Stefan Schnell

SAP Application Server RFC Connection with UiPath

Table of Contents

Disclaimer

Document History

Version	Name	Date	Description
1.00	Stefan Schnell	01.11.2020	Initial document creation
1.01	Stefan Schnell	12.12.2020	Table with ABAP data types added
1.02	Stefan Schnell	22.03.2020	BAPI Source and explanation about DATS added
1.03	Stefan Schnell	03.04.2021	Restructuring, explanation about leading zeros and how to find an RFM added
1.04	Stefan Schnell	04.04.2021	ABAP example in Search via Description added, remove information about the Data Modeller because it is obsolete and new links in the List of References added
1.05	Stefan Schnell	01.05.2021	List of interesting transaction codes, tables, RFMs and programs added, Using System Trace to Monitor RFC Activities added, Debug ABAP Code from BAPI corrected, Modify XAML to use Installed NCo added
1.06	Stefan Schnell	11.06.2022	Deleted the description how to build a NuGet package with NCo, because this approach is obsolete. Advanced parameter AbapDebug correction and how to use SAP GUI from BAPI Activity added.

Abstract

This document describes various technical communication possibilities between an SAP back-end system and UiPath. The calls are based on the RFC interface of the SAP system. It describes the direct use of RFC calls as well as BAPI and ABAP usage. The calls to the SAP system are made by UiPath and the results of the function calls are used in UiPath further more. This is illustrated by some examples, also in comparison to the calls that are executed directly in the SAP system.

Application Programming Interfaces (APIs) to an SAP ERP system, which bases on RFC interface, especially the BAPI, are stable in a normal case. They are subject to a defined technical documentation, change guidelines and a versioning. In comparison to a graphical UI, the execution speed is very fast, too. These can be very good reasons for a possible using. However, precise knowledge of the technical interface, fields and data types is required for this kind of using.

Target Audience

The target audience of this document are Citizen Developer and RPA Developer who needs to connect to SAP back-end via RFC.

Software Requirements

The necessary software requirements are an installed SAP GUI for Windows, at least version 7.60, and an installed UiPath Studio. If you want to use the BAPI activity it is necessary to install SAP dotNET Connector x86 NCo too. In the SAP system you need the necessary rights to call the transaction codes and, for the use of ABAP, a developer key.

Important Hint

Do not experiment in production systems. Develop and test only in the designated development systems.

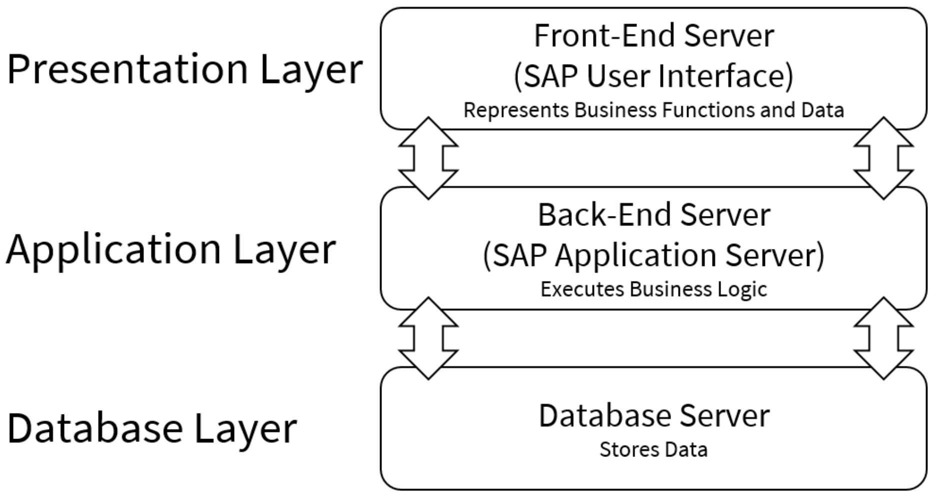
About the Author

Name:	Stefan Schnell
Age:	57
Employer:	BWI GmbH in Meckenheim, Germany, IT system house of the German Federal Armed Forces
Occupation:	Senior Systems Engineer in Service Architecture Department
Vocation:	The building of integration scenarios between front-end and SAP back-end environments, with different platforms, interfaces and programming languages.
Meet me at:	UiPath Forum SAP Community LinkedIn Private Site

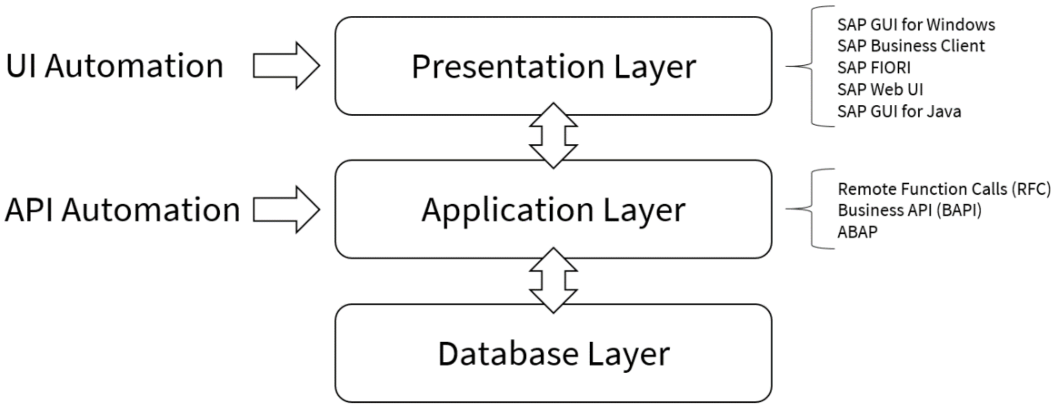
Three-Tier Architecture and its Automation Approaches

The multitier architecture is a client-server architecture which is split into multiple physically separated layers. This reduces the complexity of dependencies within a system. This has advantages, on the one hand for the understanding and the other hand for the maintenance of the system. Individual layers can be easily exchanged, without having effects to the whole system.

One approach is the Three-Tier Architecture, which is used by SAP. The presentation layer is the front-end, where the user works with. The application layer, also known as back-end server, contains the business logic. The database layer, also known as database server, stores the information and delivers the data to the business logic on demand.



Robotic Process Automation (RPA), as a non-invasive technology, works primarily with the presentation layer, based on the User Interface (UI). But there is also the possibility to communicate with the application layer via an Application Programming Interface (API).

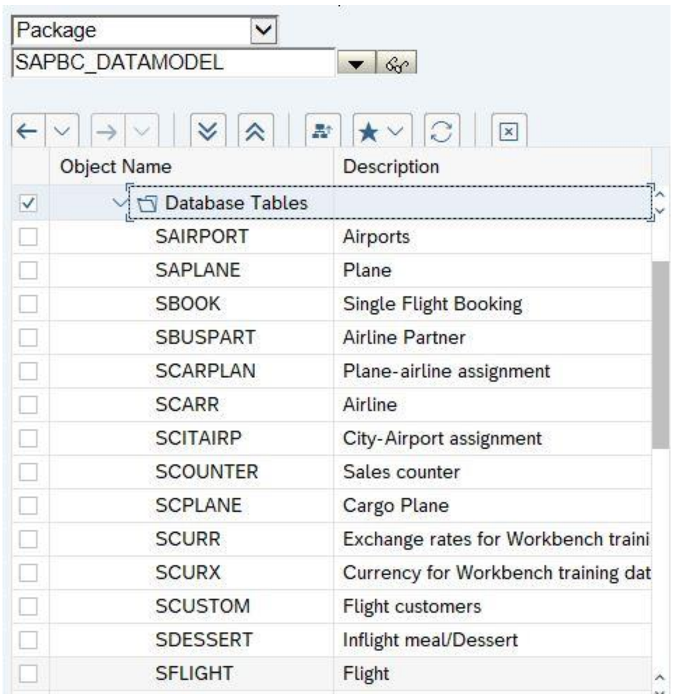


The communication with the application layer is realized via a technical interface.

SFLIGHT Example

Demo Data Model SFLIGHT

The SAP data model SFLIGHT is for training purposes. It is an ideal playground for any kinds of experiments. All examples in this document use this data model. The development objects are in the packages SAPBC_DATAMODEL and SAPBC_BOR, among others. You can open one of these packages in the Object Navigator, with the transaction code (TAC) SE80, to have a closer look at the development objects. Here a few data tables:



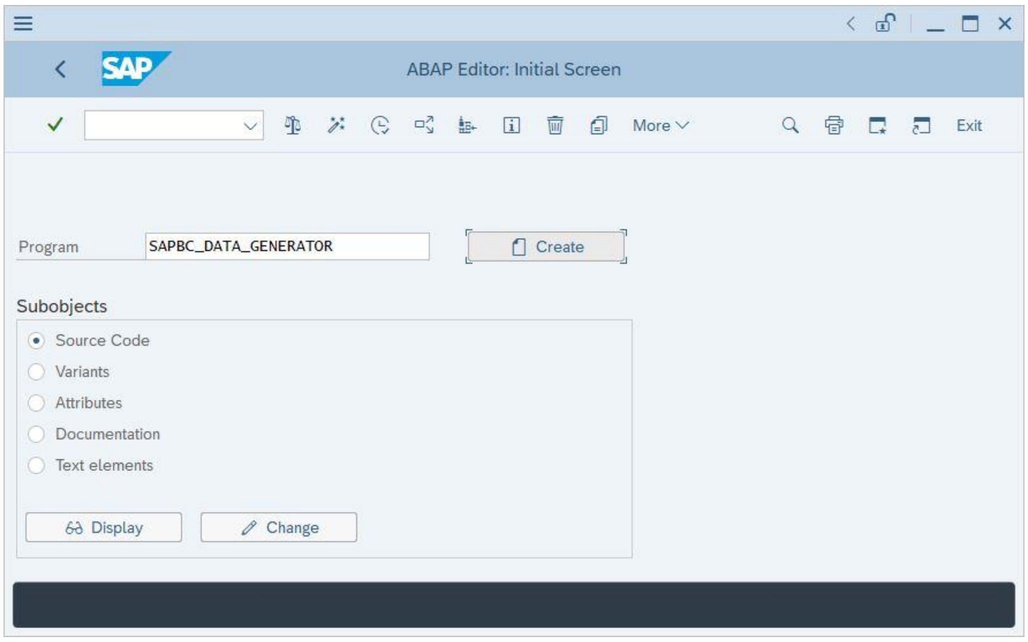
Package
SAPBC_DATAMODEL

Object Name	Description
Database Tables	
SAIRPORT	Airports
SAPLANE	Plane
SBOOK	Single Flight Booking
SBUSPART	Airline Partner
SCARPLAN	Plane-airline assignment
SCARR	Airline
SCITAIRP	City-Airport assignment
SCOUNTER	Sales counter
SCPLANE	Cargo Plane
SCURR	Exchange rates for Workbench traini
SCURX	Currency for Workbench training dat
SCUSTOM	Flight customers
SDESSERT	Inflight meal/Dessert
SFLIGHT	Flight

Hint: The Data Modeller (TAC SD11) is obsolete.

Generate Data in SFLIGHT Tables

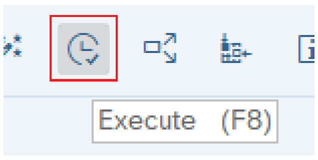
In a normal case contains the table of the SFLIGHT data model no data. To create data, call with the ABAP Editor (TAC SE38) the report SAPBC_DATA_GENERATOR.

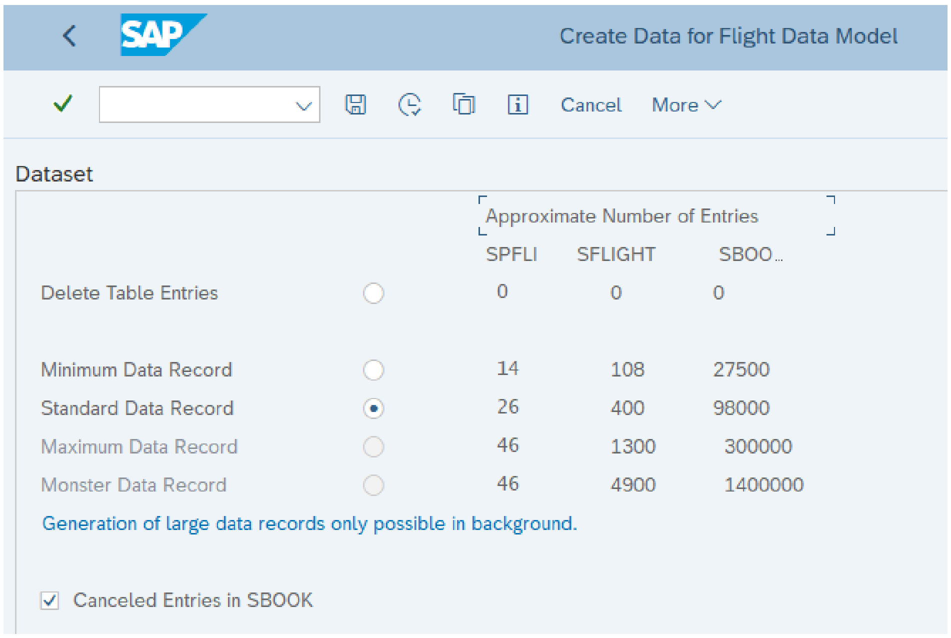


Program: SAPBC_DATA_GENERATOR

Subobjects
- [✓ Source Code]
- Variants
- Attributes
- Documentation
- Text elements

Press the Execute button in the toolbar, or the key F8.





Dataset

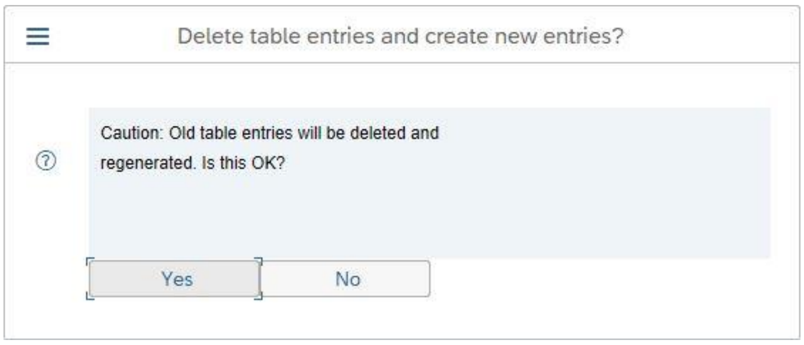
		Approximate Number of Entries		
		SPFLI	SFLIGHT	SBOO...
Delete Table Entries	○	0	0	0
Minimum Data Record	○	14	108	27500
Standard Data Record	●	26	400	98000
Maximum Data Record	○	46	1300	300000
Monster Data Record	○	46	4900	1400000

Generation of large data records only possible in background.

☑ Canceled Entries in SBOOK

Press Execute again to create data.

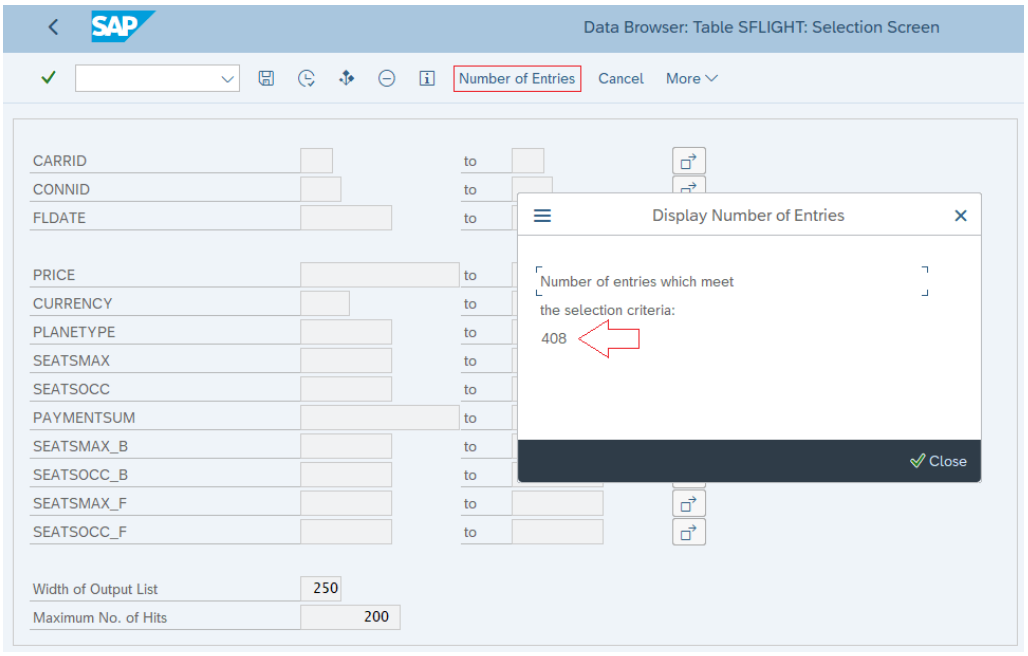
Press Yes button to accept the deletion of existing data.



Delete table entries and create new entries?

Caution: Old table entries will be deleted and regenerated. Is this OK?

Check with Data Browser (TAC SE16) the number of entries in table SFLIGHT.



Data Browser: Table SFLIGHT: Selection Screen

CARRID		to	
CONNID		to	
FLDATE		to	
PRICE		to	
CURRENCY		to	
PLANETYPE		to	
SEATSMAX		to	
SEATSOCC		to	
PAYMENTSUM		to	
SEATSMAX_B		to	
SEATSOCC_B		to	
SEATSMAX_F		to	
SEATSOCC_F		to	

Width of Output List 250
Maximum No. of Hits 200

Display Number of Entries

Number of entries which meet the selection criteria:
408

Each SAP ABAP system contains the SFLIGHT demo model for experimental purposes. It includes a wide area of different development objects which can be used for this, e.g. like tables, function modules, reports, UIs etc. This offers an optimal basis.

Hint: In some cases, special authorization objects are required. If the desired result is not achieved, then the Authorization Check (TAC SU53) can be used to determine whether missing authorizations are the reason.

Different SAP Application Layer Automation Approaches

The following automation approaches uses one technical method, RFC.

RFC

Remote Function Calls (RFC), also known as Remote Procedure Calls (RPC) or Remote-Enabled Function Modules (RFM), is the standard SAP interface for technical communication between SAP systems. RFC calls a function to be executed in a remote system. RFC is an SAP specific protocol and bases on CPI-C (Common Programming Interface for Communication), developed by IBM.

RFC is used with UiPath via a library called SAP dotNET Connector NCo with the Invoke Code activity.

Execute Remote-Enabled Function Modules from UiPath

BAPI

The Business Application Programming Interface (BAPI) is a formal defined interface to the business object model and uses RFC. In comparison to the RFM there are some implementation restrictions for BAPI function modules, e.g.:

- BAPIs must not execute 'COMMIT WORK' commands.
- BAPIs must not produce any screen output.
- No exceptions are used in BAPIs, all error messages must be returned in BAPIRET structure. How to Read BAPIRET2 Structure Easily
- …

BAPI is used with UiPath via an activity called UiPath.SAP.BAPI.Activities.

Execute BAPI Function Modules from UiPath

ABAP

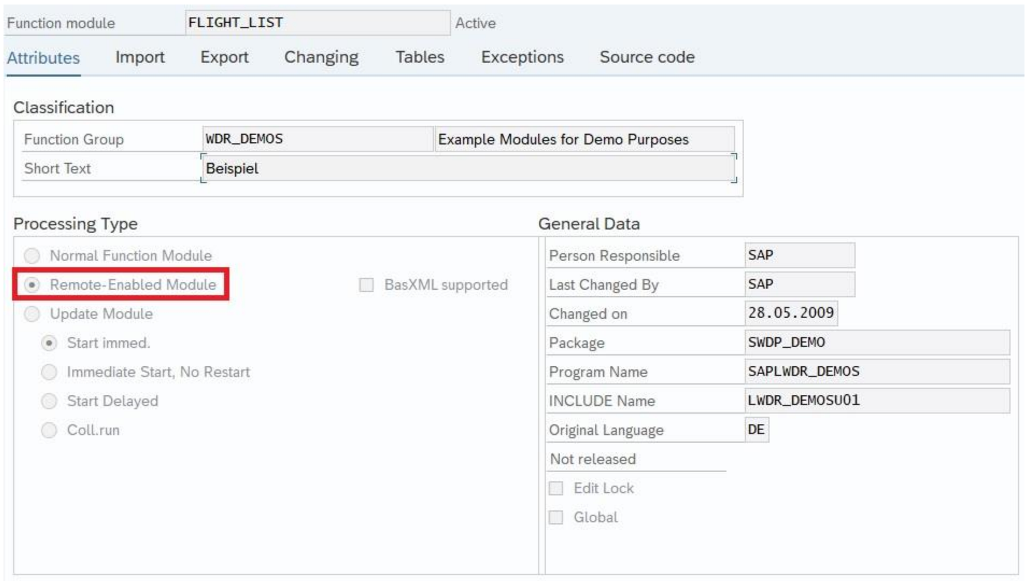
Advanced **B**usiness **A**pplication **P**rogramming language (ABAP, formerly known as **A**llgemeiner **B**erichts-**A**ufbereitungs-**P**rozessor) is the programming language in SAP environments. The interface to use ABAP with UiPath is the RFM RFC_ABAP_INSTALL_AND_RUN.

ABAP is used with UiPath via an activity called ABAPRunner.Activities.

Execute ABAP Code from UiPath

Execute Remote-Enabled Function Modules from UiPath

Function modules are modularization elements in the ABAP programming language. They encapsulate some functions that can be reused. They provide an interface to pass data to and from the function module. Remote function enabled modules are special flagged function modules which can be called and executed from outside of the application server.



Function module FLIGHT_LIST Active

Classification

Function Group	WDR_DEMOS	Example Modules for Demo Purposes
Short Text	Beispiel	

Processing Type

- Normal Function Module
- [✓ Remote-Enabled Module]
- BasXML supported
- Update Module
  - [✓ Start immed.]
  - Immediate Start, No Restart
  - Start Delayed
  - Coll.run

General Data

Person Responsible	SAP
Last Changed By	SAP
Changed on	28.05.2009
Package	SWDP_DEMO
Program Name	SAPLWDR_DEMOS
INCLUDE Name	LWDR_DEMOSU01
Original Language	DE

Not released

- Edit Lock
- Global

To communicate with the SAP Application Server via RFC you can use the SAP dotNETConnector (NCo). You can find a description how to install NCo in the post BAPI Functionality in UiPath.

Hint: It could be possible that it is necessary to install the Microsoft Visual C++ Redistributables.

Hint: If you do not have access to the SAP Support Portal you can collect the necessary files manually. With the standard installation of the SAP GUI for Windows you can find the NCo libraries in the Global Assembly Cache (GAC). Look in the directory C:\Windows\Microsoft.NET\assembly\GAC_32 and here you can find the libraries sapnco and sapnco_utils. The additional library libicudecnumber.dll is in the path C:\Program Files (x86)\Common Files\SAP Shared\Kernel_753 and rscp4n.dll is also in the GAC C:\Windows\Microsoft.NET\assembly\GAC_32\rscp4n.

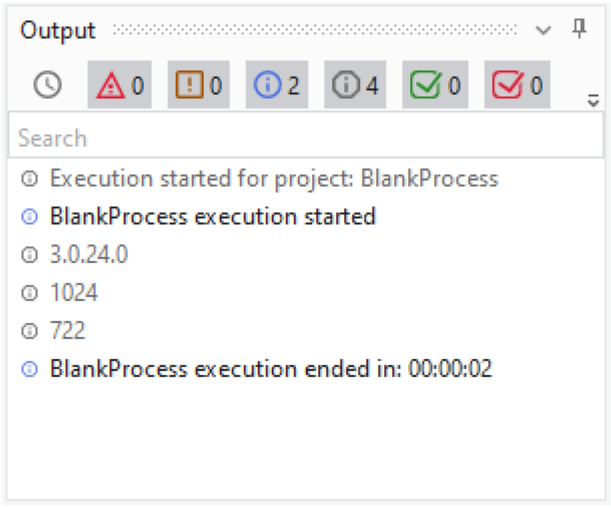
Hint: If you collect the libraries from an SAP GUI for Windows installation, it is not necessary to install an additional Microsoft Visual C++ Redistributable. The necessary Redistributable is available with the installation of the SAP GUI for Windows. In this case it is absolutely necessary to install an SAP GUI for Windows on each target system.

After the installation of the NCo modify the XAML file of your project. Add a variable from the type `SAP.Middleware.Connector.SAPConnectorInfo` to your project.

Add an Invoke Code activity to the workflow and try the following VB.NET code …

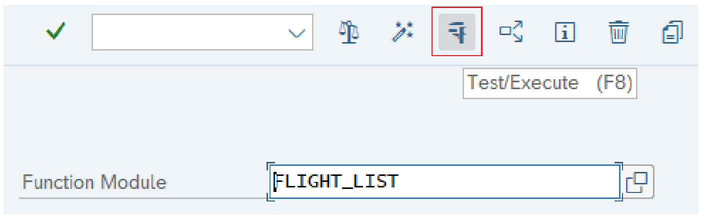
```
'-Begin---------------------------------------------------------------

Dim Version As String
Dim PatchLevel As Integer
Dim Release As String

Version = SAP.Middleware.Connector.SAPConnectorInfo.Version
PatchLevel = SAP.Middleware.Connector.SAPConnectorInfo.KernelPatchLevel
Release = SAP.Middleware.Connector.SAPConnectorInfo.KernelRelease

Console.WriteLine(Version)
Console.WriteLine(PatchLevel.ToString())
Console.WriteLine(Release)

'-End-----------------------------------------------------------------
```

… and you should see in the output window this result, when you execute the workflow.



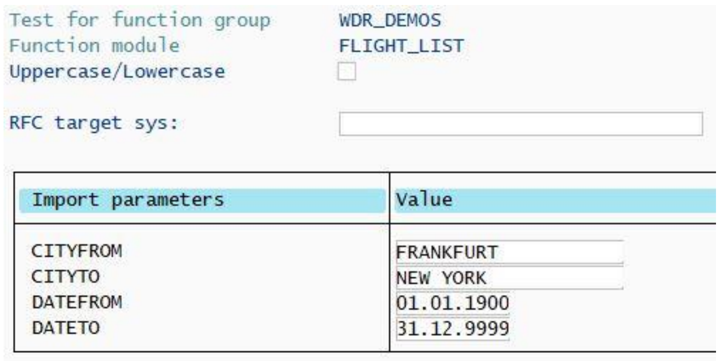
```
Execution started for project: BlankProcess
BlankProcess execution started
3.0.24.0
1024
722
BlankProcess execution ended in: 00:00:02
```

Now we try in the Function Builder (TAC SE37) the RFM FLIGHT_LIST. Press button Execute in the toolbar, or key F8.



Function Module	FLIGHT_LIST

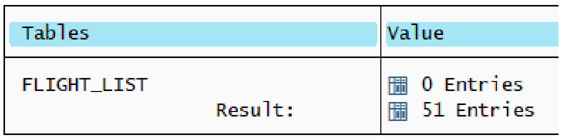
Add in the import parameters the necessary entries and press button Execute in the toolbar, or key F8.



Test for function group	WDR_DEMOS
Function module	FLIGHT_LIST
Uppercase/Lowercase	☐
RFC target sys:	

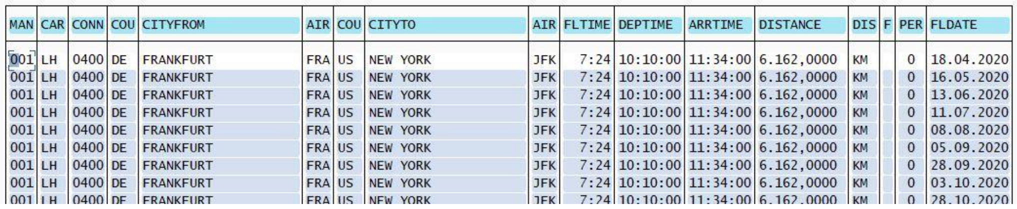
Import parameters	Value
CITYFROM	FRANKFURT
CITYTO	NEW YORK
DATEFROM	01.01.1900
DATETO	31.12.9999

Double click in the table FLIGHT_LIST the result entries …



Tables	Value
FLIGHT_LIST	0 Entries
Result:	51 Entries

… to see the delivered data.



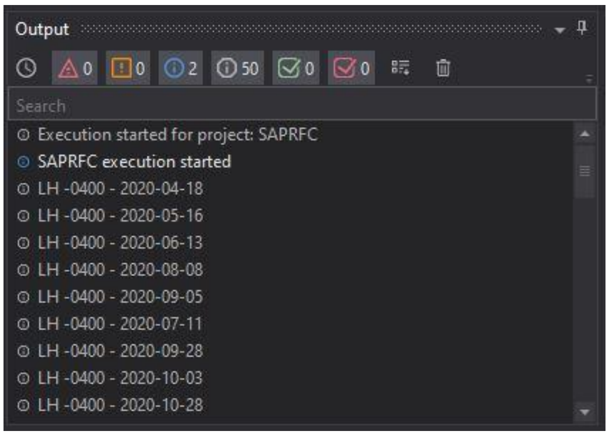
MAN	CAR	CONN	COU	CITYFROM	AIR	COU	CITYTO	AIR	FLTIME	DEPTIME	ARRTIME	DISTANCE	DIS	F	PER	FLDATE
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	18.04.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	16.05.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	13.06.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	11.07.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	08.08.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	05.09.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	28.09.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	03.10.2020
001	LH	0400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162,0000	KM		0	28.10.2020

Now we want to do the same with UiPath.

To do the same in UiPath use an Invoke Code activity with the following VB.NET code …

```
'-Begin-----------------------------------------------------------------

Dim cfgParams As SAP.Middleware.Connector.RfcConfigParameters
Dim destination As SAP.Middleware.Connector.RfcDestination
Dim rfcFunction As SAP.Middleware.Connector.IRfcFunction
Dim FlightList As SAP.Middleware.Connector.IRfcTable
Dim Line As SAP.Middleware.Connector.IRfcStructure

cfgParams = New SAP.Middleware.Connector.RfcConfigParameters
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.Name, "Test" )
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.AppServerHost, _
  "ABAP702")
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.SystemNumber, "00")
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.Client, "001")
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.User, "BCUSER")
cfgParams.Add(SAP.Middleware.Connector.RfcConfigParameters.Password, "minisap")

destination =
SAP.Middleware.Connector.RfcDestinationManager.GetDestination(cfgParams)

rfcFunction = destination.Repository.CreateFunction("FLIGHT_LIST")
rfcFunction.SetValue("CITYFROM", "FRANKFURT")
rfcFunction.SetValue("CITYTO", "NEW YORK")
rfcFunction.SetValue("DATEFROM", "19000101")
rfcFunction.SetValue("DATETO", "99991231")
rfcFunction.Invoke(destination)
FlightList = rfcFunction.GetTable("FLIGHT_LIST")

For Each Line In FlightList
  Console.WriteLine( _
    Line.GetValue("CARRID").ToString.Trim & " - " & _
    Line.GetValue("CONNID").ToString.Trim & " - " & _
    Line.GetValue("FLDATE").ToString.Trim _
  )
Next

'-End-------------------------------------------------------------------
```

… to see this result, when you execute the workflow.



```
Execution started for project: SAPRFC
SAPRFC execution started
LH -0400 - 2020-04-18
LH -0400 - 2020-05-16
LH -0400 - 2020-06-13
LH -0400 - 2020-08-08
LH -0400 - 2020-09-05
LH -0400 - 2020-07-11
LH -0400 - 2020-09-28
LH -0400 - 2020-10-03
LH -0400 - 2020-10-28
```

How to Find an RFM

Open the Data Browser (TAC SE16) and open table TFDIR. This table contains all function modules of an SAP system.

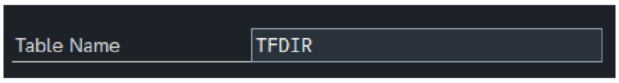
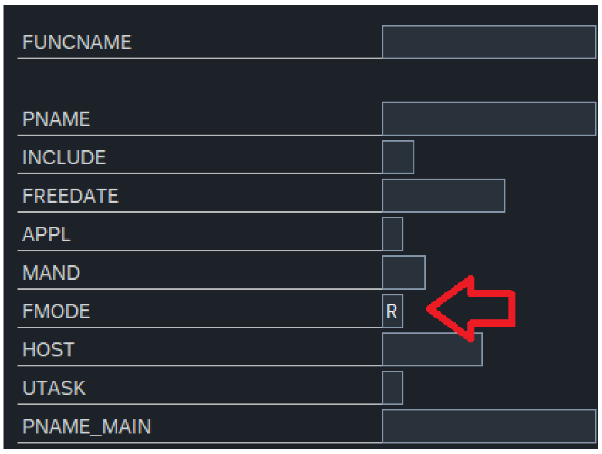


Table Name	TFDIR

In the selection screen type in field FMODE (Type of Function Module) an R, which means this function module is remote enabled.



FUNCNAME	
PNAME	
INCLUDE	
FREEDATE	
APPL	
MAND	
FMODE	R
HOST	
UTASK	
PNAME_MAIN	

Add the information you know in the available fields to find the RFM you are searching for.

Search via Description

Open the Data Browser (TAC SE16) and open table TFTIT. This table contains from all function modules the short text.

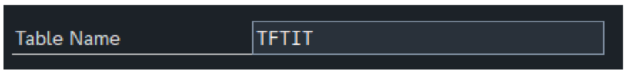
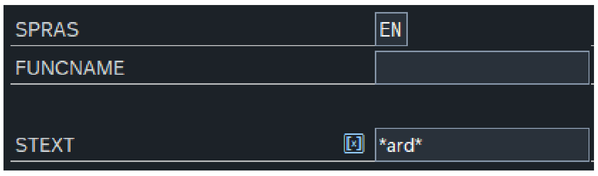


Table Name	TFTIT

In the selection screen you have the possibility to add the language key (SPRAS) and text, text fragments and wildcards (STEXT). On this way a function module can be found via the description. But in a second step, via table TFDIR, it is necessary to detect if the function module is remote enabled, as described above.



SPRAS	EN
FUNCNAME	
STEXT	*ard*

Here an approach how to realize a search via short description, on remote enabled function modules, in ABAP:

```
TYPES: BEGIN OF ty_FuncMod,
         FUNCNAME TYPE RS38L_FNAM,
         STEXT    TYPE RS38L_FTXT,
       END OF ty_FuncMod.

DATA:
  lt_FuncMod TYPE STANDARD TABLE OF ty_FuncMod
  .

SELECT ShortText~FUNCNAME ShortText~STEXT
  FROM TFTIT AS ShortText
  INNER JOIN TFDIR AS FuncMod ON ShortText~FUNCNAME = FuncMod~FUNCNAME
  INTO CORRESPONDING FIELDS OF TABLE lt_FuncMod
  WHERE FuncMod~FMODE = 'R' AND
        ShortText~SPRAS = 'E' AND
        ShortText~STEXT LIKE '%ard%'.
```

The result is available in the internal table lt_FuncMod and each short text contains *ard*, e.g. as C*ard*, H*ard*ware, Dashbo*ard* or Wiz*ard*.

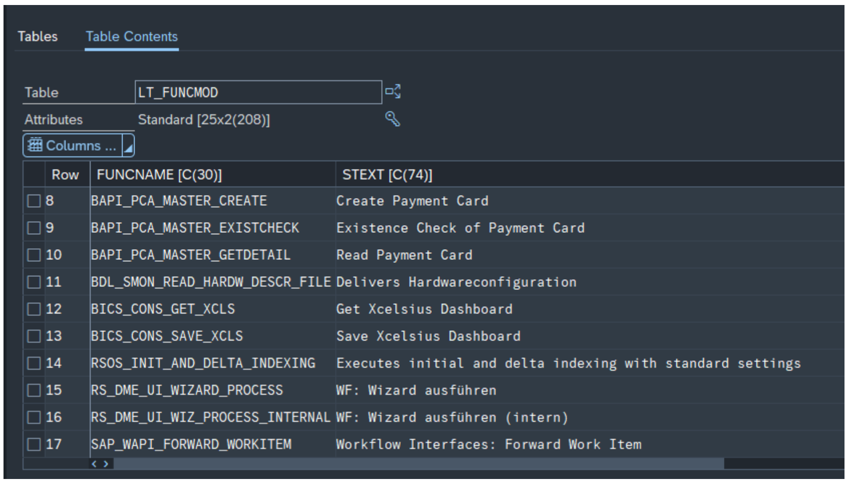


Table: LT_FUNCMOD
Attributes: Standard [25x2(208)]

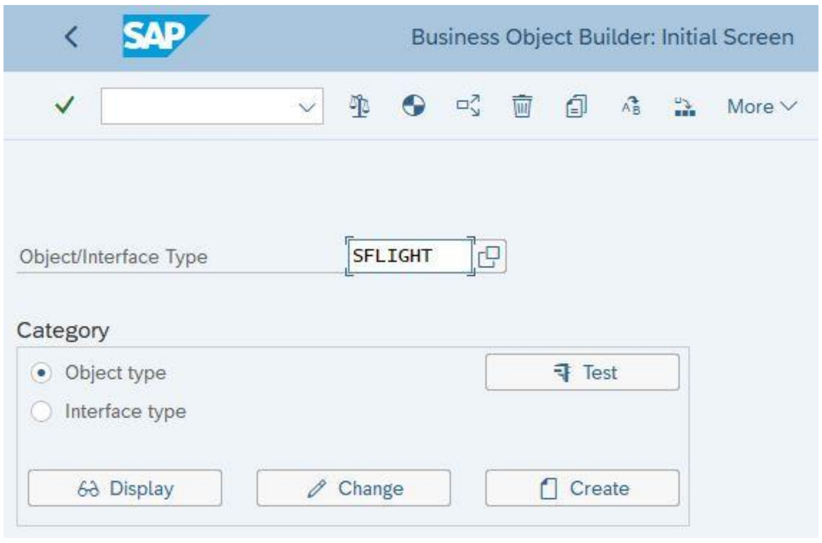
Row	FUNCNAME [C(30)]	STEXT [C(74)]
8	BAPI_PCA_MASTER_CREATE	Create Payment Card
9	BAPI_PCA_MASTER_EXISTCHECK	Existence Check of Payment Card
10	BAPI_PCA_MASTER_GETDETAIL	Read Payment Card
11	BDL_SMON_READ_HARDW_DESCR_FILE	Delivers Hardwareconfiguration
12	BICS_CONS_GET_XCLS	Get Xcelsius Dashboard
13	BICS_CONS_SAVE_XCLS	Save Xcelsius Dashboard
14	RSOS_INIT_AND_DELTA_INDEXING	Executes initial and delta indexing with standard settings
15	RS_DME_UI_WIZARD_PROCESS	WF: Wizard ausführen
16	RS_DME_UI_WIZ_PROCESS_INTERNAL	WF: Wizard ausführen (intern)
17	SAP_WAPI_FORWARD_WORKITEM	Workflow Interfaces: Forward Work Item

Hint: This kind of search is case sensitive, %card% and %Card% are different and delivers different results.

Execute BAPI Function Modules from UiPath

BAPI is a standardized programming interfaces enabling external applications to access business processes and data in SAP systems. BAPIs are defined in the Business Object Repository (BOR). You can find more information in the BAPI Programming Guide (CA-BFA) at SAP Help Portal.

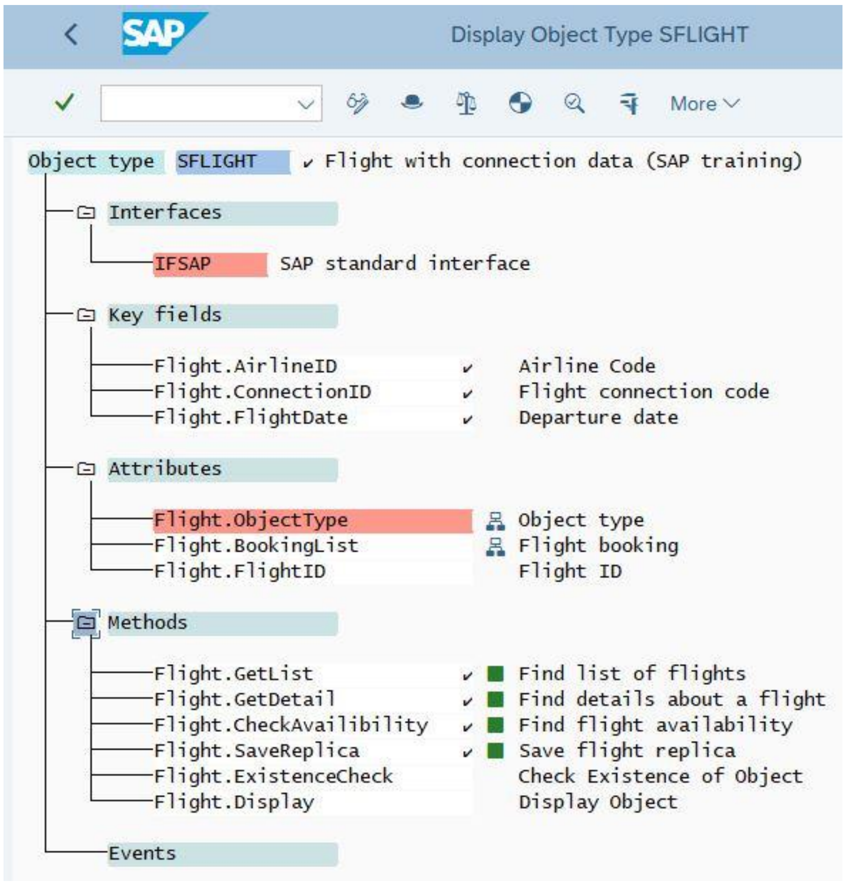
Open in the Business Object Builder (TAC SWO1) the Business Object (BO) SFLIGHT.



Object/Interface Type: SFLIGHT

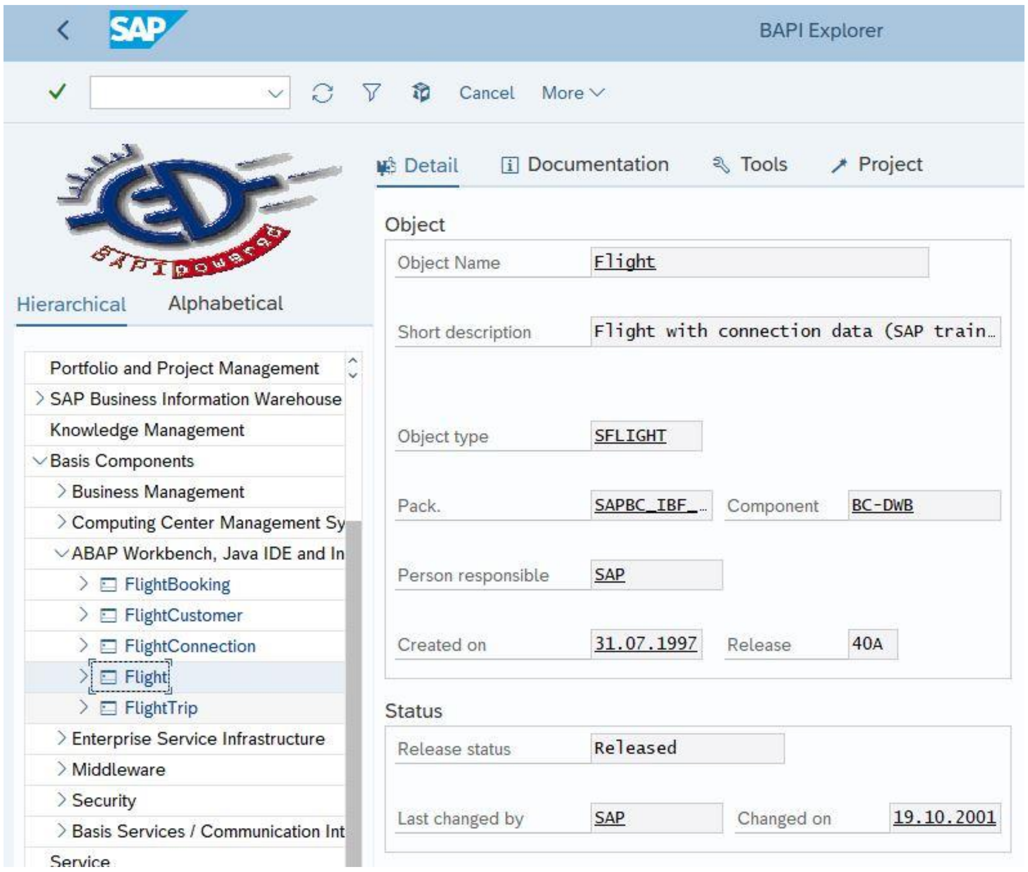
Category
Object type [✓]
Interface type

Now you can view e.g. all attributes and methods etc.



```
Object type  SFLIGHT   ✓ Flight with connection data (SAP training)
  Interfaces
      IFSAP              SAP standard interface
  Key fields
      Flight.AirlineID               ✓    Airline Code
      Flight.ConnectionID            ✓    Flight connection code
      Flight.FlightDate              ✓    Departure date
  Attributes
      Flight.ObjectType                   Object type
      Flight.BookingList                  Flight booking
      Flight.FlightID                     Flight ID
  Methods
      Flight.GetList                 ✓ ■  Find list of flights
      Flight.GetDetail               ✓ ■  Find details about a flight
      Flight.CheckAvailibility       ✓ ■  Find flight availability
      Flight.SaveReplica             ✓ ■  Save flight replica
      Flight.ExistenceCheck               Check Existence of Object
      Flight.Display                      Display Object
  Events
```

Open in the BAPI Explorer (TAC BAPI) with the object Flight.



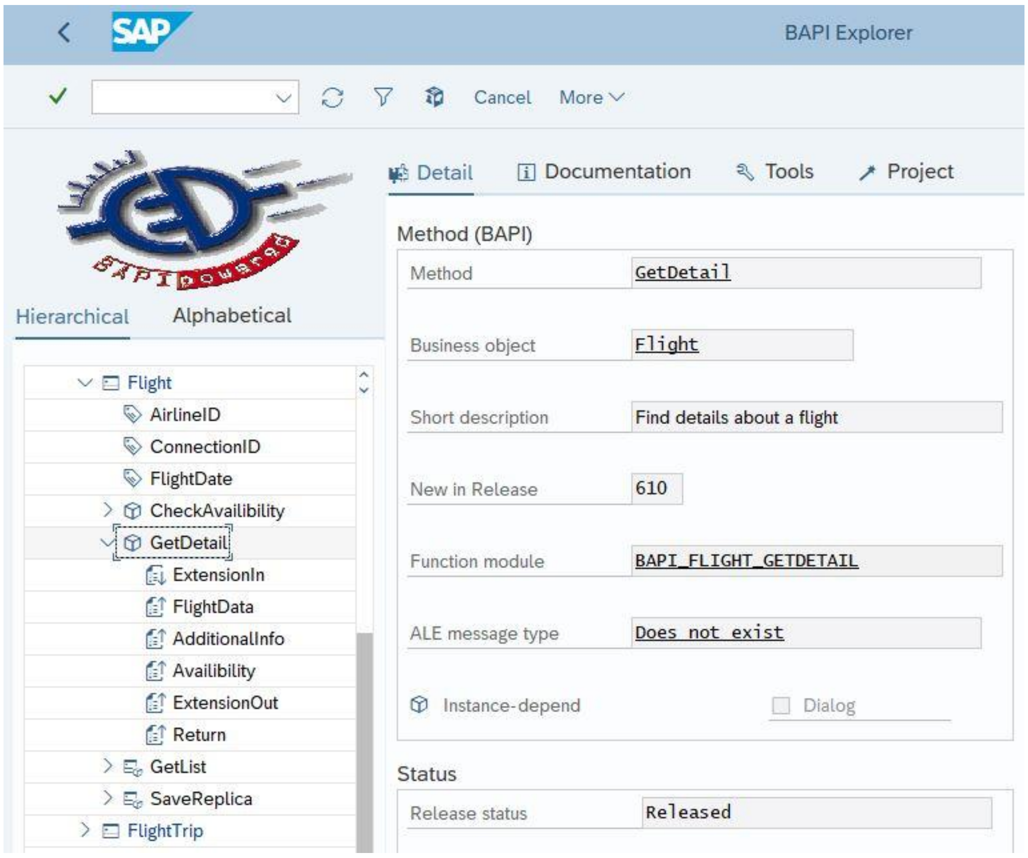
Object

Object Name	Flight		
Short description	Flight with connection data (SAP train…		
Object type	SFLIGHT		
Pack.	SAPBC_IBF_…	Component	BC-DWB
Person responsible	SAP		
Created on	31.07.1997	Release	40A

Status

Release status	Released		
Last changed by	SAP	Changed on	19.10.2001

Now you can view all information in a clear structure.



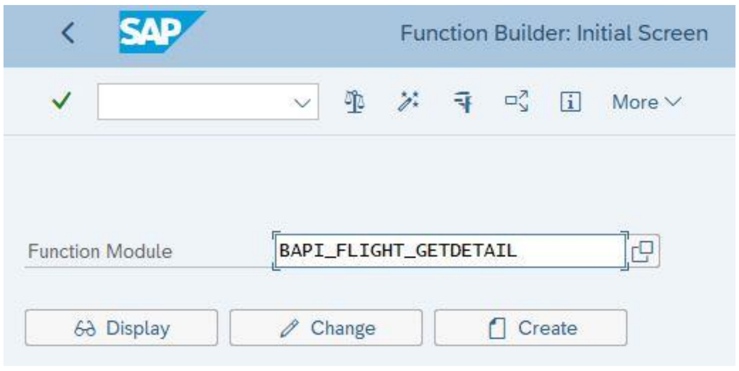
Method (BAPI)

Method	GetDetail
Business object	Flight
Short description	Find details about a flight
New in Release	610
Function module	BAPI_FLIGHT_GETDETAIL
ALE message type	Does not exist
Instance-depend	Dialog

Status

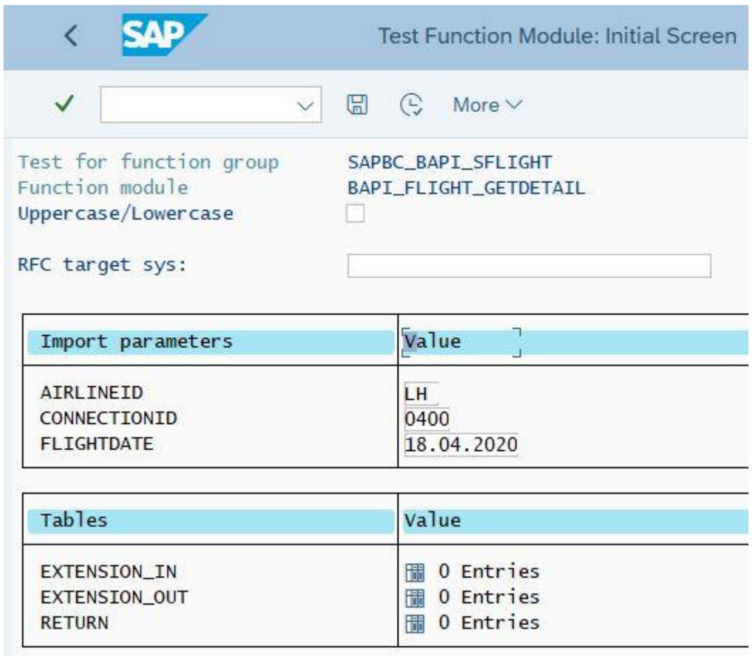
Release status	Released

Open in the Function Builder (TAC SE37) the function module BAPI_FLIGHT_GETDETAIL and press Execute button in the toolbar.



Function Module: BAPI_FLIGHT_GETDETAIL

Add the necessary values ...

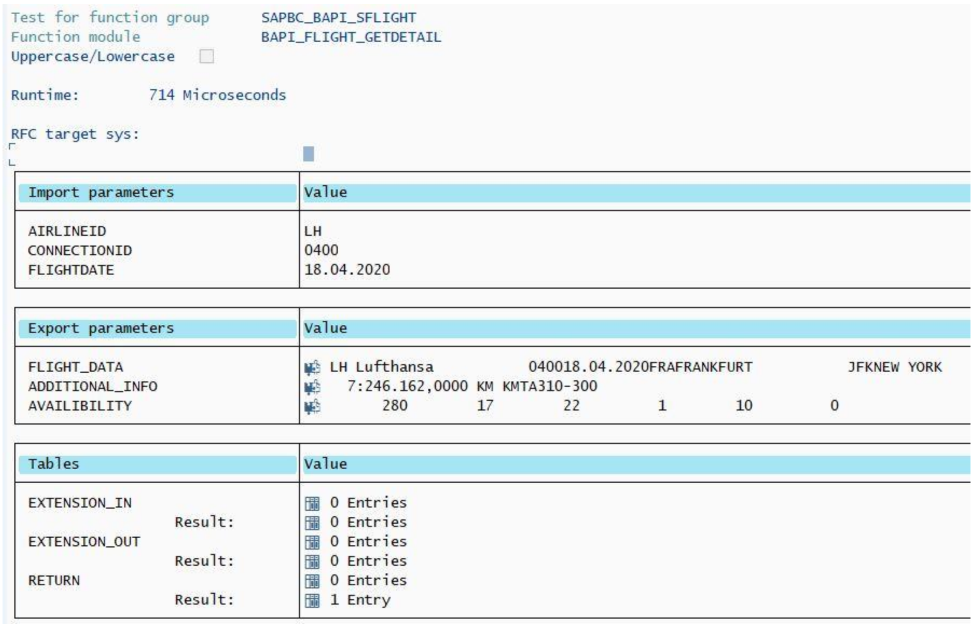


```
Test for function group     SAPBC_BAPI_SFLIGHT
Function module             BAPI_FLIGHT_GETDETAIL
Uppercase/Lowercase         ☐

RFC target sys:
```

Import parameters	Value
AIRLINEID	LH
CONNECTIONID	0400
FLIGHTDATE	18.04.2020

Tables	Value
EXTENSION_IN	0 Entries
EXTENSION_OUT	0 Entries
RETURN	0 Entries

… and look at the results.



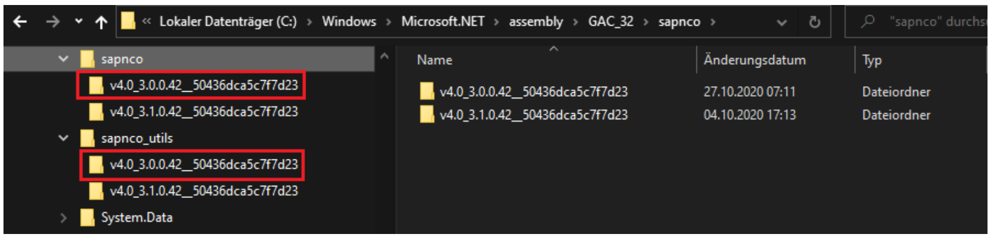
```
Test for function group     SAPBC_BAPI_SFLIGHT
Function module             BAPI_FLIGHT_GETDETAIL
Uppercase/Lowercase         ☐

Runtime:          714 Microseconds

RFC target sys:
```

Import parameters	Value
AIRLINEID	LH
CONNECTIONID	0400
FLIGHTDATE	18.04.2020

Export parameters	Value
FLIGHT_DATA	LH Lufthansa 040018.04.2020FRAFRANKFURT JFKNEW YORK
ADDITIONAL_INFO	7:246.162,0000 KM KMTA310-300
AVAILIBILITY	280 17 22 1 10 0

Tables		Value
EXTENSION_IN		0 Entries
	Result:	0 Entries
EXTENSION_OUT		0 Entries
	Result:	0 Entries
RETURN		0 Entries
	Result:	1 Entry

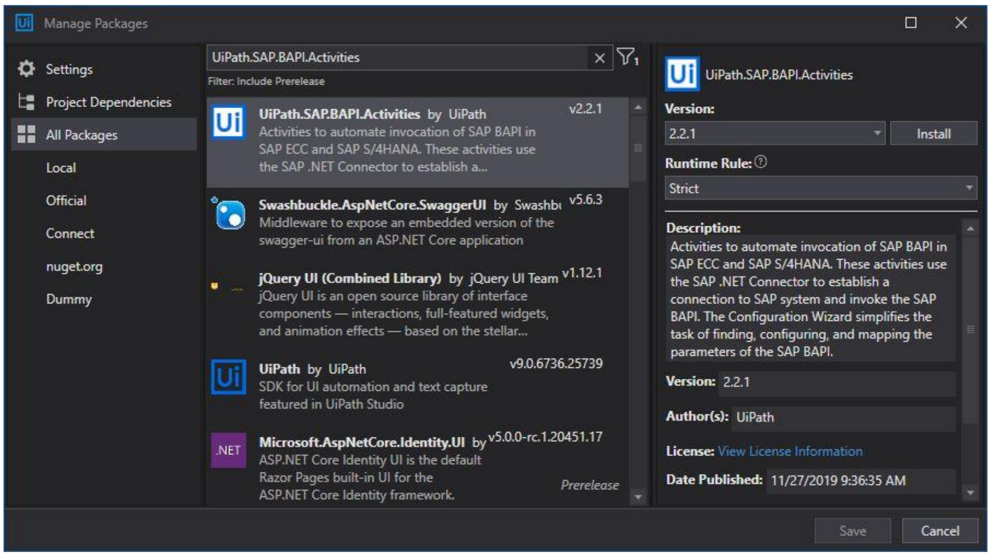
Hint: To use the BAPI Activity it is necessary to install SAP dotNET Connector NCo. If you don't have access to the SAP Support Portal to download the NCo and you have installed an SAP GUI for Windows you can use the available installation, with a small adjustment in the Global Assembly Cache (GAC). Duplicate in the GAC, in the subdirectories sapnco and sapnco_utils, the existing directories and change the version number from 3.1.0.42 to 3.0.0.42. The installation of the BAPI activity and its operation works perfectly without additional NCo installation. But for this operation you need administration rights.


« Lokaler Datenträger (C:) › Windows › Microsoft.NET › assembly › GAC_32 › sapnco ›

- sapnco
  - v4.0_3.0.0.42__50436dca5c7f7d23
  - v4.0_3.1.0.42__50436dca5c7f7d23
- sapnco_utils
  - v4.0_3.0.0.42__50436dca5c7f7d23
  - v4.0_3.1.0.42__50436dca5c7f7d23
- System.Data

Name	Änderungsdatum	Typ
v4.0_3.0.0.42__50436dca5c7f7d23	27.10.2020 07:11	Dateiordner
v4.0_3.1.0.42__50436dca5c7f7d23	04.10.2020 17:13	Dateiordner

Install the BAPI Package.


Manage Packages

Settings
Project Dependencies
All Packages
Local
Official
Connect
nuget.org
Dummy

UiPath.SAP.BAPI.Activities
Filter: Include Prerelease

UiPath.SAP.BAPI.Activities by UiPath v2.2.1
Activities to automate invocation of SAP BAPI in SAP ECC and SAP S/4HANA. These activities use the SAP .NET Connector to establish a...

Swashbuckle.AspNetCore.SwaggerUI by Swashbu v5.6.3
Middleware to expose an embedded version of the swagger-ui from an ASP.NET Core application

jQuery UI (Combined Library) by jQuery UI Team v1.12.1
jQuery UI is an open source library of interface components — interactions, full-featured widgets, and animation effects — based on the stellar...

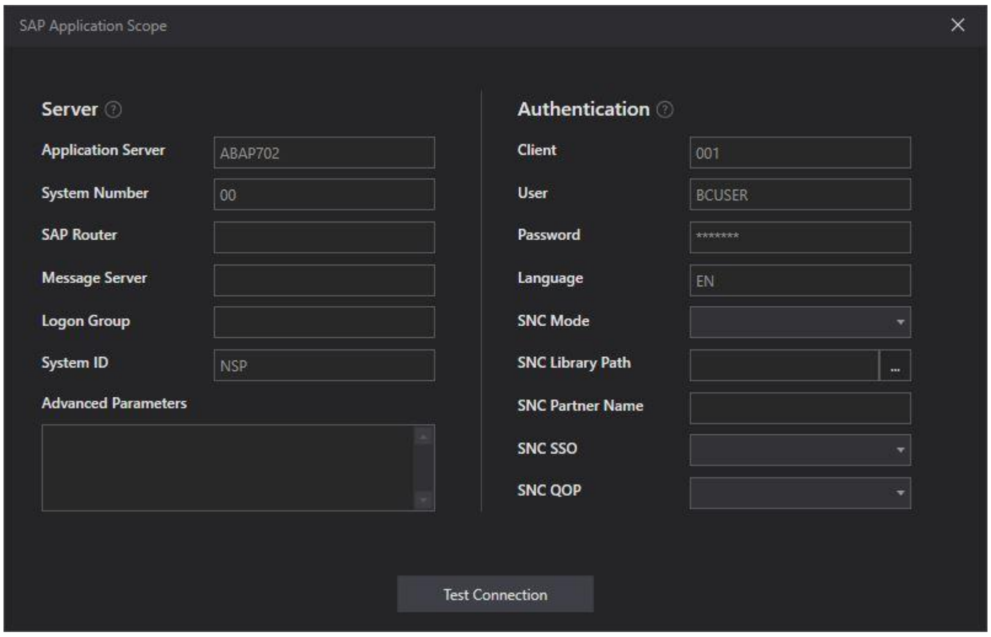
UiPath by UiPath v9.0.6736.25739
SDK for UI automation and text capture featured in UiPath Studio

Microsoft.AspNetCore.Identity.UI by v5.0.0-rc.1.20451.17
ASP.NET Core Identity UI is the default Razor Pages built-in UI for the ASP.NET Core Identity framework.
Prerelease

UiPath.SAP.BAPI.Activities
Version: 2.2.1 Install
Runtime Rule: Strict
Description: Activities to automate invocation of SAP BAPI in SAP ECC and SAP S/4HANA. These activities use the SAP .NET Connector to establish a connection to SAP system and invoke the SAP BAPI. The Configuration Wizard simplifies the task of finding, configuring, and mapping the parameters of the SAP BAPI.
Version: 2.2.1
Author(s): UiPath
License: View License Information
Date Published: 11/27/2019 9:36:35 AM

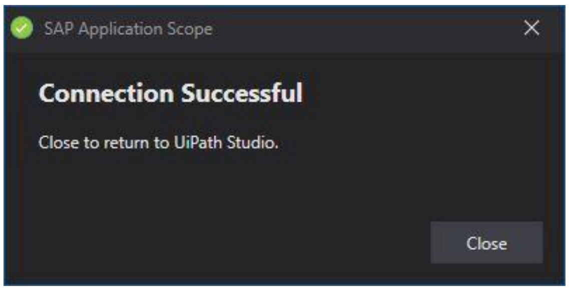
Save Cancel

Define your application scope.


SAP Application Scope

Server		**Authentication**	
Application Server	ABAP702	Client	001
System Number	00	User	BCUSER
SAP Router		Password	*******
Message Server		Language	EN
Logon Group		SNC Mode	
System ID	NSP	SNC Library Path	
Advanced Parameters		SNC Partner Name	
		SNC SSO	
		SNC QOP	

Test Connection

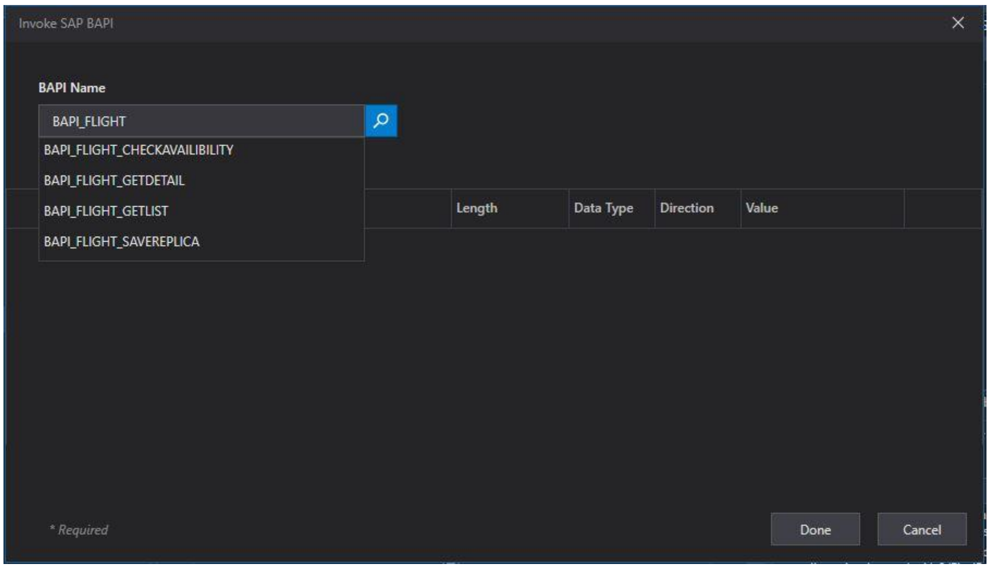


SAP Application Scope

Connection Successful

Close to return to UiPath Studio.

Invoke the BAPI method BAPI_FLIGHT_GETDETAIL.



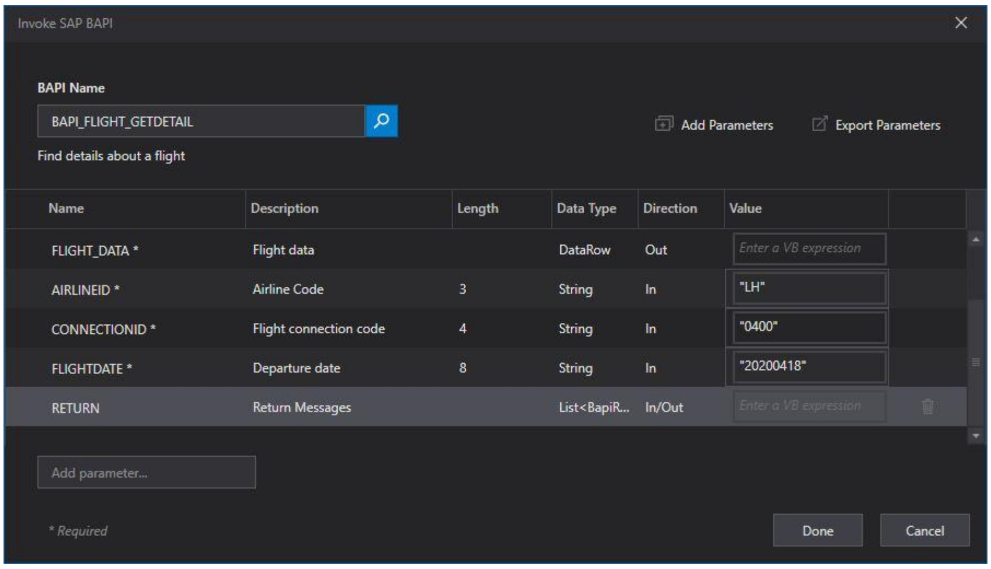
Invoke SAP BAPI

BAPI Name: BAPI_FLIGHT

- BAPI_FLIGHT_CHECKAVAILIBILITY
- BAPI_FLIGHT_GETDETAIL
- BAPI_FLIGHT_GETLIST
- BAPI_FLIGHT_SAVEREPLICA

Length	Data Type	Direction	Value

* Required



Invoke SAP BAPI

BAPI Name: BAPI_FLIGHT_GETDETAIL

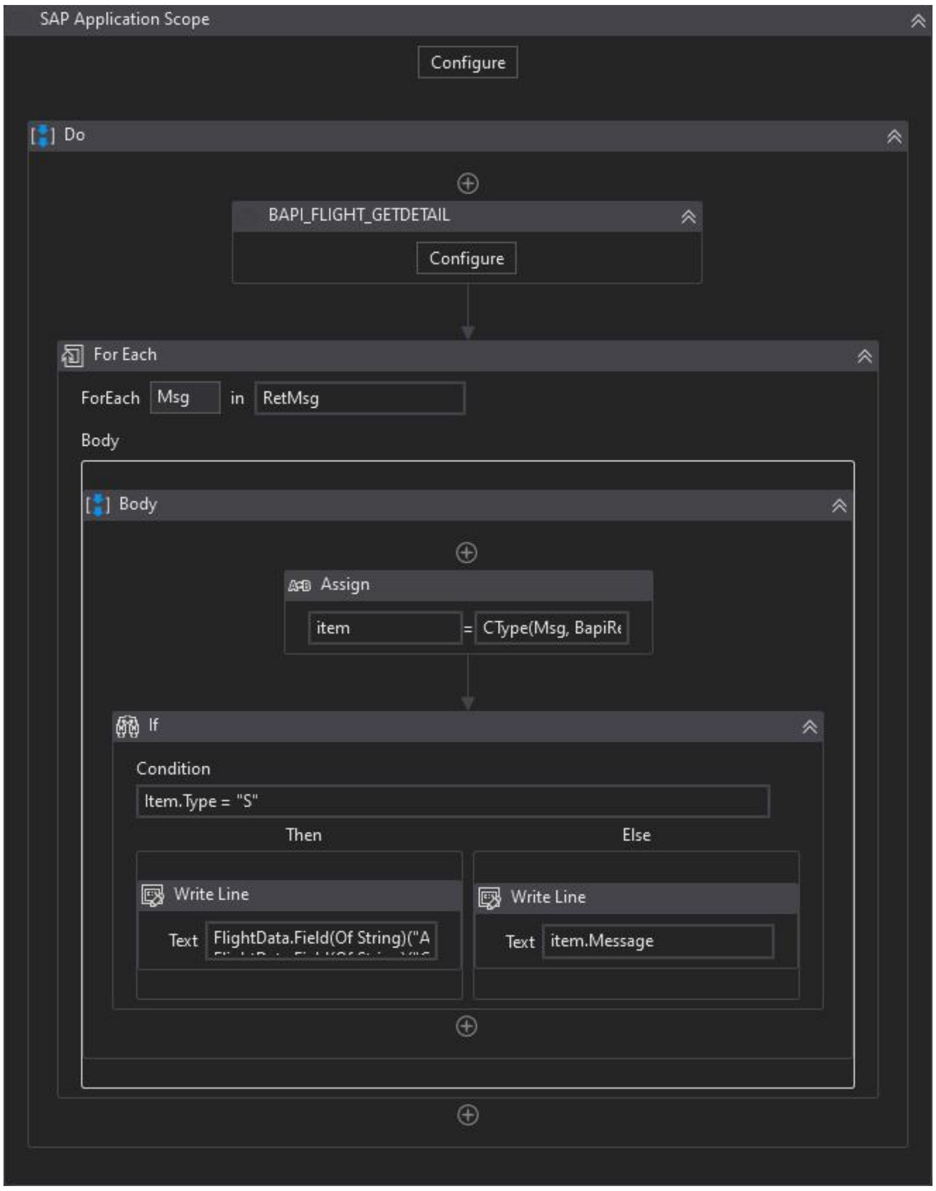
Find details about a flight

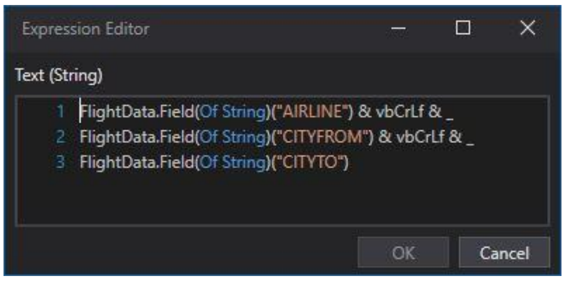
Name	Description	Length	Data Type	Direction	Value
FLIGHT_DATA *	Flight data		DataRow	Out	Enter a VB expression
AIRLINEID *	Airline Code	3	String	In	"LH"
CONNECTIONID *	Flight connection code	4	String	In	"0400"
FLIGHTDATE *	Departure date	8	String	In	"20200418"
RETURN	Return Messages		List<BapiR...	In/Out	Enter a VB expression

Add parameter...

* Required

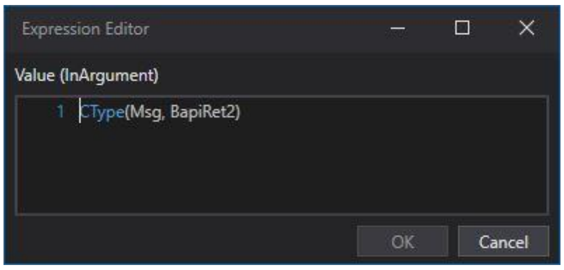
Build the workflow and run it …





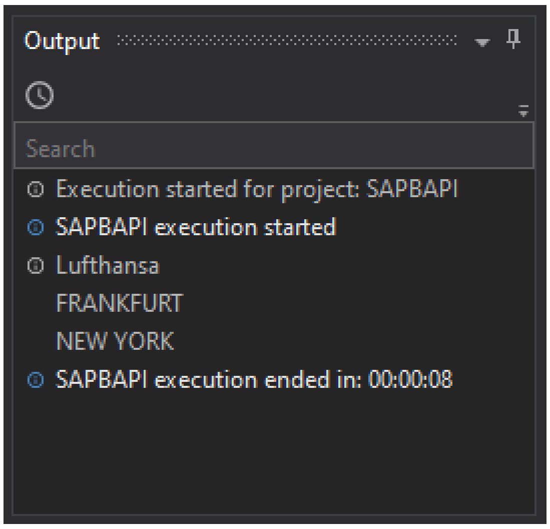
```
FlightData.Field(Of String)("AIRLINE") & vbCrLf & _
FlightData.Field(Of String)("CITYFROM") & vbCrLf & _
FlightData.Field(Of String)("CITYTO")
```

```
FlightData("AIRLINE").ToString & vbCrLf & _
FlightData("CITYFROM").ToString & vbCrLf & _
FlightData("CITYTO").ToString
```



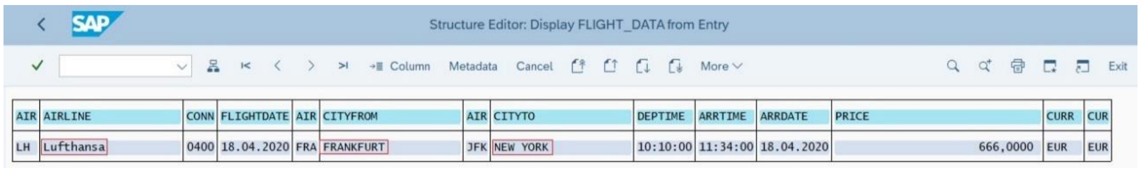
```
CType(Msg, BapiRet2)
```

… and look at the results.



```
Execution started for project: SAPBAPI
SAPBAPI execution started
Lufthansa
FRANKFURT
NEW YORK
SAPBAPI execution ended in: 00:00:08
```

Here for comparison the same result displayed in the SAP system.

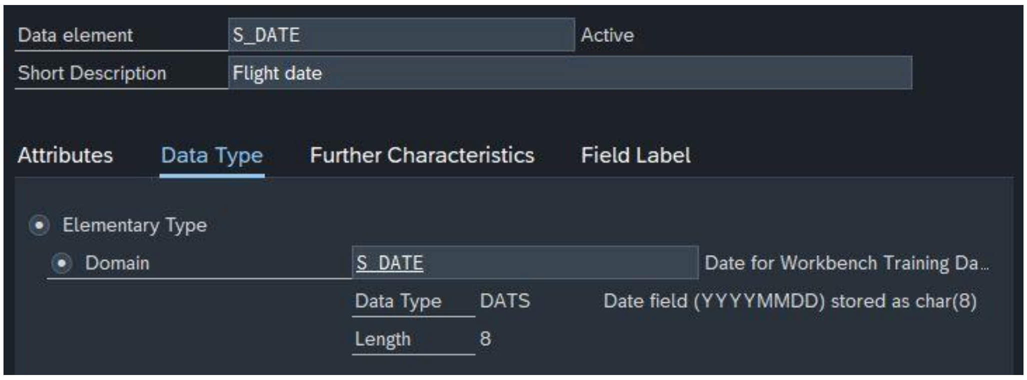


AIR	AIRLINE	CONN	FLIGHTDATE	AIR	CITYFROM	AIR	CITYTO	DEPTIME	ARRTIME	ARRDATE	PRICE	CURR	CUR
LH	Lufthansa	0400	18.04.2020	FRA	FRANKFURT	JFK	NEW YORK	10:10:00	11:34:00	18.04.2020	666,0000	EUR	EUR

Data Type DATS

The data type DATS describes objects of the type CHAR with a length of 8 characters. It is designed for calendar dates in the format YYYYMMDD. DATS is assigned to the internal ABAP data type D.

In our example the flight date is from the type DATS.

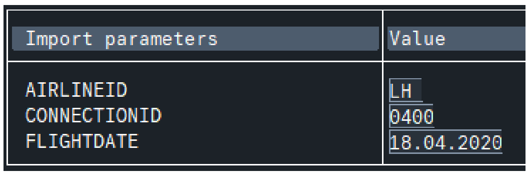


Data element	S_DATE	Active
Short Description	Flight date	

Attributes | Data Type | Further Characteristics | Field Label

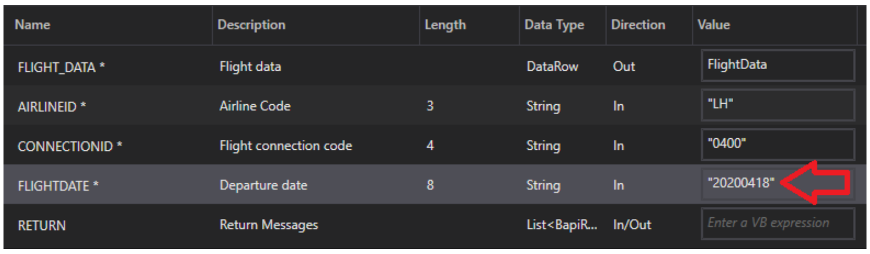
● Elementary Type
● Domain: S_DATE — Date for Workbench Training Da…
Data Type: DATS — Date field (YYYYMMDD) stored as char(8)
Length: 8

In the SE37 it is necessary to type DD.MM.YYYY …



Import parameters	Value
AIRLINEID	LH
CONNECTIONID	0400
FLIGHTDATE	18.04.2020

… but in the BAPI activity it is necessary to type YYYYMMDD.



Name	Description	Length	Data Type	Direction	Value
FLIGHT_DATA *	Flight data		DataRow	Out	FlightData
AIRLINEID *	Airline Code	3	String	In	"LH"
CONNECTIONID *	Flight connection code	4	String	In	"0400"
FLIGHTDATE *	Departure date	8	String	In	"20200418"
RETURN	Return Messages		List<BapiR...	In/Out	Enter a VB expression

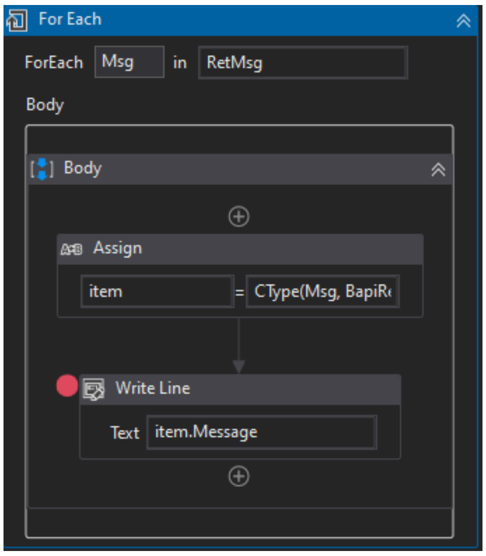
Leading Zeros

In many character fields of the SAP tables, e.g. like table KNA1 General Data in Customer Master and the field KUNNR customer number, leading zeros are also stored. Your customer number 4711, which can be seen in the UI, is in the table 0000004711. The field KUNNR is a character field with the length of ten characters. When using BAPI functions, it may therefore be necessary to include the leading zeros.

How to Read BAPIRET2 Structure Easily

BAPIRET2 is a list of structure. With a For Each activity you can loop over the records. To get access to the single entries of the structure you can use the VB.NET CType command, to convert the object type to BapiRet2.

```
CType(Msg, BapiRet2)
```


For Each

ForEach Msg in RetMsg

Body

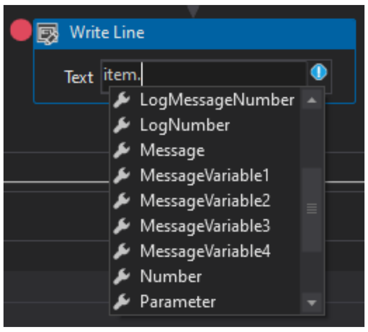
Assign

item = CType(Msg, BapiR

Write Line

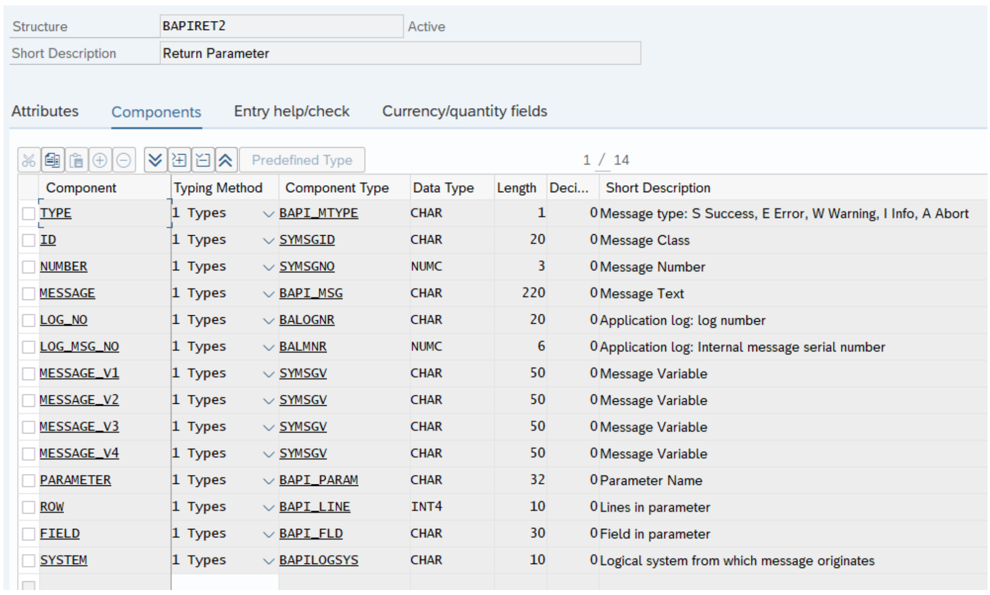
Text item.Message

With this step you have easy access to any element of the structure.


Write Line

Text item.

- LogMessageNumber
- LogNumber
- Message
- MessageVariable1
- MessageVariable2
- MessageVariable3
- MessageVariable4
- Number
- Parameter


Structure	BAPIRET2 Active
Short Description	Return Parameter

Attributes | Components | Entry help/check | Currency/quantity fields

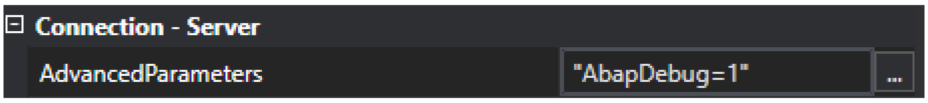
1 / 14

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
TYPE	1 Types	BAPI_MTYPE	CHAR	1	0	Message type: S Success, E Error, W Warning, I Info, A Abort
ID	1 Types	SYMSGID	CHAR	20	0	Message Class
NUMBER	1 Types	SYMSGNO	NUMC	3	0	Message Number
MESSAGE	1 Types	BAPI_MSG	CHAR	220	0	Message Text
LOG_NO	1 Types	BALOGNR	CHAR	20	0	Application log: log number
LOG_MSG_NO	1 Types	BALMNR	NUMC	6	0	Application log: Internal message serial number
MESSAGE_V1	1 Types	SYMSGV	CHAR	50	0	Message Variable
MESSAGE_V2	1 Types	SYMSGV	CHAR	50	0	Message Variable
MESSAGE_V3	1 Types	SYMSGV	CHAR	50	0	Message Variable
MESSAGE_V4	1 Types	SYMSGV	CHAR	50	0	Message Variable
PARAMETER	1 Types	BAPI_PARAM	CHAR	32	0	Parameter Name
ROW	1 Types	BAPI_LINE	INT4	10	0	Lines in parameter
FIELD	1 Types	BAPI_FLD	CHAR	30	0	Field in parameter
SYSTEM	1 Types	BAPILOGSYS	CHAR	10	0	Logical system from which message originates

How to Debug ABAP Code from BAPI Activity

Sometimes it could be very informative to know what a BAPI function module (FM) is doing. With a tiny trick it is very easy to do that. Add in the AdvancedParameters field of the Connection - Server section in the properties of the SAP Application Scope the entry:

```
"AbapDebug=1"
```

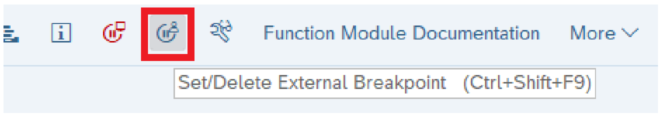


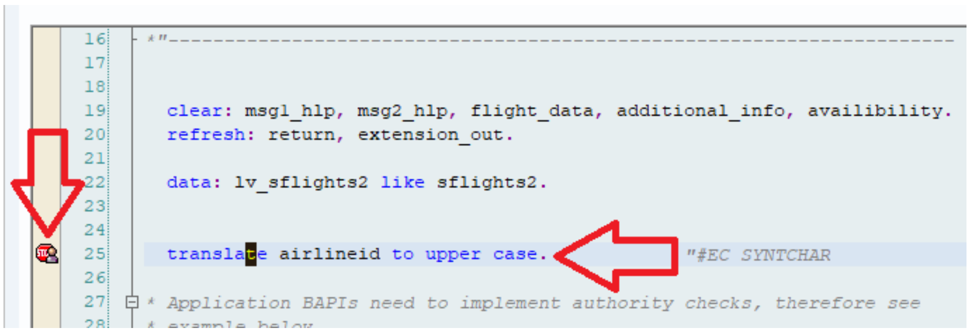
Connection - Server

AdvancedParameters	"AbapDebug=1"

Hint: If several parameters are to be added, they must be separated by a semicolon.

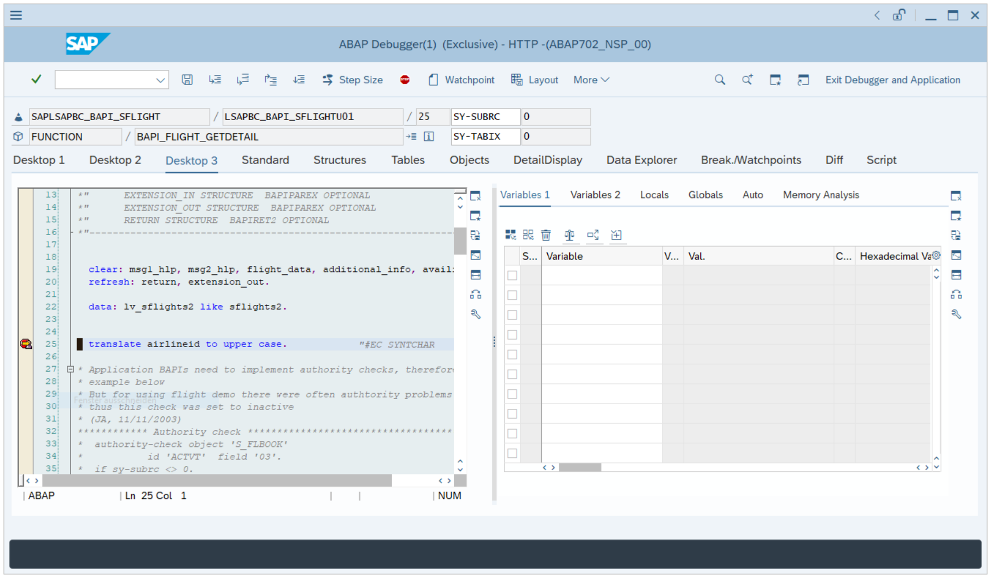
Open the BAPI FM with the Function Builder (TAC SE37) and set an external breakpoint at the first possible ABAP code line.


Function Module Documentation
More
Set/Delete External Breakpoint (Ctrl+Shift+F9)



```
*"----------------------------------------------------------------------


  clear: msg1_hlp, msg2_hlp, flight_data, additional_info, availibility.
  refresh: return, extension_out.

  data: lv_sflights2 like sflights2.


  translate airlineid to upper case.          "#EC SYNTCHAR

* Application BAPIs need to implement authority checks, therefore see
* example below
```

With the execution of the workflow in UiPath Studio opens the ABAP Debugger …

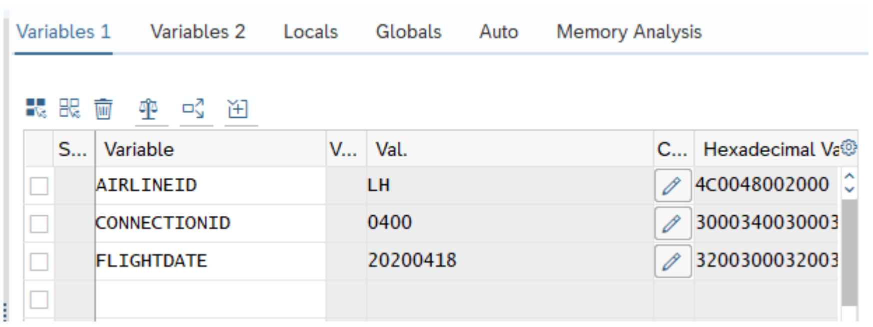


SAPLSAPBC_BAPI_SFLIGHT	LSAPBC_BAPI_SFLIGHTU01	25	SY-SUBRC	0
FUNCTION	BAPI_FLIGHT_GETDETAIL		SY-TABIX	0

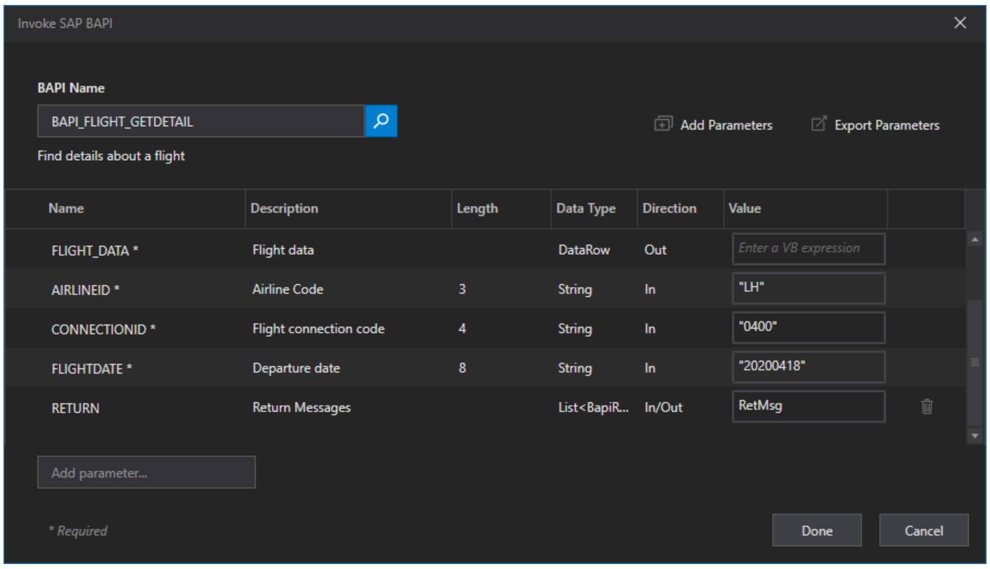
```
*"       EXTENSION_IN STRUCTURE  BAPIPAREX OPTIONAL
*"       EXTENSION_OUT STRUCTURE  BAPIPAREX OPTIONAL
*"       RETURN STRUCTURE  BAPIRET2 OPTIONAL
*"----------------------------------------------------------------------


  clear: msg1_hlp, msg2_hlp, flight_data, additional_info, avail
  refresh: return, extension_out.

  data: lv_sflights2 like sflights2.


  translate airlineid to upper case.          "#EC SYNTCHAR

* Application BAPIs need to implement authority checks, therefor
* example below
* But for using flight demo there were often authtority problems
* thus this check was set to inactive
* (JA, 11/11/2003)
************ Authority check ***********************************
*  authority-check object 'S_FLBOOK'
*           id 'ACTVT'  field '03'.
*  if sy-subrc <> 0.
```

ABAP | Ln 25 Col 1 | NUM

… and parameters and processing can be examined in detail.



S...	Variable	V...	Val.	C...	Hexadecimal Va
☐	AIRLINEID		LH		4C0048002000
☐	CONNECTIONID		0400		3000340030003
☐	FLIGHTDATE		20200418		3200300032003
☐					



BAPI Name: BAPI_FLIGHT_GETDETAIL

Find details about a flight

Name	Description	Length	Data Type	Direction	Value
FLIGHT_DATA *	Flight data		DataRow	Out	Enter a VB expression
AIRLINEID *	Airline Code	3	String	In	"LH"
CONNECTIONID *	Flight connection code	4	String	In	"0400"
FLIGHTDATE *	Departure date	8	String	In	"20200418"
RETURN	Return Messages		List<BapiR...	In/Out	RetMsg

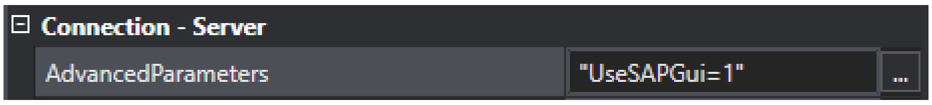
Add parameter...

* Required

How to use SAP GUI from BAPI Activity

Some (old) BAPIs needs an SAP GUI attached to the connection, because they try to send screen output to the client while executing. To handle this requirement, add in the AdvancedParameters field of the Connection - Server section in the properties of the SAP Application Scope the entry:

```
"UseSAPGui=1"
```



Connection - Server

AdvancedParameters	"UseSAPGui=1"

Possible values are:

- 0 = No SAP GUI (default)
- 1 = Use SAP GUI
- 2 = Use hidden (black) SAP GUI

Hint: If several parameters are to be added, they must be separated by a semicolon.

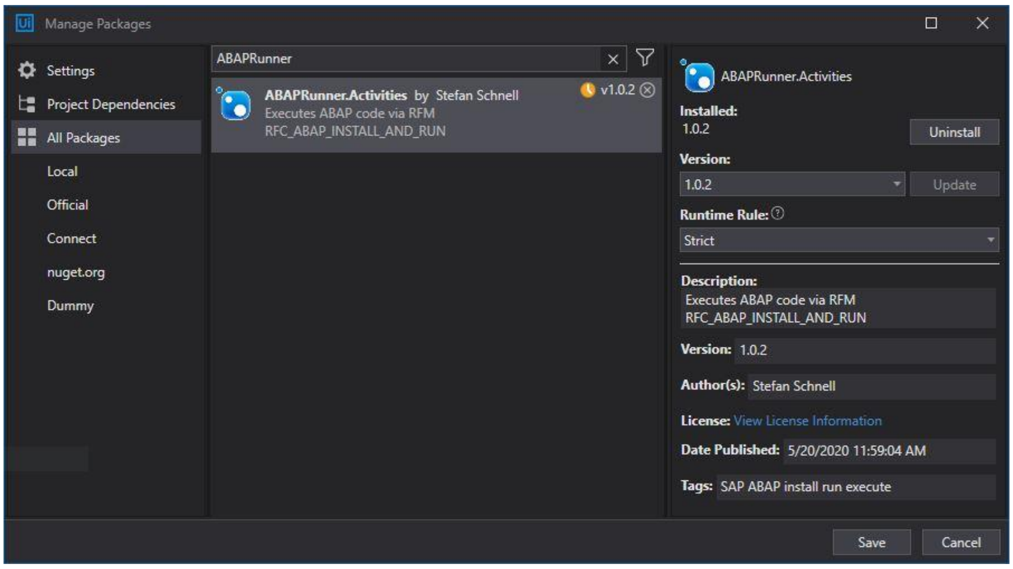
Execute ABAP Code from UiPath

Advanced Business Application Programming language, or ABAP in short form, is used in SAP environments. ABAP programming language is used to get or processing information in an SAP ERP or ECC system. It is only available in SAP application server back-end systems. You can’t use ABAP outside an SAP system.

ABAPRunner uses the RFM RFC_ABAP_INSTALL_AND_RUN to execute ABAP programs from outside an SAP system. ABAPRunner needs SAP NetWeaver RFC library, which comes with the SAP GUI for Windows installation.

Important hint: The RFM RFC_ABAP_INSTALL_AND_RUN is available on any SAP system, but you can use it only in development and quality assurance systems.

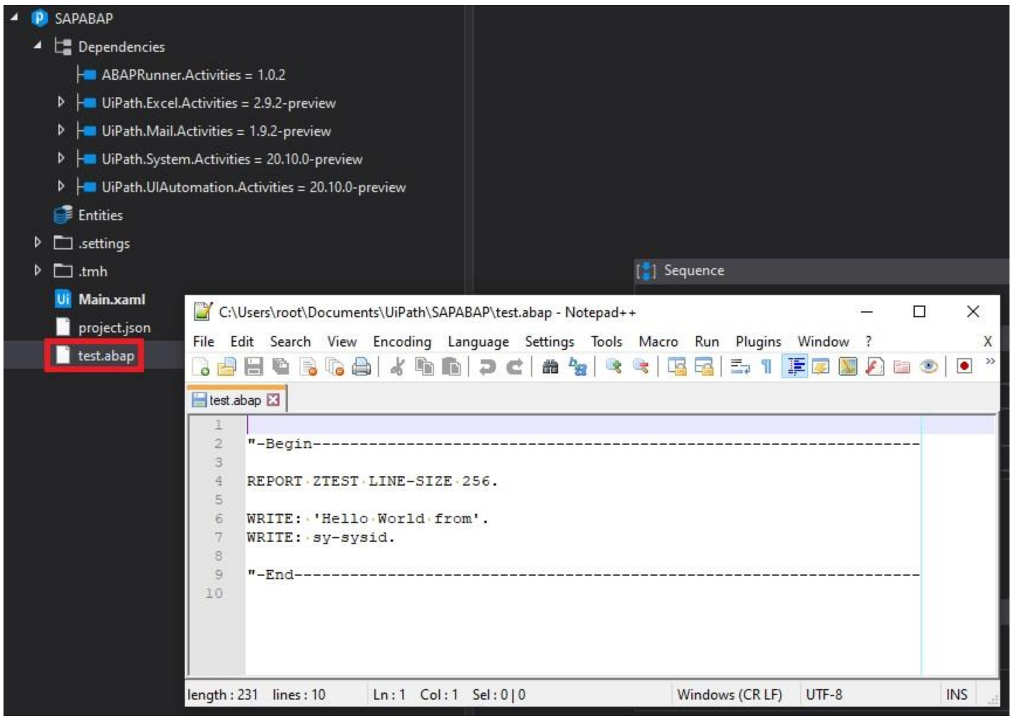
Install the ABAPRunner Package.



ABAPRunner.Activities by Stefan Schnell v1.0.2
Executes ABAP code via RFM RFC_ABAP_INSTALL_AND_RUN

ABAPRunner.Activities
Installed: 1.0.2
Version: 1.0.2
Runtime Rule: Strict
Description: Executes ABAP code via RFM RFC_ABAP_INSTALL_AND_RUN
Version: 1.0.2
Author(s): Stefan Schnell
License: View License Information
Date Published: 5/20/2020 11:59:04 AM
Tags: SAP ABAP install run execute

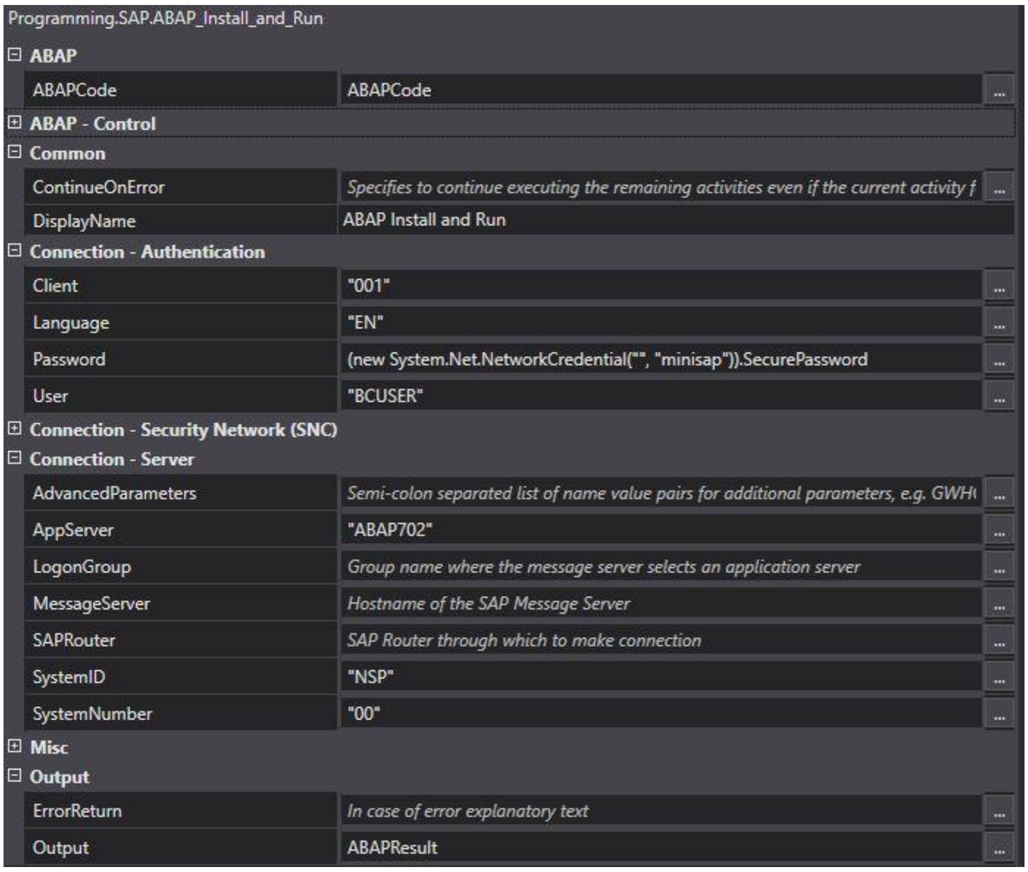
Store the ABAP code in your project directory.


SAPABAP
Dependencies
ABAPRunner.Activities = 1.0.2
UiPath.Excel.Activities = 2.9.2-preview
UiPath.Mail.Activities = 1.9.2-preview
UiPath.System.Activities = 20.10.0-preview
UiPath.UIAutomation.Activities = 20.10.0-preview
Entities
.settings
.tmh
Main.xaml
project.json
test.abap
Sequence
C:\Users\root\Documents\UiPath\SAPABAP\test.abap - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
test.abap
length : 231 lines : 10 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```

"-Begin-----------------------------------------------------------------

REPORT ZTEST LINE-SIZE 256.

WRITE: 'Hello World from'.
WRITE: sy-sysid.

"-End-------------------------------------------------------------------

```

```
*-Begin-------------------------------------------------------------------

Report zTest Line-Size 256.

Write: 'Hello World from'.
Write: sy-sysid.

*-End---------------------------------------------------------------------
```

Configure the ABAP_Install_and_Run activity.



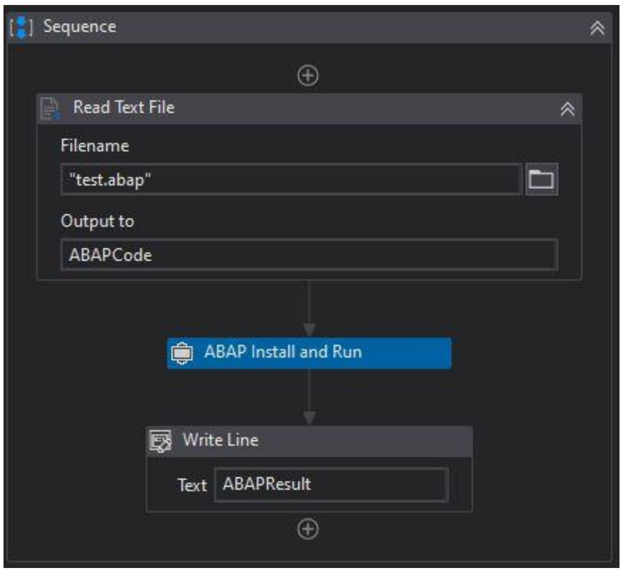
Programming.SAP.ABAP_Install_and_Run

ABAP	
ABAPCode	ABAPCode
ABAP - Control	
Common	
ContinueOnError	*Specifies to continue executing the remaining activities even if the current activity f*
DisplayName	ABAP Install and Run
Connection - Authentication	
Client	"001"
Language	"EN"
Password	(new System.Net.NetworkCredential("", "minisap")).SecurePassword
User	"BCUSER"
Connection - Security Network (SNC)	
Connection - Server	
AdvancedParameters	*Semi-colon separated list of name value pairs for additional parameters, e.g. GWH*
AppServer	"ABAP702"
LogonGroup	*Group name where the message server selects an application server*
MessageServer	*Hostname of the SAP Message Server*
SAPRouter	*SAP Router through which to make connection*
SystemID	"NSP"
SystemNumber	"00"
Misc	
Output	
ErrorReturn	*In case of error explanatory text*
Output	ABAPResult

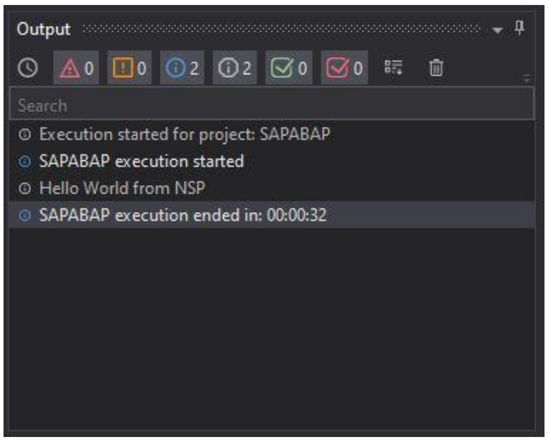
Hint: To set the password, you must use a secure string. Use the following pattern:

```
(new System.Net.NetworkCredential("", "myPassword")).SecurePassword
```

Build the workflow and run it …



… and look at the results.



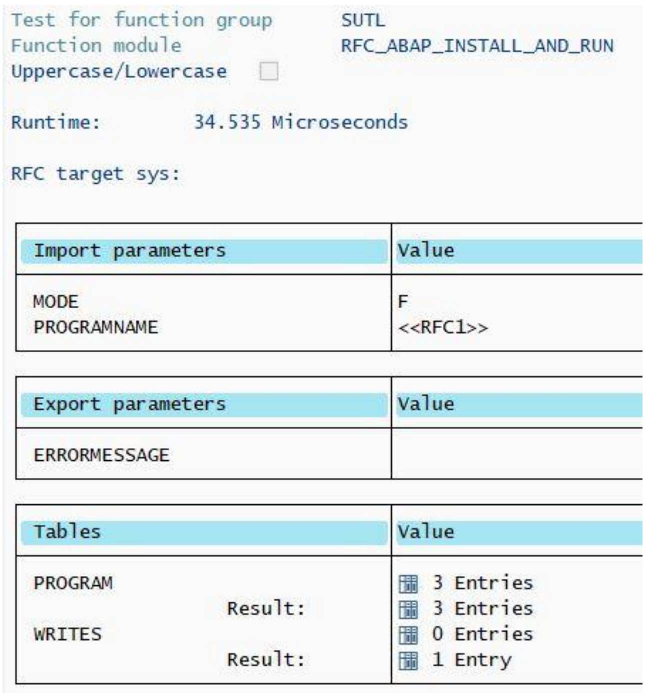
```
Execution started for project: SAPABAP
SAPABAP execution started
Hello World from NSP
SAPABAP execution ended in: 00:00:32
```

The direct use of ABAP can be applied to many use cases …

- Generating test data with all possibilities of the ABAP language.
- Execution of non-remote enabled function modules.
- Check data with SAP Open SQL statements.
- …

Especially in the field of test automation this approach can be used very profitably.

Here for comparison the same result displayed in the SAP system with the Function Builder (TAC SE37).



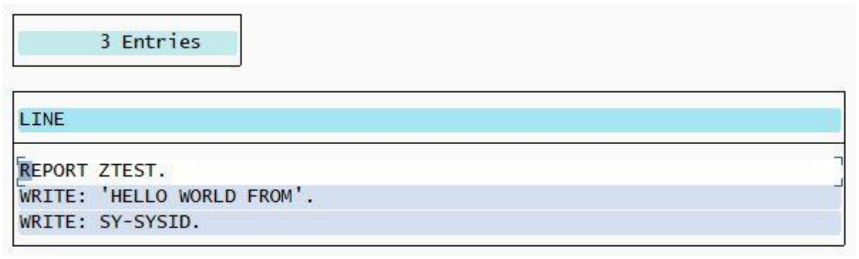
```
Test for function group     SUTL
Function module             RFC_ABAP_INSTALL_AND_RUN
Uppercase/Lowercase  ☐

Runtime:        34.535 Microseconds

RFC target sys:
```

Import parameters	Value
MODE	F
PROGRAMNAME	<<RFC1>>

Export parameters	Value
ERRORMESSAGE	

Tables		Value
PROGRAM		3 Entries
	Result:	3 Entries
WRITES		0 Entries
	Result:	1 Entry

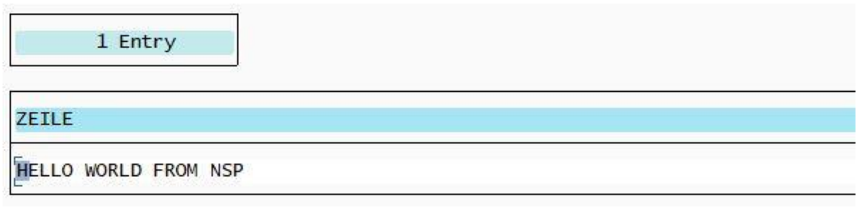
Here the import in the table PROGRAM.



3 Entries

```
LINE
REPORT ZTEST.
WRITE: 'HELLO WORLD FROM'.
WRITE: SY-SYSID.
```

And here the result in the table WRITES.

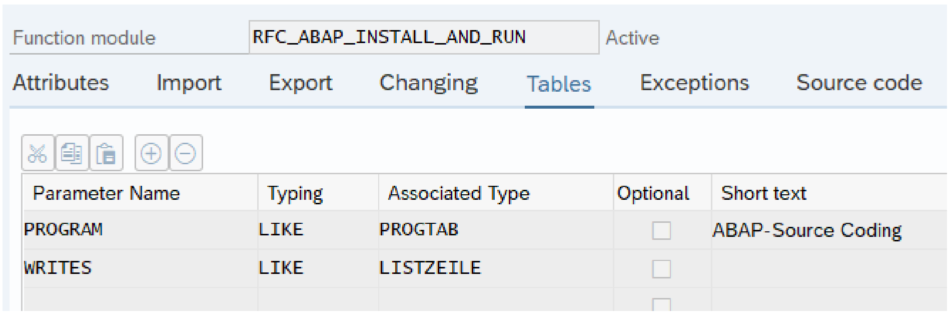


1 Entry

```
ZEILE
HELLO WORLD FROM NSP
```

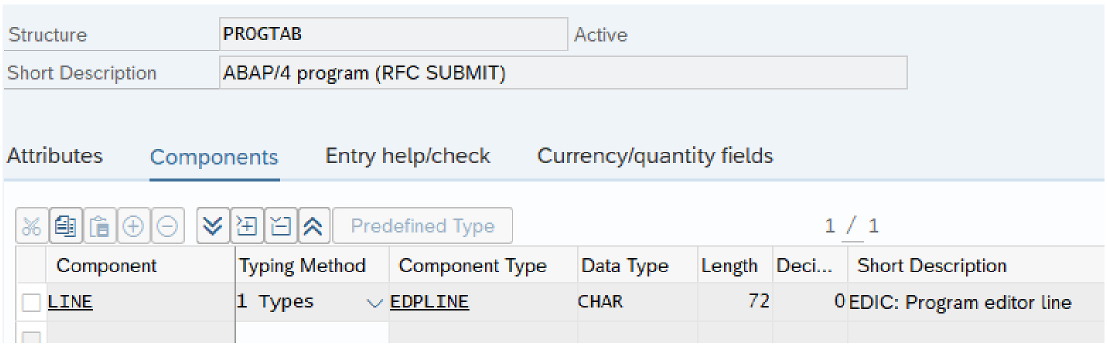
A Few Restrictions of RFC_ABAP_INSTALL_AND_RUN

The RFM RFC_ABAP_INSTALL_AND_RUN has some restrictions, which have to be considered.


Function module RFC_ABAP_INSTALL_AND_RUN Active

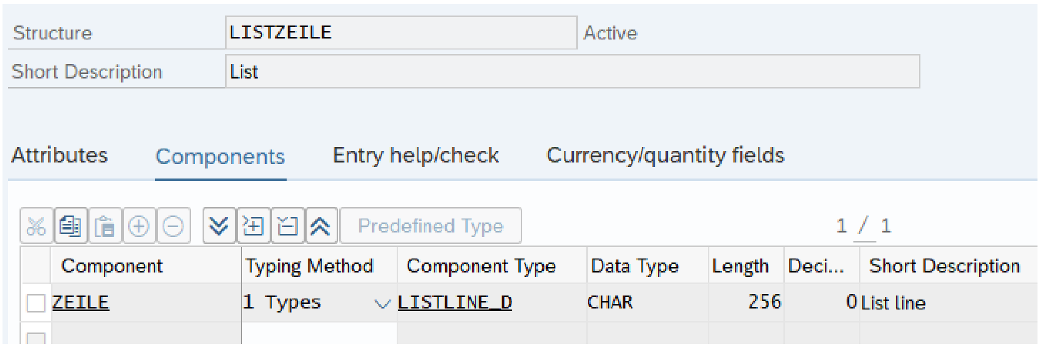
Parameter Name	Typing	Associated Type	Optional	Short text
PROGRAM	LIKE	PROGTAB	☐	ABAP-Source Coding
WRITES	LIKE	LISTZEILE	☐	

The import of the program lines is stored in the PROGRAM table in the component LINE. The data type of LINE is CHAR with a length of 72 characters. No ABAP code line may be longer than 72 characters.


Structure PROGTAB Active
Short Description ABAP/4 program (RFC SUBMIT)

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
LINE	1 Types	EDPLINE	CHAR	72	0	EDIC: Program editor line

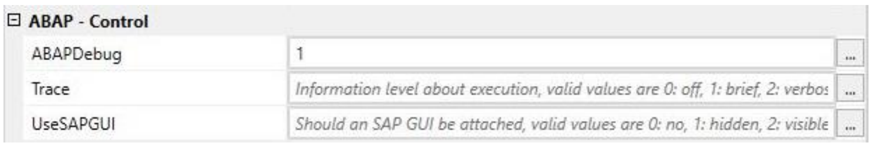
The return values are stored in the WRITES table in the component ZEILE. The data type of ZEILE is CHAR with a length of 256 characters. That is the reason why the report option LINE-SIZE is set to 256. It determines the number of characters in the line buffer as well as the number of columns in the list displayed.


Structure LISTZEILE Active
Short Description List

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
ZEILE	1 Types	LISTLINE_D	CHAR	256	0	List line

How to Debug ABAP Code from ABAPRunner

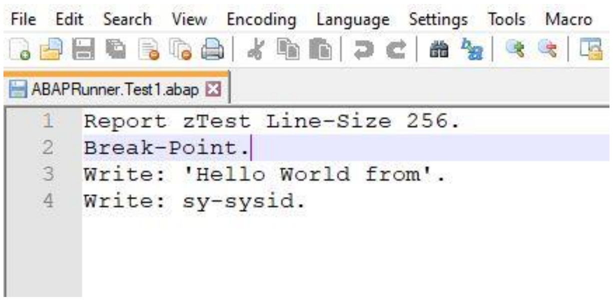
To debug your ABAP code in the context of the SAP application server it is necessary to set 1 in the ABAPDebug field of the ABAP – Control section in the properties of ABAP_Install_and_Run.



ABAP - Control	
ABAPDebug	1
Trace	Information level about execution, valid values are 0: off, 1: brief, 2: verbos
UseSAPGUI	Should an SAP GUI be attached, valid values are 0: no, 1: hidden, 2: visible

Add in the code the command …

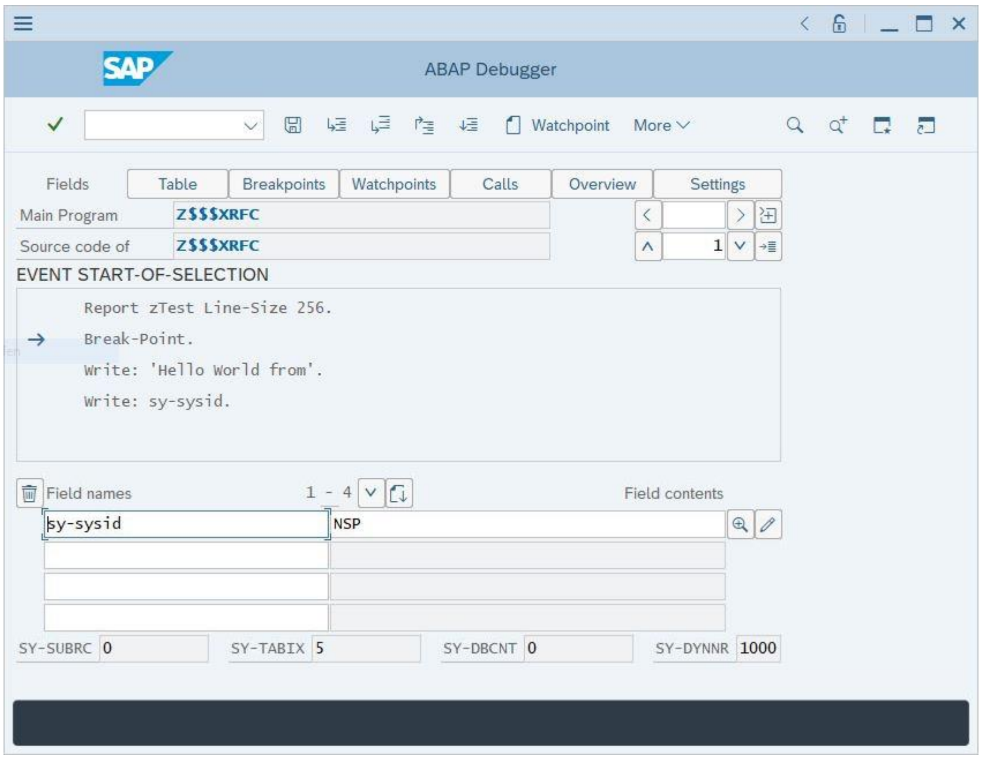
```
Break-Point.
```



File Edit Search View Encoding Language Settings Tools Macro
ABAPRunner.Test1.abap

```
Report zTest Line-Size 256.
Break-Point.
Write: 'Hello World from'.
Write: sy-sysid.
```

If the workflow is now executed, the ABAP Debugger opens and the code can be examined step by step.



Main Program Z$$$XRFC
Source code of Z$$$XRFC
EVENT START-OF-SELECTION

```
    Report zTest Line-Size 256.
→   Break-Point.
    Write: 'Hello World from'.
    Write: sy-sysid.
```

Field names 1 - 4	Field contents
sy-sysid	NSP

SY-SUBRC 0 SY-TABIX 5 SY-DBCNT 0 SY-DYNNR 1000

Internal ABAP Data Types

Value range of domain INTTYPE with description and dotNET data type mapping		
8	Integer number (8-byte integer)	long (System.Long)
C	Character string	string (System.String)
D	Date (Date: YYYYMMDD)	string (System.String)
F	Floating point number to accuracy of 8 bytes	double (System.Double)
I	Integer number (4-byte integer with sign)	int (System.Int32)
L		
N	Character string with digits only	string (System.String)
P	Packed number	decimal (System.Decimal)
T	Time (Time: HHMMSS)	string (System.String)
V	Character string (old Dictionary type VARC)	
X	Byte sequence (heXadecimal)	byte[] (Array of System.Byte)
a	Decimal floating point number, 16 digits	decimal (System.Decimal)
b	Integer number (1-byte integer, integer number <= 254)	Byte (System.Byte)
e	Decimal floating point number, 34 digits	decimal (System.Decimal)
g	Character string with variable length (ABAP type STRING)	string (System.String)
h	Table type (Internal table)	IRfcTable
j	Static boxed components	
k	Generic boxed components (Enumerated type)	
l	Reference to data object (Data reference)	
r	Reference to class/interface (Object reference)	
s	Integer number (2-byte integer, integer number <= 65535, only for length field before LCHR or LRAW)	Short (System.Short)
u	Structured type, flat	IRfcStructure
v	Structured type, deep	
y	Byte sequence with variable length (ABAP type XSTRING)	byte[] (Array of System.Byte)

Modify XAML to use Installed NCo

To use RFC calls it is necessary to add a few lines in the XAML file of your project.

Add in the Activity tag the following attributes:

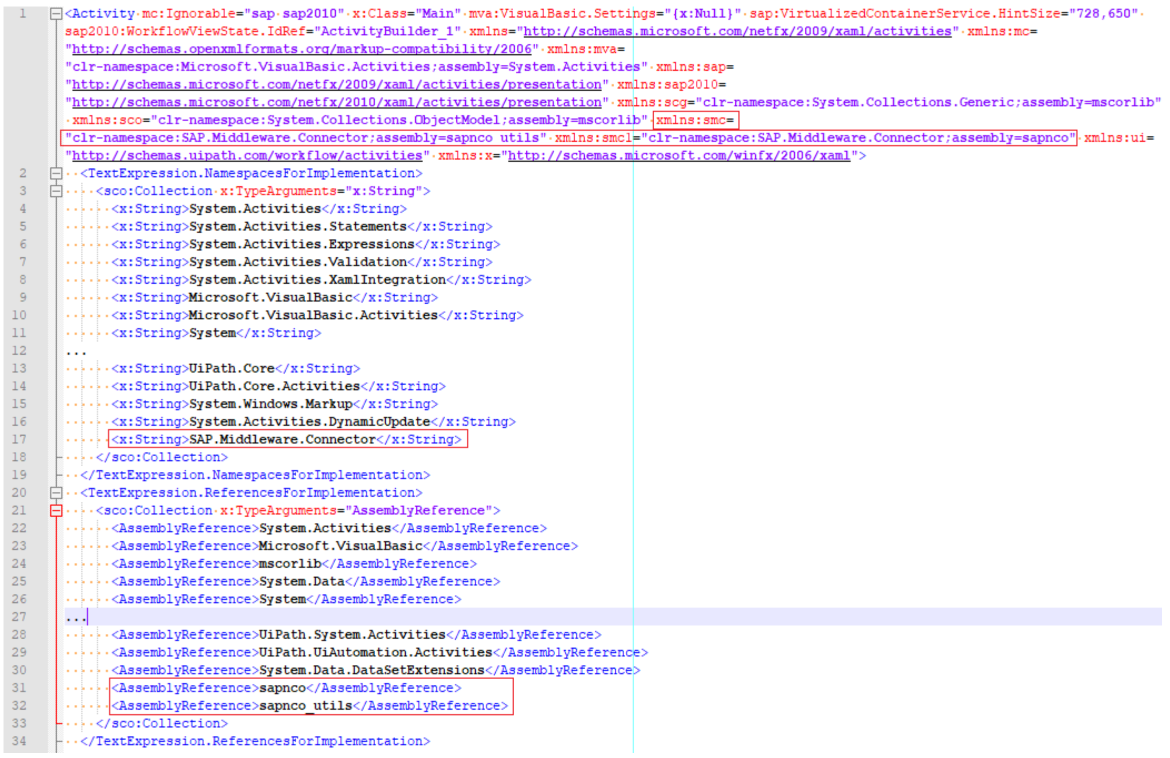
```
xmlns:smc="clr-
namespace:SAP.Middleware.Connector;assembly=sapnco_utils"
xmlns:smc1="clr-namespace:SAP.Middleware.Connector;assembly=sapnco"
```

Add at the end of the TextExpression.NamespacesForImplementation tag in the sco:Collection the line:

```
<x:String>SAP.Middleware.Connector</x:String>
```

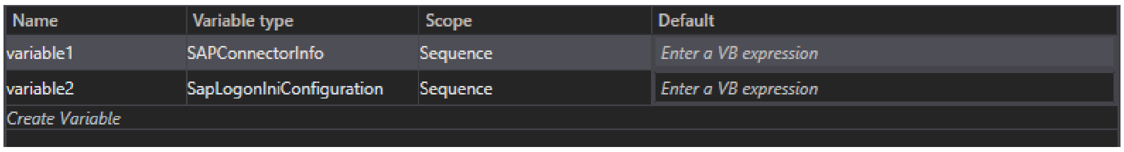
Add at the end of the TextExpression.ReferencesForImplementation tag in the sco:Collection the lines:

```
<AssemblyReference>sapnco</AssemblyReference>
<AssemblyReference>sapnco_utils</AssemblyReference>
```



```
<Activity mc:Ignorable="sap sap2010" x:Class="Main" mva:VisualBasic.Settings="{x:Null}" sap:VirtualizedContainerService.HintSize="728,650"
sap2010:WorkflowViewState.IdRef="ActivityBuilder_1" xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities" xmlns:mc=
"http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:mva=
"clr-namespace:Microsoft.VisualBasic.Activities;assembly=System.Activities" xmlns:sap=
"http://schemas.microsoft.com/netfx/2009/xaml/activities/presentation" xmlns:sap2010=
"http://schemas.microsoft.com/netfx/2010/xaml/activities/presentation" xmlns:scg="clr-namespace:System.Collections.Generic;assembly=mscorlib"
 xmlns:sco="clr-namespace:System.Collections.ObjectModel;assembly=mscorlib" xmlns:smc=
"clr-namespace:SAP.Middleware.Connector;assembly=sapnco_utils" xmlns:smc1="clr-namespace:SAP.Middleware.Connector;assembly=sapnco" xmlns:ui=
"http://schemas.uipath.com/workflow/activities" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <TextExpression.NamespacesForImplementation>
    <sco:Collection x:TypeArguments="x:String">
      <x:String>System.Activities</x:String>
      <x:String>System.Activities.Statements</x:String>
      <x:String>System.Activities.Expressions</x:String>
      <x:String>System.Activities.Validation</x:String>
      <x:String>System.Activities.XamlIntegration</x:String>
      <x:String>Microsoft.VisualBasic</x:String>
      <x:String>Microsoft.VisualBasic.Activities</x:String>
      <x:String>System</x:String>
...
      <x:String>UiPath.Core</x:String>
      <x:String>UiPath.Core.Activities</x:String>
      <x:String>System.Windows.Markup</x:String>
      <x:String>System.Activities.DynamicUpdate</x:String>
      <x:String>SAP.Middleware.Connector</x:String>
    </sco:Collection>
  </TextExpression.NamespacesForImplementation>
  <TextExpression.ReferencesForImplementation>
    <sco:Collection x:TypeArguments="AssemblyReference">
      <AssemblyReference>System.Activities</AssemblyReference>
      <AssemblyReference>Microsoft.VisualBasic</AssemblyReference>
      <AssemblyReference>mscorlib</AssemblyReference>
      <AssemblyReference>System.Data</AssemblyReference>
      <AssemblyReference>System</AssemblyReference>
...
      <AssemblyReference>UiPath.System.Activities</AssemblyReference>
      <AssemblyReference>UiPath.UiAutomation.Activities</AssemblyReference>
      <AssemblyReference>System.Data.DataSetExtensions</AssemblyReference>
      <AssemblyReference>sapnco</AssemblyReference>
      <AssemblyReference>sapnco_utils</AssemblyReference>
    </sco:Collection>
  </TextExpression.ReferencesForImplementation>
```

It is necessary to create two dummy variables, one with a type from the library sapnco and one with a type from the library sapnco_utils.

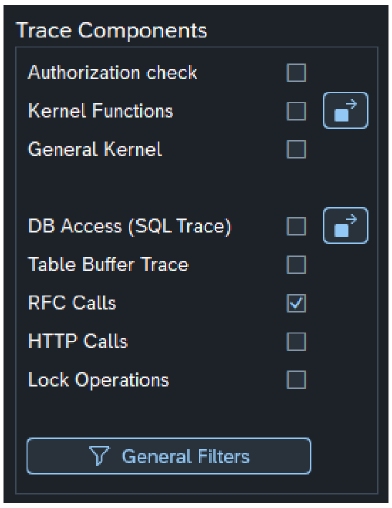


Name	Variable type	Scope	Default
variable1	SAPConnectorInfo	Sequence	*Enter a VB expression*
variable2	SapLogonIniConfiguration	Sequence	*Enter a VB expression*
Create Variable			

Now you can use SAP NCo in your code activity.

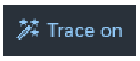
Using System Trace to Monitor RFC Activities

To record the internal SAP activities, it is possible to use the System Trace with the TAC ST01. You can record e.g. authorization, SQL and RFC activities. What should be monitored can be selected easily.

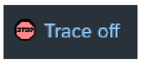

Trace Components

Authorization check	☐
Kernel Functions	☐
General Kernel	☐
DB Access (SQL Trace)	☐
Table Buffer Trace	☐
RFC Calls	✓
HTTP Calls	☐
Lock Operations	☐

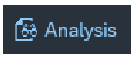
Activate the trace with the Trace on button in the toolbar.



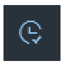
Execute your RFC activities, e.g. call a function with the BAPI activity. After this has been done, the trace can be deactivated again with the Trace off button.



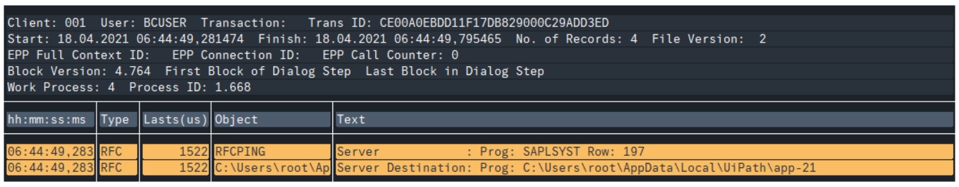
Now press the button Analysis to see the recorded activities.



Choose in the upcoming selection screen what you want to see in detail and start the Reporting.



In the Trace Display you can find now in detail all RFC activities. It becomes visible which RFM was called by which external program.



```
Client: 001  User: BCUSER  Transaction:   Trans ID: CE00A0EBDD11F17DB829000C29ADD3ED
Start: 18.04.2021 06:44:49,281474  Finish: 18.04.2021 06:44:49,795465  No. of Records: 4  File Version:  2
EPP Full Context ID:   EPP Connection ID:   EPP Call Counter: 0
Block Version: 4.764  First Block of Dialog Step  Last Block in Dialog Step
Work Process: 4  Process ID: 1.668

hh:mm:ss:ms  Type  Lasts(us) Object          Text
06:44:49,283 RFC        1522 RFCPING         Server             : Prog: SAPLSYST Row: 197
06:44:49,283 RFC        1522 C:\Users\root\Ap Server Destination: Prog: C:\Users\root\AppData\Local\UiPath\app-21
```

With a double click in one line you can display even more details. What is the name of the caller, which ABAP program is executed, how much time was needed etc.

With the system trace you can track very precisely which RFMs are called. To see even more details the SQL trace can be added. Then all used tables and SQL clauses can be seen.

List of Interesting Transaction Codes

Transaction Code	Description
SE16 / SE16N	Data Browser to view tables
SE37	Function Builder to execute function modules
SA38 / SE38	ABAP Editor to execute programs
SE84	Repository Information System
SE80	ABAP Workbench
BAPI	BAPI Explorer
SU53	Display Authorization Data
ST01	System Trace to monitor RFC, SQL and other activities
ST05	Display performance trace
SM50	Process overview
SM66	Global process overview
SM37	Job overview
SM04	User list
SLG1	Application Log
SM21	System Log
ST22	ABAP Runtime Error

List of Interesting Tables

Table	Description
TSTC	SAP Transaction Codes
TSTCT	Transaction Code Texts
TFDIR	Function Modules
TFTIT	Function Modules Texts
FUPARAREF	Parameters of Function Modules
TADIR	Directory of Repository Objects
DD02L	SAP Tables
DD02T	SAP Table Texts
DD03L	Table Fields
DD03T	Table Fields Texts
CVERS	Release Status of Software Components in System
PAT03	Patch Directory
RFCDES	Destination table for Remote Function Call
REPOSRC	Report Source Code

List of Interesting RFMs

RFM	Description
RFC_PING	Pings an SAP systems
RFC_SYSTEM_INFO	Delivers different information about the SAP system
RFC_READ_TABLE	External access to SAP tables to read the content
GET_SYSTEM_NUMBER	Delivers double digit instance number
GET_SYSTEM_NAME	Delivers the system ID (SID)
RFC_CLIENT_INFO	Find information on RFC client (Server Function)
RFC_LOGIN_INFO	Returns system information
RFC_ABAP_INSTALL_AND_RUN	Installation and execution of an ABAP program

List of Interesting Programs

Program	Description
RSPARAM	Displays all profile parameter
RSRFCCHK	RFC destinations with logon data
RS_ABAP_SOURCE_SCAN	Source scan ABAP report

Conclusion

These approaches to use RFM, BAPI and ABAP with UiPath offers a wide range of possibilities to execute functions or programs in an SAP back-end system. That allows a very high degree of integration. With the BAPI activity, the functions of the business objects can be used. With the NCo activity, which allows to use RFMs, all remote functions can be used. And the ABAPRunner activity offers the possibility to use ABAP code seamlessly in development and quality assurance systems. This allows SAP automation on a technical level via RFC calls to be easily implemented, for many use cases with high performance.

List of References

No.	Title	Reference
1	SAP .NET Connector 3.0 Overview	Link
2	SAP .NET Connector 3.0 Programming Guide	Link
3	SAP .NET Connector 3.0 Tutorial	Download link
4	Tutorials on SAP, C# and more	Link
5	How to Patch UiPath.SAP.BAPI to use it without additional SAP NCo installation	Link

Rights