[SAPlink Plugin Developer Documentation](https://wiki.scn.sap.com/wiki/display/ABAP/SAPlink+Plugin+Developer+Documentation)

[Перейти к концу метаданных](https://wiki.scn.sap.com/wiki/display/ABAP/SAPlink+Plugin+Developer+Documentation" \l "page-metadata-end)

* Создал(а) Guest, редактировал(а) [Gregor Wolf](https://wiki.scn.sap.com/wiki/display/~w5c5d4t) [дек 29, 2013](https://wiki.scn.sap.com/wiki/pages/diffpagesbyversion.action?pageId=14059&selectedPageVersions=6&selectedPageVersions=7)

[Переход к началу метаданных](https://wiki.scn.sap.com/wiki/display/ABAP/SAPlink+Plugin+Developer+Documentation" \l "page-metadata-start)

Introduction

SAPLink already features a wide range of plugins. However sometimes it is necessary to develop a new one. This page outlines the basic steps.

Four simple steps

**1) Create the plugin class**

Every plugin is developed as a normal ABAP class which inherits from the class ZSAPLINK. Every plugin class conforms to the convention:

ZSAPLINK\_<your plugin type>

For example ZSAPLINK\_CLASS for a plugin which imports and exports ABAP Object classes. Now, go to the transaction SE80. Create the plugin class and let it inherit from the class ZSAPLINK.

**2) Register the plugin**

SAPLink looks at the table VSEOEXTEND and selects all classes which inherit from the class ZSAPLink. This means that SAPLink will automatically register your plugin!

**3) Write the plugin**

After you have created the new plugin class you can start coding. In order to get your plugin working you will have to implement at least 5 template methods. the term "template method" describes a so called design pattern and you can read more about it on [Wikipedia](http://en.wikipedia.org/wiki/Template_method_pattern). It basically means that the ZSAPLink class features abstract methods which will be called by the SAPLink framework during various occasions. However the class ZSAPLink only provides the templates for those methods and you have to provide the implementation them. This design makes it possible to treat all plugin classes in the same way. The following list gives a brief explanation for each necessary method.

CHECKEXISTS - Checks if the object already exists in the system  
CREATEOBJECTFROMIXMLDOC - Parses the XML document and creates the object in the system  
CREATEIXMLDOCFROMOBJECT - Converts the object into a XML document which can be saved on the client machine  
DELETEOBJECT - Deletes the object  
GETOBJECTTYPE - Returns the type of the object for example the class plugin returns: CLAS

How to implement these methods is up to you but it might be a good idea to see how the Guru's are doing it. Take a look at the plugin for programs written by Rich Heilman class ZSAPLINK\_PROGRAM.

**4) Testing**

Make sure to test your plugin carefully. It might be a good idea to create an object especially for testing which features all possible (edge) cases and include it when you release the plugin.

SAPlink Plugin Development Frequently Asked Questions

**General Questions**

**I would like to start to developing a SAPlink plugin. Is someone already working on the same issue?**

First check the plugins available for download in the code repository. If you do not find the plugin you are looking for, check in the [SAPlink discussion group](https://www.assembla.com/spaces/saplink/messages) for any mention of someone working on the plugin. If nothing is available you can either post a topic in the discussion group to introduce your plugin idea or, if you are impatient, you can start writing it the plugin.

In any case, when things start moving, we suggest you create a separate project for the plugin or join one that corresponds to the area you are working on.

**Technical Aspect**

**How do I start a plugin project?**

Project name : use saplink-\*  
Labels : Include "ABAP", "SAPlink", "SAP" and "SDN" as labels.  
Discussion groups : Add a link to [SAPlink discussion group](https://www.assembla.com/spaces/saplink/messages)