Problemstellung

In SAP Fiori ist es wichtig, sicherzustellen, dass Benutzer nur die Berechtigungen haben, die sie für ihre Aufgaben benötigen, insbesondere wenn es um Änderungen an Daten geht. Eine Möglichkeit, dies zu überprüfen, ist die Verwendung der „**What You See Is What You Get**“ (WYSIWYG)-Methodik. Sie ermöglicht es, dass Benutzer in den Apps nur die Aktionen sehen können, für die sie auch berechtigt sind.

Nach unserem Kenntnisstand ist eine derartige Anforderung nicht über die Boardmittel von SAP Fiori umsetzbar. Ein Blick in die UI5 Demo Kit Dokumentation offenbart folgendes: *Moreover, common security mechanisms, which are usually taken for granted, like user authentication, session handling, authorization handling, or encryption are not part of SAPUI5 and need to be handled by the server-side framework and/or custom code of the application.*

Leseberechtigungen auf Daten können in den Fiori (zumeist) unterliegenden CDS-Views (Core Data Services) durch CDS-DCL-Objekte (Data Control Language) problemlos und elegant gesteuert werden. Doch was ist, wenn verschiedene Typen von Anwendern mit verschiedenen Arten von Berechtigungen auf ein und denselben Datensatz zugreifen wollen?

```
@EndUserText.label: 'Restrict Sales Data Access by Sales Organization'
define role Z_SALES_DATA_ACCESS {

    // Berechtigung prüfen gegen ein Berechtigungsobjekt
    grant select on Z_SALES_DATA
    where salesOrganization = aspect pfcg_auth ( 'V_VORG' , 'VKORG' );

}
```

„grant update“ und „grant delete“ sind in einer DCL nicht vorgesehen

Was sagt SAP zu diesem Problem?

SAP sieht derartige Funktionalität in Fiori-Apps nicht vor. Folgende Alternativ-

Lösungen sind möglich:

- A) Jedem Typ von Anwender wird eine eigene Applikation zur Verfügung gestellt
- B) Implementierung von kundeneigenen Lösungen
- C) Verwendung von separaten CDS-Views, welche auf die jeweiligen Änderungsberechtigungen per DCL eingeschränkt wurden

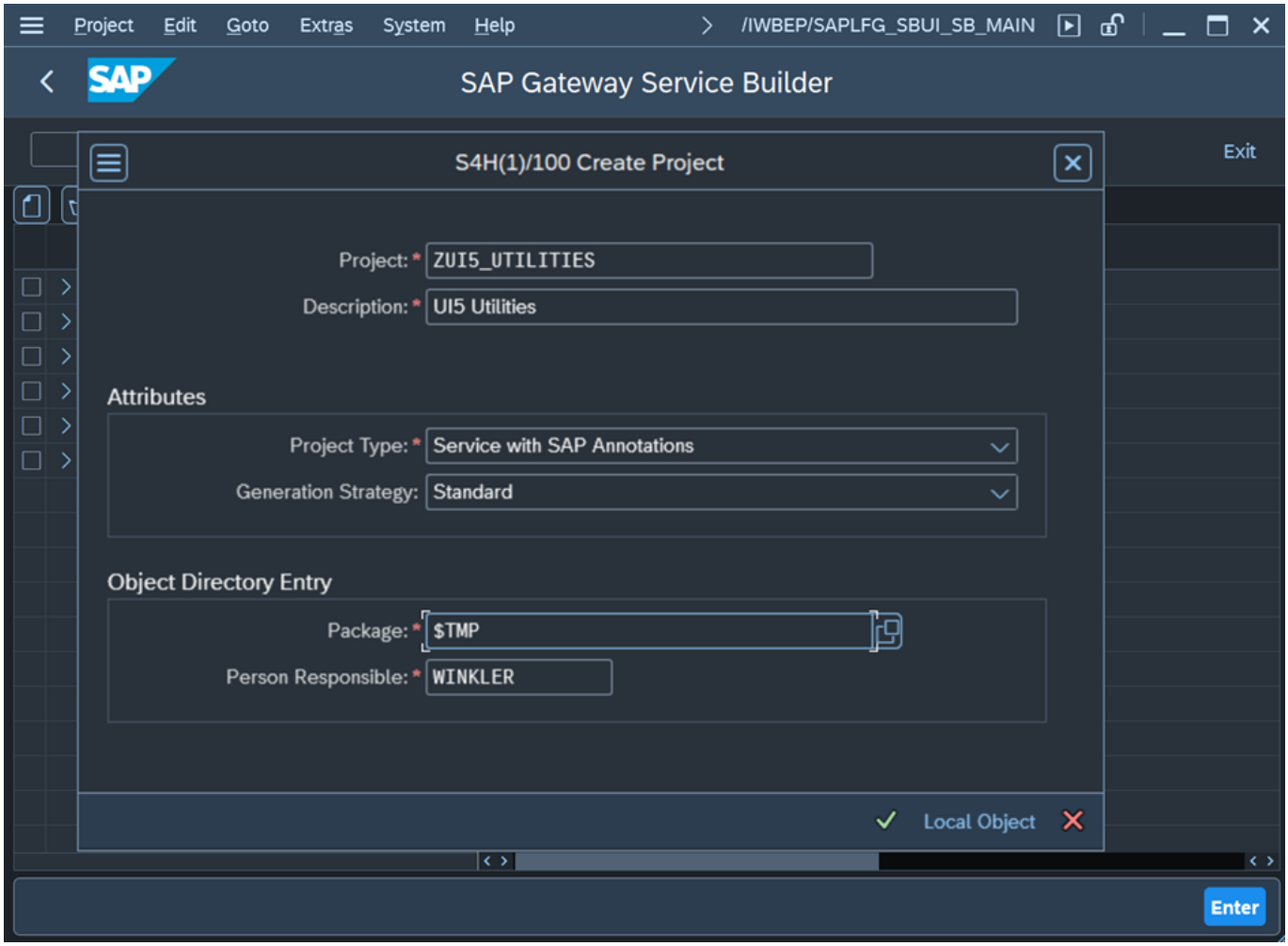
Mit der Lehrmeinung A wollen wir uns an dieser Stelle nicht zufriedengeben. Dieser Blogartikel greift die Variante B auf und demonstriert die Implementierung eines einfachen Web-Service für die Durchführung von PFCG-Berechtigungsprüfungen im UI5-Frontend. Auf Variante C werden wir ggf. in einem zukünftigen Artikel eingehen. An dieser Stelle sei ausdrücklich erwähnt, dass wir uns gern belehren lassen, sollte eine bessere Lösung für die beschriebene Problematik existieren oder mit neuen Releases auf uns warten.

Lösung über einen Function Import

Function Imports in SAP sind spezielle Funktionen in OData-Services, die komplexe Geschäftslogik ausführen. Sie ermöglichen das Abrufen, Bearbeiten oder Löschen von Daten durch definierte Operationen, die über Standard-CRUD-Operationen hinausgehen. Auch wenn es in modernen Releases andere Möglichkeiten gibt, ähnliche Funktionen bereitzustellen, sollte die dargestellte Methode in allen gängigen SAP-Releases ohne zusätzliche Voraussetzungen umsetzbar sein. Wir gehen einzig davon aus, dass im darunterliegenden ABAP-Stack die SAP-Gateway-Komponente zur Verfügung steht.

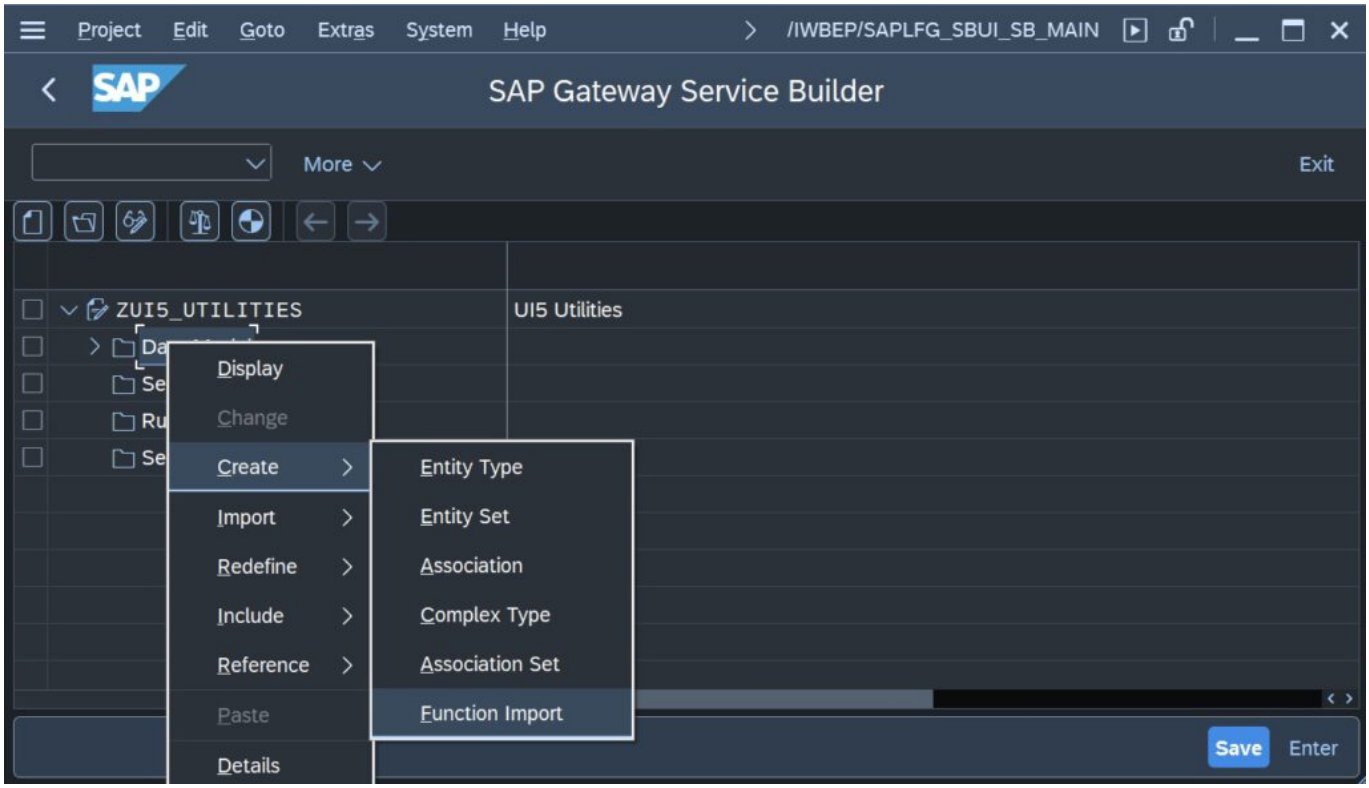
1. Anlage SEGW Projekt mit Function Import

Zunächst legen wir in der Transaktion SEGW ein neues Gateway Service Builder-Projekt ZUI5_UTILITIES an. Die Einstellungen belassen wir auf Standard.

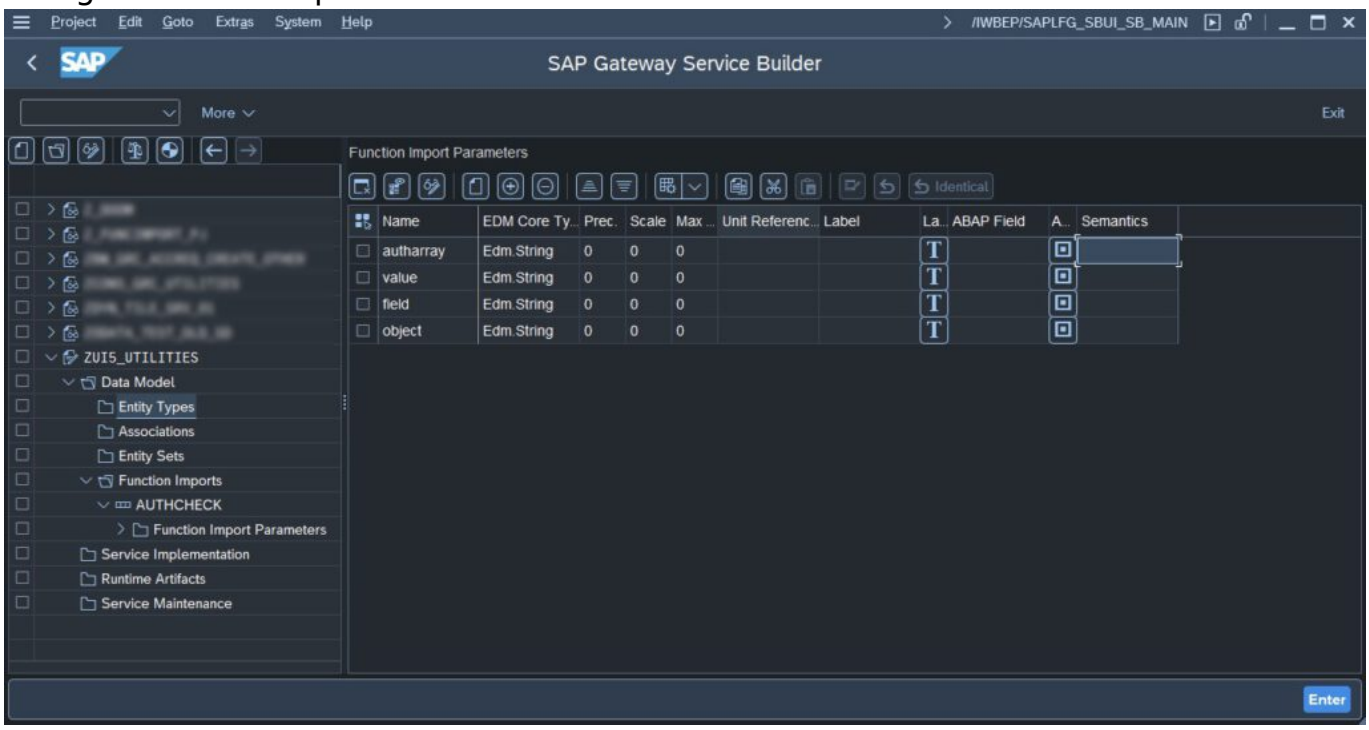


Anlage SEGW Project

Anschließend wird ein Function Import erstellt und unter „Function Import Parameters“ einfache Parameter mit dem Datentyp Edm.String eingetragen. Wir verwenden für diese Demonstration sowohl die einfache Kombination aus genau einem Objekt, eines Feldes und einem Wert, als auch den Import-Parameter autharray, welcher für komplexere Abfragen im JSON-Format gedacht ist.

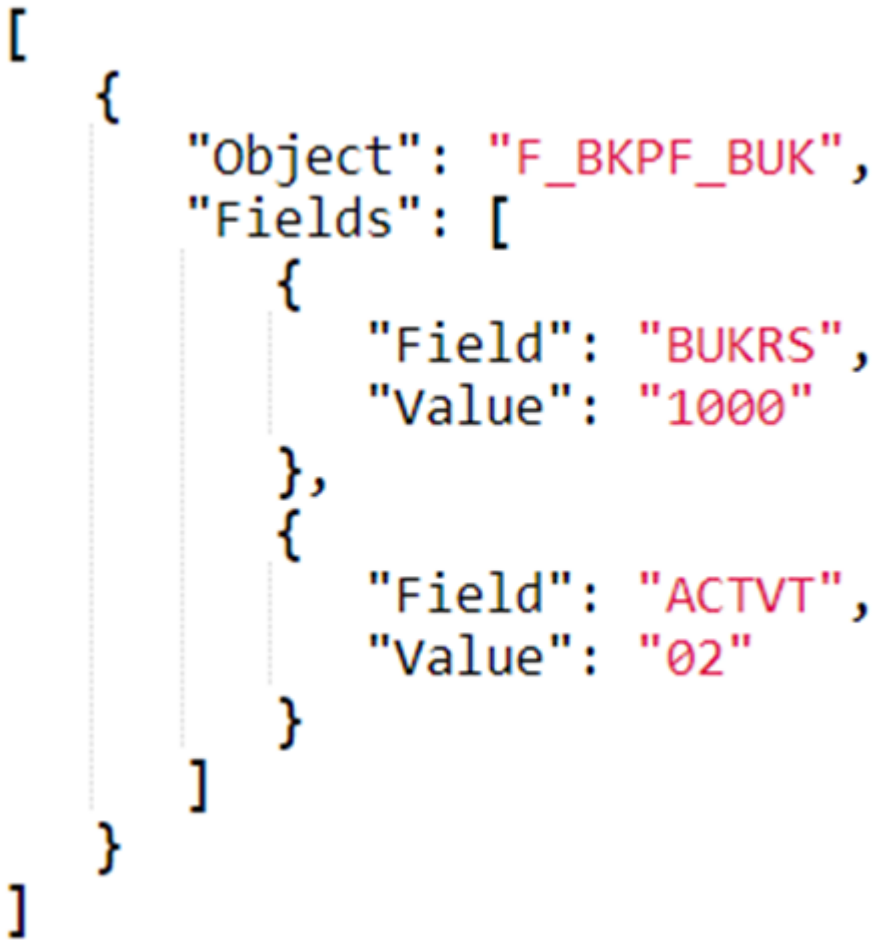


Anlage Function Import



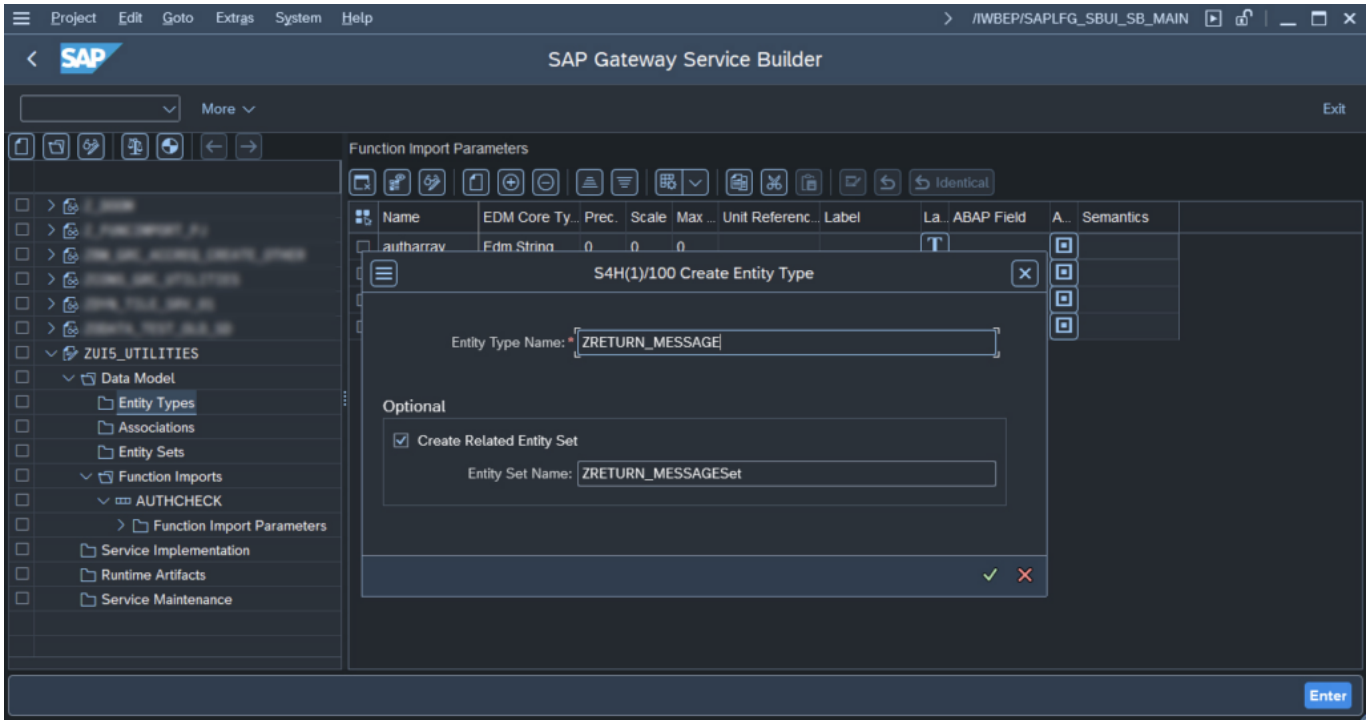
Import Parameter

Eine komplexe Berechtigungsprüfung könnte der folgenden JSON-Notation entsprechen. Auf die Implementierung einer derartigen generischen Prüfung werden wir in einem Folgeartikel eingehen.

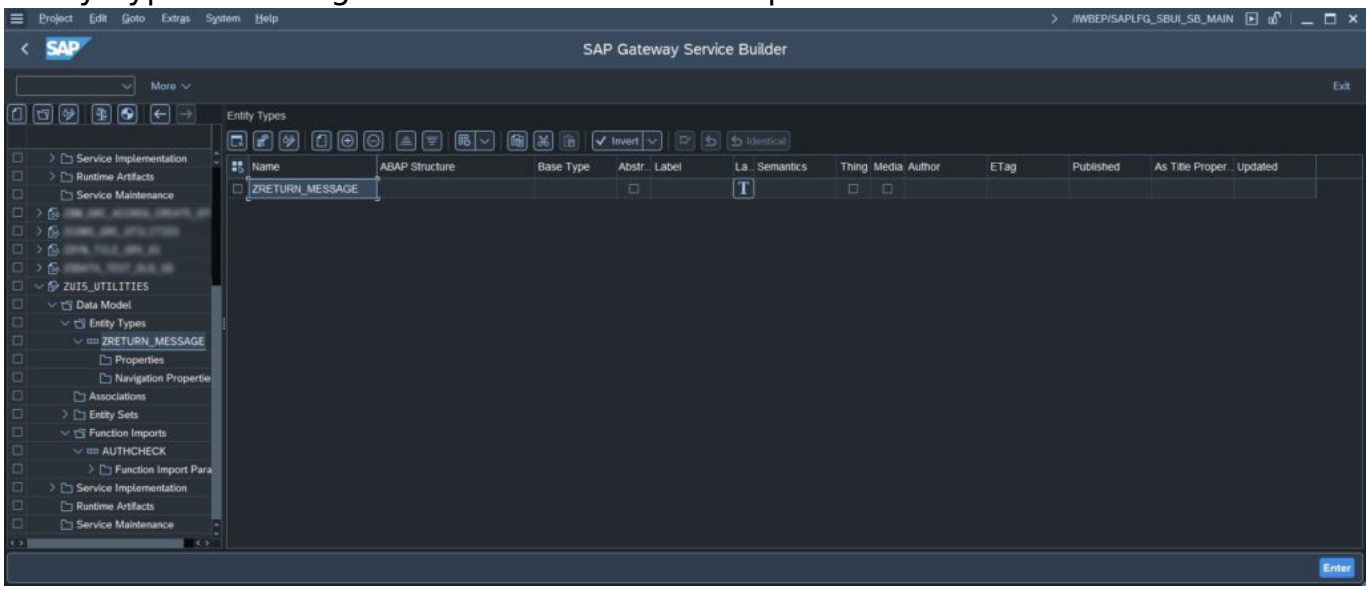


Mögliche JSON-Struktur

Für die Rückgabe des Function Imports wird eine Rückgabestruktur benötigt. Diese legen wir im Reiter „Entity-Types“ an und erstellen gleichzeitig ein Entity-Set, auch wenn es aktuell nicht notwendig ist. Für die Rückgabestruktur verwenden wir in diesem möglichst einfachen Beispiel ZRETURN_MESSAGE mit der Eigenschaft message, welche als ebenfalls als Edm.String zurückgegeben wird.

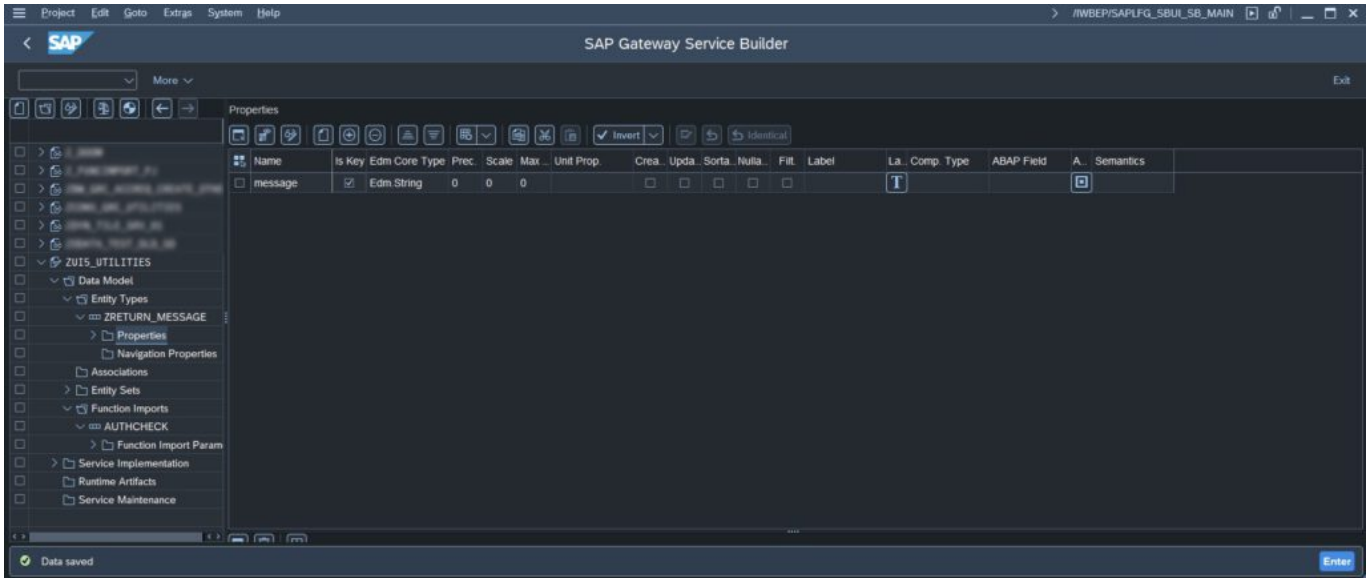


Entity Type für Rückgabewerte des Function Imports



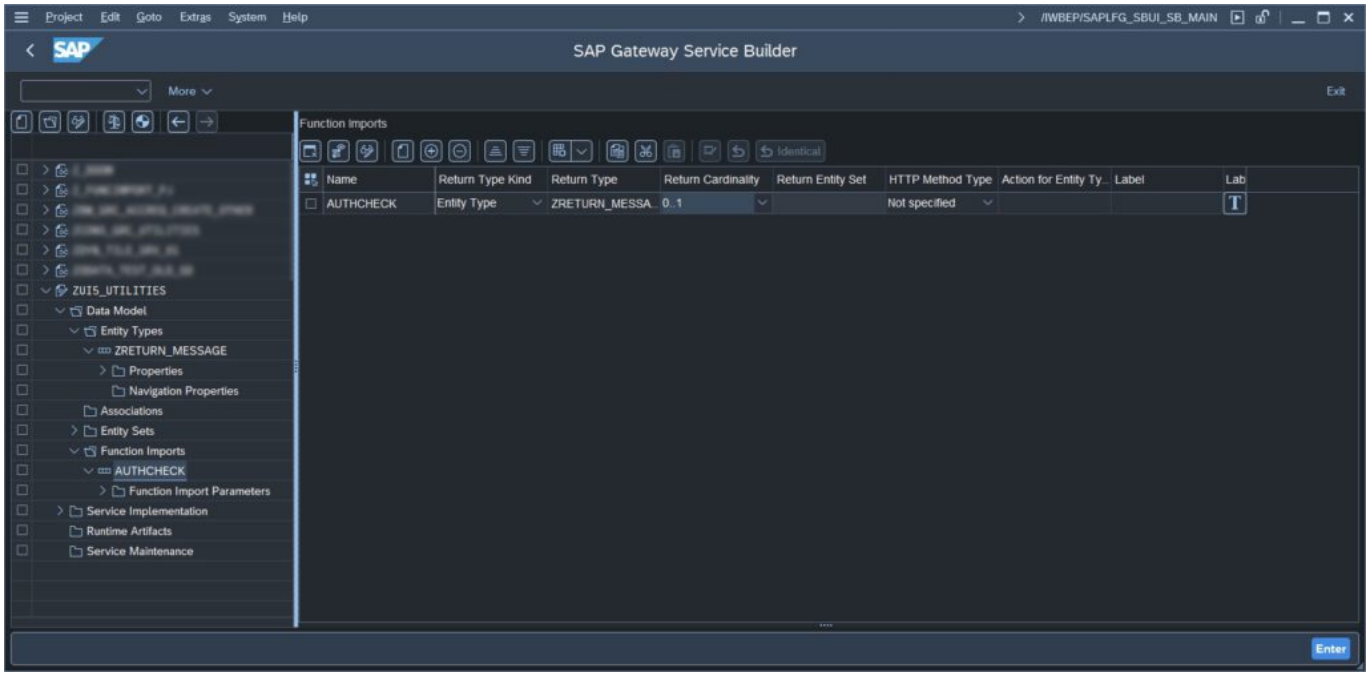
Entity Type ZRETURN_MESSAGE

Selbstverständlich dient die einfache Gestaltung einer Rückgabe lediglich der Veranschaulichung. In praktischen Anwendungen sollten sinnvolle Rückgabeparameter mit einer Steuerung von HTTP-Status-Codes kombiniert werden.



Property message des Entity Types ZRETURN_MESSAGE

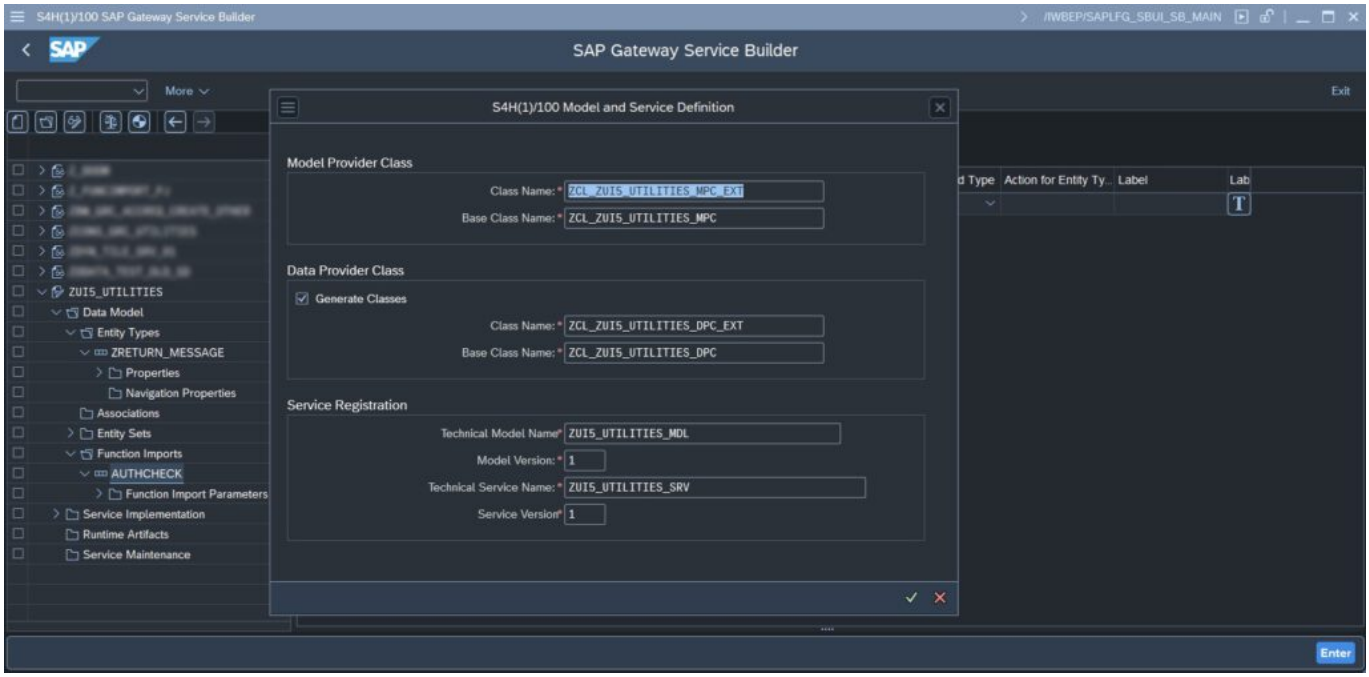
Ist der Entity Type definiert, kann dieser im Function Import hinterlegt werden. Bitte beachten, dass wir für diese Demonstration lediglich ZRETURN_MESSAGE und die Kardinalität 0..1 angegeben haben. Für praktische Implementierungen wird die „optionale“ Kardinalität nicht empfohlen, worauf einen das SAP-System auch gern und häufig hinweisen wird.



Return-Parameter im Function Import

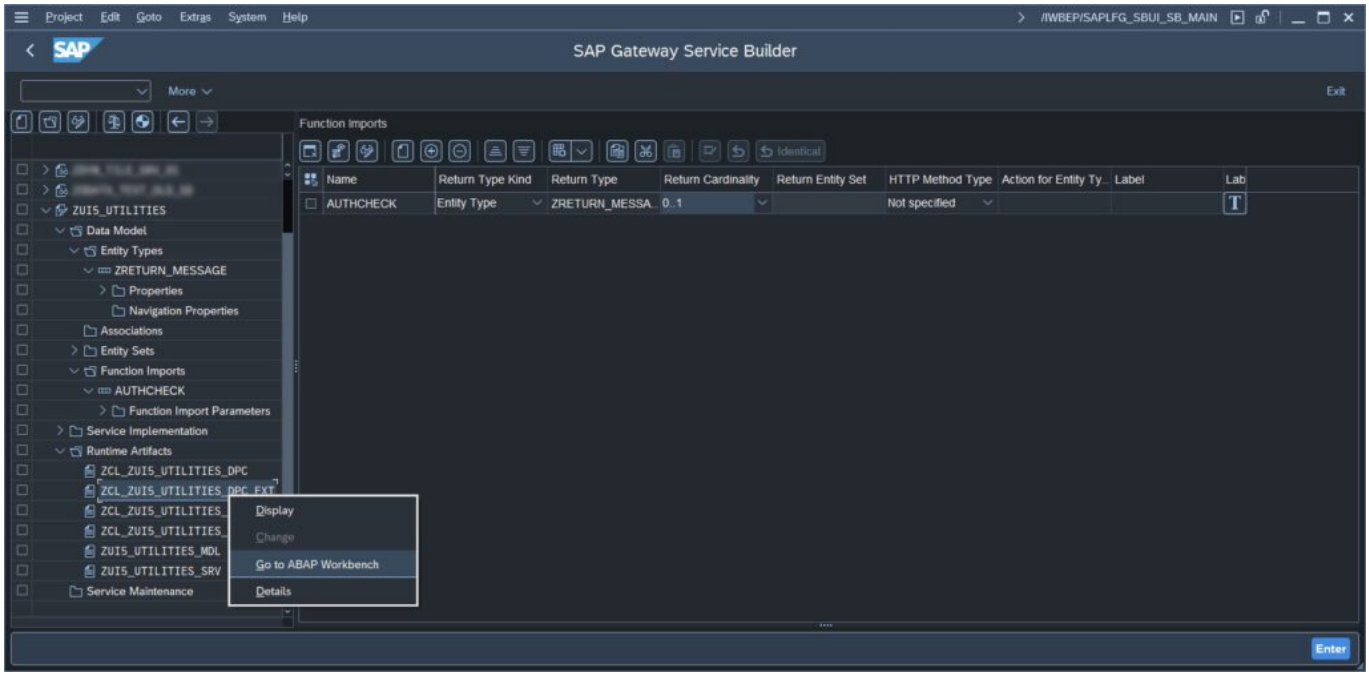
2. Generieren der ABAP-Klassen

Sind die Vorbereitungen erledigt, können wir den Gateway-Service generieren lassen. Dazu drücken wir das „BMW-Symbol“ und bestätigen die Vorschläge der zu generierenden Klassen.



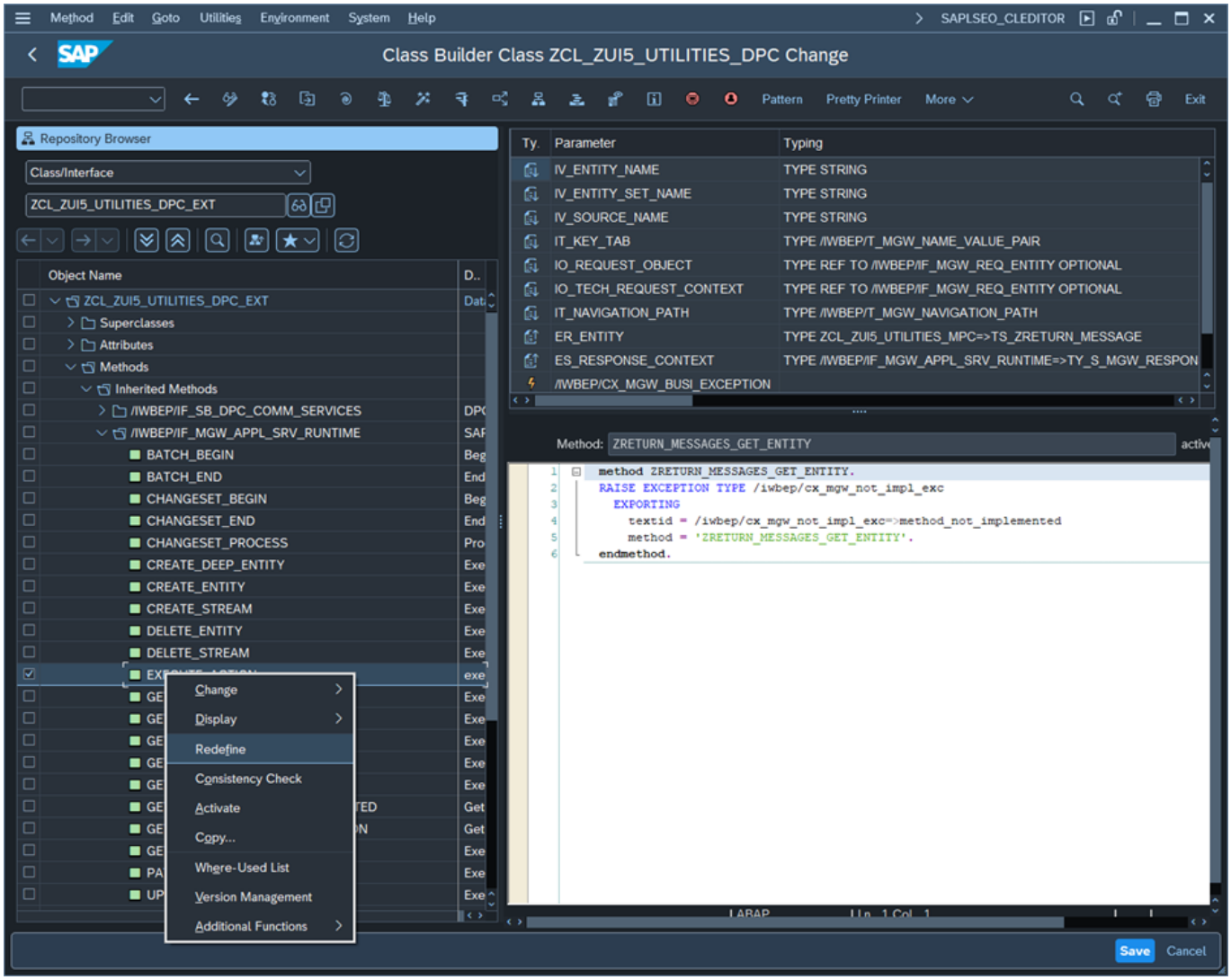
ABAP-Klassen generieren

Unter Umständen wird an dieser Stelle ein Fehler geworfen, welcher den Anwender auffordert, die Klassen nochmals zu generieren. Wir tun genau dieses und sollten anschließend lediglich Warnmeldungen erhalten. Ist die Generierung abgeschlossen, navigieren wir zu den Runtime-Artifacts und springen in die ABAP Workbench der *_DPC_EXT Klasse.



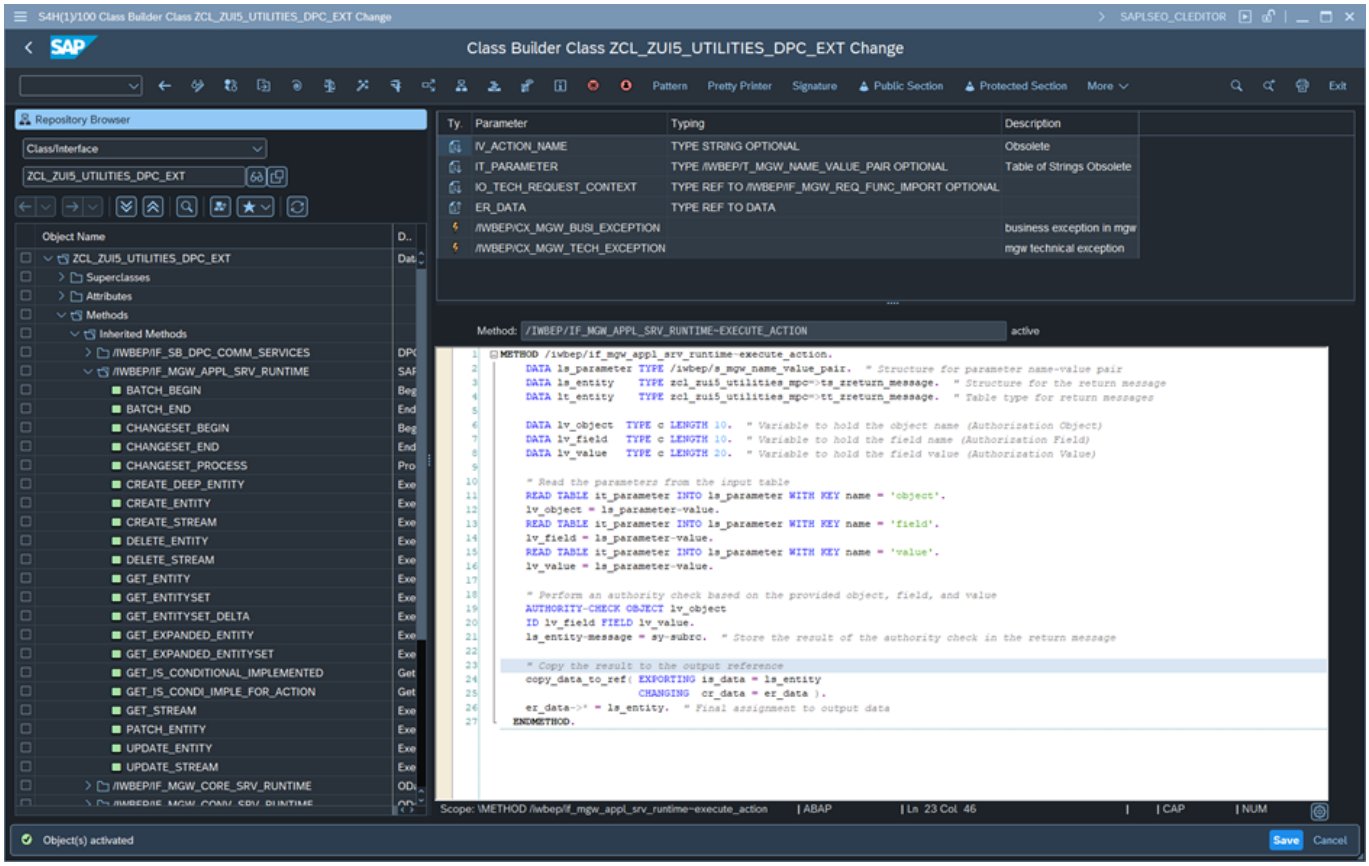
Absprung in die ABAP Workbench

In der Klasse /IWBEP/IF_MGW_APPL_SRV_RUNTIME redefinieren wir die Methode EXECUTE_ACTION, wie im folgenden Bild dargestellt.



Redefinition der Handler-Methode

Die Methode EXECUTE_ACTION führt, vereinfacht gesagt, die Aktion aus, die wir im Function Import als OData Service definiert haben. Über ABAP-Code ist es uns an dieser Stelle möglich, beliebige Logik zu implementieren. Der dargestellte ABAP-Code führt eine einfache ABAP-Berechtigungsprüfung mit genau einem Objekt, Feld und Wert durch und liefert den Return-Code als Message an den OData-Service zurück.



Einfache Berechtigungsprüfung

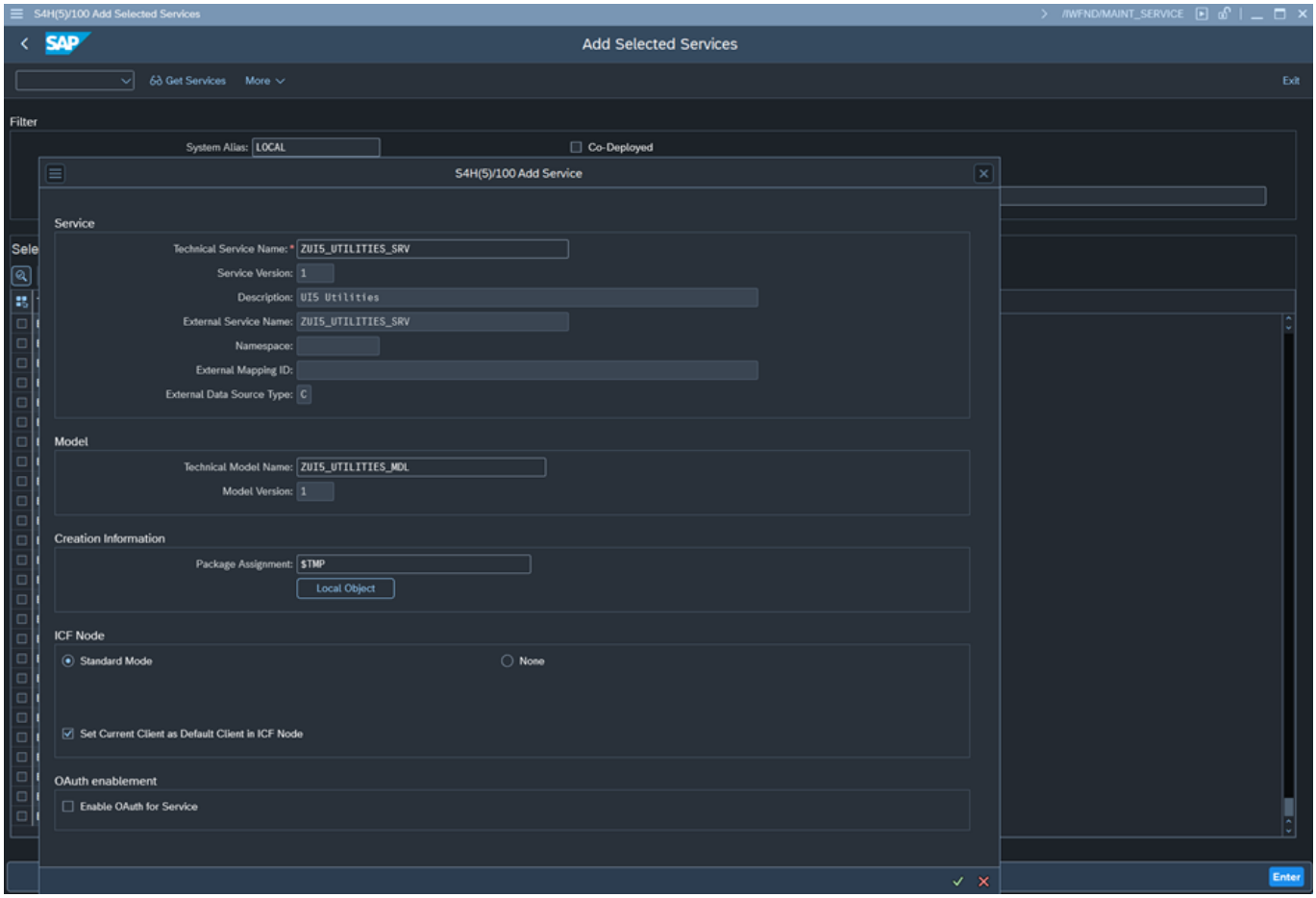
Anbei der Beispielcode. Bitte beachten, dass wir in einem der nächsten Teile auf den generischen Teil separat eingehen werden.

```
METHOD /iwbep/if_mgw_appl_srv_runtime~execute_action.
    DATA ls_parameter TYPE /iwbep/s_mgw_name_value_pair.  " Structure
for parameter name-value pair
    DATA ls_entity    TYPE zcl_zui5_utilities_mpc=>ts_zreturn_message.
" Structure for the return message
    DATA lt_entity    TYPE zcl_zui5_utilities_mpc=>tt_zreturn_message.
" Table type for return messages

    DATA lv_object  TYPE c LENGTH 10.  " Variable to hold the object
name (Authorization Object)
    DATA lv_field   TYPE c LENGTH 10.  " Variable to hold the field
name (Authorization Field)
    DATA lv_value   TYPE c LENGTH 20.  " Variable to hold the field
```

```
value (Authorization Value)

    " Read the parameters from the input table
    READ TABLE it_parameter INTO ls_parameter WITH KEY name =
'object'.
    lv_object = ls_parameter-value.
    READ TABLE it_parameter INTO ls_parameter WITH KEY name = 'field'.
    lv_field = ls_parameter-value.
    READ TABLE it_parameter INTO ls_parameter WITH KEY name = 'value'.
    lv_value = ls_parameter-value.

    " Perform an authority check based on the provided object, field,
and value
    AUTHORITY-CHECK OBJECT lv_object
    ID lv_field FIELD lv_value.
    ls_entity-message = sy-subrc.  " Store the result of the authority
check in the return message

    " Copy the result to the output reference
    copy_data_to_ref( EXPORTING is_data = ls_entity
                      CHANGING  cr_data = er_data ).
    er_data->* = ls_entity.  " Final assignment to output data
  ENDMETHOD.
```

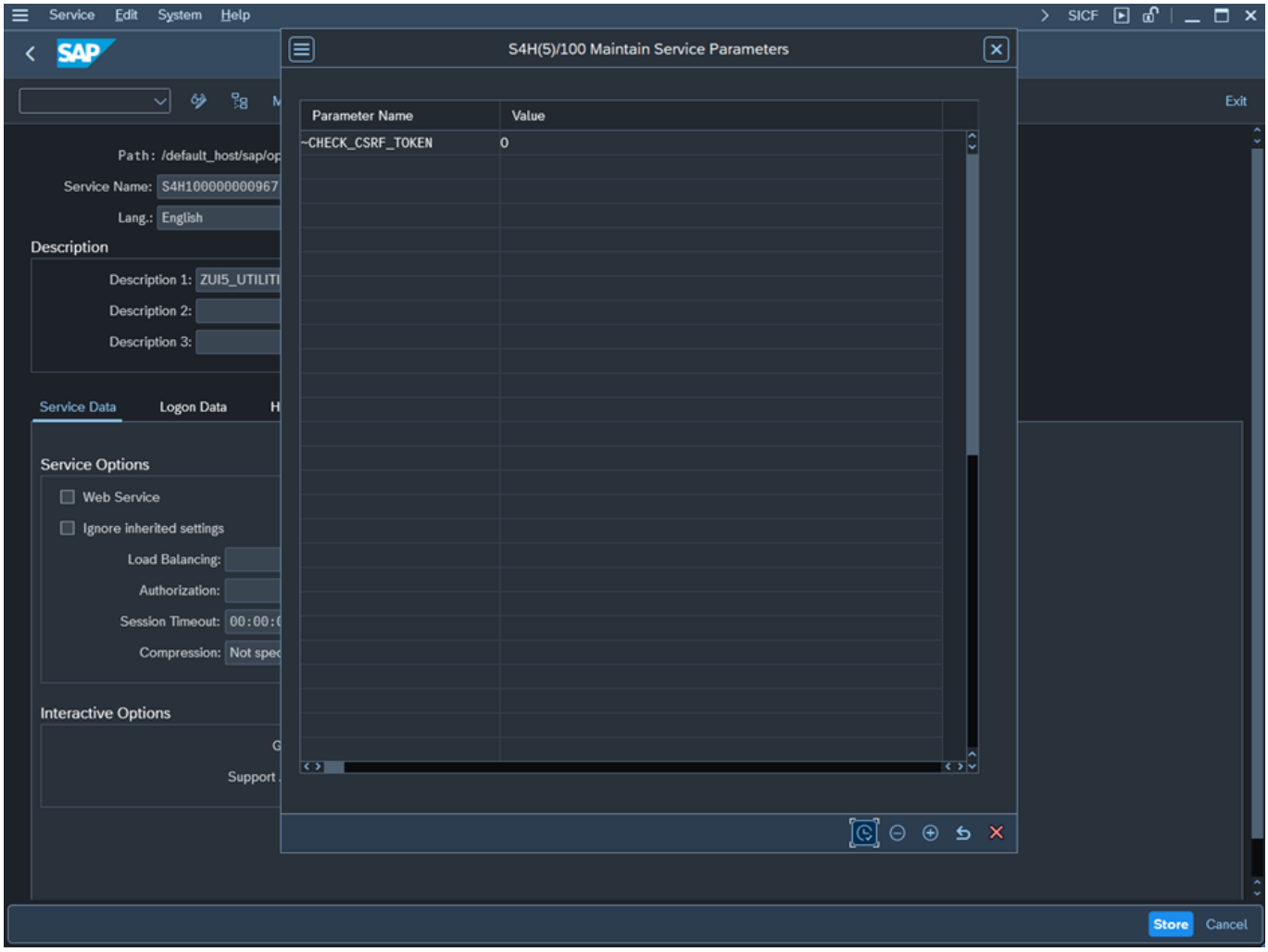
3. Registrieren des Services

Um den Service verwenden zu können, müssen wir ihn im ABAP-Gateway registrieren. Dazu navigieren wir in die Transaktion /IWFND/MAINT_SERVICE und klicken auf „Add Services“. In dem sich geöffneten Untermenü wählen wir den System Alias „LOCAL“ (Achtung: Abweichung möglich je nach Systemkonfiguration) und bestätigen die Auswahl. Anschließend müsste der erstellte Service auswählbar sein. Wir bestätigen die Auswahl und fügen den Service hinzu, ohne zusätzliche Einstellungen vorzunehmen.



Registrierung des Services in /IWFND/MAINT_SERVICE

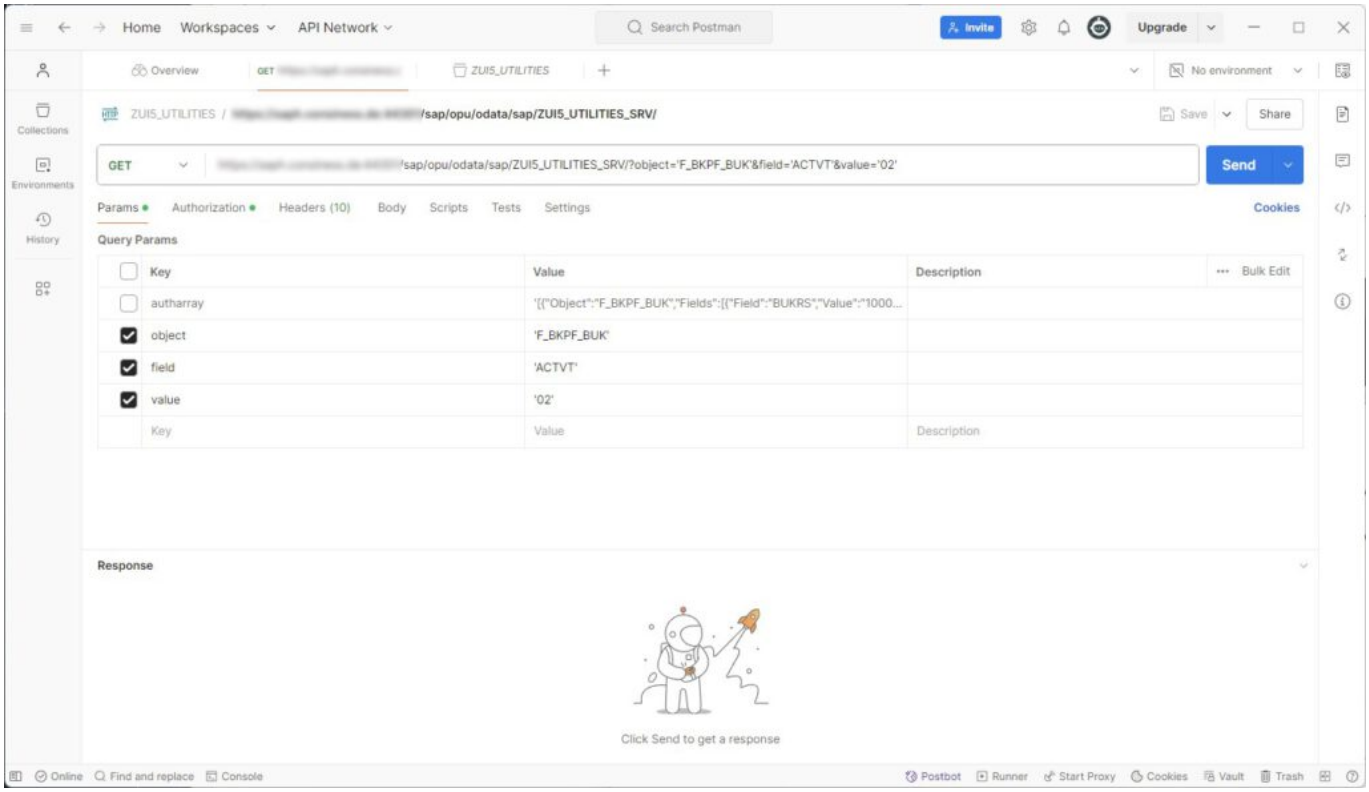
Ist der Service registriert, suchen wir ihn in der /IWFND/MAINT_SERVICE und öffnen im Kontext Menü unten den Link „Configure SICF-Service“. Im SICF-Service öffnen wir die „GUI-Configuration“ und tragen dort den Parameter ~CHECK_CSRF_TOKEN mit dem Wert 0 ein. Diese Einstellung deaktiviert die CSRF-Token Abfrage und ist notwendig, um den Service einfacher mit einem Tool wie Postman testen zu können.



SICF CSRF-Prüfung deaktivieren

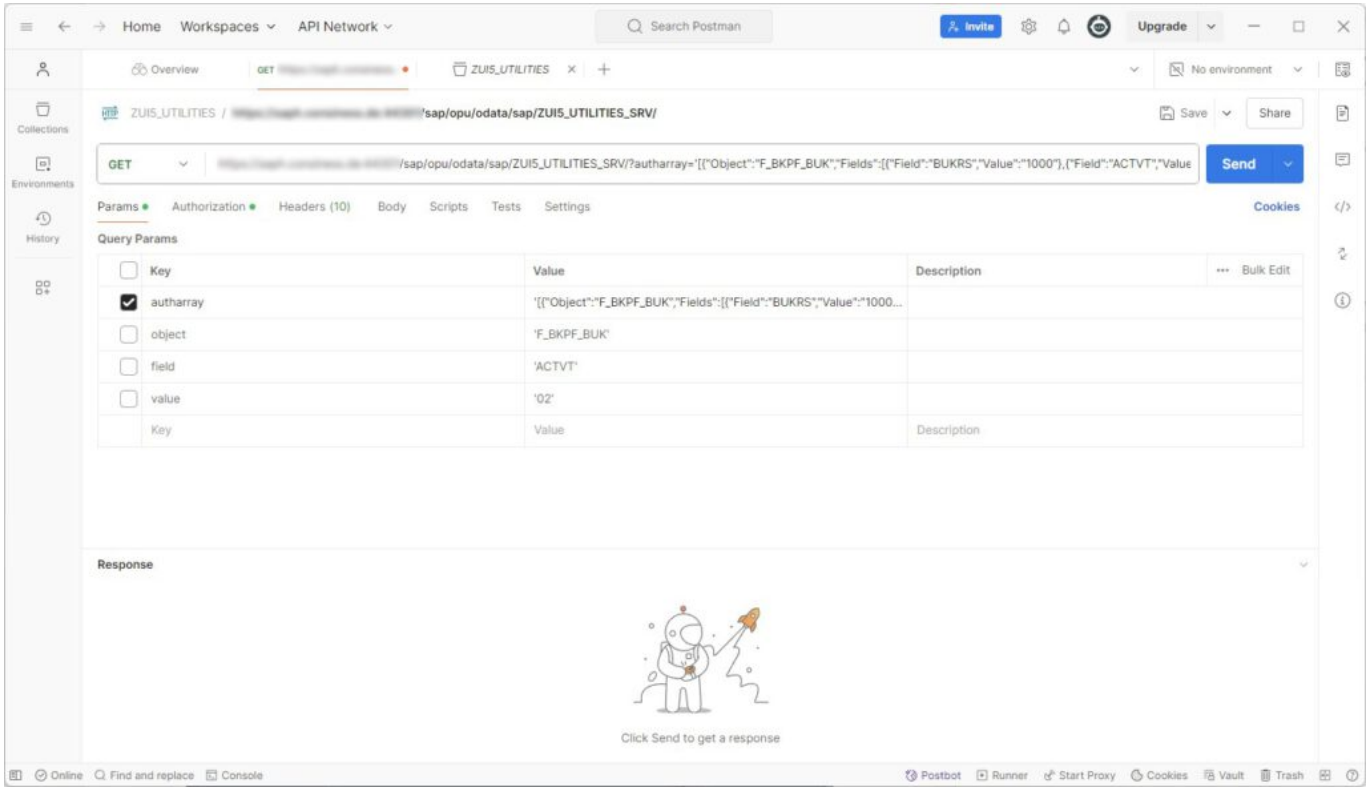
4. Testen des Services

Für den Test verwenden wir das Tool Postman, welches in der Basis-Version frei verfügbar heruntergeladen werden kann. Als Abfrage-Methode wählen wir GET und als Ziel-URL: <server-addresse>/sap/opu/odata/ZUI5_UTILITIES_SRV, welche im Normalfall gültig sein sollte. Falls dies nicht funktioniert, kann die korrekte Addresse über die /IWFND/MAINT_SERVICE oder SICF in Erfahrung gebracht werden (...).



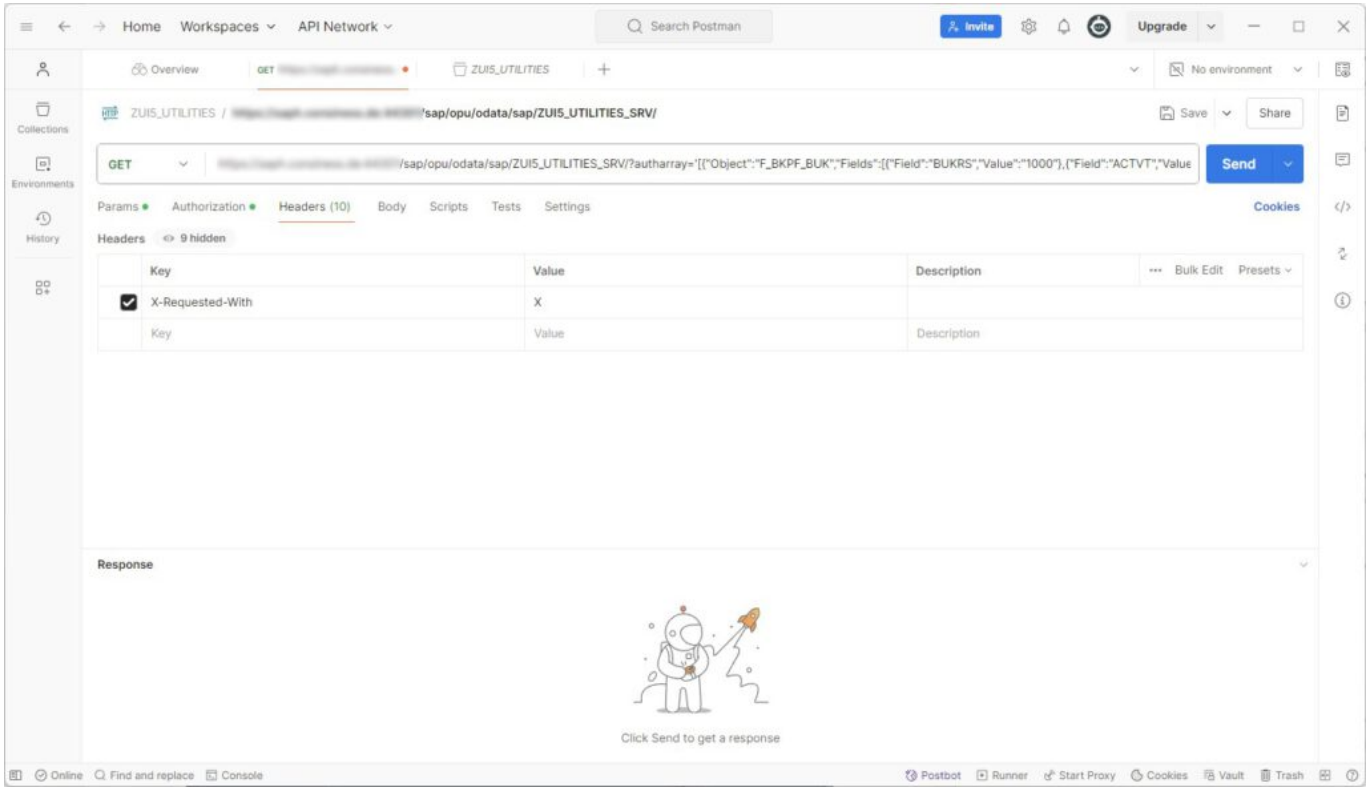
Test-Setup einfache Prüfung

Als Query-Parameter legen wir für das einfache Szenario eine Prüfung auf F_BKPF_BUK mit der ACTVT 02 fest. Bitte beachten, dass alle Übergabeparameter mit dieser Vorgehensweise über einfache Anführungsstriche als String deklariert werden müssen.



Test-Setup komplexe Prüfung

Damit keine CSRF-Fehler auftreten, ist es außerdem wichtig den Header X-Requested-With mit einem beliebigen Wert z.B. „X“ mitzusenden.



X-Requested-With Header zur Umgehung der CSRF-Prüfung

Sind alle genannten Schritte korrekt durchgeführt worden, sollten wir jetzt in der Lage sein, einen Request abzufeuern, welcher im Backend eine einfache Berechtigungsprüfung dessen Return-Code an das Frontend weiterleitet.

```
<?xml version="1.0" encoding="utf-8"?>
<entry xml:base="https://                    /sap/opu/odata/sap/ZUI5_UTILITIES_SRV/" xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
    <id>https://                    /sap/opu/odata/sap/ZUI5_UTILITIES_SRV/ZRETURN_MESSAGESet('0%20')</id>
    <title type="text">ZRETURN_MESSAGESet('0%20')</title>
    <updated>2024-10-07T13:06:44Z</updated>
    <category term="ZUI5_UTILITIES_SRV.ZRETURN_MESSAGE" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
    <link href="ZRETURN_MESSAGESet('0%20')" rel="self" title="ZRETURN_MESSAGE"/>
    <content type="application/xml">
        <m:properties>
            <d:message>0 </d:message>
        </m:properties>
    </content>
</entry>
```

Return Message als XML

Die Anwendung dieser Technik demonstrieren wir in Teil 2.

Über den Autor

Hendrik Winkler ist Partner der consiness und Lead Architekt für Identity und Access Management Lösungen. Er kann auf umfangreiche Expertise in SAP ABAP, GRC, Cloud-Technologien und SAP Identity Management zurückgreifen. Mit über zehn Jahren in der IT-Branche hat er sich auf die Entwicklung und Implementierung von komplexen IAM-Systemen spezialisiert, wobei er stets ein Auge auf Sicherheit, Benutzerfreundlichkeit und Compliance hat.

Der Artikel ist auch bei Linkedin erschienen:

https://www.linkedin.com/pulse/berechtigungen-zur-laufzeit-sap-fiori-apps-prüfen-hendrik-winkler-t52we/?trackingId=wDswEpGqSTKQNxEUIC%2BX7g%3D%3D