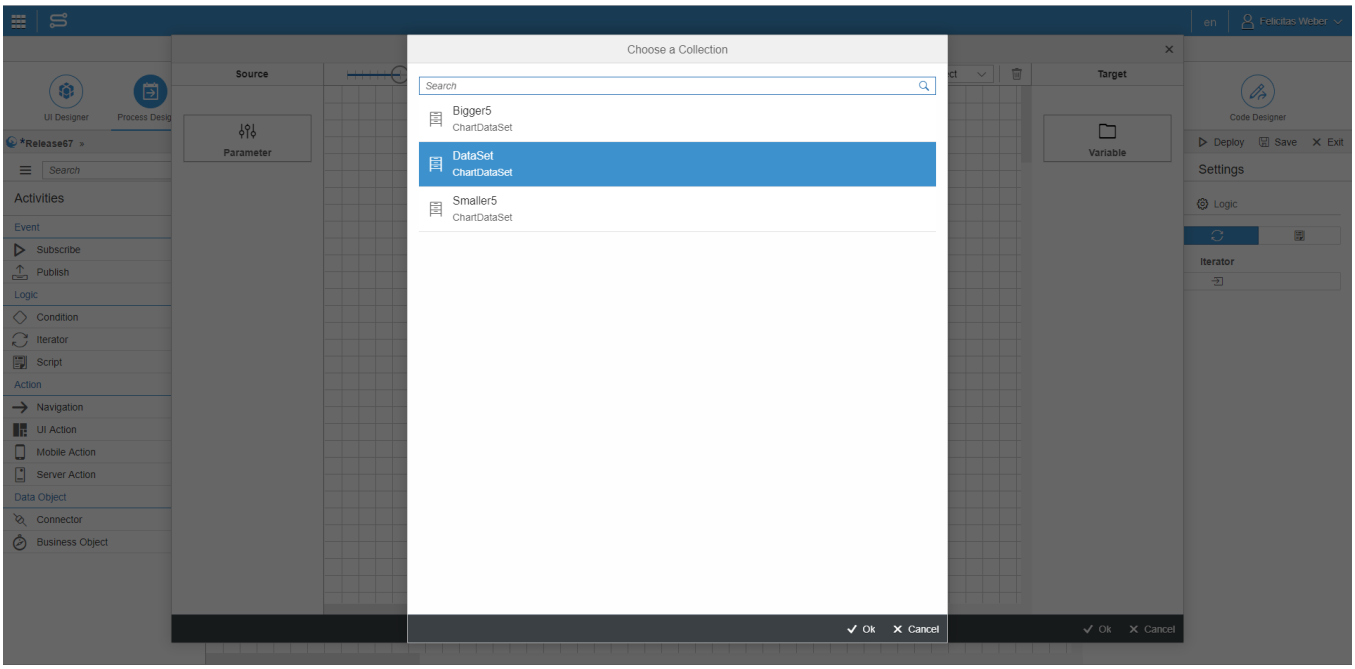
The Iterator selection helper can be accessed by opening the mapping dialog. The mapping dialog can also be opened by double clicking on the shape.

You can drag and drop the parameter in the drawing area in the middle. By double clicking on the parameter you can select the collection you want to iterate.


en
Felicitas Weber
Application Details
UI Designer
Process Designer
Data Workbench
Security
Code Designer
*demo_felicitasweber
Preview
Deploy
Save
Exit
Search
Process Dashboard
Iterator
Settings
Activities
100%
Angled
Logic
Event
Subscribe
Publish
Logic
Condition
Iterator
Script
Action
Navigation
UI Action
Mobile Action
Server Action
Data Object
Connector
Business Object
Iterator
Configure
each
after
Iterator
Mapping



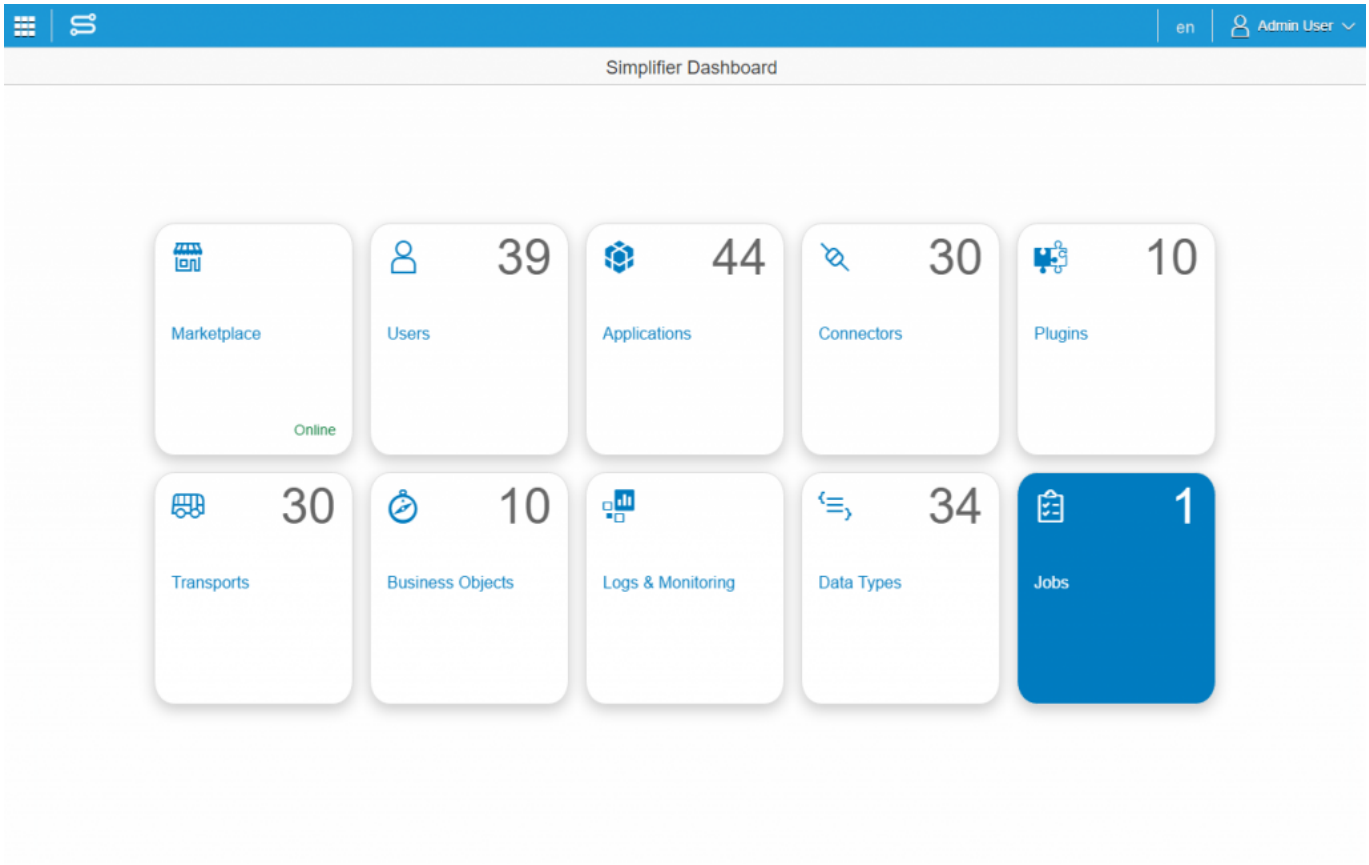
The mapping dialog is aware of its type and offers only collection variables that are defined in the app.



By opening the Mapping Dialog you can drag and drop the element on the left and on the right in the middle. By double clicking on it, you find all paramter/variables that can be mapped listed in a pop up.

Jobs

Jobs are server side executed and recurring tasks which use Business Objects as logic. They can be found by clicking on the “Jobs” tile on the Dashboard.



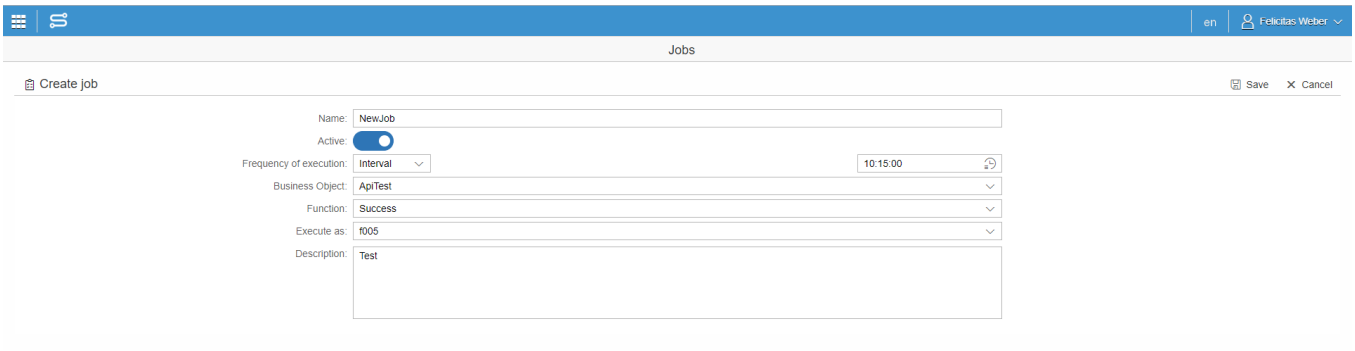
They are used to automate Business Objects. Commands can be defined and will be executed by the Simplifier on a periodic basis. You may schedule the jobs in order to trigger them without further user intervention.

The red and green icons within the **overview** indicate whether the job is active. The red icons signify that it’s inactive and the green icons signify that it’s active.

To create a new job, you have to name it and set an interval after which the job is called in repeatedly. Selecting the Business Object and the function provides the job with functionality. Furthermore you can choose the user who should execute the job (in case he has all permissions required). The active time zone on the server applies to all dates and time settings. You can choose between the 5 following intervals:

Interval

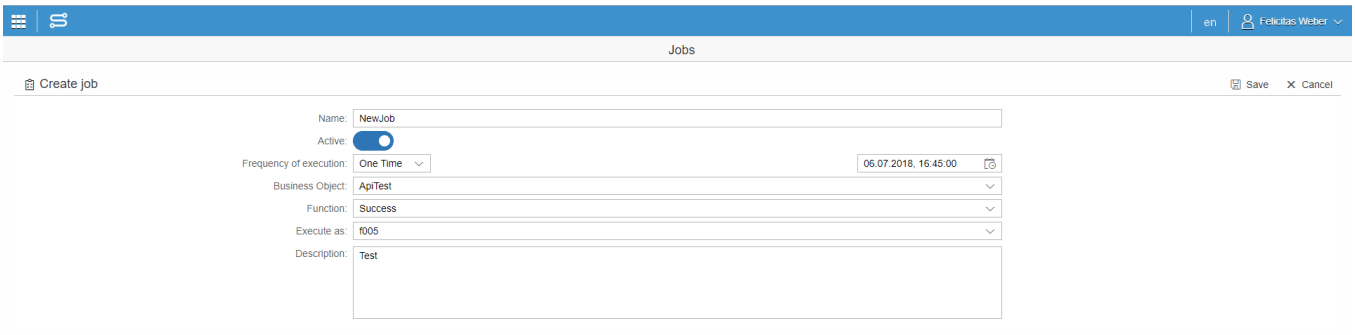
It’s an recurring interval at which the job is executed in the format <hh>:<mm>:<ss>. If the new execution overlaps with a previous run, it’s only started after it has been completed and after a further interval.



One Time

With this you can set an actual date <dd>.<MM>.<yyyy> and time <hh>:<mm>:<ss>.

If the selected time is in the past, the job will **not** be executed!

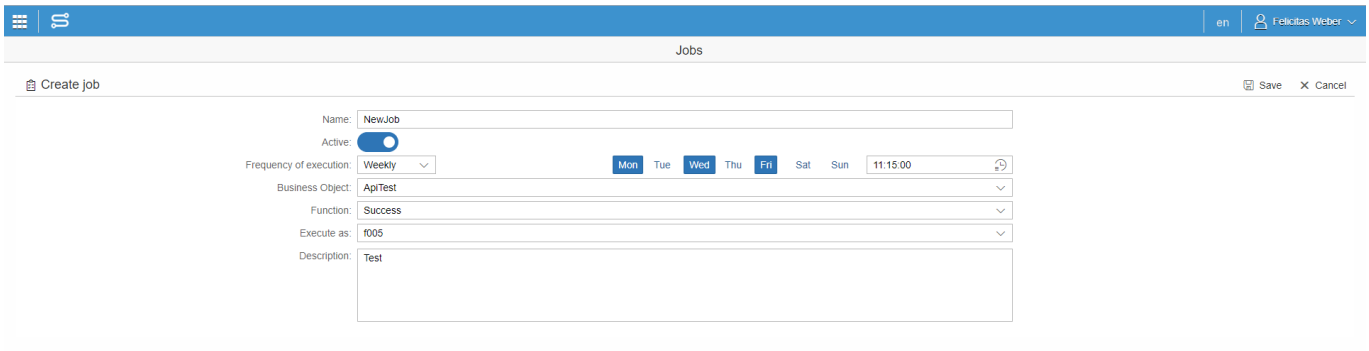


Daily

You can set a daily time in the format: <hour>:<minute>:<second>

Weekly

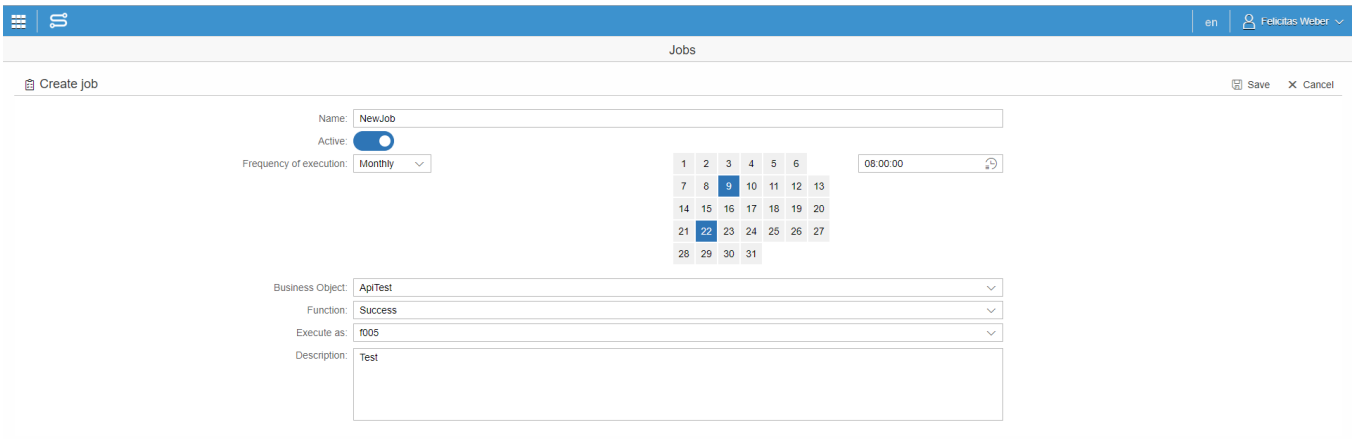
If you select this interval, it will be repeated every week. Set it in <hour>:<minute>:<second> and the appropriate day(s).



Monthly

Set this interval if you want that it’s repeated monthly. Set it in <hour>:<minute>:<second> and the day(s) within this month.

If the selected day doesn’t occur in this month, e.g. the 30th February, the job will **not** be executed in this month.

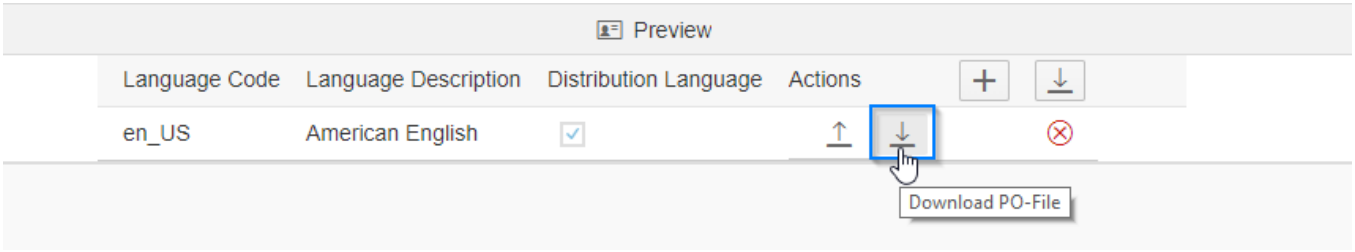

en
Felicitas Weber
Jobs
Create job
Save
Cancel
Name:
NewJob
Active:
Frequency of execution:
Monthly
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
08:00:00
Business Object:
ApiTest
Function:
Success
Execute as:
f005
Description:
Test

Language

Applications can be localized easily by adding language files (.po-format) to the application. A fallback language has to be defined. At default the language will be set according to user's environment, browser or device language.

For the translation of the individual applications, .po files are used in the Simplifier. Those can be viewed, downloaded and uploaded via the Language tab in the UI Designer.

To create a new translation file, you can download a template from this overview by clicking on the corresponding button.



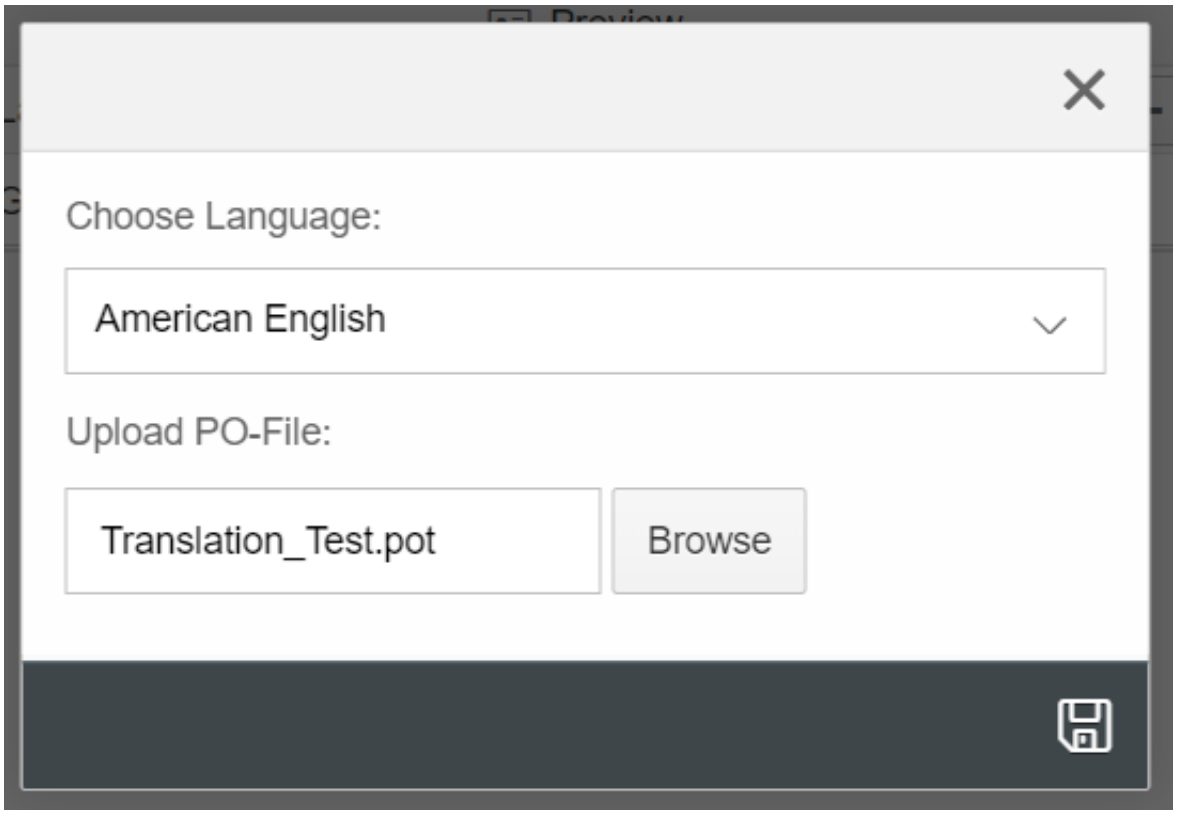
The downloaded file can be edited in the text editor of your choice (e.g. Notepad ++).

Important: translatable values must be flagged within a widget in order for them to appear in the default language file!

```
"POT-Creation-Date: \n"
"PO-Revision-Date: \n"
"Last-Translator: Your Name <email@example.com>\n"
"Language-Team: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

# Default translation: Sprache ändern
msgid "Main$Main_Button$text"
msgstr "Change language"

# Default translation:
msgid "Main$Main_Button$tooltip"
msgstr ""

# Default translation: Dies ist ein Test
msgid "Main$Main_Text$text"
msgstr "This is a test"

# Default translation:
msgid "Main$Main_Text$tooltip"
msgstr ""
```

In this template, all values marked as translatable in the widgets used are listed and can be translated into the target language.

This file can then be added via the Plus icon. You have to select the target language and the file you want to edit.



If the desired language is not available, please contact your administrator, as these are maintained in the server settings.

To change the language at runtime, the following script must be inserted in the Process Designer:
sap.ui.getCore().getConfiguration().setLanguage(“en”), where “en” can be replaced by the desired abbreviation.

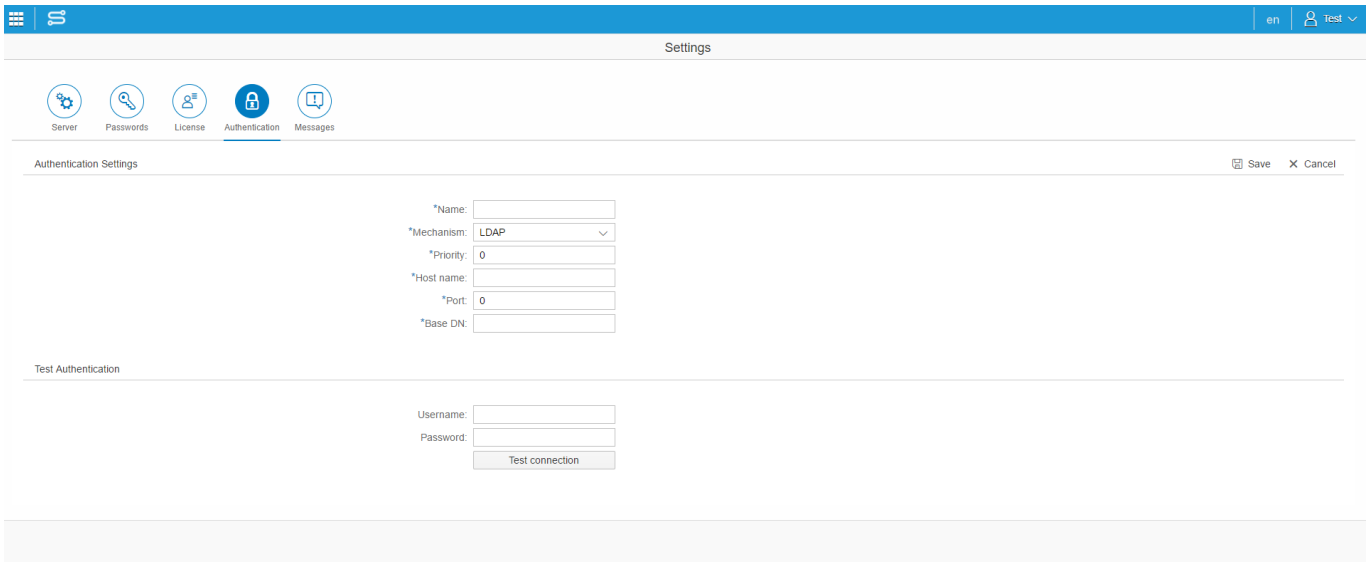
To start an application with a specific language, the URL of the following query parameters can be added:

?sap-language=en

_____

LDAP

Using LDAP:

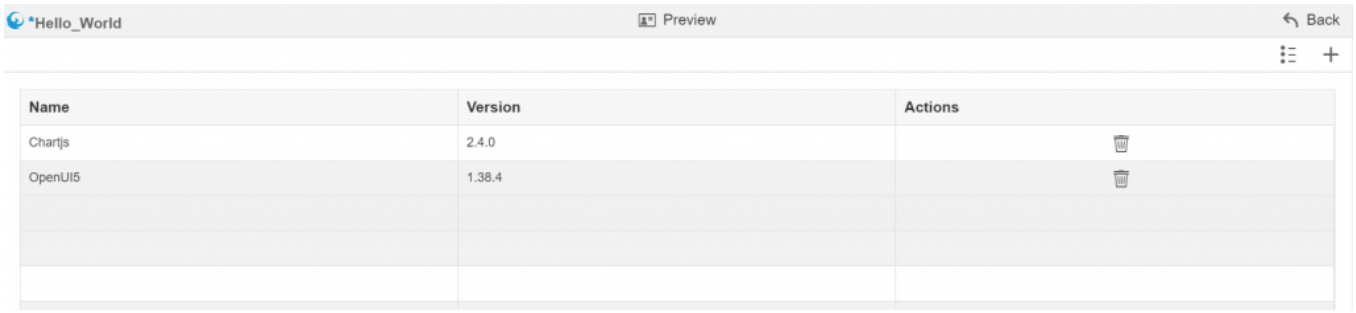


Name	Defines the name of the authentication
Mechanism	LDAP
Priority	This defines the order in which the authentication mechanisms are processed
Host name	This defines the IP/ Host of the authentication service
Port	This defines the authentication port
Base DN	This defines the Domain name of the entry point

You can test the connection by inserting a Username and Password.

Libraries

You can assign different libraries to your application and get an overview on their dependencies.
Go to “Integration of external Libraries” for more details.



List of Plugins

The Simplifier comes with the following plugins:

Plugin Name	Description
keyValueStorePlugin	Offers a Key/Value Store Database based on Java MapDB
pdfPlugin	Offers a PDF Template Designer and Generator based on wkthmltopdf
contentRepoPlugin	Offers a Content Repository for saving and sharing documents in a folder hierarchy with own permission objects
jsonStore	Offers a Database Plugin to save and read back customer information (and application data) from apps in json format.
sessionPlugin	Offers a User-Session Plugin to save and manage User Session
stepByStepGuidancePlugin	Offers a generator (like a wizard) to build simple apps consisting of one information (image, text, video) per screen and one of the predefined functions.
captcha	Server-Side Generation of Captcha Images for the UI5 Captcha Widget

You can find more plugins on the Marketplace.

List your PDF Templates

List Templates

To list your templates, you need the following parameter:

URL		/client/1.0/PLUGIN/pdfPlugin/adminTemplateList
Input-Parameter	None	
Output-Parameter	**Value**	JSON-Array with all template names

Example output:

```
{     "success": true,     "value": [         "templatename1",         "templatename2",         "templatename3"     ] }
```

Locally

You can install the Docker engine on your local machine and start the Simplifier Docker for development and testing.

The Docker Engine runs on

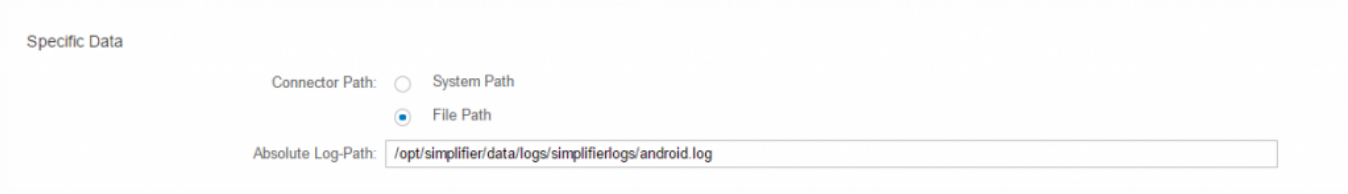
- Windows 10 Pro
- Mac OS
- Linux

To see how to start the Simplifier, click here.

Logging Write Connector Details

Choose the Connector Path:

System Path



File path

How to use a Logging Connector in a REST-Client

You can choose between READ and WRITE.

Write Logging Connector Handler

```
{
   "connectorName": "WriteLoggingConnectorHandler",
   "json": {
            "msg": "Test Log Konnektor umgeschrieben",
            "logLvl": "debug",
            "logGroup": "pickIt",
            "deviceMac": "",
            "timestamp": 1425293416
   }
 }
```

Read Logging Connector Handler

```
{
```

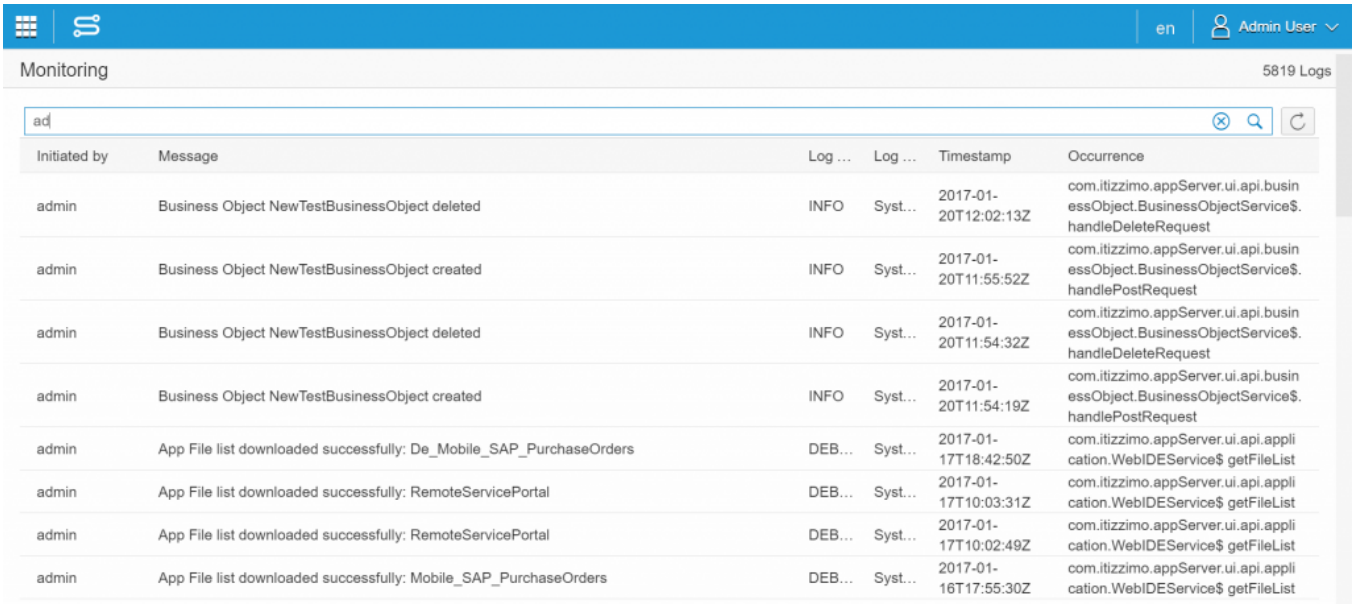
```
  "connectorName": "ReadLoggingConnectorHandler",
  "json": {
            "logGroup": "pickIt",
            "logLvl": "debug"
  }
}
```

Logic

Currently the Logic shape is divided into Condition, Iterator and Script.

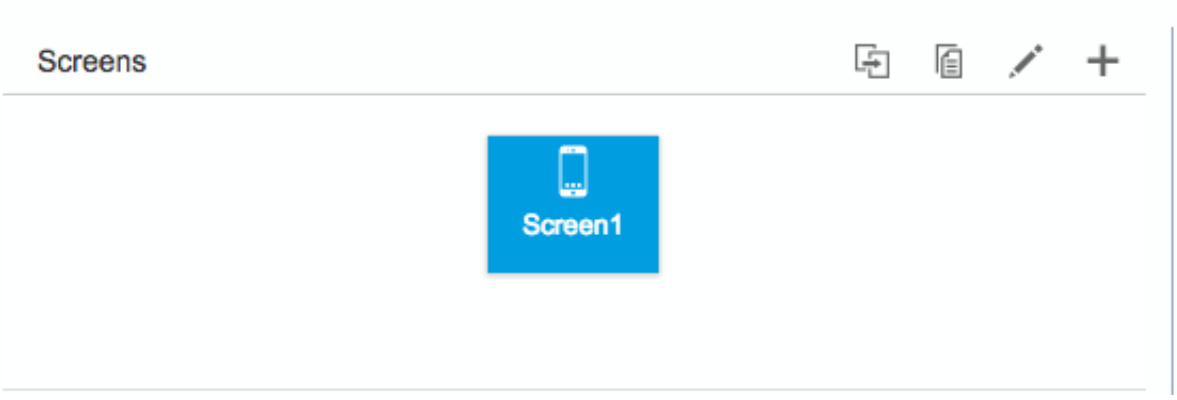
Logs & Monitoring

The error log is the central logging and monitoring area where all data-streams and errors are logged and saved.
You can display all data of a specific User by using the search bar.



Manage Screens

Each process step can be configured as a screen. A screen represents one single user interface page and can contain several widgets.



In the screen section you can do the following actions:

Copy Screen — Copy a screen and paste it into the actual or a different application.

Paste Screen — Paste the previously copied screen into the current or a different application.

Edit Screen — Activate the edit mode to change screen order or delete screens.

Add Screen — Add a new screen to the actual application.

Mapping Collections

Collections represent multiple results of Structs. For example a databank request may deliver a list of addresses from numerous people and you can map them to the equivalent Widget within a Data Object.

The Input Mapping

Unfortunately, the input mapping of collections is not yet possible. Nevertheless, you can use the “Script” tile and map your data via code.

There is a general pattern you can follow and adapt it to your specific data. To map data to your table, you must have a predefined Connector with at least one Connector Call.
Now click on the Script tile and edit the code.

Step 1

Build an object.

Step 2

Add payload to your object. If the input parameter of your Connector Call are filled in with an “Alias”, you can add them as keys to your object.
Otherwise, you need an escape function. Example:

```
payload[escape("soap/_-ITIZ_-OBJ_BUS2012_UPDATE/IT_PO_ITEMS(1)/QUANTITY")] = "X"
```

```
var dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT = {};
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['PLZ'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerPlz", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Ort'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerOrt", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Strasse'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerStr", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Hausnr'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerHsn", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Telefon'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerTelefon", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Mail'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerMail", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Type'] = this.getItemValue("BusinessPartner", "TextSAPType", "text");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Name 2'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerPersonVorn", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['First Name'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerPersonVorn", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Last Name'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerPersonNachn", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Name 1'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerPersonNachn", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Title'] = this.getItemValue("BusinessPartner", "MComboBoxBusinessPartnerPersonAnrede", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Grp 1'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerGruppeName", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Org 1'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerFirmaName", "value");
dataForSAP_BU_ISUBusinesspartner__ITIZ_BU_OBJ_ISUPARTNER_CREATE_FLAT['Org 2'] = this.getItemValue("BusinessPartner", "InputBusinessPartnerFirmaZus", "value");
```

Step 3

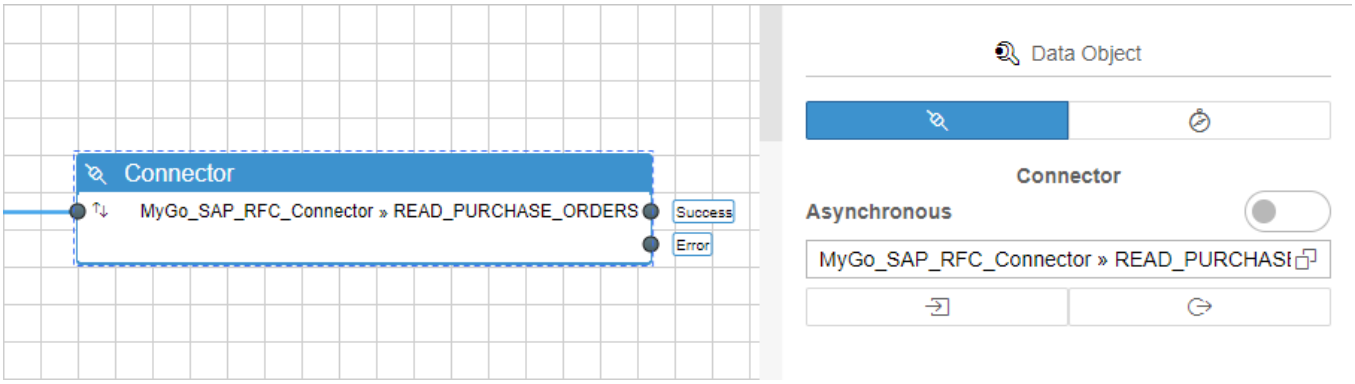
Add a global function call with the following Parameters:

1. Parameter: Connector name
2. Parameter: Connector Call name
3. Parameter: Keys you added above (payload)
4. Parameter: Success function
5. -7. Parameter: Error function, showBusy Indicator and error message

```
this.callConnectorCall("CONNECTOR", "CONNECTOR_CALL", dataForSAP, successCallback, true, true, null);
```

The Output Mapping

The output mapping of a Data Object to table, a list or a select Widget can be easily done in the Process Designer.



Click on “Output Mapping” and add the output parameter (on the left) and your table widget (on the right) to the screen.
Double click on each to select the items.
Now map the equivalent items together via drag & drop.

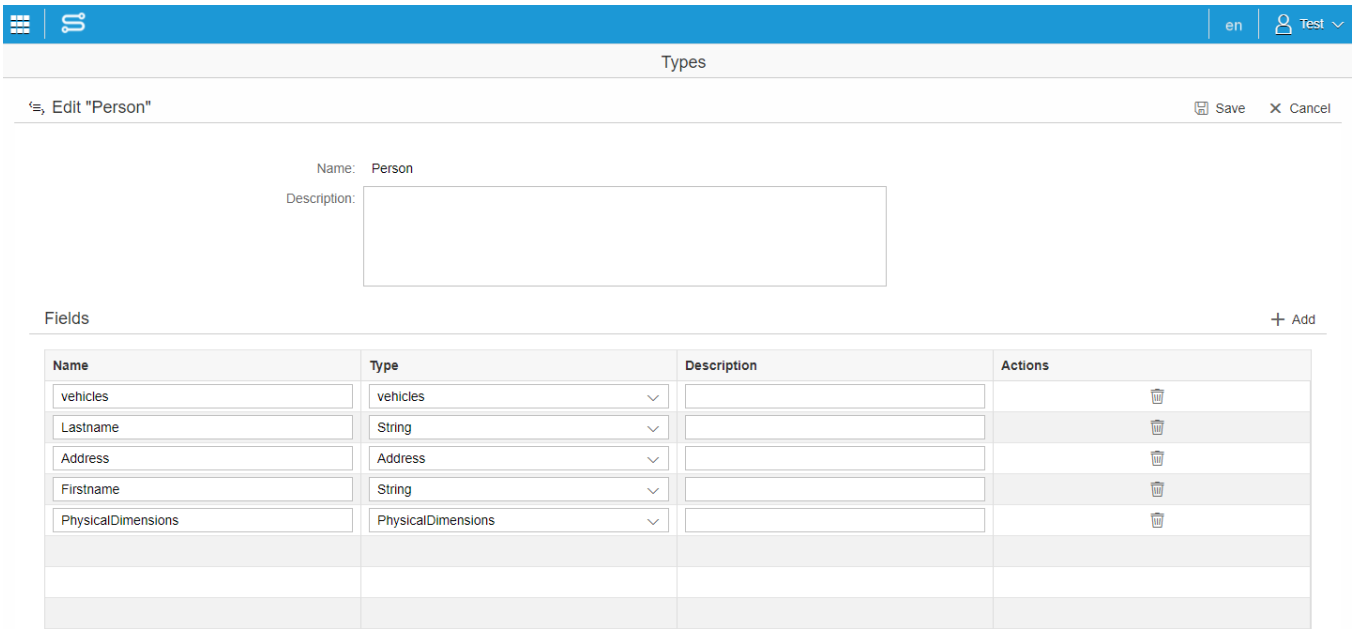
Mapping Structs

Structs describe a package of Domain Types. For example the Struct “address” contains different Domain Types like Name, Street, City, ZIP Code, etc.

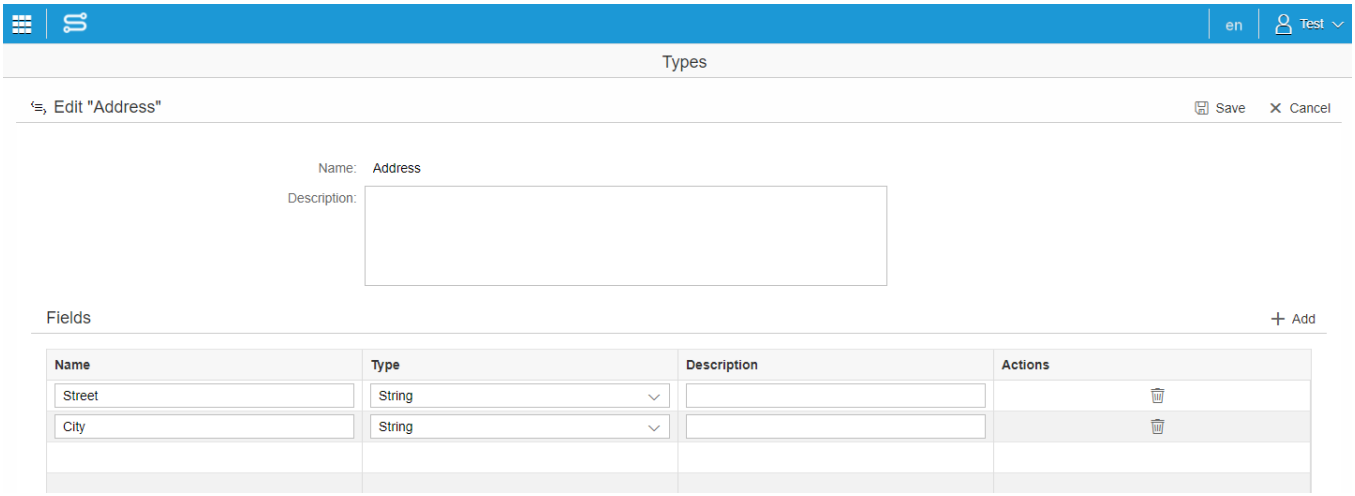
You can map Structs as Input and Output parameter within your Data Object or when using a UI Action.

Example:

In the example below, we want to use a Business Object that needs the information first name, last name, street and city as Input parameter. Prior to our work in the edit mode of the user story (Process Designer), we have created the Struct Data Type “Person” that contains several Domain Types like “Firstname” and another Struct which is called “Address”.



Struct Data Type “Person”

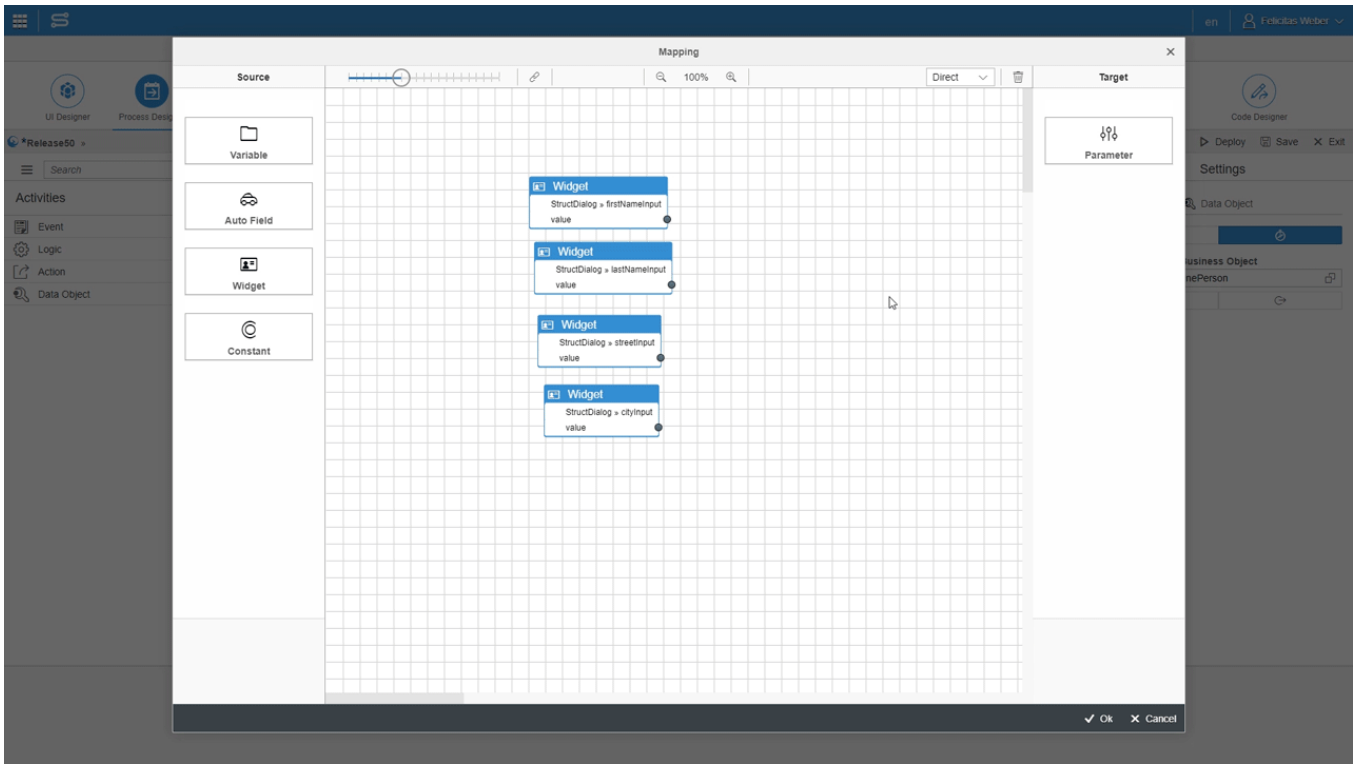


Struct Data Type “Address”

Within the Input Mapping of our Business Object, we can assign the Input fields with the equivalent data to the Struct “Person” that the Business Object needs.

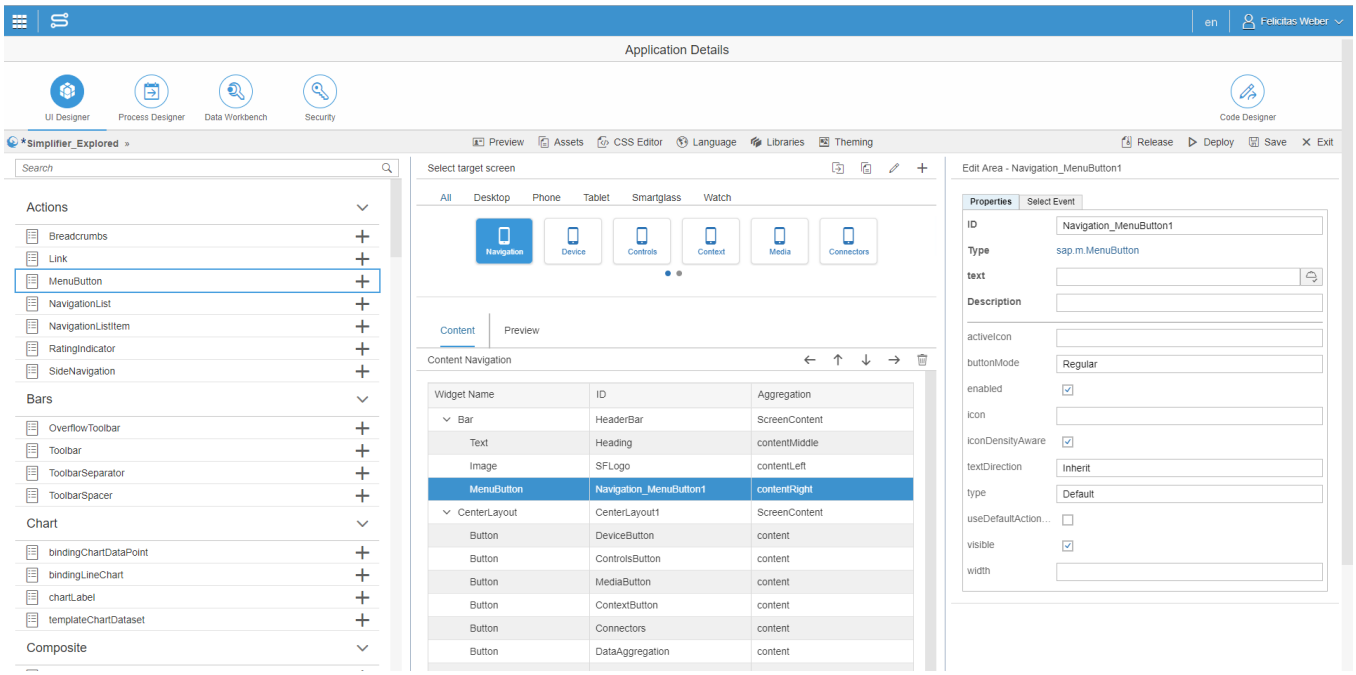
By double-clicking on the Struct you can choose exactly which parameter you want to select.

Navigate higher or deeper in the Struct by clicking on the arrows (e.g. to select the Domain Type “Street” within the Struct “Address”).

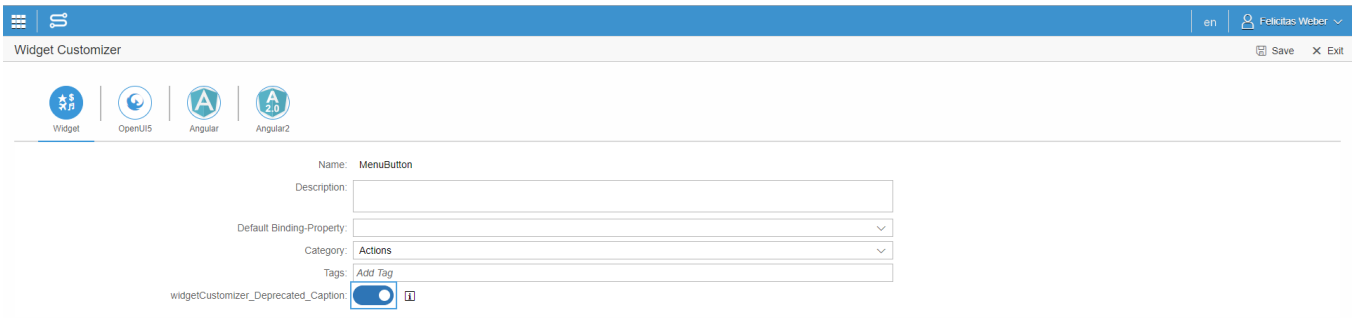

Mapping
Source
Target
100%
Direct
Variable
Auto Field
Widget
Constant
Parameter
Widget
StructDialog » firstNameInput
value
Widget
StructDialog » lastNameInput
value
Widget
StructDialog » streetInput
value
Widget
StructDialog » cityInput
value
Ok
Cancel

Mark Widgets as deprecated

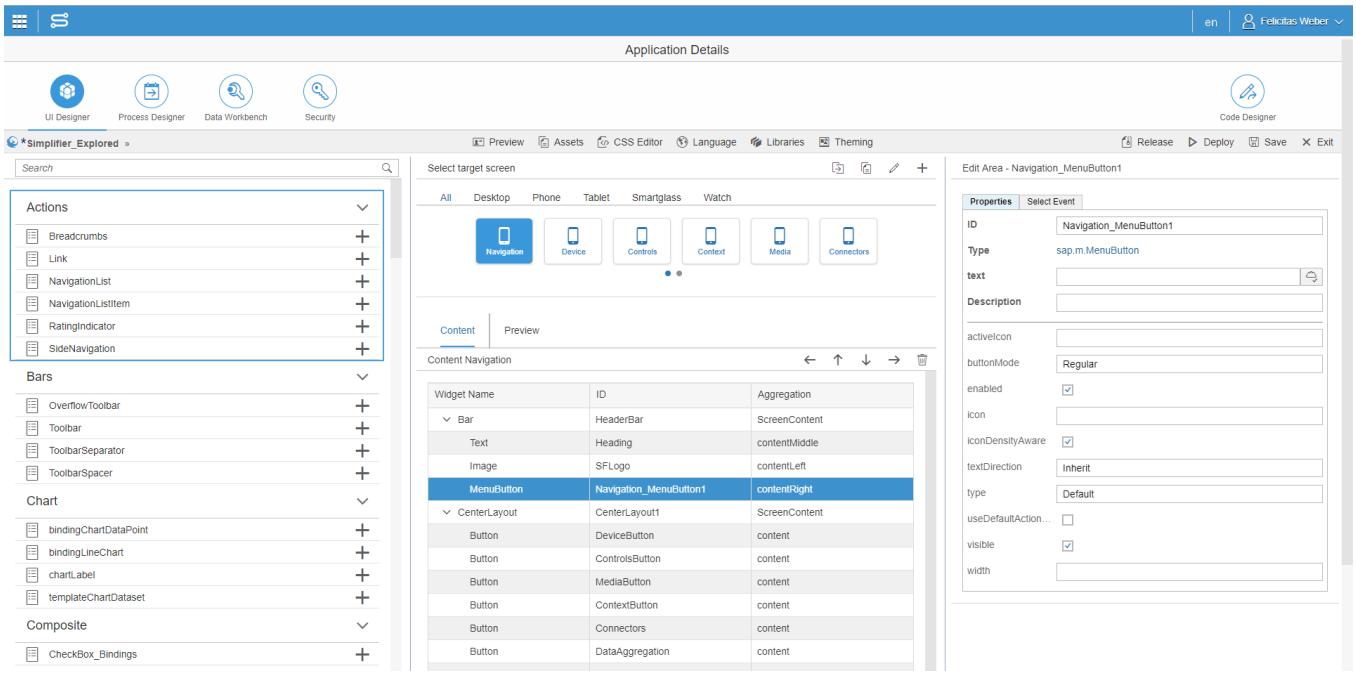
You can mark widgets as deprecated to prevent them from being displayed within the UI Designer.



For that you have to go back to the application overview and switch to the widgets tab. Search for the appropriate widget and edit it. All you have to do now is to activate the toggle ‘Deprecated’ of the widget and click on save.



If deprecated is active, the Widget will no longer be available on the left side of the UI Designer.



Deprecated widgets that were used before they were marked as deprecated, will still be shown within the application content. The application will also still be saved and deployed without issues or warnings and runs as before.

Marketplace

The marketplace module is central to the Simplifier Experience and Workflow. It does not only provide a variety of applications ready to import to your own account, but also any other type of module content such as Connectors, Virtual Connector packages, Business Objects, custom widgets and more.

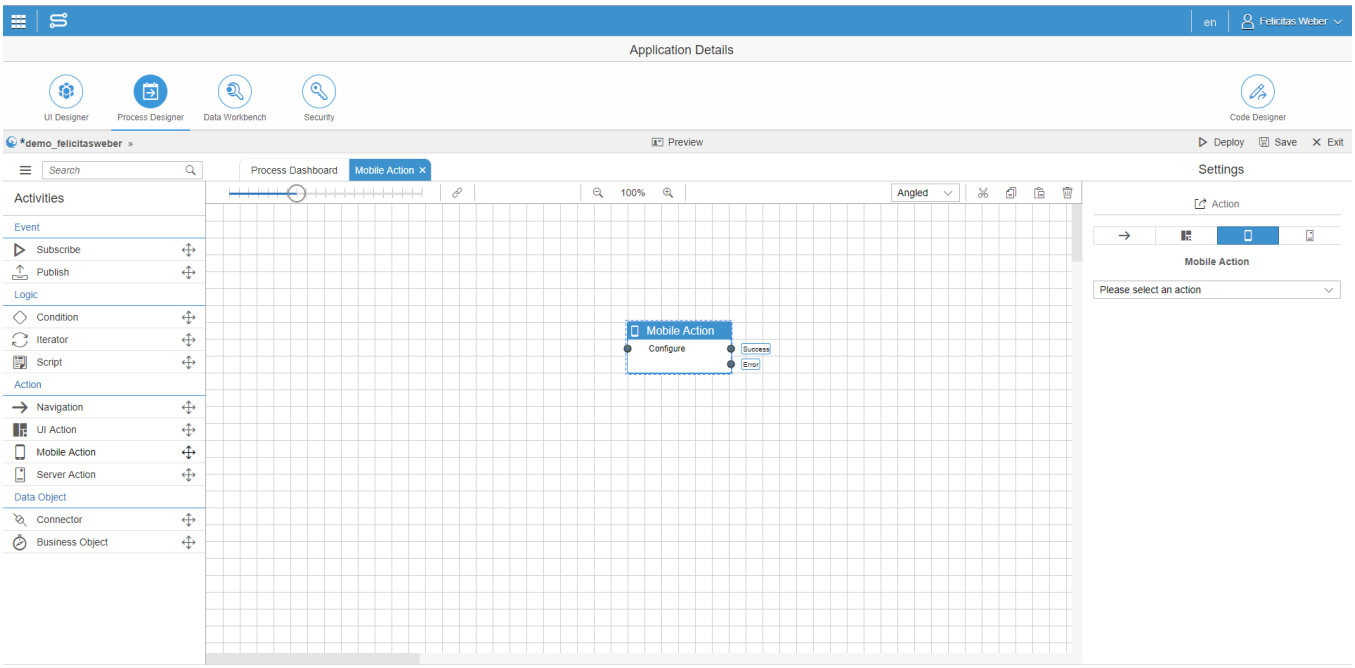
To add any type of content to your own account, browse the marketplace and click “request” to send us an e-mail with all the information we need so we can provide the content to you.

In the future, this feature will be automated, so you can add content with one click.

Mobile Action

The Mobile Action Element enables you to use mobile features within the application.

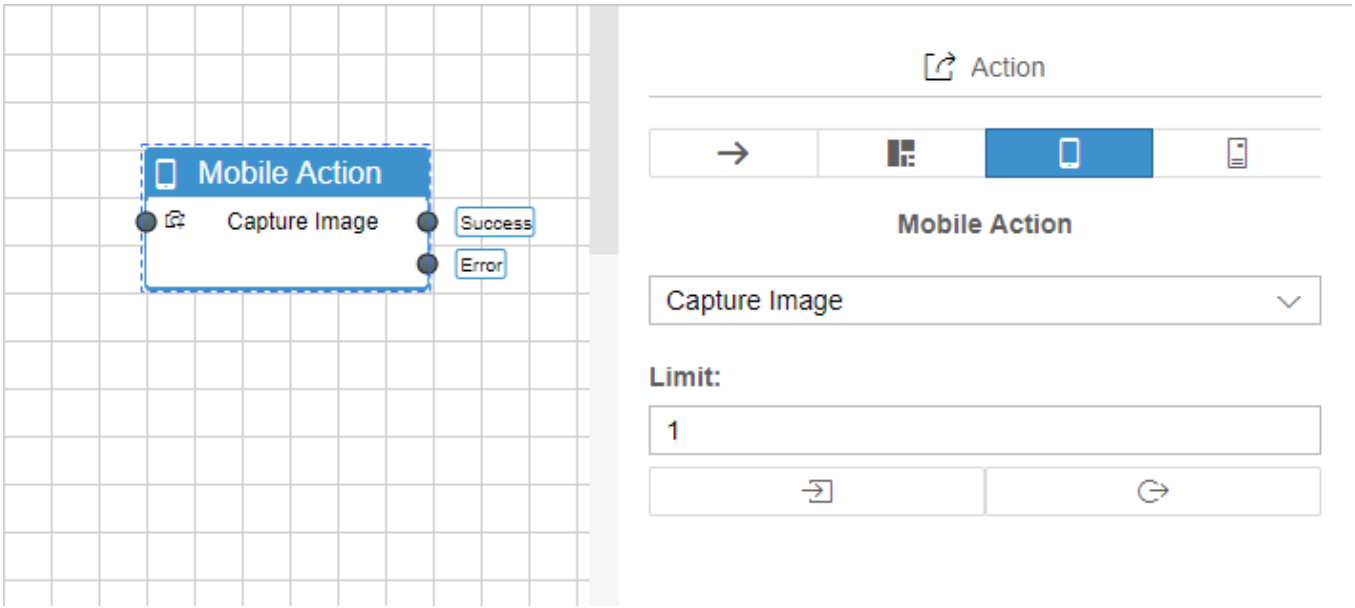
Drag and drop the activity “Mobile Action” underneath “Action” into the drawing area.



Currently, there are 14 different actions implemented in the Simplifier. Use the Input/Output Mapping to define the source for the input (request) and the destination for its output (response).

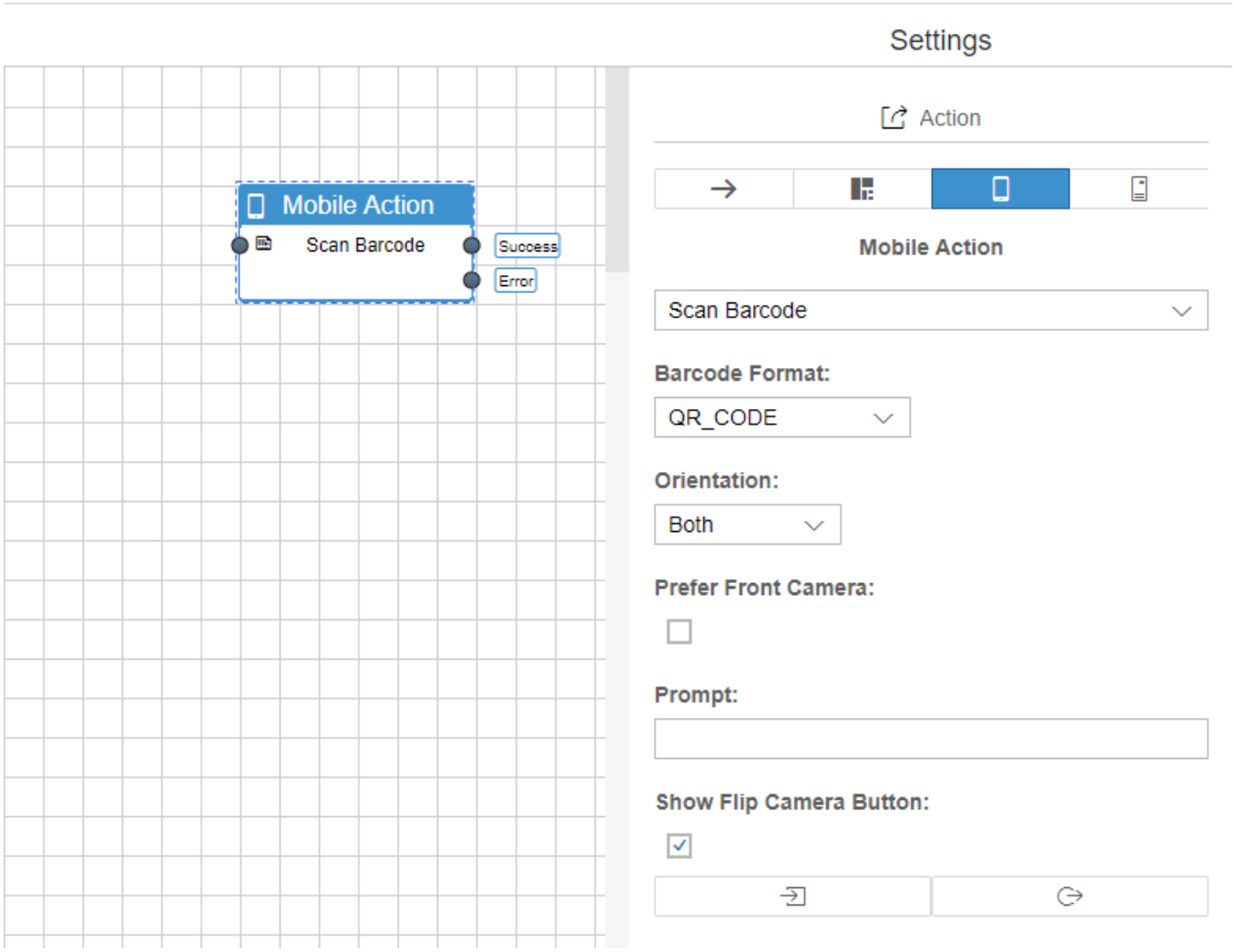
Capture Image

Determine how many pictures should be taken.



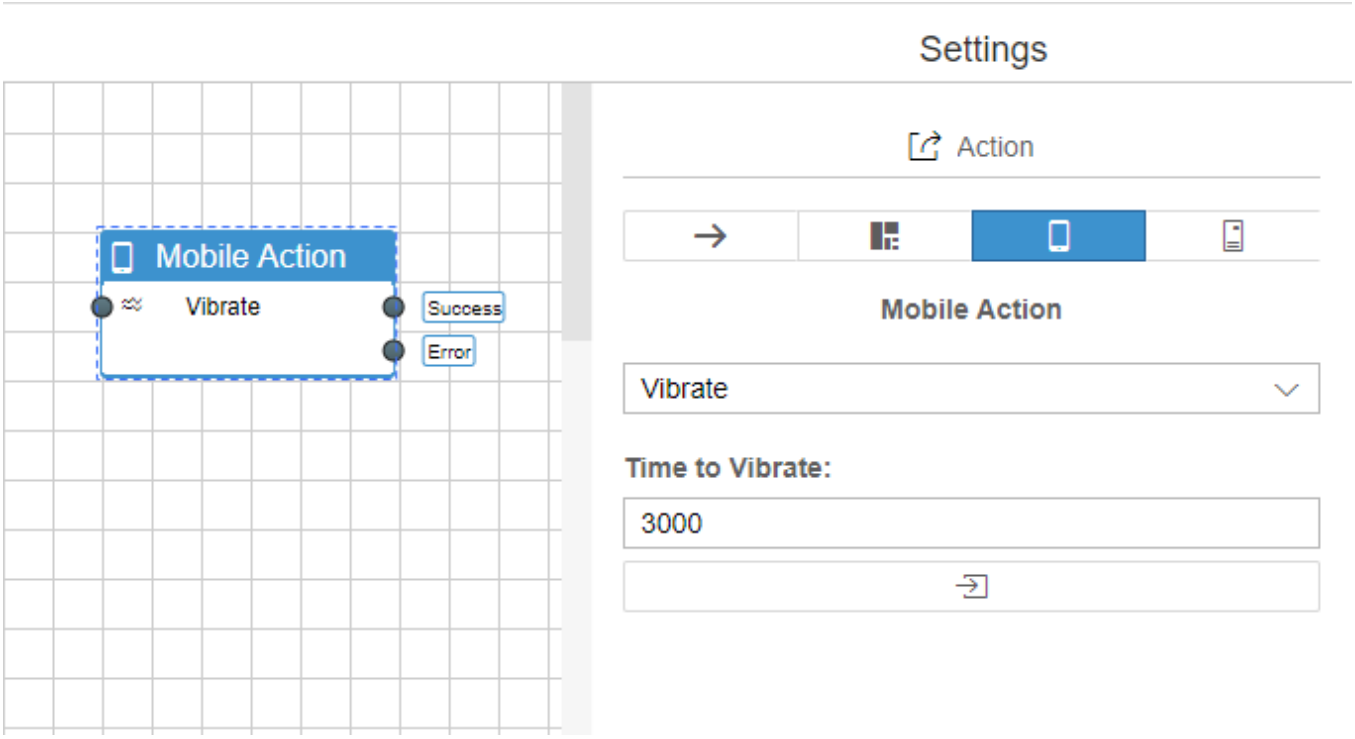
Scan Barcode

Set the format of the code you want to scan, the orientation of the camera and write a text in the prompt property.



Vibrate

Define how many miliseconds the mobile device should vibrate.

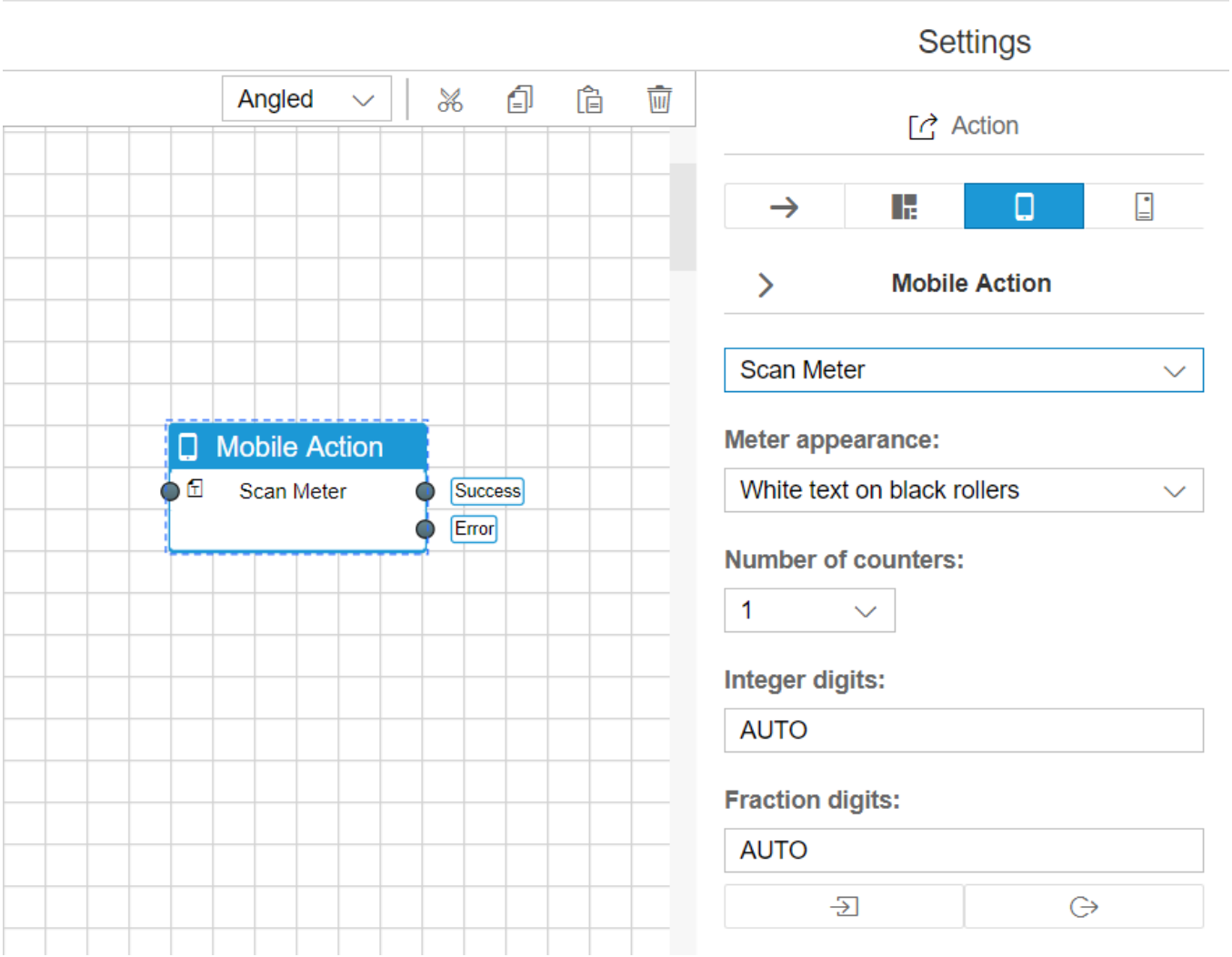


Scan Meter

Parameter	DataType	Valid Values
Meter appearance	String	See the table below
Number of counters	Integer	1, 2 (from dropdown)
Integer digits	Integer/String	AUTO, 1-n
Fraction digits	Integer/String	AUTO, 1-n

You can choose from the following values for meter appearance:

Key	Value
mechanical_black	White text on black rollers
mechanical_white	Black text on white rollers
lcd	LCD Display
lcd_edl21	EDL21 and similar meters with LCD 7 (three-digit OBIS code to the left)



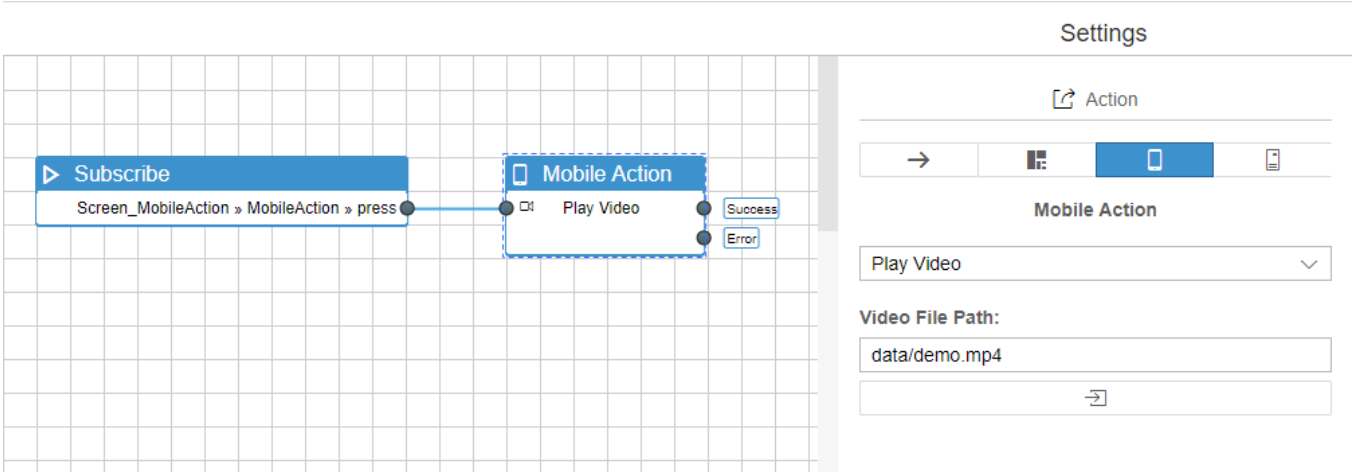
For the output mapping you can use the following parameter:

Paramter	DataType	Description
image	string	Image of the scanning process as data url
line1	string	Result of scanning the first counter (always set)
line2	string	Result of scanning the second counter (only set if 2 counters were scanned)
line1ObisCode	string	Result of scanning the first counter's OBIS code (only set if meter appearance was set to lcd_edl21)
line2ObisCode	string	Result of scanning the second counter's OBIS code (only set meter appearance

was set to lcd_edl21 and 2 counters were scanned)

Play Video

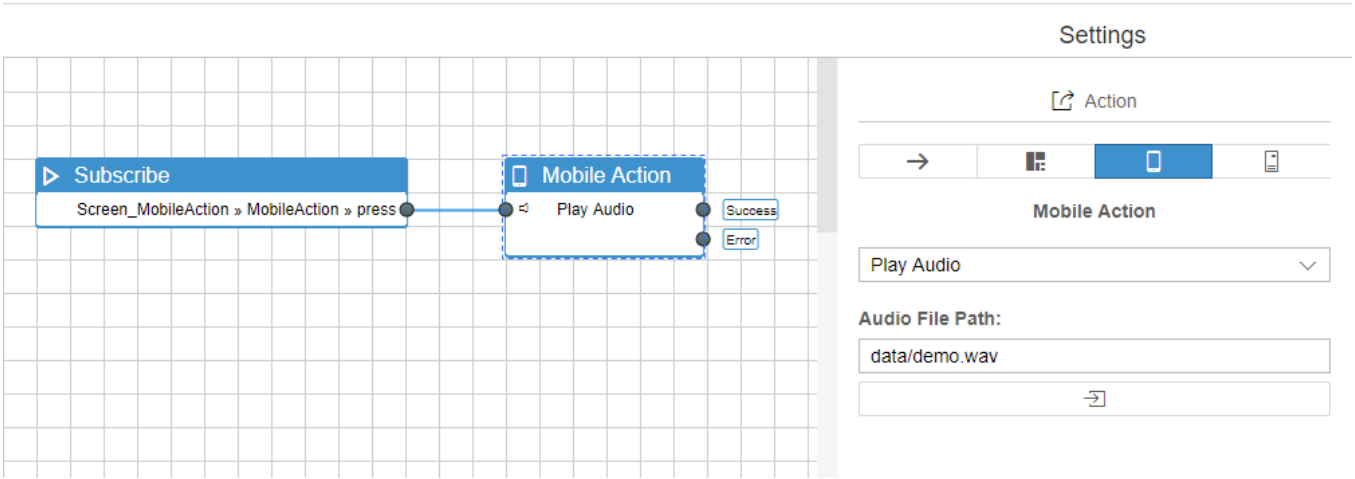
By providing the mobile action a path to a video, it will play the video in a full screen mode.



***NOTE:** Please use this mobile action with caution. Some devices don't support this feature.

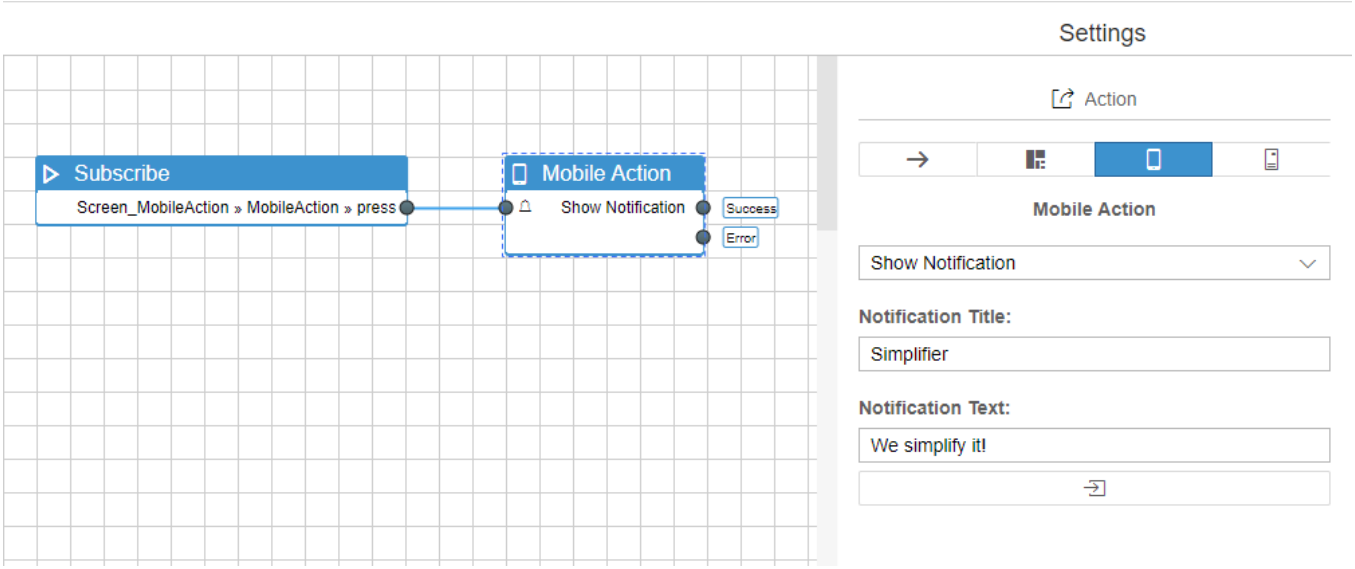
Play Audio

The mobile action Play Audio just needs an audio file path.



Show Notification

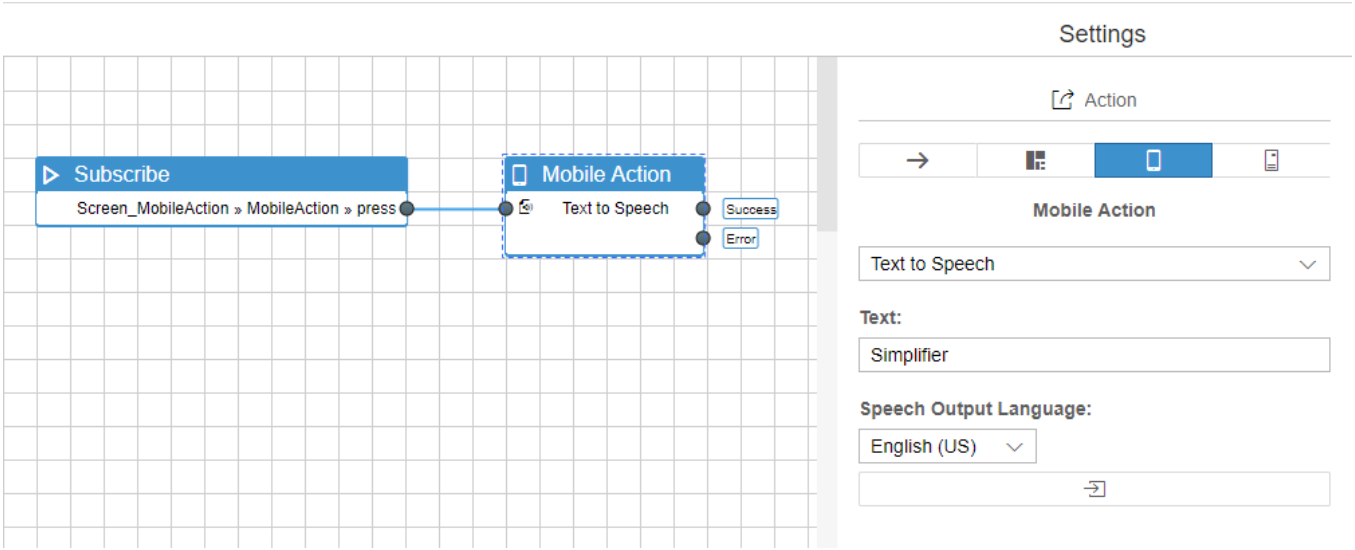
This mobile action will trigger a native notification.



So the Notification will be displayed on your mobile device.

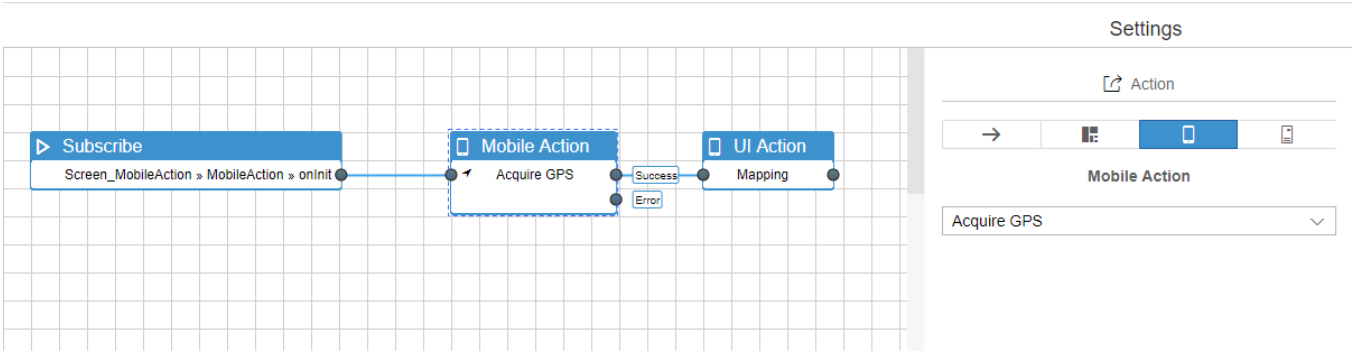
Text to Speech

With this mobile action you can specify a text to be read out, as well as the output language. You can choose between English (US) and German (DE).

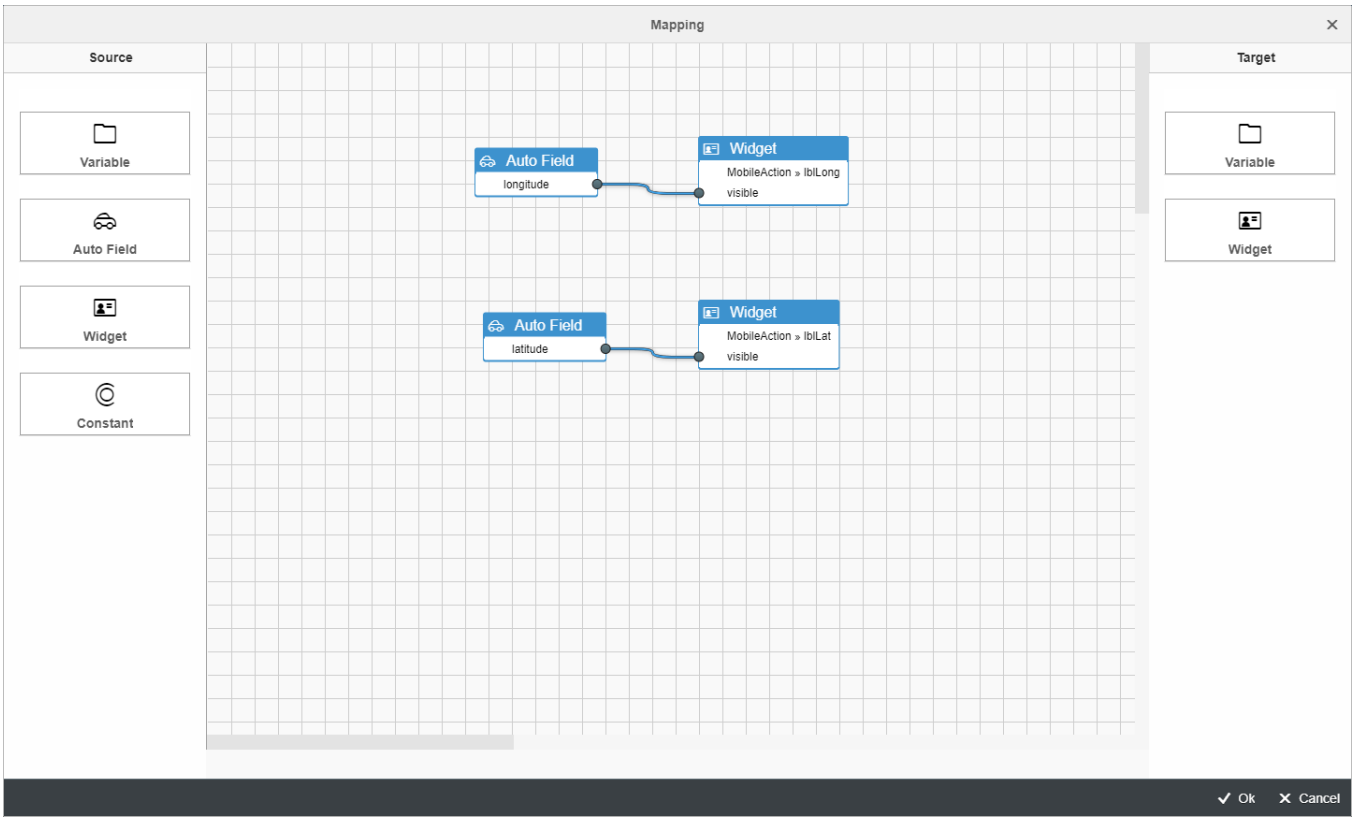


Acquire GPS

The mobile action GPS need to be initialized at first. It’s recommend to do this in a screen onInit function.

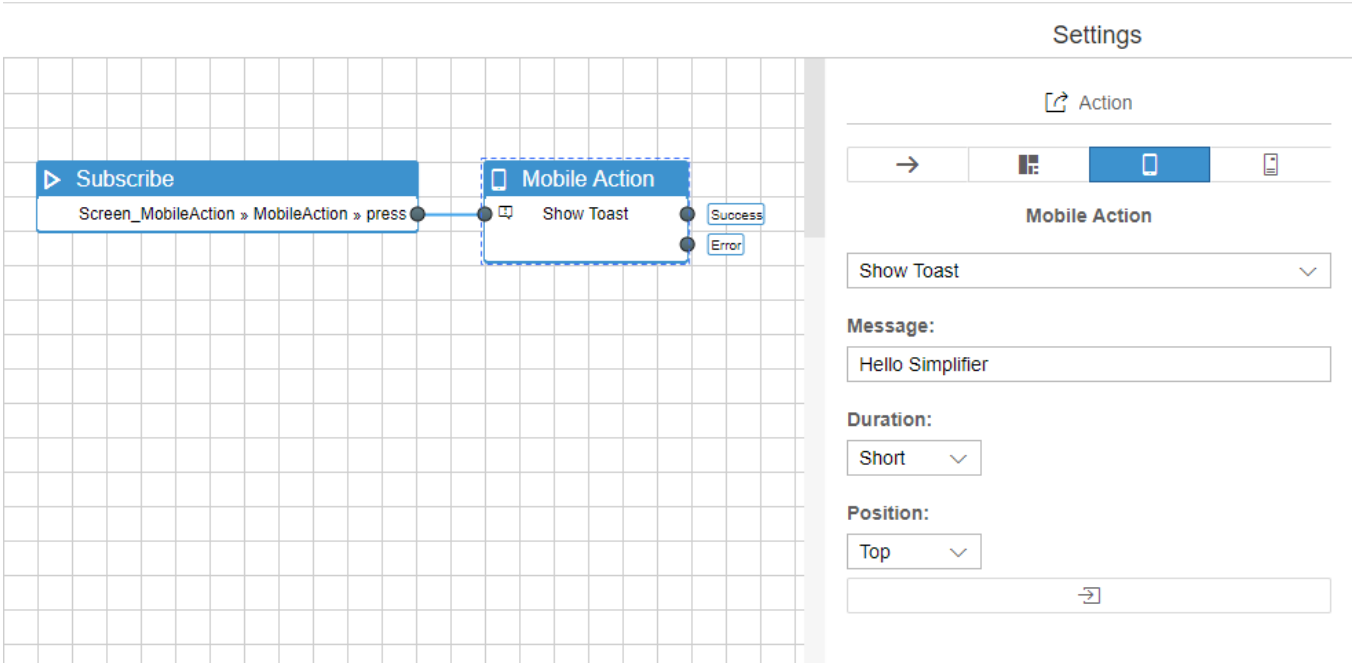


When it’s initialized, you can access the longitude and latitude by the Auto Field type Geolocation. For that you need to open the mapping dialog of the UI Action. Then you can drag the auto field in the middle and by double clicking on it you can select the longitude/latitude/. Then connect it with the appropriate widget.



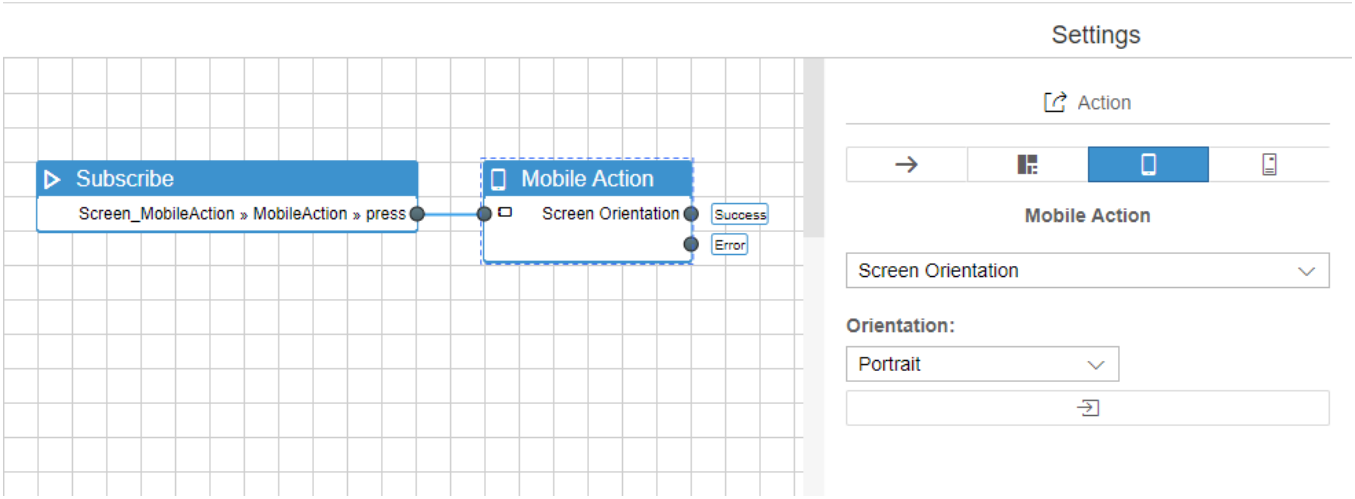
Show Toast

This mobile action will show a native toast, with a constant or by using the input mapping. You can specify the duration how long the toast should be displayed (Short or Long) and the position (Top, Center or Bottom).

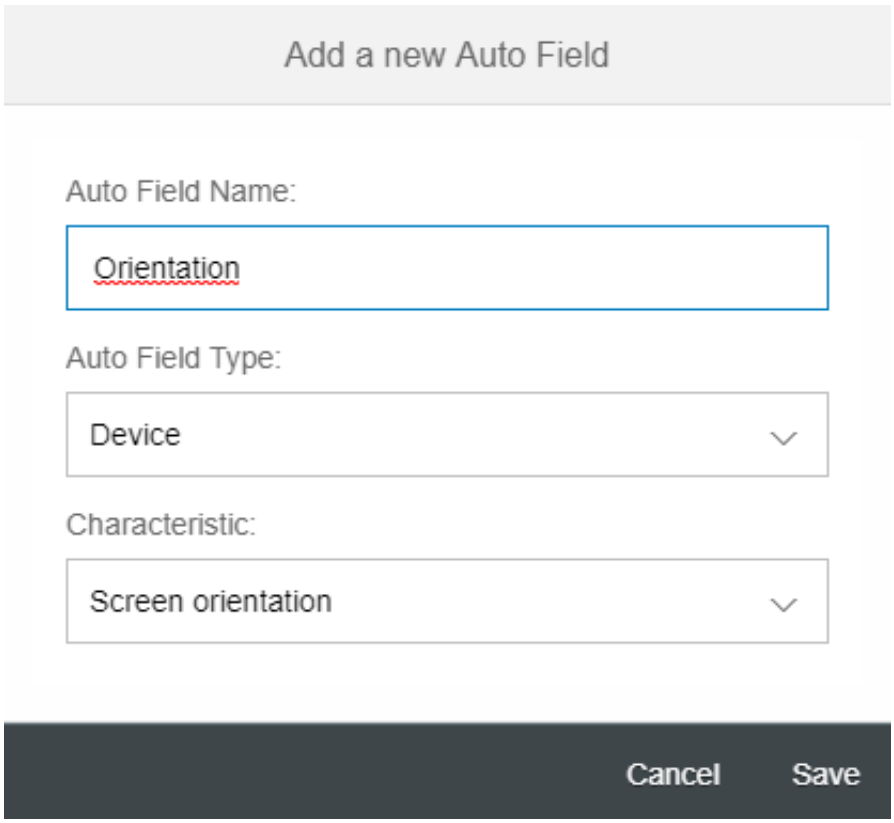


Screen Orientation

You can set and lock the screen orientation of the mobile device. Possible values are Landscape, Portrait, Landscape (Primary), Landscape (Secondary), Portrait (Primary), Portrait (Secondary) and Unlock.

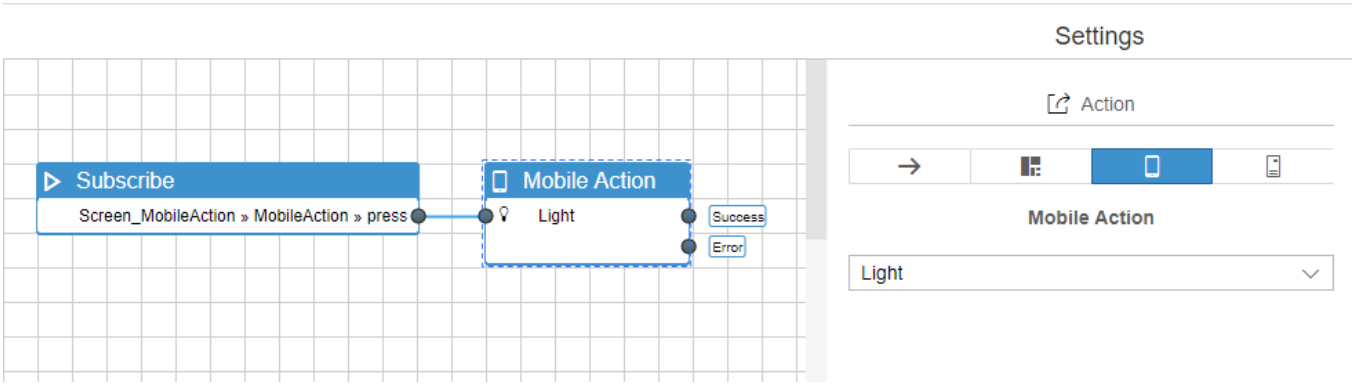


The current screen orientation can be accessed by a new characteristic of the Device Auto Field.



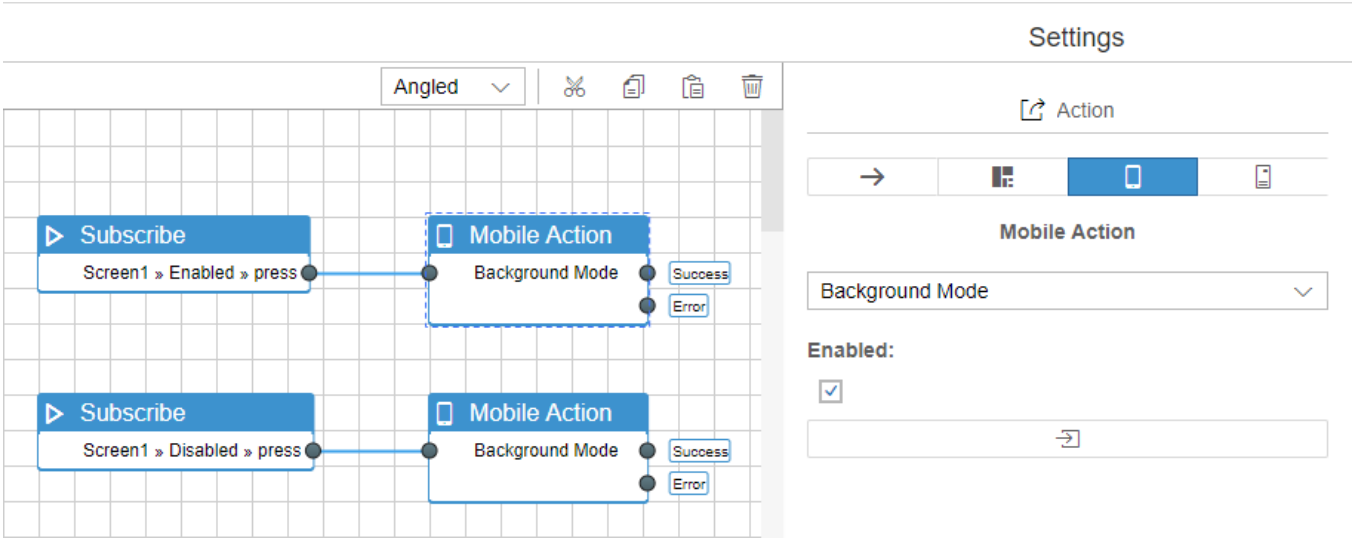
Light

With this mobile action, your mobile device can toggle light.



Background Mode

If the background mode is enabled, JavaScript will be executed even if the app is running in the background or the display is locked.



Read the subpages to use other mobile actions.

Mobile Action for WebRTC Calls

General infos

Plugin to provide WebRTC P2P functionality via Intel’s WebRTC SDK.

Intel WebRTC SDK Version: 3.4

Platforms:

- Android (4.0.4+ – x86, armV7)
- iOS (8+)

Cordova: 6+

When the local user connects to a remote user, this plugin overlays the cordova webview to show local and/or remote media streams.

General event object description

```
{
  "action": "<action>", //see "Possible Actions"
  "message": "<String>", //some message depending on action and success
}
```

Possible Actions

Active Actions

Actions triggered by local user

- INIT – takes stun/turn-, userconfig and initializes peer connection
- LOGIN – login to peerserver
- LOGOUT – logout from peerserver
- INVITE – inite some user
- HANGUP – peerconnection to remote user disconnected
- PUBLISH – a/v media published to remote user
- UNPUBLISH – a/v media unpublished
- SWITCH_CAMERA – switch camera front <-> back
- SEND_DATA – send data to remote user
- SCALE_VIEW – view got scaled
- SWITCH_SPEAKER – switching from internal to external speaker
- FULLSCREEN – view has been set to fullscreen
- START_SHARESCREEN – Started sharing/streaming screen content
- STOP_SHARESCREEN – stopped screen sharing

Passive Actions

Actions triggered by server/remote user/network

- ONINVITED – remote user invited local user
- ONACCEPTED – remote user accepted invitation from local user

- ONDENIED – remote user denied invitation from local user
- ONDATARECEIVED – remote user sent date to local user
- ONCHATSTOPPED – connection to remote user stopped
- ONCHATSTARTED – connection to remote user initialized
- ONDISCONNECTED – local user got disconnected from remote server
- ONSTREAMADDED – remote user published a/m media to local user

Cordova API

init

Initializes view, connects to server.

runTimeSuccessCallback / runTimeErrorCallback: is being used to provide a callback interface during plugin runtime

If the plugin has already been initialized, the plugin just attaches the runtime callbacks to regain context.

```
IntelWebRTC.init(config, runTimeSuccessCallback, runTimeErrorCallback);
```

logout

if connected to peer it closes the connector, disconnects from Server, removes view components

```
IntelWebRTC.logout();
```

invite

connect to desired peer

```
IntelWebRTC.invite("peerid", scopedSuccessCB, scopedErrorCB);
```

publish

publishes audio/video to connected peer (is automatically called when peerconnection is established)

```
IntelWebRTC.publish(scopedSuccessCB, scopedErrorCB);
```

unpublish

stops sending audio/video to connected peer

```
IntelWebRTC.unpublish(scopedSuccessCB, scopedErrorCB);
```

hangup

disconnects from currently connected peer

```
IntelWebRTC.hangup(scopedSuccessCB, scopedErrorCB);
```

switchcamera

cycles through cameras

```
IntelWebRTC.switchCamera(scopedSuccessCB, scopedErrorCB);
```

sendData

sends a string to connected peer

```
IntelWebRTC.sendData("",scopedSuccessCB, scopedErrorCB);
```

scaleView

scales and aligns view to given parameters

```
IntelWebRTC.scaleView(0.5, "top_left", scopedSuccessCB, scopedErrorCB);
```

fullscreen

sets callUI to fullscreen

```
IntelWebRTC.fullscreen(scopedSuccessCB, scopedErrorCB);
```

startShareScreen

starts screen sharing

```
IntelWebRTC.startShareScreen(scopedSuccessCB, scopedErrorCB);
```

stopShareScreen

stops screen sharing

```
IntelWebRTC.stopShareScreen(scopedSuccessCB, scopedErrorCB);
```

Examples

```
var config = {
               "connectionConfig": {
                 "username": "paul",
                 "server": "https://simplifier-dev.itizzimo.com:8090"
               },
               "iceConfig": [
                 {
                   "url": "stun:turn.itizzimo.com:3478"
                 },
                 {
                   "url": "stun:turn.itizzimo.com:3479"
                 },
                 {
                   "url": "stun:turn.itizzimo.com:5349"
                 },
                 {
                   "url": "stun:turn.itizzimo.com:5350"
                 },
                 {
                   "url": "turn:turn.itizzimo.com:3478?transport=tcp",
                   "username": "admin",
                   "password": "admin"
                 },
                 {
                   "url": "turn:turn.itizzimo.com:3478?transport=udp",
                   "username": "admin",
                   "password": "admin"
                 },
                 {
                   "url": "turn:turn.itizzimo.com:3479?transport=tcp",
                   "username": "admin",
                   "password": "admin"
                 },
                 {
                   "url": "turn:turn.itizzimo.com:3479?transport=udp",
                   "username": "admin",
                   "password": "admin"
                 },
                 {
                   "url": "turn:turn.itizzimo.com:5349?transport=tcp",
                   "username": "admin",
                   "password": "admin"
                 },
                 {
```

```
                    "url": "turn:turn.itizzimo.com:5349?transport=udp",
                    "username": "admin",
                    "password": "admin"
                  },
                  {
                    "url": "turn:turn.itizzimo.com:5350?transport=tcp",
                    "username": "admin",
                    "password": "admin"
                  },
                  {
                    "url": "turn:turn.itizzimo.com:5350?transport=udp",
                    "username": "admin",
                    "password": "admin"
                  }
                ],
                "viewConfig": {
                  "scalingFactor": 0.3,
                  "gravity": "bottom_right"
                },
                "mediaConfig": {
                  "maxWidth": 640,
                  "maxHeight": 480,
                  "cameraDialog": true,
                  "preferFrontCamera": false,
                  "maxFps": 30,
                  "maxVideoBandwidth": 5000,
                  "maxAudioBandwidth": 200,
                  "videoCodec": "H264"
                },
                "debug": {
                  "local": {
                    "fps": false,
                    "bitrate": false,
                    "audiolevels": false
                  },
                  "remote": {
                    "fps": false,
                    "bitrate": false,
                    "audiolevels": false
                  }
                }
              };


var runtimeSuccessCallback = function(event){
    console.log("Success: " + event);
}

var runtimeErrorCallback = function(event){
    console.warn("Error: " + event);
}

//initialize plugin
IntelWebRTC.init(config, runtimeSuccessCallback, runtimeErrorCallback);
```

```
Success: successCallback ->  {"action":"INIT","message":"initialized"}

Success : successCallback -> {"action":"LOGIN","message":"stresstest2"}

//socket could not connect
Error : errorCallback -> {"action":"LOGIN","message":
"com.intel.webrtc.base.WoogeenException: io.socket.engineio.client.EngineIOException:
 websocket error","success":false}

//socket rejected (e. g. user not valid) - Logout needs to be called to reinitialize
connection with new credentials
Error: errorCallback -> {"action":"LOGIN","message":
"com.intel.webrtc.p2p.WoogeenP2PException: Socket.IO reports connect to signaling ser
ver failed.","success":false}

//initialized twice
Error: errorCallback -> {"action":"INIT","message":
"already initialized","success":false}
```

Invite

Invite some user which is also connected to peerserver

```
IntelWebRTC.invite('<username>');

Success: {"action":"INVITE","message":"stresstest4"}
Success: {"action":"ONACCEPTED","message":"stresstest4"}
Success: {"action":"ONCHATSTARTED","message":"stresstest4"}

Error: {"action":"INVITE","message":
"com.intel.web
rtc.p2p.WoogeenP2PException: Target user is unreachable.","success":false}
```

Logout

```
IntelWebRTC.logout(function(success){console.log(success)}, function(error){
console.log(error)});
```

Event Samples

Login

Initiator: IntelWebRTC::init

```
Success : successCallback -> {"message":"connected","action":"LOGIN"}

Error : errorCallback: -> {"message":"connected","success":false,"action":"LOGIN"}
```

Disconnected

Initiator: Device is offline / Connection to server lost Triggers on each connection retry (every ~5 Seconds)

```
Error: errorCallback: ->  {"action":"ONDISCONNECTED","message":""}
```

When connection is established again:

```
Success: successCallback -> {"action":"LOGIN","message":"stresstest2"}
```

Sample Eventlog

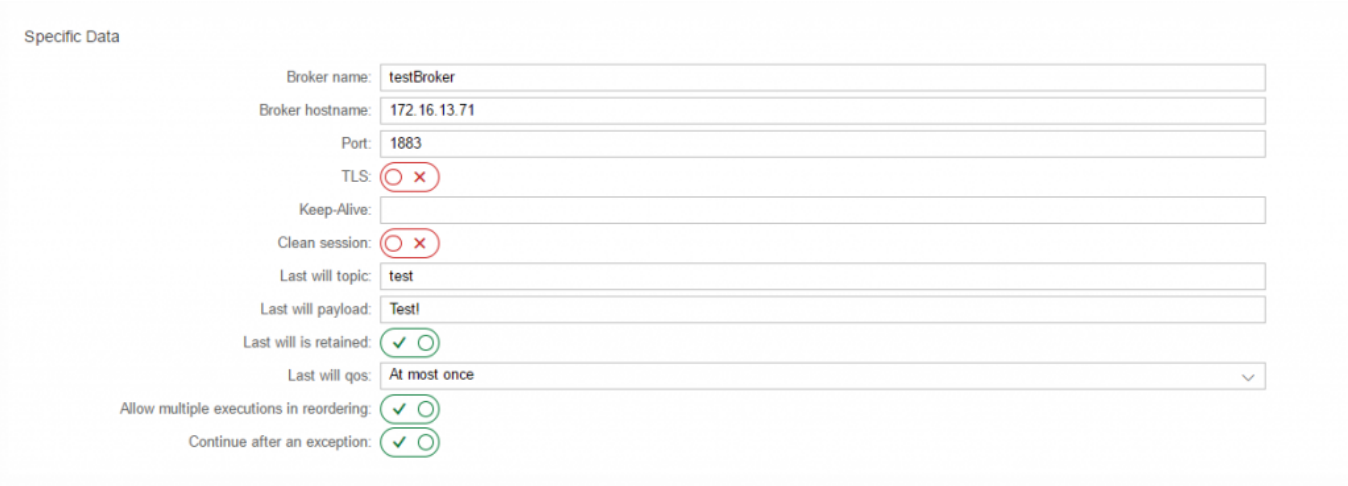
```
//login initialized
IntelWebRTC.init(config, runtimeSuccessCallback, runtimeErrorCallback);
//initialization success
Success: {"action":"INIT","message":"initialized"}
//initial view parameters added to layout
Success: {"action":"SCALE_VIEW","message":""}
//login success
Success: {"action":"LOGIN","message":"stresstest2"}
//remote invited local user
Success: {"action":"ONINVITED","message":"stresstest3"}
//local user accepted
Success: {"action":"ONCHATSTARTED","message":"stresstest3"}
//local user published stream (happends auto on ONCHATSTARTED)
Success: {"action":"PUBLISH","message":"stresstest3"}
//remote user published stream
Success: {"action":"ONSTREAMADDED","message":"stresstest3"}
//local user hung up
Success: {"action":"HANGUP","message":"stresstest3"}
//connection to remote user closed
Success: {"action":"ONCHATSTOPPED","message":"stresstest3"}
//tablet set to airplane-mode -> device now offline
Error: {"action":"LOGIN","message":
"com.intel.webrtc.base.WoogeenException: io.socket.engineio.client.EngineIOException:
 websocket error"}
//user got successfully disconnected
Success: {"action":"ONDISCONNECTED","message":""}
//6 reconnection errors
(6)Error: {"action":"LOGIN","message":
"com.intel.webrtc.p2p.WoogeenP2PException: Socket.IO reports connect to signaling ser
ver failed."}
//airplane-mode turned off, wifi connected, connected to server again
```

```
Success: {"action":"LOGIN","message":"stresstest2"}
//local user invites remote user
IntelWebRTC.invite("stresstest3")
//successfully invited remote user
Success: {"action":"INVITE","message":"stresstest3"}
//remote user accepted invitation
Success: {"action":"ONACCEPTED","message":"stresstest3"}
//connection to remote user established
Success: {"action":"ONCHATSTARTED","message":"stresstest3"}
//local user published stream
Success: {"action":"PUBLISH","message":"stresstest3"}
//remote user published stream
Success: {"action":"ONSTREAMADDED","message":"stresstest3"}
//view scaling while in call
IntelWebRTC.scaleView(0.2,"top_right")
//scaling succeeded
Success: {"action":"SCALE_VIEW","message":""}
//remote user disconnected
Success: {"action":"ONCHATSTOPPED","message":"stresstest3"}
```

More Info

Intel WebRTC SDK: https://software.intel.com/en-us/webrtc-sdk

MQTT Connector Details



Broker Name	The name of the broker. This is used by the Call to select the correct broker.
Broker Hostname	Hostname of the broker IPv4 OR hostname!
Port	The port, the broker is running on (1883 is standard for TCP and 8883 is standard for SSL).
TLS	TLS Encryption. Not yet available!
Keep-Alive	The number of seconds the client should be kept alive if no activity takes place.
Clean-Session	Determines whether the session is cleaned after a reconnection.
Last will topic	The topic for which a message is sent if the client dies (testament).
Last will payload	The message for the testament.
Last will retained	Flag to decide whether the message should be retained by the broker.
Last will QoS (Quality of Service)	Flag to decide how often the message will be sent.
Allow multiple executions in reordering	If you send the “order” list to rearrange the given operations, you can decide with the flag whether operations should be executed several times.
Continue after an exception	Flag to decide whether the next operation should be executed after an error or whether the execution completely stops.

How to use a MQTT Connector in a REST-Client

You can choose between 2 operations: PUBLISH and SUBSCRIBE.

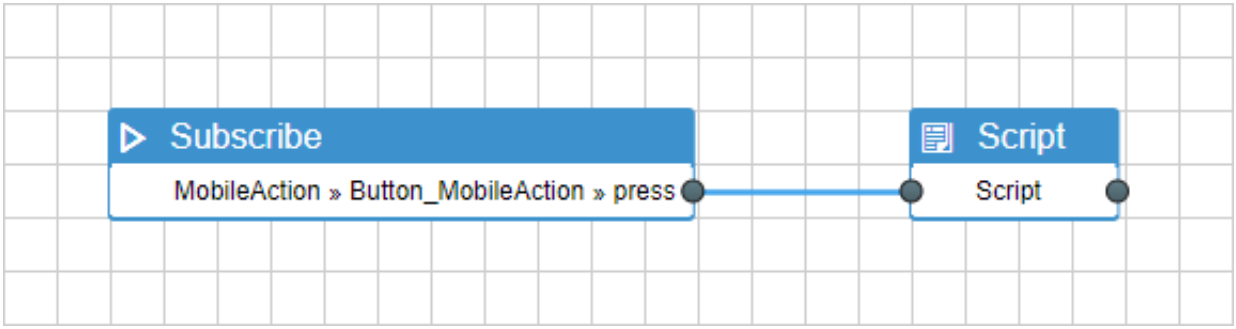
Build the payload like this:

Parameter	Description
operationType	Type of the operation, PUBLISH or SUBSCRIBE.

brokerName	The name of the broker, as it is specified in the Connector Details. If no name is given, the default broker is selected.
returnSet (for SUBSCRIBE operations)	Option which information the incoming messages contain. “STANDARD”: Only the payload and the topic are displayed. “WITH_QUALITY_INFORMATION”: Equivalent to the “STANDARD” but additionally with information whether the message is duplicated / withheld and QoS. “VERBOSE”: As before but with Client ID as well as message ID.

Native Mobile Action

To use other native mobile actions you need to implement Code via Script.



With the Simplifier, you can build your application using web technologies like JavaScript.
If you want to use native functionalities, you need a bridge between those technologies. Therefore you can use pre-installed Cordova Plugins in the Simplifier Client and integrate different predefined APIs.

Here is a list of Plugins to use and a link to their documentation on github:

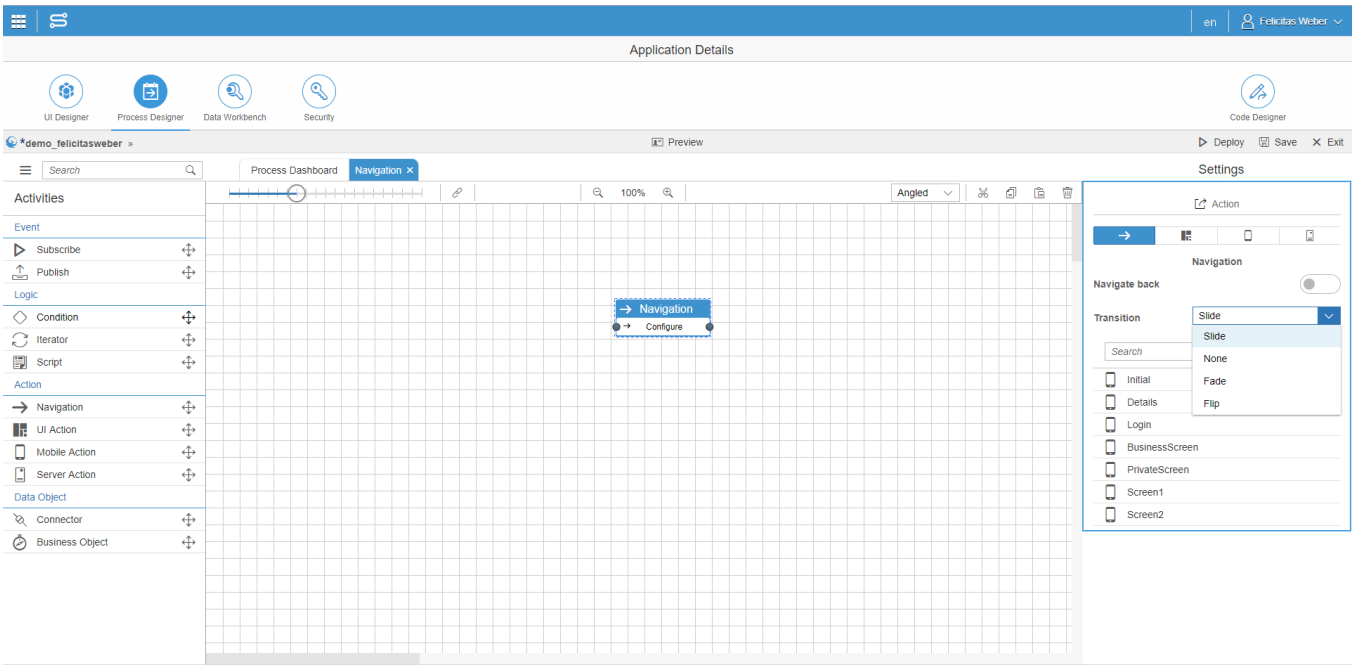
Plugin	**Link to Documentation**
Intel WebRTC (1.2.0)	https://stash.itizzimo.com/scm/cor/cordova-plugin-intelwebrtc
PDF Viewer (1.0.0)	https://stash.itizzimo.com/scm/cor/cordova-plugin-embedded-pdf-viewer.git
Record Video/ Take Picture (3.0.2)	https://github.com/apache/cordova-plugin-media-capture
Device information (2.0.2)	https://github.com/apache/cordova-plugin-device
Device orientation (2.0.1)	https://github.com/apache/cordova-plugin-device-orientation
Storage (6.0.1)	https://github.com/apache/cordova-plugin-file
File IO (1.7.1)	https://github.com/apache/cordova-plugin-file-transfer
Geolocation (4.0.1)	https://github.com/apache/cordova-plugin-geolocation
Network information (2.0.1)	https://github.com/apache/cordova-plugin-network-information
Alerts/ Beeps/ logs (2.0.1)	https://github.com/apache/cordova-plugin-dialogs
Notifications (0.9.0-beta 3)	https://github.com/katzer/cordova-plugin-local-notifications
Statusbar (2.4.2)	https://github.com/apache/cordova-plugin-statusbar
Vibration (3.0.1)	https://github.com/apache/cordova-plugin-vibration
Barcode Scanning (7.1.2)	https://github.com/phonegap/phonegap-plugin-barcodescanner
Beacons (3.6.1)	https://github.com/petermetz/cordova-plugin-ibeacon
Cordova inapp Browser (2.0.2)	https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-inappbrowser
Keyboards (1.2.0)	https://github.com/cjpearson/cordova-plugin-keyboard
Wikitude (8.0.0)	https://github.com/Wikitude/wikitude-cordova-plugin.git
Text to Speach (0.2.3)	https://github.com/vilic/cordova-plugin-tts
Battery (1.0.1)	https://github.com/apache/cordova-plugin-battery-status
Flashlight (3.2.0)	https://github.com/EddyVerbruggen/Flashlight-PhoneGap-Plugin
Splashscreen (5.0.1)	https://github.com/apache/cordova-plugin-splashscreen
Keepscreen on (4.3.0)	https://github.com/EddyVerbruggen/Insomnia-PhoneGap-Plugin
Screen orientation (3.0.1)	https://github.com/apache/cordova-plugin-screen-orientation

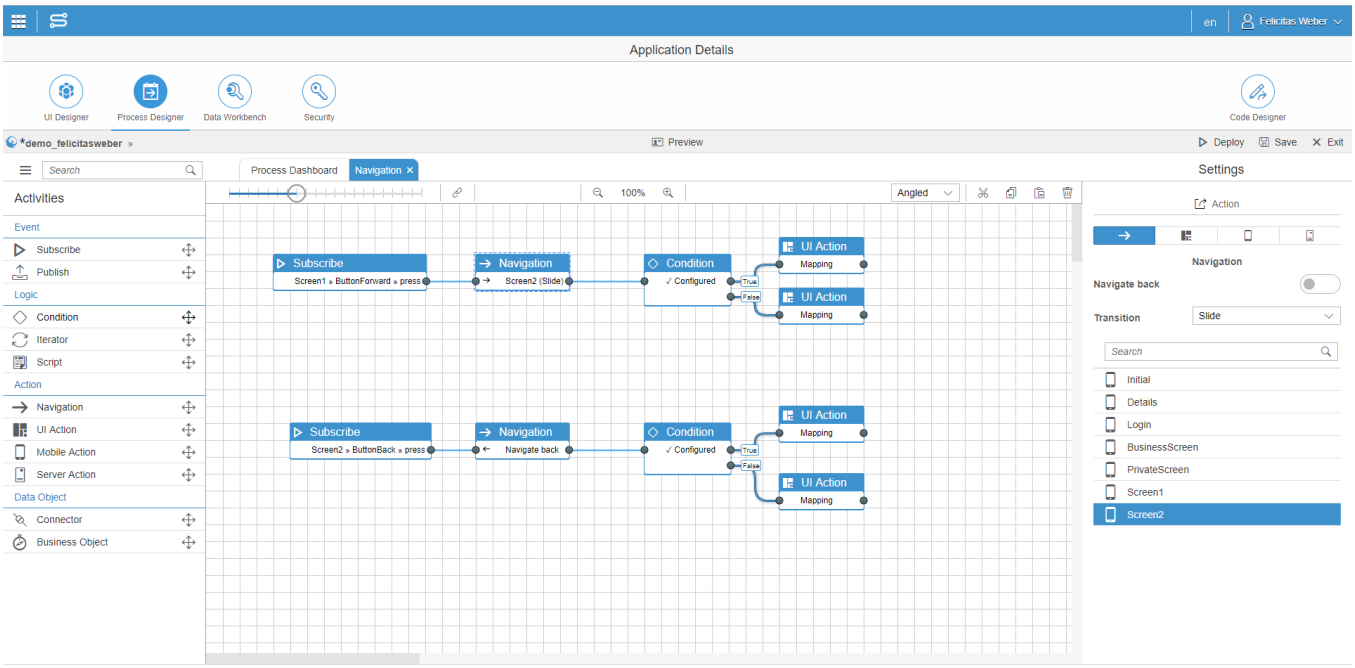
Speech recognition (0.3.0)	https://github.com/macdonst/SpeechRecognitionPlugin
Stream audio and video in a fullscreen (0.1.4)	https://github.com/nchutchind/cordova-plugin-streaming-media
Document viewer (0.9.9)	https://github.com/sitewaerts/cordova-plugin-document-viewer
Document handler (1.0.14)	https://github.com/PolarCape/polarcape-cordova-plugin-document-handler.git
Bluetooth (1.2.2)	https://github.com/don/cordova-plugin-ble-central
Runtime permissions (1.2.0)	https://github.com/apache/cordova-plugin-compat
Remove arm64 (0.0.1)	https://github.com/Ponsen/cordova-plugin-remove-arm64
Whitelist (1.3.3)	https://github.com/apache/cordova-plugin-whitelist
Build info (1.1.0)	https://github.com/Durzan666/cordova-plugin-buildinfo
Background mode (0.7.2)	https://github.com/katzer/cordova-plugin-background-mode
ODG SDK (1.1.0)	https://stash.itizzimo.com/projects/COR/repos/cordova-plugin-odgsdk/browse
Fullscreen (1.0.3)	https://github.com/filfat-Studios-AB/cordova-plugin-fullscreen
Simplifier (2.0.8)	https://stash.itizzimo.com/scm/cor/cordova-plugin-simplifier.git
Record / Play Audio (5.0.2)	https://github.com/apache/cordova-plugin-media
Social Sharing (5.4.1)	https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin
Native toast message (2.7.0)	https://github.com/EddyVerbruggen/Toast-PhoneGap-Plugin
Accelerometer (2.0.1)	https://github.com/apache/cordova-plugin-device-motion

Navigation

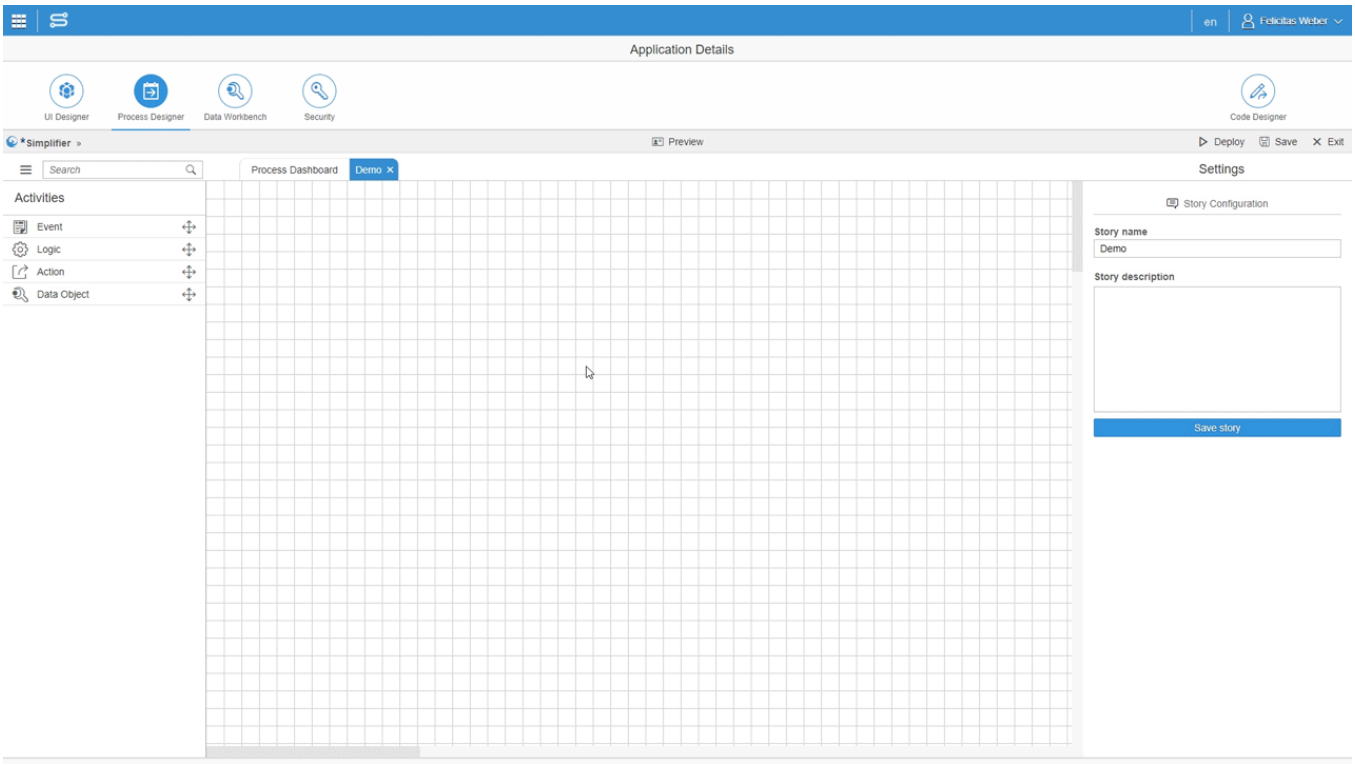
The Navigation Element is used to trigger a navigation from one screen to another. By dragging it to the main screen and selecting it with a click a list of screens to navigate to will appear in the right pane.

You can set that the navigation should navigate back, or choose between several transitions: Slide, Fade, Flip or None.


en
Felicitas Weber
Application Details
UI Designer
Process Designer
Data Workbench
Security
Code Designer
*demo_felicitasweber
Preview
Deploy
Save
Exit
Search
Process Dashboard
Navigation
Settings
Activities
100%
Angled
Event
Subscribe
Publish
Logic
Condition
Iterator
Script
Action
Navigation
UI Action
Mobile Action
Server Action
Data Object
Connector
Business Object
Navigation
Configure
Action
Navigation
Navigate back
Transition
Slide
Slide
None
Fade
Flip
Search
Initial
Details
Login
BusinessScreen
PrivateScreen
Screen1
Screen2



Example: In most cases a navigation will be used when clicking on a button, so by combining a click Event element with a navigation element this can easily be achieved.


en
Felicitas Weber
Application Details
UI Designer
Process Designer
Data Workbench
Security
Code Designer
Simplifier
Preview
Deploy
Save
Exit
Search
Process Dashboard
Demo
Settings
Activities
Event
Logic
Action
Data Object
Story Configuration
Story name
Demo
Story description
Save story

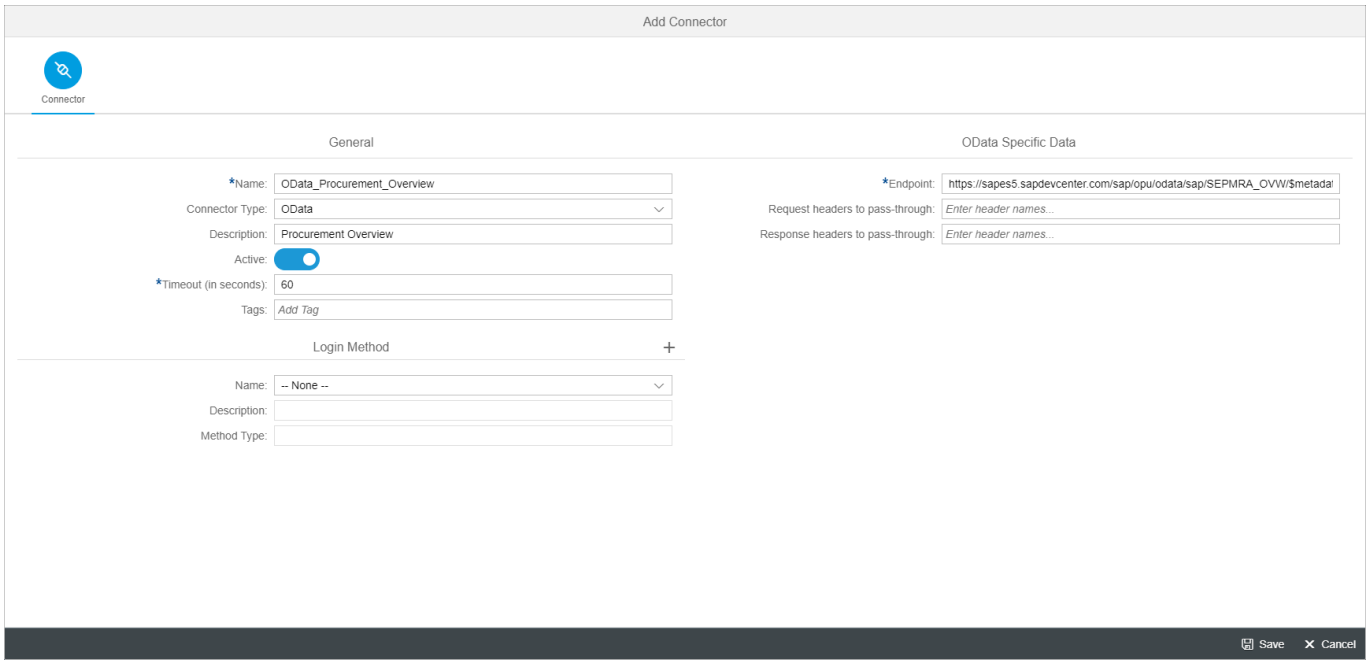
OData Connector Details

OData (Open Data Protocol) enables the creation of REST-based data services that allows resources identified by Uniform Resource Identifiers (URIs) and defined in a data model to be published and edited through the use of HTTP messages by Web clients.

It allows flexible access to the databases and to create, read, update and delete (CRUD) valid data on the web. OData is able to provide consistent semantics for data exchange in client-server communication.

Contrary to the other connectors, OData does not require any connector calls because the metadata is updated automatically.

Create a new connector and select OData as connector type. Within the specific data on the right, define the endpoint. Optionally, you can also define request headers to pass through and response headers to pass through.

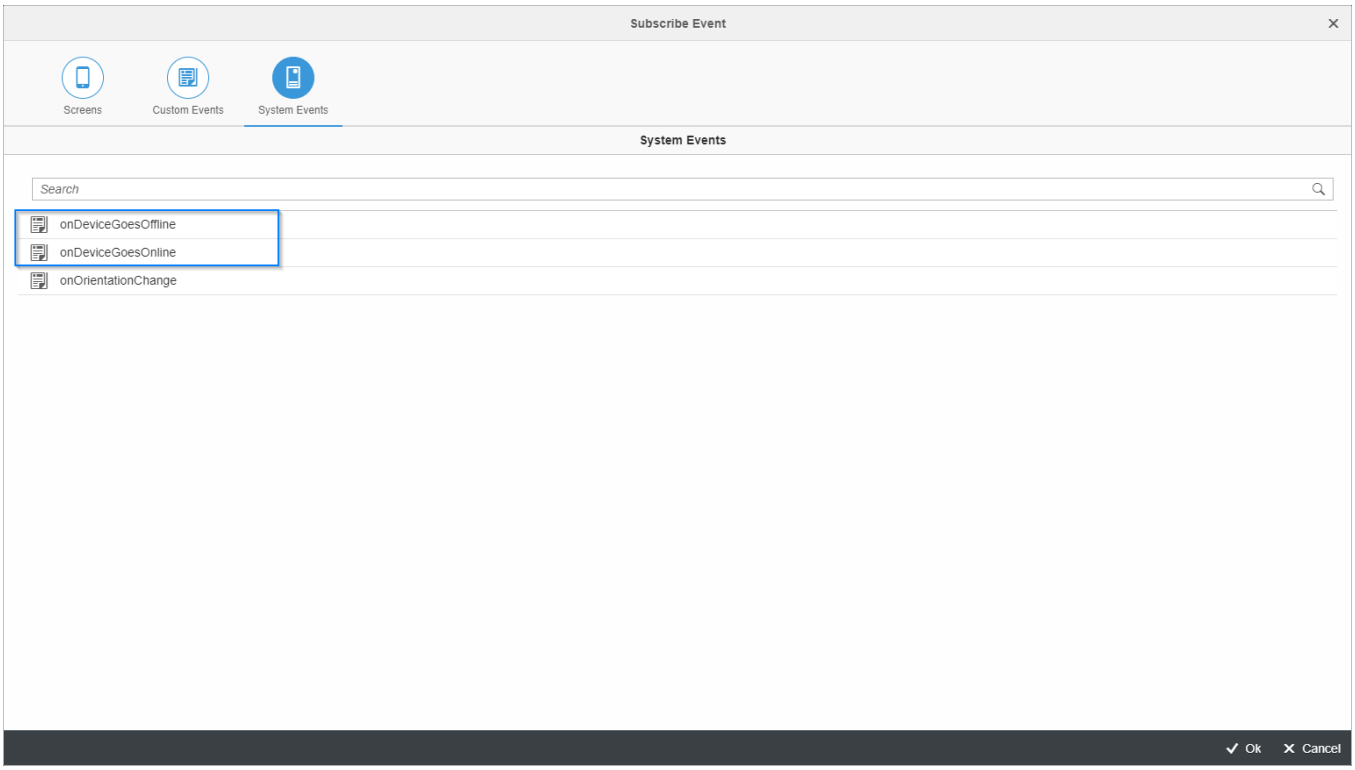

Add Connector
Connector
General
*Name:
OData_Procurement_Overview
Connector Type:
OData
Description:
Procurement Overview
Active:
*Timeout (in seconds):
60
Tags:
Add Tag
Login Method
Name:
-- None --
Description:
Method Type:
OData Specific Data
*Endpoint:
https://sapes5.sapdevcenter.com/sap/opu/odata/sap/SEPMRA_OVW/$metadat
Request headers to pass-through:
Enter header names...
Response headers to pass-through:
Enter header names...
Save
Cancel

Offline Applications

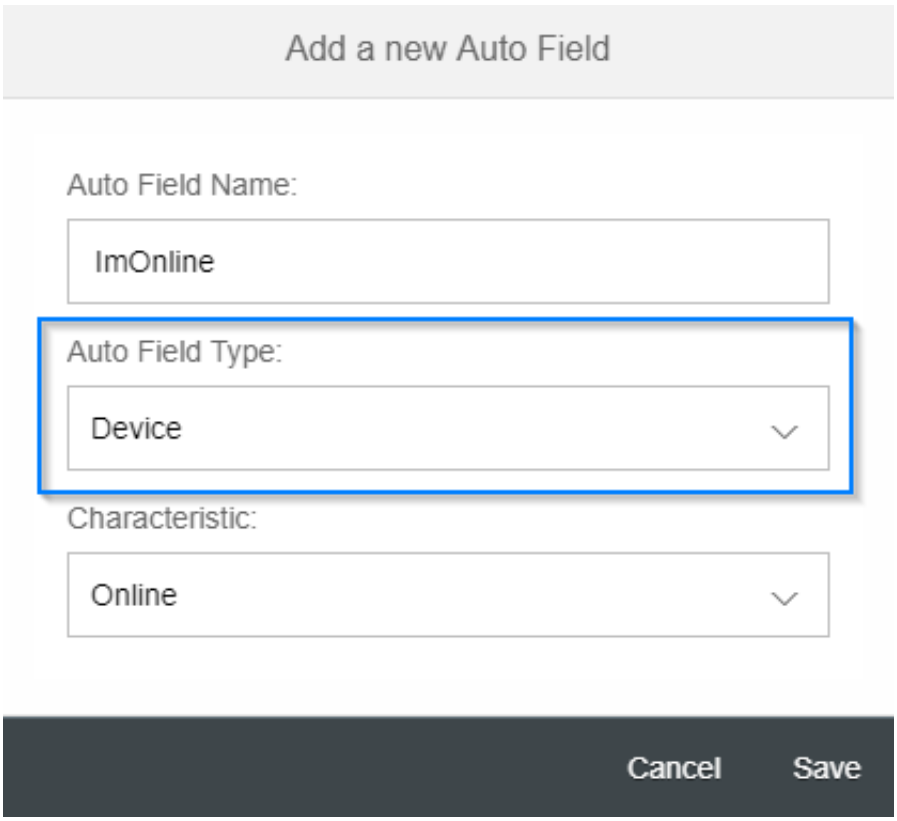
You can react on losing connection of your application/device in the Process Designer.

There are two default events added in every application.

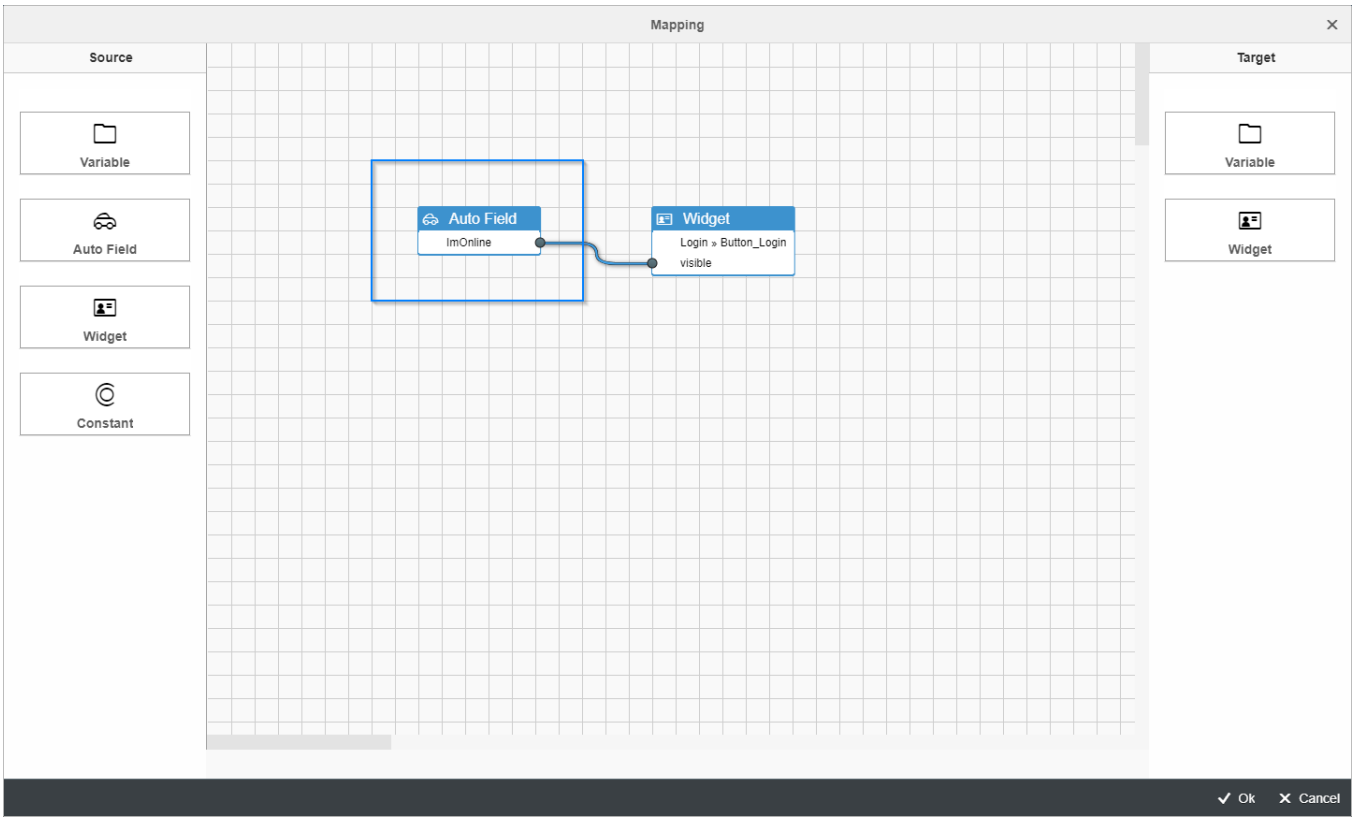
Event	Function
onDeviceGoesOnline	Will be triggered when the device has established a connection to the Wi-Fi or LTE.
onDeviceGoesOffline	Will be triggered when the device has lost the connection to Wi-Fi or LTE.



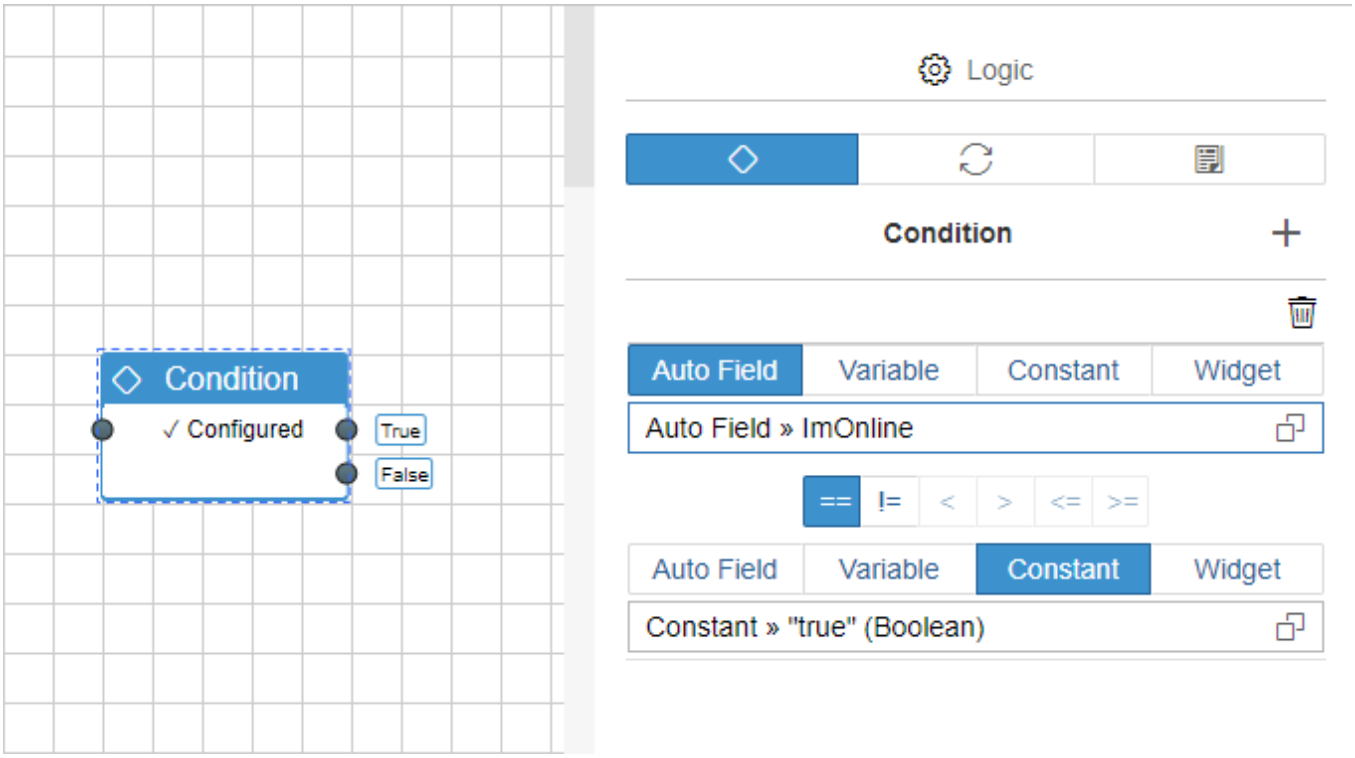
The autofield type “Device” holds the current connection state – characteristic: online. How to create a new autofield, you can see here.



It can be used in every Mapping and also in Conditions.



Mapping



Condition

On Premise

On Premise means that as a customer you run the Simplifier on your own servers. Whether this is a dedicated root server, a virtual machine in VM Ware or a cloud computing instance at a cloud provider like Amazon does not matter in this case.

Dedicated Root Server

Virtual Machine

Elastic Compute Cloud

You are responsible for the installation, operation, maintenance, updates of the Simplifier instance and guaranteed availability.

To learn how to deploy the Simplifier on-premises, look at our checklist.

OPC-UA Call Examples

This section contains examples for the triggering of all three call variants.

READ Call

this.callConnectorCall("TIA_Connector", "TIA_READ" , payload, function(data) {

console.log(JSON.stringify(data, null, 4));

WRITE Call

this.callConnectorCall("TIA_Connector", "TIA_WRITE", payload, function(data) {

console.log(JSON.stringify(data, null, 4));

READWRITE Call

this.callConnectorCall("TIA_Connector", "TIA_READWRITE", payload, function(data) {

console.log(JSON.stringify(data, null, 4));

OPC-UA Connector Calls

This section describes the necessary parameters and data types for OPC-UA connector calls. There are 3 different kind of calls

- Calls for READ operations
- Calls for WRITE operations
- Calls for BROWSE operations

NOTE: Please refer for the necessary data types here.

OPC-UA Connector Data Types

This section describes the nescessary collections as well as their meaning, required to use OPC-UA READ/WRITE connector calls.

Collections

- **Values:** List of values that should be written into nodes.
- **Actions:** List of actions consisting of READ or WRITE that should be performed.
- **Order:** List of the order in which the actions should be performed.
- **Node Names:** List of node names, that should be either written into or read from.
- **Namespace Indices:** List of the corresponding name space index for each node.

OPC-UA Monitoring Requests

After sending the subscription, the OPC-UA Connector will subscribe the given nodes for monitoring requests. Every change will be sent through the websocket to the client and can be received by the *onMessage* function, which has been described here
,

Request JSON

var requestData = {
"operation": String,
"nodes": Array of String,
"namespaceIndices": Array of String,
"publishingInterval": Float,
"clientHandlingID": Integer,
"samplingInterval": Float,
"queueSize": Integer,
"discardOldestItem": Boolean,
"monitoringMode": String,
"returnedTimestamps": String
}

---

operation

The operation, which should be done by the asynchronous OPC-UA connector. Only the following options are valid at the moment.

- MONITORING_SUBSCRIBE

---

nodes

The names of the nodes, which should be subscribed. The name is one of two parts, which defines a node Id.

NOTE: If more than one item should be subscribed with one request, then the amount of *nodes* must be the same as the amount of *namespaceIndices*.

---

namespaceIndices

The namespace indices where the nodes resides

NOTE: If more than one item should be subscribed with one request, then the amount of *namespaceIndices* must be the same as the amount of *nodes*.

---

publishingInterval

The interval, in which the OPC-UA server will publish the changes in the subscribed nodes. According to the *queueSize* all changes, which are still stored in the queue will be published. The used timeunit are *milliseconds*.

---

clientHandlingID

An internal unique handling key for the OPC-UA server to distinguish the monitoring subscriptions.

NOTE: Each request must have an unique clientHandlingID or else only message from the latest subscription will be returned.

---

samplingInterval

The interval in which the OPC-UA server will sample the subscribed nodes for changes. This parameter will be only used if *SAMPLING* is the chosen *monitoringMode*. The used timeunit are *milliseconds*.

---

queueSize

The size of the queue that logs all changes of the subscribed nodes. Each node has an own queue. If the queue is full and new changes occurs, then the oldest changes are discarded automatically.

---

discardOldestItem

If this option is selected, then the oldest entry in the queue of each subscribed node will be discarded regardless whether the queue is not full or not.

---

monitoringMode

The monitoring mode. The following two modes are available:

- **Reporting:** Reports all changes after a defined publishing interval.
- **Sampling:** Samples the nodes after a defined interval and then returns the changes.

---

returnedTimestamps

This option decides what timestamps are returned in each message. The application server timestamp will be returned every time. For now the following options are allowed:

- Both: The timestamp of the OPC-UA Server as well as the source will be returned.
- Neither: Only the application server timestamp will be returned.
- Server: The timestamp of the OPC-UA server will be returned.
- Source: The timestamp of the source of the node will be returned.

OPC-UA Monitoring Requests Examples

This section contains examples for the OPC-UA Monitoring Request Subscriptions

Attention

The status code of a frame might return either “Good” or “Bad” if later case occurs, then also the OPC-UA Error Message will be returned!

Request with Reporting Both Timestamps

```
var requestData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode","myOtherNode"],
"namespaceIndices": [2,2],
"publishingInterval":1000.0,
"clientHandlingID": 1,
"samplingInterval": 1000.0,
"queueSize": 2,
"discardOldestItem": true,
"monitoringMode": "Reporting",
"returnedTimestamps": "Both"
}
```

This request subscribes the nodes “myNode” and “myOtherNode” that resides in namespace 2 to the OPC-UA server, which will report the latest 2 changes after 1 second. The returning message frame contains the application server, OPC-UA server and source timestamps. The oldest item will be dropped.

Returning Message Frames

```
{

“frameType”:”data”,
“subscriptionKey”:”bb827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myNode”,
“namespaceIndex”:2,
“value”:”448828049?,
“appServerTime”:”Wed Oct 19 19:37:42 CEST 2016?,
“OPCUAServerTime”:”Mon Jan 01 01:00:00 CET 1601?,
“OPCUASourceTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}
```

{

“frameType”:”data”,
“subscriptionKey”:”bb827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myOtherNode”,
“namespaceIndex”:2,
“value”:”22223423?,
“appServerTime”:”Wed Oct 19 19:37:43 CEST 2016?,
“OPCUAServerTime”:”Mon Jan 01 01:00:00 CET 1601?,
“OPCUASourceTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}

Request with Reporting Source Timestamp

```
var requestData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode","myOtherNode"],
"namespaceIndices": [2,2],
"publishingInterval":500.0,
"clientHandlingID": 1,
"samplingInterval": 1000.0,
"queueSize": 1,
"discardOldestItem": true,
"monitoringMode": "Reporting",
"returnedTimestamps": "Source"
}
```

This request subscribes the nodes “myNode” and “myOtherNode” that resides in namespace 2 to the OPC-UA server, which will report the last change after 500 milliseconds. The returning message frame contains the application server and source timestamps. The oldest item will be dropped but it does not matter, as the queue can contain only one element.

Returning Message Frames

{

“frameType”:”data”,
“subscriptionKey”:”aa827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myNode”,
“namespaceIndex”:2,
“value”:”324234?,
“appServerTime”:”Wed Oct 19 19:37:42 CEST 2016?,
“OPCUASourceTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}

{

“frameType”:”data”,
“subscriptionKey”:”aa827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myOtherNode”,
“namespaceIndex”:2,
“value”:”2333543?,
“appServerTime”:”Wed Oct 19 19:37:43 CEST 2016?,
“OPCUASourceTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}

Request with Reporting Server Timestamp

```
var requestData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode","myOtherNode"],
"namespaceIndices": [2,2],
"publishingInterval":2000.0,
"clientHandlingID": 1,
"samplingInterval": 1000.0,
"queueSize": 20,
"discardOldestItem": false,
"monitoringMode": "Reporting",
"returnedTimestamps": "Server"
```

}

This request subscribes the nodes “myNode” and “myOtherNode” that resides in namespace 2 to the OPC-UA server, which will report the last 20 changes after 2 seconds. The returning message frame contains the application server and OPC-UA server timestamps. The oldest item will not be dropped.

Returning Message Frame

{

“frameType”:”data”,
“subscriptionKey”:”cc827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myNode”,
“namespaceIndex”:2,
“value”:”4488280898?,
“appServerTime”:”Wed Oct 19 19:37:42 CEST 2016?,
“OPCUAServerTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}

{

“frameType”:”data”,
“subscriptionKey”:”cc827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myOtherNode”,
“namespaceIndex”:2,
“value”:”22223425?,
“appServerTime”:”Wed Oct 19 19:37:43 CEST 2016?,
“OPCUAServerTime”:”Mon Jan 01 01:00:00 CET 1601?,
“statusCode”:”Good”

},
“success”:true

}

}

Request with Reporting Neither Timestamps

```
var requestData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode","myOtherNode"],
"namespaceIndices": [2,2],
"publishingInterval":100.0,
"clientHandlingID": 1,
"samplingInterval": 1000.0,
"queueSize": 5,
"discardOldestItem": true,
"monitoringMode": "Reporting",
"returnedTimestamps": "Neither"
}
```

This request subscribes the nodes “myNode” and “myOtherNode” that resides in namespace 2 to the OPC-UA server, which will report the last 5 changes after 100 millieconds. The returning message frame contains only the application server timestamp. The oldest item will be dropped.

Returning Message Frames

```
{

“frameType”:”data”,
“subscriptionKey”:”dd827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myNode”,
“namespaceIndex”:2,
“value”:”448828042?,
“appServerTime”:”Wed Oct 19 19:37:42 CEST 2016?,
“statusCode”:”Good”

},
“success”:true

}

}
```

```
{

“frameType”:”data”,
“subscriptionKey”:”dd827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{
```

“data”:{

“nodeName”:”myOtherNode”,
“namespaceIndex”:2,
“value”:”22223421?,
“appServerTime”:”Wed Oct 19 19:37:43 CEST 2016?,
“statusCode”:”Good”

},
“success”:true

}

}

Request with Sampling

```
var requestData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode"],
"namespaceIndices": [2],
"publishingInterval":1000.0,
"clientHandlingID": 1,
"samplingInterval": 100.0,
"queueSize": 10,
"discardOldestItem": true,
"monitoringMode": "Sampling",
"returnedTimestamps": "Neither"
}
```

This request subscribes the nodes “myNode” that resides in namespace 2 to the OPC-UA server, which will sample the node every 100 milliseconds and return the last 10 changes after 1 seconds. The returning message frame contains only the application server timestamp. The oldest item will be dropped.

Returning Message Frame

{

“frameType”:”data”,
“subscriptionKey”:”ff827118-f1b0-2170-9937-f8c7e1620107?,
“content”:{

“data”:{

“nodeName”:”myNode”,
“namespaceIndex”:2,
“value”:”448828043?,
“appServerTime”:”Wed Oct 19 19:37:42 CEST 2016?,
“statusCode”:”Good”

},
“success”:true

```
}

}
```

OPC-UA Payload Examples

This section contains examples for the payload of all three call variants.

NOTE: If multiple commands are issued, all fields must contain the same amount of elements i.e. *nodes*, *namespaceInidces*, *values*, *actions* and *order* must have the same length.

READ Call

```
payload = {
"operation":"READ",
"nodes":["myNodeName"],
"namespaceIndices":[3]
}
```

This call will read the value from the node with the name “myNodeName”, which resides in the namespace 3.

```
payload = {
"operation":"READ",
"nodes": ["myNodeName","myOtherNodeName"],
"namespaceIndices": [3,2]
}
```

This call will read the value from the node with the name “myNodeName” which resides in the namespace 3 followed by “myOtherNodeName” residing in namespace 2.

READ Call with order

```
payload = {
"operation":"READ",
"nodes": ["myNodeName","myOtherNodeName"],
"namespaceIndices": [3,2],
"order": [1,0]
}
```

This call will read the value from the node with the name “myOtherNodeName”, which resides in namespace 2 followed by “myNodeName”, which resides in the namespace 3.

```
payload = {
"operation":"READ",
"nodes": ["myNodeName","myOtherNodeName"],
"namespaceIndices": [3,2],
"order": [1,1]
}
```

This call will read the value from the node with the name “myOtherNodeName”, which resides in namespace 2 followed by “myOtherNodeName” residing in the namespace 3.

WRITE Call

```
payload = {
"operation":"WRITE",
"nodes":["myIntNode"],
"namespaceIndices":[3],
```

```
"values":[12]
}
```

This call will write the value 12 into the node with the name “myIntNode”, which resides in the namespace 3.

```
payload = {
"operation":"WRITE",
"nodes": ["myIntNode","myStringNode"],
"namespaceIndices": [3,3],
"values": [12,"No"]
}
```

This call will write the value 12 into the node with the name “myIntNode” which resides in the namespace 3 followed by writing the value “No” into the node with the name “myStringNode” residing in the namespace 3.

WRITE Call with order

```
 payload = {
"operation":"WRITE",
"nodes": ["myIntNode","myStringNode"],
"namespaceIndices": [3,3],
"values": [12,"No"],
"order": [1,0]
}
```

This call will write the value “No” into the node with the name “myStringNode” which resides in the namespace 3 followed by writing the value 12 into the node with the name “myIntNode” residing in the namespace 3.

```
payload = {
"operation":"WRITE",
"nodes": ["myIntNode","myStringNode"],
"namespaceIndices": [3,3],
"values": [12,"No"],
"order": [1,1]
}
```

This call will write the value “No” into the node with the name “myStringNode” which resides in the namespace 3 followed by writing the value “No” into the node with the name “myStringNode” residing in the namespace 3.

READWRITE Call

```
payload = {
"operation":READWRITE",
"nodes": ["myIntNode","myOtherNodeName"],
"namespaceIndices": [3,3],
"actions":["WRITE","READ"],
"values":[12,""]
}
```

This call will write the value 12 into the node with the name “myIntNode” which resides in the namespace 3 followed by reading the value of the node with the name “myOtherNodeName” residing in the namespace 3.

NOTE: Please note that values for reading operations will be ignored.

READWRITE Call with order

```
payload = {
"operation":READWRITE",
"nodes": ["myIntNode","myOtherNodeName"],
"namespaceIndices": [3,3],
"actions":["WRITE","READ"],
"values":[12,""],
"order":[1,0]
}
```

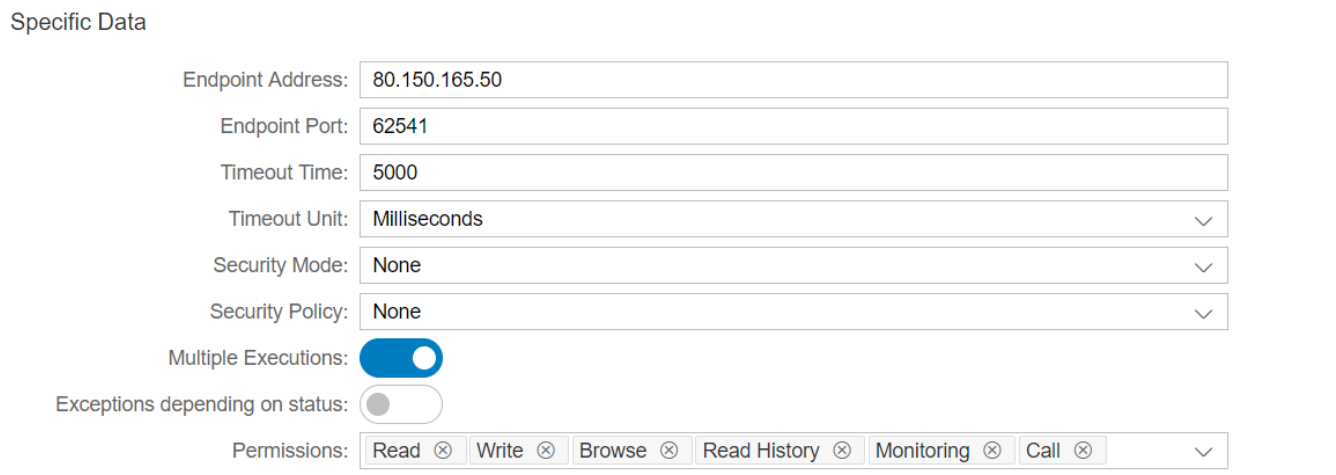
This call will read the value of the node with the name “myOtherNodeName” which resides in the namespace 3 followed by writing the value 12 into the node with the name “myIntNode” residing in the namespace 3.

```
payload = {
"operation":READWRITE",
"nodes": ["myIntNode","myOtherNodeName"],
"namespaceIndices": [3,3],
"actions":["WRITE","READ"],
"values":[12,""],
"order":[1,1]
}
```

This call will read the value of the node with the name “myOtherNodeName” which resides in the namespace 3 followed by reading the value of the node with the name “myOtherNodeName” residing in the namespace 3.

NOTE: Please note that values for reading operations will be ignored.

OPC/UA Connector Details



Endpoint Address

The IPv4 address of the OPC-UA server, this server should connect.

Endpoint Port

The port of the OPC-UA server, this server should connect.

Timeout Time

The timeout time for one OPC-UA operation. The maximum possible timeout time will be one hour regardless of the given input.

Timeout Unit

The corresponding timeout unit for one OPC-UA operation. Three units are supported

1. MILLISECONDS
2. SECONDS
3. MINUTES

Security Mode

The security mode the OPC-UA server. The following modes are available (For now only “None” is accepted)

1. Sign
2. None
3. Sign and Encrypt

Security Policy

The encryption of the OPC-UA server connection. The following four modes are available (For now only “None” is accepted)

1. None
2. Basic 128 Bit RSA 15
3. Basic 128 Bit
4. Basic 128 Bit SHA 256

Multiple Executions

If you reorder the operations in your code and want the possibility to do multiple executions, activate the switch.

Exceptions depending on status

Not implemented yet!

Permissions

Determines what kind of OPC-UA operations this connector might perform. There are currently 7 modes

1. Read
2. Write
3. Browse
4. Read History
5. Write History
6. Monitoring
7. Call

Open Authorization (OAuth)

To set OAuth as authentication, make sure you have **administrator rights**.

After you have logged in as usual in the login mask with your user credentials, click on your name in the upper right corner and then on the settings.

Switch to the Authentication tab in the upper toolbar and click the add icon on the right.

Authentication Settings

First, fill out the following fields:

Name	Specify any name.
Priority	0 – The priority can be ignored in this case.
Mechanism	Select OAuth.
Mechanism Type	Windows Live – At the moment there is only Windo

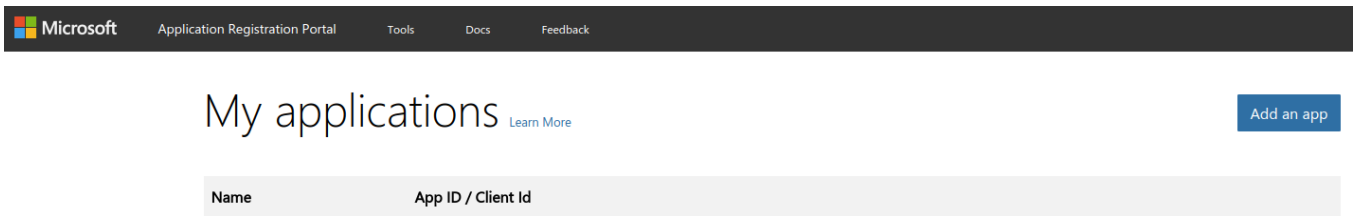
OAuth Settings

You need to fill out the Client Id and the Client Secret. To do this, switch to Windows Live in your browser.

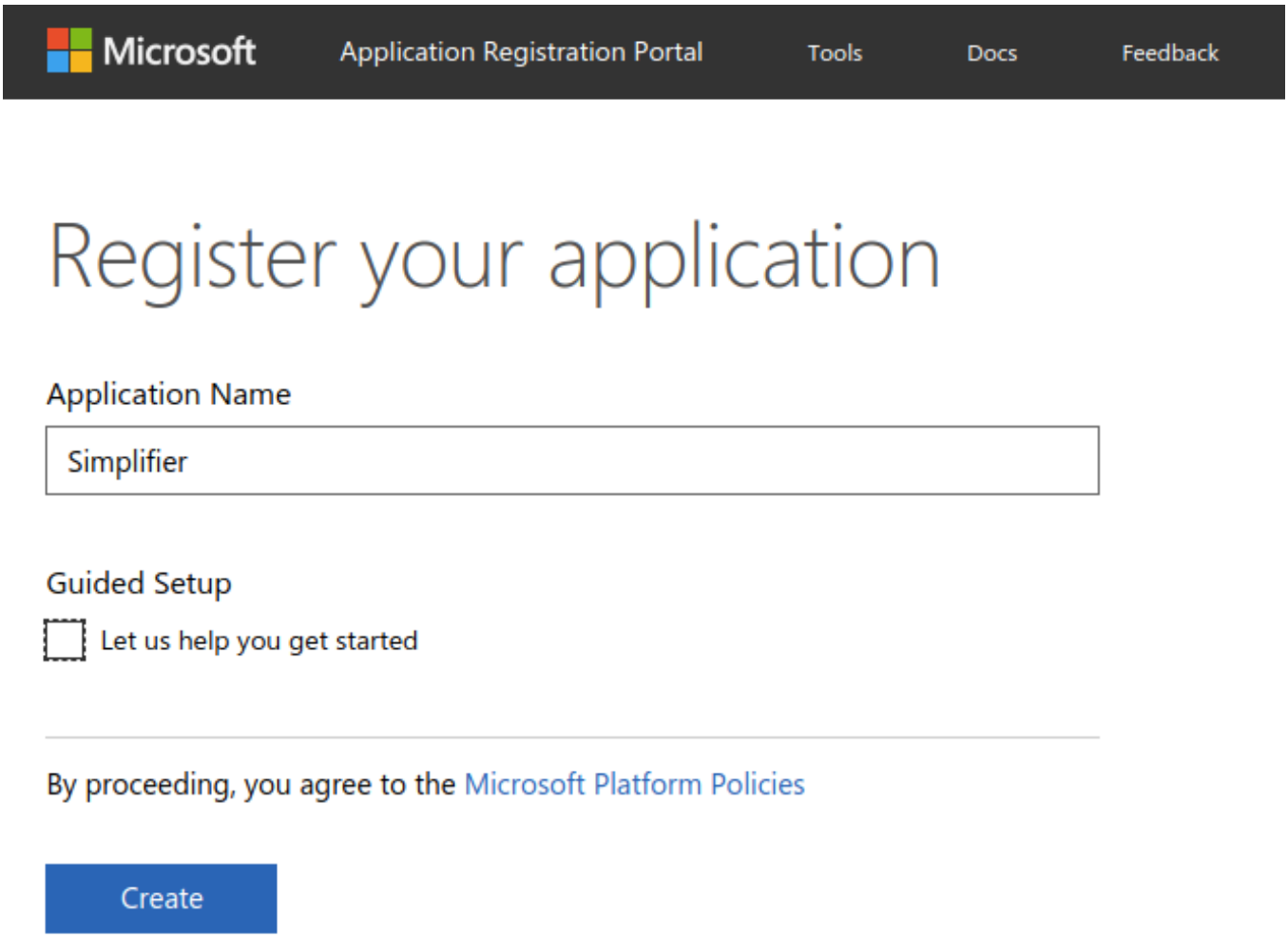
```
https://apps.dev.microsoft.com
```

First, you need to log in to Windows Live with your credentials.

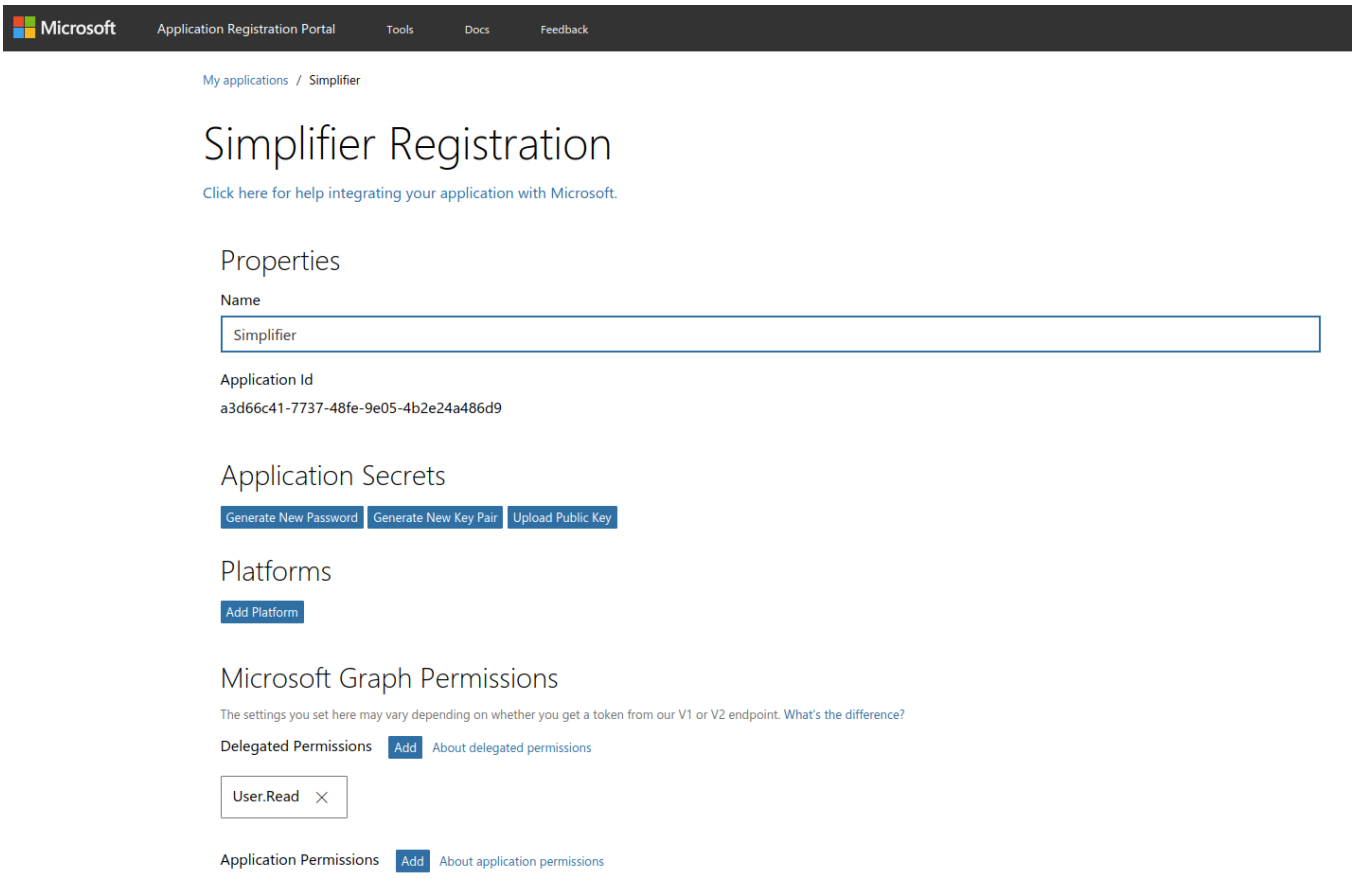
After you have successfully logged in, click the “Add an App” button on the right.



A new window opens in which you have to assign a name for the application. In this case, it’s “Simplifier”.



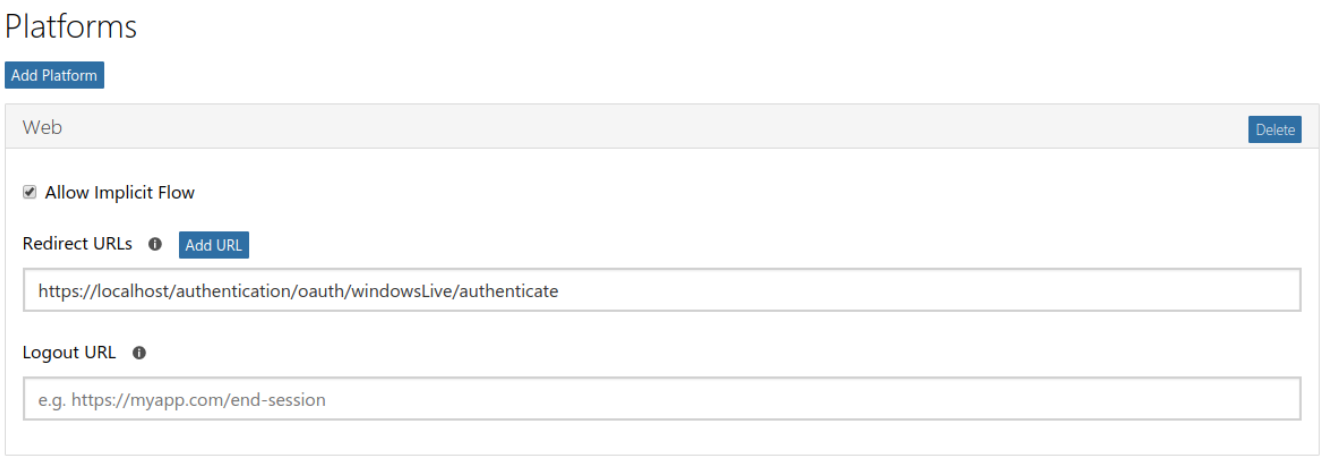
After you have assigned a name, click on “Create”.



In the next step, copy the application Id and paste it into the Simplifier at Client Id.

Click the “Generate New Password” button. A new pop-up opens, in which you can copy the password and paste it into the Simplifier at Client Secret.

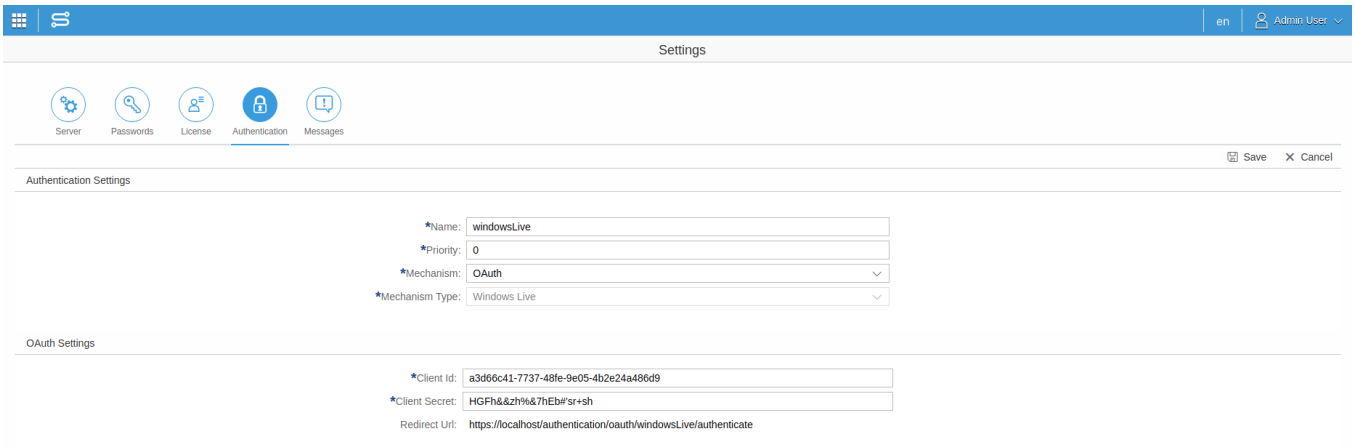
Next, click on the “Add Platform” button under the “Platforms” top item and select “Web”.



After you have selected”Web”, insert the redirect url from the simplifier, which was generated automatically.

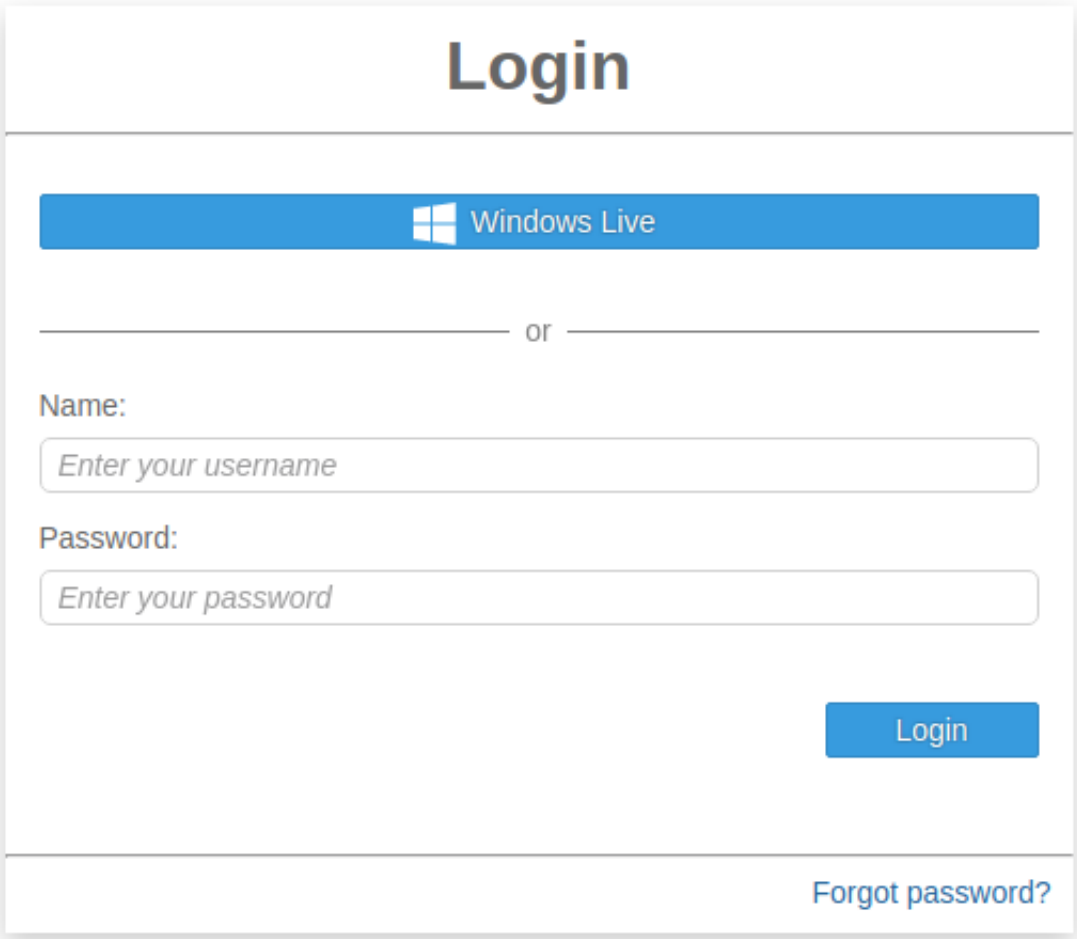
Click on “Save” at the bottom left.

Your screen should now look like this:

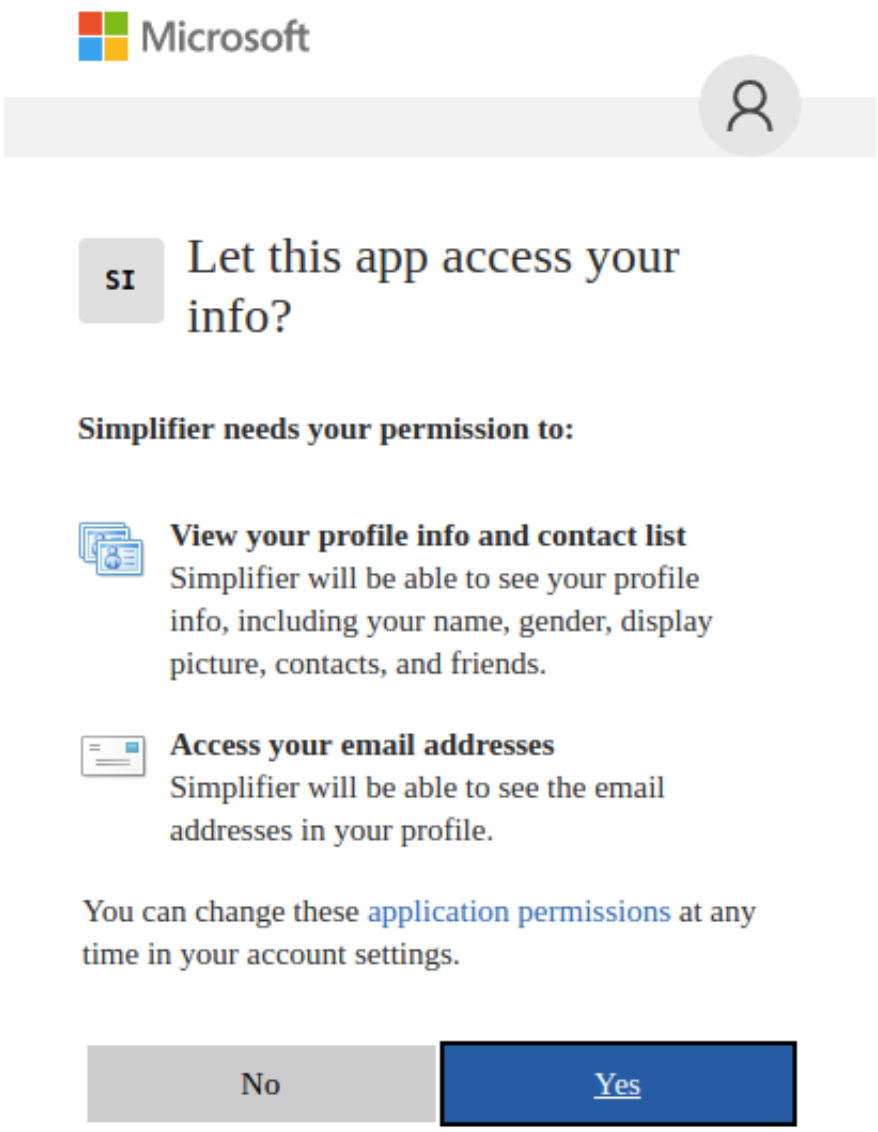


Do not forget to click on “**Save**” in the Simplifier as well.

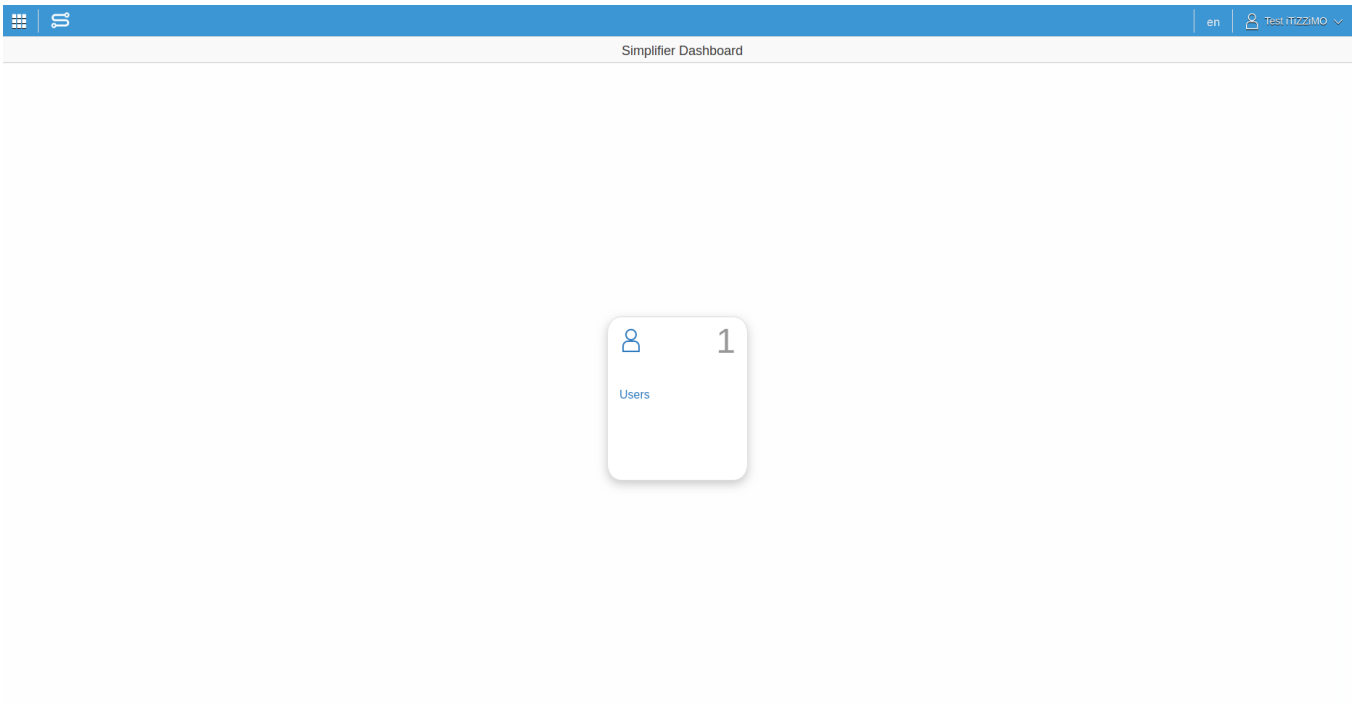
If you now want to log in to Simplifier, the corresponding button becomes clickable.



Click on the “Windows Live” button in the login mask. For the first time you will be redirected to the Microsoft login screen, where you have to log in with your email and password.



You must authorize the Simplifier by clicking on “Yes”. **If you do not grant the permissions, a login in Simplifier via OAuth is not possible.**



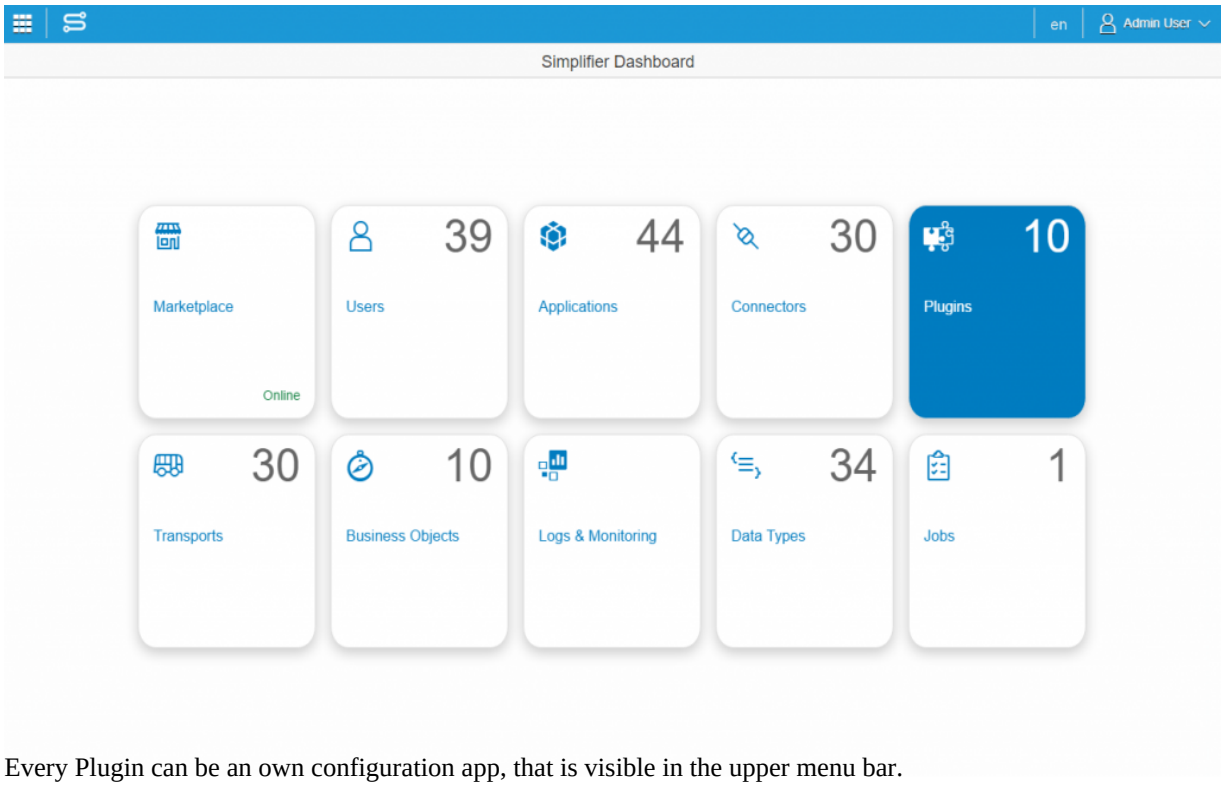
After clicking “Yes”, you will be redirected to the Simplifier. A user with minimal rights is automatically created by the first login. This user must be assigned the corresponding and required rights by an admin.

Other Clouds

By providing the Simplifier Image on Docker Hub, we enable deployment on various public clouds and structures, e.g. Container as a Service (CaaS) on Google Container Engine (GKE), Amazon Web Services (AWS), Azure and SAP Cloud.

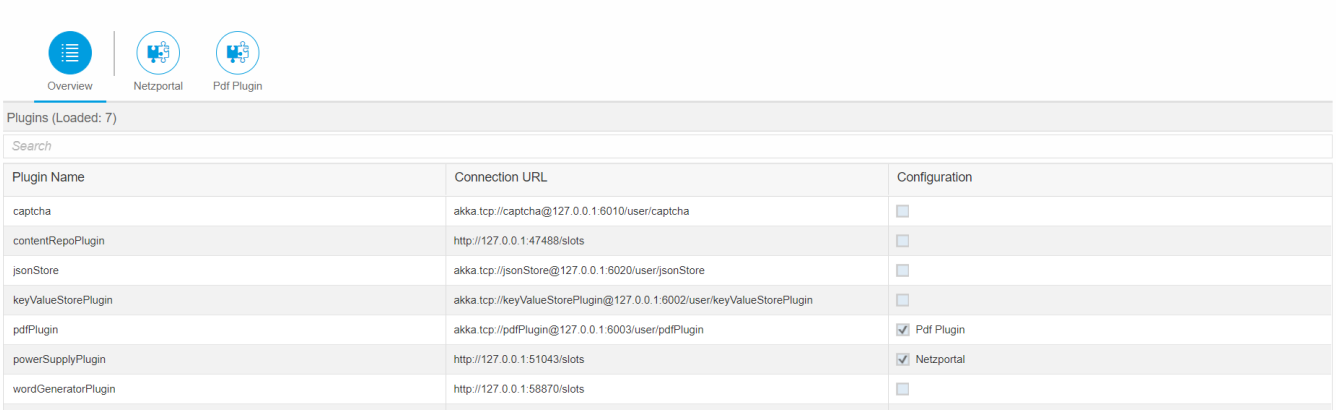
Overview

To see the active Plugins, click on the Plugin tile.



Every Plugin can be an own configuration app, that is visible in the upper menu bar.

You can see the following information in the table above:



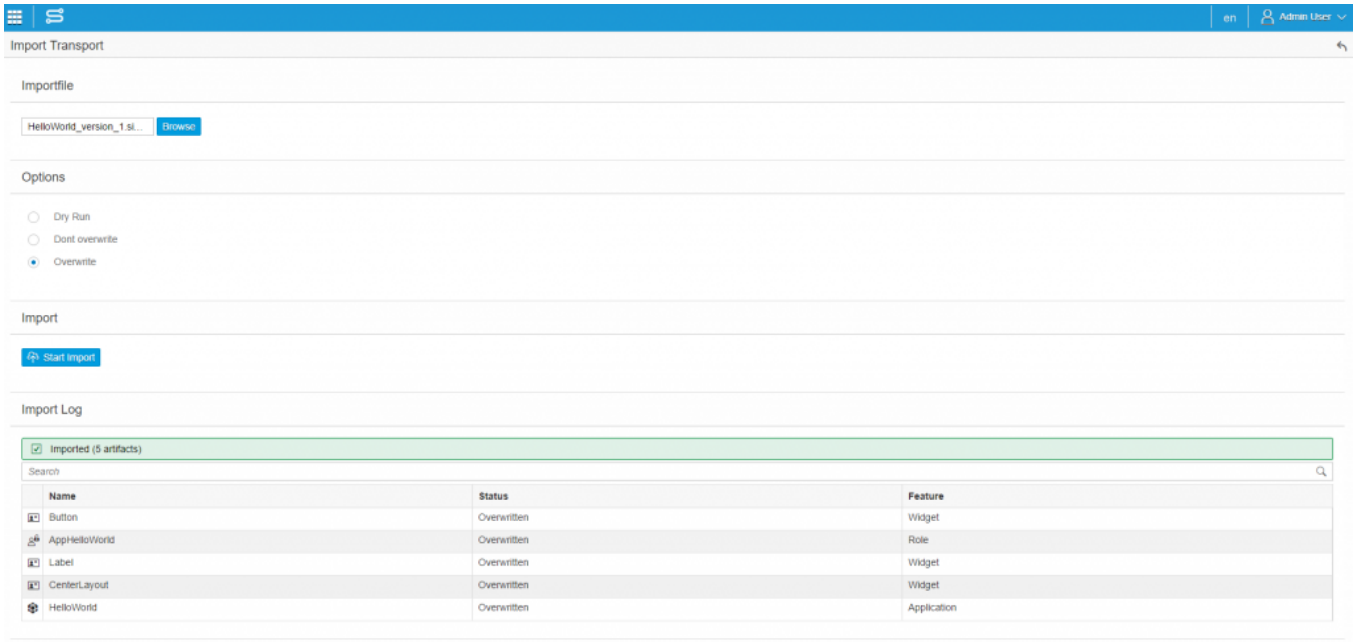
PluginName
Technical name of the Plugin.

Connection URL
On what server runs the plugin and which tcp port does it use.

Configuration
If this plugin offers you a configuration user interface, the checkbox is activated and you can see them as tabs in the upper bar.

Overwrite data

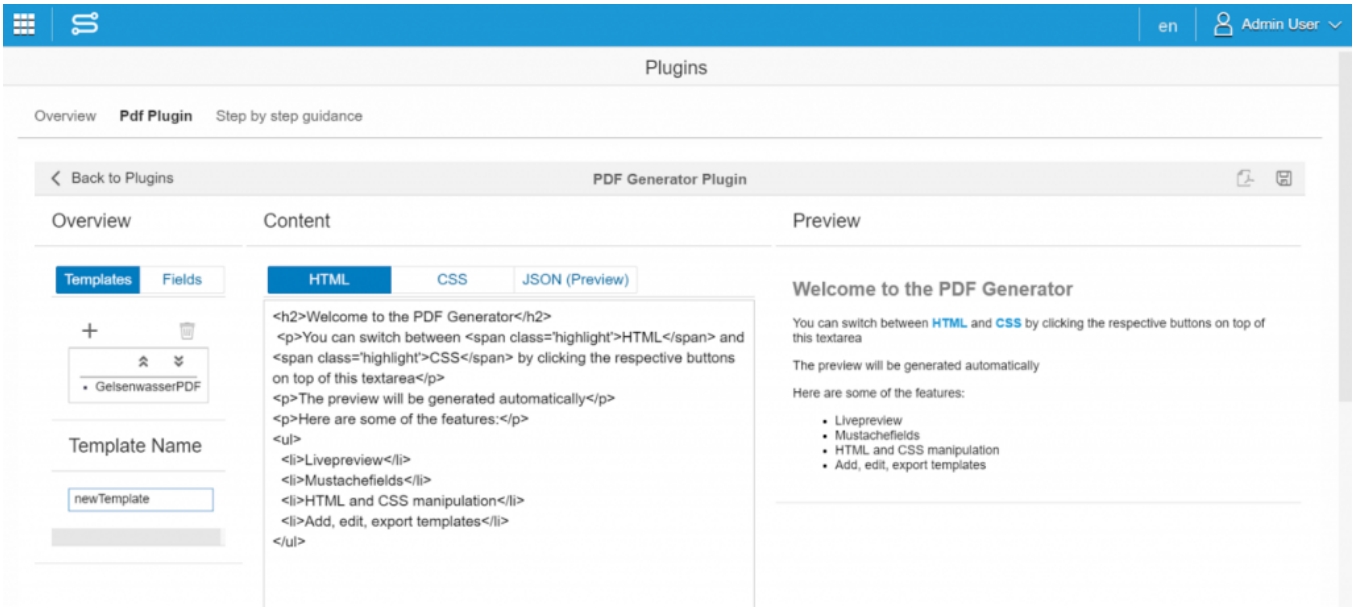
Be careful if you choose to overwrite the existing data. If you changed e.g. the properties of the widget “Button” and import a transport which contains the “old” button widget, your new changes will be overwritten. To avoid this, delete your existing application on the system and import your transport with the option “Don’t overwrite”!



PDF Plugin

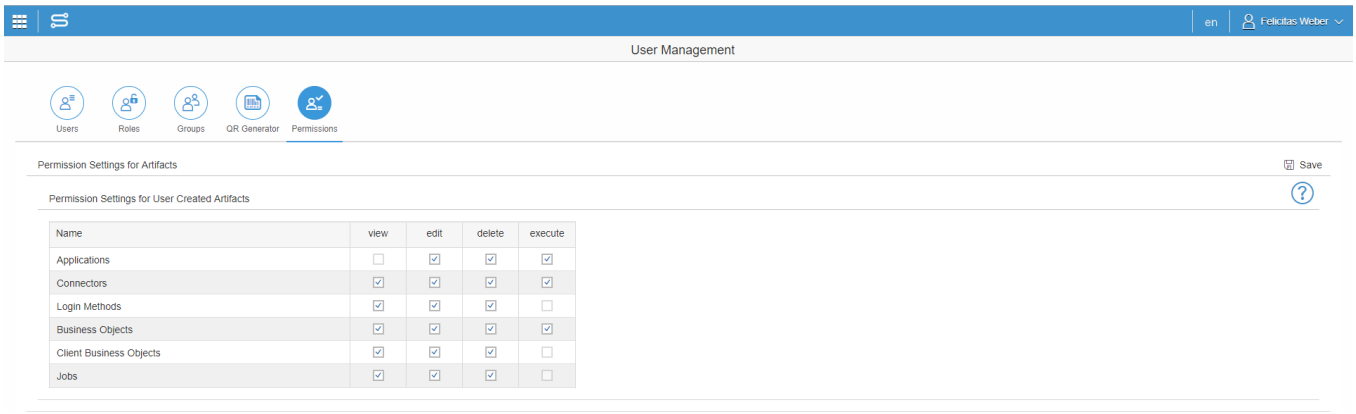
The PDF Plugin can be used to generate files with a fix layout but dynamic values.

The sub-articles describe how to use it.



Permissions

The ownership concept provides that a user only can use artifacts, that were created by himself.



You have the possibility to share your own artifacts with other users. The permissions a user receives for a shared artifact has to be set by the admin.

Plugin development

Open developer plugin-Doc in new tab

Plugins

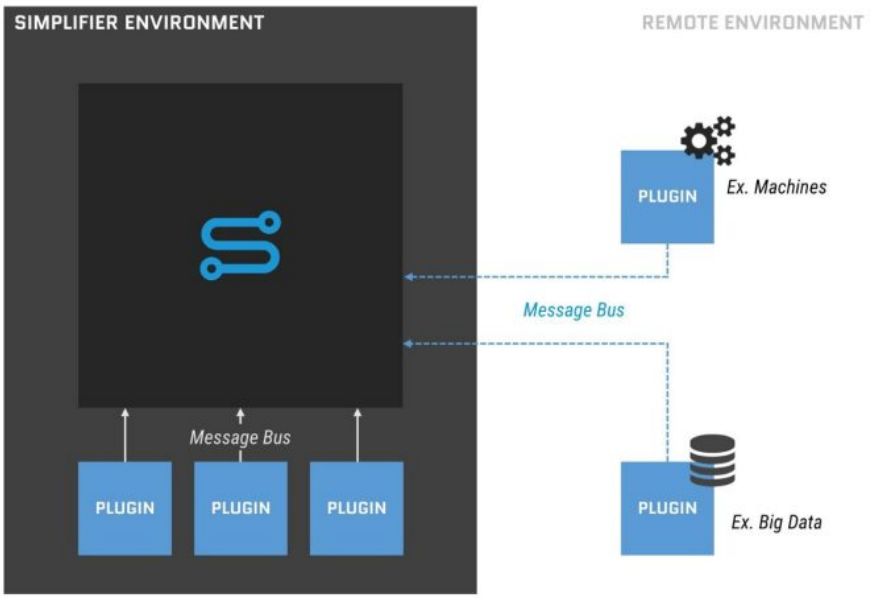
Plugins extend the Simplifier core features in all manners. They are standalone programs that can contain

- Proprietary interfaces that are not supported by connectors
- Embedded system calls for hardware programming
- Business logic like our business objects
- User interfaces that are different to our business apps

In short: Every time you are not able to configure logic, interfaces or ui elements within the Simplifier, you can do it with your own plugin.

Plugins are coded JAR Files that can be developed in the classic way like Eclipse, Netbeans or another Java IDE. The Plugins can be ran standalone and remotely and communicate with the Simplifier via akka message bus system over tcp. Every Plugin has its own port number and, if necessary, also its own ip addresses.

Plugins can be coded in any programming language, it’s only necessary that an HTTP interface will be provided.



Simplifier Plugin Architecture

Plugins via Script

```
this.callPlugin(pluginName, slot, payload, callback, showBusyIndicator, failonError,
failCallback)
```

pluginName	the name of the plugin
slot	slot (function, interface) within the plugin
payload	a JSON object with the required parameters for the plugin
callback	function, which is called after the successful execution of the plugin
showBusyIndicator	boolean value that indicates whether the screen has to be blocked by a loading bar (true) or not (false)
failonError	boolean value that indicates whether the plugin should be called in case of an error of the function passed via “failCallback” (false) or not (true)
failCallback	function, which is called in case of an error in the plugin, is false for “failOnError” is passed

Preview

Applications can be deployed rapidly and previewed within a standard web browser and on the mobile device using the Simplifier Mobile Client.

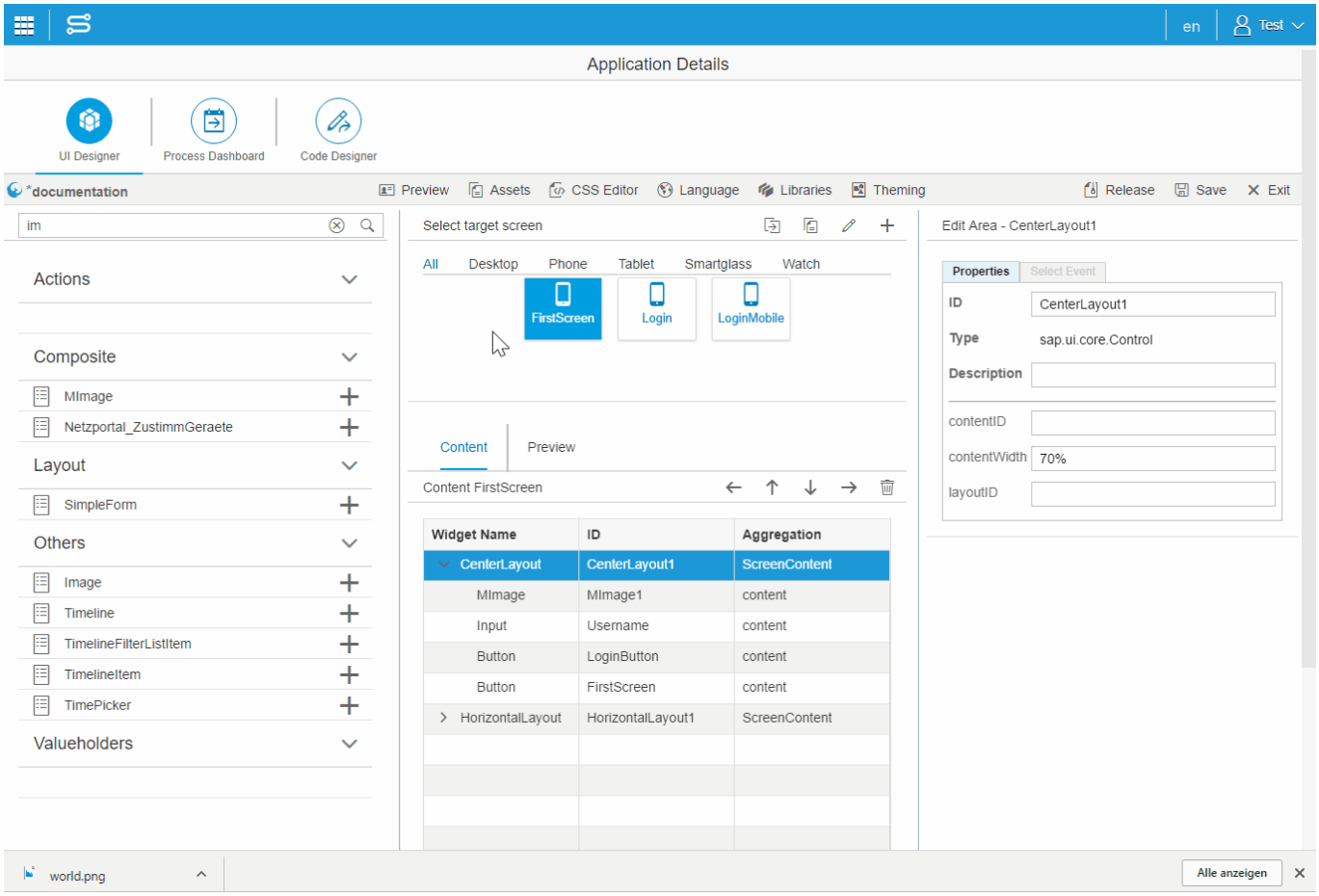
If you click on preview, it opens a new tab. Everytime you will deploy your application, the tab will be reloaded automatically.

You can simulate your preview for different mobile devices with the Chrome Developer Tools or use it for debugging.

To access the DevTools, open your app preview in Google Chrome. Either:

- Select the **Chrome menu**  at the top-right of your browser window, then select **Tools** > **Developer Tools**.
- Or use Ctrl+Shift+I (or Cmd+Opt+I on Mac).

Via the toggle device toolbar, you can choose different devices like Galaxy S5, Iphone 6 or Ipad to preview your application.



(We used the short-code Ctrl+Shift+I to access the DevTools.)

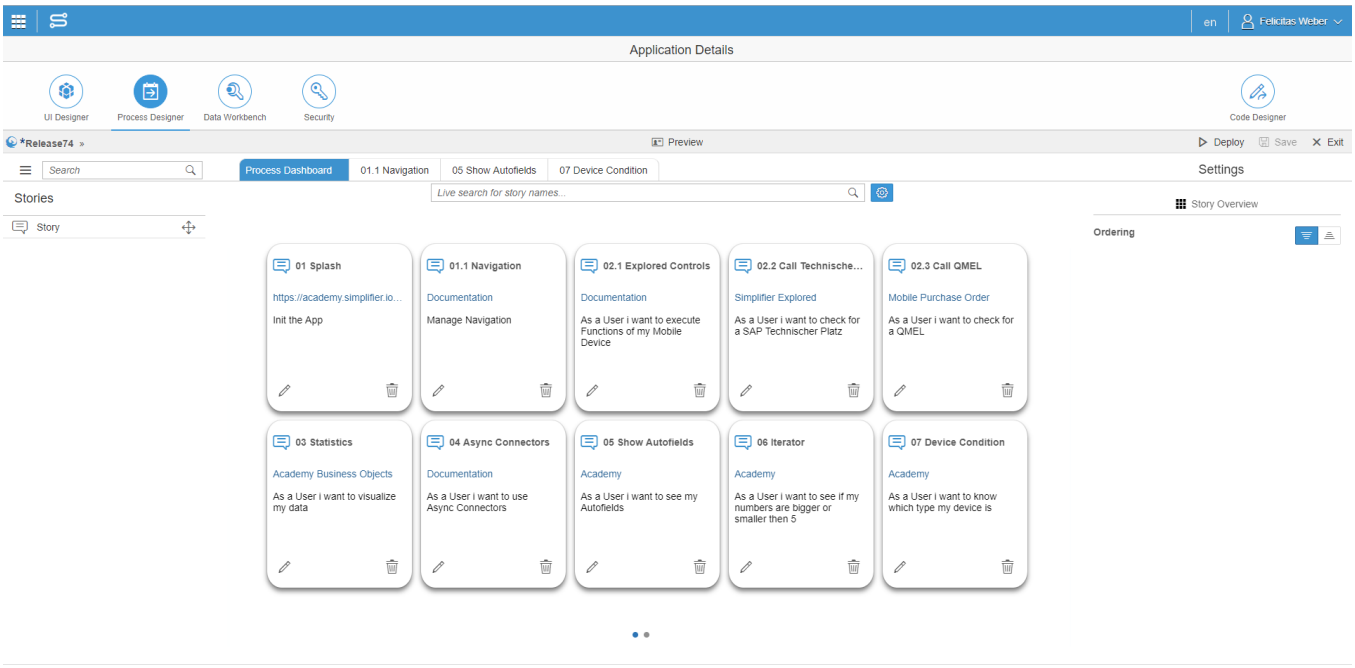
Process Designer

The Process Dashboard

The Process Designer allows to define process flows as the result of an event in an intuitive, graphical way. The Process Designer tab is located next to the UI Designer. By that, it follows the classic workflow of creating an app: First, you design the screens of your app, followed by the definition of the workflow between these elements in the Process Dashboard.

The Process Dashboard is divided into different User Stories which you can save individually, so it’s possible to work on it simultaneously with several people. After saving the entire app in the UI Designer or deploying it in the Process Designer, all stories will be merged into one. If you save an user story in the Process Designer, the changes won’t be displayed in the preview as long as it’s not deployed.

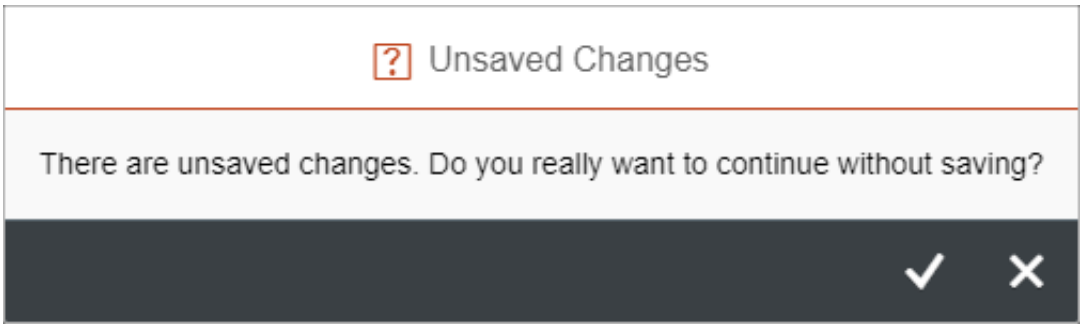
You can add a new story by dragging “Story” on the left into the middle. A pop-up will appear to set a name, an external link and the link text for it and a story description. You can open multiple user stories at the same time. To switch between the user stories, simply switch between the tabs.



To edit or delete an existing one, click on the equivalent icons. The stories will be orderd alphabetically (or by number if you use them). It is possible to sort the user stories in ascending or descending order. You can do this on the right.

By double clicking on the whole tile, you will be forwarded to the Process Designer where you can work on your story.

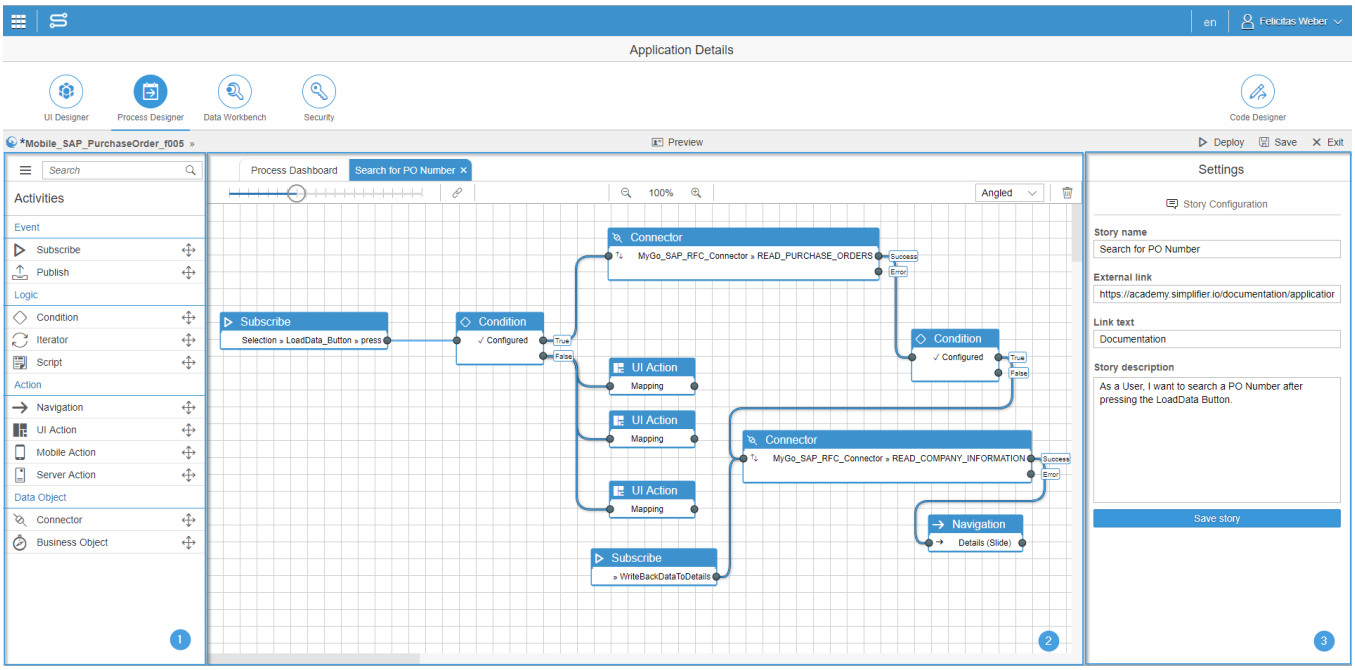
If you leave a user story and have unsaved changes, the following pop up appears:



Process Designer

The main view of the Process Designer is split into three main sections:

1. The selection of the building blocks on the left, namely Event, Logic, Action and Data Object. The activities are subdivided again. You can see this in the subitems of this page.
2. The drawing area in the center.
3. On the right side you can either see your user story, edit it and save it. Or you have also the possibility to configure your shapes on the right side.



Deep Search

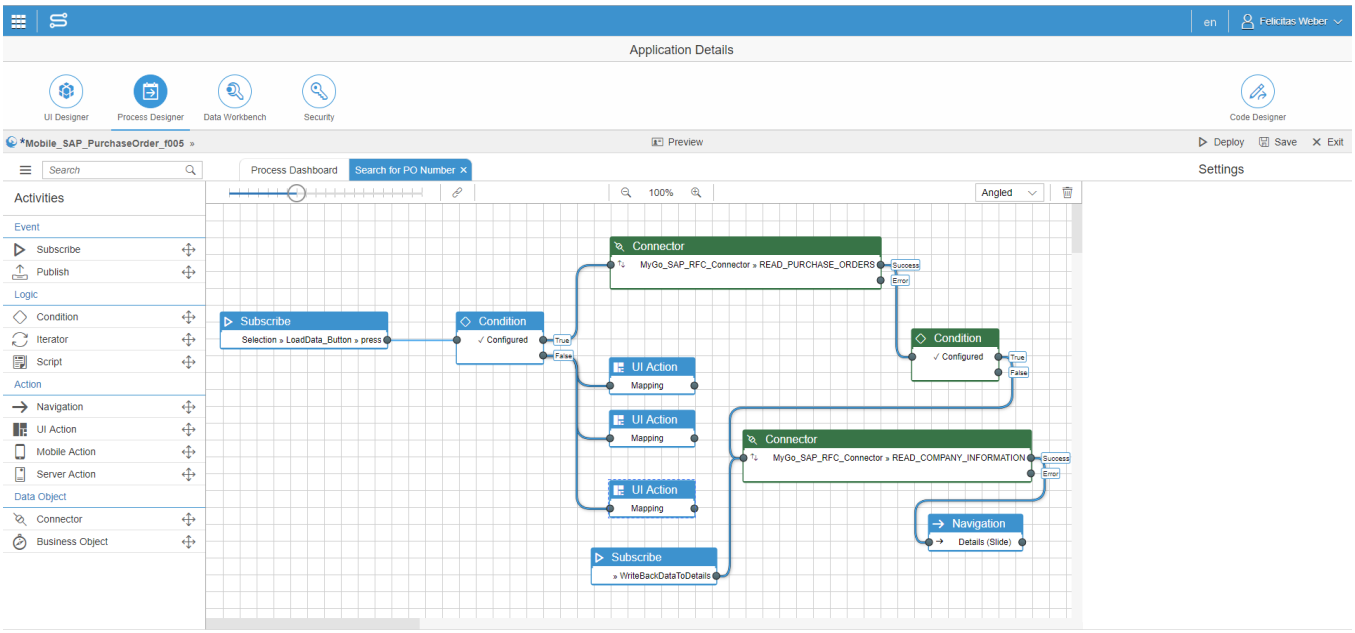
You have the possibility to search within the **Process Dashboard** for occurrences of elements.

Therefore, the user can now switch between the live search for story names (frontend search) and the search for widget names (backend search) in the Process Designer. While the search for story names continues to apply a live filter to the list of tiles, the search for widget names must be explicitly confirmed by pressing the search button or entering the search field. So there are two different UIs.

1. **Live search for story names:**

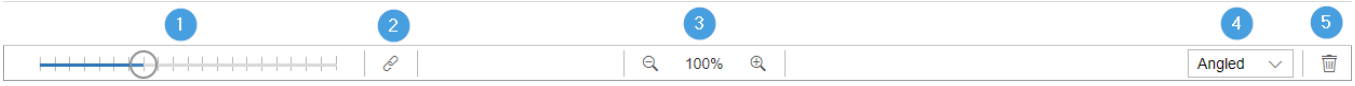
2. **Search for widget names:**

As a result all activities using widgets whose names match the search term are highlighted in green.



Dynamic Process Settings

The toolbar within the Process Designer gives you the control of appearance of your process.

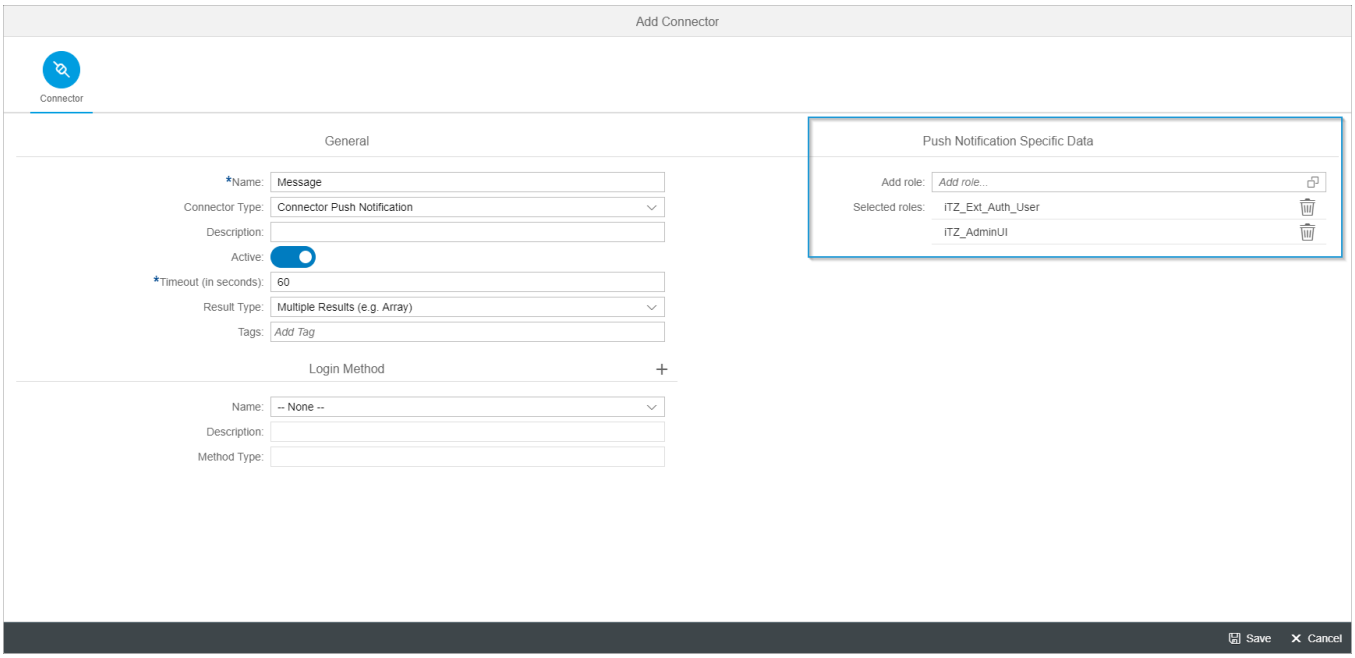


Number	Function	Description
1	Grid resolution	This slider changes the size of the grid in combination with the “snap to grid” function (2), you have full control to properly align your process.
2	Snap to grid	When deactivating (click on the icon), you can freely move all activities.
3	Zoom	You can zoom out (left icon), zoom in (right icon) and reset the zoom with the icon in the middle.
4	Connection settings	

Push Notification Connector Details

With the Push Notification Connector, you can send a notification to all logged in users who have the selected role on the Simplifier instance.

Select the roles to which you want to send a message in the connector configuration mask. Unfortunately, a multi-select is not yet possible and you have to select several roles in succession.



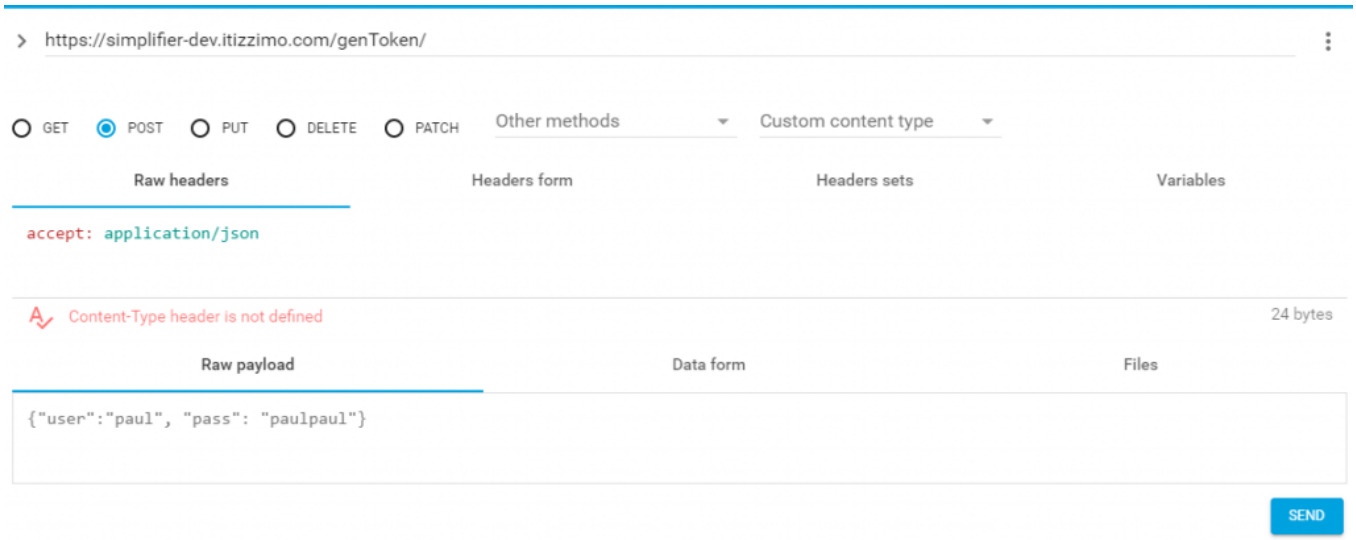
To create the connector calls for this connector, take a look at Push Notification Connector Calls.

How to use a Push Connector in a REST-Client (e.g. the Advanced Rest-Client)

Step 1: You need a Token

```
Request Type: POST
Content Type: application/json
Payload:
{"user":"<username>", "pass": "<password>"}
```

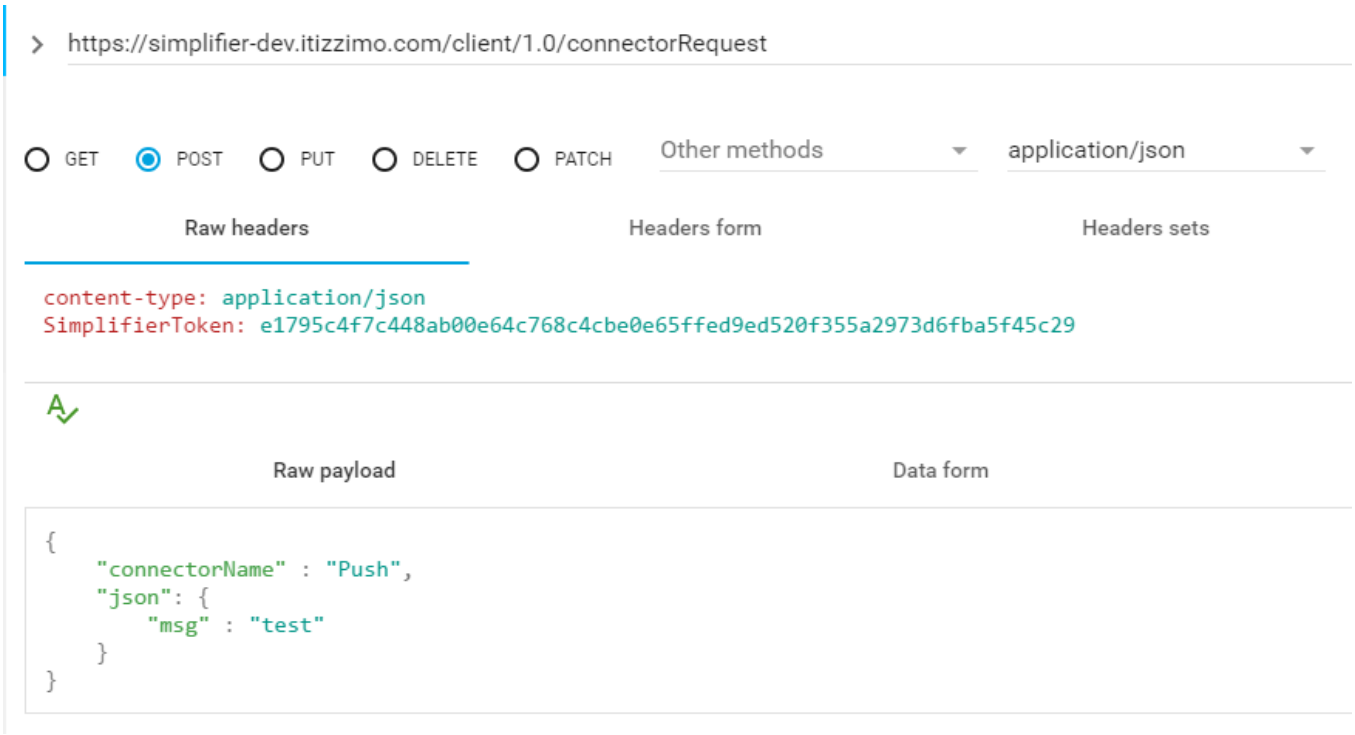
```
Returns:
{
"result": "someprettylongtoken",
"success": true
}
```



Step 2: Make a Connector request with this Token

- Example 1: Send a message as push notification

```
XHR Type: POST
Content Type: application/json
Header: SimplifierToken
Payload:
{
 "connectorName" : "<connector_name>",
 "json": {
 "msg" : "<some_string>"
 }
}
```



Sample:

```
{
 "connectorName" : "Push",
 "json": {
 "msg" : "test"
 }
}
```

- Example 2: Send an order as push notification

```
XHR Type: POST
Content Type: application/json
Header: SimplifierToken
Payload:
{
  "connectorName" : "<connector_name>",
  "json": {
  "msg": "{"order_id":"<NotificationMessage> <header> <message_id>1234567890</message
_id><created>date_time</created> <request> <site>9999</site> // Asset Nr.<resource>St
ation_1</resource> // Ressource/ Workplace<order>variant_123</order> // Order-/ SFC N
r.<variant>variant_123</variant> // Variant Nr.<assy>assembly_123</assy> // Assembly
```

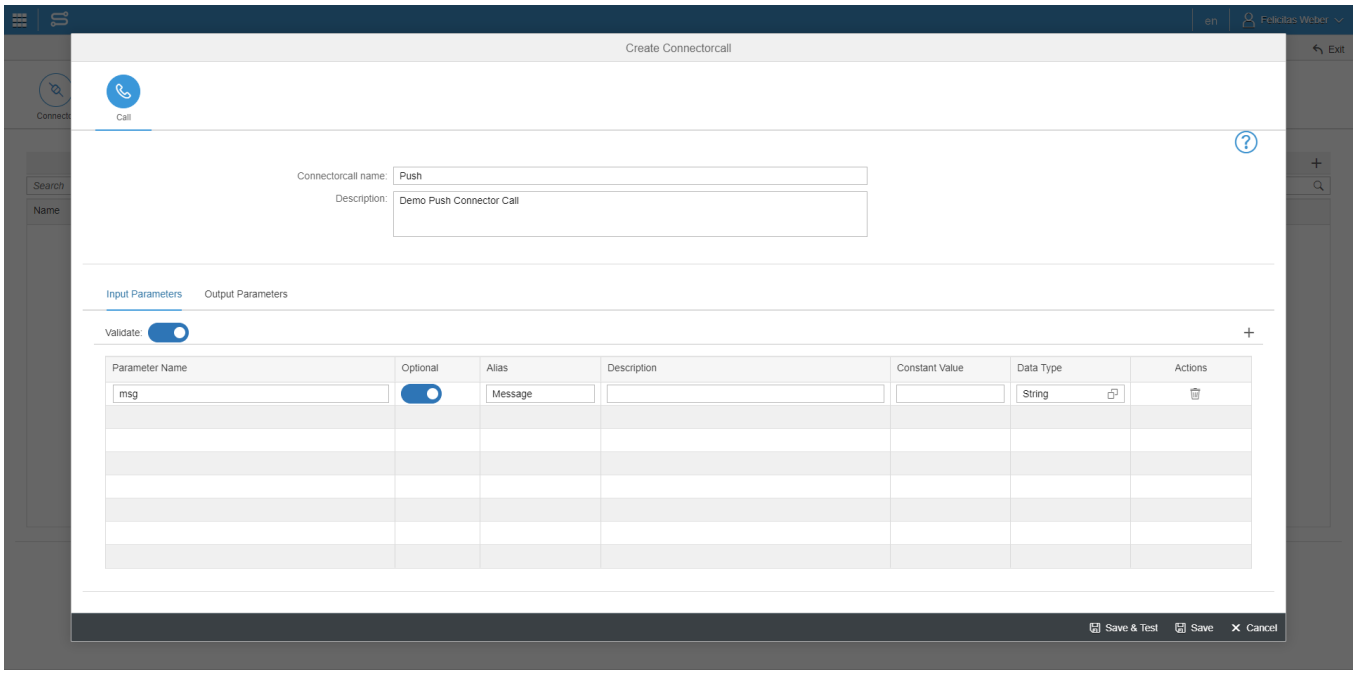
```
Group Nr. <time>180</time> // Time allowance </request> </header> </NotificationMessa
ge>"}"
}
}
```

Push Notifications Connector Calls

The Push Notifiaction Connector Call requires only one input parameter to be defined:

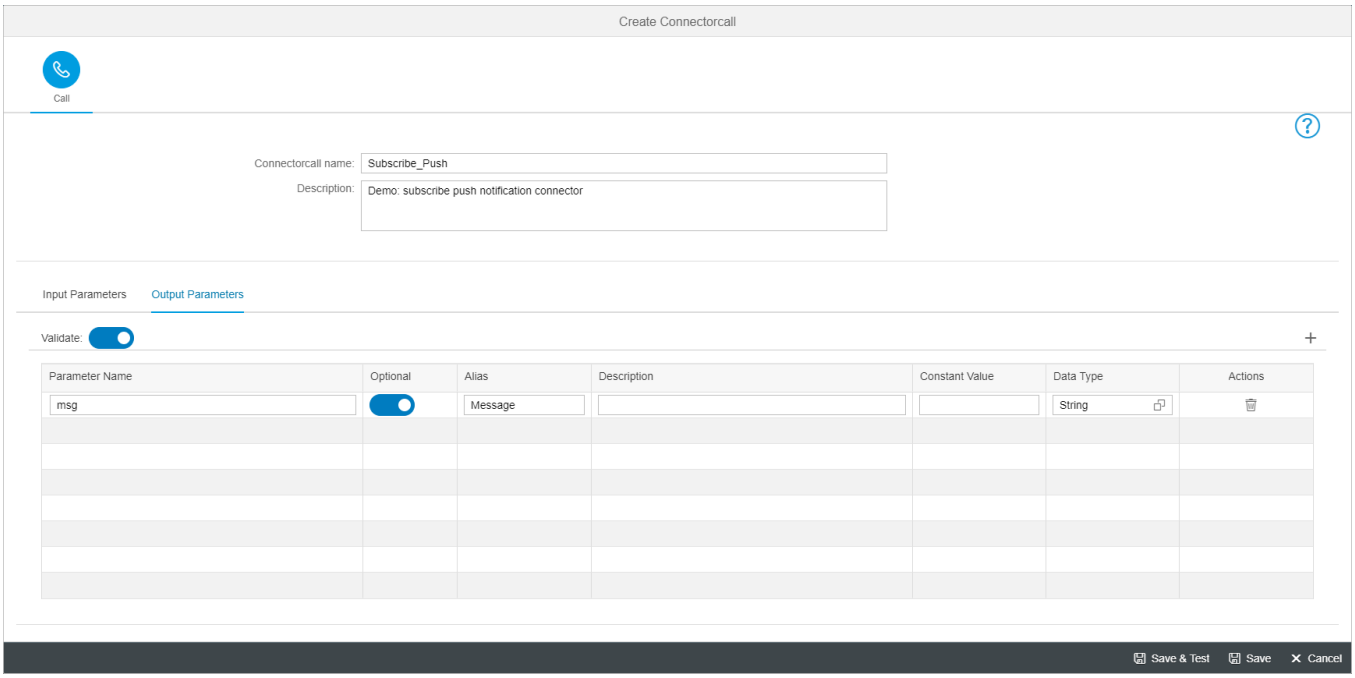
Parameter Name	Data Type
msg	String

Example:



You can subscribe to a push notification connector. For this, create a new connector call in which you only have to define one output parameter. Input parameters are not necessary for this.

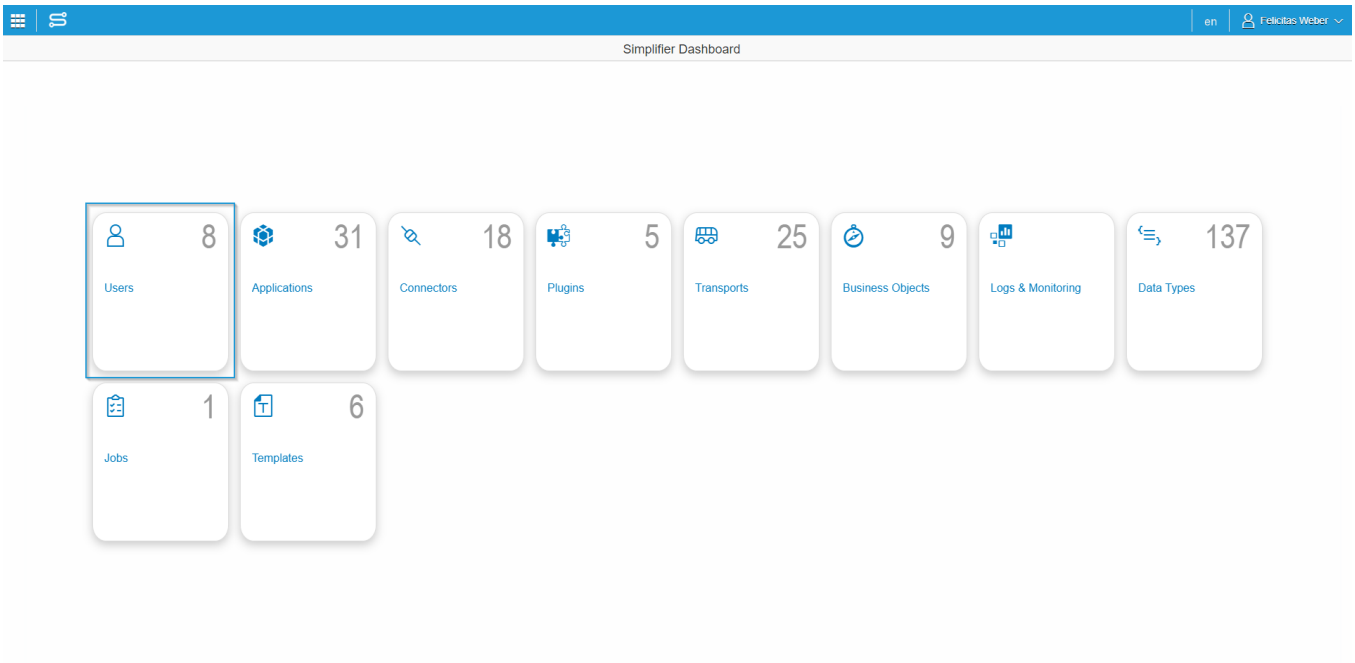
Parameter Name	**Data Type**
msg	String

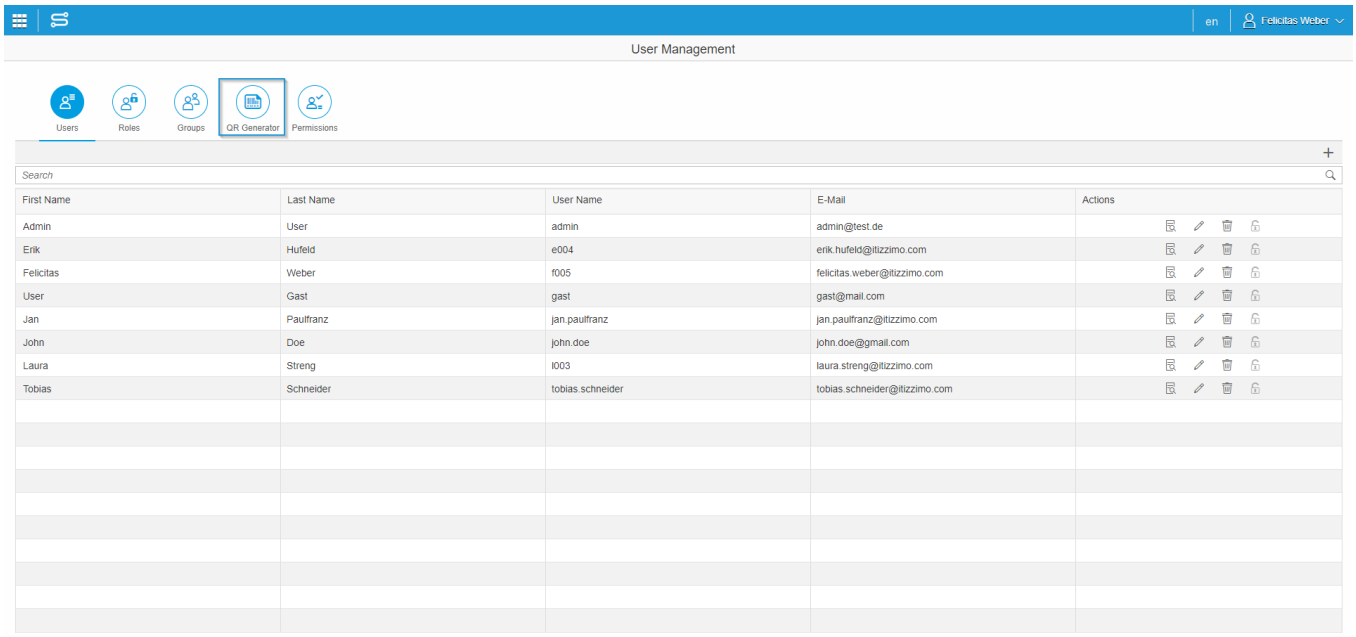

Create Connectorcall
Call
Connectorcall name:
Subscribe_Push
Description:
Demo: subscribe push notification connector
Input Parameters
Output Parameters
Validate:
Parameter Name
Optional
Alias
Description
Constant Value
Data Type
Actions
msg
Message
String
Save & Test
Save
Cancel

QR Login-Generator

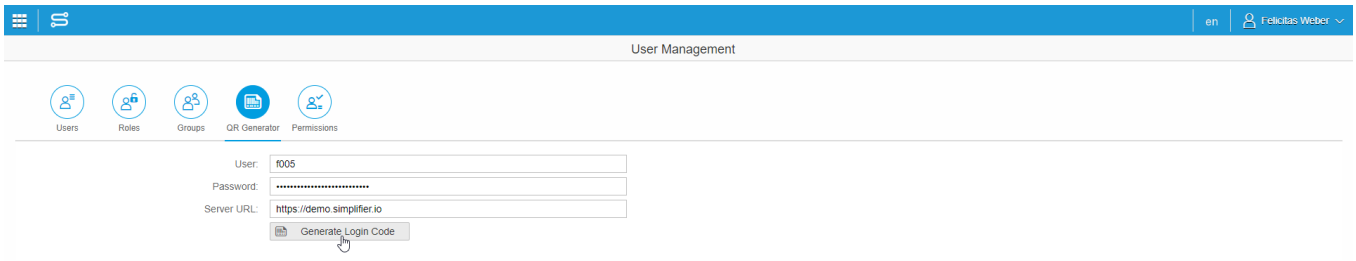
The QR Login-Generator lets you generate your individual QR-Code for login purposes especially for wearable devices without keyboard. You don’t need to type in your credentials anymore, just scan the QR-Code and you are logged in on your instance. Especially when using smart glasses, this feature is comfortable as it takes just a quick scan to enter Server URL, username and password without typing.

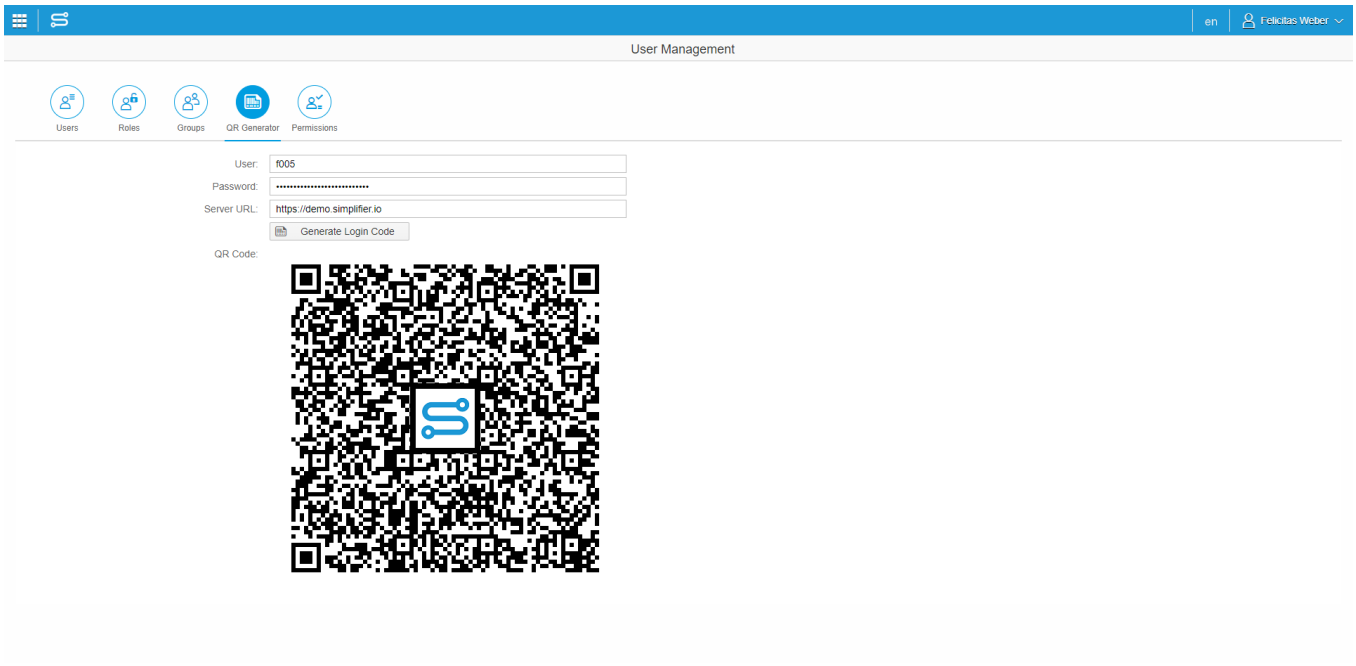
To generate such a QR code, go to the tile **Users** on the Simplifier dashboard and switch then to the tab **QR Generator** in the user overview.


en
Felicitas Weber
Simplifier Dashboard
8
Users
31
Applications
18
Connectors
5
Plugins
25
Transports
9
Business Objects
Logs & Monitoring
137
Data Types
1
Jobs
6
Templates



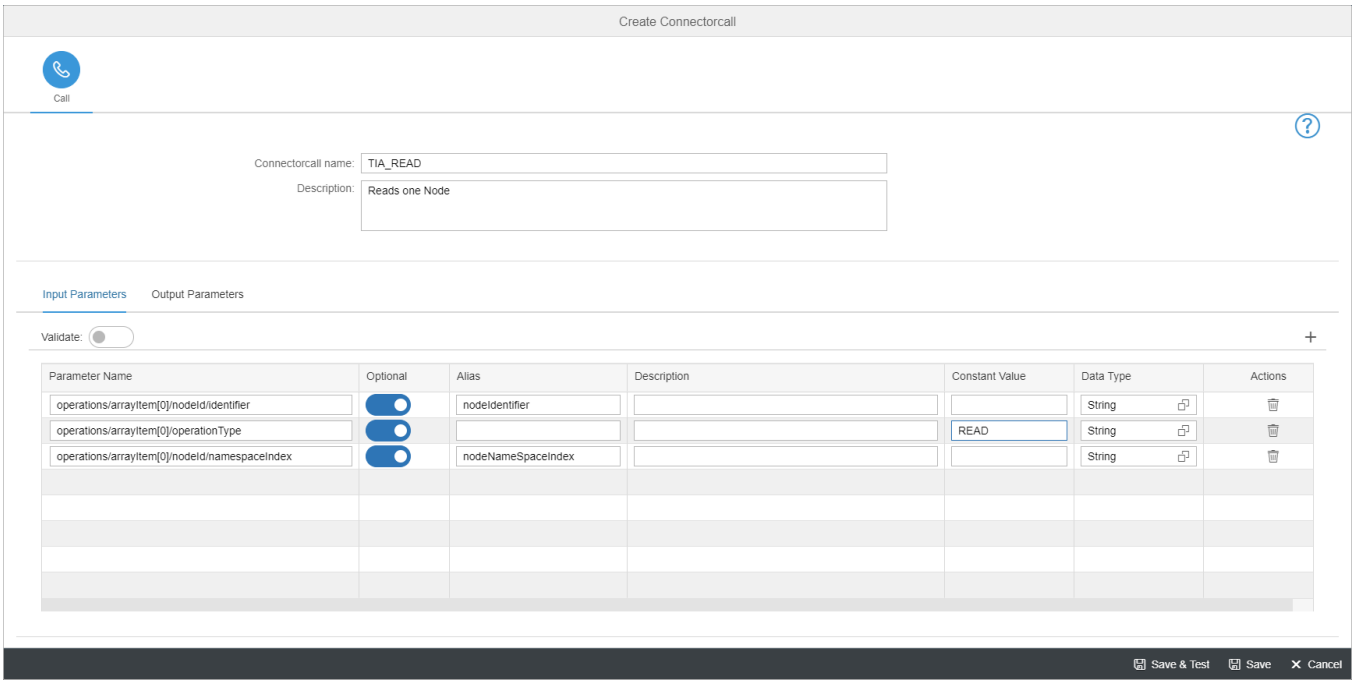
You have to enter the required user with password and the Server URL. By default, this is already filled in. Generate the Login Code and now you can log in to other devices easily and quickly by scanning the QR Code.


en
Felicitas Weber
User Management
Users
Roles
Groups
QR Generator
Permissions
User:
f005
Password:
Server URL:
https://demo.simplifier.io
Generate Login Code


en
Felicitas Weber
User Management
Users
Roles
Groups
QR Generator
Permissions
User:
f005
Password:
Server URL:
https://demo.simplifier.io
Generate Login Code
QR Code:

READ Call - OPC/UA Connector

Call for READ operations (The name TIA_READ is the arbitrary chosen name for this call)



Input Parameter

For the READ Connector Call, you need to configure the “**operationType** ” and the “**nodeId**” (consisting of 2 parameter: **identifier** and **namespaceIndex**). Furthermore you can define the **Identifier Type** (optional) and the **order** of the operations in the code (optional).

operationType: Defines which operation you want to execute, in this case “READ”.
Parameter Name: operations/arrayItem[0]/operationType
Constant Value: READ
Data Type: String

nodeID: Defines the identification of the OPC/UA node. It is split in 2 Parameter:

- **Identifier**:
  Parameter Name: operations/arrayItem[0]/nodeId/identifier
  Data Type: String or Numeric
- **NamespaceIndex**:
  Parameter Name: operations/arrayItem[0]/nodeId/namespaceIndex
  Data Type: String
  In every namespace, each ID must be unique (it is possible to use the String “7617” and the Numeric 7167 together in one namespace)

identifierType (optional): Searches for the Identifier with a fixes Data Type.
Parameter Name: operations/arrayItem[0]/identifierType
Constant Value: String or Numeric

order (optional): Order in which the actions should be performed.

This parameter changes the execution order of the specified READ commands e.g. if we have three READ commands. Command 1 reads from node x, command 2 reads from node y and command 3 from node z, the normal execution order would be: x, y, z.

By specifying the read order with numbers starting from 0 to the last command number subtracted by one (e.g. for three commands it would be 2), the execution will be changed according to the defined number.

For example by adding “order”:[2,1,0] to your code, you switched the operations, so the read commands would be executed the following way. z, x, y.

NOTE: The specific commands are NOT defined here!

Output parameters

You can return all Output Parameter like this:

Parameter Name: /
Data Type: String

If you want to get only selected Output Parameter, use the following syntax:

Parameter Name: operationsResult/[0]/dataType/name (exemplary)
Data Type: depends on the Parameter you want to be returned.

For now, only the complete unformatted JSON will be returned.

Receive message via Process Dashboard

To receive messages via the Process Dashboard, you need to perform the following steps:

1. Drag the Asynchronous Data Object into the drawing area.
2. Select the appropriate connector and the connector call for receiving the message.
3. You can continue with or without script.

Continue with script:

```
var lv_msg = data.data.msg;
```

data.data.msg must eventually be adapted to the output parameter specified in the connector call. lv_msg is the received message (type String, can be e.g. a JSON string, then parse to get object:

```
JSON.parse(data.data.msg)
```

It returns the object:

```
{propertyOne: 'abc', propertyTwo: 5}
```

Continue without script:

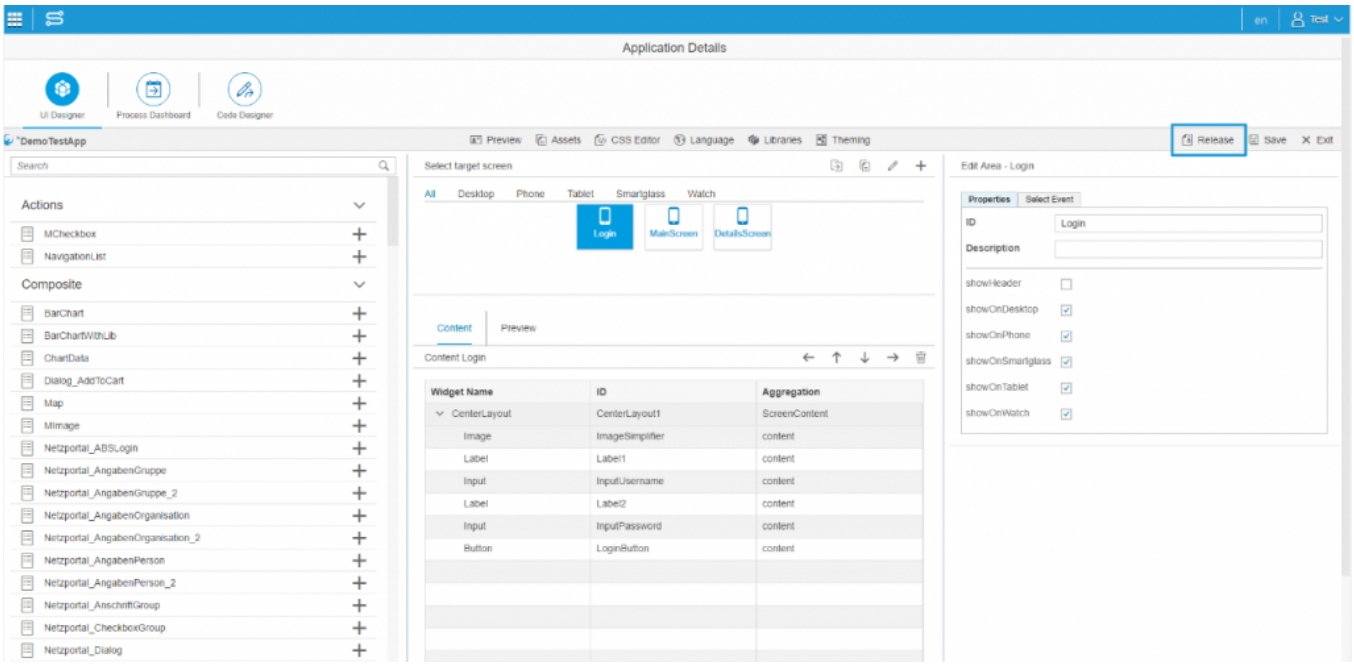
Use the output mapping to map received messages, e. g. to a widget or to save it in a variable.

Release your Application

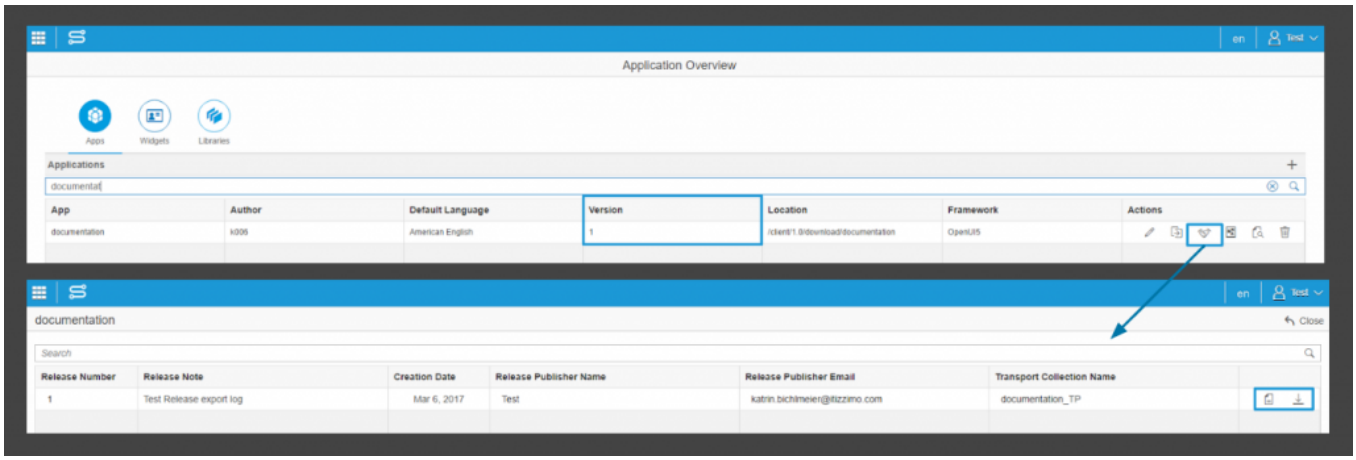
Besides the creation of individual transports, there is a second approach to export your data to another system. By releasing your app after you finished your project or along the way for testing purpose.

Everytime you create a new application, a transport file will be generated automatically with your app's name. It is advisable to update this Transport during your working process (add every feature you use).

For releasing the app, your Transport has to be approved by an Admin in advance. If the approval was given, click on the "Release" button in the UI Designer to export your app. Next, you can add a release note.



If your release was successful, you can see the current version of your app in the Application Overview. Click on the "Show Releases" Icon for an overview of the metadata and a direct download option.



Release Notes:

Release:	documentation
Release Number:	1
Release Note:	Test
Creation Date:	2017-03-28T15:21:46.506+02:00
Release Publisher Name:	Test
Release Publisher Email:	[redacted]
Transport Collection Name:	documentation_TP

Release Contents

Applications

Application: documentation
Description:
Author: k006
Languages:
- Language Code: de_DE
- Language Code: en_US (Default Language, Distribution Language)

Assets:
- Name: MobileAPIDemo.css (Type: text/css)
- Name: Positive.mp3 (Type: audio/mp3)
- Name: sf_logo.png (Type: image/png)

Screens:
Screen: Navigation
Description:
Properties:
- showOnDesktop: true
- showOnPhone: true
- showOnWatch: true
- showOnTablet: true
- showHeader: false
- showOnSmartglass: true

Events:
- onInit: SimpleScript
- onBeforeRendering: SimpleScript
- onAfterRendering: JavaScript

Request Types (Asynchronous Connectors)

This section contains a description of all possible websocket requests for an asynchronous connector (For now only the subscription and unsubscription are documented).

Subscription Request

This request subscribes the websocket to an asynchronous connector and will receive messages from it.

```
var request = null;
request = {
"frameType": "subscribe",
"subscriptionKey":[requestUUID],
"json": [requestData]
};
```

Unsubscription Request

This request unsubscribes the websocket from an asynchronous connector and will not receive messages from it any more. Whether the connector stops working or not depends on the connector implementation.

```
var request = null;
request = {
"frameType": "unsubscribe",
"subscriptionKey": [requestUUID]
};
```

[requestUUID]

A unique String, which acts as the identification key.
An example key might be **“bb827118-f1b0-2170-9937-f8c7e1620107”**.

[requestData]

A Json object, which will be send to the respective connector. Examples can be found here and here.

Required Data Types Examples

In the following sections you will find instructions how to create datatypes, that are required for certain connector calls.

Requirements for Remote Service applications

(On-Premise Installation)

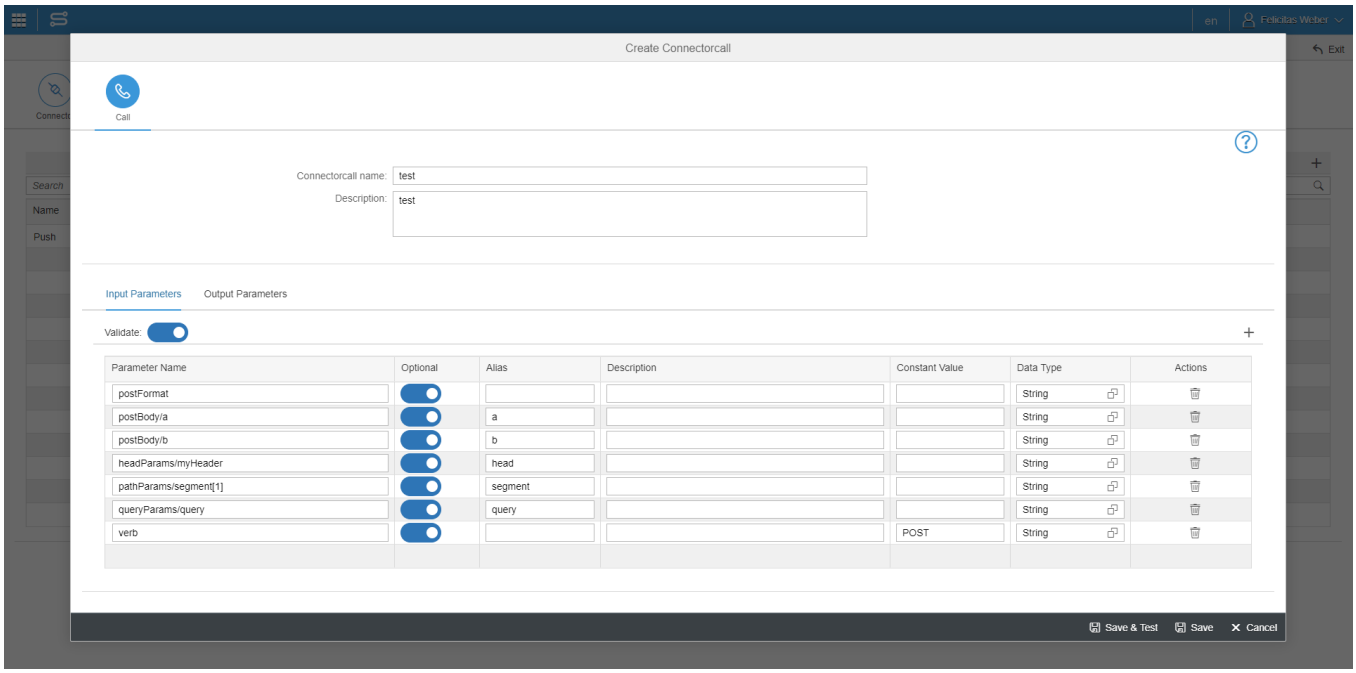
Remote Service applications demand some additional requirements, especially on the network side. To successfully establish a connection between two clients, e.g. the Remote Service Portal, running in the browser on a Desktop machine and the Remote Service Client running on a Smart Glass, they must either reside in the same network, or a STUN-/TURN-server must be used.

The minimum requirements for using a remote service application are:

- Simplifier server and clients must either be in the same physical network segment or be connected via a virtual private network (VPN). If this requirement can’t be fulfilled, a STUN-/TURN server can be used as an intermediate communication partner.
- If you want to use a STUN-/TURN server, the following TCP and UDP ports must be open: 3478, 3479, 5349, 5350.
- Fixed domain for the server and valid certificates for this domain. Remote Service requires these certificates because it runs exclusively over HTTPS for security reasons.
- If you plan to use remote service over a cellular network, be sure that your provider allows these services in your contract. In case of Deutsche Telekom, this means that option “Voice over IP” must be enabled.

REST Connector Calls

This section describes the necessary parameters and data types for REST connector calls.



- ***postFormat***: format and Content-Type of postBody to be send to REST Endpoint. Possible values are :
  - JSON: Json fomat
  - PLAIN: Plain text format
  - FORM: URLencoded from
  - XML: xml format
- ***verb***: REST Verb with possible values :
  - GET
  - POST
  - DELETE
  - PUT
  - OPTION
  - HEAD

- **pathParams (optional)**: key/value object defining ordered list of path segments added to Rest endpoint.
- Exp.: {(**up to version 2.5.56.12:** (“pathParams/segment[1]”:”add”, “pathParams/segment[2]”:”to”,”pathParams/segment[3]”:”path”),**since version 2.5.57.10:** (“pathParams[1]”: “add”, “pathParams[2]”: “to”, “pathParams[3]”: “path”)} will add the path segments “add/to/path” to the endpoint.

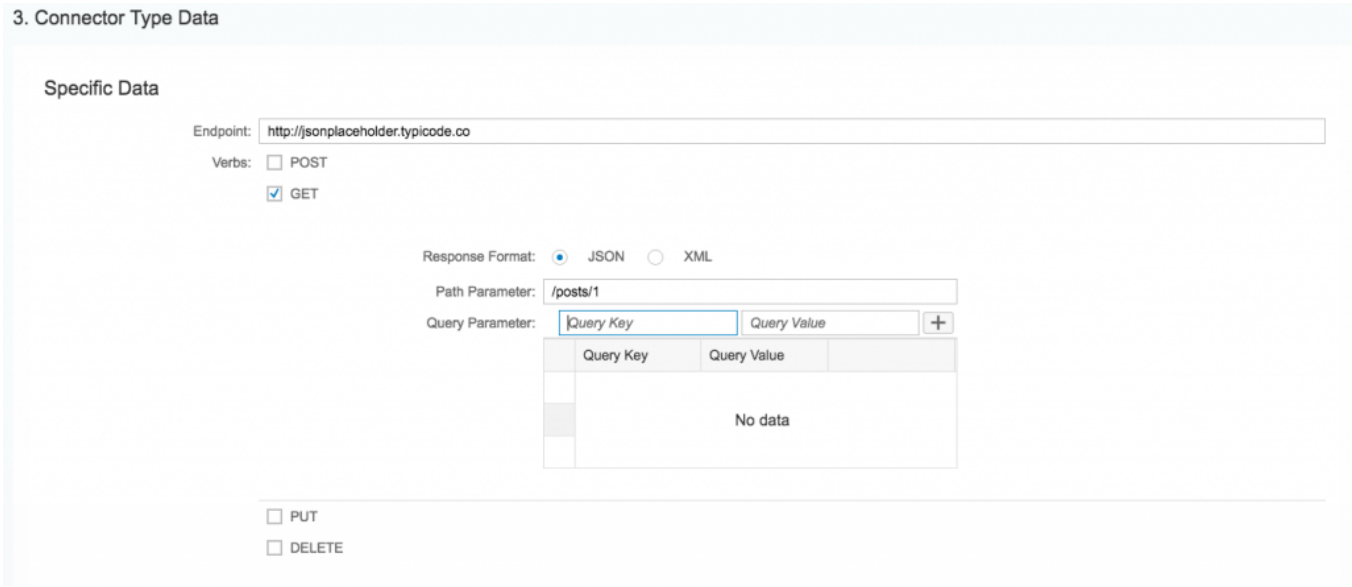
- ***postBody (optional)***: key/value object defining the post body. If postBody PLAIN is selected, a string JSON representation of the key/value list will be send to the REST Endpoint
- ***headParams (optional)***: key/value object defining additional headers of the REST Request. Any head parameters concerning possible authentication will be add, regardless of any earlier definition by headerParams feature.
- ***queryParams (optional)***: key/value object defining query path parameter added to the REST endpoint

Connector execute result

When the connector is executed, it will only return as JSON as a result, if the content-type of the http result is “application/json”. In all other cases, the connector will return a json object with the following keys:

- ***RESTBinaryResult*** – with the BASE64 encoded result body
- ***Content-Type*** – mime type of result body

REST Connector Details



Endpoint

The URL of the Rest Service without specific paths.

Verb

The Operation that is supported by this REST Service. You can set the path and query parameters as well as the response format (XML or JSON) for each operation.

REST Connector Details

Connector endpoint

The URL defines the REST endpoint.

Reverse Proxy Requirements

The Simplifier Server needs a typcial Reverse Proxy as standard setup.

The Reverse Proxy should provide the following services for a secure setup

- SSL Offloading
- Virus / Malware Scanning
- Web Application Firewall

Forwarded Ports:

- 443 (HTTPS)
- 8090 (Websocket)

Port 443

- HTTPS (valid certificate)
- modern, secure TLS configuration (incl. HTTP Strict Transport Security)
- HTTP2 if possible
- Header:
    - “Upgrade”: Pass from client (for WebSockets)
    - “X-Real-IP”: IP address of the client
    - “X-Forwarded-Forwarded-For”: Remote address of the client, or X-Forwarded-For Header of the parent proxy server.
    - “X-Forwarded-Proto”: Original protocol of the request (“http” or “https”)
    - CORS Headers (see below)
    - (Temporary) Redirect from “/” to either AdminUI (“/UserInterface/”) or App of choice (“/appDirect/$appName”)
    - Proxy connection/read/send Timeout to high value, e. g. 10 min
    - Maximum body size (post, put) to appropriate value, e. g. 20 MB (doesn’t have to be too big, because packets > 20 MB are transferred as single chunks)
    - Proxy forwarding to Simplifier AppServer port 8080 (if on another server, must be reachable via firewall)

Proxy must be able to pass through WebSocket connections!

Port 8090

- HTTPS (valid certificate)
- modern, secure TLS configuration (see above)
- http2 if possible
- proxy connection/read/send Timeout to high value, e. g. 10 min
- proxy forwarding to Simplifier AppServer port 8090 (if on another server, must be reachable via firewall)

Proxy must be able to pass through WebSocket connections!

CORS headers

Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to allow a user agent to access selected resources from a server located on a different origin (domain) than the currently used Web site. A user agent makes a cross-origin HTTP request when requesting a resource from another domain, protocol, or port than the one from which the current document originates.

The CORS mechanism supports secure cross-domain queries and data transfers between browsers and web servers. Modern browsers use CORS in an API container such as XMLHttpRequest or Fetch to minimize the risks of cross-origin HTTP requests.

For request methods' POST',' GET',' PUT',' DELETE':

Header name
Access-Control-Allow-Origin
Access-Control-Allow-Credentials
Access-Control-Allow-Methods
Access-Control-Allow-Headers

Additionally for request method 'OPTIONS':

Header name
Access-Control-Max-Age
Response name
Empty Content

The following paths should be configured for routing it back to the Simplifier

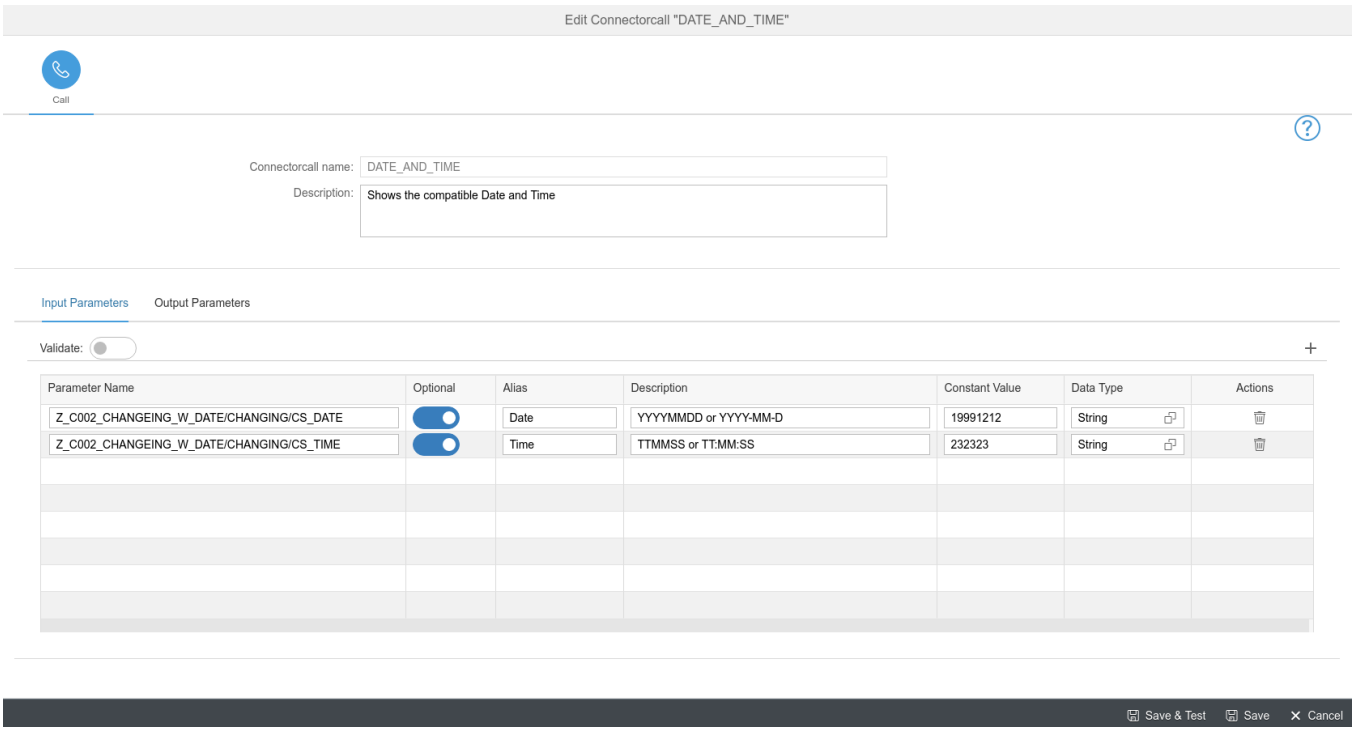
Location / Path	Description
"^/genToken/$"	The Simplifier Authentification Service based on Tokens
"^/assets/(.*)$"	The static assets like images , pdf files etc for an Application
"^/client/(.*)$"	The Client REST API to access business objects, connector or plugins
"^/library-managed/(.*)$"	Third Party Javascript Libraries that need for the HTML5 Applications
"^/library-static/(.*)$"	Third Party Javascript Libraries that need for the HTML5 Applications
"^/appDirect/(.*)$"	Hosting Path for the created HTML5 Applications
"^/UserInterface/(.*)$"	Admin Backend Interface Application (should only accessible in a secure environment, internal network)
"^/authentication/(.*)$"	External Authentication Provider for e.g. oAuth
"^/passwordExpired/(.*)$"	Password Reset Page for Admin Interface
"^/marketplace/(.*)$"	Simplifier Marketplace
"^/develop/(.*)$"	Plugin Interface

RFC Connector Call - EXECUTE

To use the RFC Connector Call with the operation EXECUTE, you have to configure the following input parameter in the Simplifier:

- [FUNCTION_MODULE_NAME]/[PARAMETER_TYPE]/[PARAMETER_NAME]
- [X]/[FUNCTION_MODULE_NAME]/[PARAMETER_TYPE]/[PARAMETER_NAME] (if you want to execute batch calls)

The second one is optional.



operation

In this case: EXECUTE

operationTarget

Defines which parameters you pass – you can transfer imports, changings and tables and then fire the function.

- configuration/operation/operationTarget

- [X]/configuration/operation/operationTarget

The value is a list of one or more values, but only one of these values exists once:

- CHANGING
- IMPORT
- TABLE

returnInformation

Defines the return parameter.

- [X]/configuration/operation/returnInformation

The value is a list of the following values, but only one of them can be used at the same time.

- INPUT
- OUTPUT

---

additionalReturnSetInformation

Returns additional changing parameter to see if the tables have changed somehow when executing the function block (point of time is depending on the returnSet).

- [X]/configuration/operation/additionalReturnInformation

The value is a list of the following values, but only one of them can be used at the same time.

- IMPORT
- TABLE
- EXCEPTION
- EXPORT
- CHANGING

If you’ve set INPUT as returnInformation, the changing parameter before the function has been executed will be displayed.
If you’ve set OUTPUT as returnInformation, the changing parameter after the function has been executed will be displayed.

RFC Connector Call - GET

To use the RFC Connector Call with the operation GET, you have to configure the following input parameter in the Simplifier:

- [X]/Z_C002_CHANGEING_W_DATE (with any constant value)
- Z_C002_CHANGEING_W_DATE (with any constant value)

- [X]/configuration/operation/operationType GET

- configuration/operation/operationType GET

---

operation

In this case: GET

---

operationTarget

Defines the exact information you want to return from the function block. Choose between:

- [X]/configuration/operation/operationTarget (list that is passed)

With the following values (but the value may only appear once in the list, e.g. [import, template];[export])

- IMPORT (returns all parameter, that are passed to the function)
- EXPORT (returns all export parameter)
- CHANGING (returns the changing parameter that is used by the function)
- TABLE (returns table structures, that are stored in the function)
- EXCEPTION (returns exceptions, that are defined within the function)
- TEMPLATE (returns all)

---

returnInformation

Lists the complete structure (with SAP metadata) of the function block. Choose between:

- [X]/configuration/operation/returnInformation

- configuration/operation/returnInformation

Enter the value as a list. [STRUCTURE, METADATA]

Structure is the default.

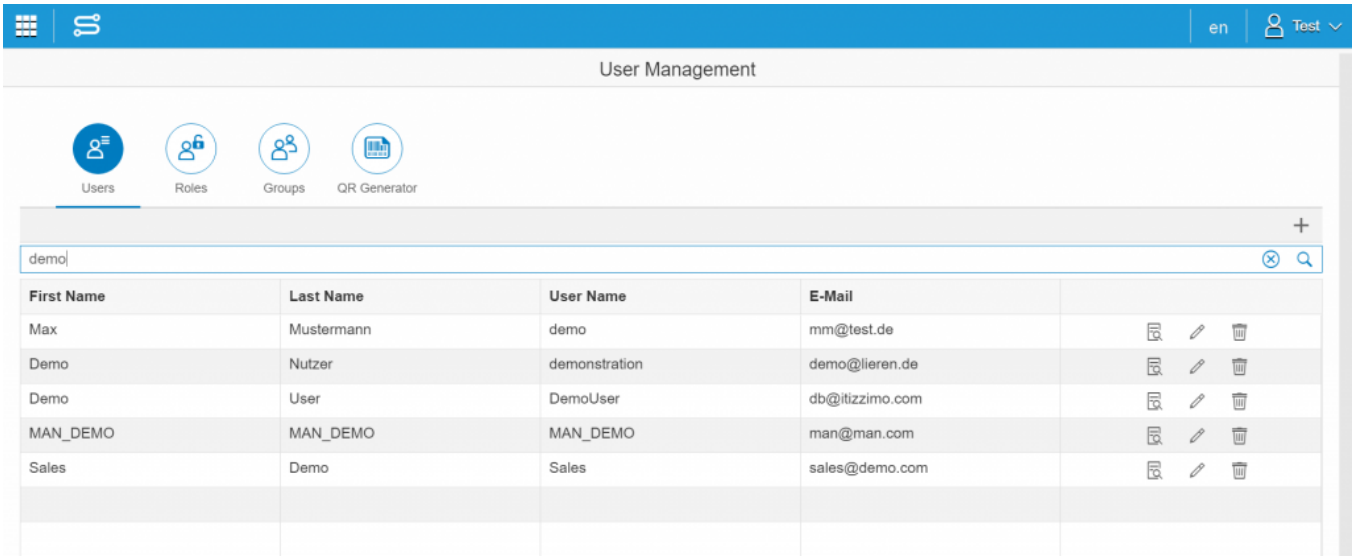
RFC Connector Calls

This section describes the necessary parameters and data types for RFC connector calls. There are two different kind of calls

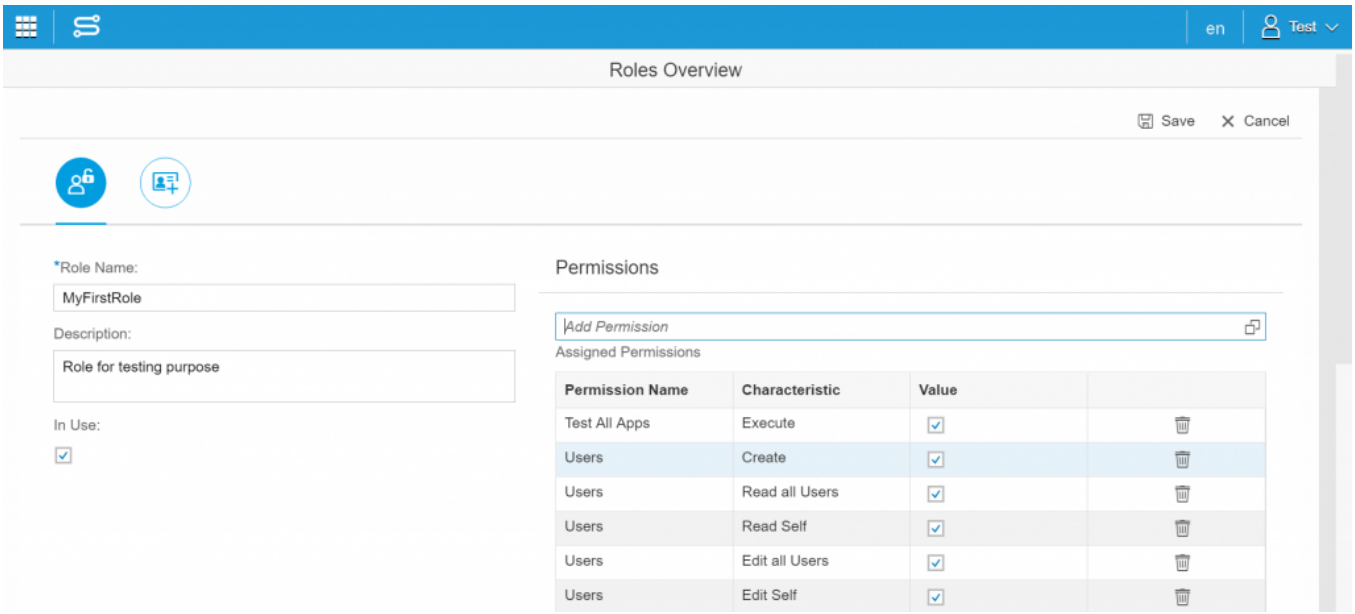
- Calls for GET operations
- Calls for EXECUTE operations

Role Overview

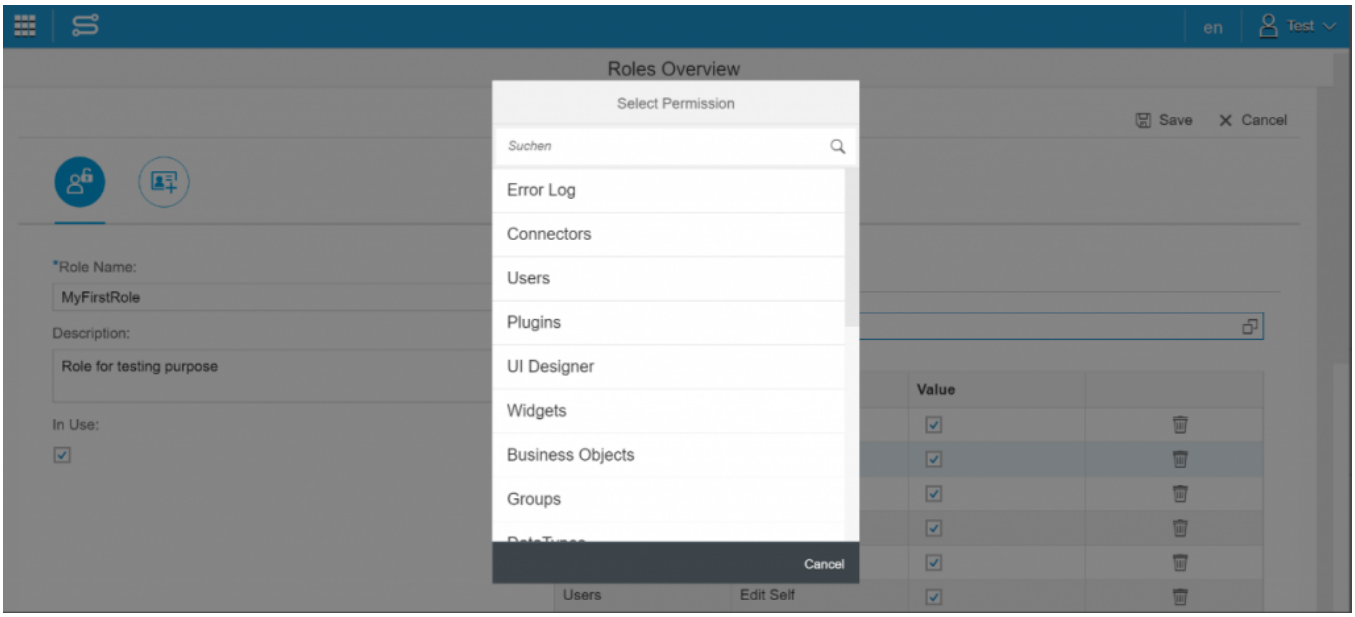
The section Role Overview defines permissions for your users. Here you can configure roles that you can then apply to your users in order to permit or restrict certain actions, like administering applications and users or allow the use of special applications and restrict the use of others. Usually, there is one administrator-role that is allowed to use all features and functions of the Simplifier.



To create a new role , click on the plus icon to enter the create dialog. To add permission, use the selection helper on the right and choose the needed permission category. To enable or disable a single permission, click on the checkbox in the corresponding column. Finally, assign the role to existing users. Hit the save button to finish the role creation or the update.

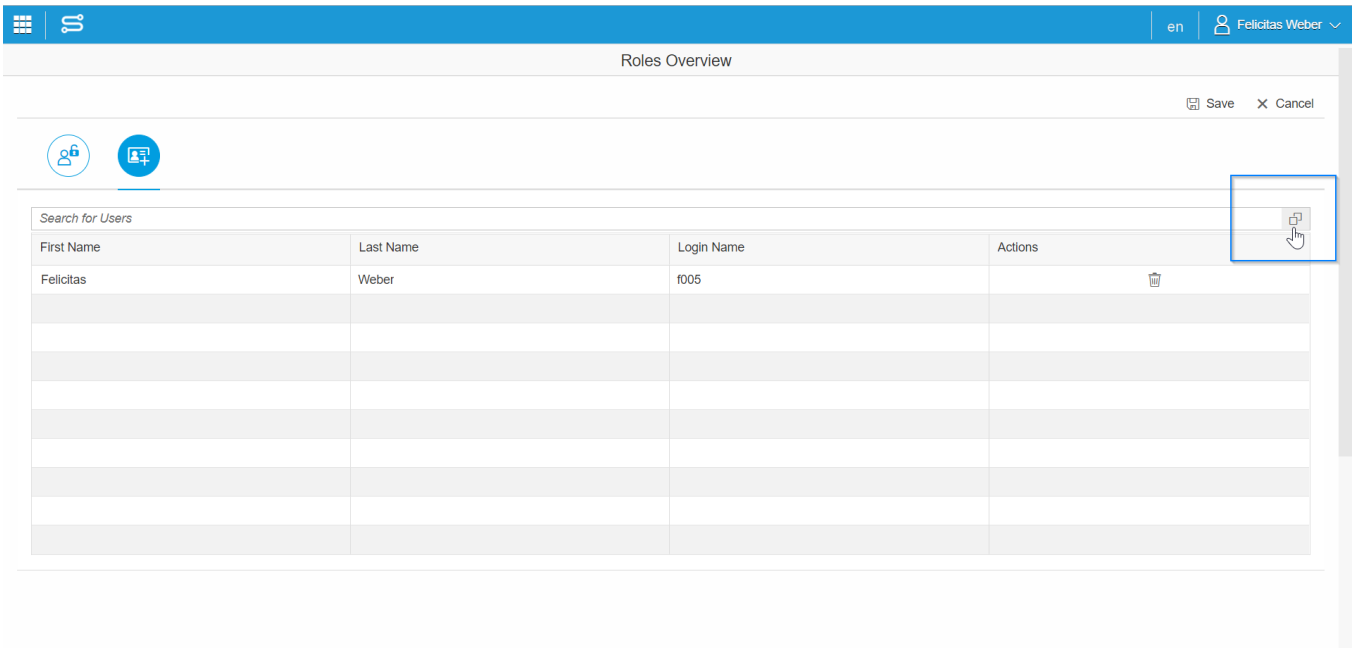


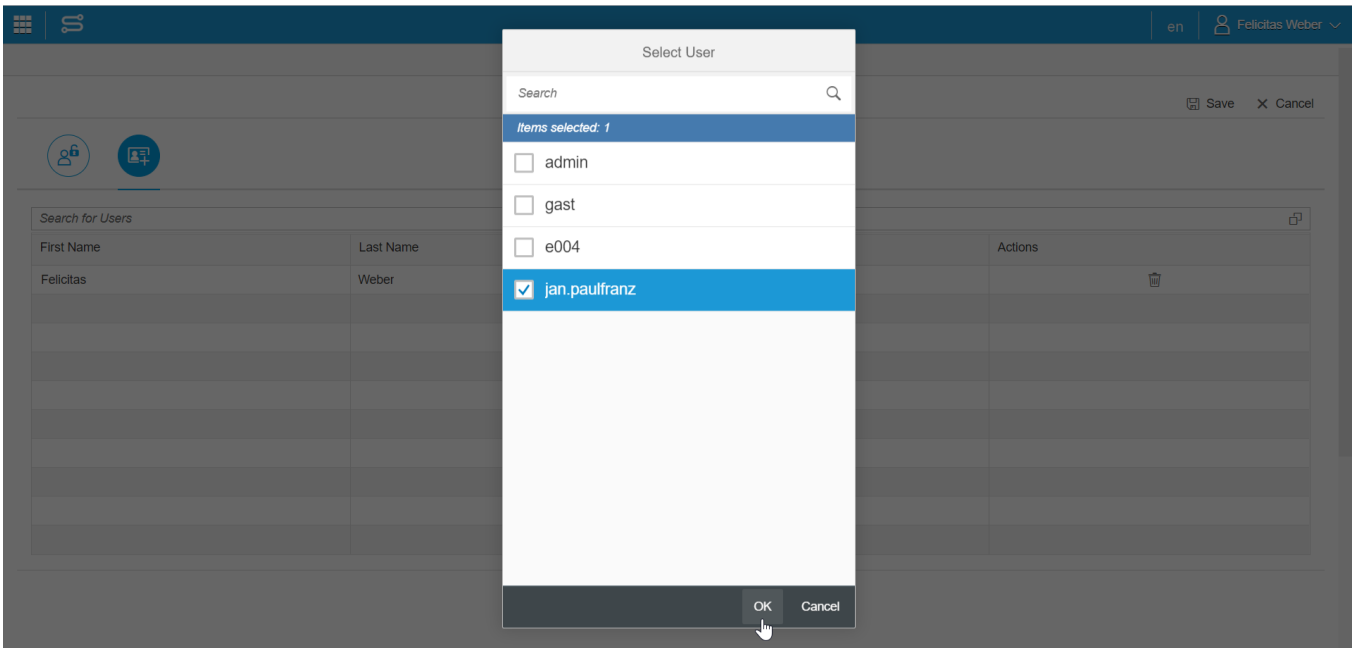
Create a new role

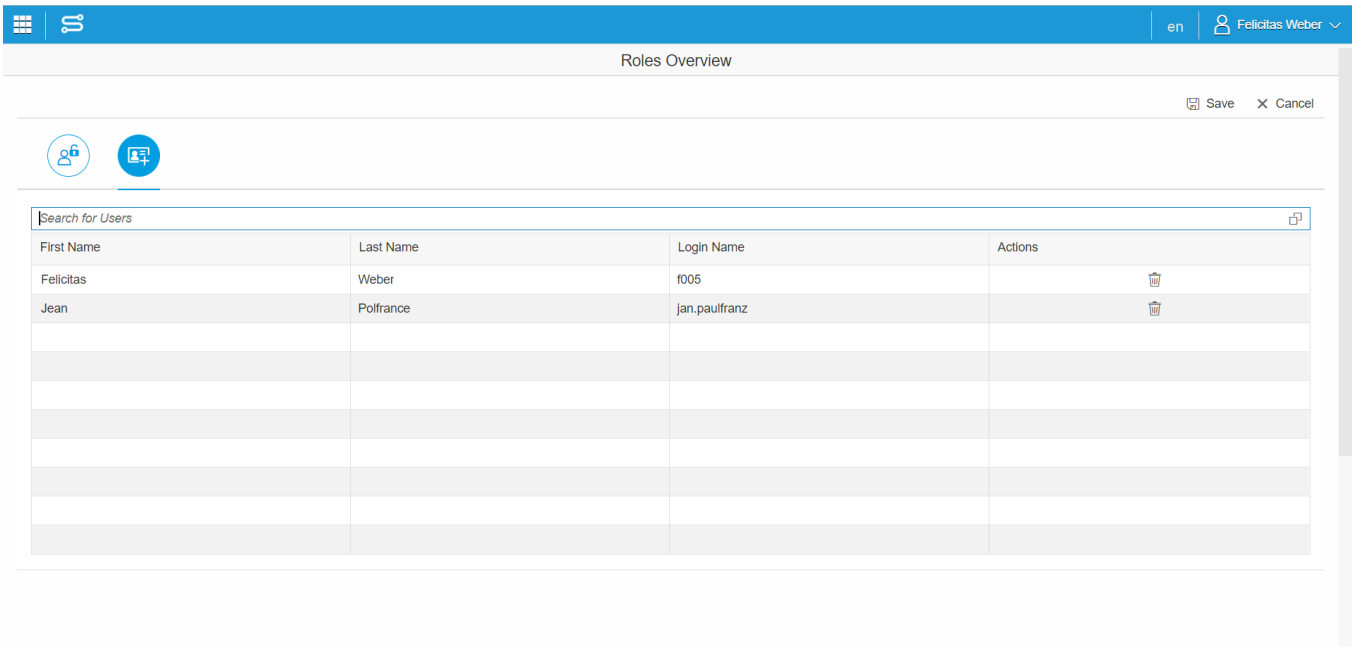


Add a permission

Within the right tab you can add a user to the role. Use the Selector for this.


en
Felicitas Weber
Roles Overview
Save
Cancel
Search for Users
First Name
Last Name
Login Name
Actions
Felicitas
Weber
f005


en
Felicitas Weber
Save
Cancel
Search for Users
First Name
Last Name
Actions
Felicitas
Weber
Select User
Search
Items selected: 1
admin
gast
e004
jan.paulfranz
OK
Cancel


en
Felicitas Weber
Roles Overview
Save
Cancel
Search for Users
First Name
Last Name
Login Name
Actions
Felicitas
Weber
f005
Jean
Polfrance
jan.paulfranz

Run Simplifier Docker locally

Short Instructions

Create the directory which will host all external user-specific data

For Linux and Mac
$ mkdir -p /home/simplifier/data
$ export SIMPLIFIER_DIR=”/home/simplifier/data”

For Windows, please replace “SIMPLIFIER_DIR” with an absulute path.
Example: “C:\Docker\Simplifer\Data”

Install SSL certificates:
$ mkdir -p SIMPLIFIER_DIR/certs
$ cp <certificate.pem> SIMPLIFIER_DIR/certs/default.crt
$ cp <keyfile.pem> SIMPLIFIER_DIR/certs/default.key

Run docker:
Alternative 1: with SSL/Certificates
$ docker run -d -v $SIMPLIFIER_DIR:/opt/simplifier/data \
-p 80:80 -p 443:443 -p 8090:8090 \
—name=simplifier itizzimo/simplifier

Alternative 2: without SSL/Certificates
$ docker run -d -v $SIMPLIFIER_DIR:/opt/simplifier/data \
-p 80:8080 -p 8090:8091 \
—name=simplifier itizzimo/simplifier

Browse the Simplifier:

Now you can open the Simplifier in your browser.
http(s)://localhost/UserInterface

To avoid certificate errors, you can create a local host entry for your domain stored in the certificate for the HTTPS connection.

e.g. https://simplifier.<yourcompany>/UserInterface

_____

SAP

With the Simplifier we offer several connectors/options to interface with SAP:

1. SAP RFC Connector
2. SOAP Interface (based on generic SOAP Interface in the iTiZZiMO SAP Namespace ITIZ)
3. oData / REST Connector (XI / PI / Hana, etc.)

In the following sub-sites you can see the BAPIs we offer, that can be accessed through above mentioned Interfaces within our own SAP Namespace ITIZ.

SAP ERP6 Change Document

SAP ERP6 (Enterprise-Resource-Planning) Change Document

Method	SAP Transaction	Description
READ	SE38 – RSSCD200	Load a change document usign the using the object key.
SEARCH_BY_DATA	SE38 – RSSCD200	Search for a change document
SEARCH_BY_DATE	SE38 – RSSCD200	Search for a change document via date range.

SAP ERP6 Business Partner

SAP ERP6 (Enterprise-Resource-Planning) Business Partner

Method	SAP Transaction	Description
CREATE	BP	Create Business Partner master data.
READ	BP	Load a Business Partner.
UPDATE	BP	Update Business Partner master data.
DELETE	BP	Delete Business Partner.
CREATE_BANK_DATA	BP	Create bank data.
CREATE_TAX_DATA	BP	Create tax data.
CREATE_INDUSTRY_SECTOR	BP	Create industry sector.

SAP ERP6 Document

SAP ERP6 (Enterprise-Resource-Planning) Document

Method	SAP Transaction	Description
CREATE	CV01N	Create document object master data.
READ	CV03N	Load a document using the document object key.
UPDATE	CV02N	Update document object master data.
DELETE	CV02N	Mark document for deletion.
CREATE_ORIGINAL_FILE	CV02N	Create a file from raw content, including original.
READ_ORIGINAL_FILE	CV03N	Read a file's contents with document number and file ID.
UPDATE_ORIGINAL_FILE	CV02N	Update file details.
DELETE_ORIGINAL_FILE	CV02N	Delete file from document, including original.

SAP ERP6 Material

SAP ERP6 (Enterprise-Resource-Planning) Material

Method	SAP Transaction	Description
CREATE	MM01	Create material master data.
READ	MM03	Load material using the material number.
UPDATE	MM02	Update material master data.
DELETE	MM02	Mark material for deletion.
READ_ADDITIONAL_DATA	mm03	Read additional material master data.
READ_MATERIAL_AVAILABILITY		ATP information.

SAP ERP6 Object Status

SAP ERP6 (Enterprise-Resource-Planning) Object Status

Method	SAP Transaction	Description
READ	Various (e.g. QM03)	Read an object status based on the object numbers.
READ_HISTORY		Read system / user status change documents.
SET_STATUS_OF_OBJ		Set system / user status of an object.
RESET_STATUS_OF_OBJ		Removing the system / user status with no reference number to object X.

SAP ERP6 Status Profile

SAP ERP6 (Enterprise-Resource-Planning) Status Profile

Method	SAP Transaction	Description
READ	BS03	Load an status profile based on the object numbers.
SEARCH	BS03	Search a status scheme.

SAP ERP6 User

SAP ERP6 (Enterprise-Resource-Planning) User

Method	SAP Transaction	Description
READ	SU01D	Read a SAP User.

SAP HR Personal Time Management

SAP HR Personal Time Management

Method	SAP Transaction	Description
CREATE	PA61	Load contemporary document master data using the contemporary document object key.
READ	PA61	Read contemporary document object master data.
UPDATE	PA61	Updatecontemporary document object master data.
DELETE	PA61	Delete a contemporary document.
READ_TIME_STATEMENT	PT40	Read remaining leave.

SAP ISU Business Partner

SAP ISU (Industry Solution for Utilities) Business Partner

Method	SAP Transaction	Description
CREATE	BP	Create ISU Business Partner master data.
READ	BP	Load a ISU Business Partner.
UPDATE	BP	Update ISU Business Partner master data.
DELETE	BP	Mark a ISU Business Partner for deletion.
SEARCH	BP	Search for a ISU Business Partner.
CREATE_FLAT	BP	Simplified creation of ISU Business Partner.

SAP ISU Connection Object

SAP ISU (Industry Solution for Utilities) Connection Object

Method	SAP Transaction	Description
CREATE	ES55	Create connection object master data.
READ	EBAA	Loading a connection object based on the connection object number.
UPDATE	ES64	Update connection object master data.
DELETE	N/A	Deletion of a connection object is not possible.
SEARCH	EBAA	Search for a connection object.
READ_HIERARCHY	EBAA	Read the hierarchy of a connection object.

SAP ISU Device Location

SAP ISU (Industry Solution for Utilities) Device Location

Method	SAP Transaction	Description
CREATE	ES65	Create device location master data.
READ	ES67	Load a device location using the device location object number.
INSTALL	IE02	Installation of a device location.
DISMANTLE	IE02	Dismantling of a device location.

SAP ISU Premise

SAP ISU (Industry Solution for Utilities) Premise

Method	SAP Transaction	Description
CREATE	ES60	Create premise master data.
READ	ES62	Load a premise using the premise object number.
UPDATE	ES61	Update premise master data.
DELETE	ES61	Mark a premise for deletion.
SEARCH	ES62	Search for a premise.

SAP ISU Utility Installation

SAP ISU (Industry Solution for Utilities) Utility Installation

Method	SAP Transaction	Description
CREATE	ES30	Create Utility Installation object master data.
READ	ES32	Load a Utility Installation based on the Utility Installation object number.

SAP ITIZ DB Table

SAP ITIZ DB (Database) Table

Method	SAP Transaction	Description
READ	SE16n	Read a database table.
READ_FIELDS	SE16n	Read the available fields to a database table.

SAP ITIZ Key Value

SAP ITIZ Key Value

Method	SAP Transaction	Description
READ	SE16n	Load data from the SAP database, which can be used as a key value.
READ_DATA_ELEM	SE11	Read ABAP domain fixed values ??of data element X.

SAP MM Entry Sheet

SAP MM (Material Management) Entry Sheet

Method	SAP Transaction	Description
CREATE	ML81	Create a MM Entry Sheet.
READ	ML82	Load a MM Entry Sheet using the Entry Sheet number.
DELETE	ML81	Mark a MM Entry Sheet for deletion.

SAP MM Goods Movement

SAP PM (Material Management) Goods Movement

Method	SAP Transaction	Description
CREATE	MIGO	Create a goods movement.
READ	MIGO	Load a goods movement using the object key.
CANCEL	MBST	Cancel a goods movement.

SAP MM Purchase Order

SAP MM (Material Management) Purchase Order

Method	SAP Transaction	Description
CREATE	ME21N	Create Purchase Order master data.
READ	ME23N	Load a Purchase Order using the Purchase Order object number.
UPDATE	ME22N	Update Purchase Order master data.
DELETE	ME22N	Mark a Purchase Order for deletion.
SEARCH	ME23N	Search for purchase orders by vendor ID / purchase organization / factory / group / order type / status.

SAP MM Service Master

SAP MM (Material Management) Service Master

Method	SAP Transaction	Description
CREATE	AC01	Create MM Service Master.
READ	AC03	Load a MM Service Master based on the Service Master number.
UPDATE	AC02	Update MM Service Master.
DELETE	AC02	Mark MM Service Master for deletion.

SAP PM Equipment

SAP PM (Plant Maintenance) Equipment

Method	SAP Transaction	Description
CREATE	IE01	Create equipment master data.
READ	IE03	Load an equipment using the equipment number.
UPDATE	IE02	Update equipment master data.
DELETE	IE02	Mark an equipment for deletion.
SEARCH	IE03	Search for an equipment.
READ_CLASSES	IE03	Read the class characteristics of an quipment.
READ_STATUS	IE03	Read the user and system status of an equipment.
CHANGE_CLASSES	IE02	Update the equipment class charactersitics.
CREATE_BY_REFERENCE	IE01	Create an equipment according to a model X.
LOAD_BY_BAPI_GETLIST		Search for the point of use via BAPI_EQUI_GETLIST.
INSTALL	IE02	Installation of an equuipment.
DISMANTLE	IE02	Dismantle an equipment.

SAP PM Functional Location

SAP PM (Plant Maintenance) Functional Location

Method	SAP Transaction	Description
CREATE	IL01	Create functional location master data.
READ	IL03	Load a functional location.
UPDATE	IL02	Update a functional location.
DELETE	IL02	Delete a functional location.
READ_HIER	IL03	Read the hierarchy of a functional location.
CONV_EXT_2_INT	SE37	Converts the primary key from the external format to the internal format.
READ_CLASS	IL02	Read class characteristics of a functional location.

SAP PM Maintenance Notification

SAP PM (Plant Maintenance) Maintenance Notification

Method	SAP Transaction	Description
CREATE	IQS1	Create a maintenance notification.
READ	IQS3	Load a maintenance notification.
UPDATE	IQS2	Update a maintenance notification.
DELETE	IQS2	Delete a maintenance notification.
SEARCH	IQS3	Search for maintenance notification.

SAP PM Maintenance Order Confirmation

SAP PM (Plant Maintenance) Maintenance Order Confirmation

Method	SAP Transaction	Description
CREATE	IW41	Create a maintenance order confirmation.
READ	IW43	Load a maintenance order confirmation using the object key.

SAP PM Service Notification

SAP PM (Plant Maintenance) Service Notification

Method	SAP Transaction	Description
CREATE	IW54	Create a service notification using the object key.
READ	IW53	Load a service notification using the object key.
UPDATE	IW52	Update a service notification.
DELETE	IW54	Delete a service notification.
ADD_DATA	IQS2	Add data to the service notification.
SEARCH_NOTF_LIST	IW53	Search for a service notification with the notification list.

SAP PM Service Order

SAP PM (Plant Maintenance) Service Order

Method	SAP Transaction	Description
CREATE	IW31	Create a service order using the object key.
READ	IW33	Load a service order using the object key.
UPDATE	IW32	Update a service order.

SAP PP Production Order

SAP PP Production Order

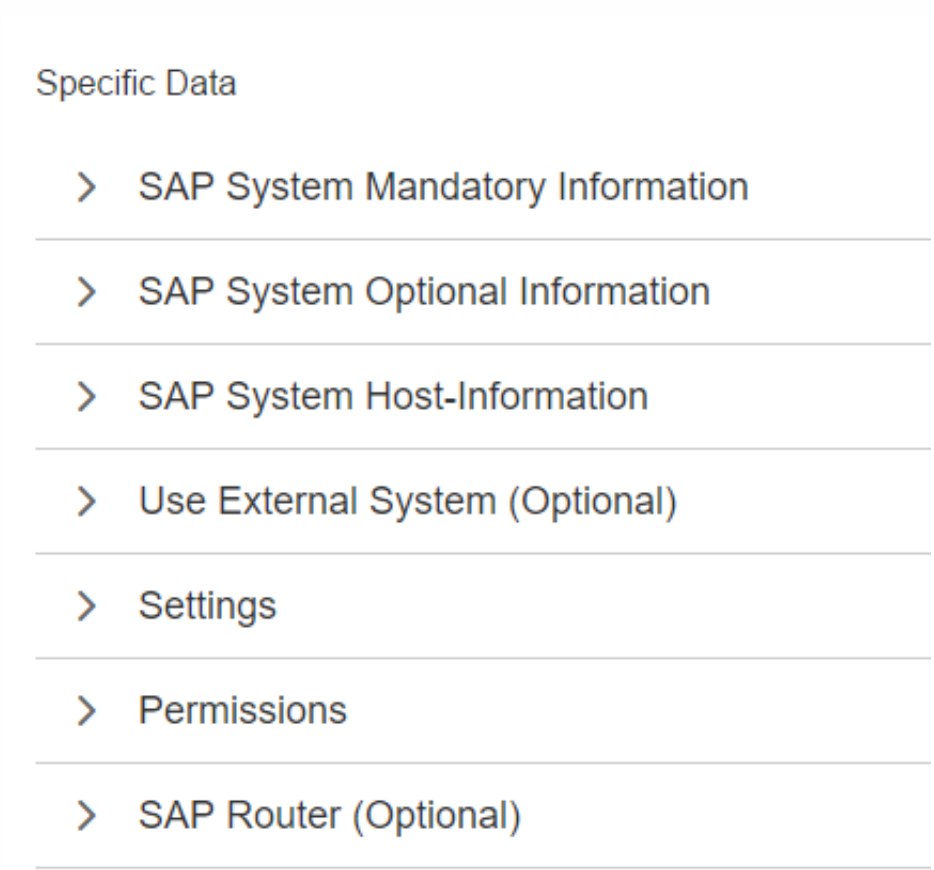
Method	SAP Transaction	Description
READ	CO03	Loading an MM service master using the primary key.
CREATE	CO01	Create MM service master.
READ_DOCUMENTS	CO03	Read the document overview of a production order.
RELEASE	CO02	Release of a production order.

SAP QM Quality Notification

SAP QM Quality Notification

Method	SAP Transaction	Description
READ	QM03	Read a quality notification with a primary key.

SAP RFC Connector Details



SAP System Mandatory Information

The following parameters are mandatory:

Parameter	Description	Example	Mandatory
System ID	Unique ID of SAP System	ID4	Yes
Client Number	Number of SAP Client	100	Yes
System Number	SAP System Instance Numeber	80	Yes
Language	Default Language for SAP Logon	de	Yes

Mandatory Information

SAP System Optional Information

The following parameters are optional:

Parameter	Description	Example	Mandatory
Username Alias	Alias of Username	RFCTEST	No
SAP System Group	Groupname of SAP Systems	GRP	No
R3 System Number	Number of R3 System	80	No
SAP Client Type	Not defined (Default) R2 System R3 System External System	Not defined	No

Optional Information

SAP System Host-Information

Please fill out either Application Server or Gateway Server Hostname to define the endpoint to the SAP Backendsystem

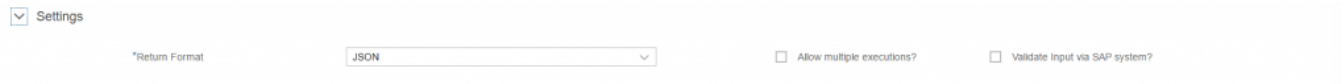
Parameter	Description	Example	Mandatory
Application Server Hostname	IP Address of SAP System	10.110.25.7	No
Gateway Server Hostname	IP Address or Hostname of Gateway Server	10.100.26.0	No
Message Server Hostname	Message Server IPv4 address	10.110.29.0	No

	or hostname		
Gateway Host Hostname	IP Address or Hostname of the Gateway Host Server	10.110.29.9	No

Use External System (Optional)

Use External System (optional)

Settings



Settings

Permissions

Permissions

SAP Router (Optional)

SAP Router (optional)

SAP SD Billing Document

SAP SD (Sales & Distribution) Billing Document

Method	SAP Transaction	Description
CREATE	VF01	Create a Billing Document.
READ	VF03	Load a Billing Document based on the Billing Document Number.
DELETE	VF02	Mark a Billing Document for deletion.

SAP SD Customer

SAP SD (Sales & Distribution) Customer

Method	SAP Transaction	Description
CREATE	XD01	Create customer object master data.
READ	XD03	Load a customer object based on the customer object number.
UPDATE	XD02	Update of customer object master data.
DELETE	XD06	Mark a customer for deletion.
SEARCH	XD03	Search for customer.
CREATE_BY_REFERENCE	XD01	Create a customer modelled on customer X.
SEARCH_BY_ADRC	XD03	Search for a customer based on the address.

SAP SD Customer Quotation

SAP SD (Sales & Distribution) Customer Quotation

Method	SAP Transaction	Description
CREATE	VA21	Create Quotation object master data.
READ	VA23	Loading a Quotation object using the offer object number.
UPDATE	VA22	Update Quotation object master data.
DELETE	VA22	Mark the Quotation for deletion.
DELETE_POSITION_BY_NO	VA22	Delete a Quotation item.
CREATE_POSITION_BY_NO	VA22	Create a Quotation item.
REJECT	VA22	Select a reason for rejecting the Quotation.
REJECT_BY_POS	VA22	Select a reason for rejecting the Quotation by position.

SAP SD Sales Order

SAP SD (Sales & Distribution) Sales Order

Method	SAP Transaction	Description
READ	VA03	Loading a sales order object based on the customer order object number.
CREATE	VA01	Create sales order object master data.
UPDATE	VA02	Update sales order object master data.
DELETE	VA02	Mark a sales order for deletion.
SEARCH	VA03	Search a sales order.
READ_ORD_STATUS	VA03	Read sales order status.
CREATE_BY_CUSTOMER_QUOTATION	VA01	Create a sales order with reference to the customer quotation.
REJECT	VA02	Reject a sales order.
REJECT_POS_NO	VA02	Reject sales order individually.

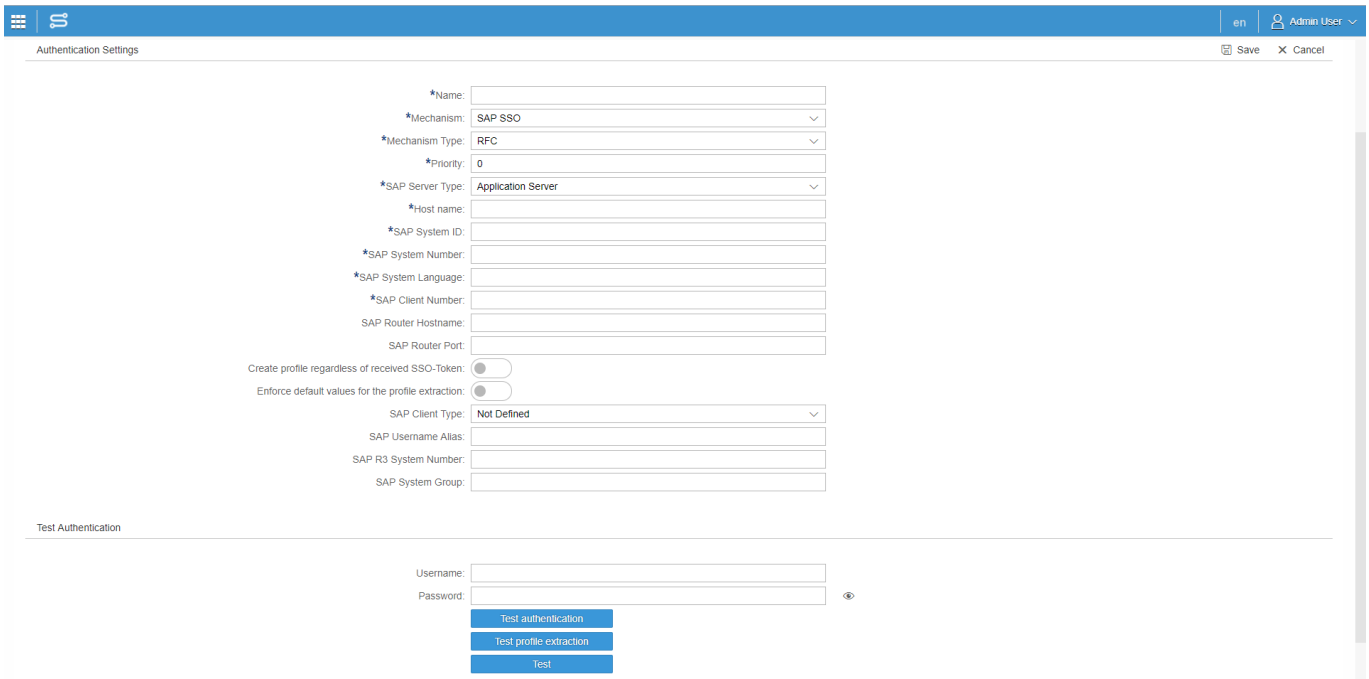
SAP SD Vendor

SAP SD (Sales & Distribution) Vendor

Method	SAP Transaction	Description
CREATE	XK01	Create a SD Vendor.
READ	XK03	Load a SD Vendor.
UPDATE	XK02	Update SD Vendor.
DELETE	XK02	Mark a SD Vendor for deletion.

SAP Single Sign On via RFC

Furthermore you can use the mechanism type RFC. It works on the same principle as SOAP. You can log on to the Simplifier via SAP and start connector calls with the SSO2 Token.



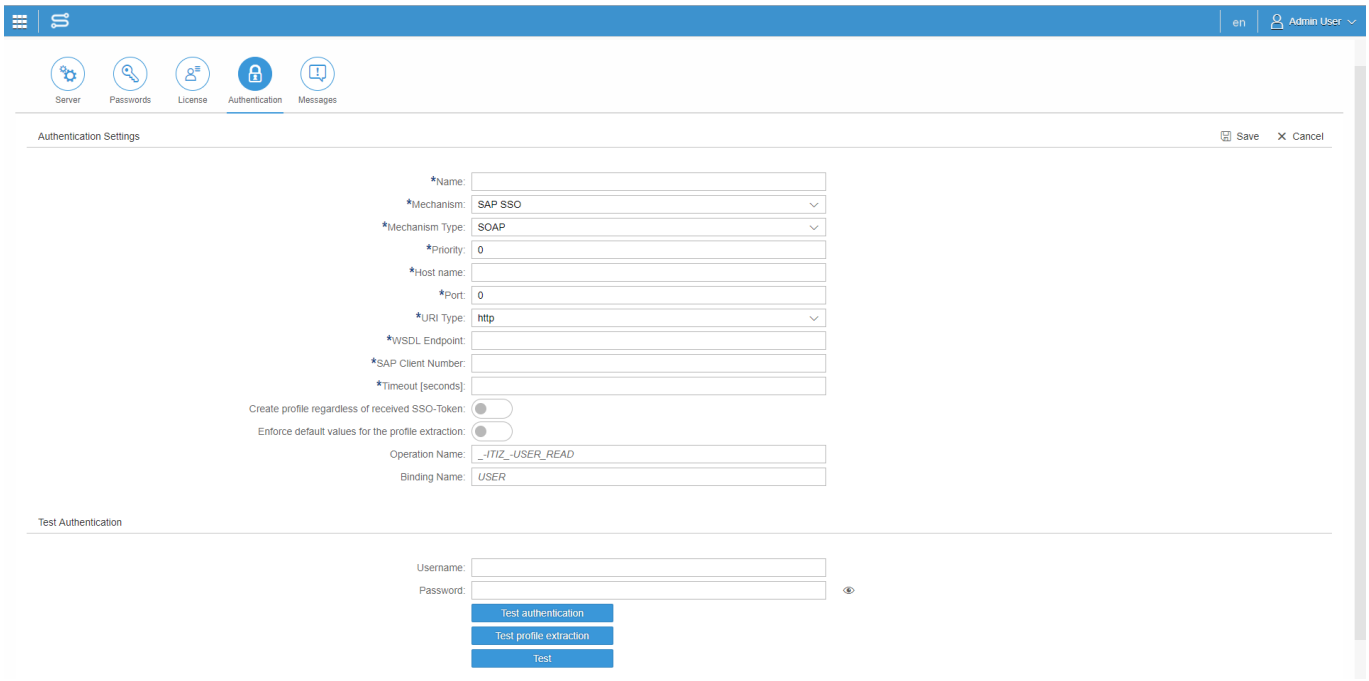
Name	Defines the name of the authentication
Mechanism	SAP SSO
Mechanism Type	RFC
Priority	This defines the order in which the authentication mechanisms are processed
SAP Server Type	The type of the SAP Server: Application Server, Gateway Service, Gateway Host, Message Server, Message Server Port
Host name	This defines the IP/ Host of the authentication service
SAP System ID	This defines the ID of the SAP System
SAP System Number	This defines the number of the SAP System
SAP System Language	This defines the language of the SAP System
SAP Client Number	This defines the number of the SAP Client
SAP Router Hostname	This defines the hostname of the SAP Router
SAP Router Port	This defines the authentication port of the SAP Router
Create profile regardless of received SSO-Token	If this switch is on, then the Simplifier will try to extract the profile. The used authentication method will be only: {Basic-Authentication}. If the username or password consists of non-alphanumerical symbols, then the authentication required for retrieving the user details might fail.
Enforce default values for the profile extraction	If this switch is on, then all missing information in the profile will be replaced by default values. This will even happen when critical information, required for further extraction steps is missing. If this switch is off, then the extraction process will be terminated and an error will be returned.
SAP Client Type	The type of the SAP Client: Not defined, R2, R3 or External
SAP Username Alias	You can assign an alias
SAP R3 System Number	This defines the number of the SAP R3 System
SAP System Group	This defines the group of the SAP System

You can test the profile extraction and token retrieval as well as the whole chain seperately.

SAP Single Sign On via SOAP

Using SAP SSO

You can use the SOAP service to authenticate via a SAP system and get a SSO2 Token for SAP SOAP webservices. The user synchronization is done by the iTiZZiMO own SAP RFC Module *-ITIZ-_USER_READ* or with a different SOAP-Webservice, which calls the standard SAP-Module *BAPI_GET_USER_DETAIL*.



Name	Defines the name of the authentication
Mechanism	SAP SSO
Mechanism Type	SOAP
Priority	This defines the order in which the authentication mechanisms are processed
Host name	This defines the IP/ Host of the authentication service
Port	This defines the authentication port
URI Type	This defines the URI Type (HTTP or HTTPS)
WSDL Endpoint	This is composed of Host name, Port, URI Type and Client Number
Client Number	This defines the SAP Client Number
Ignore Invalid SSL Certificates	You can ignore invalid SSL Certificates
Timeout (in seconds)	You can set a timeout
Operation Name	This defines the Operation name
Binding Name	This defines the Binding name

You can test the connection by inserting a Username and Password.

SAP WM Transfer Order

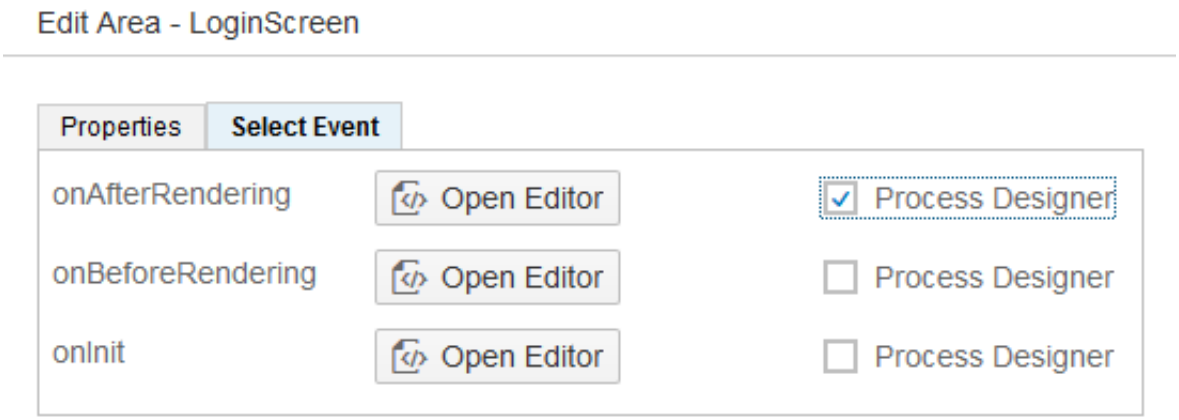
SAP WM (Warehouse Management) Transfer Order

Method	SAP Transaction	Description
CREATE	LT01	Create a Transfer Order with a single position.
READ	LT21	Load a Transfer Order using the stock number and the Transfer Order number.
DELETE	LT15	Cancel a Transfer Order.
CONFIRM	LT12	Confirm a Transfer Order.

Screen Events

Screen Events

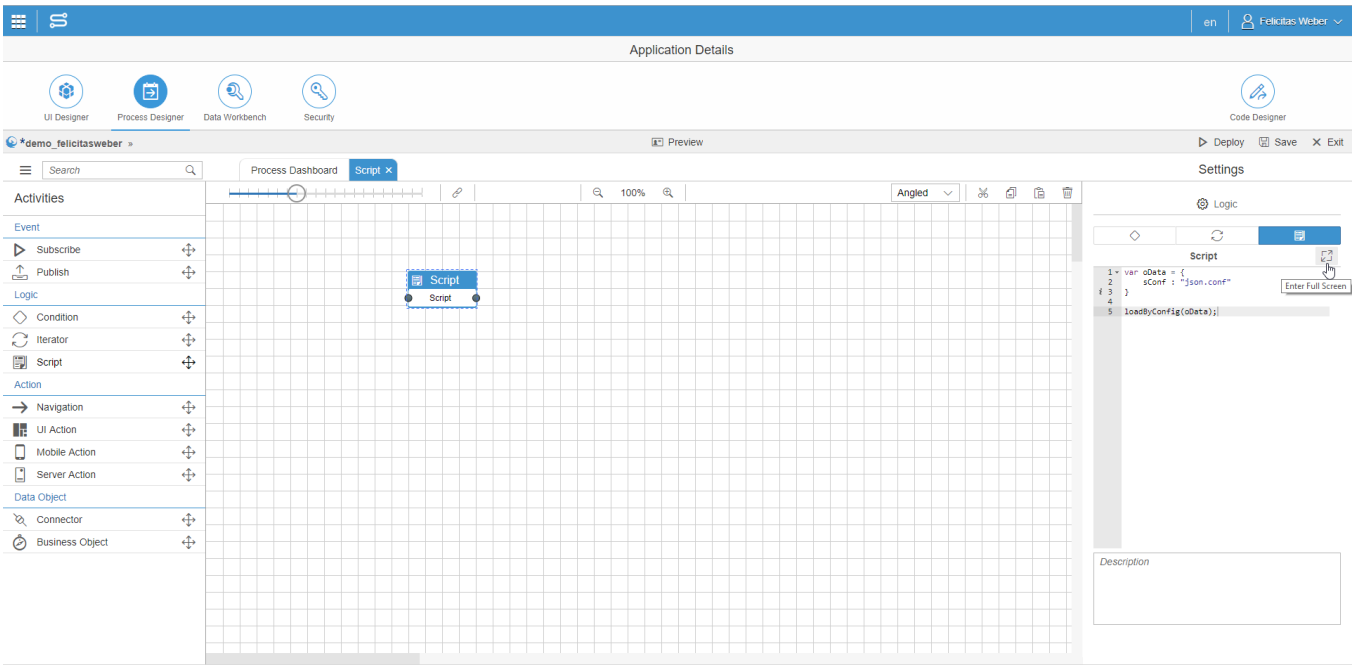
To read data from backend systems through a simplifier connector or execute logic you can use screen events.
You need to check the box in order to work with the event of this widget in the edit mode of your user story (Process Designer) later.

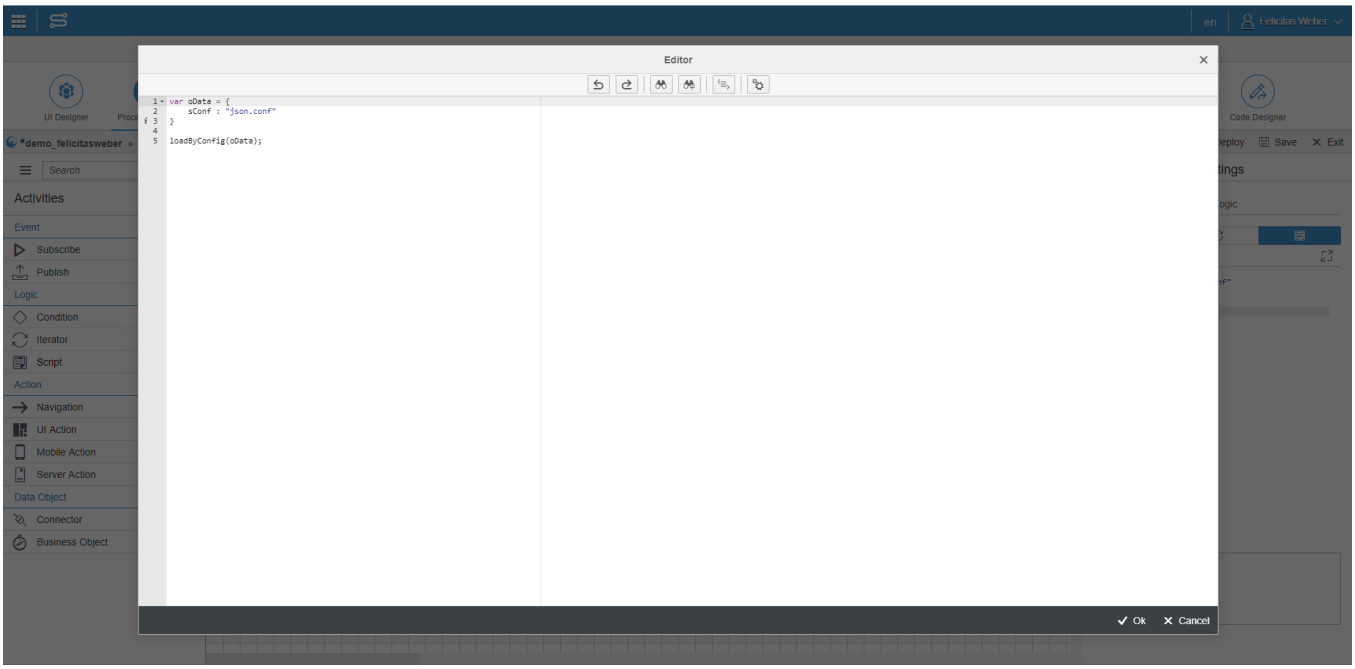


Script

The Script element allows you to integrate a custom JavaScript snippet. It is activated by the previous event. Drag the activity “Script” underneath “Logic” in the drawing area.

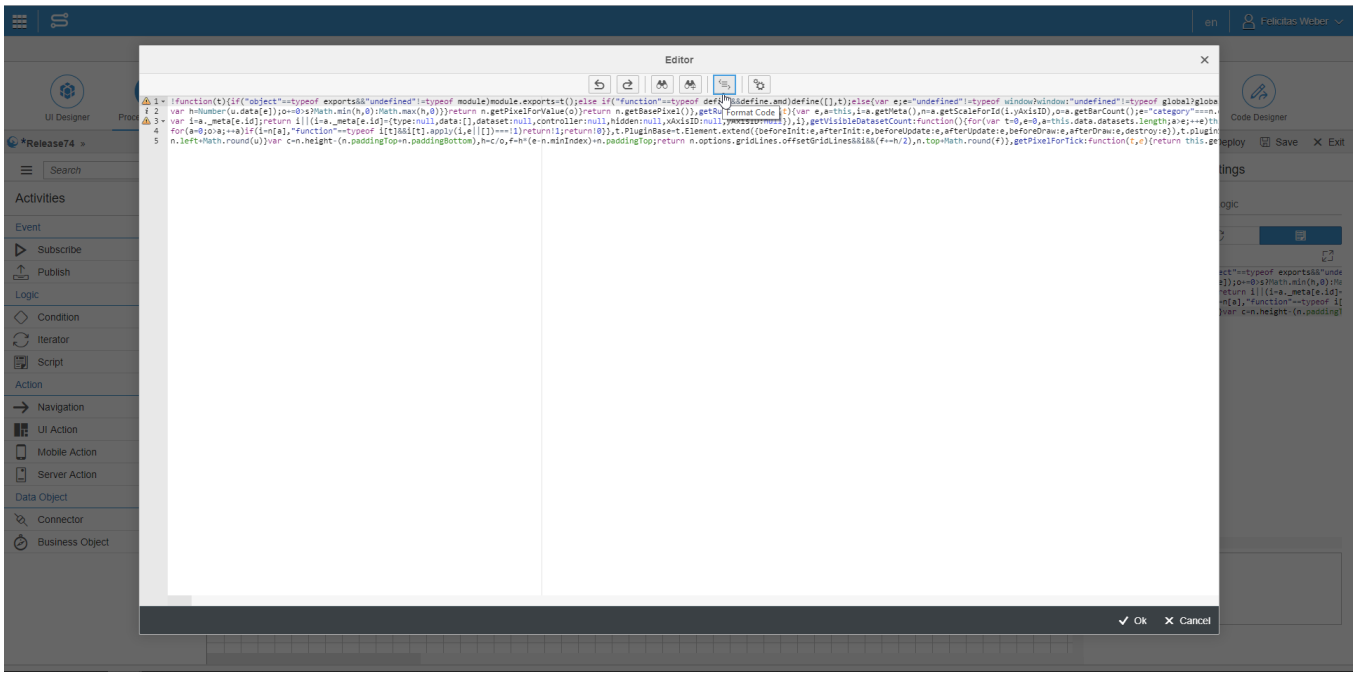
The activity “Script” shows a preview of the JavaScript full screen and its (optional) description on the right side. You can also enter code in this section, the changes were saved and also transferred in the full screen. The full screen will be opened by double clicking on the shape or by clicking on the appropriate icon.


en
Felicitas Weber
Application Details
UI Designer
Process Designer
Data Workbench
Security
Code Designer
*demo_felicitasweber
Preview
Deploy
Save
Exit
Search
Process Dashboard
Script
Settings
Activities
Event
Subscribe
Publish
Logic
Condition
Iterator
Script
Action
Navigation
UI Action
Mobile Action
Server Action
Data Object
Connector
Business Object
100%
Angled
Logic
Script
var oData = {
sConf : "json.conf"
}
loadByConfig(oData);
Enter Full Screen
Description

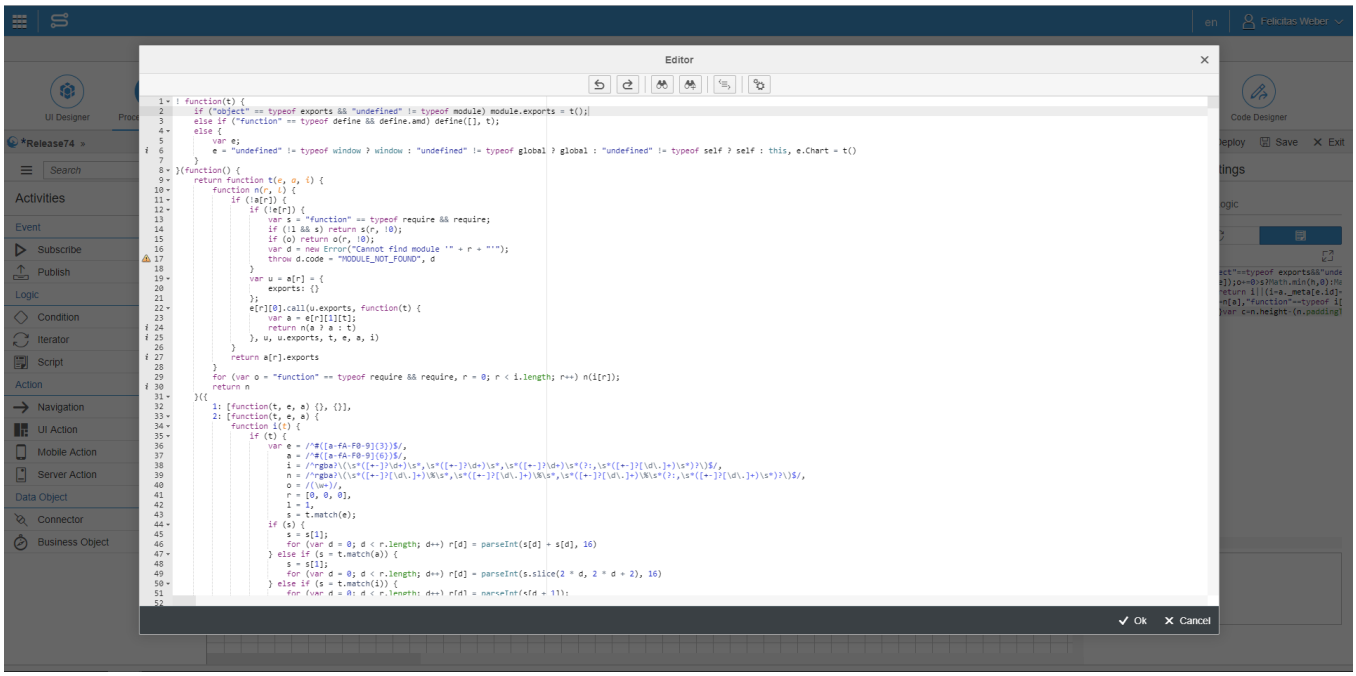


Pretty Print

You can prettify JavaScript code with just one click. Therefore you have to have the script activity on full screen. By clicking on the appropriate icon, your code will be simply prettified.



unformatted code

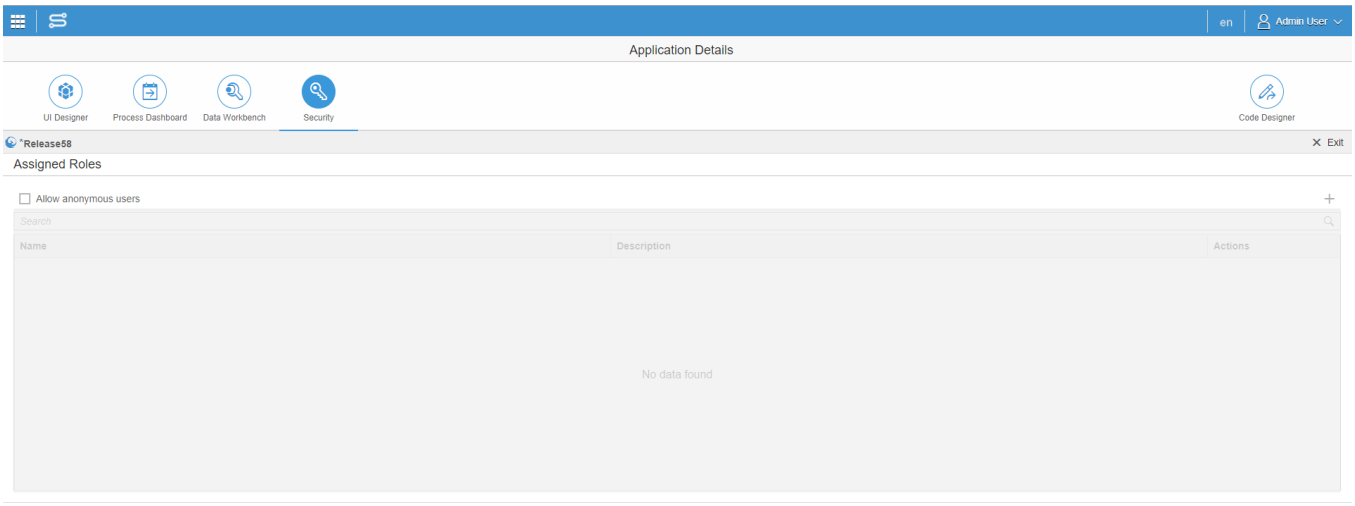


formatted code

Security

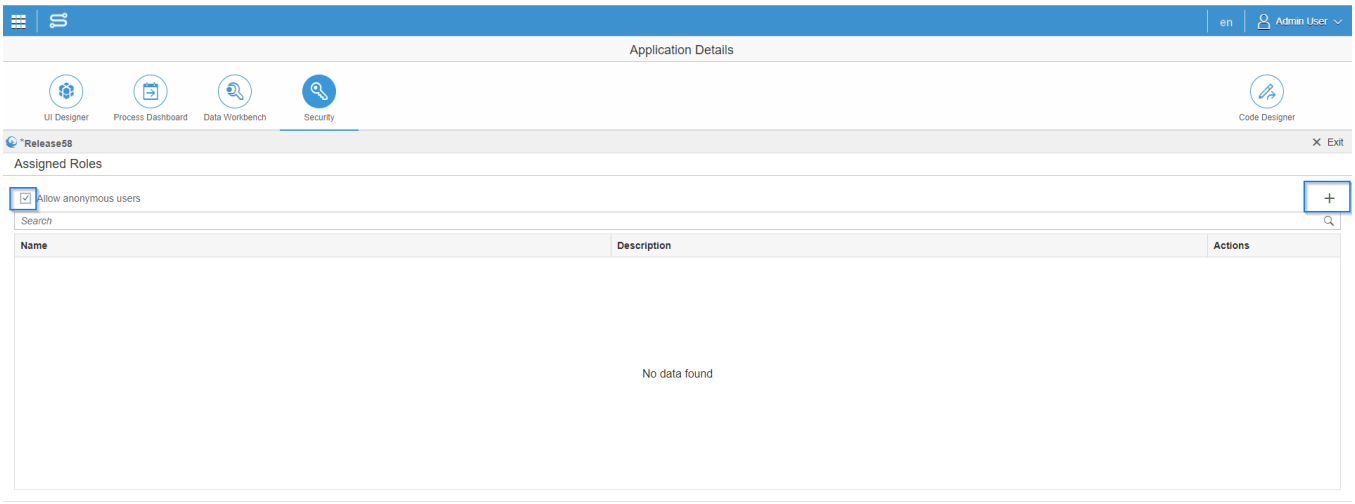
Assignment

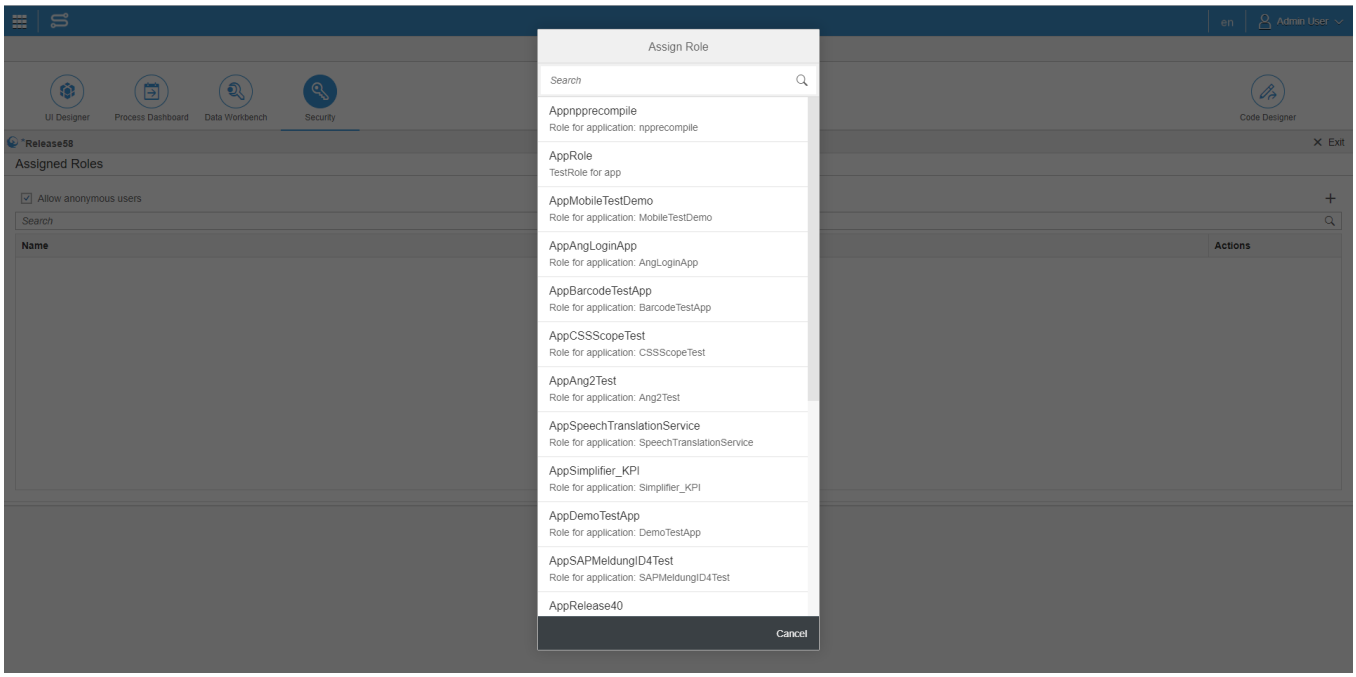
You can assign roles to apps, which is the first step in order to have anonymous logins.

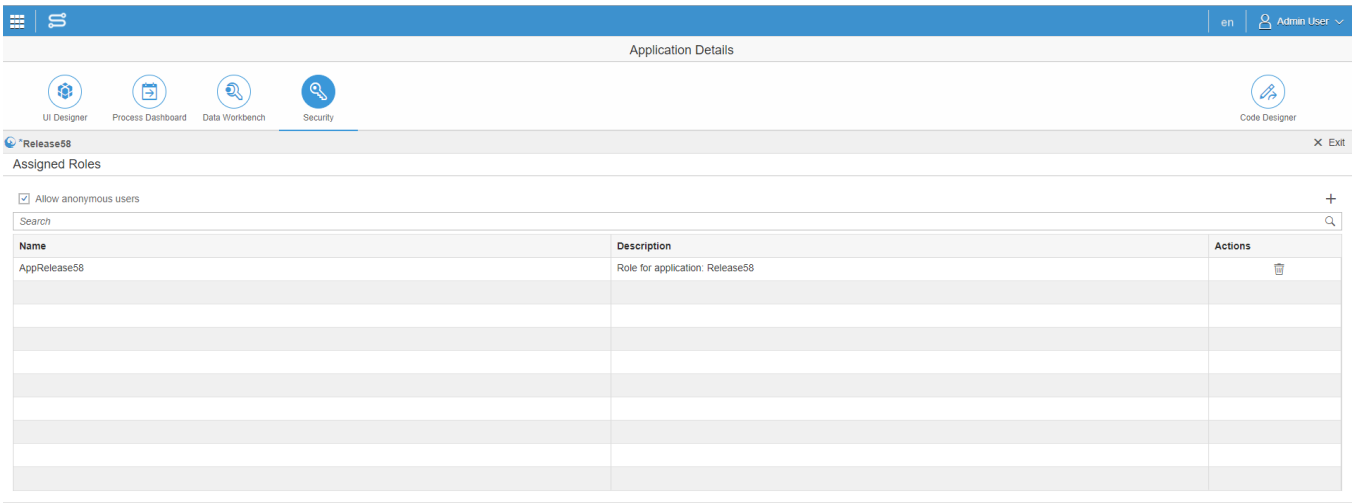


All assigned roles to the app are imported and exported by the transport system. By default it is not allowed for anonymous users to execute data operations in the app.

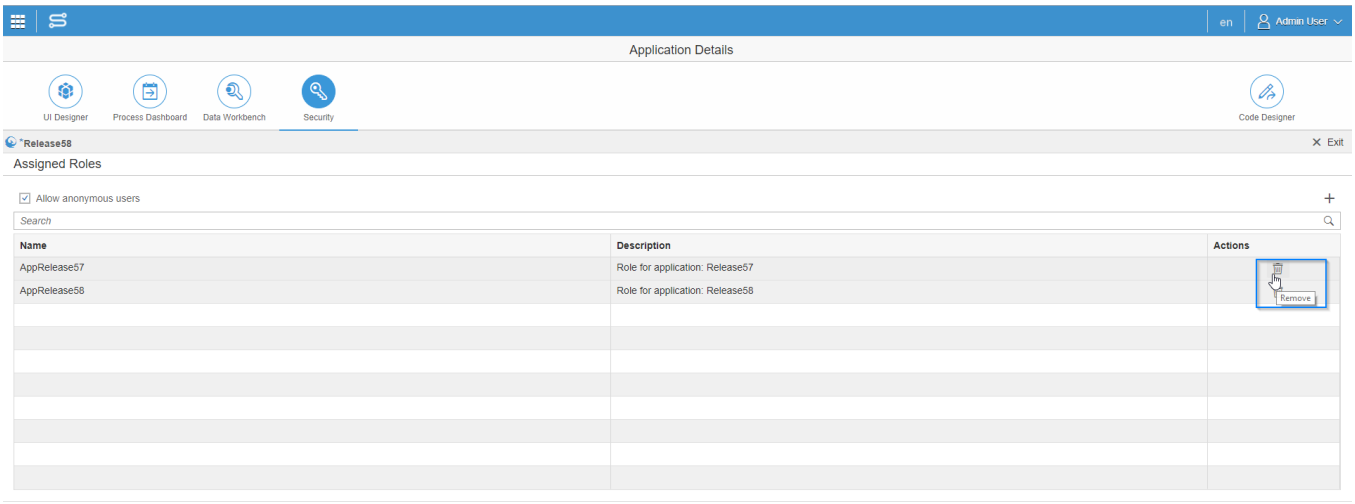
Allowing anonymous users must be allowed with a conscious consent. For this you have to tick the checkbox besides “Allow anonymous users”. After that, the plus icon is available to assign a role.


en
Admin User
Application Details
UI Designer
Process Dashboard
Data Workbench
Security
Code Designer
*Release58
Exit
Assigned Roles
Allow anonymous users
Search
Name
Description
Actions
No data found

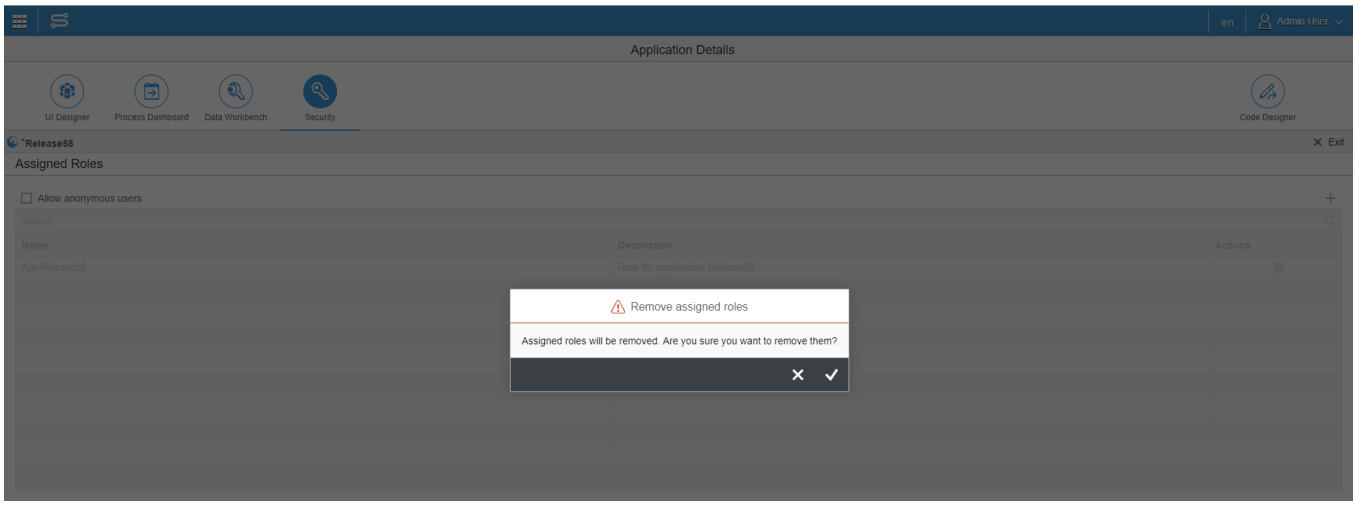

en
Admin User
UI Designer
Process Dashboard
Data Workbench
Security
Code Designer
*Release58
Exit
Assigned Roles
Allow anonymous users
Search
Name
Actions
Assign Role
Search
Appnpprecompile
Role for application: npprecompile
AppRole
TestRole for app
AppMobileTestDemo
Role for application: MobileTestDemo
AppAngLoginApp
Role for application: AngLoginApp
AppBarcodeTestApp
Role for application: BarcodeTestApp
AppCSSScopeTest
Role for application: CSSScopeTest
AppAng2Test
Role for application: Ang2Test
AppSpeechTranslationService
Role for application: SpeechTranslationService
AppSimplifier_KPI
Role for application: Simplifier_KPI
AppDemoTestApp
Role for application: DemoTestApp
AppSAPMeldungID4Test
Role for application: SAPMeldungID4Test
AppRelease40
Cancel



You can also remove roles. All you have to do is click on the corresponding icon.



When disallowing anonymous users, all assigned roles will be removed.



Runtime

When you navigate to an application, an authentication of the app for accessing an application token is triggered. All roles assigned to the app are linked to the application token. This feature allows anonymous users to e.g. execute connectors.

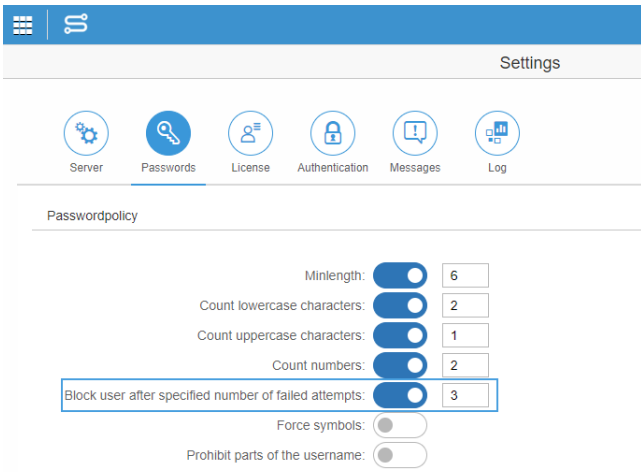
As a further security aspect, the app needs to provide a security functionality in order to obtain an application token.

Security Guidelines

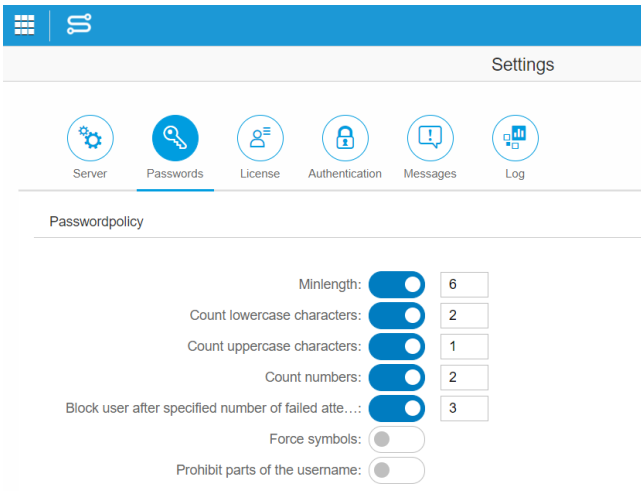
The following tables refer to the OWASP Top 10 Security Measures 2017 and explain the security measurements and relevant settings to avoid security flaws within the Simplifier platform.

A1: Injection	**Simplifier Prevention**
User supplied data is not validated, filtered or santizied by the application. Hostile data is used directly with dynamic queries or non-parameterized calls for the interpreter without context-aware escaping. Hostile data is directly used or concatenated, so that the SQL or command contains both structure and hostile data in dynamic queries, commands, or in stored procedures.	Double validation against data type definition (client and bac All connector calls are parameterized, including SQL queries All connector calls are parameterized, including SQL queries

A2: Broken Authentication	**Simplifier Prevention**
You may have authentication weakness if your application… Permits credential stuffing, where the attacker has a list of valid usernames and passwords. Permits brute force or other automated attacks.	The Simplifier uses a token authentication. This token expire server security settings). The permission object view own user details prevents the app We secure the authentication by deactivating the account afte



Permits default, weak or well-known passwords.	The admin can secure the password by setting password rules



Uses weak or ineffectual credential recovery and forgot password processes, such as “knowledge-based answers”, which cannot be made safe.	Actual we provide an one-time link to reset the password via questions.
Using plain text, encrypted, or weakly hashed passwords permits the rapid recovery of passwords using GPU crackers or brute force tools.	Our password does not exist in the database. We are using en
Has missing or ineffective multi-factor authentication.	We provide multi-factor authentication through OAuth 2.0 au

A3: Sensitive Data Exposure	**Simplifier Prevention**
Data of a site is transmitted in clear text, internally or externally.	The Simplfier runs behind web proxy and all data is encrypte
Sensitive data is stored in clear text, including backups.	The Simplfier doesn’t save personal data or sensitive data on existing backend systems for that.
Old or weak cryptographic algorithms are used either by default or in older code.	We rely on bcrypt.
Default crypto keys are in use, weak crypto keys are generated or re-used, or proper key management or rotation is missing.	We rely on bcrypt.
Encryption is not enforced, e.g. security policies or headers are missing.	If you use our provided reverse proxy template, this can not h

A4: XML External Entities (XXE)	**Simplifier Prevention**
Your application accepts XML directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.	We only support XML formats with REST and SOAP connec and no direct upload or data processing within an application
Any of the XML processors in the application or SOAP based web services has document type definitions (DTDs) enabled.	Yes, we parse WSDL and consider DTD if available and gen definitions within the Simplifier itself.
If your application uses SOAP prior to version 1.2, it is likely susceptible to XXE attacks if XML entities are being passed to the SOAP framework.	We support SOAP version 1.2
SAST tools can help detect XXE in source code, although manual code review is the best alternative in large, complex apps with many integrations.	We only support XML formats with REST and SOAP connec and no direct upload or data processing within an application
Being vulnerable to XXE attacks likely means that you are vulnerable to other billion laughs denial-of-service attacks.	We only support XML formats with REST and SOAP connec and no direct upload or data processing within an application

A5: Broken Access Control	**Simplifier Prevention**
Bypassing access control checks by modifying the URL, internal app state,	This can not happen because every API is secured by permiss

or the HTML page, or simply using a custom API attack tool.	user interface itself.
Allowing the primary key to be changed to another's users record, such as viewing or editing someone else's account.	This can not be happen because every API is secured by perm user interface itself.
Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.	The Simplifier authentication token is assigned to the client f unique attributes like system configuration, browser version, etc. If you use a foreign authentification token to gain privile client system and using the same browser and user credential foreign token.
Metadata manipulation, such as replaying or tampering with a JWT access control token or a cookie or hidden field manipulated to elevate privileges.	The Simplifier authentication token is not stored in a cookie the client fingerprint.
CORS misconfiguration allows unauthorized API access.	If you use our recommend reverse proxy example you will be misconfiguration.
Force browsing to authenticated pages as an unauthenticated user, or to privileged pages as a standard user or API not enforcing access controls for POST, PUT and DELETE.	All APIs are secured by permission objects.

A6: Security Misconfiguration	**Simplifier Prevention**
Unnecessary features are enabled or installed.	Features are enabled via permission objects.
Default accounts and their passwords are still enabled and unchanged.	Every single installation will get an automatically generated are not using default passwords.
Does your error handling reveal stack traces or other overly informative error messages to users?	The logging and monitoring feature clears backend password include security related information within the logs.
Old configurations are used with updated software which may not be backward compatible.	All configuration objects are automatically migrated to the ne
Security settings in application servers, application frameworks, libraries, databases, etc. are not set to secure values.	In our cloud environment we are using very strong passwords
The server doesn't send security directives to client agents or they are not set to secure values for web applications.	HSTS headers are set in the web reverse proxy.
Any of your software is out of date.	Every customer is forced to upgrade once a year – older versi support that includes also security patches.

A7: Cross-Site Scripting (XSS)	**Simplifier Prevention**
Reflected XSS: Your app or API includes unvalidated and unescaped user input as part of HTML output or there is no content security policy header.	We use the content security header.
Stored XSS: Your app or API stores unsanitized user input that is viewed at a later time by another user or an administrator.	All API fields are validated against data type defintions to pr
DOM XSS: JavaScript frameworks, single page apps, and APIs that dynamically include attacker-controllable data to a page are vulnerable to DOM XSS.	We provide encoding function also for JavaScript for HTML

A8: Insecure Deserialization	**Simplifier Prevention**
The serialization mechanism allows for the creation of arbitrary data types, AND	We transform every data to JSON format with sanitize option
There are classes available to the application that can be chained together to change application behavior during or after deserialization, or unintended content can be used to influence application behavior, AND	We only accept valid JSON data. Each data object will be ch default.
The application or API accepts and deserializes hostile objects supplied by an attacker, or an application uses serialized opaque client side state without appropriate tamper resistant controls. OR	We only accept valid JSON data. Each data object will be ch default.

Security state sent to an untrusted client without some form of integrity control is likely vulnerable to deserialization.	Every client has to be authenticated to check if the client is tr

A9: Using Components with Known Vulnerabilities	**Simplifier Prevention**
You don’t know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.	All depending libraries are checked against security exploits
Your software is out of date. This includes the OS, Web/App Server, DBMS, applications, APIs and all components, runtime environments and libraries.	All components are checked against update with VersionEye
You don’t know if they are vulnerable. Either if you don’t research for this information or if you don’t scan them for vulnerabilities on a regular base.	All components are checked against update with VersionEye
You don’t fix nor upgrade the underlying platform, frameworks and dependencies in a timely fashion.	We provide security hotfixes for the last 2 major releases (1 y
You don’t secure the components’ configurations.	We secured the complete configuration.

A10: Insufficient Logging & Monitoring	**Simplifier Prevention**
Auditable events, such as logins, failed logins, and high value transactions are not logged.	Failed login attempts and valid logins are logged as such as e call. No matter what kind of value this transaction is.
Logs of applications and APIs are not monitored for suspicious activity.	All applications are monitored.
Alerting thresholds and response escalation according to the risk of the data held by the application is not in place or effective.	Actual it is not possbile to alert security response teams or ad security events – this will be available in further releases.

Send message via Process Dashboard

To send messages via the Process Dashboard, you need to take the following steps:

1. Drag the Asynchronous Data Object activity into the drawing area.
2. Select the appropriate connector and the connector call for sending the message.
3. Fill the parameters in the input mapping.

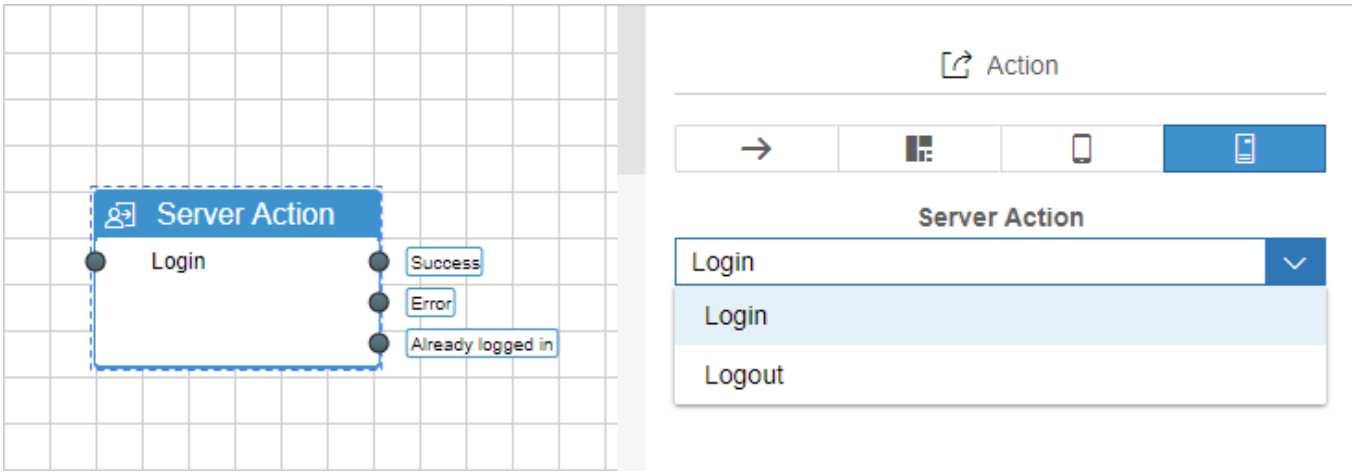
Send message via Script

An object can be passed as a message by converting it into a JSON string:

```
JSON.stringify({propertyOne: 'abc', propertyTwo: 5})

var lo_payload = {
'msg:'<requested message>'
};

function successCallback(po_result) {
//Enter any code that will be executed in case of success
}

function errorCallback(po_result) {
//Enter any code that will be executed in case of error
}

this.callConnector("<connector-
name>", lo_payload, successCallback, true, true, errorCallback);
```

Server Action

The Server Action enables you to configure the login or logout for an application.



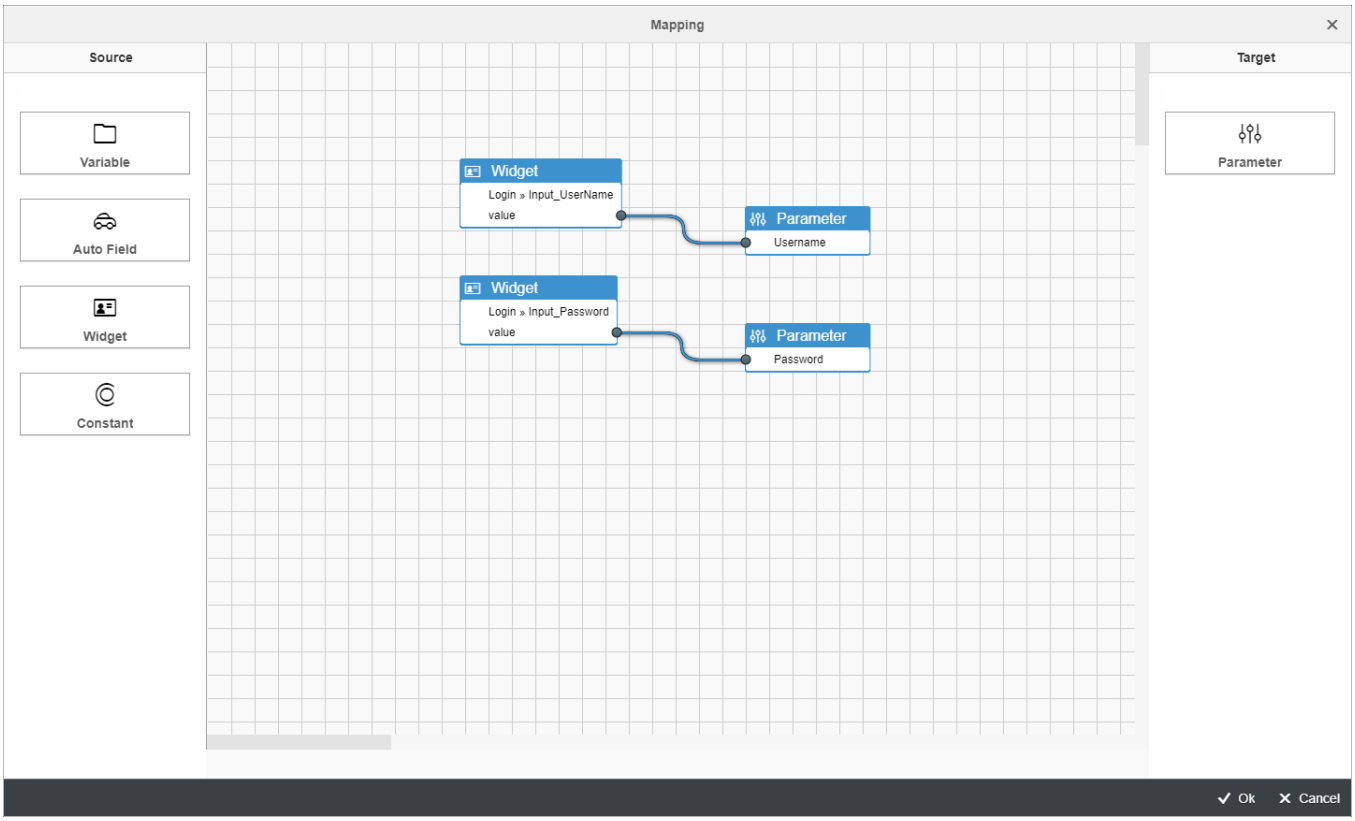
After mapping the according parameters you can decide which port you'd like to connect.

Port	Description
success	The process will be executed after a successful login or logout.
error	The process will be executed after a failed login or logout.
already logged in	The process will be executed if the user is already logged in.

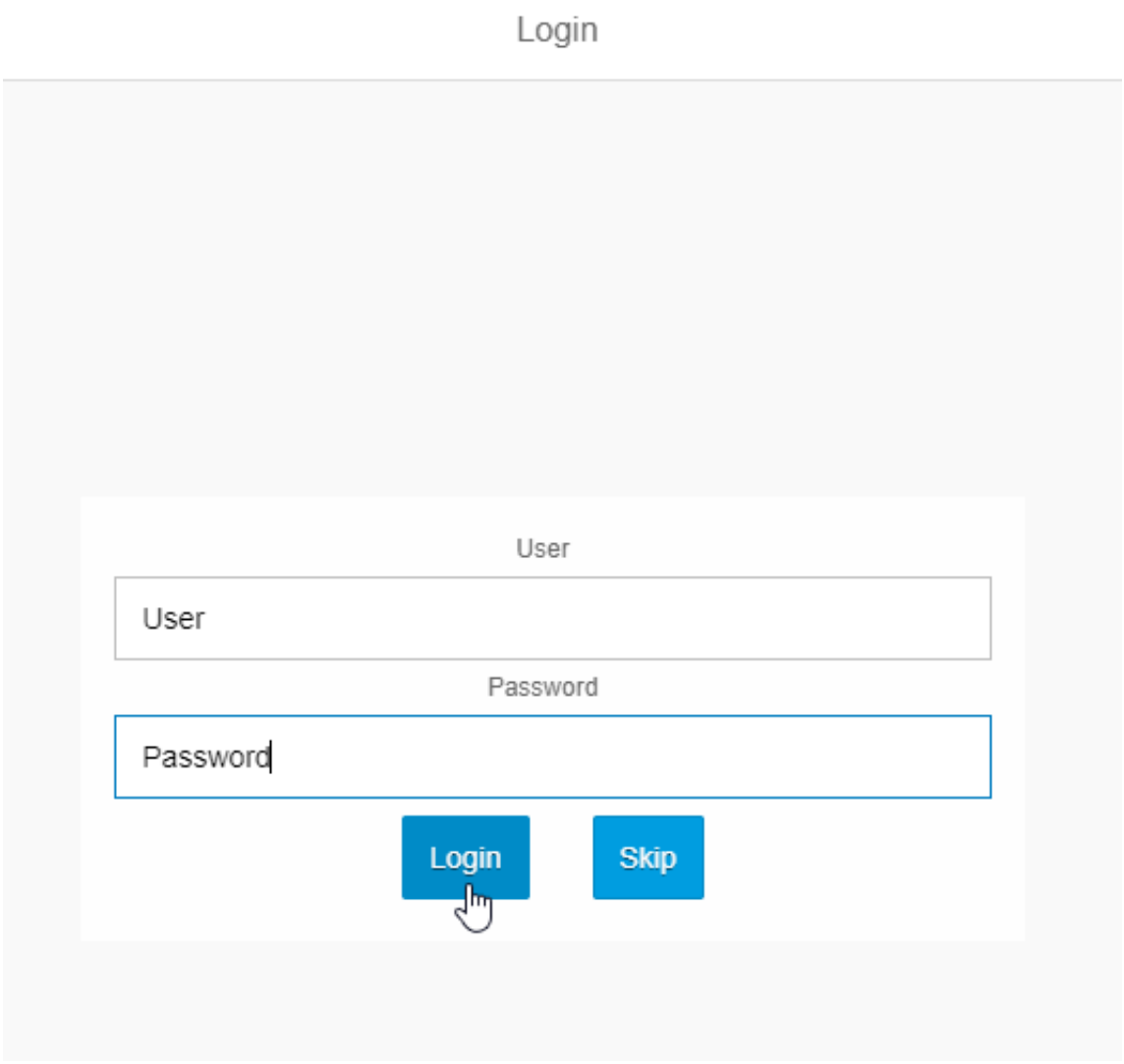
User Credentials

The Server Action needs two parameter as input, if you want the user of the application to fill in his username and password.

You can map the widgets in which the user fills in the information with the equivalent input parameter. (By double clicking on the Widget, you can choose the exact property that you need).



The Login screen of your application might look like this:

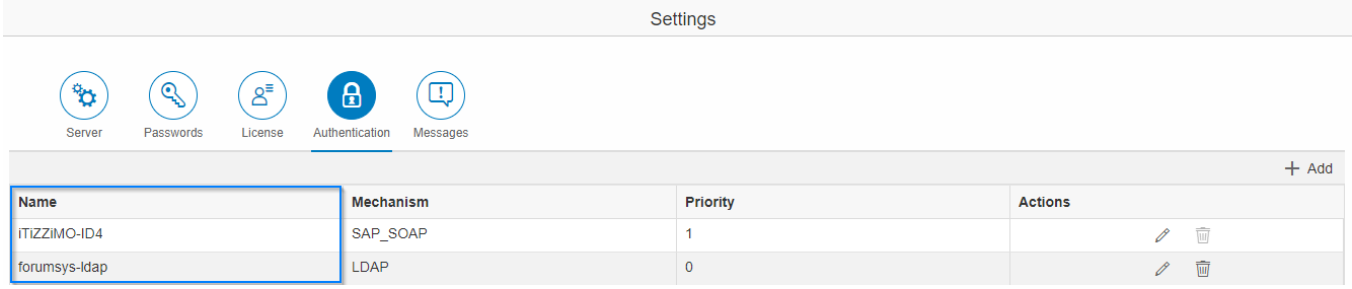


Single Sign-on

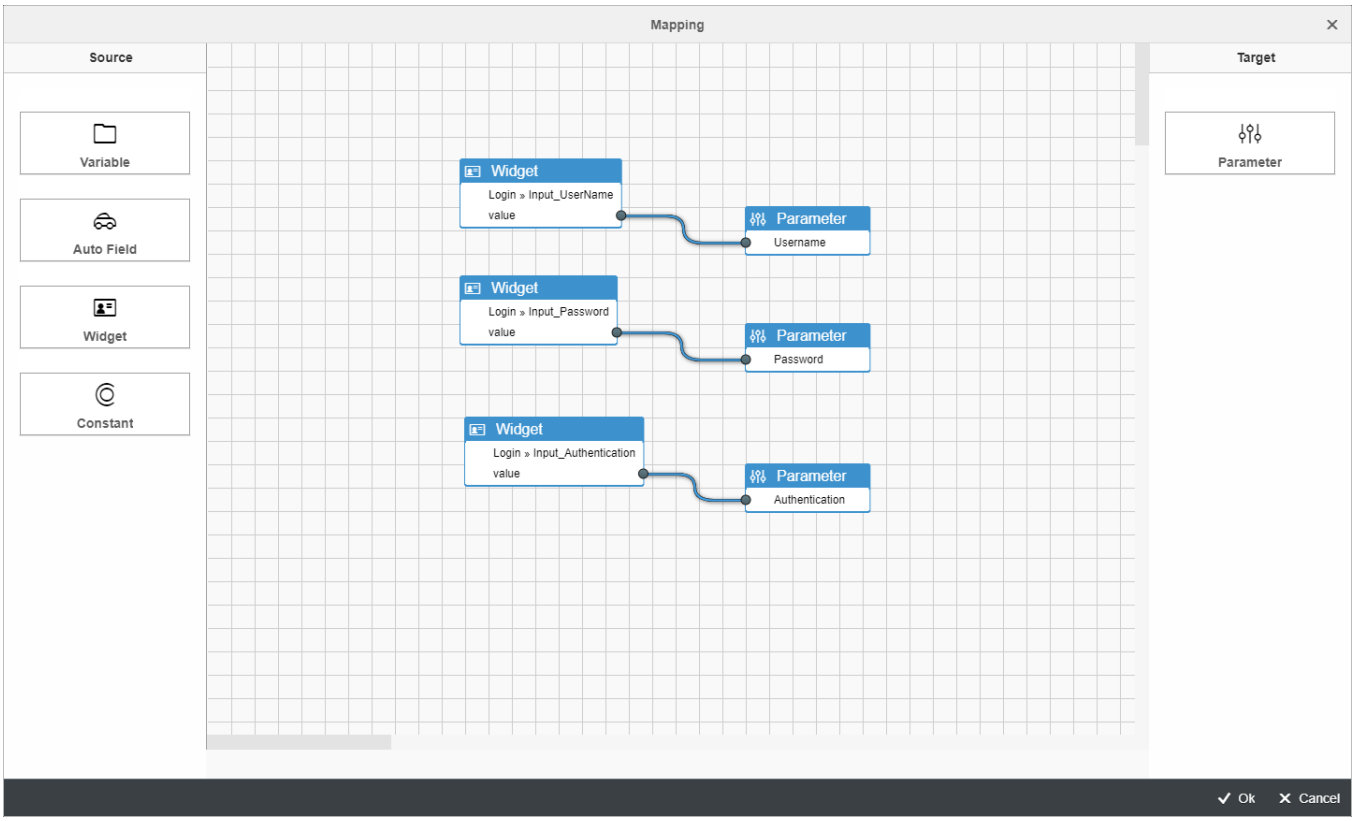
You can use a Single Sign-on if you want to allow a user to use a single set of login credentials to access multiple applications. In difference to the first login method you have to map an additional parameter, the authentication.

Therefore you have to add this authentication method to the Simplifier in advance (this will be an admin task – take a look at "Settings for Admins").

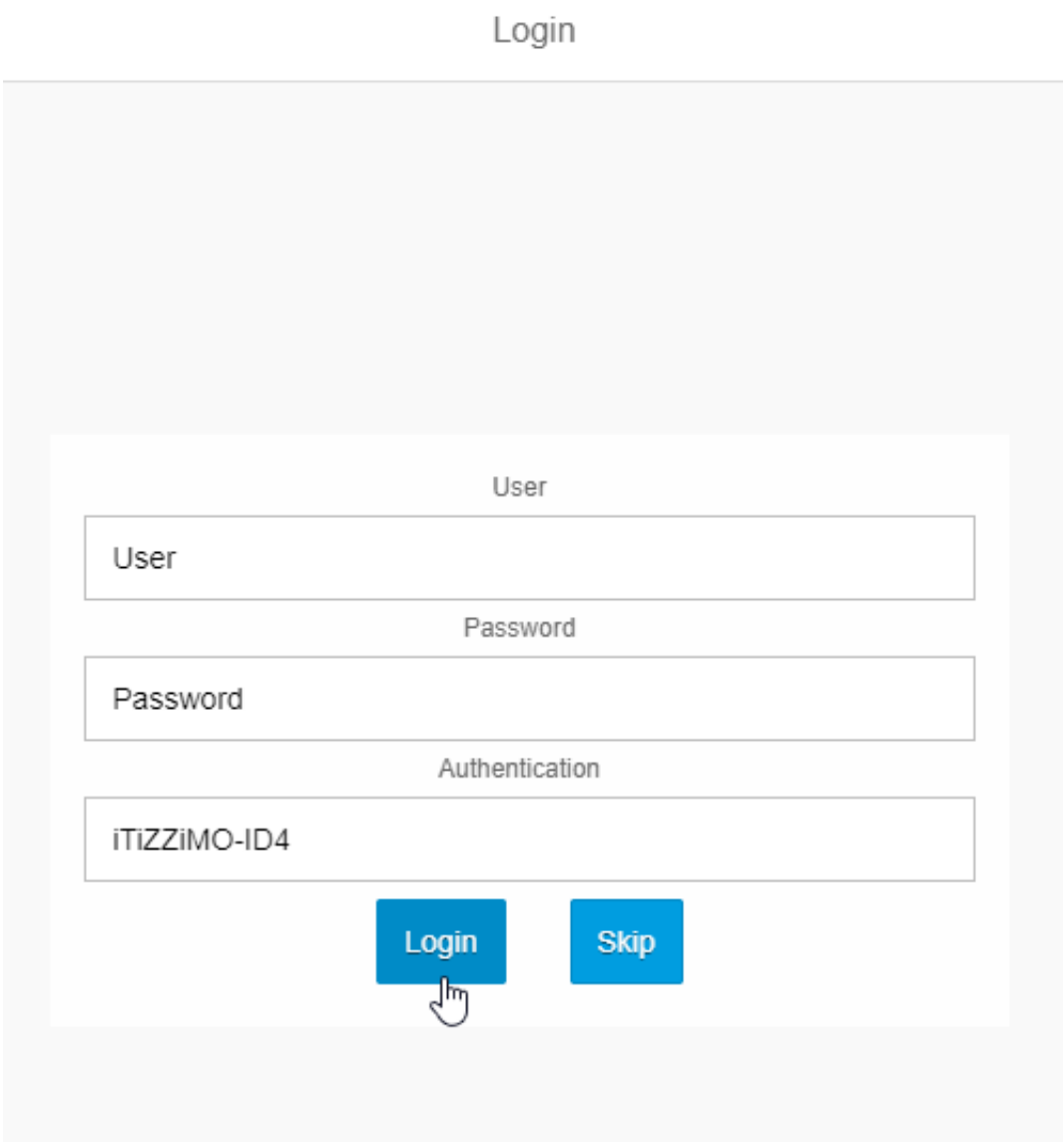
The name that was given to the authenciation is the one that is needed as an input parameter.



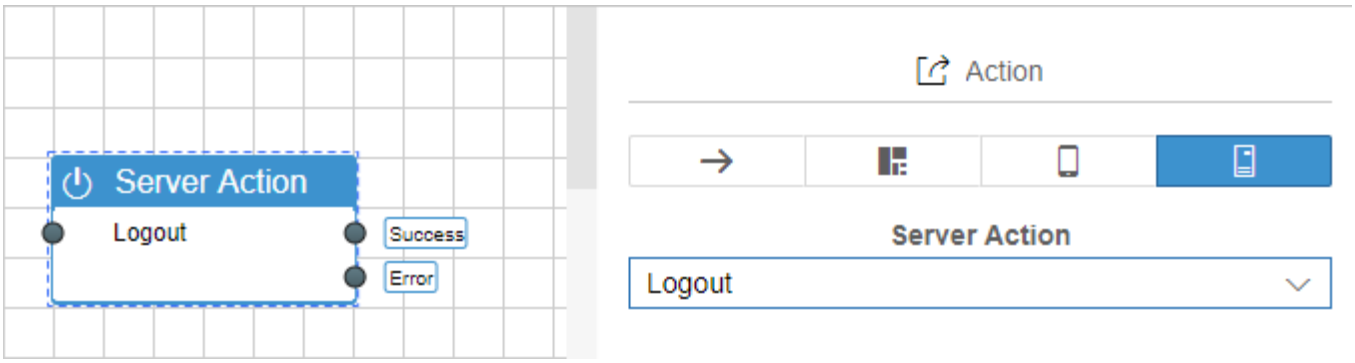
This is what the Input Mapping should look like:



The Login screen of your application might look like this:



For the Server Action Logout you don’t have to configure anything additionally.


Server Action
Logout
Success
Error
Action
Server Action
Logout

Server-Side Business Object API

You can access any methods, like Logging, Utils/Tools, Email, Connectors, Business Objects, Plugins, Users, Permissions, Groups and Roles of the Simplifier by using the Simplifier Object.

Logging

```
Simplifier.Log.info(message: string): void

Simplifier.Log.info(message: string, details: string|object): void


Simplifier.Log.warn(message: string): void

Simplifier.Log.warn(message: string, details: string|object): void


Simplifier.Log.error(message: string): void

Simplifier.Log.error(message: string, details: string|object): void


Simplifier.Log.debug(message: string): void

Simplifier.Log.debug(message: string, details: string|object): void
```

Examples:

```
Simplifier.Log.warn("Test log entry");

Simplifier.Log.info("Test log entry with Details", "Details String");

Simplifier.Log.error("Test log entry with Details", {'key': 'value'});
```

Utils/Tools

```
Simplifier.Util.xml2json(xml: string): string

Simplifier.Util.json2xml(json: string): string

Simplifier.Util.encodeBase64(string: string): string
```

```
Simplifier.Util.decodeBase64(json: string): string
```

Examples:

```
var encoded = Simplifier.Util.encodeBase64("String to encode");

var decoded = Simplifier.Util.decodeBase64(encoded);
```

Email

```
Simplifier.Email.sendTemplateMail(payload: string|object): void
```

Examples:

```
Simplifier.Email.sendTemplateMail({

templateNamespace: "MyNamespace", templateName: "MyTemplate", emailCharset: "UTF_8",
emailMime: "text/html",

receiver: "test@test.de", receiverCC: ["cc@test.de"], subject: "My Mail", "data": {"V
ar1": "Replacement1"}

});
```

Connectors

```
Simplifier.Connector.<ConnectorName>(payload?: string|object): object

Simplifier.Connector.<ConnectorName>.<CallName>(payload?: string|object): object
```

Examples:

```
var connectorResult = Simplifier.Connector.MySoap({bindingName: "Binding", "operation
Name": "MyOp", "soap": {"foo": "bar"}});
```

```
var connectorCallResult = Simplifier.Connector.MySoap.myCall({"Foo": "bar"});
```

Business Objects

```
Simplifier.BusinessObject.<BOName>.<MethodName>(payload?: string|object, parametrized
?: boolean = true): object

Simplifier.CurrentBusinessObject.<MethodName>(payload?: string|object, parametrized?:
 boolean = true): object  (--> BOName = Currently executed Business Object)
```

Examples:

```
var otherMethodResult = Simplifier.BusinessObject.OtherBO.someMethod({"foo": "bar"});

var unparamtetrizedResult = Simplifier.BusinessObject.OtherBO.someMethod("{\"foo\": \
"baz\"}", false);

var sameBoResult = Simplifier.CurrentBusinessObject.someMethodOnSameBO({"baz": "baz"}
);

var noArgsResult = Simplifier.CurrentBusinessObject.methodWithoutArgs();
```

Plugins

```
Simplifier.Plugin.<PluginName>.<SlotName>(payload?: string|object): object
```

Examples:

```
var repos = Simplifier.Plugin.contentRepoPlugin.listRepos();

var newRepo = Simplifier.Plugin.contentRepoPlugin.createRepo({name: "MyRepo"});

Simplifier.Plugin.contentRepoPlugin.updateRepo("{\"id\": 15}");
```

Users

```
Simplifier.User.getAll(): Array<object>

Simplifier.User.getById(id: number): object|null

Simplifier.User.getByName(loginName: string): object|null

Simplifier.User.getCurrentUser(): object|null


Simplifier.User.create(data: object|string): object


Simplifier.User.update(id: number, data: object|string): object

Simplifier.User.update(loginName: string, data: object|string): object


Simplifier.User.delete(id: number): void


Simplifier.User.checkLogin(login: string, password: string): boolean


Simplifier.User.assignRole(loginName: string, roleId: string): void

Simplifier.User.assignRole(id: number, roleId: string): void

Simplifier.User.unassignRole(loginName: string, roleId: string): void

Simplifier.User.unassignRole(id: number, roleId: string): void


Simplifier.User.assignGroup(loginName: string, roleId: string): void

Simplifier.User.assignGroup(id: number, roleId: string): void

Simplifier.User.unassignGroup(loginName: string, groupId: number): void

Simplifier.User.unassignGroup(id: number, groupId: number): void


Simplifier.User.getAttributes(id: number): Array<object>

Simplifier.User.getAttributes(loginName: string): Array<object>

Simplifier.User.getAttribute(id: number, name: string, category: string): string|null
```

```
Simplifier.User.getAttribute(loginName: string, name: string, category: string): stri
ng|null

Simplifier.User.setAttribute(id: number, name: string, category: string, value?: stri
ng|null = null): void

Simplifier.User.setAttribute(loginName: string, name: string, category: string, value
?: string|null = null): void



Simplifier.User.resetPasswordWithEmailTemplate(id: number, emailData: object|string):
 void

Simplifier.User.resetPasswordWithEmailTemplate(loginName: string, emailData: object|s
tring): void



Simplifier.User.setPassword(oneTimeHash: string, newPassword: string): void
```

Permissions

```
Simplifier.Permission.checkPermission(permissionName: string, characteristic: string)
: boolean

Simplifier.Permission.checkPermissionCharacteristic(permissionName: string, character
istic: string, characteristicValue: string): boolean
```

Groups

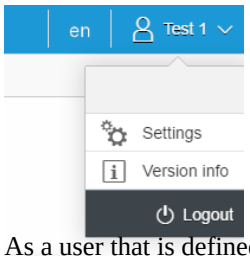
```
Simplifier.Group.getAll(): Array<object>

Simplifier.Group.getById(id: number): object|null



Simplifier.Group.create(data: object|string): object



Simplifier.Group.update(id: number, data: object|string): object



Simplifier.Group.delete(id: number): void
```

```
Simplifier.Group.getUsersByGroup(id: number): Array<object>
```

Roles

```
Simplifier.Role.getAll(): Array<object>

Simplifier.Role.getById(id: string): object|null


Simplifier.Role.create(data: object|string): object


Simplifier.Role.update(id: string, data: object|string): object


Simplifier.Role.delete(id: string): void


Simplifier.Role.getUsersByRole(id: string): Array<object>
```

Settings



As a user that is defined as an admin by the corresponding role, you can hit the settings button by clicking on your user profile name on the top right corner.

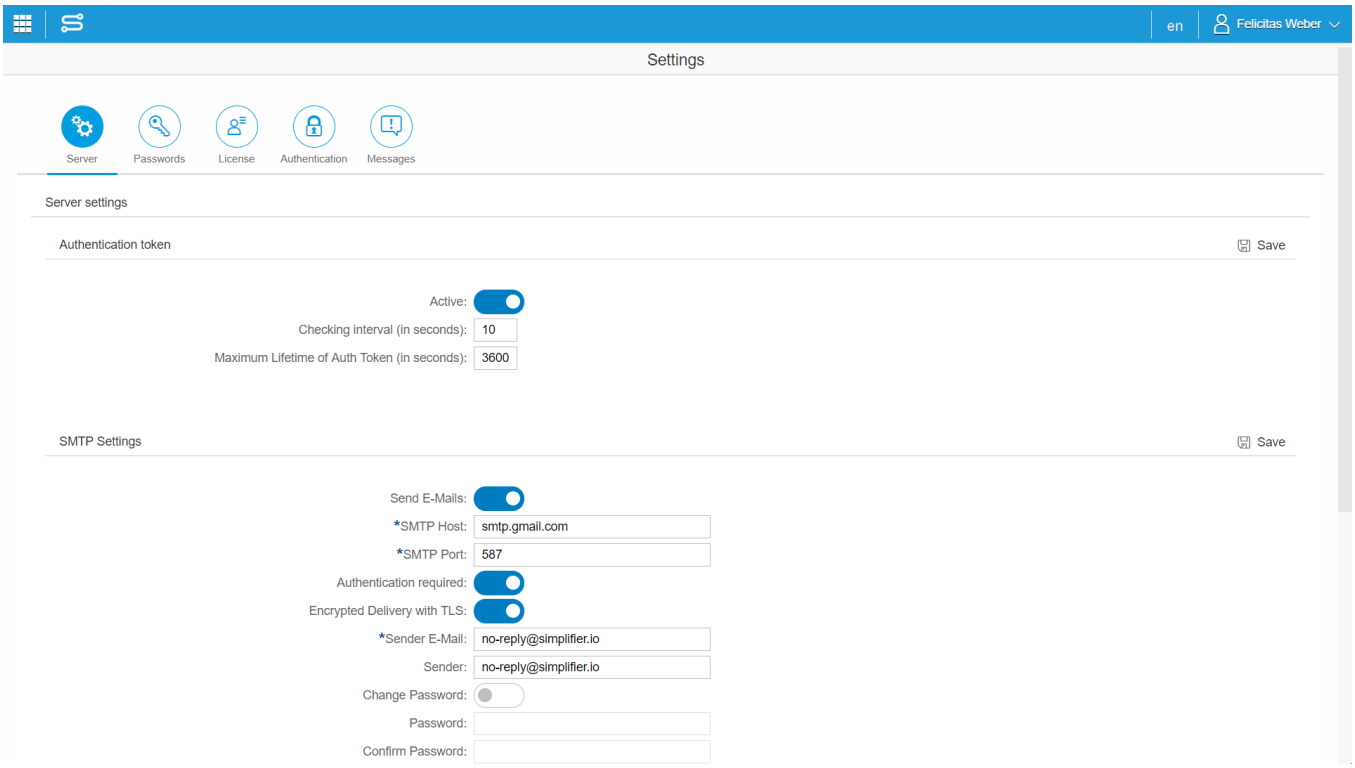
Within the settings panel, you can define several settings.

Server Settings

The **Server settings** control your session. At the subtitle **Authentication token** you can set the lifetime of the authentication token and its checking interval.

If you set the authentication token to active, the checking interval to 10 seconds and the maximum lifetime of authentication token to 3600 seconds, that means that every 10 seconds it will be checked if you are inactive. If you are inactive for about 3600 seconds, you will be automatically logged out.

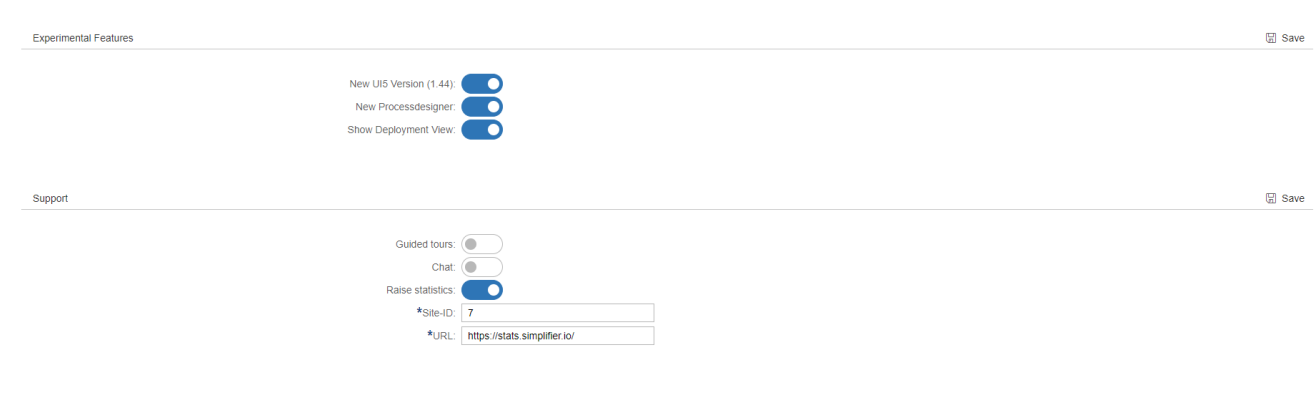
Within the **SMTP Settings**, you can define the email settings.



Send E-Mails	
SMTP Host	Here you define the outgoing mail server, for example smtp.gmail.com
SMTP Port	Specify the port of the outgoing mail server, for example 587
Authentication required	If activated, a user/password authentication from the outgoing mail server is required.
Encrypted with TLS	If activated, the transmission is encrypted by TLS.
Sender E-Mail	Enter the sender's email address, for example no-reply@simplifier.io
Sender	Specify the name of the sender to be displayed.
Password	Select a secure password.

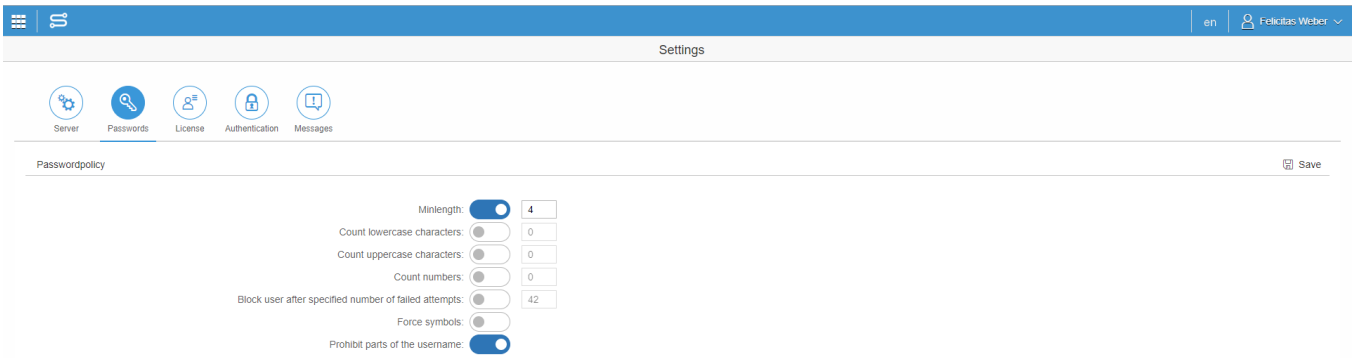
You can switch on **Experimental Features**. It's not guaranteed that these features work propery.

At the subtitle **Support** you can also activate guided tours and the chat function. If you want to raise statistics, you can also activate it.



Passwords

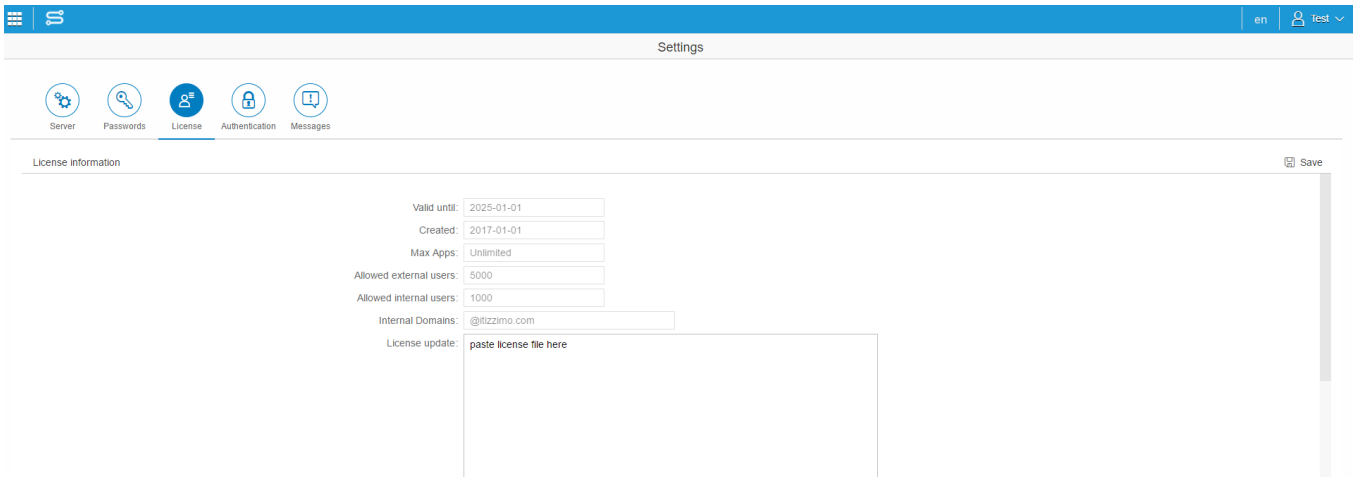
The **Password settings** specify the password policy for your users.



Minlength	Minlength defines the minimum length of a password. Set a higher number for added security
Count lowercase characters	This defines the minimum count of lowercase characters in passwords. For example, setting it to 4 means that a password has to be “abcdEFG” (at least four lowercase characters)
Count uppercase characters	This defines the minimum count of uppercase characters in passwords. For example, setting it to 4 means that a password has to be “ABCDefg” (at least four uppercase characters)
Count numbers	This defines the minimum count of numbers in passwords. For example, setting it to 4 means that a password has to be “1234abc” (at least four numerical characters)
Block user after specified number of failed attempts	This prevents Brute Force Attacks. You can specify a limit for failing logins. If the user exceeds the limit he will be blocked. To unlock the user, the admin has to do this via the user management or the user can reset his password to unlock himself.
Force symbols	This forces the user to use at least one symbol, such as $%&#
Prohibit parts of the username	This option prohibits users from using their username or parts of it for their password. For example, the user “John” cannot use a password like “John123”

License

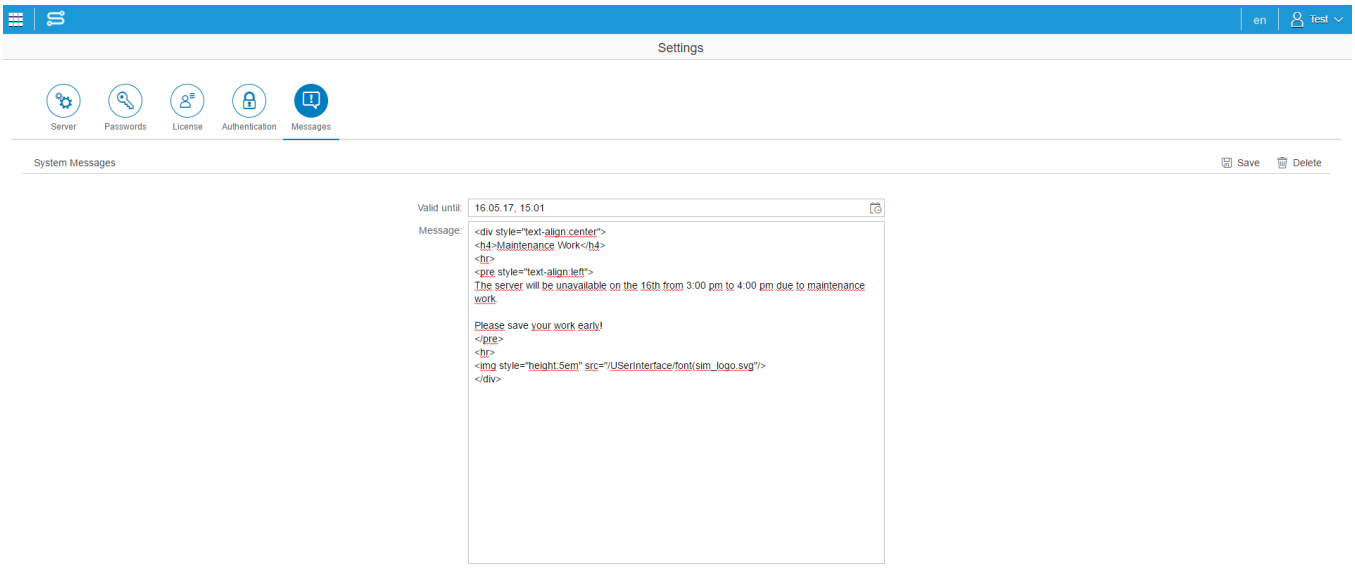
The **License settings** offer information about the current license you use. If your license expires, you can update it here.



License Settings

Messages

The **Message settings** allow you to store system messages, which are transmitted to all logged in users, e.g. if you want to inform them about maintenance work.
With corresponding rights (assigned Role “System Messages”), you can add a validity date. The messages can be written in HTML.



Settings Messages

As soon as a message is stored, it is pushed to all user. User who log in later receive the message as well. The message is displayed as an overlay and must be closed manually.

Simplifier Client API

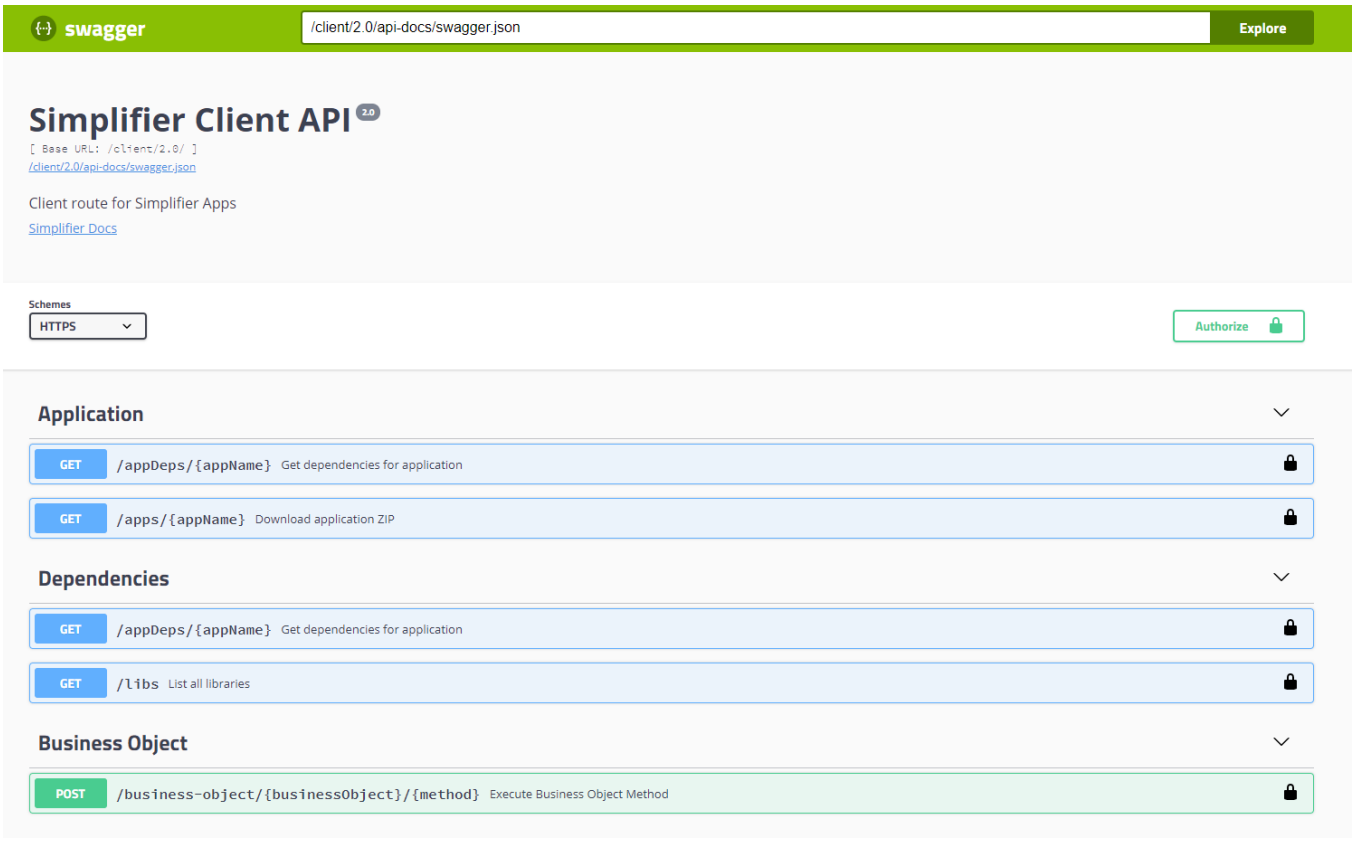
All external programs, business applications and the mobile client access the Simplifier via the Simplifier Client API.

The Simplifier Client API is documented by Swagger and can be accessed by opening the URL:

`https://<pcmr.simplifier.io>/client/2.0/api-docs/`

A detailed documentation of the client API is displayed and you can try out the API using the tools built into Swagger. All changes in the API are automatically transferred to the documentation.

Note: Make sure that you have select HTTPS under schemes.



Application

GET	/appDeps/{appName}	Get dependencies for application
GET	/apps/{appName}	Download application ZIP

Dependencies

GET	/appDeps/{appName}	Get dependencies for application
GET	/libs	List all libraries

Business Object

POST	/business-object/{businessObject}/{method}	Execute Business Object Method

Execution

POST	/business-object/{businessObject}/{method}	Execute Business Object Method
POST	/connector/{connector}	Execute Connector
POST	/connector/{connector}/{call}	Execute Connector Call
POST	/pluginSlot/{plugin}/{slot}	Execute Plugin Slot

Connector

POST	/connector/{connector}	Execute Connector
POST	/connector/{connector}/{call}	Execute Connector Call

Download

GET	/apps/{appName}	Download application ZIP
GET	/libs/{libId}	Download library ZIP

Library

GET	/libs	List all libraries

Mobile Client

POST	/log/MobileClient	Post general message to simplifier logger
POST	/log/WebView	Post WebView message to simplifier logger

Example

GET	/ping	Get default ping/pong message
POST	/ping	Get custom ping/pong message

Plugin

POST	/pluginSlot/{plugin}/{slot}	Execute Plugin Slot

Simplifier

GET	/version	Get version of the currently running

Simplifier app server

Simplifier Cloud

The Simplifier Cloud is the default deployment option when you get started with the Simplifier. As an integrated solution, the Simplifier Cloud includes automatic backups, monitoring, high availability, and more. The availability is at least **99.95 %**.

The Simplifier Cloud will be deployed in Deutsche Telekom’s data centers OTC. Data protection and data security are a matter of trust.
The protection and security of your data has the highest priority. The Open Telekom Cloud is operated and data is secured in our own, highly secure twin-core data centers in Germany. Data processing is subject to the strict requirements of the European Data Protection Ordinance (DSGVO).

In addition, the Open Telekom Cloud is certified according to the Trusted Cloud Data Protection Profile (TCDP) 1.0. This certifies that the Open Telekom Cloud – currently one of the few cloud offerings on the market – offers companies the technical requirements necessary to meet the basic European data protection requirements (DSGVO).
Further information can be found here.

Simplifier Cloud SLA

You will find the SLA of the Simplifier Cloud below:

1. The Licensor operates the provided Platform under the criteria of highest possible care, reliability, and availability of 99.7%, 7 days a week, 24 hours a day, 365 days a year.
   Downtimes are not considererd in the calculation of availability if the Licensee expressly agreed to it in writing or if these downtimes are default times mentioned in paragraphs 4 and 5.
2. The Licensee has the right to reduce the agreed monthly fee for the maximum of 2% for every full percentage point that is below the assured availability. Any further claims of the customer remain unaffected.
3. Down times for maintenance, servicing and upgrades are explicitly pre-announced in writing between the parties, if these deviate from the default times mentioned in paragraphs 4 and 5.
4. The maintenance of the Simplifier is carried out by default on Saturday from 22:00 to 24:00 CET (Central European Time). Divergent or in addition necessary maintenance must be announced to the Licensee in writing, at least four weeks in advance and require the Licensee‘s written approval.
5. Every six months a release upgrade will be installed, namely on Saturday of the month’s last weekend, unless the contracting parties agree in a specific case for a different arrangement.
6. Updates and upgrades, that change the process, have to be announced to the Licensee in writing and before the implementation and will require written approval.

SOAP Connector Calls

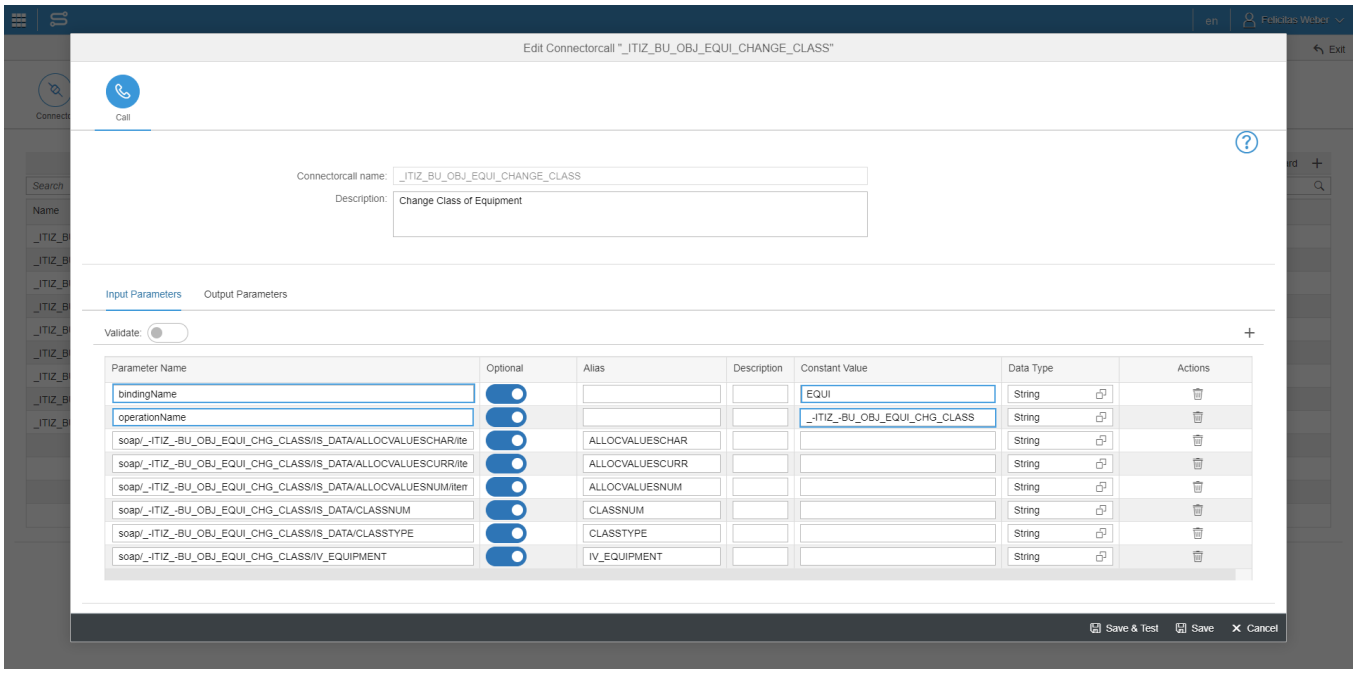
SOAP is a network protocol. You can either stick to the standard when using the protocol like SAP or use it manually like Microsoft. The way the protocol is used decides on how you have to build your Connector Call.

A) Stick to the standard protocol

In this case, the SOAP Connector Call requires two input parameters to be defined:

bindingName	The name attribute of your wsdl:binding attribute in your WSDL file
operationName	The name of the wsdl:operation attribute in your WSDL file for the operation that you want to call in your connector call

Example:



Parameters, that are required by the called SOAP operation, take the following form:

soap/<operationName>/<parameterName>, e.g. soap/_-ITIZ_-BU_OBJ_EQUI/index

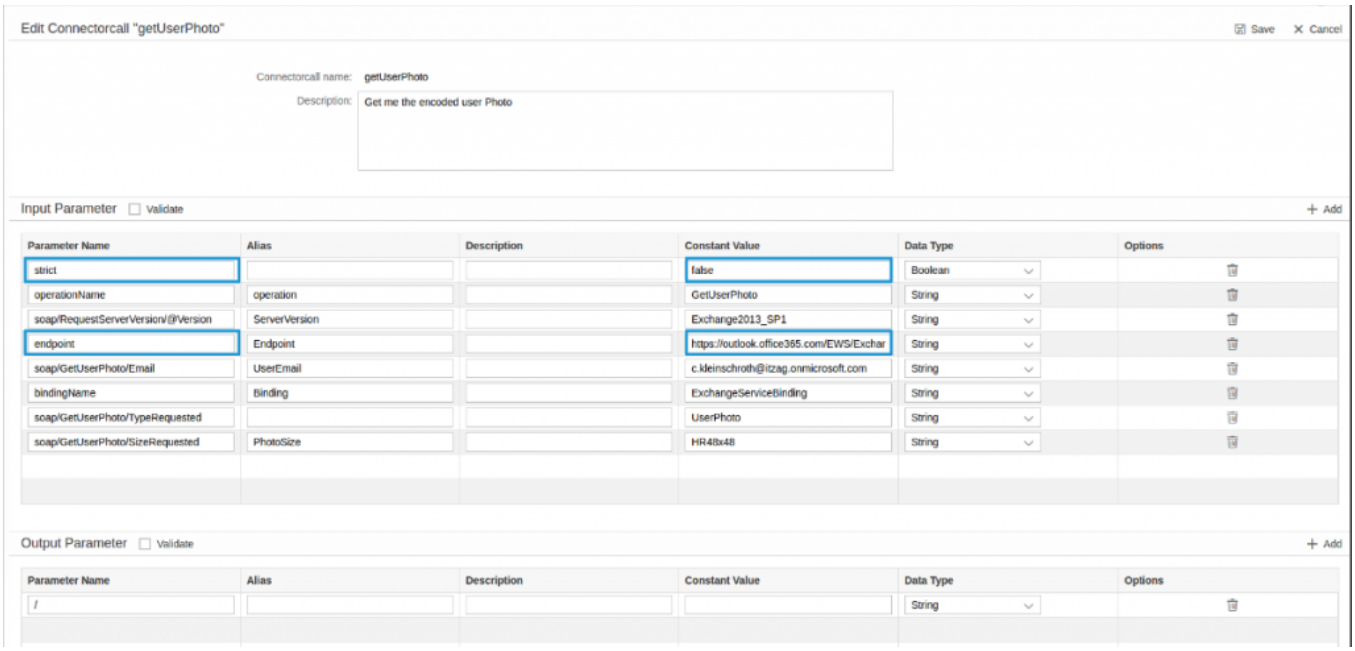
---

B) Use the protocol manually

In this case, the SOAP Connector Call requires two additional input parameters to be defined:

bindingName	The name attribute of your wsdl:binding attribute in your WSDL file.
operationName	The name of the wsdl:operation attribute in your WSDL file for the operation that you want to call in your connector call.
strict	If you don’t use the standards, you have to set this parameter. The Data Type is “boolean”. Set the value to “false”.
endpoint	By adding this as parameter, you can choose the endpoint manually. If the endpoint is not set, the endpoint which is defined in the WSDL file will be used.

Example:



All possible nodes (even if marked as optional) have to be defined in the Connector Call. They take the same form as in the standard use of the protocol:

soap/<operationName>/<parameterName>, e.g. soap/Query/index

Inline attributes are characterized like this:

soap/<xml-node>/@attribute, e.g. soap/RequestServerVersion/@Version

---

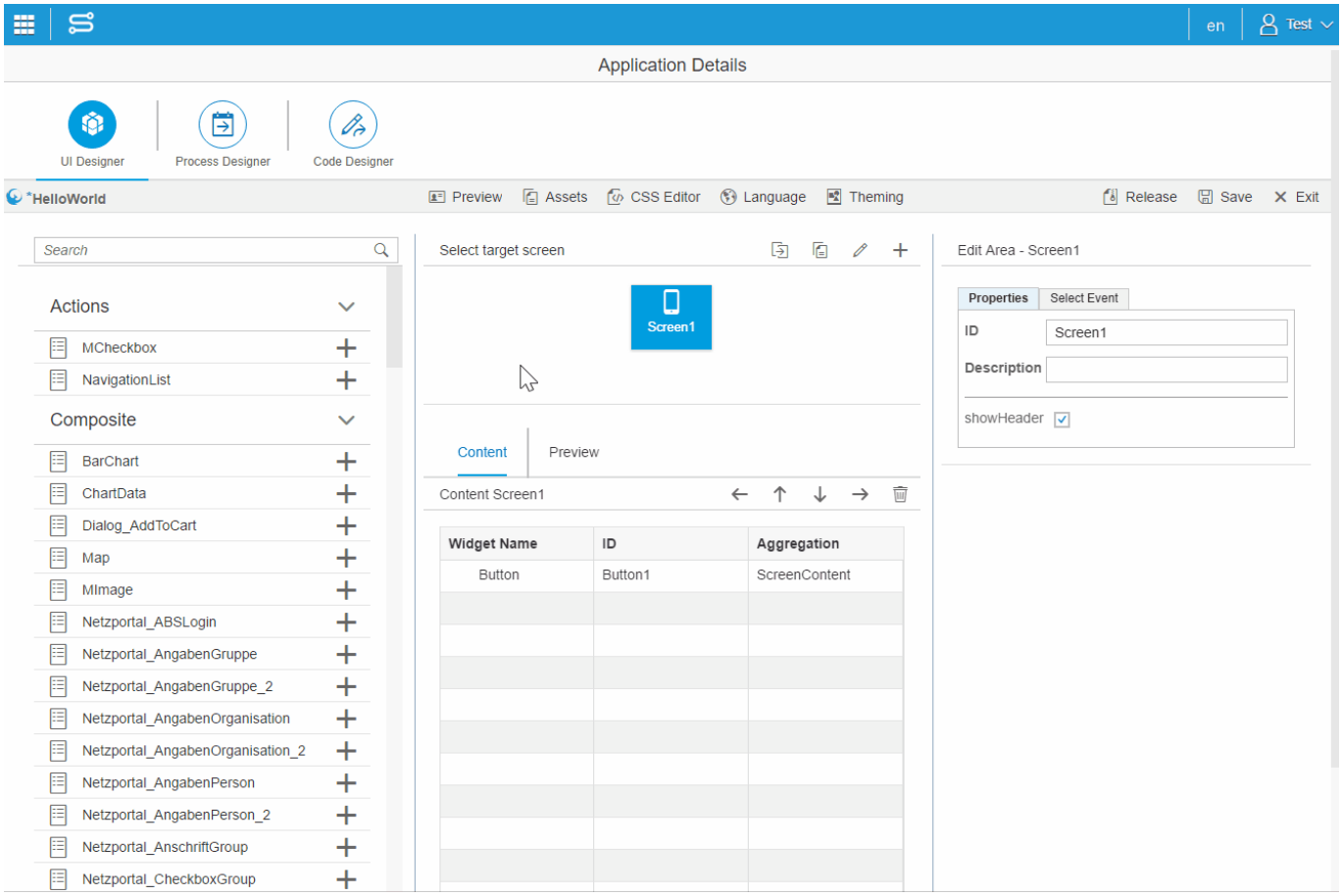
SOAP Connector Details

WSDL URL

The URL to the Web Service Description Language Document (WSDL) of the SOAP Service. Please don’t mix up with the SOAP Service Endpoint.

Sort Widgets in the right order

Widgets can also be nested in a structured hierarchy – representing the positions within the document structure. For example, a table must have columns and rows – so the columns and rows have to be added nested underneath the table element. So in the first step, add the table to the existing screen. After that, select the table and then add columns.



The column widget is now nested under the table widget.2

SQL Connector Calls

A SQL Connector Call requires the two parameter “**mode**” and “**request**“.

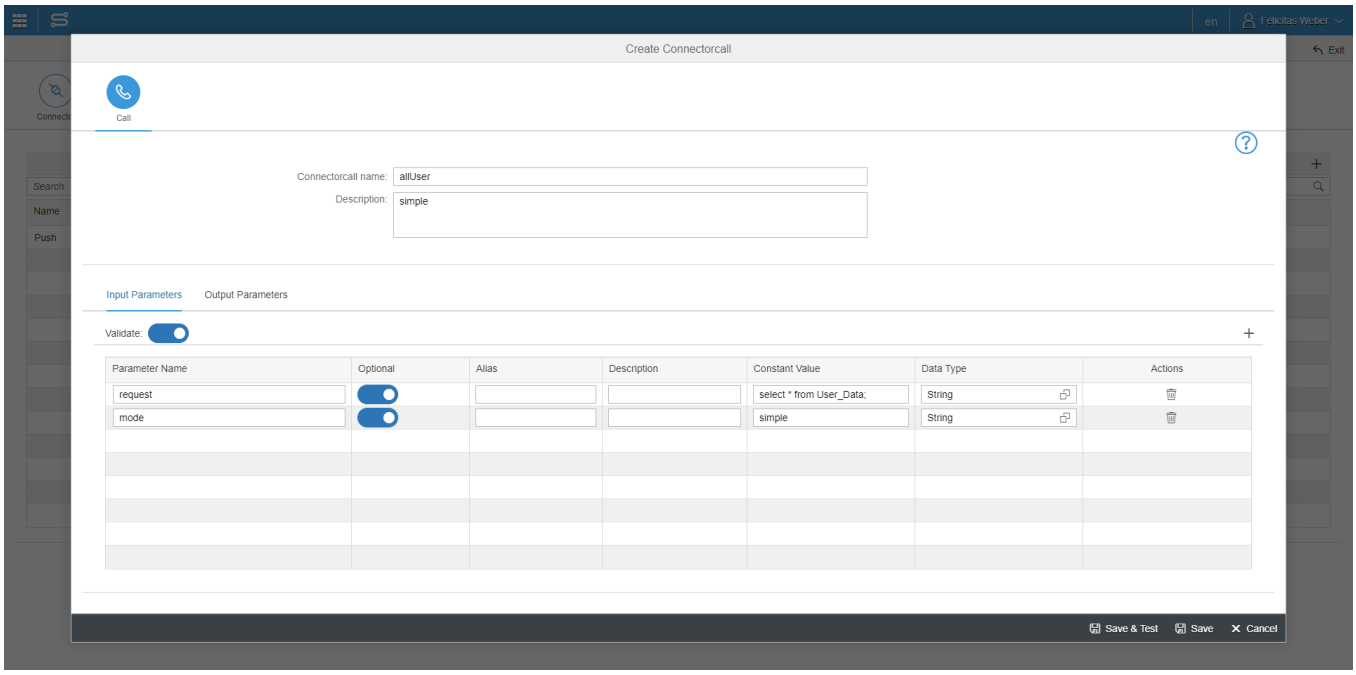
You can choose between 3 different modes:

- simple
- query
- execute

Simple Call

The “Simple Call” corresponds to an unparameterized SQL Call.

The SQL request is defined by a “String” without variable substitution.



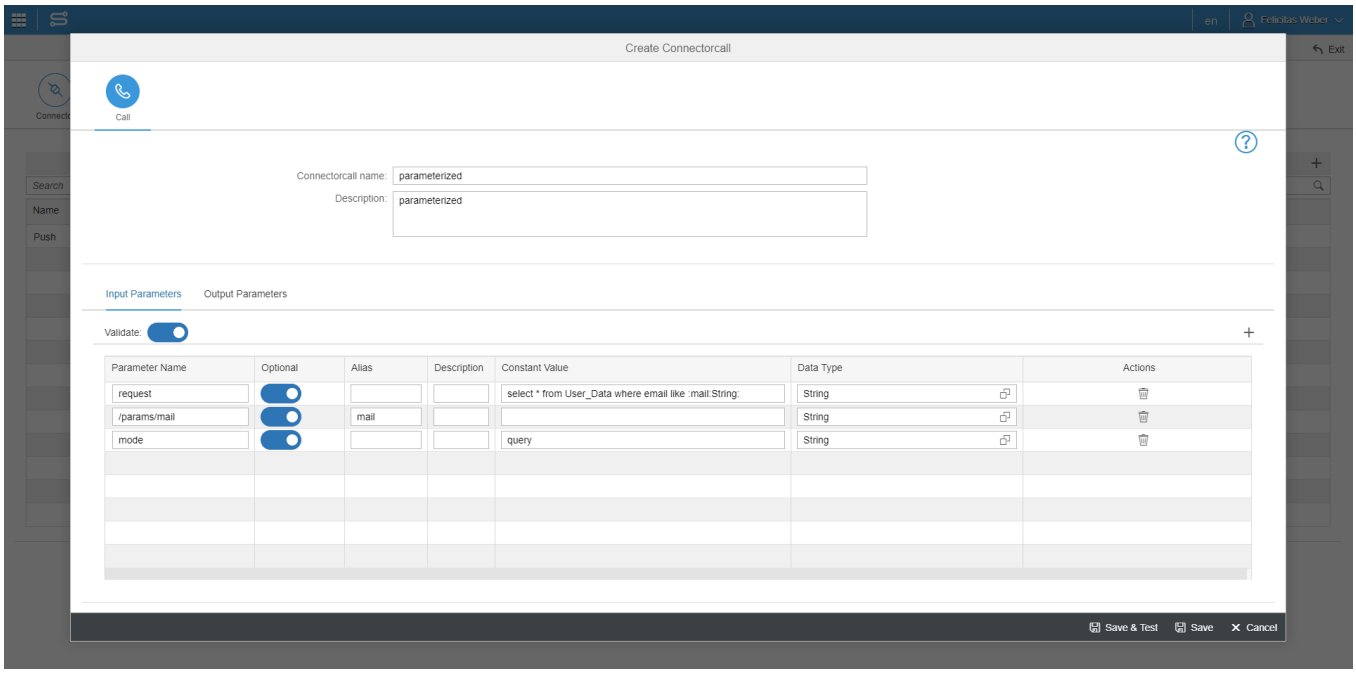
In order to be able to process the output of the SQL request, it must be defined in the Output Parameter of the Connector Call.

Currently, you can still map the entire Json result to a single parameter.
This is possible by specifying a “/” as the parameter name and the Data Type “String” in the Output Parameters.

Query Call

The “Query Call” is used to transfer the result from a SQL statement to the Simplifier.
(Usually, SELECT statements have a result)

The SQL request of the “Query Call” is parameterized (in contrary to the “Simple Call”).



Definition of the constant value for the request parameter

The Input value of the request parameter is represented by a name and a Simplifier Data Type. In our example, we used the name “mail” and the Data Type “String”.

A parameter definition is initiated and ended with a colon. Between the colons is a pair of values, separated by a colon as well, which represents the name and the type of the parameter.

In a parameterized SQL request: select * from USER_Data where email like :mail:String:

If you use more values in a SQL statement you have to use a syntax separated by comma, e.g. VALUES (:id:String:,:mail:String:).

Definition of the parameter in the Connector Call

In order to use the parameter that has been defined in the SQL request via the Connector Call interface, a new Connector Call parameter must now be inserted, which corresponds to the parameter definiton of the SQL request.

All Connector Call parameters that refer to a SQL request have the prefix “/params/”.
In the example above, a new parameter name “/params/mail” with the Simplifier Data Type “String” has been created.

The parameter expressions “alias”, “description” and “constant value” are equivalent to the usual connector calls.

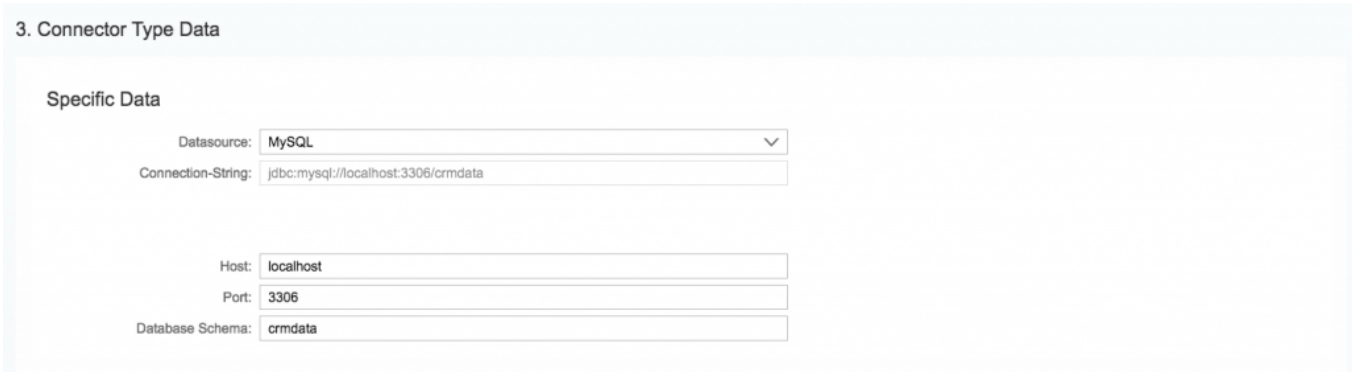
The result of a “Query Call” can be assigned via the Output Parameters equivalent to the “Simple Call”.

Execute Call

The “Execute Call” is quite similar to the “Query Call”. But with the “Execute Call”, **no** result is sent back to the Simplifier.

Typically SQL INSERTs, UPDATEs, or DELETEs are displayed in that way.

SQL Connector Details



Datasource

The JDBC Driver of the supported Database: MySQL, Sybase, Oracle, PostgreSQL, SQLite, HANA and Microsoft SQL

Host

Hostname of the database server

Port

TCP Port of the database server

Database Schema

Name of the Database

If you have created an SQL connector with the SQLite datasource, files are created for a special directory for SQLite data files. The Connector creates an SQLite database on the Simplifier Server in the Connector Properties if it’s not available. When changing the connector, a backup of the file is created.

Attention: Deleting the connector also deletes the data file.

Start PDF Generation

Start PDF Generation

URL	/client/1.0/PLUGIN/pdfPlugin/generatePdf	
Input-Parameter	**Template**	Template name
	Session	Session ID (to retrieve transaction data)
	Config	PDF parameter for the generation as JSON String (optional)
Output-Parameter	**Value**	JSON Object with the parameter "**jobId**" (contains the generated JobID)

Example for a call:

```
{     "template": "templatename",     "session": "12345678910",     "config": "{\"ori
entation\" : \"Portrait\",\"page-size\" : \"A4\",\"margin-top\" : \"1in\",\"margin-bo
ttom\" : \"1in\",\"margin-left\" : \"1in\",\"margin-right\" : \"1in\",\"footer-
center\" : \"[page] / [toPage]\"}\"}" }
```

Output example:

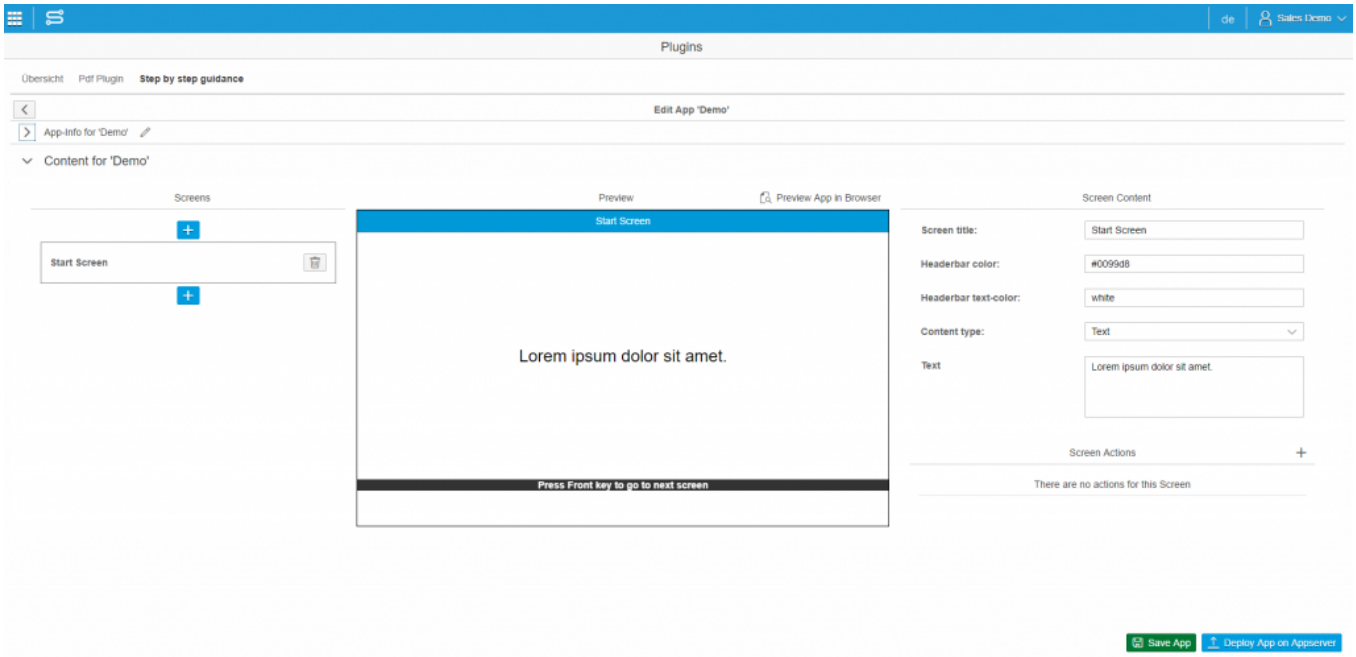
```
{     "value": {         "jobId": "a1ef6b46-1671-425a-947c-
d9e552d4e755"     },     "success": true }
```

The specified parameters for the config correspond to the parameters for wkhtmltopdf or the PdfConfig class of the wrapper. This can be used to control page formats, margins, headers, footers, etc.

Step by Step Guidance

The Step-By-Step Guidance Plugin allows you to build simple apps consisting of one information (text, image, video) per screen and one of the predefined screen actions. The Plugin is useful if you want to create a qick Apps for maintenance, strictly defined processes or quality assurance.

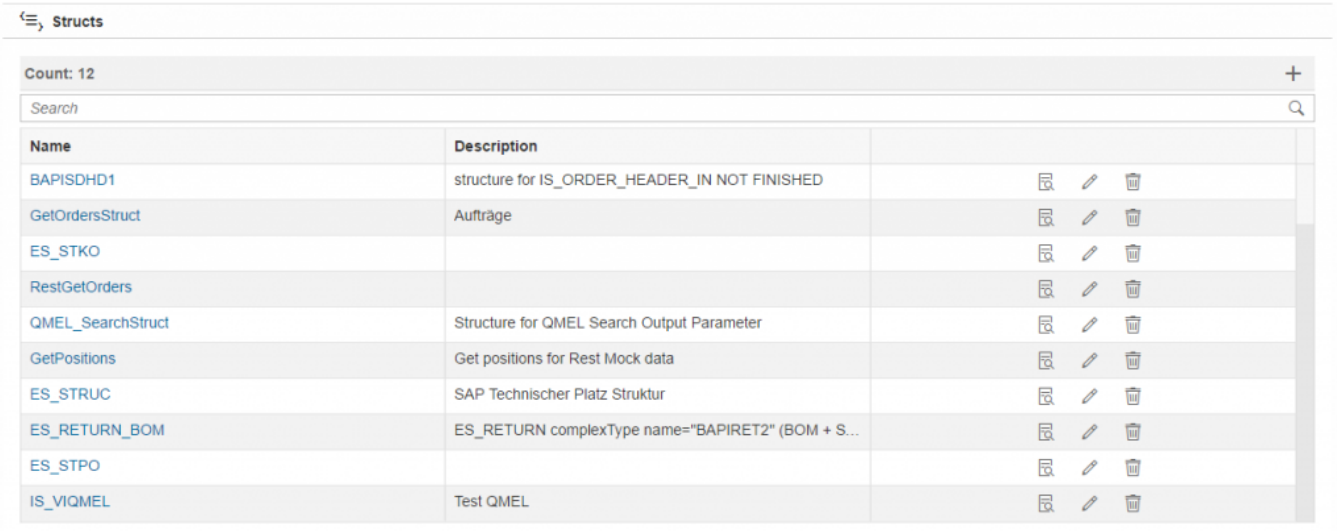
By clicking on the Button “Deploy App on Appserver”, you can open the App with the UI Designer and customize it further.



Struct Type

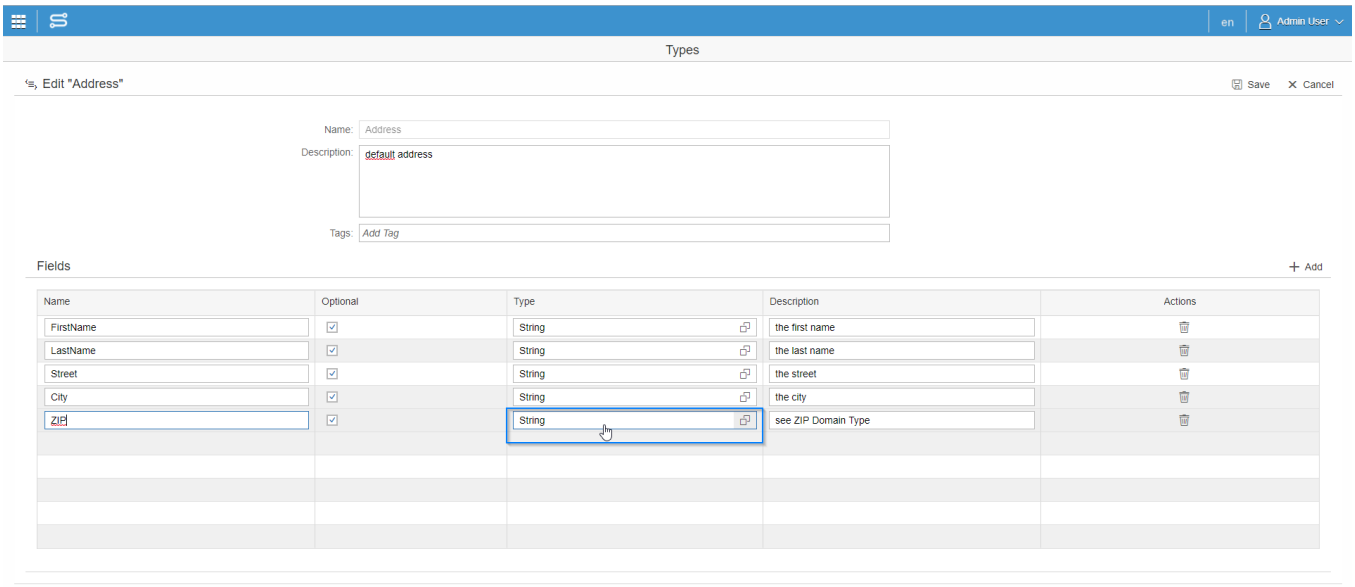
Structs describe a package of Domain Types. For example the Struct “address” contains different Domain Types like Name, Street, City, ZIP Code, Mobile Number etc.

To create a new Struct Type click on the “+” button.

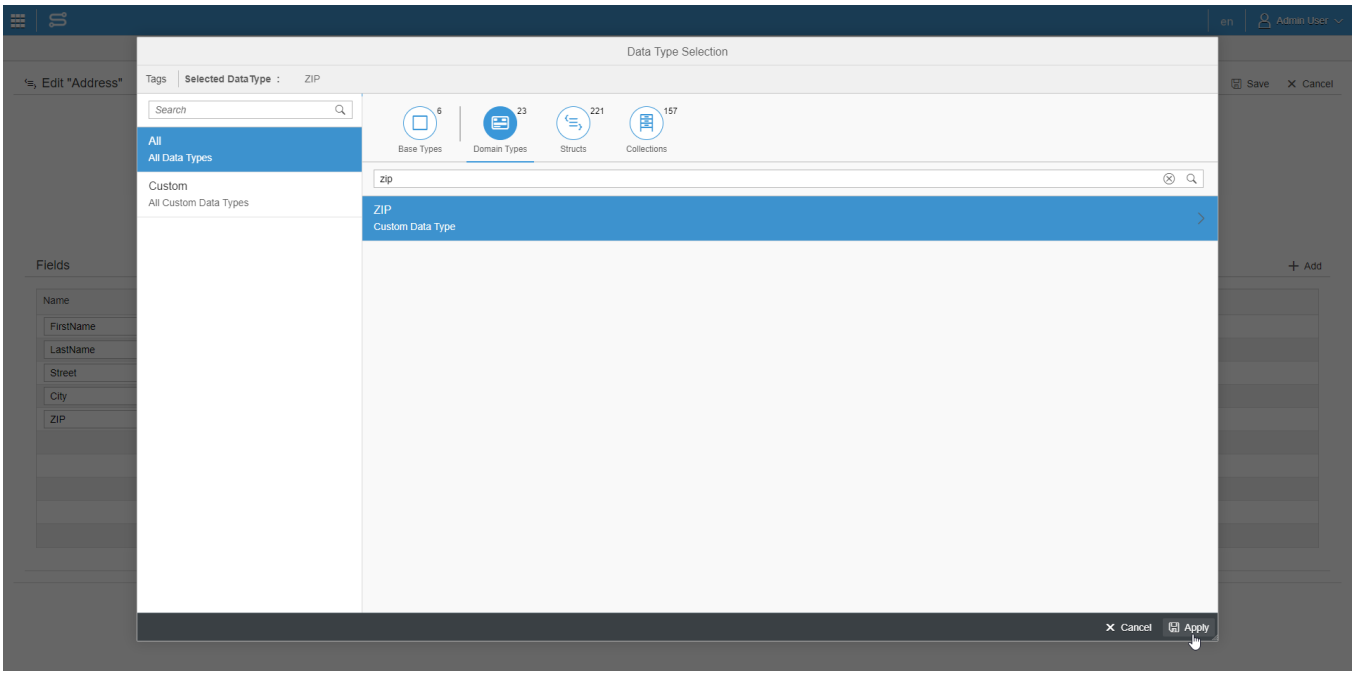


You can define a unique struct name:

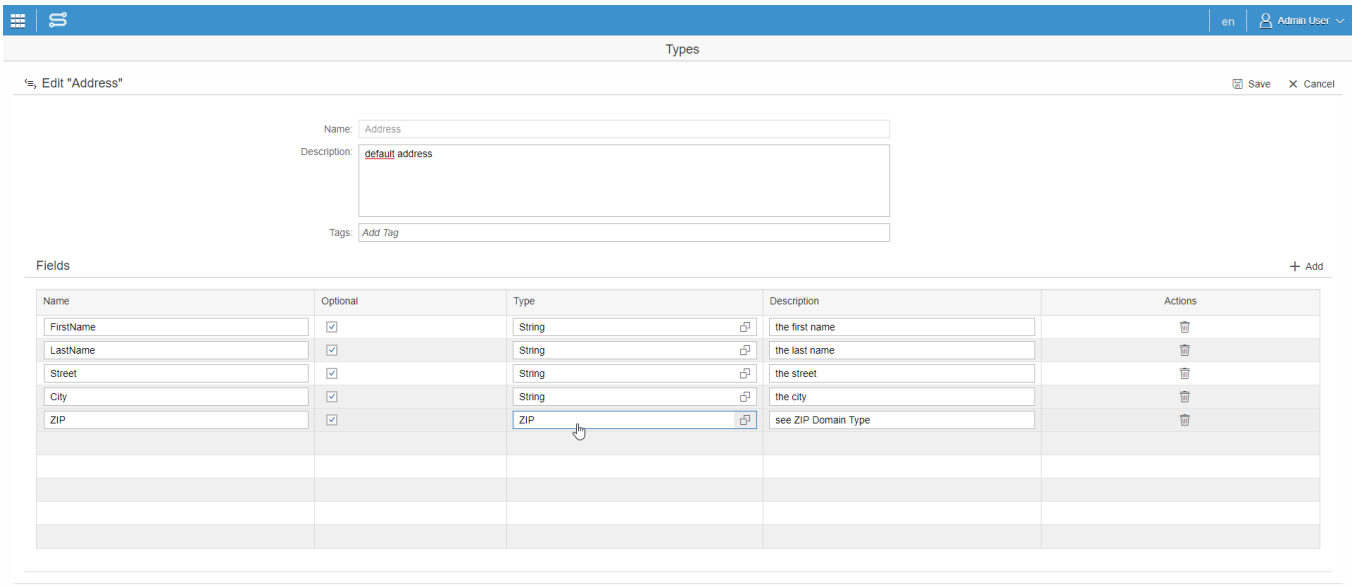
Click into the appropriate field to select the type.



Click e.g. on Domain Types on the top and search for the domain type in the searchbar.



After clicking on apply, the type will be applied.



In the fields table can you add the following Data Types:

Name

Name of the field in the structure.

Optional

With this setting it isn't necessary to provide all fields in Connector or Business Object Calls.

Type

Base or Domain Type or even another structure or collection.

Description

Description of the field.

System Connectors

This area covers all information related to specific Backend Connectors for existing Systems (e.g. SAP, Salesforce, etc.)

Technical call of a PDF Plugin

Administration and Generation

You can manage your templates or start a new creation via REST.

All calls are started via a POST request, which may contain parameters in JSON format in the body.
The return is always in JSON format. It includes the result parameter “success” to indicate whether the call was successful.

If the call is not successful, an error code (as a return parameter “code”) and an error message (as a return parameter “message”) are returned.

The following errors are possible:

Error Code	Message	At Request
1	template name invalid	adminTemplateFetch, adminTemplateEdit, adminTemplateDelete, generatePdf, adminTemplateAdd
2	template not existing	adminTemplateFetch, adminTemplateEdit, adminTemplateDelete, generatePdf
3	template name already in use	adminTemplateAdd
400	bad request	All
500	internal error	All

(Errors 400 and 500 correspond to the HTTP status codes and indicate errors when parsing the parameter or processing errors)

The sub-sites describe the merging and saving of PDFs as well as the different requests to manage your templates.

Technology

The Simplifier uses the following standard web-technology for creating applications:

SAPUI5 and its open-source variant OpenUI5 help you build enterprise-ready Web apps that are responsive to all devices. The JavaScript UI library and development toolkit contains many feature-rich controls and implements the award-winning SAP Fiori user experience. It helps developers ease and speed up the development of full-blown HTML5 Web applications.

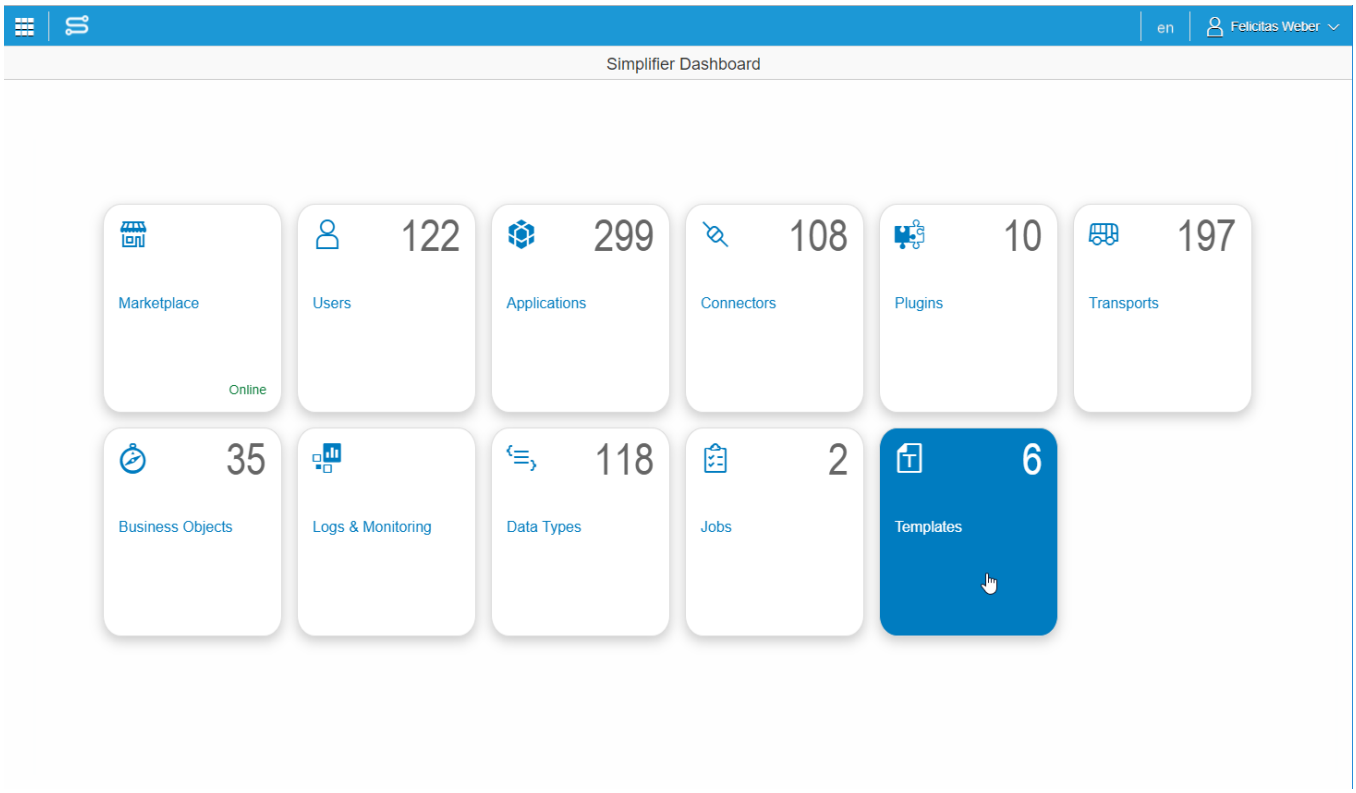
We implemented the Update to Angular 2 in the Simplifier. Now you can configure the UI and logic of your Angular app the same way as with UI5.

Angular 2 allows the creation of extremely versatile, fast, powerful and functional web applications. These applications run in the browser and thus bring the very reactive and therefore great user experience of mobile apps to the desktop or browser.

With the integrated Ionic framework, you can build highly interactive native and progressive web apps.

Templates

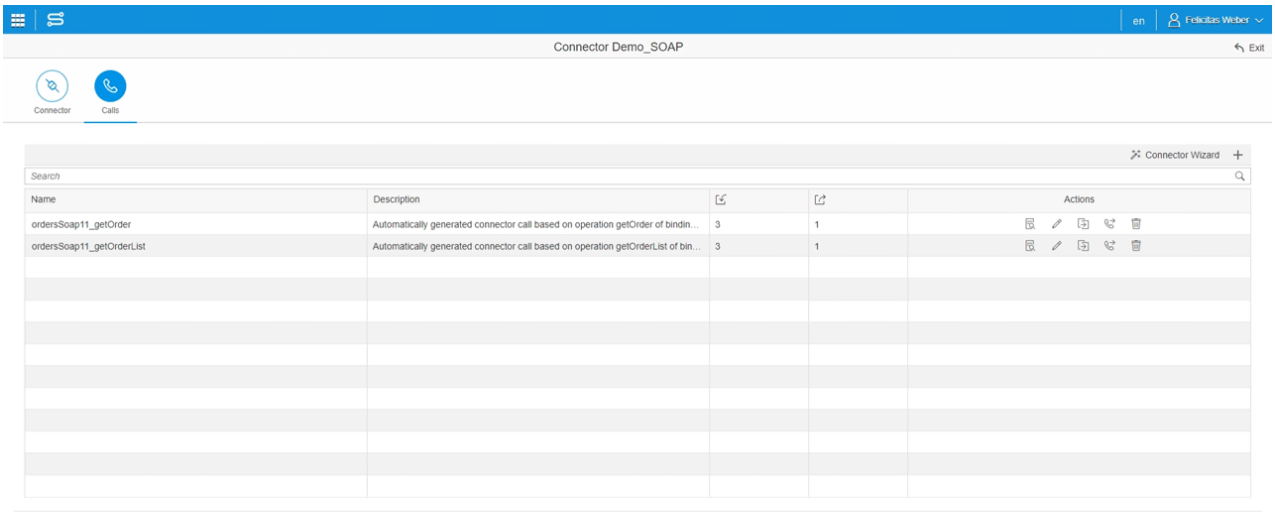
In the new Tile you can create, edit and delete Namespaces and Templates.


en
Felicitas Weber
Simplifier Dashboard
Marketplace
Online
122
Users
299
Applications
108
Connectors
10
Plugins
197
Transports
35
Business Objects
Logs & Monitoring
118
Data Types
2
Jobs
6
Templates

Test a Connector Call

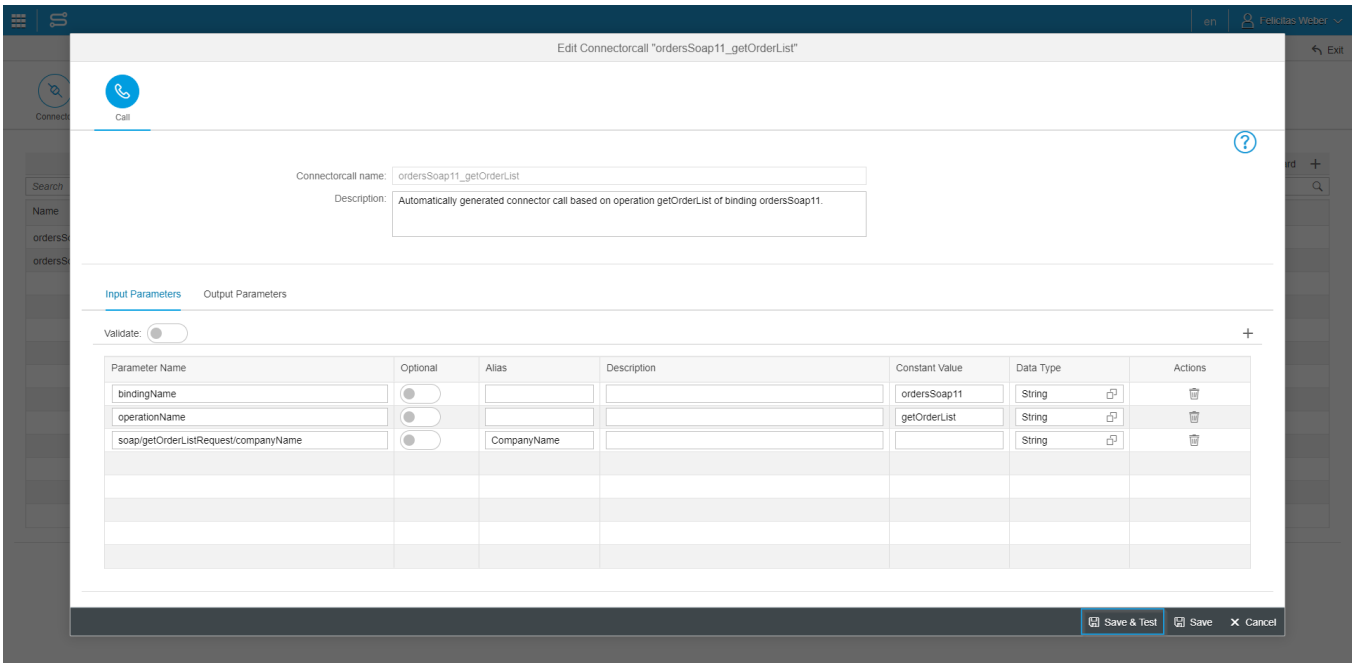
As an admin, you can test your connector call by clicking on the “Call” icon in the connector call overview.

- The upcoming dialog provides an input field for all input parameters, that do not have a constant value.
- To see at a glance which data type is configured for the parameter, a column is displayed.
- The “Details” button displays the current configuration of the input parameters of the call.
- The “Test” button runs the call and displays the result in JSON format.



The input validation in the test UI only takes place if “Validate” is active for input parameters. The test parameters are always passed as strings.

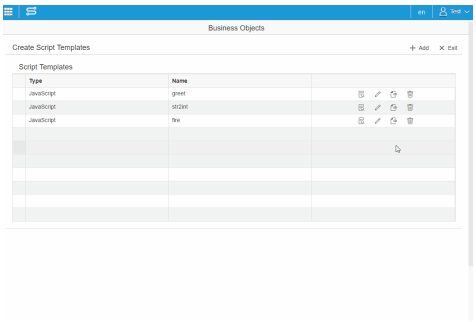
There is also a “Save & Test” button within the configuration of the connector call itself. The same dialog is opened as above.


Edit Connectorcall "ordersSoap11_getOrderList"
Call
Connectorcall name:
ordersSoap11_getOrderList
Description:
Automatically generated connector call based on operation getOrderList of binding ordersSoap11.
Input Parameters
Output Parameters
Validate:
Parameter Name
Optional
Alias
Description
Constant Value
Data Type
Actions
bindingName
ordersSoap11
String
operationName
getOrderList
String
soap/getOrderListRequest/companyName
CompanyName
String
Save & Test
Save
Cancel

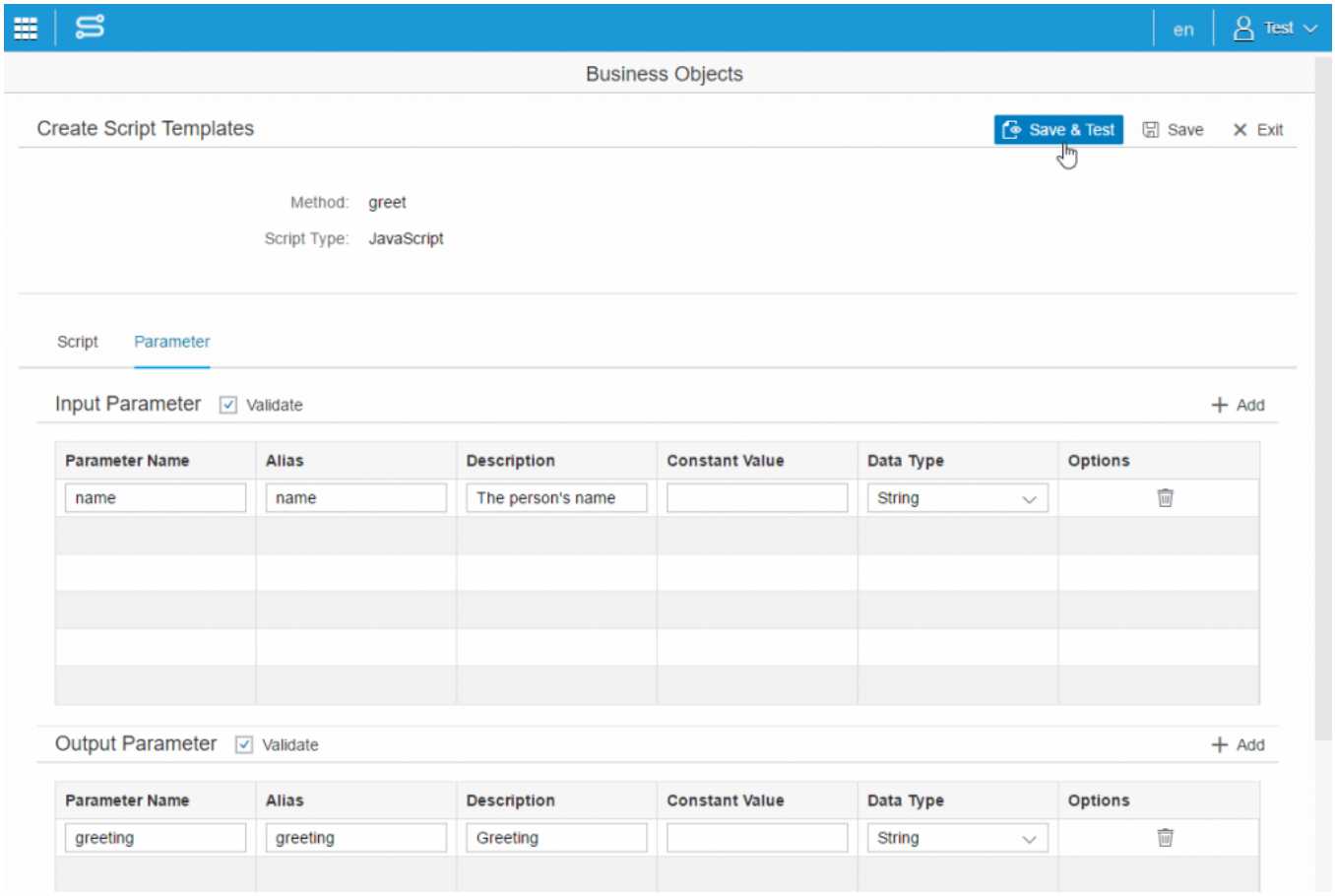
Test Business Script Templates

As an Admin, you can test your Script Template equivalent to the test of a Connector Call. Click on the “Test” icon in theScript Template Overview.

- The upcoming dialog provides an input field for all input parameters, that do not have a constant value.
- The “Details” button displays the current configuration of the input parameters of the Template.



- There is also a “Save & Test” Button within the configuration page of the Template itself.



- The “Test” button runs the Template and displays the result in JSON format

Testing WebSocket Connection through Reverse Proxy

Testing WebSocket Connections through Reverse Proxy

The WSS Protocol is used for realtime data communication and server to client push mechanism.

To test a sucessfull connection , following the steps below

1. Login via /UserInterface/ to the Administration Interface+
2. Set a System Message like “Web Connection Test” following this documentation (Chapter Messages)
3. Logout
4. Login again into the Administration UI via the Reverse Proxy and you should see the message pushed by Websocket from Server

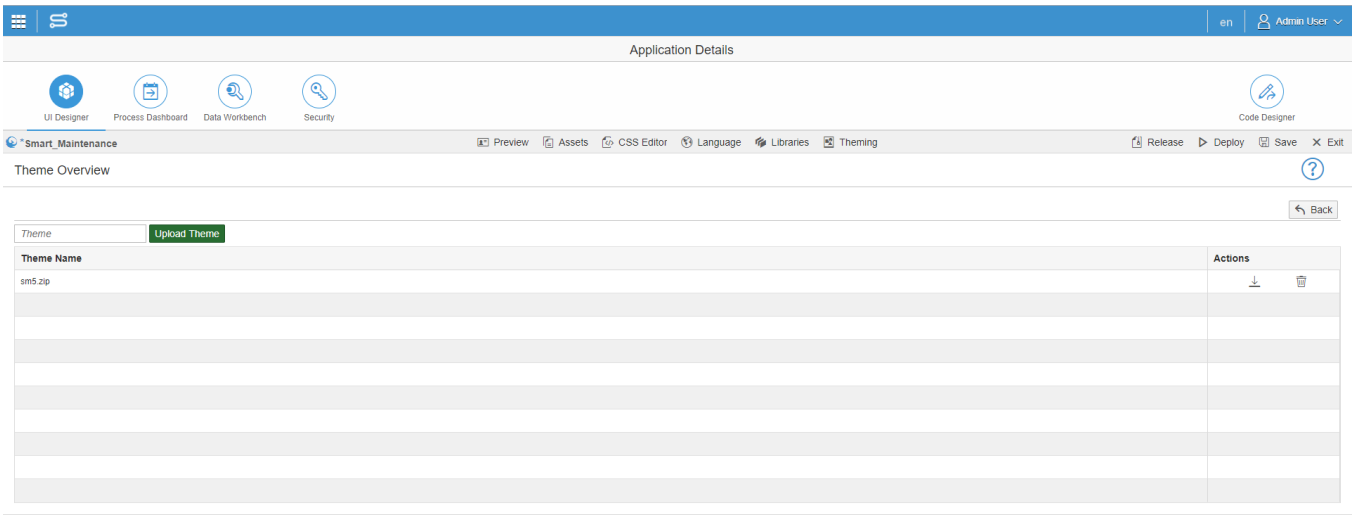
If not, please check the Reverse Proxy Requirements

Theming

You can upload a particular theme to give your app a specific look corresponding to your Corporate Design. To do this, you need a third party less compiler as an external tool.

When the less compiler has run through, you have to put all CSS files in a ZIP file. Then upload it in the UI Designer under Theming.

You have the possibility to download your already uploaded theme again. The icon for this is located to the left of the delete icon.



The directory structure of the theme must be structured exactly like this and is analogous to the OpenUI5 directory structure:

sap / library name / themes / theme name / library.css

Open UI5 offers multiple library options. You can insert all libraries in one file. The paths you can use are:

- sap.m
- sap.tnt
- sap.ui.commons
- sap.ui.core
- sa.ui.dt
- sap.ui.layout
- sap.ui.suite
- sap.ui.table
- sap.ui.unified
- sap.ui.ux3
- sap.uxap

Here you can see details of the Belize Theme.

Token Generation

A token is required for a secure communictation with the application server. You can generate a token with the following AJAX-Request

```
var token = null;
$.ajax("http://localhost:8080/genToken/client/1.0", {
method: 'GET',
headers: {'Authorization': '[Authorization]' // [userName] : [userPassword]},
success: function(data) {
if (!data.result) {
alert("Error retrieving token: " + data.message);
return;
}
token = data.result;
}});
```

[userName]

The name of the user as String.

[userPassword]

The passsword of the respective user as String.

[Authorization]

The authorization data required for identification. Consisting of the authorization method here the String Basic and the [userName]:[userPassword] as a Base64 encoded String.

Token, Websocket and Request Sending Example

This section contains a detailed code example for generating a token and establishing a websocket with a connector, then a complete subscription followed by a complete unsubscription example. The arbitrary chosen connector is the “TIA_Connector”

Token Generation Example

```
var token = null;
$.ajax("http://localhost:8080/genToken/client/1.0", {
method: 'GET',
headers: {'Authorization': 'Basic YWRtaW46YWRtaW4=' // admin : admin},
success: function(data) {
if (!data.result) {
alert("Error retrieving token: " + data.message);
return;
}
token = data.result;
}});
```

Attention

Using the admin as user might be a security risk. In productive systems the admin should be changed into an authorized user with the respective permissions.

Websocket Connection Example

```
var webSocket = null
webSocket = new WebSocket("ws://localhost:8080/client/1.0/connectorAsync/" + "TIA_Connector" + "?SimplifierToken=" +
token);
```

Complete Subscription Example

```
var requestSubscribeData = {
"operation": "MONITORING_SUBSCRIBE",
"nodes": ["myNode","myOtherNode"],
"namespaceIndices": [2,2],
"publishingInterval":1000.0,
"clientHandlingID": 1,
"samplingInterval": 1000.0,
"queueSize": 2,
"discardOldestItem": true,
"monitoringMode": "Reporting",
"returnedTimestamps": "Both"
}
```

```
var requestSubscribe = null;
request = {
"frameType": "unsubscribe",
"subscriptionKey":"bb827118-f1b0-2170-9937-f8c7e1620107"
};
```

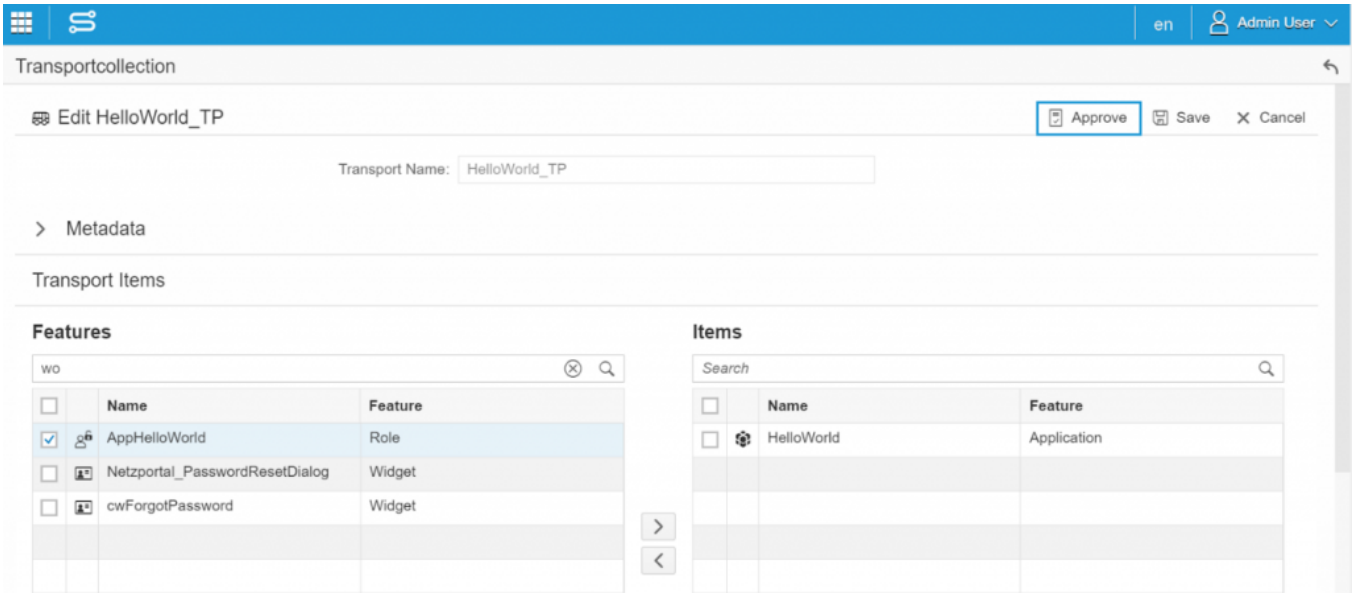
```
var requestSubscribe = null;
requestSubscribe = {
"frameType": "subscribe",
"subscriptionKey":"bb827118-f1b0-2170-9937-f8c7e1620107",
"json": requestData
};

webSocket.send(requestSubscribe);
```

Complete Subscription Example

```
var requestUnsubscribe = null;
requestUnsubscribe = {
"frameType": "unsubscribe",
"subscriptionKey":"bb827118-f1b0-2170-9937-f8c7e1620107"
};

webSocket.send(requestUnsubscribe);
```

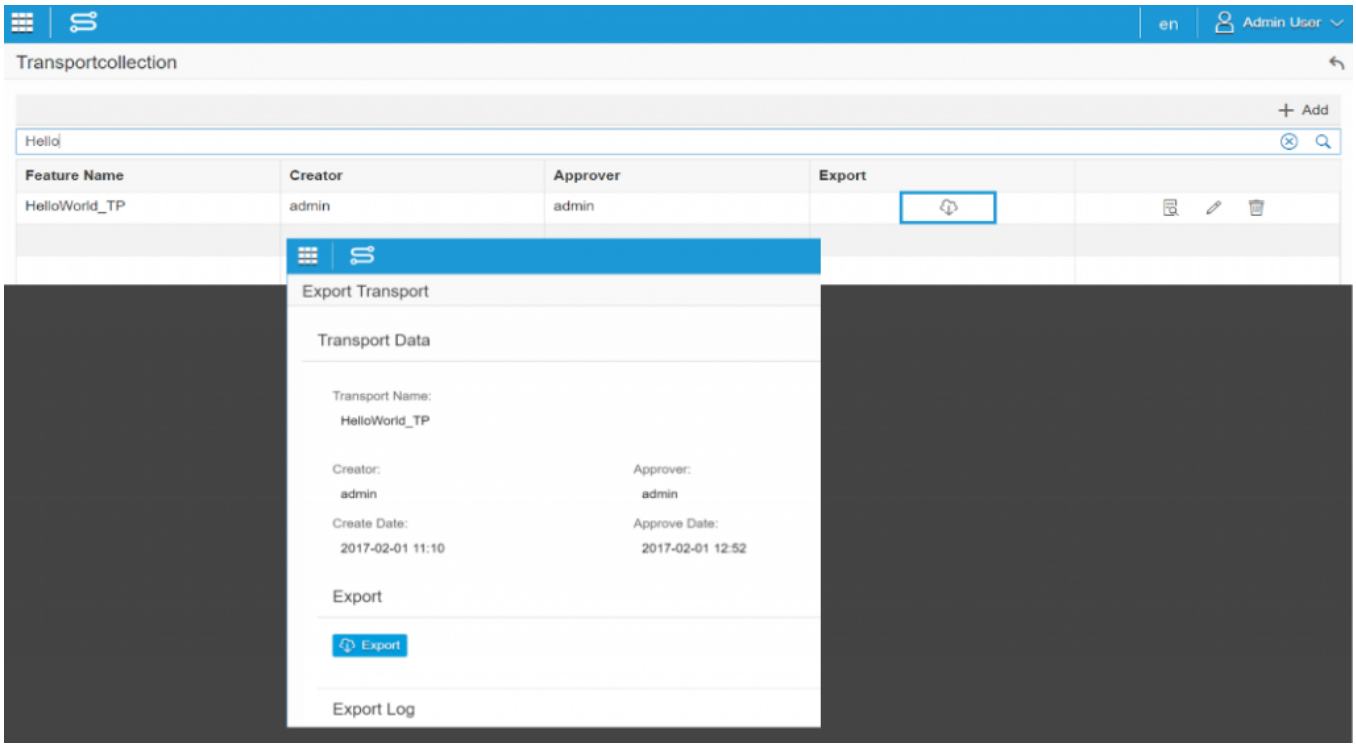
Transport Approval and Execution

Before a transport is available for downloading, it has to be approved by an Admin. With the right permissions, you will find the “Approve” button next to the “Save” and “Cancel” options in the Transport.
Attention: Save the Transport in advance, else your changes won’t appear in the Transport file!



If the approval process was successful, a download icon appears in the Export column and your transport is now ready for downloading. By clicking on the icon, some metadata about the creator, approver and dates are provided.

The downloaded file can be used to import the data on a different system.

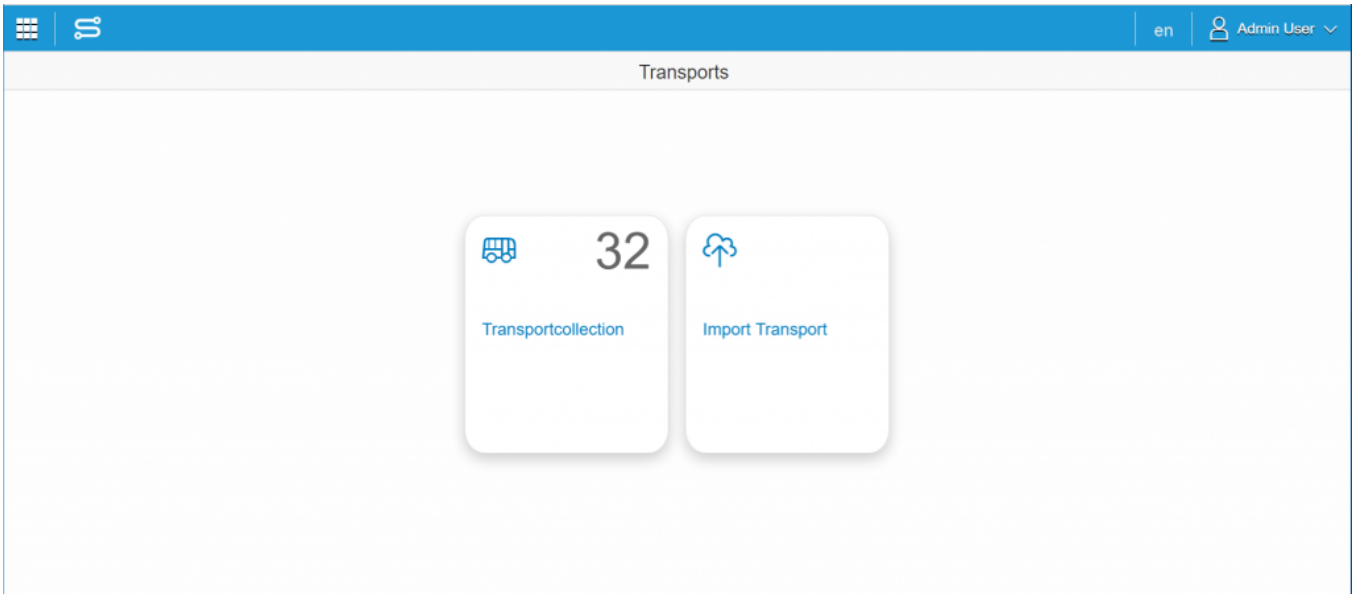

en
Admin User
Transportcollection
Add
Hello
Feature Name
Creator
Approver
Export
HelloWorld_TP
admin
admin
Export Transport
Transport Data
Transport Name:
HelloWorld_TP
Creator:
admin
Approver:
admin
Create Date:
2017-02-01 11:10
Approve Date:
2017-02-01 12:52
Export
Export
Export Log

Transports

Imagine you've build a fancy application for one of your customers on a Simplifier instance that you've reserved exclusively for development tasks. Now that it's ready to roll out, you're faced with the question on how to bring this application finally to the customer's Simplifier instance? How can you export and import it, preserving all its dependencies, like connectors and associated roles?

That's when Transports come into play. They allow you to define all of your app's artifacts and let the Simplifier collect them to create a bundle.

Clicking on the "Transports" tile forwards you to an intermediate page where you can choose between the Transportcollection and importing Transports.



Typical Use-Case PDF Plugin

1. **Create a HTML template for PDF generation, including variables and loops in Mustache format.**
2. **Upload the template with a selected name via the REST interface.**
    1. If you want to correct your template later, change it via the REST interface.
    2. Upload all required images and stylesheets as uploads in the AppServer.
3. **In the App:**
    1. Generation of dynamic data.
    2. Upload the generated dynamic data under a unique session ID in the key-value store (via the REST interface of the KV store).
       The payload needs an unique key and the dynamic data you want to save.

```
var payloadKeyValueStore = {
    key: "sessiondata/myTestApp/" + sap.ui.getCore().getModel().getData().sessionKey,
    content: btoa(JSON.stringify(sap.ui.getCore().getModel().getData()))
};

this.callPlugin("keyValueStorePlugin", "puthttp", payloadKeyValueStore, callbackStoreSession, true, false, onKeyValueStureFail)
```

    3. Optional: Uploading additional documents for merging into the key-value store (via the REST interface of the KV store).
    4. Optional: Uploading a list of these document keys as a merge list in the key-value store (via the REST interface of the KV store).
    5. Start of the PDF generation with the template name, the session ID and (optional) configuration data for the PDF generation via the REST interface.

```
var callbackStoreSession = function(data){
    if(data.success){
        var payloadGeneratePdf = {
            "template": "templatename",
            "session": " myTestApp/" + sap.ui.getCore().getModel().getData().sessionKey,
            //"config": "{\"orientation\" : \"Portrait\",\"page-size\" : \"A4\",\"margin-top\" : \"1in\",\"margin-bottom\" : \"1in\",
            //\"margin-left\" : \"1in\",\"margin-right\" : \"1in\",\"footer-center\" : \"[page] / [toPage]\")\"}"
        };

        busyDialog.setText("Antrags PDF wird generiert. Bitte haben Sie etwas Geduld.");
        $this.callPlugin("pdfPlugin", "generatePdf", payloadGeneratePdf, callbackGeneratePdf, true, false, onGeneratePdfFail);
    }
};
```

    6. Remember the jobID.
4. **Asynchronous execution of the PDF generation (background operation).**
    1. Call the Template.
    2. Call the dynamic data.
    3. Evaluate the data in the template.
    4. Call the static assets (images, stylesheets) from the AppServer.
    5. Convert the finished HTML-file in a PDF.
    6. Get the merge list and the merge binaries.
    7. Merge the created PDF with the merge files.
    8. Save the finished PDF in the key-value store.
5. **Call the key-value store via the REST interface with the restrained jobID to see, if the PDF has already been finished.**

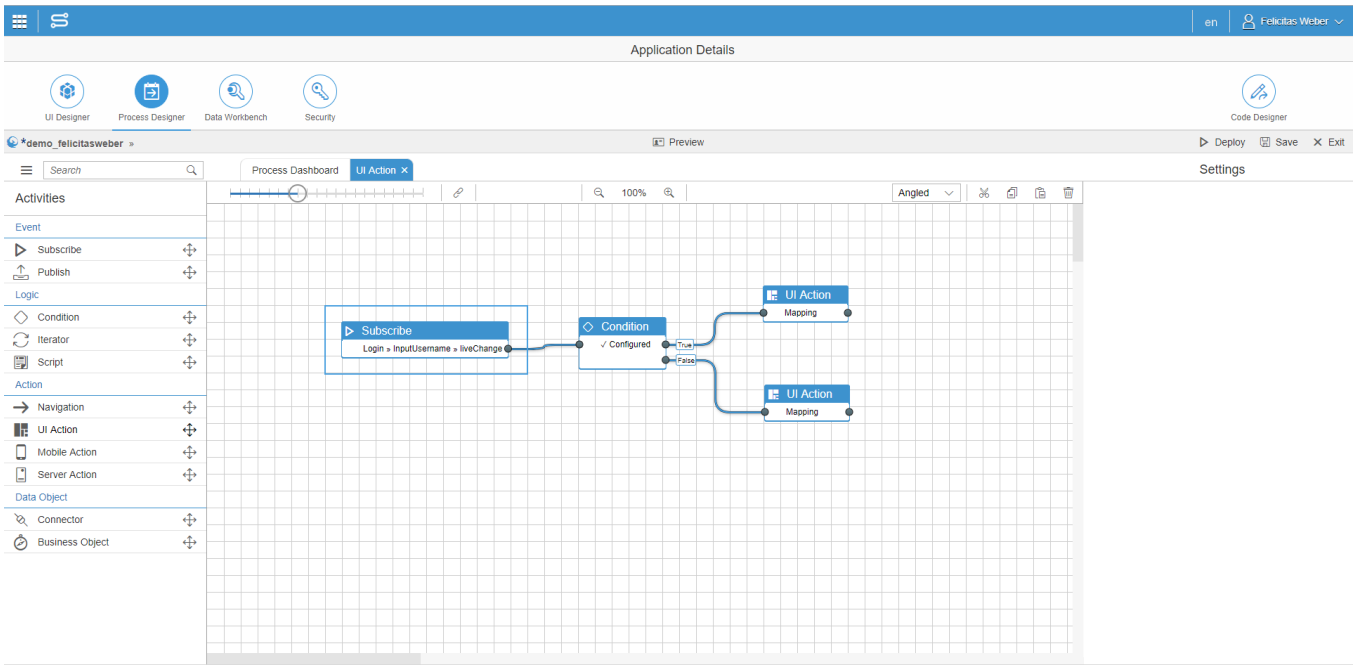
```
var jobId;

var callbackGeneratePdf = function(data){
    if(data.success){
        jobId = {
            key: "pdf/"+data.value.jobId+".pdf"
        };

        //timeout for demo purpose
        setTimeout(function(){
             $this.callPlugin("keyValueStorePlugin", "gethttp", jobId, callbackFetchPdf, true, false, onKeyValueStureFail);
        },10000);
    }
};


var callbackFetchPdf = function(data){
    if(data.success){
        var pdfData = data.result;
        window.open("data:application/pdf;base64,"+pdfData);
    }
};
```

UI Action

With the UI Action element you can map different Widgets, Variables and Auto Fields to another.

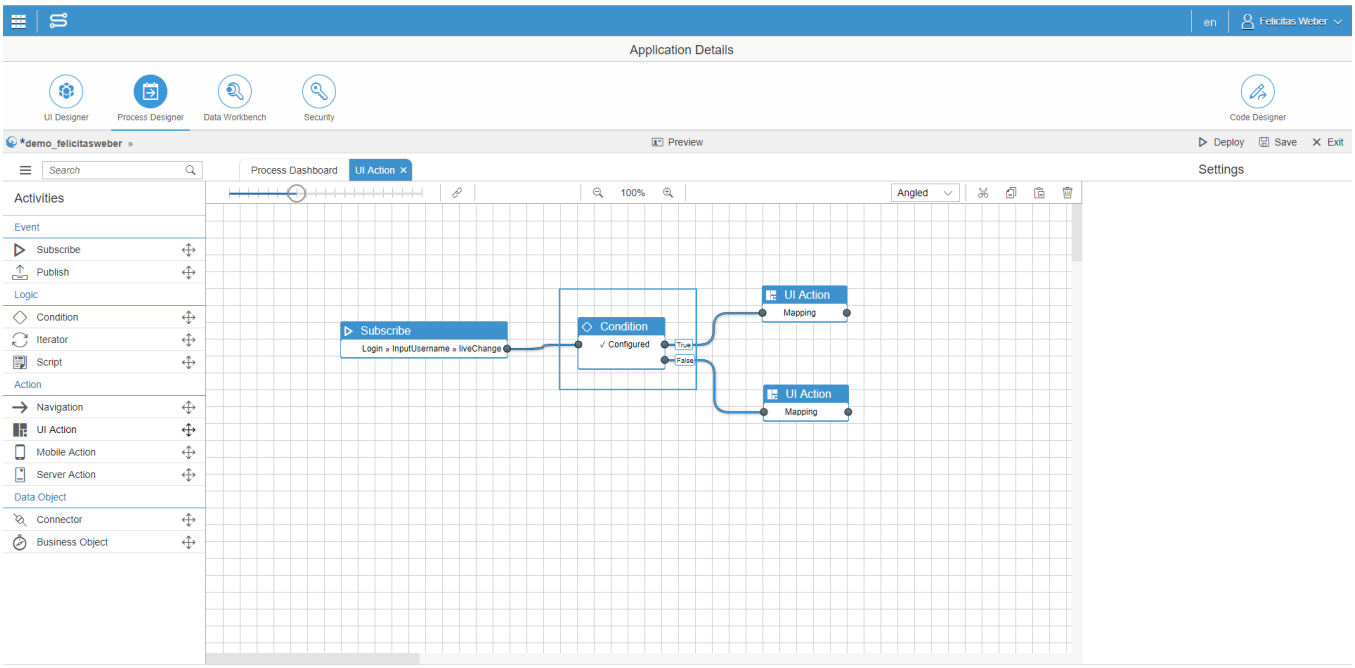
Lets say we have a very simple login screen. There is an input field for your name and a button to submit.
We want the button not to be responsive, as long as the login field is empty. This can be achieved with the UI Action in the Process Dashboard.

At first, we assign the input field “InputUsername” to the Event.



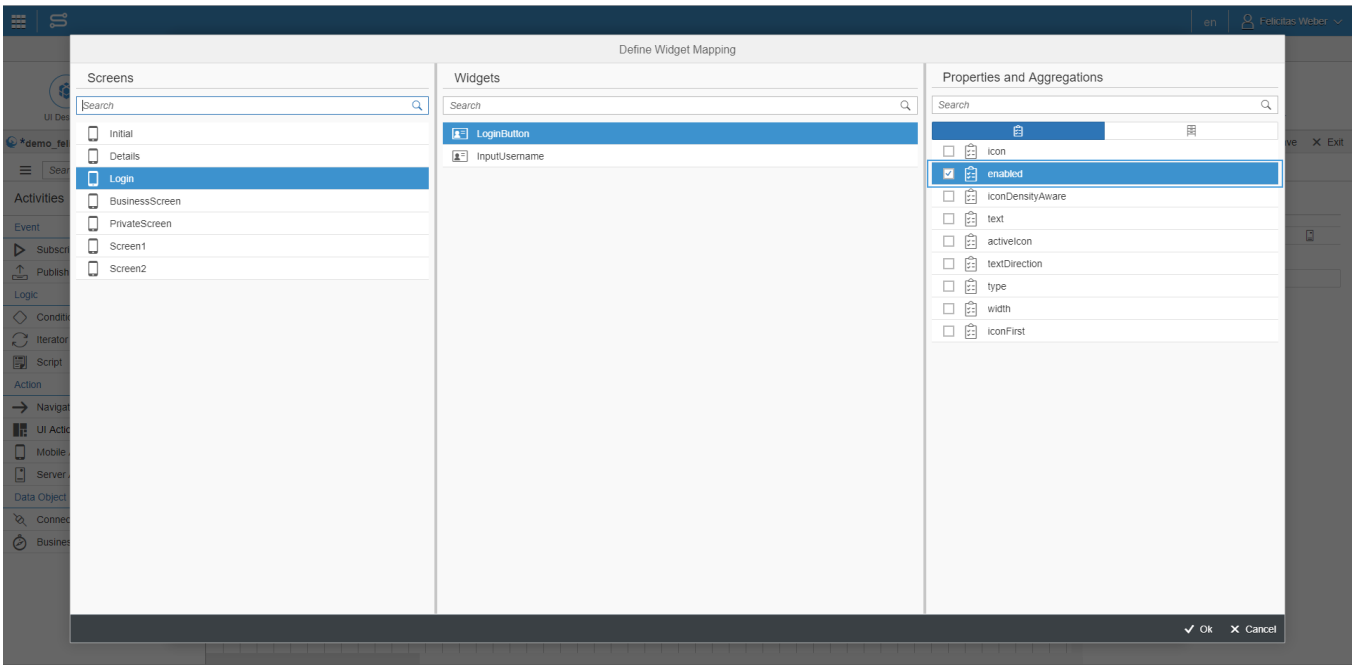
Step 1

Next, we need a condition. Check, if the value of the input field equals an empty string.

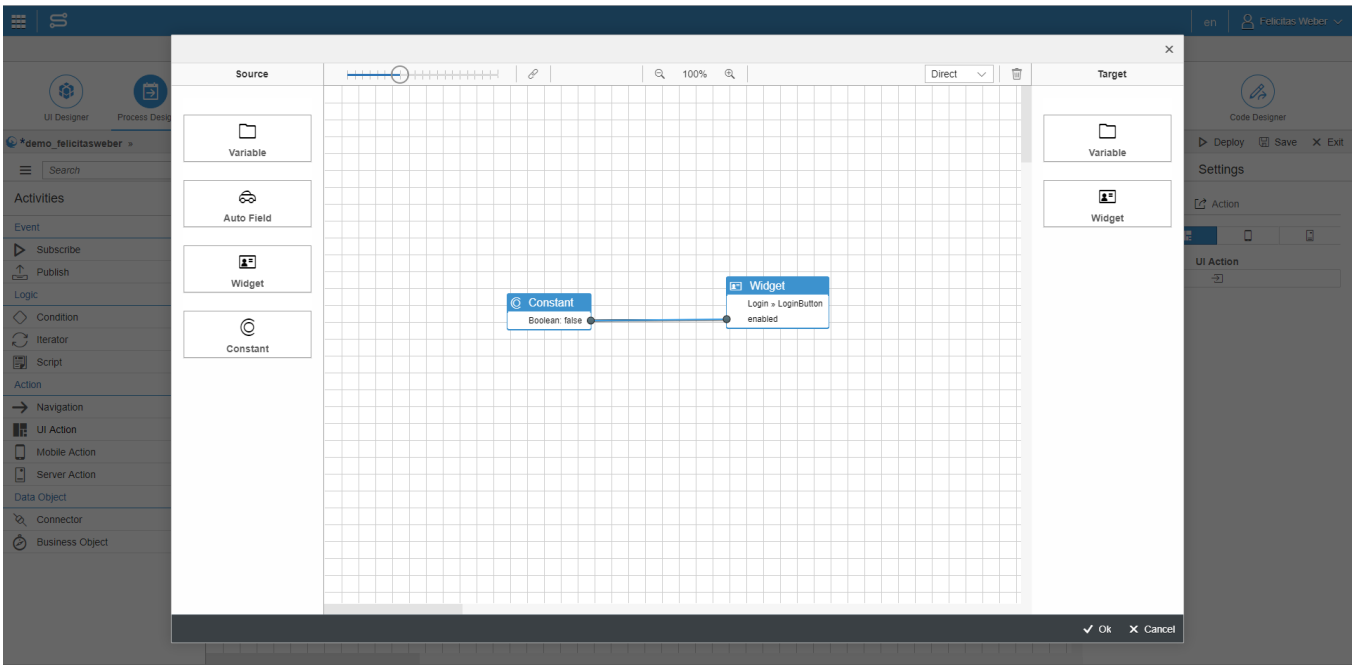


Step 2

If the condition is true and there is nothing written in the input field, the button should not be responsive. Add an UI Action in Step 3 and choose within the mapping dialog the constant (**boolean**, **false**) on the left side (source) and map it to the equivalent widget on the right side (target). Select the screen, widget and property (in this case “enabled”).

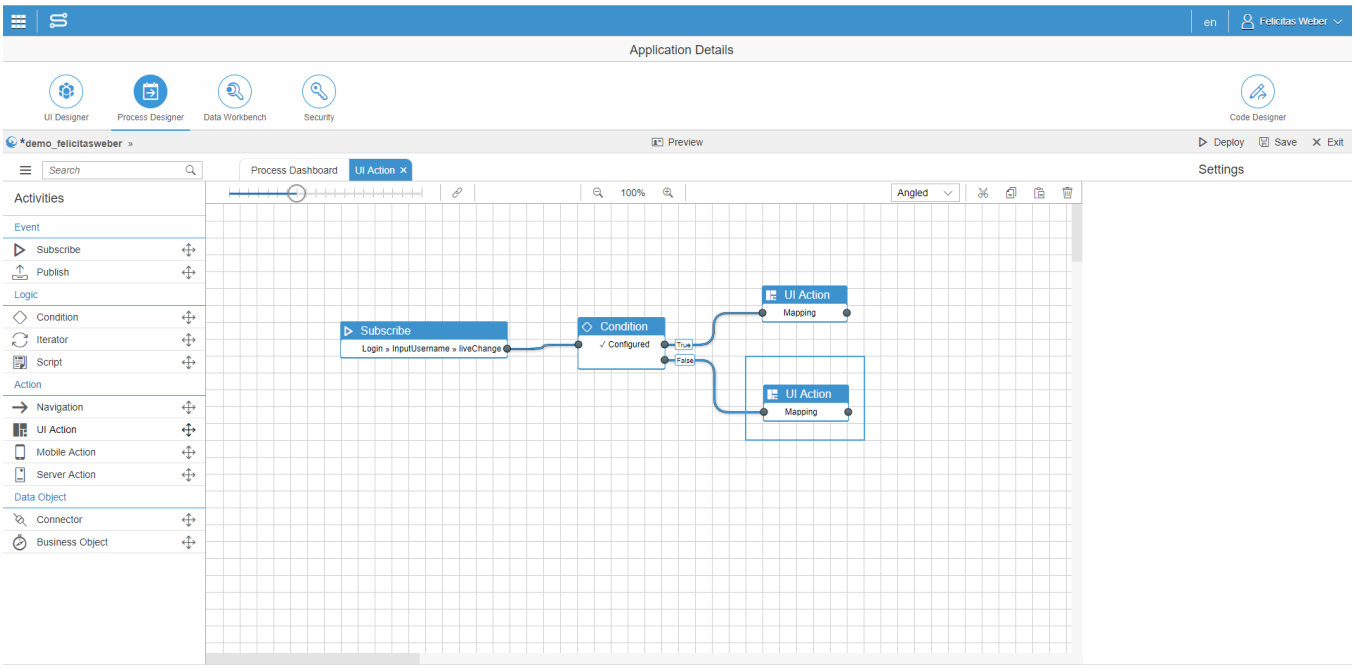

Define Widget Mapping
Screens
Search
Initial
Details
Login
BusinessScreen
PrivateScreen
Screen1
Screen2
Widgets
Search
LoginButton
InputUsername
Properties and Aggregations
Search
icon
enabled
iconDensityAware
text
activeIcon
textDirection
type
width
iconFirst
Ok
Cancel

Step 3

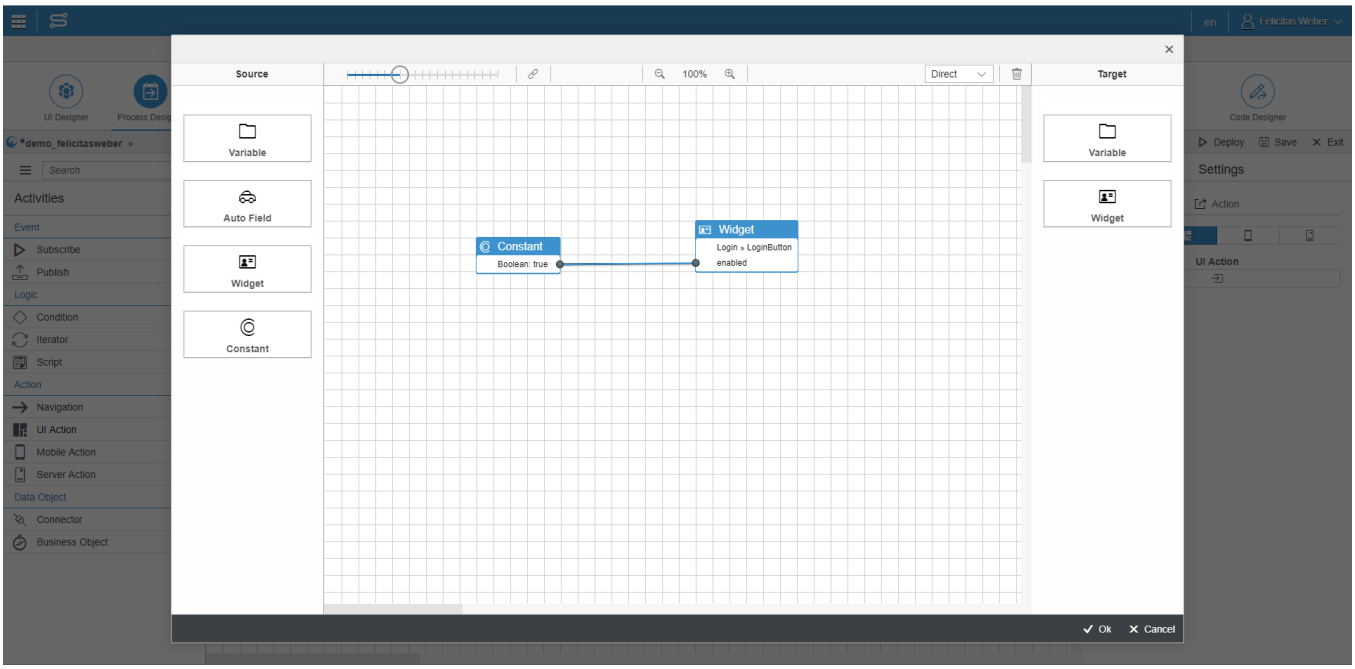


Mapping Dialog

For the other scenario, when there actually is an input written, you can connect the condition to a second UI Action and set the **boolean** via the constant on **true**.

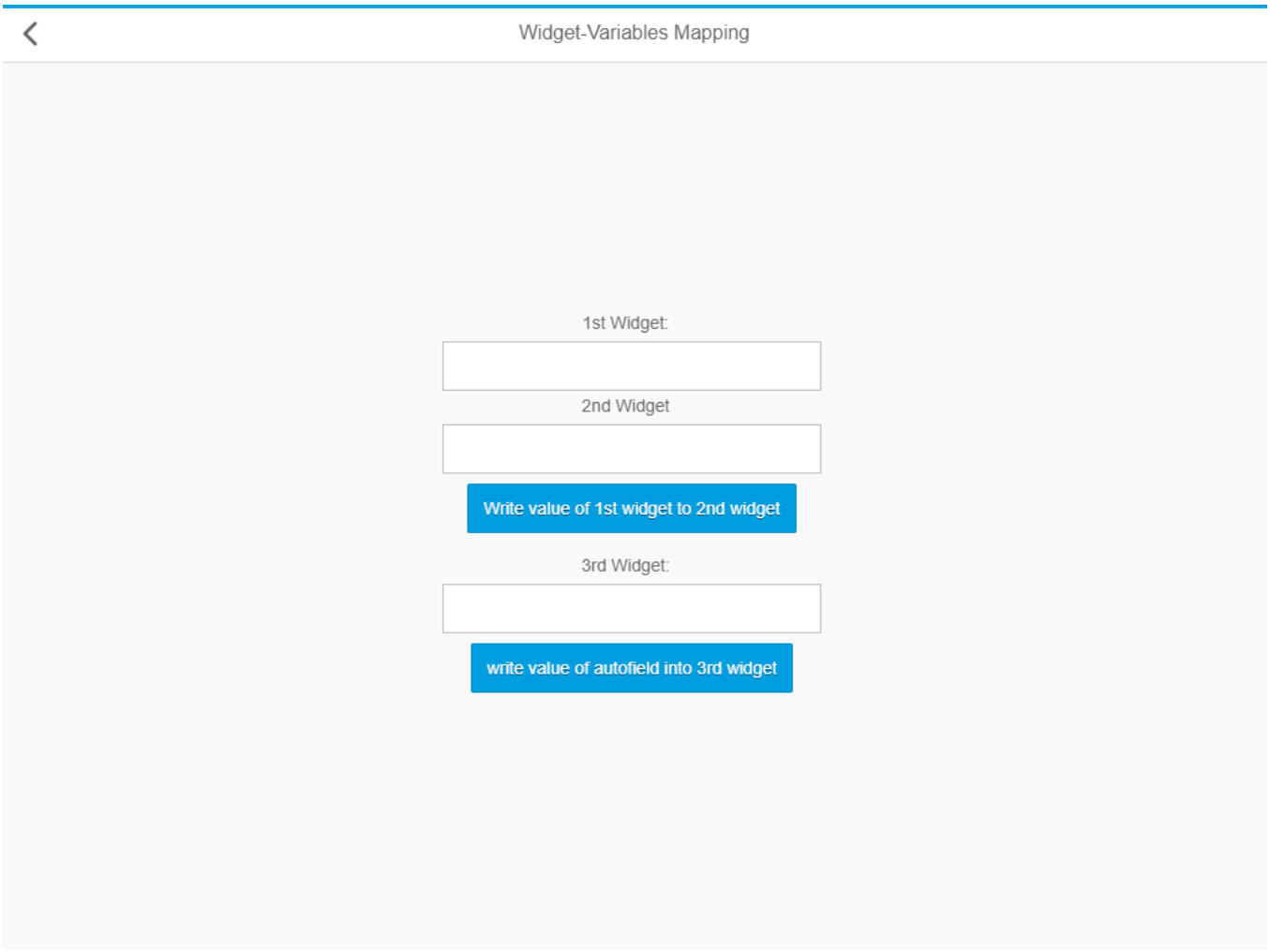

en
Felicitas Weber
Application Details
UI Designer
Process Designer
Data Workbench
Security
Code Designer
*demo_felicitasweber
Preview
Deploy
Save
Exit
Search
Process Dashboard
UI Action
Settings
Activities
Event
Subscribe
Publish
Logic
Condition
Iterator
Script
Action
Navigation
UI Action
Mobile Action
Server Action
Data Object
Connector
Business Object
100%
Angled
Subscribe
Login » InputUsername » liveChange
Condition
✓ Configured
True
False
UI Action
Mapping
UI Action
Mapping

Step 4



Mapping Dialog

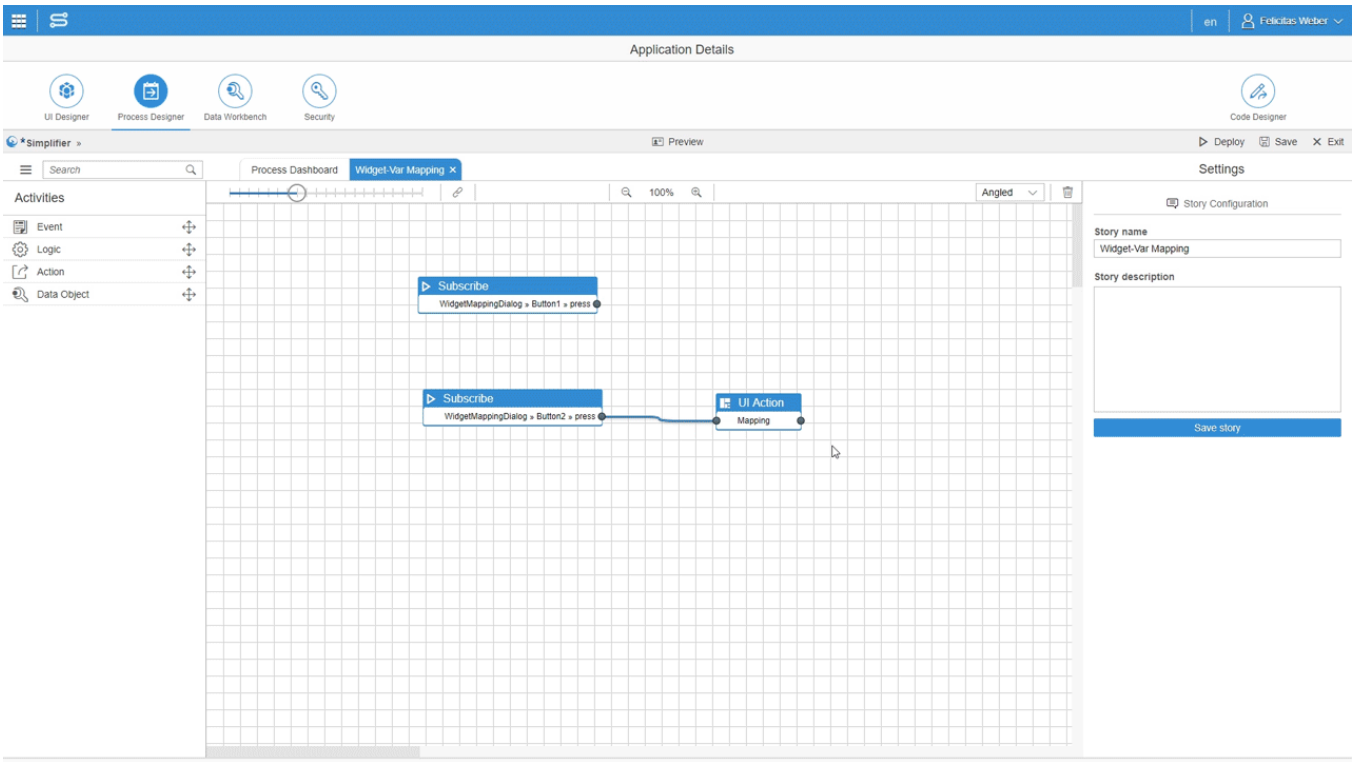
In this example we have an application that writes the value of the first Widget to the second Input field and writes the value of the Auto Field (username) to the third Widget after clicking on the corresponding buttons.



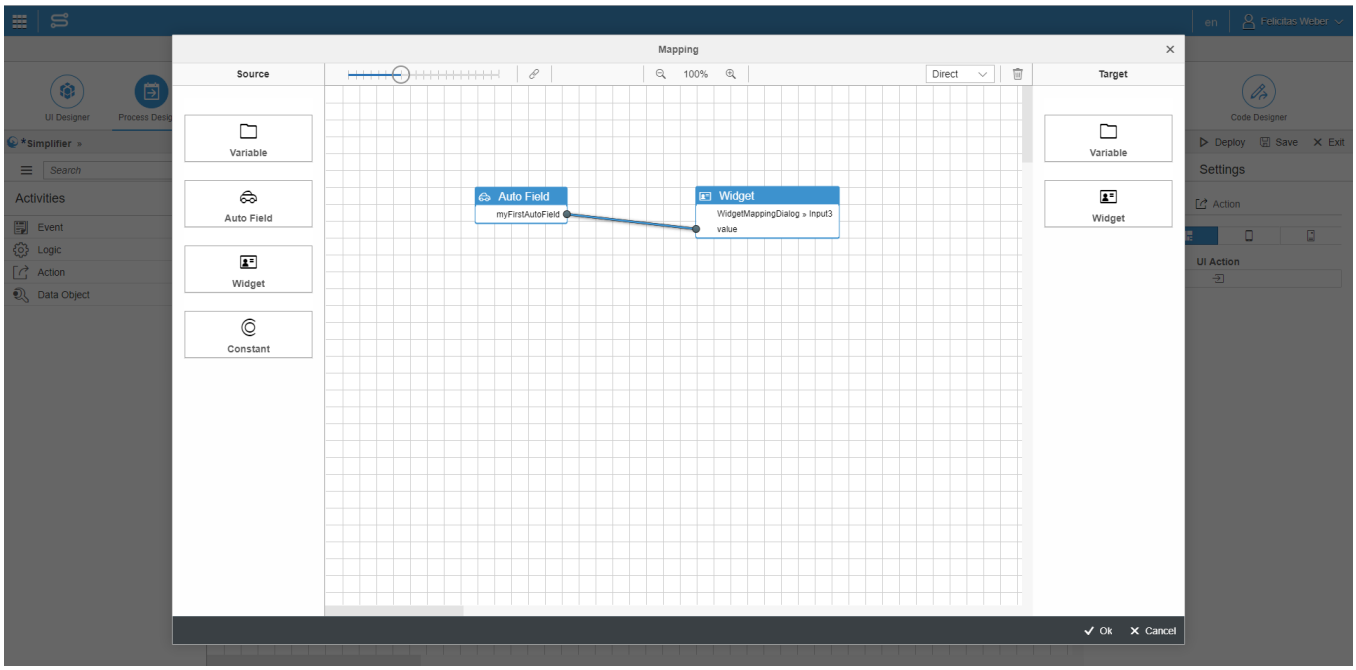
Now in the Process Designer, we use 2 Events and 2 UI Actions for this.

We've already added the press Event on the first Button and now we have to add the UI Action. By clicking on the "mapping dialog", we can open the editor.

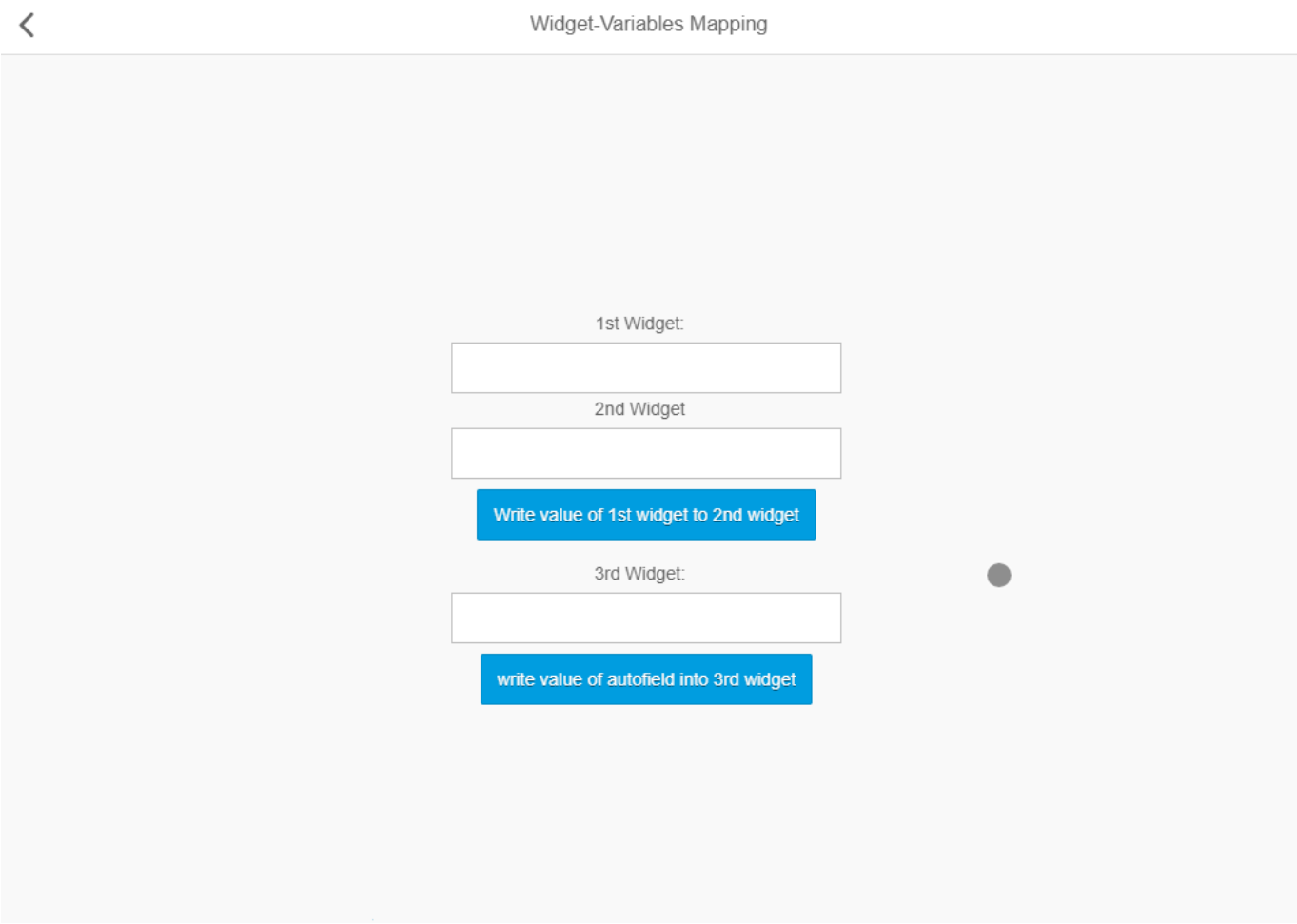
We connect the value of the Widget Input1 with the Widget Input2. We can select the property that we'd like to connect by double clicking on the Widget.



To connect the Auto Field with the third Input field, we add the Auto Field and select the one we need, also by double clicking on it.

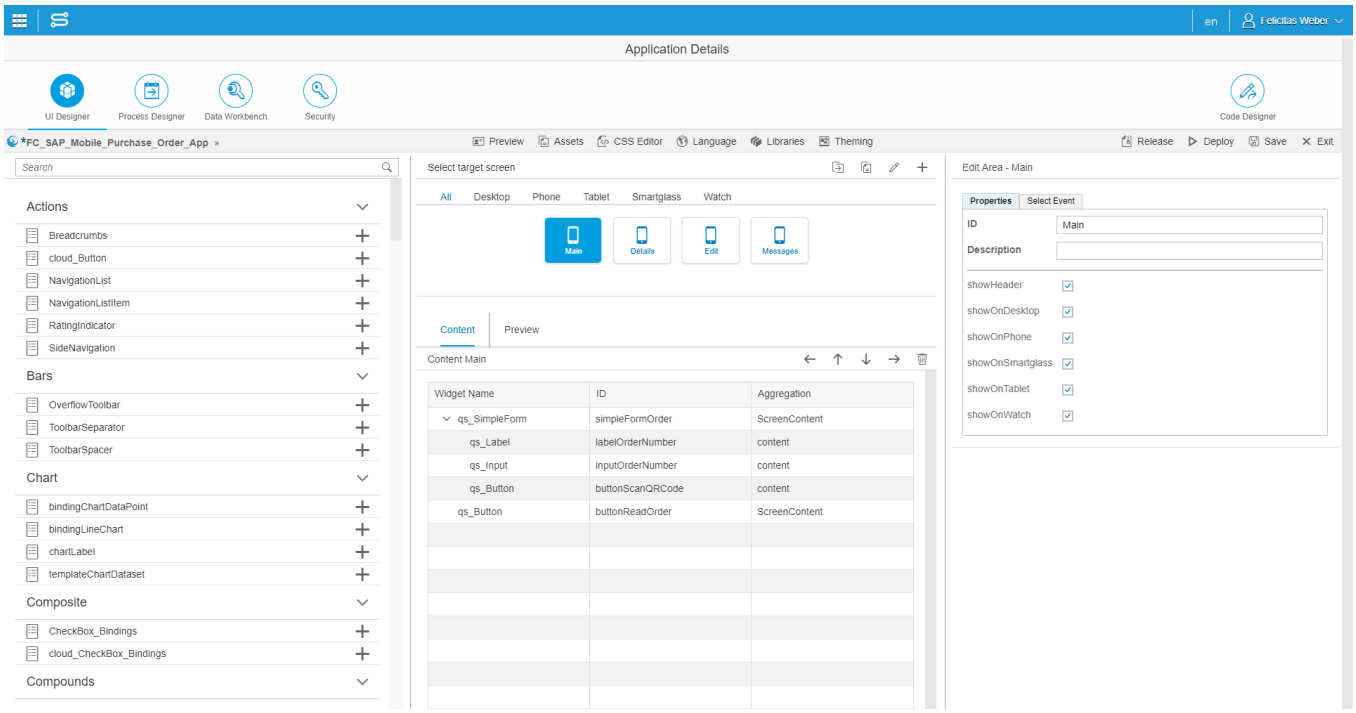


This is the result of our mapping:


Widget-Variables Mapping
1st Widget:
2nd Widget
Write value of 1st widget to 2nd widget
3rd Widget:
write value of autofield into 3rd widget

UI Designer

With the UI Designer you can create and design the user interfaces of your applications. The interface is divided into four sections: On the left side, you can browse the widgets available for your UI. The center presents the content structure as well as an overview about the different screens. On the editing area to the right side, you can change parameter of the widget that is currently selected in the content structure.



You have several options in the upper menu bar to navigate to specific features:

Preview

Preview your App in your browser – be aware that there are no mobile features like camera available. To preview your mobile apps, use the Simplifier Mobile Client.

Assets

You can upload assets like documents, images, CAD models or many other file types.

CSS Editor

Use the CSS Editor to change the layout of your app.

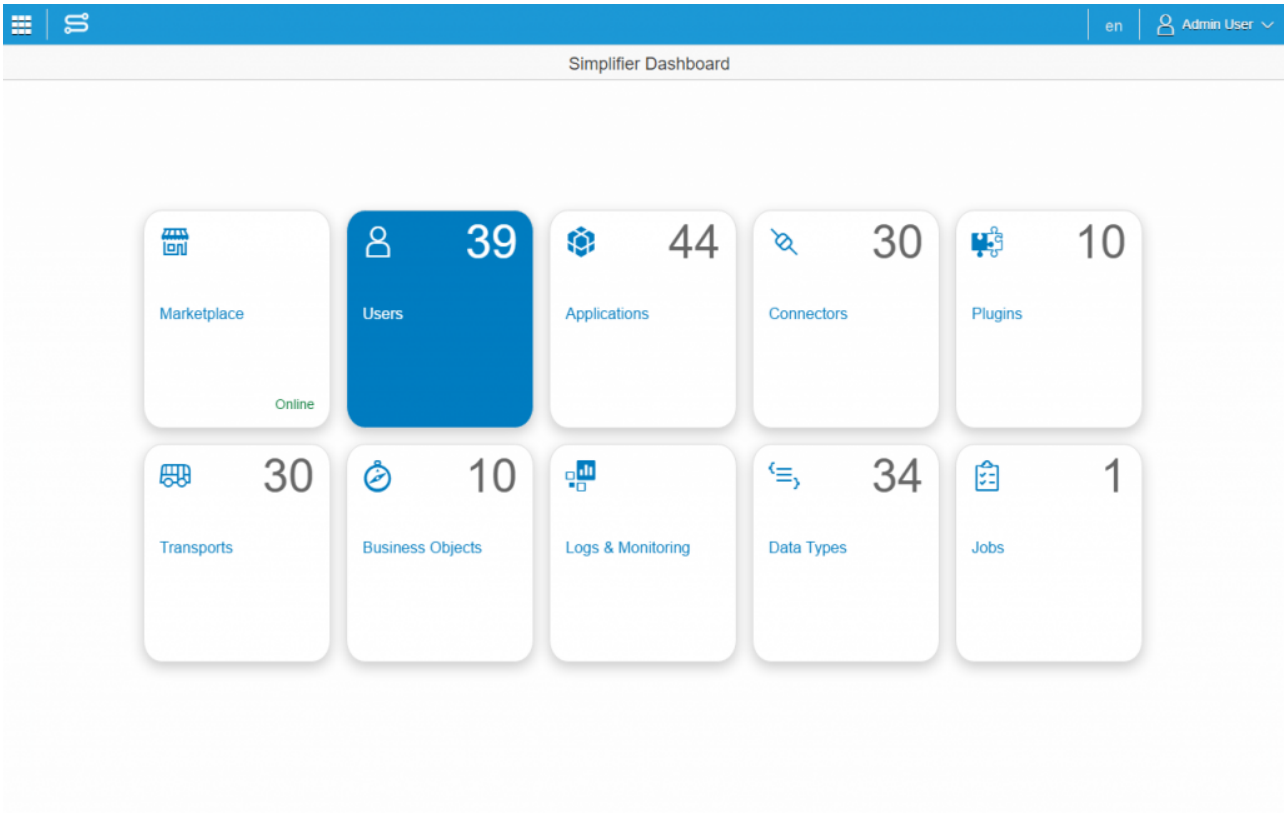
Language

Translate your App into different languages and set the fallback language if the client uses a language that is not available.

 **Libraries**	Add Libraries to your app and get an overview about their dependencies.
 **Theming**	Upload a particular theme to give your app a specific look according to your corporate design guidelines.
 **Release**	If you have tested your app successfully, you can release a new version with release notes.
 **Deploy**	Saves and deploys the current app configuration. It’s necessary to deploy your application before starting the preview.
 **Save**	Save the current app configuration.
 **Exit**	Exit the current app configuration.

User Management

The User Management module lets you define many different parameters for users and the roles they take.
The QR-Code generator for easy accessing, especially with smart glasses, is also located here.

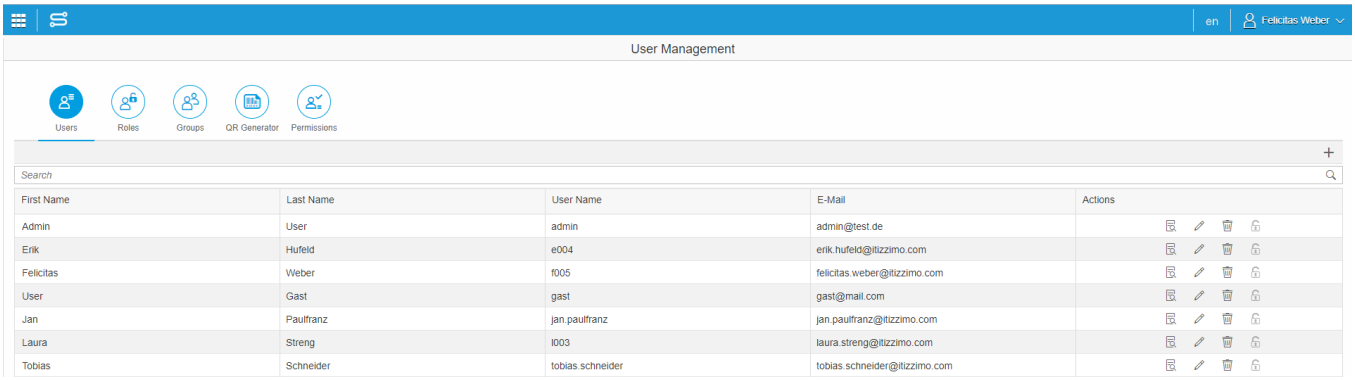


The following table explains the different User Management functions and their meanings:

User	User Master data	Master data for a user account like Email, Username,Expire Date, Address Data
Roles	Collection of permissions	A Role represent an amount of permissions and can be assigned to specific user
Permission	Permissions	Permissions are an authorization like granting access to a specific connector or business app
Groups	Collection of users	A user group contains several users and could be used for workflow logic in business apps like informing a team via email or push notification about a certain event or task

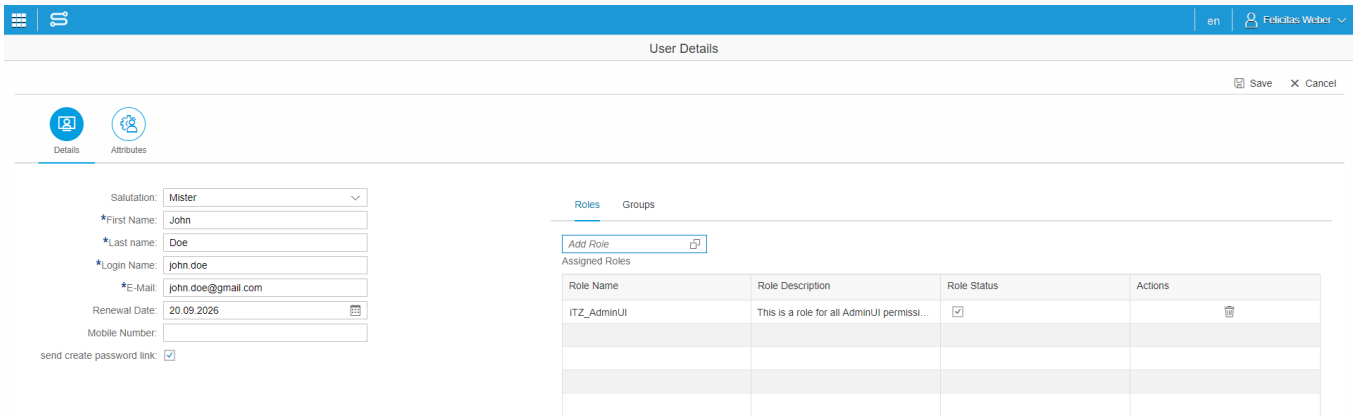
User Overview - Create a new user

The user overview presents you all users of your own Simplifier instance. Aside from first name and last name you also can see the username and the expiry date of the user. Four icons on the right side allow you to see details of the user, edit or delete the user (only when your role has the permissions) and see if the user has been blocked.



click on the “+” icon on the right to add a new user

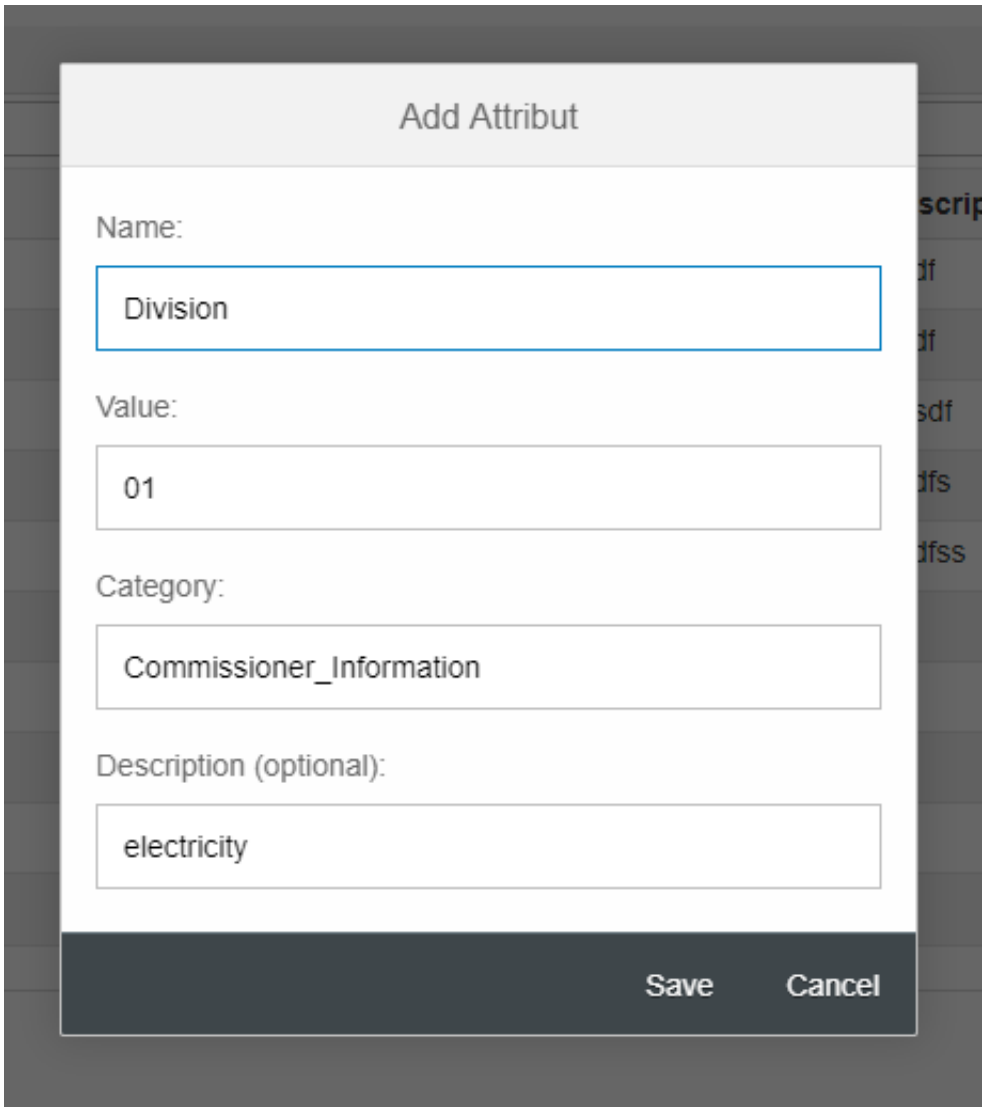
To add a new User click on the “+” icon on the right above the User Table. Within Detail View fill out all required User Information and assign a Role or Group for the User



Fill out all required User Information

After creating a user, a one-time-link via email will be sent to the created user. After email activation via the link, a new password can be set.

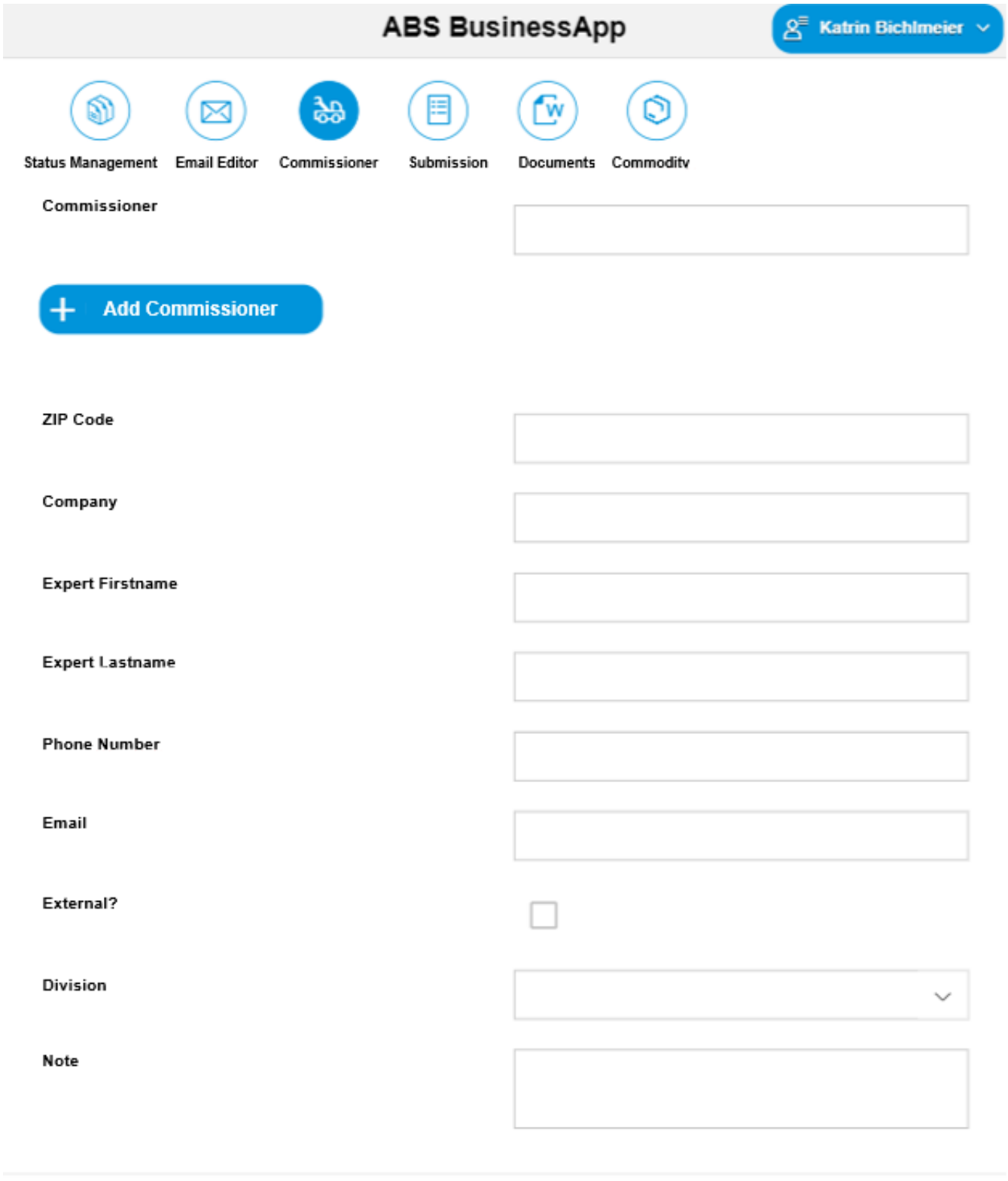
You can also assign different **attributes** to a user.



Example Use Case:

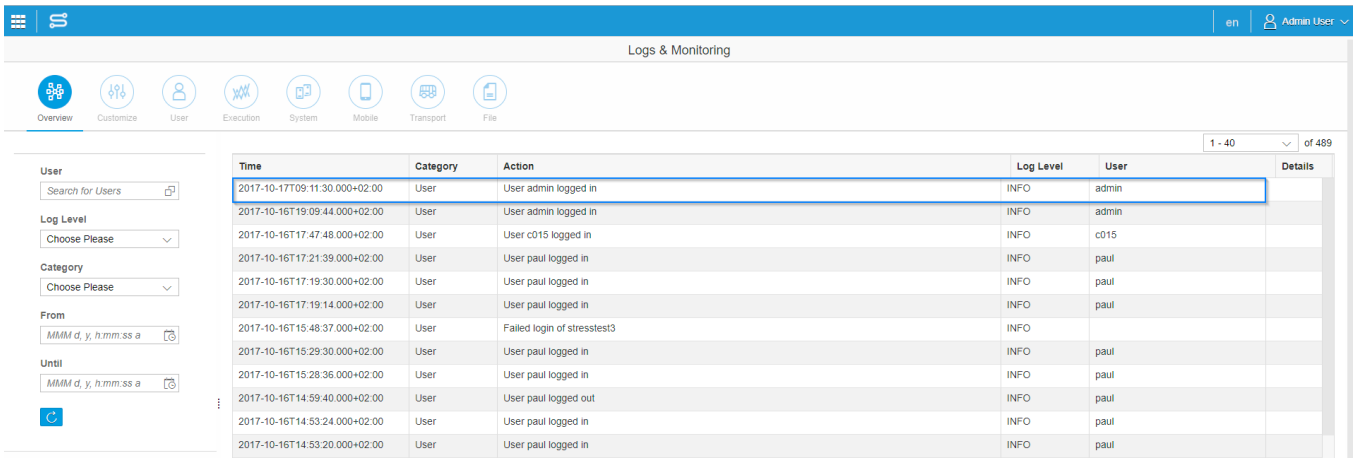
Let’s say we have a utility company.

They use an application to administrate their commissioners. They can add a new commissioner within the app and assign different attributes to him, e.g. a ZIP code or in which division he is working (gas, water, electricity or district heating). Now every commissioner is created as an own user in the Simplifier and you can manage the attributes directly in the user management.


ABS BusinessApp
Katrin Bichlmeier
Status Management
Email Editor
Commissioner
Submission
Documents
Commoditv
Commissioner
Add Commissioner
ZIP Code
Company
Expert Firstname
Expert Lastname
Phone Number
Email
External?
Division
Note

User-Log

User-Log logs the Login and Logout actvities from the users.



User Log

Using OPC-UA Connector Calls

As for now, OPC-UA connector calls can only be called programmatically in the javascript code. In order to do this, please open the code designer, as described here.
We have defined the basic OPC-UA connector calls with the following names after the colon.

READ Call: TIA_READ

WRITE Call: TIA_WRITE

READWRITE Call: TIA_READWRITE

For each call, it will be described how to fill the nescessary paramters and how to use the call.

Websocket API Documentation (Incomplete)

This section contains a description for the websocket methods. As for now this API is considered incomplete.

---

onOpen

The function, which will be executed, when the websocket is opend. The code block can be found below:

webSocket.onopen = function(message) {[FunctionLogic]};

---

onClose

The function, which will be executed, when the websocket is beign closed. The code block can be found below:

webSocket.onclose = function(message) {[FunctionLogic]};

---

onError

The function, which will be exceuted, when the websocket throws an error. The code block can be found below:

webSocket.onerror = function(message) {[FunctionLogic]};

---

onMessage

The function, which will be executed, when the websocket returns a message. The code block can be found below:

webSocket.onmessage = function(message) {[FunctionLogic]};

---

close

This function closes the websocket.

webSocket.close();

---

send

A specific request will be send to the websocket and interpreted by the other side of the connection. In this case a connector:

webSocket.send([request]);

[FunctionLogic]

This parameter contains the function, which will be executed.

[request]

This parameter is a JSON object send to the respective function. The requests for the Connector can be found here.

Websocket Communication with Connectors

The following sections contain information about how to use asynchronous connector subscriptions and unsubscriptions. For now, the subscription and unsubscription process can be initialized only programmatically and solely with the OPC/UA Connector.

Websocket Generation

After receiving a token, as described here, a websocket connection can be established with the application server with the following code:

```
 var webSocket = null
webSocket = new WebSocket("ws://localhost:8080/client/1.0/connectorAsync/" + [ConnectorName] + "?SimplifierToken=" + [Token]);
```

[ConnectorName]

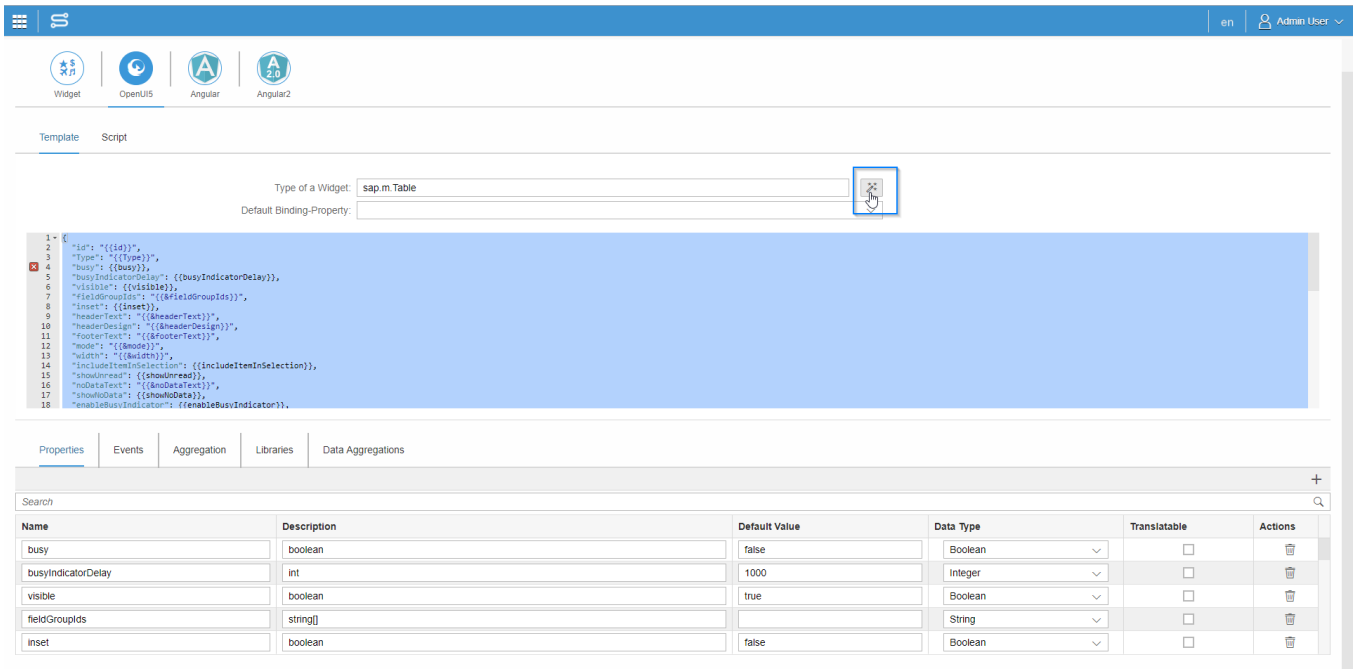
This parameter represents the connector, that the websocket connection will be established to.

[Token]

The generated token object.

Widget Assistant

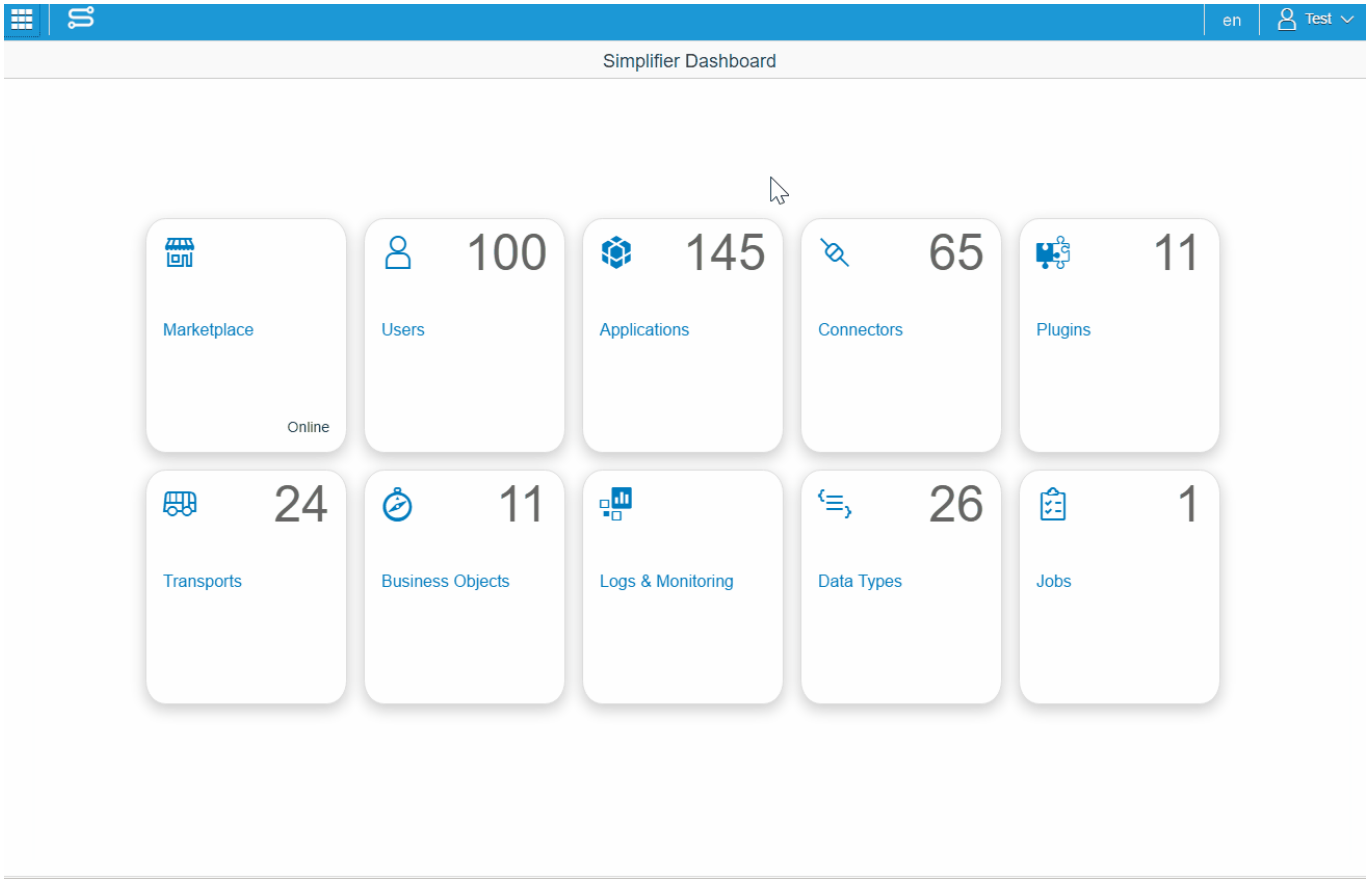
An assistant is provided for maintaining OpenUI5 widgets. If you enter any UI5 control type in the Widget Customizer on the OpenUI5 tab and click the magic wand button, the template, properties, events, and aggregation is filled with the information of the control and all its ancestors.


en
Admin User
Widget
OpenUI5
Angular
Angular2
Template
Script
Type of a Widget:
sap.m.Table
Default Binding-Property:
Properties
Events
Aggregation
Libraries
Data Aggregations
Search
Name
Description
Default Value
Data Type
Translatable
Actions
busy
boolean
false
Boolean
busyIndicatorDelay
int
1000
Integer
visible
boolean
true
Boolean
fieldGroupIds
string[]
String
inset
boolean
false
Boolean

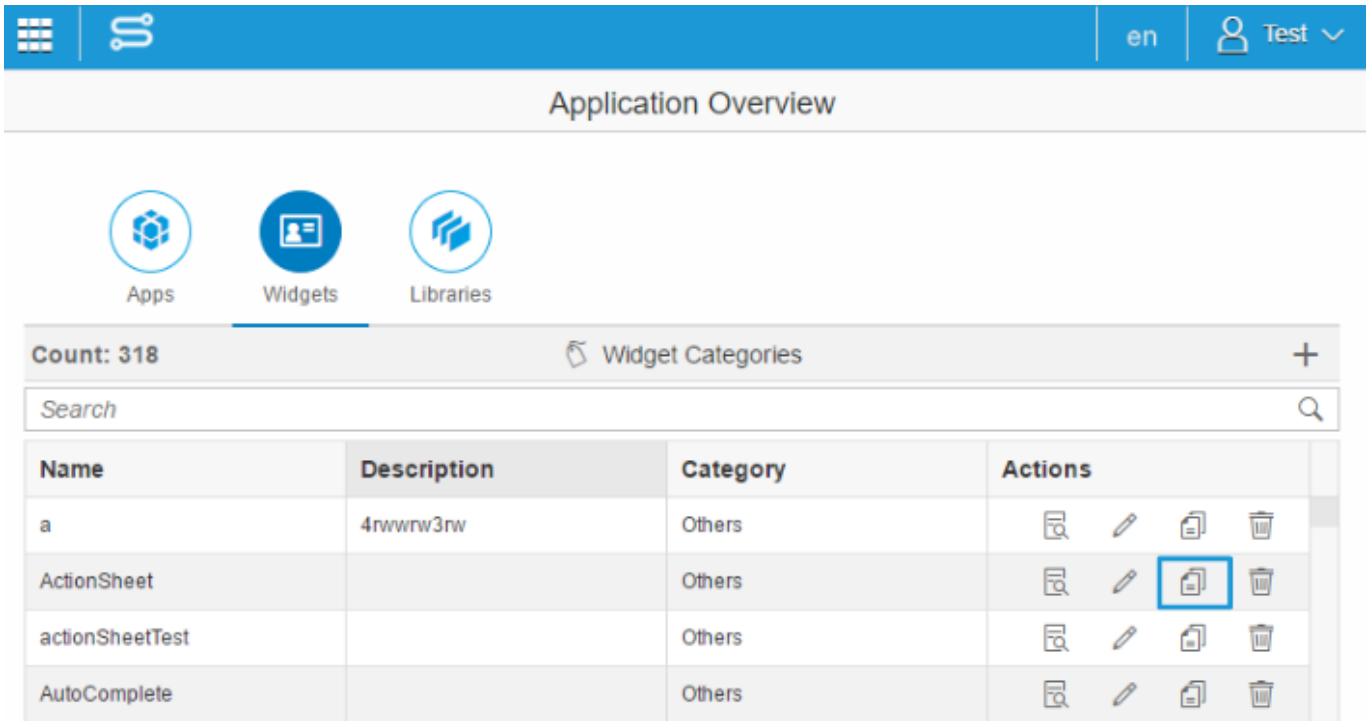
Widgets

A Widget represents a specific element in the user interface (e.g. checkbox, button or login screen).

To customize a Widget, click on the “Applications” tile and choose the Widgets tab. Press the “+” button to add a new one.



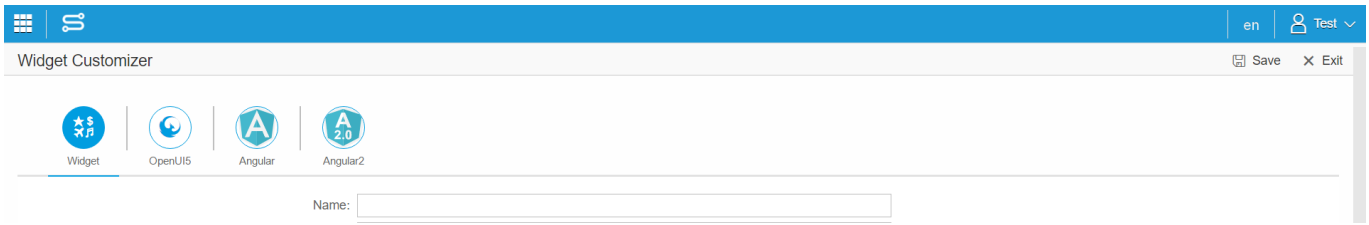
We added a feature, so you are able to copy an existing Widget as well. Click on the icon and give it a unique name.



If you want to create a new Widget, you will be forwarded to the Widget Customizer. There you'll have 4 different tabs:

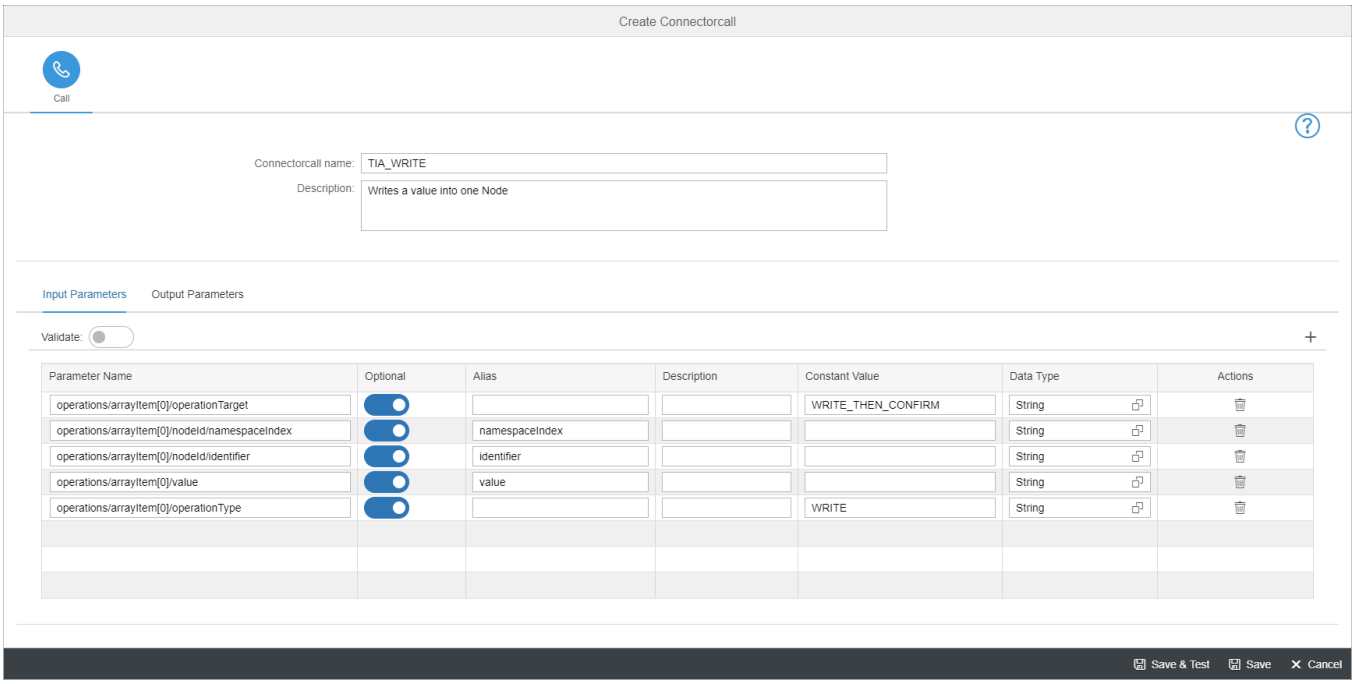
The first one describes all global settings of the Widget which are independent from UI5 or Angular (e.g. category).
The OpenUI5, Angular and Angular 2 tabs contain the specific parameters and settings for each framework.

Later on, the Simplifier will choose the right parameters depending on your application type.



WRITE Call - OPC/UA Connector

Call for WRITE operations (The name TIA_WRITE is the arbitrary chosen name for this call)



Input Parameter

For the WRITE Connector Call, you need to configure the "**operationType** " and the "**nodeId**" (consisting of 2 parameter: **identifier** and **namespaceIndex**). Furthermore you need to define the **operationTarget** and the **value**.

operationType: Defines which operation you want to execute, in this case "WRITE".
Parameter Name: operations/arrayItem[0]/operationType
Constant Value: WRITE
Data Type: String

nodeID: Defines the identification of the OPC/UA node. It is split in 2 Parameter:

- **Identifier**:
  Parameter Name: operations/arrayItem[0]/nodeId/identifier
  Data Type: String or Numeric
  Constant Value: 84
- **NamespaceIndex**:
  Parameter Name: operations/arrayItem[0]/nodeId/namespaceIndex
  Data Type: String
  Constant Value: 0
  In every namespace, each ID must be unique (it is possible to use the String "7617" and the Numeric 7167 together in one namespace)

operationTarget:
Parameter Name: operations/arrayItem[0]/operationTarget

Constant Value: Choose between

- WITHOUT_EVERYTHING
- WRITE_ONLY
- CONFIRM_ONLY
- WRITE_THEN_CONFIRM
- READ_ONLY
- READ_THEN_WRITE
- READ_THEN_CONFIRM
- READ_THEN_WRITE_THEN_CONFIRM

values The values, which are going to be written into the nodes.
Parameter Name: operations/arrayItem[0]/value
Data Type: String
The position of the values will equal the position of the node. For example if node a is on the first position and node b on the second, whereas value x is on the first and value y on the second position then value x will be written into node a and value y into node b.

NOTE: The specific commands are NOT defined here!

Output parameters

You can return all Output Parameter like this:

Parameter Name: /
Data Type: String

If you want to get only selected Output Parameter, use the following syntax:

Parameter Name: operationsResult/[0]/newValue/value (exemplary)
Data Type: depends on the Parameter you want to be returned.

For now only the complete unformatted JSON will be returned.

Simplifier Academy

Courses & Documentation

PDF generated April 30, 2019 at 8:38 AM