SAP Scripting Tool: an Introduction how to use it

Applies to:

SAP Scripting Tool for Eclipse, version 0.8.8.

Summary

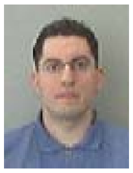
This article gives a short introduction to the SAP Scripting Tool, its motivation, installation and description of the different parts.

Author: Frederic Ahring

Company: SAP AG

Created on: 12 October 2006

Author Bio



Frederic Ahring (frederic.ahring@sap.com)

Frederic started to write programs at the age of 11, first for fun, later freelancing during his studies. He has several years of experience with Java, Perl and PHP after dropping Pascal and Delphi as his favorite languages. After graduating at the University of Applied Sciences, Karlsruhe, he joined SAP in July, 2006. His current work on the SAP Scripting Tool combines his experiences in Java and scripting languages.

Table of Contents

Scripting and SAP

Connecting to a SAP backend is often needed for custom solutions, from small reporting tools to information gathering for display on an Internet site. The Java connector JCo for accessing BAPIs is available, but in many situations smaller solutions than full-blown J2EE engines are preferred. These are often implemented in scripting languages such as Perl, Ruby, Python and especially PHP. While there are existing connectors for BAPI-RFCs and Webservice soapclients, documentation and sample code was sparse. We at SAP are aware that accessing SAP backends through scripting languages is desired by our clients and decided to help.

In the last months there was a series of articles, done by Vasil Bachvarov, providing information and samples for using the RFC and Webservice connectors. Meanwhile we have been working on a whole toolset to facilitate the scripting developer to easily generate library functions and sample code, ready to use or as base for his own applications.

Since this tool is aimed at helping you, the scripting developers, in whatever tasks you need, we used a very flexible, extendable architecture. The Eclipse project was chosen as platform since it provides a stable foundation, a large user community, a great plugin mechanism and already has extensions for developing several scripting languages. Craig Cmehils ‘Scripting in a Box’ is also based on Eclipse and was well-received.

Using the Scripting Tool

Installation and Prerequisites

The SAP Scripting Tool is based on the Eclipse Rich Client Platform. Go to www.eclipse.org and download either the full JDK or just the platform or use some Eclipse-based version for your own scripting language, like PHPeclipse or Scripting in a Box. Make sure you have at least version 3.1.0 of the Eclipse base.

Install Eclipse, set your preferences and your proxy, if needed.

The Scripting Tool can be installed via the Eclipse Update Manager. There is no remote update site available yet, so please download the ZIP file and save it locally. In the Update Manager (Help / Software Updates / Find and Install / Search for new features to install) you can then add a “New Archived Site”. Select the previously downloaded ZIP archive and name it “SAP Scripting Tool 0.8.8 (local)”.

You can then use this archive to do a normal plugin installation. Beside the “SAP Scripting Tool” you need the Eclipse Modeling Framework (EMF). If you don’t have it yet you can just install it together with the Scripting Tool, as we provide a copy in the same archive for your convenience. After pressing “next”, read and accept the licenses. Then you can (de)select the components of the Scripting Tool. Since our tool aims to help every scripter, in any language he might favor, but each scripter is usually only interested in his particular language of choice, we made the tool very flexible und pluggable. If you’re unsure, just leave the selection as is, you can uninstall parts or the whole tool later comfortably with the configuration manager.

After successful installation a restart of the workbench is recommended.

Installation of JCo library

For accessing ABAP systems with BAPIs, the Java library JCo (Java Connector) is necessary. Unfortunately this connector is platform specific and can’t be installed via Eclipse.

For Windows based Eclipse installations the needed DLL files are included, but must be installed manually. Open a Windows Explorer and navigate to the Eclipse installation base directory. In the subfolder `plugins/com.sap.jco_2.17.0/lib` there are two DLL files which must be copied to the Windows system32 folder (e.g. `C:\WINDOWS\system32`). If you are using some other SAP based programs one or both of them may already be present. Make sure the correct version is installed – if necessary, rename the old files.

These libraries will come into affect only after Eclipse is restarted.

Uninstallation

The preferred way of uninstallation is using the configuration manager (Help / Software Updates / Manage Configuration). Select the components you wish to uninstall (e.g. the whole Scripting Tool) and select “Disable” and restart. It is standard Eclipse behavior to first disable a component and in the second step to uninstall it. You have to check the “Show Disabled Features” Icon in the toolbar to see the disabled features, and then you can uninstall them.

Alternatively (not recommended) you can delete the com.sap.scripting.* components (JAR files and folders) in your Eclipse plugin/ and features/ installation folder.

Please note that plugin-specific configuration is kept in a hidden folder named .metadata in your workspace by Eclipse. The SAP Scripting Tool stores the metadata for each service there. You can safely delete the folder .metadata\.plugins\com.sap.scripting.core if you wish. This is not done by the uninstaller.

Welcome Screen

The SAP Scripting Tool adds content to the Eclipse Welcome screen. If you’ve already closes it you can access it via Help / Welcome. Currently the Overview has a short description of the Tool and in the section What’s New a link to the SDN Community page for scripting is provided.

Tutorials and samples will be provided in the appropriate sections of the Welcome screen as well. This section will grow in the future, and we also plan on adding content send in by the community.

Repository and Views

The SAP Scripting Tool adds three new views for interacting with the tool, grouped together in a new perspective. You can select it through Windows / Show Perspective / Other / SAP Scripting Tool, or add the views individually using Windows / Show View / Other under SAP Scripting Tool.

SAP Scripting Repository View

This is the main view for the tool. The known services, which are stored in the local metadata file, are displayed here. You can add a new data source, e.g. a Webservice URL, which will then be analyzed and its information stored in the internal metadata. For a stored service function a script in any supported language can then be generated and written to the file system.

Details View and Method View

These two views provide some more information about the selection in the Repository View from the metadata. The Details View shows some general information about the service, depending on the selection. The Method View is only valid when a single method (or operation in Webservice lingua) is selected and provides information about the expected input and output parameters.

This information is retrieved when importing a data source. If the remote service changes a re-import is necessary.

Repository Connectors and Generators

A repository connector is the component of the tool which connects to a specific service and retrieves the metadata. These repository connectors are Java-based and used inside the SAP Scripting Tool, whereas for the generated script a separate connector for the scripting runtime is required. The tool comes with two repository connectors to access Web services and BAPIs, though others may be added in the future.

You can start a repository connector by choosing “Import Data Source” from the context menu in the Repository View. Choose what you want to import (Web services or BAPIs) and select the repository connector which best fits your needs – e.g. for Web services you can either specify the WSDL via URL or use a locally stored WSDL file. Enter the necessary information – like authentication credentials, if needed – and in the last step you can also change the display name. This name is used to let you more easily identify the services in the repository view, but is not used in the generated scripts.

A generator is the component which creates the actual script which can then be used to access the service. Usually the generated artefacts are written to disk, either in an existing Eclipse Project (using Eclipses Resource capabilities) or to an arbitrary location in the local file system.

When selecting “Launch Script Generator” from the context menu of a method in the Repository View, a list of all available generators for this kind of service is shown. We provided basic generators for Ruby, PHP and Perl as well as a slightly more elaborate generator for PHP which also creates a sample HTML form to access this service. More generators may be provided by us or by others in future.

The user interface of the generator is generator-specific, but usually you select a file destination, some other custom options, and then get one or multiple files.

Please remember that these generated files are only meant as examples and are thus kept as simple as possible. You may use them freely in your projects, but have to make sure proper error handling or security precautions are taken, if needed.

Creating a PHP script for calling a Webservice

This chapter contains a step-by-step guide how to generate a script in PHP for accessing a Webservice. Though we primarily intend to support SAP specific Webservices the tool can of course be used for any Webservice available. In this chapter I’ll make use of this so that you can follow my example regardless of your installed SAP backend system. The webservice used here is the TerraService from www.terraservice.net, which provides a function for getting the coordinates of (some big) cities using the operation ConvertPlaceToLonLatPt.

Tool Installation

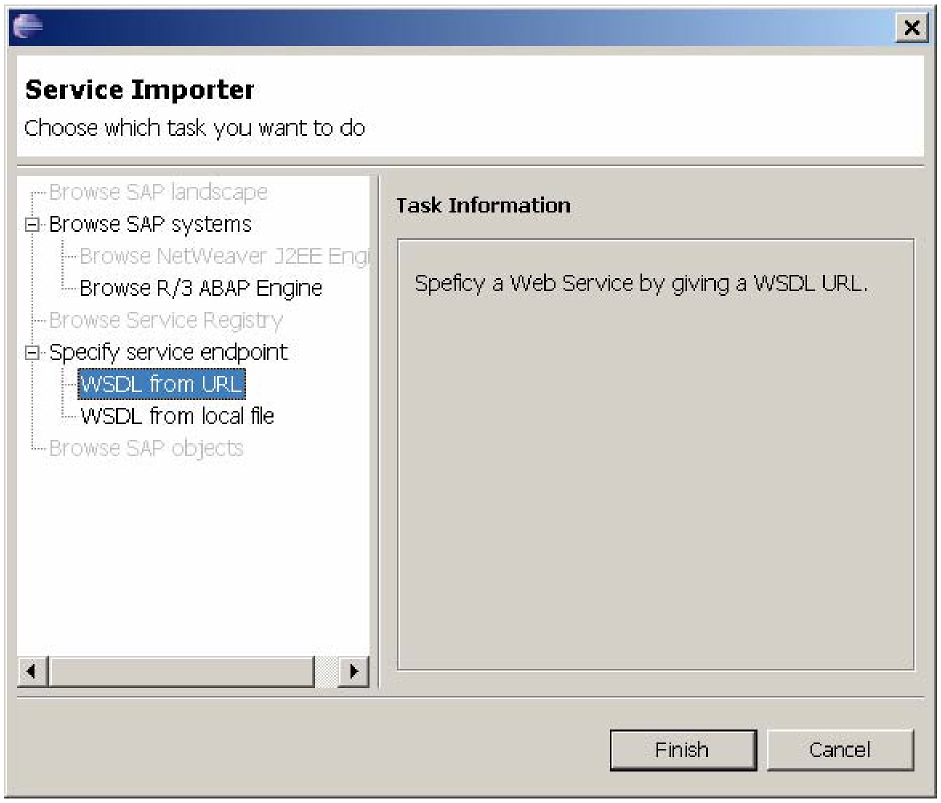
Please follow the instructions above for getting and installing Eclipse and installing the SAP Scripting Tool using the Update Manager. In this example I used the Scripting in a Box package from Craig Cmehil, which contains also an Apache Webserver with integrated PHP.

As a destination for the generated files create a new PHP project named “test”.

Then switch over to the SAP Scripting Tool perspective. Alternatively just add the SAP Scripting Repository View to the PHP perspective.

Connecting to the Web service

Launch the import data source wizard (e.g. through the context menu) and select the option to specify a Web service endpoint using an URL:

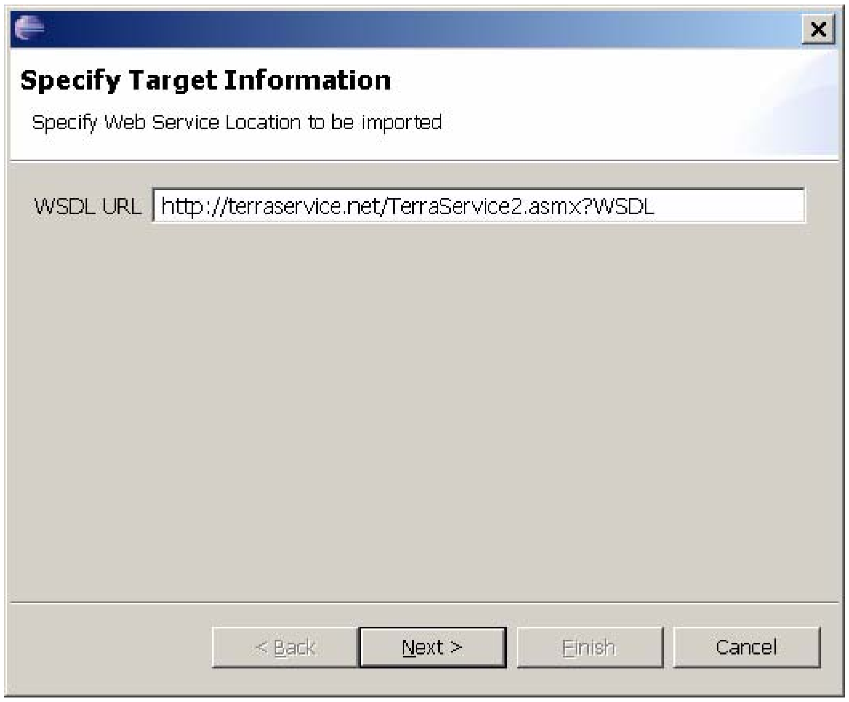


Service Importer

Choose which task you want to do

Speficy a Web Service by giving a WSDL URL.

Enter the TerraService WSDL located at http://terraservice.net/TerraService2.asmx?WSDL (you remembered to set your proxy settings, right?) and press next two times, then Finish, accepting the PortType and the proposed display name.



Specify Target Information

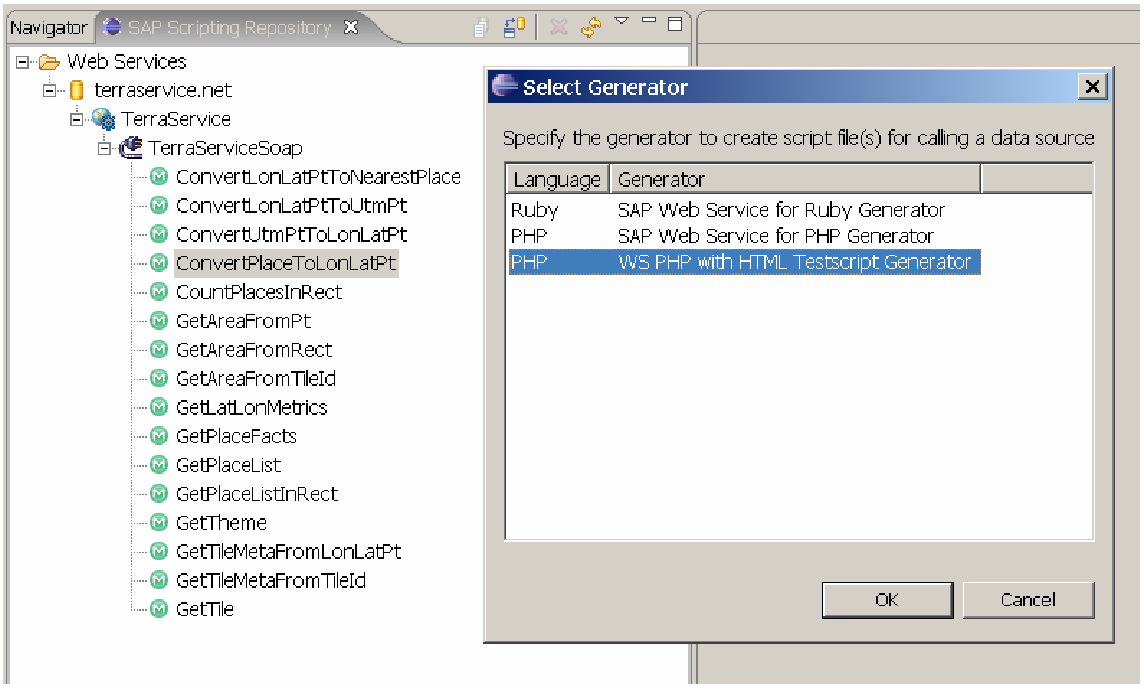
Specify Web Service Location to be imported

WSDL URL: http://terraservice.net/TerraService2.asmx?WSDL

You should now have the TerraService with its provided operations visible in the Scripting Repository View and can inspect the parameters in the Method View.

Generating the script

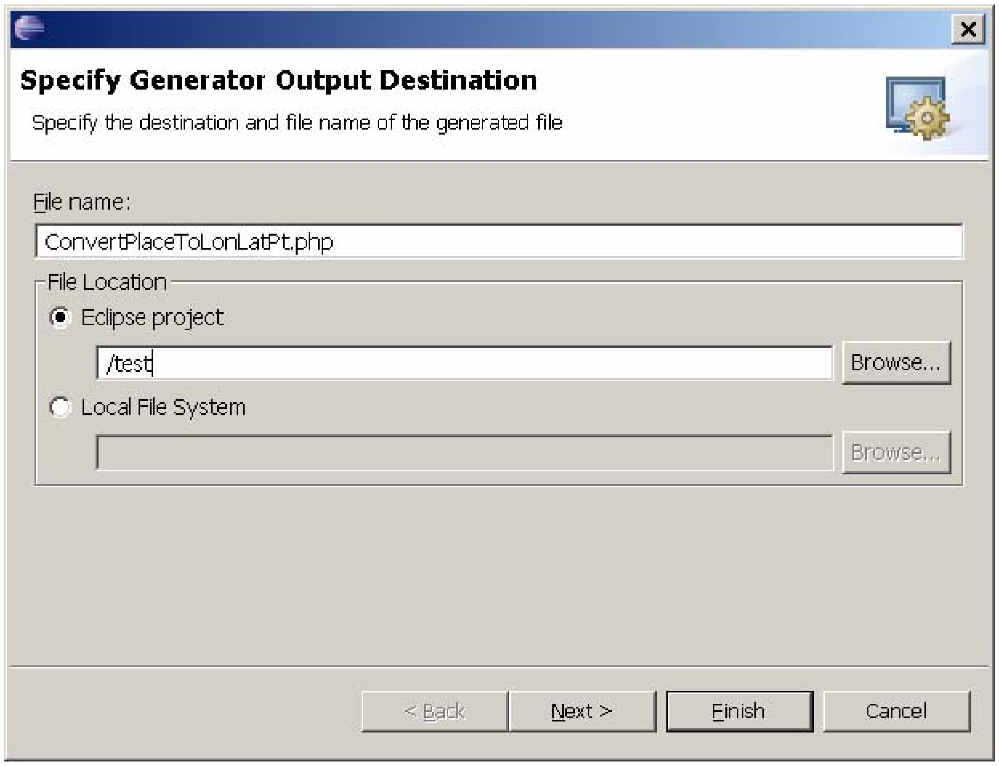
Select the ConvertPlaceToLonLagPt operation and select Launch Script Generator from the context menu. Choose the WS PHP with HTML Test script Generator.



Specify the generator to create script file(s) for calling a data source

Language	Generator
Ruby	SAP Web Service for Ruby Generator
PHP	SAP Web Service for PHP Generator
PHP	WS PHP with HTML Testscript Generator

In the next three wizard pages you can enter generator-specific options. The proposed filename is acceptable. Enter ‘test’ as the project name (or whatever name you’ve chosen).



Specify Generator Output Destination

Specify the destination and file name of the generated file

File name: ConvertPlaceToLonLatPt.php

File Location

[✓ Eclipse project] /test

Local File System

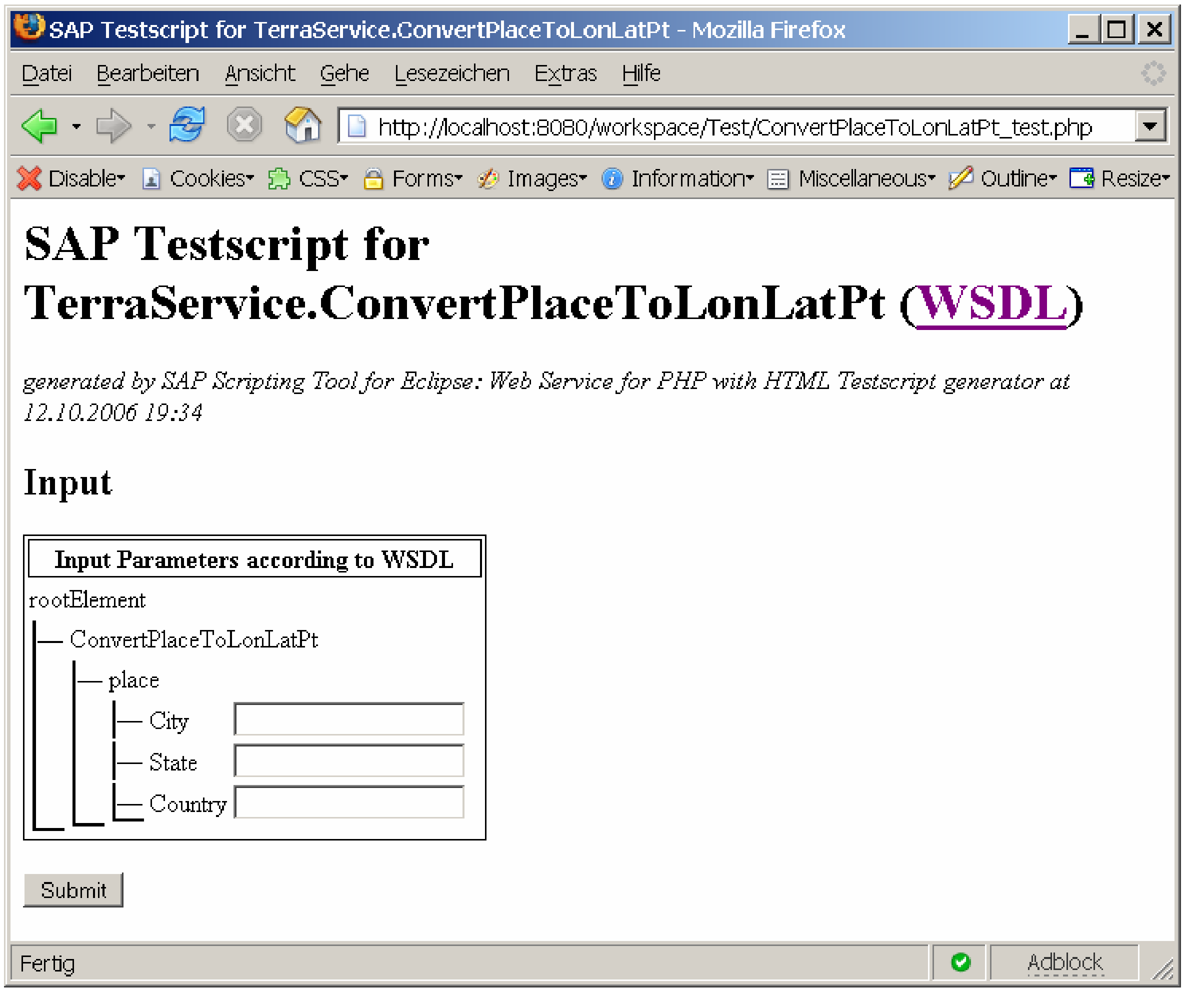
Next page is security. HTTP basic authentication is supported, but we don’t need it for this service. This generator will create the appropriate code to set the authentication options in the PHP5 SoapClient when you select Basic Authentication. You can either enter the login data here and they will be stored directly (and in plaintext) in the generated file, or leave the fields empty. In this case the generated function will expect two additional parameters which the user can enter through the form.

The third and last wizard page lets you opt to have a sample page – called Test Script here – generated. Make sure this option is checked. A filename proposal will be made based on the primary file.

A proxy can also be set here when it's needed. This proxy is used during the runtime of the PHP script to access the Webservice.

After this last step you should have two new files, ConvertPlaceToLonLatPt.php which contains the webservice call and ConvertPlaceToLonLatPt_test.php with the HTML form.

Now start the Apache server, make sure you can access the folder where these new scripts are saved and try them in your browser.


SAP Testscript for TerraService.ConvertPlaceToLonLatPt - Mozilla Firefox

http://localhost:8080/workspace/Test/ConvertPlaceToLonLatPt_test.php

SAP Testscript for TerraService.ConvertPlaceToLonLatPt (WSDL)

generated by SAP Scripting Tool for Eclipse: Web Service for PHP with HTML Testscript generator at 12.10.2006 19:34

Input

Input Parameters according to WSDL

```
rootElement
└─ ConvertPlaceToLonLatPt
   └─ place
      ├─ City     [          ]
      ├─ State    [          ]
      └─ Country  [          ]
```

Enter a city name, e.g. Chicago, and see where it's located.

The result page tries to parse the return parameters according to the metadata as well as printing them out in raw form (using the PHP command var_dump) and showing the transmitted SOAP messages.

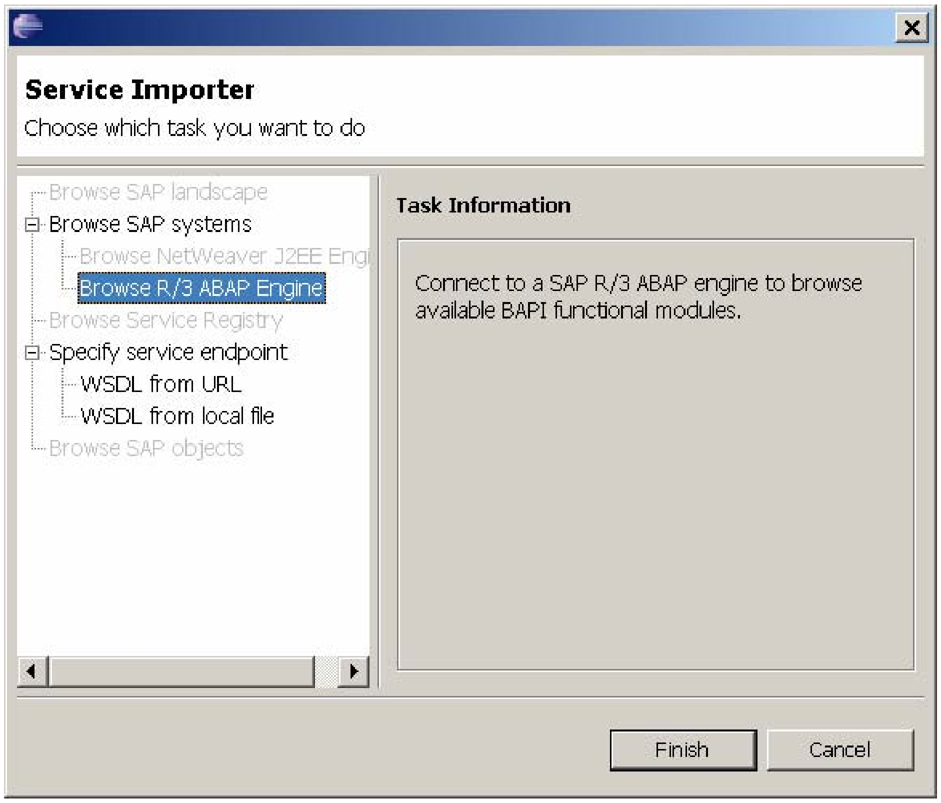
Congratulations. You can now test your Webservice and adapt the source to fit your own ideas.

Creating a PHP Script to access a BAPI

As connecting to an R/3 system always needs authentication information you need to have a system to which you have proper access rights.

Connecting to the ABAP system

Launch the import data source wizard (e.g. through the context menu) and select the option to browse an R/3 ABAP system.



Service Importer

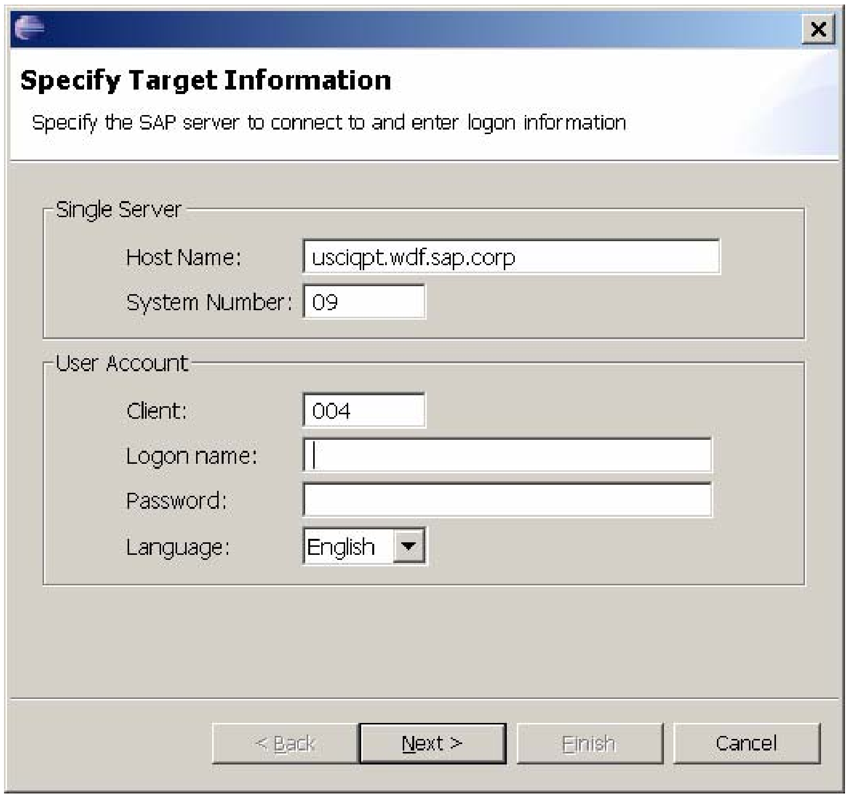
Choose which task you want to do

- Browse SAP landscape
- Browse SAP systems
  - Browse NetWeaver J2EE Engi
  - Browse R/3 ABAP Engine
- Browse Service Registry
- Specify service endpoint
  - WSDL from URL
  - WSDL from local file
- Browse SAP objects

Task Information

Connect to a SAP R/3 ABAP engine to browse available BAPI functional modules.

Enter your logon information and user credentials for this system and press Next.



Specify Target Information

Specify the SAP server to connect to and enter logon information

Single Server

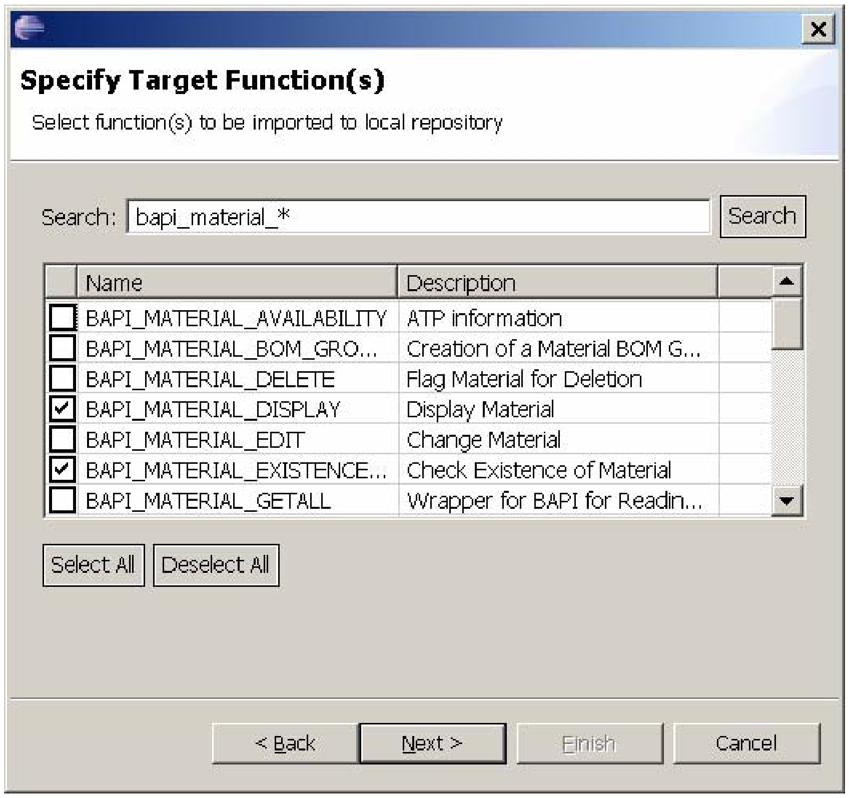
Host Name:	usciqpt.wdf.sap.corp
System Number:	09

User Account

Client:	004
Logon name:	
Password:	
Language:	English

Now you will get a screen where you can select the functions which you are interested in. Since this usually are quite a lot it is recommended to use an appropriate filtering term to narrow the search. I used "bapi_material_*" in this example to show all functions starting with this term.

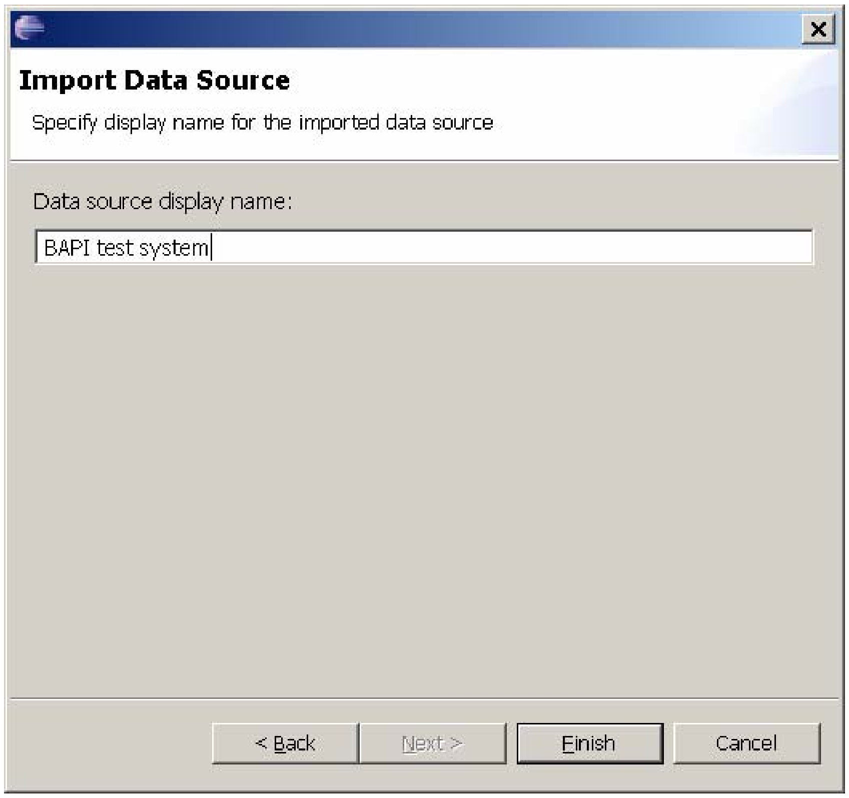
Check all functions you wish to import, then press Next to enter a name under which this selection will be displayed in the Scripting Repository view.


Specify Target Function(s)

Select function(s) to be imported to local repository

Search: bapi_material_*

	Name	Description
☐	BAPI_MATERIAL_AVAILABILITY	ATP information
☐	BAPI_MATERIAL_BOM_GRO...	Creation of a Material BOM G...
☐	BAPI_MATERIAL_DELETE	Flag Material for Deletion
✓	BAPI_MATERIAL_DISPLAY	Display Material
☐	BAPI_MATERIAL_EDIT	Change Material
✓	BAPI_MATERIAL_EXISTENCE...	Check Existence of Material
☐	BAPI_MATERIAL_GETALL	Wrapper for BAPI for Readin...


Import Data Source

Specify display name for the imported data source

Data source display name:

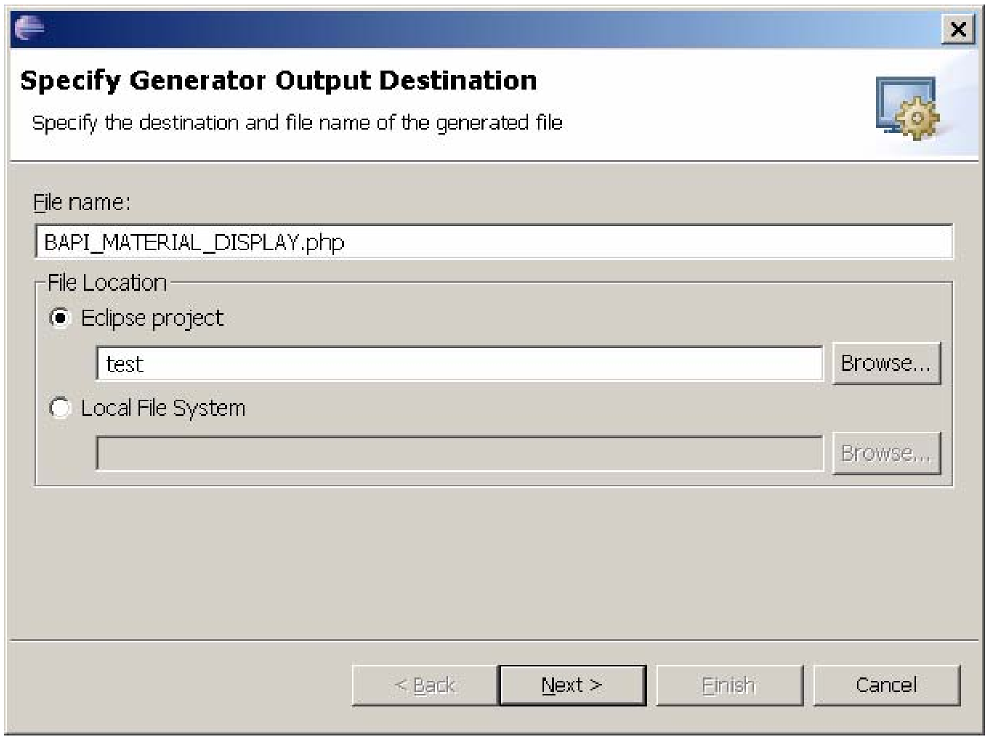
BAPI test system

As with Web services the imported BAPI functions are now visible in the Scripting Repository view and their parameters can be examined in the Method View.

Please note: the connection to the remote system is made when a filter term is entered and ‘Search’ is selected. Wrong connection information or logon credentials will only be discovered at this time.

Generating the script

Select a function which you want to connect to, launch the code generator – currently only PHP is supported as available language – and follow the wizard.



Specify Generator Output Destination

Specify the destination and file name of the generated file

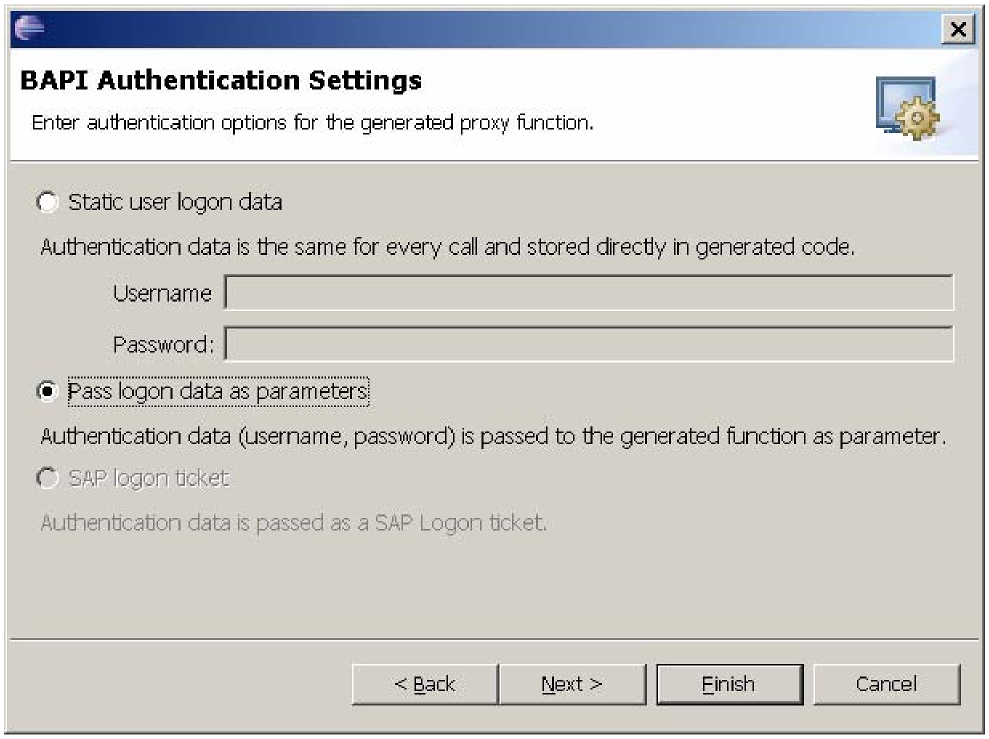
File name: BAPI_MATERIAL_DISPLAY.php

File Location

[✓ Eclipse project] test

Local File System

Depending on your application it might be wanted to always access the system with the same user or to allow for different user to be used. When you select the first option, the username and password are stored in plain text in the scripting file, thus it is recommended to only use this for test systems or during development.



BAPI Authentication Settings

Enter authentication options for the generated proxy function.

Static user logon data

Authentication data is the same for every call and stored directly in generated code.

Username

Password:

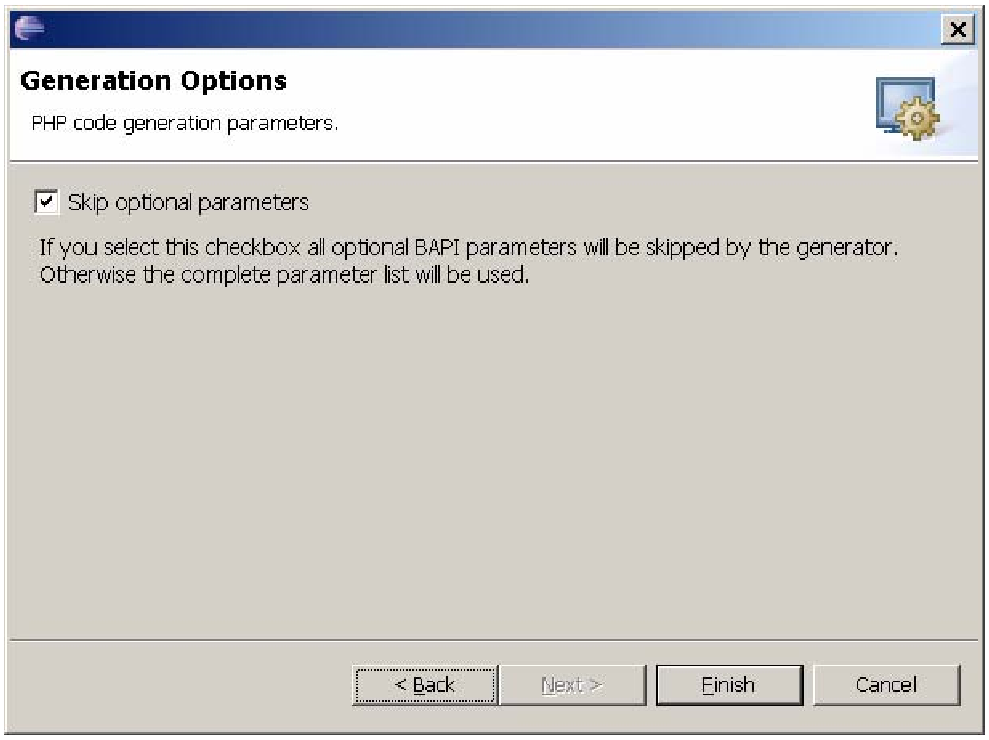
[✓ Pass logon data as parameters]

Authentication data (username, password) is passed to the generated function as parameter.

SAP logon ticket

Authentication data is passed as a SAP Logon ticket.

BAPIs often contain a vast amount of parameters, many of them optional. To reduce the number of parameters you have to fill in your script you can instruct the generator to skip all parameters marked as optional.



[✓] Skip optional parameters

If you select this checkbox all optional BAPI parameters will be skipped by the generator. Otherwise the complete parameter list will be used.

Congratulations. You now have a PHP function to access the BAPI and can use it in your own scripts to connect to a SAP backend.

Outlook

In this second release we have included first BAPI connectivity, but there are a lot of things on our minds which we could add. Generators for other languages will follow, of course. Additional to that we plan on adding enhanced browsing facilities – like browsing in SAP J2EE systems for available Web services, contacting service registries or even start from the very top, browsing the whole SAP landscape (using the SAP Landscape Directory, SLD).

On the other side making calls to BAPIs simpler is also a goal. Providing more documentation, automatic default values, or customizable parameter subsets are things we though about – the ‘skip optional parameters’ is just the beginning.

But first of all we want to make this a tool for you, the community. And therefore we would like to have your opinions, your use-cases, feature requests as well as bug reports. Feedback is very welcome, and SDN is the best place to give it.

Participating

The SDN is your opportunity to influence the way this tool evolves. Make your thoughts heard and give us feedback.

We intend to make the whole tool Open Source. Unfortunately, due to open legal questions we are currently not able to do this. The use of the tool as well as the generated output and the creation of external extension plugins are free and we hope to deliver the source code as Open Source at a later date, too.

Extending the Scripting Tool

The SAP Scripting Tool is meant to provide a platform for script developers on top of the Eclipse platform. It makes heavy use of the Eclipse extension mechanism with Extension Points where plugins can hook into.

It is a clear goal to allow for a wide range of possible plugins. The framework provides the general idea of having a repository connector which fills in metadata in a local cache and consumers like generators which acts upon this data. Other repository connectors, other scripting languages, new Views or even completely different plugins can be provided.

We will provide more detailed information about the internal architecture and how external plugins can extend the Scripting Tool in a later article.

Feedback is very welcome, so if you have an interesting project and need Extension Point support or other changes to the framework please comment in SDN.

Related Content

Scripting Language page, forums and blogs at SAP SDN Technology section

Article series from Vasil Bachvarov about Scripting Languages

‘Scripting in a Box’ from Craig Cmehil

All available on www.sdn.sap.com