

ENOVIA SmarTeam/SAP Adaptor – SS9

Implementation Guide

**BPA Delivery 7 for V5R19 (V 5.7)
Version 9.9**

Table of Contents








ENOVIA SmarTeam/SAP Adaptor - Version 9 Introduction	4
Adaptor - Installation	5
Installation and Prerequisites	5
RFC function module returning SYST fields	7
Release Notes	8
The Destination Sections	13
The destination section	13
Modifying logon data "Client", "User name", "Password" and "Language"	15
Using SNC and Single Sign-On	17
Licensing	18
LUM Server: The server should have LUM 4.6.8.3 or above installed	18
Register DLL	18
Request a license with following information	18
For Nodelock type of license	18
For Concurrent type of license	19
The Activity Sections	23
Sample Script	25
External Calls	26
Calling the ENOVIA SmarTeam/SAP Adaptor from other applications	26
Calling the ENOVIA SmarTeam/SAP Adaptor from the command line	28
Encrypting and checking passwords from external applications	28
Error Handling	30
General Section	32
Problem Solving	35
Generating trace	35
Problems related to SAP Messages	36
Problems related to SAP Exceptions	39
Problems related to SAP Adaptor Messages	40
Other issues	41
Error when storing files in SAP vaults or storage categories	42
For your analysis, please take also into account the following SAP notes:	43
Available Activity Functions	44
Navigator	44
Batch Input Functions	46
Class Functions	47
Classification Functions	48
Document Functions	51
Engineering Change Master Functions	64

Excel Functions.....	68
File Functions	71
Function Module Functions	74
Grid Functions	76
Material Functions	78
Project Functions	94
Smarteam Functions	96
Control Functions.....	103
Miscellaneous Functions	108
Available Inline Functions.....	114
Navigator	114
Available Variables and Definitions	122
Preliminary and Navigator	122
ENOVIA SmarTeam Data.....	123
SAP Data	124
General Data	124
Class Data	124
Classification Data	125
Document Data	126
EC Master Data.....	129
Function Module Call Data	130
Material Data	130
Project Data	133
Other Data	138
Excel Data.....	138
File Data	139
Grid Data	140
System Data	140
Definitions	140
Sample Configurations	143
Navigator and Preliminary	143
SAP Documents, SAP Vaults and Storage Categories, Accessing ENOVIA SmarTeam Files	144
SAP Materials.....	149
SAP Bill of Material	152
SAP Projects	158
ENOVIA SmarTeam Objects, Structures, Queries	160
Miscellaneous.....	161

ENOVIA SmarTeam/SAP Adaptor - Version 9 Introduction

The ENOVIA SmarTeam/SAP Adaptor is a tool for integrating ENOVIA SmarTeam into an SAP R/3 system in a very flexible manner. It is a client-based application built on the SAP R/3 RFC online library. It can handle documents and document structures, materials and bill of materials, engineering change masters, and classification. It can perform batch (direct) inputs and raw function module calls, manipulate files, and modify ENOVIA SmarTeam objects. Its has functions to control the program flow and miscellaneous functions like displaying messages, evaluating arithmetic expressions, storing definitions, calling external applications, and many more. Thereby the ENOVIA SmarTeam/SAP Adaptor is customizable to a very high degree. Due to its granularity most customer needs can be realized without any programming effort. But if needed, the ENOVIA SmarTeam/SAP Adaptor allows for a simple and seamless extension using standard programming languages like Visual Basic.

To reach an understanding of how the ENOVIA SmarTeam/SAP Adaptor works and how it can be configured

-  see how to [Install the ENOVIA SmarTeam/SAP Adaptor](#)
-  see how to specify a connection to an R/3 system in the [destination section](#)
-  see how to configure action sequences in [activity sections](#)
-  see [Sample Script](#) to attach the ENOVIA SmarTeam/SAP Adaptor to ENOVIA SmarTeam events
-  optional: see how the ENOVIA SmarTeam/SAP Adaptor can be called from [other applications](#) or from the [command line](#)
-  optional: see how [error handling](#) is done in the ENOVIA SmarTeam/SAP Adaptor.
-  And in case of problems, errors or unexpected behavior: see these hints for [problem solving](#).

Adaptor - Installation

Installation and Prerequisites

To install the ENOVIA SmarTeam/SAP Adaptor

- ✚ copy files smSAPif.exe, smSAPifq.exe, smSAPifc.exe and smSAPif.txt to any desired directory;
- ✚ copy configuration file smSAPif.ini provided by SmarTeam to the same directory or to ENOVIA SmarTeam's script directory;
- ✚ register files smSAPif.exe and smSAPifq.exe by double clicking them or by executing them from the command line (important: if you change the installation directory later, if you are, for instance, switching between test and production environment, there might be a problem with Windows registry that the new installation directory is not updated although you do not see an error message; please refer to [problem solving](#) chapter);
- ✚ make sure, that SAP library LibRfc32.dll version 6.20 or higher is installed (the library can be found in installation options under "Development Tools→RFC SDK Libraries");
- ✚ if you want to use SAP dialog functions, SAPGUI must be installed too;
- ✚ if you want to transfer files into SAP vaults or storage categories, copy programs sapftp.exe and saphttp.exe into the smSAPif directory (important: if you have a Unicode SAP system, you need to download the Unicode versions of sapftp.exe and saphttp.exe from SAP. You cannot use the non-Unicode versions that are installed with SAPGUI);
- ✚ verify, that the Visual Basic 6 Runtime Libraries are installed;
- ✚ Operating Systems:
 - ↳ Windows XP SP2 Professional x86 (32bit) or x64 (64bit) for demo purposes only
 - ↳ Windows Server 2003 SP2 or 2003 R2 SP2 x86 (32bit), Standard Edition or Enterprise Edition
 - ↳ Windows Server 2003 R2 x64 (64bit), Standard Edition or Enterprise Edition;
- ✚ maintain the destination section(s) of your smSAPif.ini file to fit to your SAP installation;
- ✚ make sure that all SAP users using the ENOVIA SmarTeam/SAP Adaptor have proper SAP permissions (the users must have authorization object S_RFC with value *);
- ✚ hook the activities configured in the smSAPif.ini file to ENOVIA SmarTeam hooks using ENOVIA SmarTeam's script maintenance like shown in the [sample script](#) or include the activities in your Visual Basic or other programs as explained [here](#).

In case you do not want to give full permissions (value *) to RFC authorization object S_RFC, you have to specify the following function groups according to your needs:

Activity	Function Groups	Function Module
Batch Inputs / Call Transaction	SBDC	ABAP4_CALL_TRANSACTION
BOMs	CADR, CSAP	CAD_CHANGE_BOM_WITH_SUB_ITEMS CAD_CREATE_BOM_WITH_SUB_ITEMS CAD_DISPLAY_BOM_WITH_SUB_ITEMS CSAP_BOM_ITEM_MAINTAIN CSAP_MAT_BOM_*
Classes	CADR , CLBP	BAPI_CLASS_GET_CHARACTERISTICS BAPI_CLASS_SELECT_OBJECTS RFC_SELECT_OBJECTS_VIA_CLASS
Classification	CACL, CLBPA	CACL_OBJECT_* CACL_CLASSIFICATION_SAVE BAPI_OBJCL_*
Documents	BAPT, CADR, CLBP, CVBAPI	BAPI_DOCUMENT_* BAPI_CLASS_GET_CHARACTERISTICS BAPI_TRANSACTION_COMMIT BAPI_TRANSACTION_ROLLBACK RFC_CHANGE_DOCUMENT_MASTER RFC_CREATE_DOCUMENT_MASTER RFC_DISPLAY_DOCUMENT_MASTER RFC_SELECT_DOCUMENT_MASTER RFC_SELECT_OBJ_VIA_MATCHCODE RFC_EXPLODE_PRODUCT_STRUCTURE RFC_DISABLE_EMPTY_SCREEN RFC_DETERMINE_ACCESS_PATH
Engineering Change Masters	CADR , CCAP	CCAP_ECN_HEADER_* RFC_CHANGE_CHANGE_MASTER RFC_CREATE_CHANGE_MASTER RFC_DISPLAY_CHANGE_MASTER RFC_SELECT_OBJ_VIA_MATCHCODE RFC_EXPLODE_PRODUCT_STRUCTURE RFC_DISABLE_EMPTY_SCREEN
Materials	1001UEB, BUS1001, CADR	BAPI_MATERIAL_SAVEDATA BAPI_MATERIAL_EXISTENCECHECK BAPI_MATERIAL_GETLIST BAPI_MATERIAL_GET_DETAIL BAPI_MATERIAL_GETINTNUMBER BAPI_STDMATERIAL_GETINTNUMBER RFC_CHANGE_MATERIAL_MASTER RFC_CREATE_MATERIAL_MASTER RFC_DISPLAY_MATERIAL_MASTER RFC_SELECT_OBJ_VIA_MATCHCODE RFC_EXPLODE_PRODUCT_STRUCTURE RFC_DISABLE_EMPTY_SCREEN
Project System	2001, 2054	BAPI_PROJECTDEF_* BAPI_PROJECT_GETINFO
System functions used by the SAP Adaptor	CADR, RFC1, RSAK, RSPC_API, SDTX, SRFC	RFC_SYSTEM_INFO RFC_GET_SAP_SYSTEM_PARAMETERS RFC_GET_STRUCTURE_DEFINITION_P RFC_READ_TABLE

Activity	Function Groups	Function Module
(mandatory!)		RSAP_GET_MSG_TEXT RSPC_API_SYSTEM_MESSAGE_GET
Own developments	Z_SMARTEAM, for instance	

RFC function module returning SYST fields

The ENOVIA SmarTeam/SAP Adaptor uses some R/3 RFC function modules which are not BAPIs. That means they do not return either a BapiReturn1 structure nor a BapiRet2 structure nor any other error message but just an exception called ERROR. To have access to the message issued by the RFC function the ENOVIA SmarTeam/SAP Adaptor needs an RFC function module to access some of the fields of structure SYST. The name of that function module has to be supplied in the destination section.

Check in your SAP system, if function module "RSPC_API_SYSTEM_MESSAGE_GET" does exist. If it exists, specify this name as MessageModule in the destination section. If it does not exist, create a new RFC function module from this code:

```
function z_system_message.
*"-----
*" EXPORTING
*"   VALUE(SUBRC) LIKE SY-SUBRC
*" TABLES
*"   MESS_TAB STRUCTURE BDCMSGCOLL
*" Switch RFC enabled on!
*"-----

subrc = sy-subrc.
clear  mess_tab.
refresh mess_tab.
mess_tab-tcode   = sy-tcode.
mess_tab-msgtyp  = sy-msgty.
mess_tab-msgspra = sy-langu.
mess_tab-msgid   = sy-msgid.
mess_tab-msgnr   = sy-msgno.
mess_tab-msgv1   = sy-msgv1.
mess_tab-msgv2   = sy-msgv2.
mess_tab-msgv3   = sy-msgv3.
mess_tab-msgv4   = sy-msgv4.
append mess_tab.
endfunction.
```

Since the function module will be called by the ENOVIA SmarTeam/SAP Adaptor, please make sure that it is RFC enabled.

Release Notes

Version 8.2 (BPA Drop 1)

Version 8.3

- ~ Missing function [Classification.Clear](#) added.
- ~ Number of importing, exporting, changing and tables parameters in function calls increased to 50.
- ~ New inline function [ViewfileExtension](#) added. The function gets the extension of the view file depending on a ENOVIA SmarTeam native file type.
- ~ New values added to [Material.Basicdata](#): MatlCat, EmpziesBom, BasicMatlNew
- Section "[Error when storing files in SAP vaults or storage categories](#)" added to the documentation

Version 8.4

- ~ System variables and environment variables #System.---# can be used in [TraceDir setting](#) in section [\[General\]](#).
- ~ Missing function [Document.AddLongtext](#) added and documentation updated.
- ~ Function [SplitString](#) modified. See documentation.

Version 8.5

- ~ New loop: Document.Files2 available after [Document.GetDetail2](#)
- ~ New variables available inside this loop: [#Document.Files2.xxx#](#)
- ~ New function available inside this loop: [Document.DeleteFile2](#) sets the DeleteFlag for the current document file
- ~ New function [Document.GetStructure](#)

Version 8.6 (BPA Drop 2)

- ~ Function [Material.AddBOMSubItemData2](#) and other needed functionality for handling BOM sub items added

Version 8.7

- ~ Possible length of SAP password extended to 16 characters

Version 8.8

- ~ Function [Material.ReadBOM2](#) modified in order to return an error message if BOM does not exist
- ~ New Variable [#SAP.LastBapiMessage#](#)
- ~ Documentation update in BOM sample configurations

Version 8.9

- ~ New function [Commit](#) to switch commit of BAPI transactions on and off.
- ~ New [encrypting algorithm](#) for passwords. Passwords can now be 20 characters long.

Version 8.10

- ~ New functions PWCrypt and PWCheck exposed by SapLink class. [Passwords can now be encrypted and checked from external applications.](#)

Version 8.11

- ~ BOM functions adapted to SAP release 7.0 (ECC 6.0). Field length modifications necessary for all "Matnr" fields.

Version 8.12

- ~ New setting "UseLongMaterialNumbers" in [section \[General\]](#). If set to ON, 40 characters are used for the material number instead of 18 characters.
- ~ New setting "ConvertMaterialNumbers" in [section \[General\]](#). If set to OFF, conversion of all "Matnr" fields is suppressed.
- ~ Documentation for section [\[General\]](#) added.

Version 8.13

- ~ New language IT (Italian) added to SAP Adaptor internal messages.

- ~ New inline functions [\WinDateX{SAPDATE}](#) and [\SapDateX{WINDATE}](#) allowing the conversion of Windows dates and SAP dates in external representation. (The existing inline functions [\WinDate{SAPDATE}](#) and [\SapDate{WINDATE}](#) - the functions without the X in their names - convert between Windows dates and SAP dates in internal representation YYYYMMDD).

Version 8.14 (BPA Drop 3)

- ~ New function [Material.AddBOMSubItemData](#) added and all necessary changes made in order to manage BOM sub items through BOM functions [Material.OpenBOM](#), [Material.ReadBOM](#), [Material.SaveBOM](#), etc (CSAP_MAT_BOM... function modules). This feature is available on SAP systems 7.0 or higher.

Version 8.15

- ~ New function [SwitchSAPUser](#) added. SAP user, password, language and dialog mode can now be changed inside an activity section.

Version 8.16 (BPA Drop 4)

- ~ New interface functions in classes smSAPif.SapLink and smSAPifq.SapLink: Function [Execute2\(Destination, Operation, DialogMode, LinkMode, ProgressInfo, ArgString\)](#). In this function definition parameters SmSession and SmObject are missing so that these functions can be called without having an SmSession and an SmObject. This is useful if the SAP Adaptor needs to be called from Java Script, for instance.

Version 8.17

- ~ New language SV (Swedish) added to SAP Adaptor internal messages.
- ~ In case of an unhandled error (no error handler defined) SAP Adaptor now returns exit code 1 and the error message on ArgString. Before this change exit code 0 and no error message was returned.
- ~ New loop: [Document.CharacteristicValues](#) available after [Document.GetDetail2](#)
- ~ New variables available inside this loop: [#Document.CharacteristicValues.xxx#](#)
- ~ New loop: [Document.ClassAllocations](#) available after [Document.GetDetail2](#)
- ~ New variables available inside this loop: [#Document.ClassAllocations.xxx#](#)

Version 8.18

- ~ New parameters DocBOMChangeNumber, DocBOMRevisionLevel, DocBOMValidFrom added to function [Document.AddBasicData2](#)

Version 8.19

- ~ New coding of password encryption/decryption algorithm. Old passwords still work.

Version 8.20

- ~ New language KO (Korean) added to SAP Adaptor internal messages.
- ~ New settings "DecimalSignWIN", "ThousandSignWIN" and "DateFormatWIN" in [section \[General\]](#) of the ini file.

Version 8.21

- ~ Bug fixed. The bug occurred when certain inline functions ([\Documentkey{,,,}](#), [\Mask{,}](#), [\Unmask{,}](#), [\Delete{,}](#)) were used in Set or Define statements.
- ~ Error handling in function [Material.ReadBOM2](#) modified again. If no BOM exists, flag BOMExists is set to " " but the error message is suppressed. In all other cases error and information messages are handled properly.
- ~ New variables [#SAP.DecimalSign#](#), [#SAP.DateFormat#](#), [#System.DecimalSign#](#), [#System.DateFormat#](#)
- ~ New function [SmarTeam.GetLastRevisionObject](#) and new variables [#SmarTeam.LastRevisionObject.<Attribute>#](#)
- ~ New language TR (Turkish) added to SAP Adaptor internal messages.

Version 8.22

- ~ Internal queuing of requests in smSAPifq modified.

Version 8.23

- ~ Similar to SAP user name, password and language also the [client used for SAP logon](#) can be modified in several ways.

Version 8.24

- ~ New SAP connection parameter ConnectParam introduced. See description in [Destination Sections](#).

Version 8.25 (BPA Drop 5)

- ~ "Language bug" fixed. If multiple requests were queued in smSAPifq, the SAP logon language was reset to its default value every time a new request was stored.

Version 8.26

- ~ Internal coding of Document.GetDetail2 adapted to SAP needs.

Version 8.27

- ~ Problem with / inside connection parameter ConnectParam solved.
- ~ New setting "BOMEvalWarning" in [section \[General\]](#) of the configuration file.

Version 8.28

- ~ In connection parameter ConnectParam, a variable #USER# can now be substituted with the value of connection parameter "user". See section [Using SNC and Single Sign-On](#) for more information.

Version 8.29

- ~ The "Language bug" fix in 8.25 caused a problem when converting strings to BCD numbers. This was corrected.
- ~ Reading certain SAP messages around BOM creation failed for languages where SAP internal language key differed from first character of ISO language. This was corrected.
- ~ New inline function [\SapInternalLanguage](#) added. The function converts an ISO Language key into SAP's internal language key.
- ~ After reading document characteristics with Document.GetDetail2 it is now possible to directly access a characteristic value through #Document.CharacteristicValue.<Name># instead of looping through entire characteristic list.
- ~ Format of configuration file smSAPif.ini can now be either ANSI or Unicode. This now allows for special characters in the configuration file too (as it is in smSAPif.txt).
- ~ Trace directory for SAP trace files is now set to SAP Adaptor's trace directory.

Version 8.30

- ~ New function [Document.Select](#) for performing a CV04N like search.
- ~ New function [Class.SelectObjectsDialog](#) for performing a dialog search via classification.
- ~ New inline functions [\ExpandDocumentKey](#) and [\ExpandDocumentKey2](#) added. The function converts a document key into 4 individual document key fields separated by comma. The result of these inline functions can therefore directly be used in several Document.Add... statements.

Version 8.31

- ~ "Order By" clause in function [SmarTeam.GetList](#) can now be appended by "Asc" or "Desc" indicating ascending or descending ordering.
- ~ SAP LibRfc32 function RfcIsoLangToSap fails on Asian operating systems and was therefore replaced by own code.

Version 8.32

- ~ A call to [Material.AddBasicData](#) automatically issues a call to [Material.AddHeadData](#) with the first three arguments material number, material type and industry sector. The call the other way round has been already implemented in a former version.
- ~ New parameter FunctionModule_<Name> in [section \[General\]](#) of the configuration file. You can now define alternative function modules to be used.

Version 8.33

- ~ Error handling in functions [Material.OpenBOM](#) and [Material.ReadBOM](#) improved. SAP message (29)001 "BOM not found for this material/plant/usage" is now handled automatically.
- ~ The external representation of Binary Coded Decimal (BCD) numbers was turned into a more common one. See new setting "KeepOldBCDFormat" in [section \[General\]](#) of the configuration file.

Version 9.0 (BPA Delivery Version 5.6)

- ~ New script engine. Affected functions: All [Control Functions](#) except New, SetErrorHandler and Sleep. Due to re-implementation of functions Loop/Next, looping at all tables is affected. Due to re-implementation of condition checking, functions [Material.UpdateBOMItem](#) and [SmarTeam.GetStructure\(Plus\)](#) with ReplacePhantomCondition are affected. Due to re-implementation of ini file reader, inline function [\Lookup](#) is affected.
- ~ Improved error reporting. Together with the error itself, line number and section are reported. A new text file smSAPif.txt must be installed together with this version.
- ~ The "dot" technique already implemented for reading SmarTeam attribute values (like in CN_ITEM.CN_CHANGE_NUMBER.CN_ID) can now be used in function [SmarTeam.SetData](#) for setting attribute values.
- ~ New settings "MessageLevelReferenceNotFound" and "MessageLevelReferenceNotUnique" in [section \[General\]](#) of the configuration file for controlling error situations when using previously described "dot" technique.
- ~ [Return values of Grid](#) modified on cancel. If the value selection from grid is cancelled, variables #Grid.Row#, #Grid.Col#, #Grid.RowSel#, #Grid.ColSel# now return 0 and #Grid.Value# now returns "".
- ~ New function available inside loop "Document.Links": [Document.DeleteObjectLink](#) sets the DeleteFlag for the current document object link.
- ~ Support for Unicode SAP systems. SAP Adaptor functions have been adopted to support 1 byte and 2 byte Unicode SAP systems.
- ~ Inline function [\SapInternalLanguage](#) was not executed at all. This has been fixed.
- ~ Function [Material.ChangeDialog](#) did not use the EC number set in [Material.AddBasicData](#) but took an empty string instead. This has been fixed.
- ~ Licensing mechanism now based on LUM. Old licensing methods using license file or license string still work.

Version 9.1

- ~ New SAP variable [#Document.BasicData2.RevLevel#](#) available.

Version 9.2

- ~ A bug in function [Material.GetList](#) when searching via manufacturer part number has been fixed.
- ~ Function [Material.GetList](#) now offers "not" condition in select options.
- ~ Information displayed in progress window of smSAPifq was not correct since new queuing technique was introduced in 8.22. This has been corrected.
- ~ New function [Material.CreateRevision](#) for creating a revision of a material master.

Version 9.3

- ~ New functions [ECMaster.AddObjectType](#), [ECMaster.ExistenceCheck](#) and [ECMaster.SaveHeader](#).
- ~ New variable [#ECMaster.Exists#](#).
- ~ New loop: Material.Descriptions available after [Material.GetDetail](#).
- ~ New inline function [\SapExternalLanguage](#) added. The function converts SAP's internal language key into ISO language key.
- ~ New sample configurations added to documentation.

Version 9.4

- ~ New language ZF (Chinese traditional) added to SAP Adaptor internal messages.
- ~ Inline functions [\SapExternalLanguage](#) and [\SapInternalLanguage](#) re-implemented based on LibRfc32 functions.
- ~ SAP Adaptor trace files are now generated in Unicode format.
- ~ New function [Material.UpdateBOM](#).
- ~ Improved progress bar behavior in progress window.

Version 9.5

- ~ Bug fixed in inline function [\Eval{}](#). The function failed when evaluating complex expressions.

- ~ Bug fixed with SAP structure handling on Chinese operating systems.
- ~ Bug fixed when counting downwards in a DoLoop.
- ~ Newer SAP systems have a standard function for reading error messages after exceptions. Therefore, there is no need to create an own [message module](#) in all newer SAP systems.
- ~ Functions [Document.DeleteFile2](#) and [Document.DeleteObjectLink](#) have new condition parameters.
- ~ New [command line options](#) -V for displaying version information and -P <clear password> for encrypting passwords.

Version 9.6 (BPA Delivery Version 5.7)

- ~ Bug fixed in LUM license check.

Version 9.7

- ~ New fields added to [Material.AddClientData](#), [Material.AddPlantData](#) and [Material.AddUnitOfMeasure](#).
- ~ New parameter "ObjectLinksToCopy" in functions [Document.CreateNewVersion](#) and [Document.CreateNewVersion2](#).

Version 9.8

- ~ Script engine partially re-newed with respect to variable and inline function substitution. Special characters no longer need masking. Therefore, inline functions \Mask and \Unmask are not needed any longer. Please check your configuration file smSAPif.ini. If you use \Mask and/or \Unmask inline functions, please have your configuration file adapted.

Version 9.9

- ~ LUM issue fixed. SAP Toolkit hung under some circumstances.

The Destination Sections

The destination section

The configuration file contains one or more destination sections comprising the details of an R/3 connection. If no load balancing is used, a typical destination section looks like this:

```
[ST1]
Ashost           = sapserver
Sysnr            = 00
Client           = 100
User             = joe
Password1        = 1444E8916C0248672989E318E4C3384709A9C3384709A9C338
Language         = EN
License          = LUM
#License         = SAP_TOOLKIT_LICENSE.lic
#License         = PFJQZGVAHEPKUEBDL
WaitOnCommit     = 1
DisableEmptyScreen = 1
AllowedRfcPrograms = sapftp;saphttp
MessageModule    = z_system_message
```

In case of load balancing the following is a typical destination section:

```
[ST1]
LoadBalancing    = 1
LbHost           = saplbhost
LbSysname        = ST1
LbGroup          = PUBLIC
Sysnr            = 00
Client           = 100
User             = joe
Password1        = 1444E8916C0248672989E318E4C3384709A9C3384709A9C338
Language         = EN
License          = LUM
#License         = SAP_TOOLKIT_LICENSE.lic
#License         = PFJQZGVAHEPKUEBDL
WaitOnCommit     = 1
DisableEmptyScreen = 1
AllowedRfcPrograms = sapftp;saphttp
MessageModule    = z_system_message
```

Thus a destination section has the following structure:

```
[Name of Destination]
Key1 = Value1
Key2 = Value2
...
```





Possible keys and values are:

Key	Value	Allowed values	Default value
Ashost	<Application server>	(IP address or server name)	
Sysnr	<System number>	(2 digits)	
GwHost	<Gateway server>	(IP address or server name)	
GwService	<Gateway service>	(2 digits)	
LoadBalancing	<Load balancing flag>	0 (no load balancing), 1 (use load balancing)	0
LbHost	<Load balancing server>	(IP address or server name)	
LbSysname	<Load balancing system name>	(3 characters)	
LbGroup	<Load balancing group>	(? characters)	
Client	<Client>	(3 digits)	
User	<User's name>	(12 characters)	
Password	<User's password >	(8 characters)	
Password1	<User's password encrypted>	(8 characters) Use the program "PWcrypt.exe" in the Tool Kit (to encrypt the password.  PWcrypt.exe	
Language	<Login language>	DE, EN, ES, FR, HE, IT, JA, PT, ... (see SAP documentation)	(Language of operating system)
User_<Section>	<User's name>	(Overwrites the default user for the given activity section)	
Password_<Section>	<User's password>	(Overwrites the default password for the given activity section)	
Password1_<Section>	<User's password encrypted>	(Overwrites the encrypted default password for the given activity section)	
Language_<Section>	<Login language>	(Overwrites the default password for the given activity section)	
Trace	<Trace level>	0 (trace off), 1 (trace on), 2 (plus hex trace), 3 (plus SAP trace)	0
ConnectParam	<Additional connection parameter>	(This string is appended to the connection string. It can be modified the same way as client, user, password and language. See below . It can be used to enable SAP logon via SNC. See below .)	


Key	Value	Allowed values	Default value
License	LUM	(License mechanism now based on LUM. Old licenses using <License file name> or <License string> still work)	LUM
AllowedRfcPrograms	<Allowed RFC programs>	(List of allowed RFC programmes, e.g. sapftp;saphttp)	
MessageModule	<Message function module>	(Name of an RFC function module returning SYST fields)	
DialogMode	<Dialog mode flag>	0 (do not use SAPGUI), 1 (use SAPGUI), 2 (use SAPGUI and prompt for user name and password)	0
DisableEmptyScreen	<Disable empty screen flag>	0 (do not disable empty screen), 1 (disable empty screen)	0
WaitOnCommit	<Wait on commit flag>	0 (do not wait for database commits), 1 (wait for database commits)	1
# anything	<Comment>	(Line is not used)	

Modifying logon data "Client", "User name", "Password" and "Language"

The values for SAP logon parameters "Client", "User", "Password", and "Language" are evaluated according to the following rules (without being mentioned explicitly, logon parameter "ConnectParam" can be modified the same way):

-  The operating system's language is preset as SAP R/3 logon language.
-  If the destination section of the configuration file contains a client and/or user and/or password and/or password1 and/or language definition, the respective value overwrites the operating system value.
-  If the destination section of the configuration file contains a setting "Client_<Section>" and/or "User_<Section>" and/or "Password_<Section>" and/or "Password1_<Section>" and/or "Language_<Section>" definition, the respective value overwrites the general client/user/password/encrypted password/language value.
-  If on the user's client PC an environment variable "SMARTEAM_SAP_CLIENT" and/or "SMARTEAM_SAP_USER" and/or "SMARTEAM_SAP_PASSWORD" or

"SMARTEAM_SAP_PASSWORD1" and/or "SMARTEAM_SAP_LANGUAGE" is set, its value is taken as SAP R/3 client, user name, password, encrypted password, or language.

 If on the user's client PC an environment variable "SMARTEAM_SAP_LOGONDATAFILE" is set, the file specified as value is interpreted similar to the destination section with respect to client, user name, password, encrypted password, and language. That means if the file contains any of the lines (<Section> is the name of the activity section to be executed)

```
Client      = 100
User        = myself
Password    = clear
Password1   = 1444E8916C0248672989E318E4C3... (encrypted)
Language    = EN
```

```
Client_<Section>    = 100
User_<Section>      = myself
Password_<Section>  = clear
Password1_<Section> = 1444E8916C0248672989E318E4C3... (encrypted)
Language_<Section>  = EN
```

the respective value is used as SAP R/3 logon client, user name, password, encrypted password, or language.

Client, user, password/encrypted password and language can be specified in three different kinds demonstrated in the following examples:

```
Client      = 100
Client      = SMARTEAM.USER.CN_SAP_CLIENT
Client      = SMARTEAM.USER.CN_SAP_CLIENT/100

User        = joe
User        = SMARTEAM.USER.LOGIN
User        = SMARTEAM.USER.LOGIN/joe

Password    = clear
Password    = SMARTEAM.USER.CN_SAP_PASSWORD
Password    = SMARTEAM.USER.CN_SAP_PASSWORD/clear

Password1   = 1444E8916C0248672989E318 ... (Encrypted)
Password1   = SMARTEAM.USER.CN_SAP_PASSWORD
Password1   = SMARTEAM.USER.CN_SAP_PASSWORD/1444E8916C02486729
            .... (Encrypted)

Language    = EN
Language    = SMARTEAM.USER.CN_SAP_LANGUAGE
Language    = SMARTEAM.USER.CN_SAP_LANGUAGE/EN
```

The first example of each group passes a constant, fixed value as client, user name, (encrypted) password and language. The second examples specifies client, user name, (encrypted) password

and language by means of ENOVIA SmarTeam attributes. All ENOVIA SmarTeam attributes (except structure, link and list attributes) described [here](#) can be used. The third examples are combinations of ENOVIA SmarTeam attribute and constant value as fallback meaning that the ENOVIA SmarTeam attribute is evaluated and that the fallback value is taken, if the ENOVIA SmarTeam attribute is not found or empty.

Due to this flexibility in defining user names and passwords it is possible to use different SAP accounts for different ENOVIA SmarTeam users but it is also possible to use the same SAP account for all ENOVIA SmarTeam users. Please make sure that you stay in line with SAP's license conditions whatever login method you choose.

Using SNC and Single Sign-On

In each destination section a parameter ConnectParam can be defined. This parameter holds additional connection parameters that are appended to the connection string used to connect to SAP via RFC. Therefore, ConnectParam can be used to log on to SAP using SNC. In this case, ConnectParam can look like this:

```
SNC_MODE=1 SNC_QOP=9 SNC_MYNAME="p:username@DOMAIN"  
SNC_PARTNERNAME="p:sap/sapserver@DOMAIN"  
SNC_LIB="C:\WINNT\system32\sapcrypto.dll"
```

```
SNC_MODE=1 SNC_QOP=3 SNC_MYNAME="p:CN=TEST, OU=SAP, O=COMPANY, C=US"  
SNC_PARTNERNAME="p:CN=ABC, OU=SAP, O=COMPANY, C=US"  
SNC_LIB="C:\WINNT\system32\sapcrypto.dll"
```

Please refer to the "SNC User's Guide" provided by SAP for detailed information on SNC parameters and values.

Inside ConnectParam a special substitution is provided by the ENOVIA SmarTeam/SAP Adaptor. If ConnectParam contains the string #USER#, this string is substituted by the final value of connection parameter "user" (the value evaluated after all modifications described in the [previous section](#) have been applied). For instance, if ConnectParam is defined as

```
SNC_MODE=1 SNC_QOP=9 SNC_MYNAME="p:#USER#@DOMAIN"  
SNC_PARTNERNAME="p:sap/sapserver@DOMAIN"  
SNC_LIB="C:\WINNT\system32\sapcrypto.dll"
```

and if "user" evaluates to "joe" after all modifications, ConnectParam becomes



```
SNC_MODE=1 SNC_QOP=9 SNC_MYNAME="p:joe@DOMAIN"  
SNC_PARTNERNAME="p:sap/sapserver@DOMAIN"  
SNC_LIB="C:\WINNT\system32\sapcrypto.dll"
```

This is then the actual value of ConnectParam appended to the connection string.







Licensing

LUM Server: The server should have LUM 4.6.8.3 or above installed.








Register DLL

-  Extract the LUM dll (ISPLI0BPALUM.dll) from the package / zip file.
Place it in the same directory as smSAPif.exe.
-  Register the same using regsvr32 command.
regsvr32 "<path to smSAPif.exe>\ISPLI0BPALUM.dll"

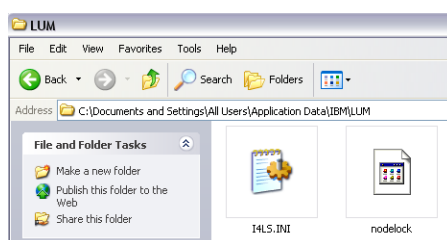
Request a license with following information:

-  Product Name: SS9.
-  License Type: nodelock or concurrent
-  Number of License: 1 for nodelock , >0 for Concurrent
-  LUM targetID: Provide the target id (run i4target)
 -  On Client side for Nodelock
 -  On Server side for Concurrent

For Nodelock type of license

-  Because we are not integrated into the NodelockKeyManagement
-  You should input yourself the parameters of your License into the nodelock file
(create it if doesn't exist)
-  This file is located in :
 -  For windows: C:\Documents and Settings\All Users\Application Data\IBM\LUM\nodelock
-  You should extract from the .lic file the following informations:
 -  VendorID (looks like an UUID)
 -  ProductPassword (a crypted key)

- 📌 ProductAnnotation (should be unset)
 - 📌 ProductVersion (should be set to 1)
 - 👤 And insert them in nodelock file by respecting the order
 - 📌 # comment : put your BPA trigram and its range date
 - 📌 vendorID ProductPassword ProductAnnotation ProductVersion
 - 📌 Example :
- # SS9 from 15-Jul-08 to 14-Jul
- c6c8ef44bcb7.4a.74.95.13.1f.00.00.00 r8ezikjzgsvk9fzf7fe3p2gn3eaa "" "1"



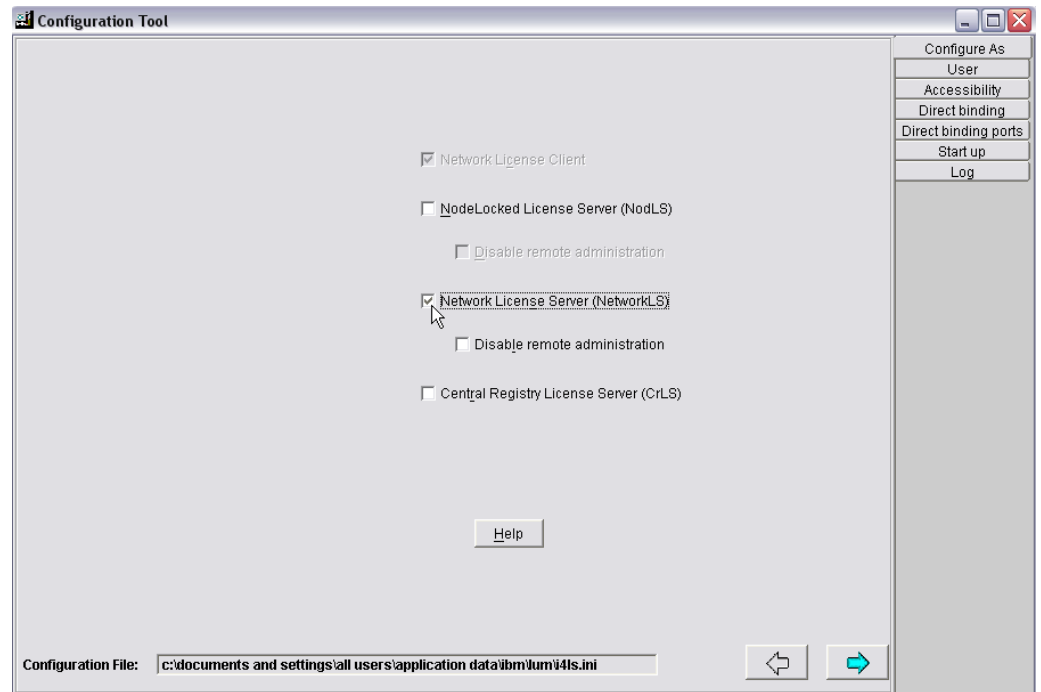
For Concurrent type of license

- 👤 Enroll the license on the LUM server
 - 📌 **Start > All Programs > License Use Runtime > Configuration tool**
(Stop the service if needed using the Service Manager Tool)

Service Manager Tool			
Service	Selected	Options	Help
Name	Status	Description	
I4LLMD.EXE	Started	Nodelock License Server	
I4LMD.EXE	Started	Network License Server	

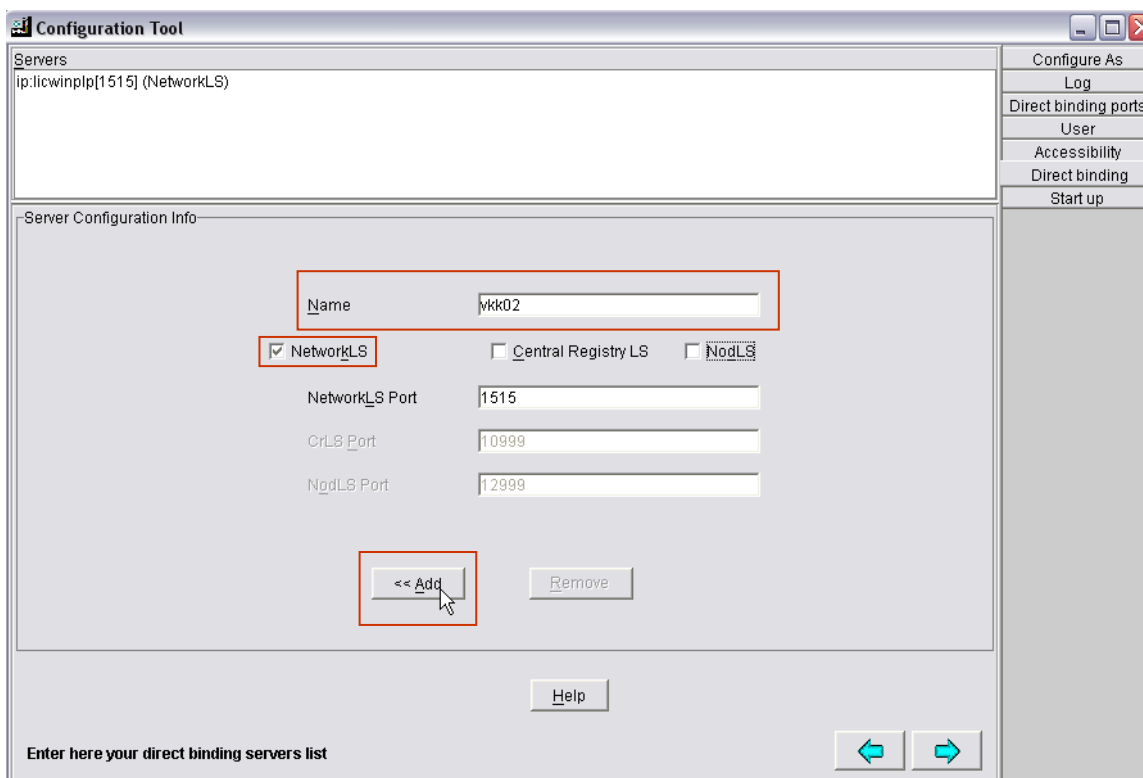
Service Manager Tool			
Service	Selected	Options	Help
Name	Status		
I4LLMD.EXE	Stopped		
I4LMD.EXE	Stopped		

- Ensure that the option for Nodelocked License Server is selected

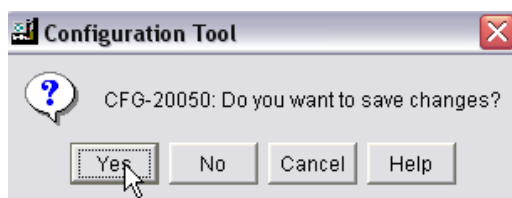


- Switch to the Direct Biding tab.
- In the Name field enter the machine name

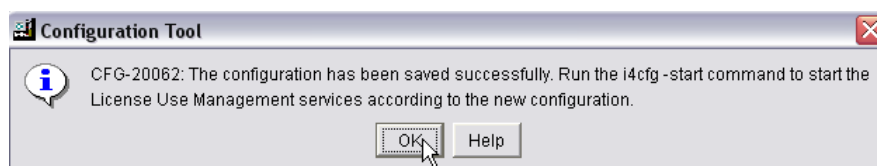
🔧 Check the box for NetworkLS and click on Add button



🔧 Close the Configuration Tool window and validate accepting the changes



Pass the message to restart the service (if previously it was stopped)



Service	Selected	Options	Help
i4LLMD.EXE	Started		NodeLock License Server
i4LMD.EXE	Started		Network License Server

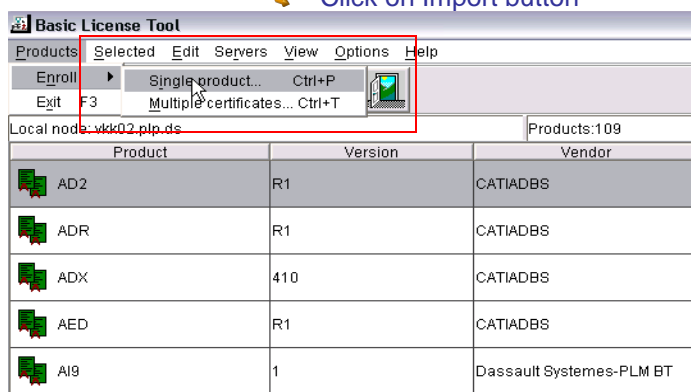
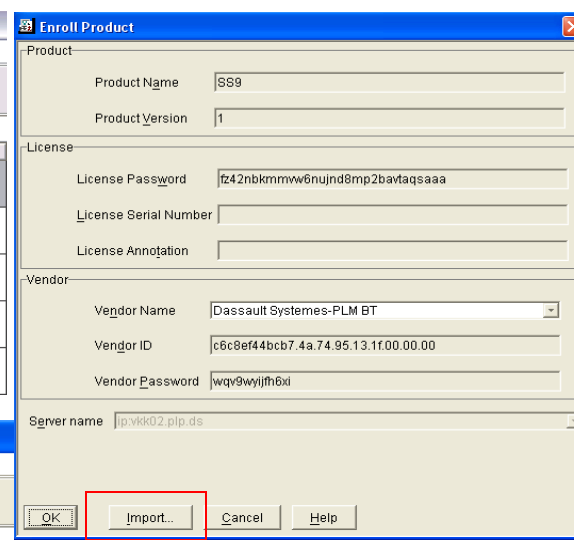
Enrolling the Concurrent License (2/2)

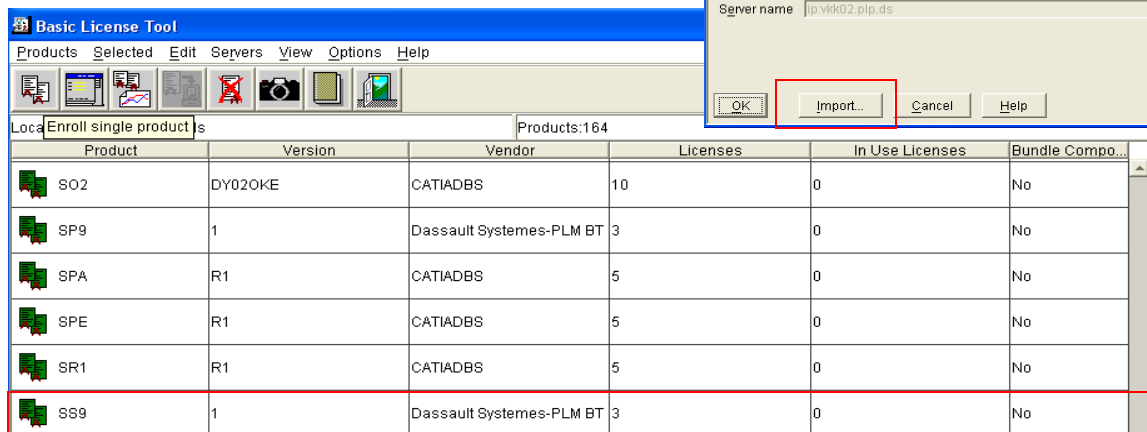
Start > All Programs> License Use Runtime > Basic License Tool

Select Product / Enroll/

Choose Single or Multiple depending on whether you want to be enrolling a single or multiple licenses at a time.

Click on Import button



Product	Version	Vendor	Licenses	In Use Licenses	Bundle Compo...
SO2	DY020KE	CATIADBS	10	0	No
SP9	1	Dassault Systemes-PLM BT	3	0	No
SPA	R1	CATIADBS	5	0	No
SPE	R1	CATIADBS	5	0	No
SR1	R1	CATIADBS	5	0	No
SS9	1	Dassault Systemes-PLM BT	3	0	No

The Activity Sections

The configuration file contains one or more activity sections comprising the details of an action sequence to be executed. Typical activity sections look like this:

```
[MaterialSelect]
New                      Material, M, Smarteam, S
M.SelectViaMatchcode
Check                   #SAP.DialogStatus# <> A
S.SetData               TDM_SAP_MAT_NUM, #M.Material#
S.Save

[DocumentSave]
New                      Document, D, Smarteam, S
S.GetData
Define                  %DOCNUM%, \Delete.{#S.Object.CN_ID#}
Define                  %VERSION%, \LeftDotLeft2{#S.Object.REVISION#}
D.AddBasicData          DRW, %DOCNUM%, 000, %VERSION%, _
                        #S.Object.CN_DESCRIPTION#, #SAP.User#, _
                        \Left2{#S.Object.CN_SAP_STATUS#}

D.ExistenceCheck
If                        #D.Exists# <> X
  D.AddClassallocation   017, V_CAD
  D.AddCharacteristicvalue 017, V_CAD, V_PARTNUMBER, _
                        #S.Object.CN_PART_NUMBER#
  D.AddCharacteristicvalue 017, V_CAD, V_PARTDESCRIPTION, _
                        #S.Object.CN_PART_DESCRIPTION#
  D.Create
Else
  Message Error, Document DRW/%DOCNUM%/000/%VERSION% already exists!
Endif
DisplayBapiMessages
```

An activity section consists of [activity functions](#), constant data, [variables and definitions](#) (either ENOVIA SmarTeam or SAP or other) and [inline functions](#).

Activity functions are the elements of which the "program code" of an activity section is built. They control the actions performed by the ENOVIA SmarTeam/SAP Adaptor. Activity functions handling batch inputs, classifications, documents, engineering change masters, files, function modules, materials, and ENOVIA SmarTeam objects are in the tool set, and functions controlling the program flow and performing miscellaneous operations are available too. The most important activity function is "[New](#)" because all objects treated by the ENOVIA SmarTeam/SAP Adaptor have to be declared and created using function "New" before they can be used.

Variables refer to SAP objects, ENOVIA SmarTeam attributes, or other data fields. Definitions are self defined abbreviations for any character strings.

Inline functions modify the data they work on. They can return parts of the argument string, delete characters in the string, extract path, filename and extension of a given full qualified filename etc.

To reach an understanding of how the ENOVIA SmarTeam/SAP Adaptor works it is recommended to read through the [sample configurations](#) and to compare each function call and each expression using an inline function with the corresponding function description. The only thing that has to be known before is how a function call is evaluated and how the activity functions work in principle.

Question one: How is the program code interpreted?

- 👤 If a line ends with an underscore "_" (line continuation character), this line and the following line are joined together eliminating the underscore.
- 👤 After that all [definitions](#) (created by [Define](#) or [Set](#)) are replaced with the respective values. If a definition name is not yet defined, it is not replaced by anything.
- 👤 Then all [variables](#) #___# are replaced with their values.
- 👤 In the next step all [inline functions](#) are evaluated, transforming their string arguments into other strings.
- 👤 At last the [activity function](#) is called with the resulting arguments.

Question two: How do the activity functions work in principle?

In principle the ENOVIA SmarTeam/SAP Adaptor works on an internal memory which can be filled in several ways:

- 👤 by calling activity functions like Document.AddBasicData, Document.AddClassAllocation, or Material.AddBIDData (most of these functions are identified by the word Add)
- 👤 by calling Smarteam functions like Smarteam.GetData, Smarteam.GetStructure or Smarteam.GetLinks
- 👤 or by calling SAP functions like Document.GetDetail, Document.GetObjectLinks, Material.ReadBOM, or Classification.GetAllocations.

Then the internal memory is used to update ENOVIA SmarTeam or SAP data using functions like

- 👤 Smarteam.SetData together with Smarteam.Save or
- 👤 Document.Save, Material.SaveBOM, Classification.SaveValidation etc.

Sample Script

A sample script “**smSAPif.bs**” that can be used to link the ENOVIA SmarTeam/SAP Adaptor to a Smarteam hook is in the Adpator Folder.



smSAPif.bs

External Calls

[Calling the ENOVIA SmarTeam/SAP Adaptor from other applications](#)

[Calling the ENOVIA SmarTeam/SAP Adaptor from the command line](#)

[Encrypting and checking passwords from external applications](#)

Calling the ENOVIA SmarTeam/SAP Adaptor from other applications

Calling the ENOVIA SmarTeam/SAP Adaptor from any other application is very similar to calling it from a [SMARTTEAM script](#).

Using Visual Basic the code for calling the smSAPif functionality will look like this:

```
Dim RC           As Long
Dim Destination As String
Dim Operation    As String
Dim ArgString    As String
Dim SAP          As smSAPif.SapLink
Dim DialogMode   As smSAPif.eDialogMode
Dim LinkMode     As smSAPif.eLinkMode
Dim ProgressInfo As smSAPif.eProgressInfo

On Error Resume Next
Set SAP = New smSAPif.SapLink
If SAP Is Nothing Then
    MsgBox "SAP Integration not installed.", _
        vbCritical Or vbOKOnly, "Error"
    Exit
End If
' Set all arguments to appropriate values and execute SAP call
RC = SAP.Execute(SmSession, SmObject, Destination, Operation, _
    DialogMode, LinkMode, ProgressInfo, ArgString)
' Evaluate RC and ArgString which are both returned
```

Accordingly the code for calling the smSAPifq functionality will look like this:

```
Dim RC           As Long
Dim Destination As String
Dim Operation    As String
Dim ArgString    As String
Dim SAP          As smSAPifq.SapLink
Dim DialogMode   As smSAPifq.eDialogMode
Dim LinkMode     As smSAPifq.eLinkMode
```

```
Dim ProgressInfo As smSAPifq.eProgressInfo

On Error Resume Next
Set SAP = New smSAPifq.SapLink
If SAP Is Nothing Then
    MsgBox "SAP Integration not installed.", _
        vbCritical Or vbOKOnly, "Error"
    Exit
End If
' Set all arguments to appropriate values and execute SAP call
RC = SAP.Execute(SmSession, SmObject, Destination, Operation, _
    DialogMode, LinkMode, ProgressInfo, ArgString)
' Evaluate RC and ArgString which are both returned
```

The difference between smSAPif and smSAPifq is that

- 👤 smSAPif does not queue a request, i. e. each request is executed immediately in a new process using a new R/3 RFC connection, whereas
- 👤 smSAPifq does queue requests which means that requests are stored and processed one after another using the same R/3 RFC connection if the connection parameters Destination and DialogMode do not change. (If they change, a new connection is established but the requests remain queued.)

For an explanation and possible values of parameters DialogMode, LinkMode and ProgressInfo see the definitions in the [sample script](#).

Additional information can be passed to the ENOVIA SmarTeam/SAP Adaptor in ArgString which is stored as [definition](#) and which is accessible through the name %ARGSTRING%. One issue of course is that ArgString and therefore %ARGSTRING% is only a single string. If multiple arguments have to be passed to the ENOVIA SmarTeam/SAP Adaptor see the corresponding [sample configuration](#).

Information can also be passed back to the calling application using the [SetExitCode](#) function. The syntax of this function is

```
SetExitCode <ExitCode>, <ResultString>
```

which means that a statement like this

```
SetExitCode 0, Last used object has object ID #SMARTEAM.OBJECT_ID#
```

returns a function result of zero and, for instance, "Last used object has object ID 4711" on ArgString to the calling application.

Function [SetExitCode](#) can also be used to pass error information back to the calling application if a fatal error occurs in the ENOVIA SmarTeam/SAP Adaptor. In that case of a non-recoverable error the ENOVIA SmarTeam/SAP Adaptor executes the error handler which was set in the current action sequence by means of function [SetErrorHandler](#). If no such error handler is set - because, for instance, the fatal error occurred before the action sequence was read - the ENOVIA SmarTeam/SAP Adaptor looks for key word ErrorHandler in [section General](#) of the configuration file. If this key word cannot be found, the error message is displayed in a popup window which is,

therefore, the default if no error handler is defined at all. See section [Error Handling](#) for more information.

Calling the ENOVIA SmarTeam/SAP Adaptor from the command line

A command line option for the ENOVIA SmarTeam/SAP Adaptor is available too. It is called smSAPifc and the syntax of the call is

```
smSAPifc Destination=<Destination>
        Operation=<Operation>
        [User=<Username>
        Password=<Password>
        Password1=<Encrypted password>
        Language=<Language>
        DialogMode=<DialogMode>
        ProgressInfo=<ProgressInfo>
        Key1=Value1
        ...
        Keyx=Valuex
        -1
        -P <clear password>
        -V]
```

Parameters Destination and Operation are needed for executing an operation. All other parameters are optional. But if specified, the parameter values overwrite the corresponding values of the [Destination section](#) of the configuration file. If a parameter -1 is passed as argument, the application terminates if another instance of the application is already running. With the -P <clear password> option you can specify a clear password. ENOVIA SmarTeam/SAP Adaptor will return a message box with this password encrypted. The -V option finally displays version information.

For an explanation of parameter meanings and allowed values see the documentation for the [Destination section](#) of the configuration file. For an explanation and possible values of parameters DialogMode and ProgressInfo see the definitions in the [sample script](#).

All parameters not equal to Destination, Operation, User, Password, Password1, Language, DialogMode, ProgressInfo, -1, -P and -V are being considered as [definitions](#) which means that Key1=Value1 yields the same result as an explicit "Define Key1, Value1" statement in the operation section.

Encrypting and checking passwords from external applications

If you need to encrypt a password from your own code, for instance on a Before-Update hook of a user's profile card, the ENOVIA SmarTeam/SAP Adaptor also provides functions that can be

called from external applications to encrypt a clear password and to check an encrypted password against a clear password. See the following VB sample code how it works.

```
Dim ClearPW      As String
Dim CryptedPW    As String
Dim CheckResult  As Boolean
Dim SAP          As smSAPif.SapLink

Set SAP = New smSAPif.SapLink
If SAP Is Nothing Then
    MsgBox "SAP Integration not installed.", _
        vbCritical Or vbOKOnly, "Error"
    Exit
End If
ClearPW = ...
CryptedPW = SAP.PWCrypt(ClearPW)
CheckResult = SAP.PWCheck(ClearPW, CryptedPW)
```

Error Handling

In principal, the ENOVIA SmarTeam/SAP Adaptor is an online integration between ENOVIA SmarTeam and SAP. That means, a request is initiated by a user activity like creating a document, updating it or releasing it, and the request then being executed without delay. Errors that might occurs during execution are directly passed to the user so that the user can take appropriate means to react.

From the system point of view there are two types of errors:

- 👤 Error issued by SAP BAPIs or RFC function modules
- 👤 Error raised by the ENOVIA SmarTeam/SAP Adaptor itself.

The first type of errors are automatically processed by the ENOVIA SmarTeam/SAP Adaptor itself. That means, error messages and error types issued by SAP function modules are internally stored. They can be accessed through certain variables so that the designer of an action sequence can react properly on SAP errors, for instance like this:

```
If #SAP.BapiStatusSummary# = A, #SAP.BapiStatusSummary# = E
    DisplayBapiMessages
    Exit 1, #SAP.BapiErrorMessages#
Endif
```

If such error handling is omitted, no message will be shown to the user and the ENOVIA SmarTeam/SAP Adaptor will continue with the next statement of the actions sequence.

The second type of error occurs for example if there is a syntax error in an activity section of the configuration file, if the connection to the desired SAP system cannot be established, if the connection breaks during execution or if there is a fatal ENOVIA SmarTeam error. This (fatal) errors are processed automatically only to a certain degree. What does this means exactly?

Errors of the second kind raise a program exception which would usually result in a program execution error. The ENOVIA SmarTeam/SAP Adaptor catches such execution errors preventing the application from crashing. By default, the ENOVIA SmarTeam/SAP Adaptor displays the error message in a popup window and exist.

Such error messages are obviously not desired if the ENOVIA SmarTeam/SAP Adaptor runs as a stand-alone application which is not user-driven but, for instance, launched by the operation system to update ENOVIA SmarTeam objects from SAP material masters every night. In this case the ENOVIA SmarTeam/SAP Adaptor cannot show any error message and wait for a user interaction. The ENOVIA SmarTeam/SAP Adaptor should handle exceptions by itself but it should handle it 'dark' without needing the user. How can such a behavior be configured? The solution is to set an error handler using function [SetErrorHandler](#). This is a section of the configuration file that will be executed if a (fatal) error occurs. Thus, the configuration file can look like this:

```
[SectionToBeExecuted]
SetErrorHandler OnError
...
```

```
[OnError]
New          File, F
F.OpenForAppend C:\SAP_ST_Log\|#System.Date#.log
F.PutLine    >>>> ERROR >>>> Time: #System.Time#
F.PutLine    #System.ErrorDescription#
F.Close
```

Using this configuration, no message will be displayed but a log file will be written that carries the time the error occurred together with the error message. Another way to process the error is a call to function [SendMail](#) to actively notify the administrator about the error.

However, there are fatal non-recoverable error that can occur before the action sequence could be read. This can, for instance, happen when the name of the action sequence to be executed is miss-spelled and can therefore not be found. In that case, an error handler could obviously not have been set, but even in that case it is possible to execute an error handler. Just add a global error handler in [section General](#) by means of key words ErrorHandler or ErrorHandler_<SectionName>. If any of these key words is found in section General, the specified error handler is executed in case of a fatal error if no error handler was set in the current action sequence. The appropriate configuration file could look like this:

```
[General]
ErrorHandler = OnError
ErrorHandler_SectionA = OnErrorA
...

[SectionA]
...

[SectionB]
SetErrorHandler OnErrorB
...

[OnError]
# This is the global error handler which is executed if no specific
error handler is set
SetExitCode    99, Error Number: #System.ErrorNumber#\nError Text:
#System.ErrorDescription#

[OnErrorA]
# This error handler will be executed on fatal errors in SectionA
Message      A, Error Number: #System.ErrorNumber#\nError Text:
#System.ErrorDescription#

[OnErrorB]
# This error handler will be executed on fatal errors in SectionB
SendMail      Error in SAP Adaptor, #System.ErrorDescription#, ...
```

Using error handlers is also an appropriate means for passing error information back to a calling application. Taking error handler OnError of the example above, the external application receives a return code 99 from function Execute and on argument ArgString the second parameter following the 99 (the error message itself) is returned. **Important notice:** Please make sure that function Execute is called modally using value ImModal for parameter LinkMode if you want to pass error information back to a calling application. Only in that case the error information is returned to the calling application because only in this case function Execute waits for the action sequence to be completed.

General Section

The configuration file smSAPif.ini can contain a section called [General] which contains certain settings that need to be set under certain circumstances only.

```
[General]
System = ON/OFF
Debug = ON/OFF
DecimalSignWIN = ,
ThousandSignWIN = .
DateFormatWIN = YYYY-MM-DD
TraceDir = S:\smSAPif\Traces
ErrorHandler = OnErrorGlobal
ErrorHandler_<SectionName> = OnErrorSectionSpecific
FunctionModule_<Name> = Z_MY_ALTERNATIVE_BAPI
ConvertMaterialNumbers = ON/OFF
UseLongMaterialNumbers = ON/OFF
KeepOldBCDFormat = ON/OFF
BOMHeaderStructure = BICSK
BOMItemStructure = CSRFCITEM
BOMEvalWarning = ON/OFF
MessageLevelReferenceNotFound =
None/Information/Warning/Error/Exception
MessageLevelReferenceNotUnique =
None/Information/Warning/Error/Exception
```

The following table describes the keys and possible values.

Key	Value	Meaning
System	ON (default)	RFC calls to the SAP system are actually performed by the ENOVIA SmarTeam/SAP Adaptor.
	OFF	The ENOVIA SmarTeam/SAP Adaptor behaves normally except the fact that no RFC calls to the SAP system are actually performed. That means that certain checks like syntax checks or calls from external applications can be performed without SAP system. Semantic checks must be avoided due to the fact that all SAP fields will keep initial values.
Debug	ON	This is for development only. All error handler will be suspended.
	OFF (default)	Always use this setting in customer environments.
DecimalSignWIN	Windows Locale setting	This setting overwrites the Windows default for the decimal sign.
ThousandSignWIN	Windows Locale setting	This setting overwrites the Windows default for the thousand sign.
DateFormatWIN	Windows Locale setting	This setting overwrites the Windows default for the date format.

TraceDir	Valid directory name to existing path	If set, the ENOVIA SmarTeam/SAP Adaptor will write trace files into that directory. For more details look here .
ErrorHandler	Section contained in smSAPif.ini file	This key specifies the global error handler. For more details look here .
ErrorHandler_<SectionName>	Section contained in smSAPif.ini file	This key specifies the local error handler for section <SectionName>. For more details look here .
FunctionModule_<Name>	Name of the alternative function module. Default: <Name>	This key defines an alternative function module to be called instead of function module <Name>. If, for instance, function Z_BAPI_DOCUMENT_CREATE2 shall be used instead of SAP standard BAPI_DOCUMENT_CREATE2, use: FunctionModule_BAPI_DOCUMENT_CREATE2 = Z_BAPI_DOCUMENT_CREATE2 Remark 1: The alternative function module must have the same interface than the original one. Remark 2: Currently not all SAP standard function modules can be substituted by this mechanism.
ConvertMaterialNumbers	ON (default)	Usually, numerical material numbers are stored in SAP with leading zeros, 000000000000123456 for instance. Therefore, the ENOVIA SmarTeam/SAP Adaptor performs all needed internal/external conversions automatically.
	OFF	If numerical material number are stored lexicographically in the desired SAP systems, automatic internal/external conversion can be set to OFF for material numbers using this setting.
UseLongMaterialNumbers	ON	Up to some level of SAP, material numbers were 18 characters long. On newer SAP systems, the length of the material number was extended to 40. This setting forces the ENOVIA SmarTeam/SAP Adaptor to use material numbers of length 40.
	OFF (default)	The ENOVIA SmarTeam/SAP Adaptor uses 18 characters for material numbers.
KeepOldBCDFormat	ON	Usually, a positive Binary Coded Decimal (BCD) number carries a "+" sign in front. 12.34, for instance, is presented as "+12.34" and 0.1234 is presented as "+.1234". If you want to stay with this external format for BCD numbers, switch this setting on.
	OFF (default)	Switching this setting off (which is the default) turns the external representation of BCD numbers into a more common one. 12.34, for instance, is presented without leading plus sign as "12.34" and 0.1234 is presented without leading plus sign but with a leading zero as "0.1234".
BOMHeaderStructure	Name of SAP structure of parameter I_BOM_HEADER of function module CAD_CHANGE_BOM_WITH_SUB_ITEMS	Usually this setting does not need to be set because the actual SAP structure is compatible to default structure BICSK. The settings needs to be set only if the actual structure contains fields that need to be filled and that are not contained in structure BICSK.

BOMItemStructure	Name of SAP structure of parameter BOM_ITEM of function module CAD_CHANGE_BOM_WITH_SUB_ITEMS	Usually this setting does not need to be set because the actual SAP structure is compatible to default structure CSRFCITEM. The settings needs to be set only if the actual structure contains fields that need to be filled and that are not contained in structure CSRFCITEM.
BOMEvalWarning	ON	Function modules CSAP_MAT_BOM_... and CSAP_BOM_ITEM_MAINTAIN export a parameter FL_WARNING. If this parameter is set to X by the function module, the SAP Adaptor fetches and processes the message from SAP.
	OFF (default)	If FL_WARNING is X, messages from SAP are not processed. (Exceptions of the function modules are processed automatically in any case.)
MessageLevelReferenceNotFound	None	Consider a Smarteam.SetData statement like this Smarteam.SetData CN_REF_ITEM.CN_MATERIAL.DESCRPTION, Steel using "dot" technique. Setting "MessageLevelReferenceNotFound" control the error handling if the requested value (here: Steel) cannot be found. If it is set to "None", no message will be issued.
	Information Warning Error	If this setting is set to "Information", "Warning" or "Error", a message box with appropriate message and message type will be displayed to the user.
	Exception	If this setting is set to "Exception", a runtime error will be raised allowing an error handler to treat the message in a desired way.
MessageLevelReferenceNotUnique	None Information Warning Error Exception	Similar setting as for "MessageLevelReferenceNotFound" except that this setting controls the situation where the requested value is not unique.

Problem Solving

In the following you find hints for solving problems that could show up in and around an SAP Adaptor environment. In any case, if you discover problems or if SAP Adaptor shows unexpected behavior, follow the instructions for [generating trace](#). Many problems can be fixed by just looking into the trace files. If you discover a problem without obvious solution, see sections "[Problems related to SAP Messages](#)", "[Problems related to SAP Exceptions](#)", "[Problems related to SAP Adaptor Messages](#)" or "[Other issues](#)". If you have problems storing files into SAP vault or storage categories, go to the [special section dealing with this problem](#).

Generating trace

If the ENOVIA SmarTeam/SAP Adaptor causes errors or if it shows unexpected behavior, trace files should be created, zipped and sent to your hotline. For creating trace files, set the Trace parameter of the destination section to "1" or "2". (Option "3" is only needed if you need to generate SAP trace files for inspection by SAP hotline team.) In all cases two trace files are created. They are named according to the calling application but with extensions ".tra" and ".trc". Exe file smSAPif.exe thus creates traces files "smSAPif.tra" and "smSAPif.trc", exe file smSAPifq.exe creates traces files "smSAPifq.tra" and "smSAPifq.trc", and exe file smSAPifc.exe creates traces files "smSAPifc.tra" and "smSAPifc.trc". Trace files are written to a directory that is found in accordance with the following rules:

- 👤 If a section [\[General\]](#) in the configuration file contains an entry like "TraceDir = <Directory>", <Directory> is taken as trace directory. <Directory> can contain all [system variables available in activity section](#), for instance
C:\Temp\#System.Env.UserName#\#System.Date#.
- 👤 If there is no such section or if the section does not contain such an entry, the ENOVIA SmarTeam/SAP Adaptor looks if directory C:\Temp does exist.
- 👤 If C:\Temp does not exist, it checks if an environment variable TEMP is set and if it points to an existing directory.
- 👤 If environment variable TEMP is not set or if that directory does not exist, the Adaptor takes Window's temporary directory which does exist in any case.
- 👤 In case of an error or unexpected behavior run the following procedure:
- 👤 Locate your trace directory as explained above.

- 👤 Delete files smSAPif.tra and smSAPif.trc (or smSAPifq.tra and smSAPifq.trc or smSAPifc.tra and smSAPifc.trc) in that directory.
- 👤 Repeat the activity that caused the error or the unexpected behavior.
- 👤 Find trace files smSAPif.tra and smSAPif.trc (or smSAPifq.tra and smSAPifq.trc or smSAPifc.tra and smSAPifc.trc) in the trace directory.
- 👤 First look at the "tra" file. It shows lines read from the configuration file smSAPif.ini before and after substitution of definitions, variables and inline functions. If your problem is due to unexpected variable values, misunderstanding of how inline functions work or just misspelling of variables, you should be aware of it right now.
- 👤 To check in more detail whether this behavior could be a bug in SAP Adaptor, look at the proper position in the "trc" file where nearly every step performed by SAP Adaptor is traced.
- 👤 In the "trc" files you also see which data is actually exchanged with SAP (look around the "call function module" sections). Trace option "1" gives a standard ("string") trace. Trace option "2" in addition prints all values passed to SAP using hex encoding. And trace option "3" is like option "2" but it makes SAP interface library librfc32.dll creating it's own trace files which are needed for inspection by SAP. (These files are named "dev_rfc.trc" and "rfcnnnnn_mmmmm.trc" where nnnnn and mmmmm are process and thread numbers that change from call to call).
- 👤 If you are familiar with ABAP programming, you can create an ABAP test program passing exactly the same values to the SAP function module as SAP Adaptor does.
- 👤 If you can reproduce the problem in ABAP test program, open OSS message for inspection by SAP or search SAP help portal if the problem is already known and fixed.
- 👤 If you cannot find the problem on your own, zip the two (or four or six) files together with configuration file smSAPif.ini, exact problem description and screen shots and/or videos and mail them to your hotline contact.

Problems related to SAP Messages

Improper characters in SAP messages

You get a message like this:

Funkcija Material.Save

» E(M3)490: Enota mere ***** ni predvidena; Preverite va# vnos «

but correct message should be:

Funkcija Material.Save

» E(M3)490: Enota mere ***** ni predvidena; Preverite vaš vnos «

(Slovanien special character š is substituted by #.)

Usually the client side code page is determined automatically by SAP's RFC library. In special cases like this one that characters are displayed improperly it can be required to use a different code page. In order to do so, go to proper destination section of the configuration file and add a line like this:

ConnectParam = CODEPAGE=1404

Please have a look at SAP documentation for all relevant code pages. Without guarantee and without implying completeness these code pages must be used: West European 1100, East European 1401, Slovenian 1404, Russian 1500, Turkish 1610, Greek 1700, Hebrew 1800, Baltic 1900, Japanese 8000, Taiwanese 8300, Chinese 8400, Korean 8500, Thai 8600, Arabic 8700.

Improper SAP values

You get a message like this:

Function Material.Save

» E(MG)142 Value XXX for field ZZZ not allowed «

You are trying to transfer a value XXX to SAP field ZZZ but XXX is not configured as a member of the value list of field ZZZ. This can have several reasons.

- First of all ask your SAP administrator for the possible values of attribute ZZZ.
- If the value comes from a Lookup attribute in Smarteam, the values can simply be missing in the list of values configured for this lookup.
- If the value comes from a Lookup attribute, check if the value to be sent to SAP comes from the Description or from the Name. It might be that you must configure #S.CN_ABC.TDM_NAME# instead of #S.CN_ABC.TDM_DESCRIPTION# its simplified equivalent #S.CN_ABC#.
- If this is not the case, check the smSAPif/q/c.tra file. Check which value comes out of Smarteam, which inline function potentially modifies the attribute and which reason makes the value not fitting to the list of allowed SAP values.

Improper SAP value for a unit field when creating material master

You get a message like this:

Function Material.Save

» E(M3)490: Unit of measure ***** is not defined; check your entry «

You are using external (language dependent) representation of a unit field (unit of measure, unit of weight, unit of volume) but in fact you have to use internal key value. For instance, for the unit of measure you cannot use EA (Each) or PC (Piece) but you must use ST (Stück). Go to your configuration file and update the respective line, for instance

Material.AddClientData BaseUom:=ST

Improper SAP value for a unit field when creating BOM

You get a message like this:

Function Material.SaveBOM2

» E(XX)99: Unit ST is not created in language EN «

You are using internal (language independent) representation for the unit of measure but in fact you have to use external value when creating or updating BOM. Substitutue internal value ST by the proper external value used in your system (PC, EA, ...)

Missing values when creating BOM

You get a message like this:

Function Material.SaveBOM2

» E(XX)99: Fill in all required entry fields «

Most of the time the statement "Material.AddBOMHeaderData2" is missing so that the BOM header passed to the function module is actually empty. Please look at the configuration file if this statement is included and at the trace files if the BOM header is really filled. If this is okay, take this ABAP test code "z_test_cad_bom_sub_items.txt" and continue testing inside SAP based on the ABAP code.

System complains about changing the change number when creating BOM

You get a message like this:

Function Material.SaveBOM2

» E(XX)99: The change number cannot be changed «

This message does not mean that you try to change the change number in SAP. It says that you modified the change number in BOM header before and after function Material.Read BOM2. Please refer to the [sample configuration on how to save a BOM](#) in order to get more information.

File transfer into SAP vault or storage category

You get a message like this:

Function Document.Save

» E(26)253: Error while storing file C:\Documents and Settings\JimBob\My Documents\Smarm «

Please consult section [Error when storing files in SAP vaults or storage categories](#)

File transfer into SAP vault or storage category

You get a message like this:

Function Document.Save

» E(26)136: File could not be accessed «

In German the message is: » E(26)2136: Datei nicht erreichbar «

It seems that this is an access problem but actually the message means that you did not specify parameter <OriginalType> of function [Document.AddFile](#) when working with non-Kpro document types. When using non-Kpro document types, you can only specify two original files for one document info record whereas for Kpro document types the number of originals is practically unlimited. When working with Kpro document types you can therefore append a new original file to the document info record by leaving parameter <OriginalType> empty whereas with non-Kpro document types you always have to specify if you are dealing with original 1 or 2. To solve this problem try setting <OriginalType> to either 1 or 2 when calling [Document.AddFile](#). If the problem is still not solved, please consult section [Error when storing files in SAP vaults or storage categories](#) for more help.

File transfer into SAP vault or storage category

You get a message like this:

Function Document.Save

» E()001: Win32 error 2: The system cannot find the file spe «

Usually this error message means that programs sapftp.exe or saphttp.exe are not found. Make sure that you have copied these files from the SAPGUI installation directory into the SAP Adaptor installation directory.

File transfer into SAP vault or storage category

You get a message like this although you have installed an RFC library 4.6, 6.20, 6.40 or above:

Funktion Document.Create

» E()001: I::001 Only available with the RFC library from 4.0C onwa «

The reason for this error is still unknown. It seems that the problem occurs when working with Unicode SAP system and non-Unicode sapftp.exe/saphttp.exe (which are installed with SAPGUI by default). Try using the Unicode versions of these programs according to SAP note 729800. If it still doesn't work, use the following workaround: Go to Sap.bs file and switch the Dialog Mode of the activity section involved from dmNoDialog to dmDialog.

Problems related to SAP Exceptions

Logon error

You get this message:

You are not authorized to logon to the target system (error code 1).

This behavior is typical if you have installed SAPGUI (LibRfc32.dll) 6.40. Try to logon with the password in capital letters.

Activity needs SAPGUI but is executed in non-dialog mode

You get this message:

Screen output without connection to user.

You are executing an activity that needs SAPGUI for screen output but you customized the activity section as non-dialog. Go to Sap.bs file and switch the Dialog mode of the activity section involved from dmNoDialog to dmDialog.

Problems related to SAP Adaptor Messages

Runtime error: File not found

You get a message

Runtime error '53':
File not found

SAP Adaptor cannot find file smSAPif.txt.

File not found: LibRfc32.dll

You get a message

File not found: LibRfc32.dll

The reason for this error message is that you did not install SAPGUI or that you have a bad SAPGUI installation.

- Check that SAPGUI is installed on the computer where SAP Adaptor is running. At least, file librfc32.dll must be located in [C:\Windows\System32](#) directory.
- If SAPGUI is installed and if the file is in that directory, SAPGUI installation might be corrupt. Uninstall SAPGUI and re-install it again.

Script error: Error while loading the ENOVIA SmarTeam/SAP Adaptor

You get a message like this

Error while loading the ENOVIA SmarTeam/SAP Adaptor
Unable to load ENOVIA SmarTeam/SAP Adaptor

SAP Adaptor exe files are not registered on the PC the activity is executed. Double click on smSAPif.exe and smSAPifq.exe to register them.

Other issues

Wrong version of SAP Adaptor used / Wrong configuration file smSAPif.ini used

Looking at the trace files for instance, you discover that you are using a wrong version of SAP Adaptor or that you are using a wrong configuration file smSAPif.ini.

As described in the installation procedure, you have to double click or to execute smSAPif.exe and smSAPifq.exe in order to register them. When using a new installation directory or when switching between different installation directories (for instance when switching between test and production environment), smSAPif.exe and smSAPifq.exe must be registered in the new directory. On some Microsoft operating systems this fails. The exe files are registered correctly the first time but an update of the registration fails: Windows registry still points to the old installation directory although there was no error message. To solve this problem, open the Registry Editor and delete the class keys of the two exe files with all their sub keys (HKEY_CLASSES_ROOT\CLSID\{50DD83F3-0F0A-4102-AC73-9FDC7E3B0C9F} and HKEY_CLASSES_ROOT\CLSID\{B6B43B03-2AC9-4EC0-A780-B809ED60CC7D}). Then register the two files again. Alternatively you can try to delete the registry entries using the unregserver option of the SAP Adaptor exe files:

```
smSAPif.exe /unregserver
smSAPifq.exe /unregserver
```

Similarly you can use the regserver options to register the exe files again:

```
smSAPif.exe /regserver
smSAPifq.exe /regserver
```

Performance problems

You discover performance problems with SAP Adaptor.

First look at the trace files smSAPif/q/c.tra. Inside the trace files with extension tra a time trace for each activity function is given.

- If only SAP activities like Document.Save, Material.Save, Document.GetDetail or Material.GetDetail are slow, consult you SAP administration. He should check the connection from the client where SAP Adaptor is running to the SAP server, he should check that there are enough dialog and working processes running in SAP and he should check the reaction time of the SAP system.
- If even simple SAP Adaptor activities like Document.AddBasicData are slow, check if a virus scanner is running on the PC where SAP Adaptor is executed and if the virus scanner is configured improperly. Since SAP Adaptor trace files are updated very often, an improperly configured virus scanner would scan the trace files every time they are touched. This can have dramatic influence on the performance up to a factor of 10.














Error when storing files in SAP vaults or storage categories

If you work with functions [Document.Create](#), [Document.Create2](#), [Document.Change](#), [Document.Change2](#), [Document.Checkin](#), [Document.Checkin2](#), [Document.CheckinReplace](#) or [Document.CheckinReplace2](#) for storing a file into a SAP vault or storage category, you might receive SAP error messages like the following:

E(26)253: Error while storing file C:\Documents and Settings\JimBob\My Documents\Smarm

(The message is not complete because SAP only supplies messages variables of length 50.)

If you discover an error like this:

-  Look at the trace file for the complete path and filename you sent to the function module.
-  Check that the specified file in the given directory really exists.
-  Check in the trace file that the DocPath variable you specified in function [Document.AddFile2](#) is terminated by an "\". (**This is the point that goes wrong most times!** SAP never puts a "\" at the end of the path by itself.)
-  If you work with SAP vaults, start transaction SPRO and check in the customization of your vault that the path you specified also end with a "\".
-  If you work with SAP vaults, check that program sapftp.exe is in the smSAPif installation directory.
-  If you work with SAP storage categories, check that program saphttp.exe is in the smSAPif installation directory.
-  If everything is fine so far, start transaction SM59 and open "TCP/IP connections".
-  If you work with SAP vaults, open RFC destination SAPFTP and select button "Test connection". If you get a timeout, inform your SAP administrator (this is an SAP problem).
-  If this check is okay, start transaction SE38 and run program RSFTP005. If you get an error, inform your SAP administrator (this is an SAP problem).
-  If you work with SAP storage categories, open RFC destination SAPHTTP and select button "Test connection". If you get a timeout, inform your SAP administrator (this is an SAP problem).
-  If this check is okay, start transaction SE38 and run program RSHTTP05. If you get an error, inform your SAP administrator (this is an SAP problem).
-  If all these checks are okay, inform your SAP administrator that you cannot find the error by yourself.
-  If you called the activity section containing the file check-in in non-dialog mode (DialogMode = dmNoDialog), you can try a workaround: switch the activity section to dialog mode (DialogMode = dmDialog) and re-run it. (Since there is no real dialog in the activity section, SAPGUI is opened invisible but nevertheless allowing the check-in operation to run through SAPGUI.)

- 👤 If this works, you are lucky but your SAP administrator should keep on finding out why the non-dialog check-in doesn't work.
- 👤 If this doesn't work, you have a really serious problem. Write an ABAP program (like "z_test_bapi_doc_create2.txt" in the SAP Adaptor) that exactly does what the ENOVIA SmarTeam/SAP Adaptor tries to do. If the ABAP program also fails, open an OSS message and let the experts solve the problem. If the ABAP program executes successfully without an error message, open an OSS message too.

For your analysis, please take also into account the following SAP notes:

- 👤 Note number 93042 - Problems with SAPFTP
- 👤 Note number 164203 - Problems with SAPHTTP
- 👤 Note number 900076 - Procedure in RFC problem case
- 👤 Note number 729800 - Checkin/checkout with document BAPIs in Unicode systems

Available Activity Functions

Navigator

Batch Input Functions

[AddRecord](#) [AddSetGetParameters](#) [CallTransaction](#) [Clear](#)

Class Functions

[AddData](#) [AddSelectionCriterion](#) [Clear](#) [GetCharacteristics](#) [SelectObjects](#)
[SelectObjectsDialog](#)

Classification Functions

[AddData](#) [AddIdentifier](#) [AddValidation](#) [Clear](#) [GetAllocations](#) [GetValidations](#)
[SaveAllocation](#) [SaveValidation](#)

Document Functions

[AddBasicData](#) [AddBasicData2](#) [AddCharacteristicValue](#) [AddClassAllocation](#)
[AddComponent](#) [AddDescription](#) [AddFile](#) [AddFile2](#) [AddFileData](#) [AddFileData2](#)
[AddLongtext](#) [AddObjectLink](#) [AddStructure](#) [CallTransaction](#) [Change](#) [Change2](#)
[ChangeDialog](#) [Checkin](#) [Checkin2](#) [CheckinReplace](#) [CheckinRelease2](#) [Clear](#) [Create](#)
[Create2](#) [CreateDialog](#) [CreateNewVersion](#) [CreateNewVersion2](#) [DecodeDataCarrier](#)
[DeleteFile2](#) [DeleteObjectLink](#) [Display](#) [DisplayProductStructure](#) [EncodeDataCarrier](#)
[ExistenceCheck](#) [GetDetail](#) [GetDetail2](#) [GetFrontendType](#) [GetLatest](#) [GetList](#)
[GetObjectLinks](#) [GetStatus](#) [GetStructure](#) [Save](#) [Save2](#) [SaveObjectLinks](#) [Select](#)
[SelectViaMatchcode](#) [SetStatus](#) [SetStatusOtherVersions](#)

Engineering Change Master Functions

[AddEffectivity](#) [AddHeaderData](#) [AddObjectType](#) [AddTextLine](#) [AddValueAssignment](#)
[ChangeDialog](#) [ChangeHeader](#) [Clear](#) [CreateDialog](#) [CreateHeader](#) [DeleteHeader](#)
[Display](#) [DisplayProductStructure](#) [ExistenceCheck](#) [ReadHeader](#) [SaveHeader](#)
[SelectViaMatchcode](#)

Excel Functions

[CloseWorkBook](#) [FindPosition](#) [GetData](#) [Launch](#) [OpenWorkBook](#) [PutData](#)
[SaveWorkBook](#) [SelectSheet](#) [SetPosition](#) [Terminate](#)

File Functions

[Close](#) [Copy](#) [Delete](#) [Execute](#) [GetLine](#) [GetList](#) [Move](#) [OpenForAppend](#)
[OpenForInput](#) [OpenForOutput](#) [PutLine](#) [WaitForTask](#)

Function Module Functions

[AddChange](#) [AddExport](#) [AddImport](#) [AddTable](#) [AddTableLine](#) [Call](#) [Clear](#) [ReadTable](#)

Grid Functions

[AddLine](#) [SetColFontBold](#) [SetColFontItalic](#) [SetColWidth](#) [SetProperties](#)
[SetRowFontBold](#) [SetRowFontItalic](#) [Show](#) [Sort](#)

Material Functions

[AddBasicData](#) [AddBIData](#) [AddBOMData](#) [AddBOMData2](#) [AddBOMDataEx](#)
[AddBOMDataExt](#) [AddBOMDataExt2](#) [AddBOMHeaderData](#) [AddBOMHeaderData2](#)
[AddBOMHeaderDataExt](#) [AddBOMHeaderDataExt2](#) [AddBOMSubItemData](#)
[AddBOMSubItemData2](#) [AddClientData](#) [AddDescription](#) [AddExtension](#)
[AddForecastParameters](#) [AddHeadData](#) [AddLongtext](#) [AddPlantData](#) [AddSalesData](#)
[AddStorageLocationData](#) [AddStorageTypeData](#) [AddTaxClassifications](#)
[AddUnitOfMeasure](#) [AddValuationData](#) [AddWarehouseNumberData](#) [AppendBOMData](#)
[AppendBOMData2](#) [ChangeDialog](#) [Clear](#) [CloseBOM](#) [CreateDialog](#) [CreateRevision](#)
[DeleteBOM](#) [Display](#) [DisplayProductStructure](#) [ExistenceCheck](#) [GetDetail](#)
[GetInternalNumber](#) [GetInternalNumber2](#) [GetList](#) [ModifyBOMItem](#) [OpenBOM](#)
[ReadBOM](#) [ReadBOM2](#) [Save](#) [SaveBI](#) [SaveBOM](#) [SaveBOM2](#) [SelectBOMItem](#)
[SelectViaMatchcode](#) [UpdateBOM](#) [UpdateBOMItem](#)

Project Functions

[AddBasicData](#) [Clear](#) [Create](#) [ExistenceCheck](#) [GetDetail](#) [GetList](#) [Update](#)

Smarteam Functions

[CopyFile](#) [CopyFileExt](#) [CreateReportObject](#) [GetData](#) [GetLastRevisionObject](#) [GetLinks](#)
[GetLinksPlus](#) [GetList](#) [GetList2](#) [GetStructure](#) [GetStructurePlus](#) [InitiateProcess](#)
[LifeCycleOperation](#) [Login](#) [Logout](#) [RefreshWindow](#) [Run](#) [Save](#) [SendMessage](#)
[SetData](#) [SetObject](#) [Terminate](#)

Control Functions

[AndIf/AndIfNot](#) [Check](#) [DoLoop](#) [DoNext](#) [Exit](#) [ExitDoLoop](#) [ExitLoop](#) [Goto](#)
[If/ElseIf\(Not\)/Else/Endif](#) [IfNot/ElseIf\(Not\)/Else/Endif](#) [Include](#) [Label](#) [Loop](#) [New](#) [Next](#)
[SetErrorHandler](#) [Sleep](#)

Miscellaneous Functions

[# \(Comment\)](#) [CallExternalFunction](#) [Define](#) [ClearBapiMessages](#) [DisplayBapiMessages](#)
[ExtractString](#) [Message](#) [Progress](#) [SaveBapiMessagesToFile](#) [SendMail](#) [Set](#)
[SetExitCode](#) [SplitString](#) [SwitchSAPUser](#) [WaitOnCommit](#)

Batch Input Functions

Name	BatchInput.AddRecord
Arguments	<ProgramName> ¹ , <ScreenNumber> ² , <ScreenBegin> ³ , <FieldName> ⁴ , <FieldValue> ⁵
Description	Adds a batch input record to the internal memory for a later call of a transaction. For opening a new screen, arguments <ProgramName>, <ScreenNumber> and <ScreenBegin> have to be supplied. Then field values are programmed using <FieldName> and <FieldValue>.
Remarks	
Example	BatchInput.AddRecord SAPLMGMM, 0060, X BatchInput.AddRecord , , , RMMG1-MATNR, #Smarteam.TDM_SAP_MAT_NUM# BatchInput.AddRecord SAPLMGMM, 0070, X BatchInput.AddRecord , , , MSICHTAUSW-KZSEL(01), X BatchInput.AddRecord , , , MSICHTAUSW-KZSEL(02), X BatchInput.AddRecord SAPLMGMM, 4000, X BatchInput.AddRecord , , , BDC_OKCODE, "/00" BatchInput.CallTransaction MM01, X, A, S

Name	BatchInput.AddSetGetParameters
Arguments	<ParameterID1>, <ParameterValue1>, <ParameterID2>, <ParameterValue2>, ...
Description	Adds set/get parameter values to the internal memory for a later call of a transaction.
Remarks	
Example	BatchInput.AddSetGetParameters MAT, Smarteam.TDM_SAP_MAT_NUM#, CSV, 2 BatchInput.CallTransaction CS01, X

Name	BatchInput.CallTransaction
Arguments	<TransactionCode> ¹ , <SkipFirstScreen> ² , <CallMode> ³ , <UpdateMode> ⁴
Description	Calls transaction <TransactionCode>. <SkipFirstScreen> = X means: skip the first screen if all required field values are given. For possible values of arguments <CallMode> and <UpdateMode> see ABAP/4 documentation for "call transaction".
Remarks	
Example	See BatchInput.AddRecord and BatchInput.AddSetGetParameters statements

Name	BatchInput.Clear
Arguments	None

Description	Clears the internal memory for batch input operations.
Remarks	
Example	BatchInput.Clear

Class Functions

Name	Class.AddData
Arguments	<ClassType> ¹ , <ClassName> ² , <Date> ³ , <Language> ⁴ , <LanguageIso> ⁵
Description	Adds class key data to the internal memory for later calls to other class functions.
Remarks	
Example	Class.AddData 017, V_CAD_AB, , D Class.AddData 001, CL013, , EN

Name	Class.AddSelectionCriterion
Arguments	<CharName> ¹ , <CharValue> ² , <NumValFrom> ³ , <NumValTo> ⁴ , <CurrencyValFrom> ⁵ , <CurrencyValTo> ⁶ , <ValueRelation> ⁷ , <Unit> ⁸ , <BaseUomIso> ⁹ , <UnitText> ¹⁰
Description	Adds a selection criterion to internal memory for a later call to functions Class.SelectObjects .
Remarks	
Example	Class.AddSelectionCriterion V_DRAWINGSNUMBER, 4711 Class.AddSelectionCriterion V_DRAWINGFORMAT, A4

Name	Class.Clear
Arguments	None
Description	Clears the internal memory of the class.
Remarks	
Example	Class.Clear

Name	Class.GetCharacteristics
Arguments	<WithValues>
Description	Calls BAPI_CLASS_GET_CHARACTERISTICS with class data needed taken from internal memory and returns a list of the characteristics of the class. If <WithValues> is set to "X", allowed values of the characteristics are returned too.
Remarks	Looping on the resulting lists of characteristics and characteristic values is provided

	by the "Loop Class.Characteristics"/"Next Class.Characteristics" and "Loop Class.CharacteristicValues"/"Next Class.CharacteristicValues" statements.
Example	<pre> Class.AddData 017, V_CAD_AB, , D Class.GetCharacteristics X Loop Class.Characteristics ... Next Class.Characteristics Loop Class.CharacteristicValues ... Next Class.CharacteristicValues </pre>

Name	Class.SelectObjects
Arguments	<MaxHits> ¹ , <StatusLocked> ² , <StatusIncomplete> ³
Description	Calls BAPI_CLASS_SELECT_OBJECTS with class and classification data needed taken from internal memory and returns a list of objects that fulfill the selection criterions. <MaxHits> gives the maximum number of objects to be returned.
Remarks	The selection criterions are defined using function Class.AddSelectionCriterion . Looping on the resulting lists of selected objects is provided by the "Loop Class.SelectedObjects"/"Next Class.SelectedObjects" statements.
Example	<pre> Class.AddData 017, V_CAD_AB, , D Class.AddSelectionCriterion V_DRAWINGSNUMBER, 4711 Class.AddSelectionCriterion V_DRAWINGFORMAT, A4 Class.SelectObjects 10 Loop Class.SelectedObjects ... Next Class.SelectedObjects </pre>

Name	Class.SelectObjectsDialog
Arguments	None
Description	Calls RFC_SELECT_OBJECTS_VIA_CLASS with class and classification data needed taken from internal memory and returns a list of objects selected by the user in dialog.
Remarks	Looping on the resulting lists of selected objects is provided by the "Loop Class.SelectedObjects"/"Next Class.SelectedObjects" statements.
Example	<pre> Class.AddData 017, V_CAD_AB Class.SelectObjectsDialog Loop Class.SelectedObjects ... Next Class.SelectedObjects </pre>

Classification Functions

Name	Classification.AddData
Arguments	<ObjectType> ¹ , <ClassType> ² , <ClassName> ³ , <ECNumber> ⁴ , <Date> ⁵ , <Language> ⁶ , <ObjNotChk> ⁷ , <WithUnassignedCharacts> ⁸ , <WithInheritedCharacts> ⁹ , <Status> ¹⁰ , <IsStandardClass> ¹¹ , <DeleteAllocation> ¹²
Description	Adds classification key data to the internal memory for later calls to other classification functions.
Remarks	For possible values of object type and class type and for an explanation see SAP documentation.
Example	Classification.AddData MARA, 001 Classification.AddData MARA, 001, Levers, , , , , , , X Classification.AddData MARA, 001, CL12345, , , , , , 1, X

Name	Classification.AddIdentifier
Arguments	<Field> ¹ , <Value> ²
Description	Adds an object identifier to the internal memory for later calls to other classification functions.
Remarks	
Example	Classification.AddIdentifier MATNR, #SmarTeam.TDM_SAP_MAT_NUM# Classification.AddIdentifier DOKAR, %DOCTYPE% Classification.AddIdentifier DOKNR, %DOCNUMBER% Classification.AddIdentifier DOKTL, %DOCPART% Classification.AddIdentifier DOKVR, %DOCVERSION%

Name	Classification.AddValidation
Arguments	<Characteristic> ¹ , <Value> ² , <DeleteField> ³ , <Instance> ⁴
Description	Adds a characteristic value or the request to delete it to the internal memory for a later call to function Classification.SaveValidation .
Remarks	
Example	Classification.AddValidation MATERIALNUMBER, #SmarTeam.TDM_SAP_MAT_NUM# Classification.AddValidation MATERIALDESCRIPT, , X

Name	Classification.Clear
Arguments	None
Description	Clears the internal memory of the classification.
Remarks	
Example	Classification.Clear

Name	Classification.GetAllocations
Arguments	None
Description	Calls CACL_OBJECT_READ_ALLOCATIONS with classification and object data needed taken from internal memory and returns a list of the object's class allocations.
Remarks	Looping on the resulting list of allocated classes is provided by the "Loop Classification.Allocations"/"Next Classification.Allocations" statements.
Example	Classification.GetAllocations Loop Classification.Allocations ... Next Classification.Allocations

Name	Classification.GetValidations
Arguments	None
Description	Calls CACL_OBJECT_READ_VALIDATION with classification and object data needed taken from internal memory and returns a list of the object's characteristic validations.
Remarks	Looping on the resulting list of characteristic validations is provided by the "Loop Classification.Validations"/"Next Classification.Validations" statements. It is also possible to directly access characteristic values without loop using #Classification.CharacteristicValue.<CharacteristicName>#
Example	Classification.GetValidations Loop Classification.Validations ... Next Classification.Validations

Name	Classification.SaveAllocation
Arguments	None
Description	Calls CACL_OBJECT_ALLOCATION_MAINT with classification and object data needed taken from internal memory.
Remarks	
Example	Classification.SaveAllocation

Name	Classification.SaveValidation
Arguments	None
Description	Calls CACL_OBJECT_VALIDATION_MAINT with classification and object data needed taken from internal memory.
Remarks	
Example	Classification.SaveValidation

Document Functions

Name	Document.AddBasicData
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <Description> ⁵ , <UserName> ⁶ , <StatusExtern> ⁷ , <StatusIntern> ⁸ , <StatusLog> ⁹ , <Laboratory> ¹⁰ , <AuthorityGroup> ¹¹ , <ECNumber> ¹² , <DeleteIndicator> ¹³ , <CadIndicator> ¹⁴ , <RefDocumentNumber> ¹⁵ , <RefDocumentPart> ¹⁶ , <RefDocumentVersion> ¹⁷ , <PreDocumentType> ¹⁸ , <PreDocumentNumber> ¹⁹ , <PreDocumentPart> ²⁰ , <PreDocumentVersion> ²¹ , <UserDefined1> ²² , <UserDefined2> ²³ , <UserDefined3> ²⁴ , <UserDefined4> ²⁵
Description	Adds a document's basic data for non K-Pro document types to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddBasicData DRW, #SmarTeam.CN_ID#, 000, #SmarTeam.REVISION#, _ #SmarTeam.CN_DESCRIPTION#, #SmarTeam.USER_OBJECT_ID#, AA, _ Created by SMARTEAM, , , , , X

Name	Document.AddBasicData2
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <Description> ⁵ , <UserName> ⁶ , <StatusExtern> ⁷ , <StatusIntern> ⁸ , <StatusLog> ⁹ , <Laboratory> ¹⁰ , <AuthorityGroup> ¹¹ , <ECNumber> ¹² , <RevisionLevel> ¹³ , <DeleteIndicator> ¹⁴ , <CadIndicator> ¹⁵ , <RefDocumentNumber> ¹⁶ , <RefDocumentPart> ¹⁷ , <RefDocumentVersion> ¹⁸ , <PreDocumentType> ¹⁹ , <PreDocumentNumber> ²⁰ , <PreDocumentPart> ²¹ , <PreDocumentVersion> ²² , <UserDefined1> ²³ , <UserDefined2> ²⁴ , <UserDefined3> ²⁵ , <UserDefined4> ²⁶ , <CMFixed> ²⁷ , <CMRelevance> ²⁸ , <DocBOMChangeNumber> ²⁹ , <DocBOMRevisionLevel> ³⁰ , <DocBOMValidFrom> ³¹
Description	Adds a document's basic data for K-Pro document types to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddBasicData DRW, #SmarTeam.CN_ID#, 000, #SmarTeam.REVISION#, _ #SmarTeam.CN_DESCRIPTION#, #SmarTeam.USER_OBJECT_ID#, _ AA, Created by SMARTEAM, , , , , X

Name	Document.AddCharacteristicValue
Arguments	<ClassType> ¹ , <ClassName> ² , <CharName> ³ , <CharValue> ⁴ , <CharValueLength> ⁵ , <DeleteValue> ⁶
Description	Adds a document's characteristic value to the internal memory for later calls to other document functions.
Remarks	Document.AddClassAllocation should have been called before.
Example	Document.AddCharacteristicValue 017, DOC_DRW, DOC_DRW_RELEASED_BY, _

	#SmarTeam.CN_RELEASED_BY#, 10
--	-------------------------------

Name	Document.AddClassAllocation
Arguments	<ClassType> ¹ , <ClassName> ² , <Status> ³ , <StandardClass> ⁴ , <ECNumber> ⁵ , <DeleteAllocation> ⁶
Description	Adds a document class to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddClassAllocation 017, DOC_DRW, ,X

Name	Document.AddComponent
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <OriginalType> ⁵ , <SourceDataCarrier> ⁶ , <StorageCategory> ⁷ , <DocFile> ⁸ , <Format> ⁹ , <Description> ¹⁰ , <Language> ¹¹ , <PhObjID> ¹² , <DeleteValue> ¹³
Description	Adds a document's component for K-Pro document types as part of the document's component list to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddComponent DRW, #SmarTeam.CN_ID#, 000, #SmarTeam.REVISION#, 2, , VaultST, _ #SmarTeam.FILE_NAME#

Name	Document.AddDescription
Arguments	<Language> ¹ , <LanguageIso> ² , <Description> ³ , <TextIndicator> ⁴ , <DeleteValue> ⁵
Description	Adds a document description to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddDescription , DE, #SmarTeam.CN_DESCRIPTION_DE# Document.AddDescription , EN, #SmarTeam.CN_DESCRIPTION_EN#

Name	Document.AddFile
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <OriginalType> ⁵ , <SourceDataCarrier> ⁶ , <DataCarrier> ⁷ , <WSApplication> ⁸ , <DocFile> ⁹ , <StatusExtern> ¹⁰ , <StatusIntern> ¹¹ , <StatusLog> ¹²
Description	Adds a document's file data for non K-Pro document types as part of the document's file list to the internal memory for later calls to other document functions.
Remarks	If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.AddFile DRW, #SmarTeam.CN_ID#, 000, #SmarTeam.REVISION#, 2, , VaultST, TIF, _

	#SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME#
--	---

Name	Document.AddFile2
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <OriginalType> ⁵ , <SourceDataCarrier> ⁶ , <StorageCategory> ⁷ , <WSApplication> ⁸ , <DocPath> ⁹ , <DocFile> ¹⁰ , <StatusExtern> ¹¹ , <StatusIntern> ¹² , <StatusLog> ¹² , <ApplicationID> ¹³ , <FileID> ¹⁴ , <Description> ¹⁵ , <Language> ¹⁶ , <DeleteValue> ¹⁷
Description	Adds a document's file data for K-Pro document types as part of the document's file list to the internal memory for later calls to other document functions.
Remarks	If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.AddFile2 DRW, #SmarTeam.CN_ID#, 000, #SmarTeam.REVISION#, 2, , VaultST, TIF, _#SmarTeam.DIRECTORY#, #SmarTeam.FILE_NAME#

Name	Document.AddFileData
Arguments	<DocFile1> ¹ , <DataCarrier1> ² , <WSApplication1> ³ , <DocFile2> ⁴ , <DataCarrier2> ⁵ , <WSApplication2> ⁶ , <SaveDocFile1> ⁷ , <SaveDataCarrier1> ⁸ , <SaveDocFile2> ⁹ , <SaveDataCarrier2> ¹⁰ , <FileSize1> ¹¹ , <FileSize2> ¹²
Description	Adds a document's file data for non K-Pro document types as part of the document's basic data to the internal memory for later calls to other document functions.
Remarks	The document's basic data must be set using Document.AddBasicData . If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.AddFileData , , , #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME# , , TIF

Name	Document.AddFileData2
Arguments	<DocFile1> ¹ , <DataCarrier1> ² , <WSApplication1> ³ , <DocFile2> ⁴ , <DataCarrier2> ⁵ , <WSApplication2> ⁶ , <SaveDocFile1> ⁷ , <SaveDataCarrier1> ⁸ , <SaveDocFile2> ⁹ , <SaveDataCarrier2> ¹⁰ , <FileSize1> ¹¹ , <FileSize2> ¹²
Description	Adds a document's file data for K-Pro document types as part of the document's basic data to the internal memory for later calls to other document functions.
Remarks	The document's basic data must be set using Document.AddBasicData2 . If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.AddFileData2 , , , #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME# , , TIF

Name	Document.AddLongtext
Arguments	<Language> ¹ , <LanguageIso> ² , <TextLine> ³ , <DeleteValue> ⁴

Description	Adds a document longtext to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddLongtext , EN, #Smarteam.CN_HISTORY#

Name	Document.AddObjectLink
Arguments	<ObjectType> ¹ , <ObjectKey> ² , <DocumentDirection> ³ , <ObjectDescription> ⁴ , <DeleteValue> ⁵
Description	Adds a document link to other SAP objects to the internal memory for later calls to other document functions.
Remarks	
Example	Document.AddObjectLink MARA, #Smarteam.TDM_SAP_MAT_NUM# Document.AddObjectLink PRPS, ST.00001

Name	Document.AddStructure
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <Quantity> ⁵ , <DeleteValue> ⁶
Description	Adds a document component to the internal memory for later calls to other document functions.
Remarks	This function makes sense for example in a loop on all SMARTEAM children.
Example	Loop Smarteam.Structure Document.AddStructure DRW, #Smarteam.Structure.CN_ID#, 000, #Smarteam.Structure.REVISION# Next Smarteam.Structure

Name	Document.CallTransaction
Arguments	<TransactionCode> ¹ , <SkipFirstScreen> ²
Description	Calls transaction <TransactionCode>. <SkipFirstScreen> = X means: skip the first screen if all document key values are given.
Remarks	
Example	Document.CallTransaction CV02N, X

Name	Document.Change
Arguments	None
Description	Calls BAPI_DOCUMENT_CHANGE with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously.

	If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.Change

Name	Document.Change2
Arguments	None
Description	Calls BAPI_DOCUMENT_CHANGE2 with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.Change2

Name	Document.ChangeDialog
Arguments	None
Description	Calls RFC_CHANGE_DOCUMENT_MASTER with data taken from internal memory. I. e. the document info record in memory can be modified using SAP's dialog transaction.
Remarks	
Example	Document.ChangeDialog

Name	Document.Checkin
Arguments	None
Description	Calls BAPI_DOCUMENT_CHECKIN with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.Checkin

Name	Document.Checkin2
Arguments	None
Description	Calls BAPI_DOCUMENT_CHECKIN2 with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".

Example	Document.Checkin2
---------	-------------------

Name	Document.CheckinReplace
Arguments	None
Description	Calls BAPI_DOCUMENT_CHECKINREPLACE with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.CheckinReplace

Name	Document.CheckinReplace2
Arguments	None
Description	Calls BAPI_DOCUMENT_CHECKINREPLACE2 with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.CheckinReplace2

Name	Document.Clear
Arguments	None
Description	Clears the internal memory for documents.
Remarks	To clear specific parts of the internal document memory use functions Document.ClearBasicData Document.ClearBasicData2 Document.ClearCharacteristicValues Document.ClearClassAllocations Document.ClearComponents Document.ClearDescriptions Document.ClearFiles Document.ClearFiles2 Document.ClearFileData Document.ClearFileData2 Document.ClearLongTexts Document.ClearObjectLinks Document.ClearStatusList Document.ClearStatusLog Document.ClearStructure

Example	Document.Clear
---------	----------------

Name	Document.Create
Arguments	None
Description	Calls BAPI_DOCUMENT_CREATE with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.Create

Name	Document.Create2
Arguments	None
Description	Calls BAPI_DOCUMENT_CREATE2 with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If you get error messages in file check-in operations, read section " Error when storing files in SAP vaults or storage categories ".
Example	Document.Create2

Name	Document.CreateDialog
Arguments	None
Description	Calls RFC_CREATE_DOCUMENT_MASTER with data taken from internal memory. I. e. the document info record in memory can be created using SAP's dialog transaction.
Remarks	
Example	Document.CreateDialog

Name	Document.CreateNewVersion
Arguments	<RefDocumentversion> ¹ , <NewDocumentversion> ² , <CopyOriginals> ³ , <ObjectLinksToCopy (DRAW;MARA;...)> ⁴ , <CadMode> ⁵
Description	Calls BAPI_DOCUMENT_CREATENEWVERSION with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If <RefDocumentversion> is not given, document version set in

	Document.AddBasicData is used. If <NewDocumentversion> is not given, document version from BasicData is used. In parameter <ObjectLinksToCopy> all object link types to be copied can be specified separated by a semi-colon.
Example	Document.CreateNewVersion #SmarTeam.PAR_REVISION#, , DRAW;MARA, X

Name	Document.CreateNewVersion2
Arguments	<RefDocumentversion> ¹ , <NewDocumentversion> ² , <CopyClassification> ³ , <CopyDocumentBom> ⁴ , <CopyOriginals> ⁵ , <ObjectLinksToCopy> ⁶ (DRAW;MARA;...)> ⁶
Description	Calls BAPI_DOCUMENT_CREATENEWVR2 with data taken from internal memory.
Remarks	The wait-on-commit parameter controls whether the database commit is performed synchronously or asynchronously. If <RefDocumentversion> is not given, document version set in Document.AddBasicData2 is used. If <NewDocumentversion> is not given, document version from BasicData2 is used. In parameter <ObjectLinksToCopy> all object link types to be copied can be specified separated by a semi-colon.
Example	Document.CreateNewVersion2 #SmarTeam.PAR_REVISION#, X, , DRAW;MARA

Name	Document.DecodeDataCarrier
Arguments	<DataCarrier> ¹ , <Hostname> ²
Description	Calls BAPI_DOCUMENT_GETDCLIST and determines the path which is associated with the given data carrier. If needed, the value of environment variable #System.Env.HOSTNAME# can be passed as <Hostname>.
Remarks	The resulting path can be accessed by #Document.DocPath#. If the data carrier cannot be resolved, i. e. if the data carrier is not found in the list returned or if no path is associated with the data carrier, #Document.DocPath# will be set to "".
Example	Document.DecodeDataCarrier #SmarTeam.CN_SAP_DATA_CARRIER# If #Document.DocPath# != ! Message Error, Data carrier #SmarTeam.CN_SAP_DATA_CARRIER# not found. Endif

Name	Document.DeleteFile2
Arguments	<Condition1> ¹ , <Or Condition2> ² , <Or Condition3> ³ , ...
Description	In a loop at Document.Files2 table this function sets the delete flag of the current loop file.
Remarks	The function can be used only inside a "Loop Document.Files2"/"Next Document.Files2" loop. Inside this loop, it sets the delete flag for the given file if any of the <Conditions> are met. A <Condition> will usually look like #Document.Files2.<Fieldname># = <Fieldvalue>. The actual delete operation is

	performed when the document is saved via Document.Change2 . The document files can be read into internal memory via function Document.GetDetail2 .
Example	Document.GetDetail2 X Loop Document.Files2 Document.DeleteFile2 #Document.Files2.WsApplication# = TIF Next Document.Files2 Document.Change2

Name	Document.DeleteObjectLink
Arguments	<Condition1> ¹ , <Or Condition2> ² , <Or Condition3> ³ , ...
Description	In a loop at Document.Links table this function sets the delete flag of the current loop object link.
Remarks	The function can be used only inside a "Loop Document.Links"/"Next Document.Links" loop. Inside this loop, it sets the delete flag for the given object link if any of the <Conditions> are met. A <Condition> will usually look like #Document.Links.<Fieldname># = <Fieldvalue>. The actual delete operation is performed when the document is saved via Document.Change or Document.Change2 or when the document's object links are saved via Document.SaveObjectLinks . The document's object links can be read into internal memory via functions Document.GetDetail and Document.GetDetail2 .
Example	Document.GetDetail2 , , , X Loop Document.Links Document.DeleteObjectLink #Document.Links.ObjectType# = MARA Next Document.Links Document.Change2

Name	Document.Display
Arguments	None
Description	Calls RFC_DISPLAY_DOCUMENT_MASTER with data taken from internal memory. I. e. the document info record in memory can be viewed using SAP's dialog transaction.
Remarks	
Example	Document.Display

Name	Document.DisplayProductStructure
Arguments	None
Description	Calls RFC_EXPLODE_PRODUCT_STRUCTURE with document's key data taken from internal memory. I. e. SAP's product structure browser will be displayed for the document in memory.

Remarks	
Example	Document.DisplayProductStructure

Name	Document.EncodeDataCarrier
Arguments	<FileName> ¹ , <FrontendType> ²
Description	Calls RFC_DETERMINE_ACCESS_PATH to split a filename with absolute path into a SAP data carrier and residual filename.
Remarks	If no <FrontendType> is specified, the function uses the value from a previous call to Document.GetFrontendType . The resulting data carrier and residual filename can be accessed by #Document.DataCarrier# and #Document.DocFile#. If no data carrier can be found, the complete filename is returned in #Document.DocFile#.
Example	Document.GetFrontendType #System.Env.HOSTNAME# Document.EncodeDataCarrier #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME# If #Document.DocFile# = #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME# Message Error, No data carrier could be found for directory #SmarTeam.DIRECTORY#. Endif

Name	Document.ExistenceCheck
Arguments	None
Description	Calls BAPI_DOCUMENT_EXISTENCECHECK with the document's key data taken from internal memory.
Remarks	If the document exists, #Document.Exists# is equal to "X" and " " otherwise. If the document is marked for deletion, #Document.IsDeleted# is equal to "X" and " " otherwise.
Example	Document.ExistenceCheck

Name	Document.GetDetail
Arguments	<GetDescriptions> ¹ , <GetLongTexts> ² , <GetObjectLinks> ³ , <GetLinkDescriptions> ⁴ , <GetStatusLog> ⁵
Description	Calls BAPI_DOCUMENT_GETDETAIL with the document's key data taken from internal memory.
Remarks	The function reads the document's basic data. The document's descriptions, long texts, object links, link descriptions, and the status log are read in addition if the corresponding argument is set to X.
Example	Document.GetDetail X, ,X, ,X

Name	Document.GetDetail2
------	----------------------------

Arguments	<GetActiveFiles> ¹ , <GetComponents> ² , <GetLongTexts> ³ , <GetObjectLinks> ⁴ , <GetStatusLog> ⁵
Description	Calls BAPI_DOCUMENT_GETDETAIL2 with the document's key data taken from internal memory.
Remarks	The function reads the document's basic data, file data, descriptions, class allocations, and characteristic values. The active files, components, long texts, object links and the status log are read in addition if the corresponding argument is set to X.
Example	Document.GetDetail2 X, X, , , X

Name	Document.GetFrontendType
Arguments	<Hostname>
Description	Calls BAPI_DOCUMENT_GETFRONTENDTYPE retrieving the frontend type of the local PC.
Remarks	When working with SAP data carriers, SAP's DMS needs to have an environment variable HOSTNAME to be set. This will usually be the input for this function. The resulting frontend type can be accessed via #Document.Frontendtype#.
Example	Document.GetFrontendType #System.Env.HOSTNAME#

Name	Document.GetLatest
Arguments	<LoadLatest> ¹ , <LoadLatestReleased> ²
Description	Calls BAPI_DOCUMENT_GETLATEST with the document's key data taken from internal memory.
Remarks	For documentation of arguments see corresponding BAPI help texts. The latest version returned by SAP is accessible by #Document.LatestVersion#.
Example	Document.GetLatest , X

Name	Document.GetList
Arguments	<DocumentType> ¹ , <DocumentNumber> ² , <DocumentPart> ³ , <DocumentVersion> ⁴ , <Description> ⁵ , <Language> ⁶ , <LanguageIso> ⁷ , <UserName> ⁸ , <StatusExtern> ⁹ , <StatusIntern> ¹⁰ , <Laboratory> ¹¹ , <AuthorityGroup> ¹² , <ECNumber> ¹³ , <DeleteIndicator> ¹⁴ , <CadIndicator> ¹⁵ , <DataCarrier> ¹⁶ , <WSApplication> ¹⁷ , <MaxRows> ¹⁸
Description	Calls BAPI_DOCUMENT_GETLIST with the search criteria specified and returns a list of documents satisfying the given criteria into internal memory.
Remarks	Looping on the resulting list of documents is provided by the "Loop Document.List"/"Next Document.List" statements.
Example	Document.GetList DRW, 10508*, 000, *, *saw*, , EN, ,RE , , , , , *, , , 100 Loop Document.List ... Next Document.List

	Document.GetList DRW, from: 4711* to: 4811*
--	---

Name	Document.GetObjectLinks
Arguments	<LinkType1 (MARA,DRAW,PRPS,KNA1,...), <LinkType2>, <LinkType3>, ...
Description	Calls BAPI_DOCUMENT_GETOBJECTLINKS with the document's key data taken from internal memory and returns a list of the document's object links with one of the link types passed as argument. For possible values of the link types and an explanation see SAP's DMS documentation. Specifying no link type returns a list of all existing object links of the document.
Remarks	Looping on the resulting list of object links is provided by the "Loop Documents.Links"/"Next Documents.Links" statements.
Example	Document.GetObjectLinks MARA, DRAW Loop Document.Links ... Next Document.Links

Name	Document.GetStatus
Arguments	None
Description	Calls BAPI_DOCUMENT_GETSTATUS with the document's key data taken from internal memory and returns the internal and external status and the status log of the document's present status into internal memory.
Remarks	
Example	Document.GetStatus

Name	Document.GetStructure
Arguments	<MultiLevelExplosion>
Description	Calls BAPI_DOCUMENT_GETSTRUCTURE with the document's key data taken from internal memory and returns the document's structure into internal memory. If <MultiLevelExplosion> is set to "X", the structure is exploded through multiple levels.
Remarks	
Example	Document.GetStructure X

Name	Document.Save
Arguments	None
Description	Performs Document.Create or Document.Change depending on whether the document in internal memory does already exist or not exist in SAP.
Remarks	

Example	Document.Save
---------	---------------

Name	Document.Save2
Arguments	None
Description	Performs Document.Create2 or Document.Change2 depending on whether the document in internal memory does already exist or not exist in SAP.
Remarks	
Example	Document.Save2

Name	Document.SaveObjectLinks
Arguments	None
Description	Calls BAPI_DOCUMENT_SAVEOBJECTLINKS adding the object links in internal memory to the document specified by the key data in internal memory.
Remarks	
Example	Document.SaveObjectLinks

Name	Document.Select
Arguments	<DocumentType>, <DocumentNumber>, <DocumentPart>, <DocumentVersion>, <UserName>, <StatusExtern>, <Laboratory>, <ECNumber>, <SkipFirstScreen>
Description	Calls RFC_SELECT_DOCUMENT_MASTER with the search criteria specified giving the user the ability to perform a CV04n like search.
Remarks	
Example	Document.Select DRW, , 000, , #SmarTeam.User.Login#, RE

Name	Document.SelectViaMatchcode
Arguments	None
Description	Calls RFC_SELECT_OBJ_VIA_MATCHCODE for documents allowing the user to select a document info record using SAP's dialog matchcode selection.
Remarks	The document key selected by the user can be passed to ENOVIA SmarTeam using the SmarTeam.SetData function.
Example	Document.SelectViaMatchcode Check #SAP.DialogStatus# <> A SmarTeam.SetData CN_ID, #Document.DocumentNumber#

Name	Document.SetStatus
Arguments	None
Description	Calls BAPI_DOCUMENT_SETSTATUS setting the status of the document, both being specified in internal memory.
Remarks	
Example	Document.SetStatus

Name	Document.SetStatusOtherVersions
Arguments	<Present external status of other version> ¹ , <Present internal status of other version> ² , <Next external status of other version> ³ , <Next internal status of other version> ⁴ , <Statuslog> ⁵ , <CompareMethod (Equal, NotEqual, Greater, GreaterOrEqual, Less, LessOrEqual)> ⁶
Description	Uses BAPI_DOCUMENT_GETLIST and BAPI_DOCUMENT_SETSTATUS for setting the status of documents with the same document type, number, and part as the document in internal memory. The compare method controls which other versions are taken into account. Default is NotEqual.
Remarks	
Example	Document.SetStatusOtherVersions FR, , UN, , Set by SMARTEAM, Less

Engineering Change Master Functions

Name	ECMaster.AddEffectivity
Arguments	<ValidFrom> ¹ , <ValidTo> ² , <DateMark> ³ , <Material> ⁴ , <SerialNrLow> ⁵ , <SerialNrHigh> ⁶ , <SernrOi> ⁷ , <FIDelete> ⁸
Description	Adds an EC master effectivity to the internal memory for later calls to other EC master functions.
Remarks	
Example	ECMaster.AddEffectivity #SmarTeam.CN_VALIDFROM#, #SmarTeam.CN_VALIDTO#

Name	ECMaster.AddHeaderData
Arguments	<ECNumber> ¹ , <Status> ² , <AuthGroup> ³ , <ValidFrom> ⁴ , <Description> ⁵ , <ReasonChg> ⁶ , <IndateRule> ⁷ , <OutdateRule> ⁸ , <Function> ⁹ , <ChangeLeader> ¹⁰ , <EffectivityType> ¹¹ , <OverridingMark> ¹² , <Rank> ¹³ , <ReleaseKey> ¹⁴ , <StatusProfile> ¹⁵ , <TechRel> ¹⁶ , <BasicChange> ¹⁷ , <DeletionMark> ¹⁸
Description	Adds an EC master's header data to the internal memory for later calls to other EC master functions.
Remarks	
Example	ECMaster.AddHeaderData #SmarTeam.CN_ECNUMBER#, 1, , #SmarTeam.CN_VALIDFROM#, _ #SmarTeam.CN_CHANGE_TEXT#, #SmarTeam.CN_CHANGE_REASON#

Name	ECMaster.AddObjectType
Arguments	<Name> ¹ , <Active> ² , <ManagementRecordRequired> ³ , <ManagementRecordGenerated> ⁴ , <GenerateForNewObjectsOnly> ⁵ , <GenerateDialog> ⁶ , <Locked> ⁷
Description	Adds an EC master's header data to the internal memory for later calls to other EC master functions.
Remarks	<Name> is the name of the object type. It can have the following values: BOM, BOM_CUS, BOM_DOC, BOM_EQUI, BOM_LOC, BOM_MAT, BOM_PSP, BOM_STD, CHAR, CLS, CLS_MAINT, CONF_PROF, DEP, DOC, HAZMAT, MAT, PHRASE, PVS, PVS_ALT, PVS_VAR, SUBSTANCE, TLIST, VAR_TAB. The six following arguments are exactly the checkboxes shown in transaction CC01, button "ObjectTypes".
Example	ECMaster.AddObjectType BOM, X, X, X

Name	ECMaster.AddTextLine
Arguments	<LineFormat> ¹ , <LineValue> ²
Description	Adds an EC master log text to the internal memory for later calls to other EC master functions.
Remarks	
Example	ECMaster.AddTextLine /, The object has to be modified according to the following rules: ECMaster.AddTextLine /, #SmarTeam.CN_CHANGE_RULE_1# ECMaster.AddTextLine /, #SmarTeam.CN_CHANGE_RULE_2# ECMaster.AddTextLine /, #SmarTeam.CN_CHANGE_RULE_3#

Name	ECMaster.AddValueAssignment
Arguments	<ValidFrom> ¹ , <ValidTo> ² , <DateMark> ³ , <Material> ⁴ , <SerialNrLow> ⁵ , <SerialNrHigh> ⁶ , <SernrOi> ⁷ , <FIDelete> ⁸
Description	Adds an EC master value assignment to the internal memory for later calls to other EC master functions.
Remarks	
Example	ECMaster.AddValueAssignment #SmarTeam.CN_VALIDFROM#, #SmarTeam.CN_VALIDTO#

Name	ECMaster.ChangeDialog
Arguments	None
Description	Calls RFC_CHANGE_CHANGE_MASTER with data taken from internal memory. I. e. the engineering change master in memory can be modified using SAP's dialog transaction.
Remarks	

Example	ECMaster.ChangeDialog
---------	-----------------------

Name	ECMaster.ChangeHeader
Arguments	None
Description	Calls CCAP_ECN_HEADER_CHANGE with data taken from internal memory. I. e. the engineering change master in memory is modified without dialog.
Remarks	
Example	ECMaster.ChangeHeader

Name	ECMaster.Clear
Arguments	None
Description	Clears the internal memory for EC masters.
Remarks	
Example	ECMaster.Clear

Name	ECMaster.CreateDialog
Arguments	None
Description	Calls RFC_CREATE_CHANGE_MASTER with data taken from internal memory. I. e. the engineering change master in memory can be created using SAP's dialog transaction.
Remarks	
Example	ECMaster.CreateDialog

Name	ECMaster.CreateHeader
Arguments	None
Description	Calls CCAP_ECN_HEADER_CREATE with data taken from internal memory. I. e. the engineering change master in memory is created without dialog.
Remarks	
Example	ECMaster.CreateHeader

Name	ECMaster.DeleteHeader
Arguments	None

Description	Calls CCAP_ECN_HEADER_DELETE with data taken from internal memory. I. e. the engineering change master in memory is deleted without dialog.
Remarks	
Example	ECMaster.DeleteHeader

Name	ECMaster.Display
Arguments	None
Description	Calls RFC_DISPLAY_CHANGE_MASTER with data taken from internal memory. I. e. the engineering change master in memory can be viewed using SAP's dialog transaction.
Remarks	
Example	ECMaster.Display

Name	ECMaster.DisplayProductStructure
Arguments	None
Description	Calls RFC_EXPLODE_PRODUCT_STRUCTURE with engineering change number taken from internal memory. I. e. SAP's product structure browser will be displayed for the engineering change master in memory.
Remarks	
Example	ECMaster.DisplayProductStructure

Name	ECMaster.ExistenceCheck
Arguments	None
Description	Checks if EC master already exists by calling RFC_READ_TABLE on table AENR with EC number taken from internal memory.
Remarks	If the EC master exists, #ECMaster.Exists# is equal to "X" and " " otherwise.
Example	ECMaster.ExistenceCheck

Name	ECMaster.ReadHeader
Arguments	None
Description	Calls CCAP_ECN_HEADER_READ with the engineering change number taken from internal memory. I. e. the engineering change master header for the given change number is read into memory without dialog.
Remarks	
Example	ECMaster.ReadHeader

Name	ECMaster.SaveHeader
Arguments	None
Description	Performs ECMaster.CreateHeader or ECMaster.ChangeHeader depending on whether the EC number in internal memory does already exist or not exist in SAP.
Remarks	
Example	ECMaster.SaveHeader

Name	ECMaster.SelectViaMatchcode
Arguments	None
Description	Calls RFC_SELECT_OBJ_VIA_MATCHCODE for engineering change masters allowing the user to select an engineering change number using SAP's dialog matchcode selection.
Remarks	The engineering change number selected by the user can be passed to ENOVIA SmarTeam using the Smarteam.SetData function.
Example	ECMaster.SelectViaMatchcode Check #SAP.DialogStatus# <> A SmarTeam.SetData CN_ECNUMBER, #ECMaster.ECNumber#

Excel Functions

Name	Excel.CloseWorkBook
Arguments	None
Description	Closes the workbook currently open.
Remarks	
Example	Excel.CloseWorkBook

Name	Excel.FindPosition
Arguments	<StartRow> ¹ , <EndRow> ² , <StartColumn> ³ , <EndColumn> ⁴ , <Findstring> ⁵ , <CaseSensitive> ⁶
Description	Sets the position to the first cell where <FindString> is found. If <CaseSensitive> is set to "X", search is case-sensitive. The search runs from <StartRow> to <EndRow> and from <StartColumn> to <EndColumn>. If <StartRow> or <StartColumn> is omitted, value 1 is used instead. If <EndRow> (<EndColumn>) is missing, the current row (column) is used. The loop from <StartRow> to <EndRow> is the "inner" loop meaning that the search runs through all specified rows before moving to the next column. If no occurrence of <FindString> is found, the current row and column remain unchanged.

Remarks	
Example	Excel.FindPosition , 20, 2, 2, Item No.

Name	Excel.GetData
Arguments	<Variable 1>, <Variable 2>, <Variable 3>, ...
Description	Reads cell values from adjacent cells of the current row into the given variables starting from the current position. The value of the first cell is written to <Variable 1>, the value of the second cell right next to the first one is written to <Variable 2>, and so on. After all variables have been read, the current row position is incremented by 1 so that the next GetData statement reads values from the next row. The starting column remains the same than in the previous GetData statement.
Remarks	The first starting position can be set using Excel.SetPosition .
Example	Excel.GetData %MATNO%, %MATTYPE%, %DIVISION%, %BASEUOM%

Name	Excel.Launch
Arguments	<Visible> ¹ , <DoNotClose> ²
Description	Launches a new Excel application. If <Visible> is set to "X", Excel will be visible. Otherwise it remains invisible. If <DoNotClose> is set to "X", Excel is not closed automatically when the ENOVIA SmarTeam Script Engine terminates.
Remarks	
Example	Excel.Launch X, X

Name	Excel.OpenWorkBook
Arguments	<FileName>
Description	Opens the workbook stored in file <FileName>.
Remarks	
Example	Excel.OpenWorkBook #SmarTeam.LocalConfig.USER_DIR#BOM-#SmarTeam.CN_ID#.xls

Name	Excel.PutData
Arguments	<Value 1>, <Value 2>, <Value 3>, ...
Description	Writes the values passed as arguments into adjacent cells of the current worksheet starting at the current position. <Value 1> is written to the cell at the current position, <Value 2> is written to the cell right next to the first one, and so on. After all values have been written, the current row position is incremented by 1 so that the next PutData statement writes values into the next row. The starting column remains the same than in the previous PutData statement.
Remarks	The first starting position can be set using Excel.SetPosition .

Example	Excel.PutData #SmarTeam.CN_ID#, #SmarTeam.CN_MATTYPE#, #SmarTeam.CN_BASEUOM#
---------	--

Name	Excel.SaveWorkBook
Arguments	<FileName>
Description	Saves the workbook under <FileName>. If <FileName> is not given, the workbook is saved under its current name.
Remarks	
Example	Excel.SaveWorkBook #SmarTeam.LocalConfig.USER_DIR#temp-#SmarTeam.CN_ID#.xls

Name	Excel.SelectSheet
Arguments	<SheetNameOrNumber>
Description	Selects (activates) the worksheet of the current workbook named <SheetNameOrNumber>. If <SheetNameOrNumber> is an integer number, it is interpreted as the sheet number instead of the sheet name. If <SheetNameOrNumber> is not given, the active sheet is selected.
Remarks	
Example	Excel.SelectSheet BOM

Name	Excel.SetColWidth
Arguments	<Width of column A>, <Width of column B>, <Width of column C>, ...
Description	Specifies the width of the Excel columns in Excel units.
Remarks	
Example	Excel.SetColWidth 10, 15, 10, 10, 30

Name	Excel.SetPosition
Arguments	<Row> ¹ , <Column> ²
Description	Sets the position in the current worksheet to row <Row> and column <Column>. If <Row> is not given, the current row remains unchanged. If <Column> is not given, the current column remains unchanged.
Remarks	The upper left cell of a worksheet has coordinates 1, 1.
Example	Excel.SetPosition , 10

Name	Excel.Sort
------	-------------------

Arguments	<Range> ¹ , <Key> ² , <Direction (Ascending, Descending)> ³
Description	Excel range <Range> will be sorted according to key <Key> in ascending or descending order.
Remarks	
Example	Excel.Sort A3:F100, A3, Ascending

Name	Excel.Terminate
Arguments	None
Description	Terminates the Excel application (Excel is closed).
Remarks	
Example	Excel.Terminate

File Functions

Name	File.Close
Arguments	None
Description	The file opened by the most recent call to File.Open is closed.
Remarks	
Example	File.Close

Name	File.Copy
Arguments	<Source> ¹ , <Target> ²
Description	The file given as <Source> is copied on the file given as <Target>.
Remarks	Both paths have to be fully qualified.
Example	File.Copy #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME#, #SmarTeam.DIRECTORY#\Temp.Extension{#SmarTeam.FILE_NAME#}

Name	File.Delete
Arguments	<Source>
Description	The file given as <Source> is deleted.
Remarks	The path has to be fully qualified.
Example	File.Delete #SmarTeam.DIRECTORY#\Filename{#SmarTeam.FILE_NAME#}.tmp

Name	File.Execute
Arguments	<ExecString> ¹ , <WindowStyle> ²
Description	Program and program arguments passed as <ExecString> are executed by the shell command. <WindowStyle> can be 0, 1, 2, 3, 4 or 6 opening the new window in different states: 0 (window is hidden), 1 (window has normal size and focus), 2 (window is minimized and has focus), 3 (window is maximized and has focus), 4 (window has normal size but no focus), 6 (window is minimized but has no focus)
Remarks	The program's path must be fully qualified or must be found in a path from the %PATH% variable.
Example	generate_tiff #Smarteam.DIRECTORY#\#Smarteam.FILE_NAME# #Smarteam.DIRECTORY#\#Smarteam.FILE_NAME#.tif, 6

Name	File.GetLine
Arguments	None
Description	The next line of the file opened by the most recent call to File.OpenForInput is read.
Remarks	Before performing a File.GetLine operation it should be checked that the end of file is not reached, i. e. that #File.EOF# is not equal to "X". The contents of the last line read can be accessed using #File.Line#, its line number being #File.Line.Number# .
Example	If #File.EOF# <> X File.GetLine SplitString #File.Line#, ; , %DESC_EN%, %DESC_DE% Smarteam.SetData CN_DESCRIPTION_EN, %DESC_EN%, CN_DESCRIPTION_DE, %DESC_DE% Endif

Name	File.GetList
Arguments	<Directory> ¹ , <Pattern> ²
Description	Create a list of all files in <Directory> that match <Pattern>.
Remarks	The "Loop File.List"/"Next File.List" statements can be used to loop at the list.
Example	File.GetList C:\Temp, ITEM*.txt Loop File.List File.OpenForInput #File.List.Directory#\#File.List.Filename# File.GetLine ... Next File.List

Name	File.Move
Arguments	<Source> ¹ , <Target> ²
Description	The file given as <Source> is moved on the file given as <Target>.
Remarks	Both paths have to be fully qualified.
Example	File.Move #Smarteam.DIRECTORY#\#Smarteam.FILE_NAME#,_

	#SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME#.save
--	--

Name	File.OpenForAppend
Arguments	<Name>
Description	The file given as <Name> is opened for append. Functions File.PutLine and File.Close refer to this file without mentioning the filename again.
Remarks	The variable #System.TemporaryFilename# can be used to generate a temporary filename (without path and extension) which is unique during the entire action sequence.
Example	File.OpenForAppend C:\\Temp\\#System.TemporaryFilename#.txt

Name	File.OpenForInput
Arguments	<Name> ¹ , <CommentCharacter> ²
Description	The file given as <Name> is opened for input. Functions File.GetLine and File.Close refer to this file without mentioning the filename again. If <CommentCharacter> is given, all input lines having <CommentCharacter> as first character will be discarded while reading.
Remarks	
Example	File.OpenForInput C:\\Temp\\ExternalData.txt, \\#

Name	File.OpenForOutput
Arguments	<Name>
Description	The file given as <Name> is opened for output and information contained in the file is overwritten. Functions File.PutLine and File.Close refer to the file without mentioning the filename again.
Remarks	The variable #System.TemporaryFilename# can be used to generate a temporary filename (without path and extension) which is unique during the entire action sequence.
Example	File.OpenForOutput C:\\Temp\\#System.TemporaryFilename#.txt

Name	File.PutLine
Arguments	<Line>
Description	The line argument is written as single line to the file which was opened by the most recent call to File.OpenForAppend or File.OpenForOutput .
Remarks	Commas have to be masked by the escape character "\\" (backslash).
Example	File.PutLine MaterialNumber = #Material.Material#, MaterialType = #Material.MaterialType#

Name	File.WaitForTask
Arguments	None
Description	Executing is suspended until the process launched by File.Execute has terminated.
Remarks	
Example	File.WaitForTask

Function Module Functions

Name	FunctionModule.AddChange
Arguments	<Name> ¹ , <Type (Char, Int, Float)> ² , <Length> ³ , <Value> ⁴
Description	Adds a changing parameter of type <Type> with name <Name> and value length <Length> to internal memory for a later to FunctionModul.Call.
Remarks	Follow this link to see how the values of function call parameters can be accessed.
Example	Like FunctionModule.AddExport ; see FunctionModule.Call

Name	FunctionModule.AddExport
Arguments	<Name> ¹ , <Type (Char, Int, Float)> ² , <Length> ³ , <Value> ⁴
Description	Adds an export parameter of type <Type> with name <Name> and value length <Length> to internal memory for a later to FunctionModul.Call.
Remarks	Follow this link to see how the values of function call parameters can be accessed.
Example	See FunctionModule.Call

Name	FunctionModule.AddImport
Arguments	<Name> ¹ , <Type (Char, Int, Float)> ² , <Length> ³
Description	Adds an import parameter of type <Type> with name <Name> and value length <Length> to internal memory for a later to FunctionModul.Call.
Remarks	Follow this link to see how the values of function call parameters can be accessed.
Example	See FunctionModule.Call

Name	FunctionModule.AddTable
Arguments	<Name> ¹ , <Type (Char)> ² , <Length> ³
Description	Adds a table parameter of type "Char" with name <Name> and value length <Length> to internal memory for a later to FunctionModule.Call.

Remarks	Only type "Char" is supported for table parameters. Follow this link to see how the values of function call parameters can be accessed.
Example	See FunctionModule.Call

Name	FunctionModule.AddTableLine
Arguments	<Name> ¹ , <Value> ²
Description	Appends a line to table <Name> in internal memory for a later to FunctionModul.Call.
Remarks	Follow this link to see how the values of function call parameters can be accessed.
Example	<pre> ... FunctionModule.AddTable DOCUMENTDESCRIPTIONS, Char, 45 FunctionModule.AddTableLine DOCUMENTDESCRIPTIONS, " D #Smarteam.CN_DESCRIPTION_DE" FunctionModule.AddTableLine DOCUMENTDESCRIPTIONS, " E #Smarteam.CN_DESCRIPTION_EN" ... FunctionModule.Call BAPI_DOCUMENT_CHANGE X, X See also FunctionModule.Call </pre>

Name	FunctionModule.Call
Arguments	<Name> ¹ , <CommitRollback> ² , <Wait> ³
Description	Calls function module <Name> taking export, import, change and table parameters from internal memory. If <CommitRollback> is set to "X", a call to BapiTransactionCommit is issued if the function call was successful, and a BapiTransactionRollback is issued if the function called was not successful. If <CommitRollback> is not set or set to any other value, no commit or rollback actions are performed. If <Wait> is equal to "X", execution is resumed until the commit is performed, while otherwise execution continues immediately.
Remarks	Follow this link to see how the values of function call parameters can be accessed.
Example	<pre> FunctionModule.AddExport DOCUMENTNUMBER, Char, 25, #Smarteam.CN_ID# FunctionModule.AddExport DOCUMENTTYPE, Char, 3, DRW FunctionModule.AddExport DOCUMENTPART, Char, 3, 000 FunctionModule.AddExport DOCUMENTVERSION, Char, 2, 00 FunctionModule.AddExport GETDESCRIPTIONS, Char, 1, X FunctionModule.AddExport GETLONGTEXTS, Char, 1, " " FunctionModule.AddExport GETOBJECTLINKS, Char, 1, " " FunctionModule.AddExport GETLINKDESCRIPTIONS, Char, 1, " " FunctionModule.AddExport GETSTATUSLOG, Char, 1, " " FunctionModule.AddImport DOCUMENTDATA, Char, 1364 FunctionModule.AddImport RETURN, Char, 548 FunctionModule.AddTable DOCUMENTDESCRIPTIONS, Char, 45 FunctionModule.Call BAPI_DOCUMENT_GETDETAIL If #FunctionModule.RfcRc# <> 0 Message E, #FunctionModule.ErrorInfo# Else Loop FunctionModule.DOCUMENTDESCRIPTIONS Message \Left{40\Mid5{#FunctionModule.DOCUMENTDESCRIPTIONS#}} </pre>

	Next FunctionModule.DOCUMENTDESCRIPTIONS Endif
--	---

Name	FunctionModule.Clear
Arguments	None
Description	Clears the internal memory for function calls.
Remarks	
Example	FunctionModule.Clear

Name	FunctionModule.ReadTable
Arguments	<TableName> ¹ , <CompareString 1> ² , <CompareString 2> ³ , <CompareString 3> ⁴ , ...
Description	Searches table <TableName> for an entry matching any of the given compare strings. If a table line matching a compare string is found, the table's loop counter is set to that row so that the value of the table line can be accessed.
Remarks	Compare strings can contain wildcard characters "*" and "+".
Example	FunctionModule.AddTable LABORATORIES, Char, 33 FunctionModule.Call C_PDM_GET_LABORATORIES FunctionModule.ReadTable LABORATORIES, KB2* Message I, \Mid4{#FunctionModule.LABORATORIES#}

Grid Functions

Name	Grid.AddLine
Arguments	<Cell value column 1>, <Cell value column 2>, <Cell value column 3>, ...
Description	Adds a line to the grid appending it as the last line. If more values are given than there are columns in the grid, the grid's column count is extended. If less values are passed, the remaining cells are left blank.
Remarks	A "\n" in a cell value will result in a line break. For displaying multiple lines in a cell correctly, parameter <WordWrap> of function Grid.SetProperties should be set to "X".
Example	Grid.AddLine Field Name, Field Value Grid.AddLine Material Number, #Material.Material# Grid.AddLine Material Description, #Material.BasicData.MatDesc#

Name	Grid.SetColFontBold
Arguments	<Column index 1>, <Column index 2>, <Column index 3>, ...
Description	Changes the font type of the columns with the given indices to Bold. If an invalid

	column index is given, it is ignored without a message.
Remarks	
Example	Grid.SetColFontBold 1

Name	Grid.SetColFontItalic
Arguments	<Column index 1>, <Column index 2>, <Column index 3>, ...
Description	Changes the font type of the columns with the given indices to Italic. If an invalid column index is given, it is ignored without a message.
Remarks	
Example	Grid.SetColFontItalic 1

Name	Grid.SetColWidth
Arguments	<Width of column 1>, <Width of column 2>, <Width of column 3>, ...
Description	Specifies the width of the grid columns. If more values are given than there are columns in the grid, superfluous values are ignored.
Remarks	The unit of the width values is a "Twip".
Example	Grid.SetColWidth 1500, 3000, 3000

Name	Grid.SetProperties
Arguments	<Height> ¹ , <Width> ² , <FixedCols> ³ , <FixedRows> ⁴ , <WordWrap> ⁵
Description	Sets some of the window and grid properties. <Height> and <Width> are the height and width of the entire window. <FixedCols> and <FixedRows> specify how many columns and rows of the grid are fixed (meaning that they do not scroll up and down or left and right). Setting <WordWrap> to "X" enables word wrapping for cell values.
Remarks	Usually column and row titles should be fixed. The unit of the height and width values is a "Twip".
Example	Grid.SetProperties 9000, 7000, 1, 1, X

Name	Grid.SetRowFontBold
Arguments	<Row index 1>, <Row index 2>, <Row index 3>, ...
Description	Changes the font type of the rows with the given indices to Bold. If an invalid row index is given, it is ignored without a message.
Remarks	
Example	Grid.SetRowFontBold 1

Name	Grid.SetRowFontItalic
Arguments	<Row index 1>, <Row index 2>, <Row index 3>, ...
Description	Changes the font type of the rows with the given indices to Italic. If an invalid row index is given, it is ignored without a message.
Remarks	
Example	Grid.SetRowFontItalic 1

Name	Grid.Show
Arguments	<Title> ¹ , <Text> ² , <DoubleClickOK> ³ , <DoubleClickEdit> ⁴ , <ShowSaveMenu> ⁵ , <ShowClipboardMenu> ⁶
Description	Display the grid that was filled and modified before by other grid functions. <Title> is the window title and <Text> is the text displayed above the grid. It can be at most 3 lines. If <DoubleClickOK> is set to "X", a double click on the grid will be interpreted as a selection and the window will be closed. If <DoubleClickEdit> is "X", a double click on a cell makes its value editable. If <ShowSaveMenu> is equal to "X", a Save menu option is displayed in the window's menu bar allowing for saving the entire grid. Setting <ShowClipboardMenu> to "X" will activate the "Copy to Clipboard" menu option allowing for copying the entire grid to the clipboard.
Remarks	A "\n" in <Text> will result in a line break.
Example	Grid.Show SAP System: #SAP.Destination#, Details of Material #Material.Material#, , X, X, X

Name	Grid.Sort
Arguments	<Column index> ¹ , <Direction (Ascending, Descending)> ²
Description	The grid will be sorted in ascending or descending order using column with index <Column index> as key.
Remarks	Before sorting the grid, rows containing column titles should be fixed using function Grid.SetProperties . Fixed rows will be excluded from the sort process.
Example	Grid.Sort 1, Ascending

Material Functions

Name	Material.AddBasicData
Arguments	<Material> ¹ , <MaterialType> ² , <IndustrySector> ³ , <Plant> ⁴ , <BOMUsage> ⁵ , <BOMAlternative> ⁶ , <RevisionLevel> ⁷ , <ECNumber> ⁸ , <ValidFrom> ⁹ , <ValidTo> ¹⁰ , <ValuationArea> ¹¹ , <ValuationType> ¹²
Description	Adds a material's basic data to the internal memory for later calls to other material functions.
Remarks	For a long time, the material number in SAP was 18 characters long. With a setting in the General section of the configuration file, long material numbers with 40

	characters can be switched on. Usually, the material number in field <Material> is converted automatically into SAP internal representation. This conversion can be suppressed by using a setting in the General section of the configuration file.
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, FERT, M, , 2

Name	Material.AddBIData
Arguments	<Batch input program name> ¹ , <Batch input screen number> ² , <FieldName1> ³ , <FieldValue1> ⁴ , <FieldName2> ⁵ , <FieldValue2> ⁶ , ...
Description	Adds batch input records for creating or changing the material master in internal memory via batch input without dialog.
Remarks	For details on batch input techniques see SAP documentation.
Example	Material.AddData SAPLMGMM, 0060, RMMG1-MATNR, \Left18{#Document.Links.ObjectKey#} Material.AddData SAPLMGMM, 0070, MSICHTAUSW-KZSEL(01), X Material.AddData SAPLMGMM, 3005, MARA-ZEINR, #SmarTeam.CN_PART_NUMBER#,_ BDC_OKCODE, =BU

Name	Material.AddBOMData
Arguments	<ItemCategory> ¹ , <ItemNumber> ² , <Component> ³ , <ComponentQuantity> ⁴ , <ComponentUnit> ⁵ , <FixedQuantity> ⁶ , <ItemText1> ⁷ , <ItemText2> ⁸ , <SortString> ⁹ , <RelCost> ¹⁰ , <RelEngin> ¹¹ , <RelPmaint> ¹² , <RelProd> ¹³ , <RelSales> ¹⁴ , <SparePart> ¹⁵ , <MatProvis> ¹⁶ , <BulkMat> ¹⁷ , <RecAllowd> ¹⁸
Description	Adds a bill of material component to the internal memory for later calls to other material functions.
Remarks	The function sets the data of a single line of the bill of material. If you are creating a new bill of material, you have to append that line using Material.AppendBOMData .
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, , , 2 Material.DeleteBOM Loop SmarTeam.Structure Material.AddBOMData L, , #SmarTeam.CN_PART_NUMBER#, #SmarTeam.CN_QUANTITY#,_ #SmarTeam.CN_QUANTITYUNIT# Material.AppendBOMData Next SmarTeam.Structure Material.SaveBOM X, X

Name	Material.AddBOMData2
Arguments	<ItemCategory> ¹ , <ItemNumber> ² , <Component> ³ , <ComponentQuantity> ⁴ , <ComponentUnit> ⁵ , <FixedQuantity> ⁶ , <ItemText1> ⁷ , <ItemText2> ⁸ , <SortString> ⁹ , <RelCost> ¹⁰ , <RelEngin> ¹¹ , <RelPmaint> ¹² , <RelProd> ¹³ , <RelSales> ¹⁴ , <SparePart> ¹⁵ , <MatProvis> ¹⁶ , <BulkMat> ¹⁷ , <RecAllowd> ¹⁸ , <CadIndicator> ¹⁹ , <SubItemIndicator> ²⁰
Description	Adds a bill of material component to the internal memory for later calls to other

	material functions.
Remarks	This function works together with other BOM2 functions. It sets the data of a single line of the bill of material. If you are creating a new bill of material, you have to append that line using Material.AppendBOMData2 .
Example	<pre> M.AddBasicData #S.TDM_SAP_MAT_NUM# , , , 2 , , , , #System.Date# M.AddBOMHeaderData2 M.ReadBOM2 M.AddBOMHeaderData2 , , , , , , , X Loop Smarteam.Structure M.AddBOMData2 L , , #Smarteam.Structure.TDM_SAP_MAT_NUM# , _ #Smarteam.Structure.CN_QUANTITY# , ST , , , , , , , , , X M.AppendBOMData2 Next Smarteam.Structure M.SaveBOM2 X </pre>

Name	Material.AddBOMDataEx (Function is obsolete; please use Material.AddBOMDataExt)
Arguments	<Offset1>, <Value1>, <Offset2>, <Value2>, <Offset3>, <Value3>, . . .
Description	Set parts of the bill of material structure STPO_API03 which cannot be set using Material.AddBOMData. Each <Offset> of the structure is set to the corresponding <Value>.
Remarks	The function sets the data of one line of the bill of material. If you are creating a new bill of material you have to append that line using Material.AppendBOMData . Using this function might cause the need to adapt the configuration after an R/3 release upgrade or after a modification of the R/3 customizing because the layout of the BOM API structures might have changed.
Example	<pre> Material.AddBasicData #Smarteam.CN_PART_NUMBER# , , , 2 Material.OpenBOM Loop Material.BOM , %LC% If #Material.BOM.ItemNo# >= 0500 Material.SelectBOMItem %LC% Material.AddBOMDataEx 625 , X Material.ModifyBOMItem Endif Next Material.BOM Material.CloseBOM </pre>

Name	Material.AddBOMDataExt
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Sets fields of the bill of material structure STPO_API03 which cannot be set using Material.AddBOMData.
Remarks	The function sets the data of one line of the bill of material. If you are creating a new bill of material you have to append that line using Material.AppendBOMData . Fieldnames are not case-sensitive. All fields of SAP structure STPO_API03 can be taken as <Fieldname>.

Example	Material.AddBOMDataExt FIDelete:=X
---------	------------------------------------

Name	Material.AddBOMDataExt2
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Sets fields of the bill of material structure CSRFCITEM which cannot be set using Material.AddBOMData2 .
Remarks	This function works together with other BOM2 functions. It sets the data of one line of the bill of material. If you are creating a new bill of material you have to append that line using Material.AppendBOMData2 . Fieldnames are not case-sensitive. All fields of SAP structure CSRFCITEM can be taken as <Fieldname>.
Example	Material.AddBOMDataExt2 Lgort:=0001

Name	Material.AddBOMHeaderData
Arguments	<BaseQuantity> ¹ , <BaseUnit> ² , <BomStatus> ³ , <BomGroup> ⁴ , <BomText> ⁵ , <AltText> ⁶ , <Laboratory> ⁷ , <AuthGroup> ⁸
Description	Adds the header of a bill of material to the internal memory for later calls to other material functions.
Remarks	
Example	Material.AddBOMHeaderData , , 10, , #SmarTeam.CN_CHANGETEXT#,_ #SmarTeam.CN_CHANGETEXT2#

Name	Material.AddBOMHeaderData2
Arguments	<BaseQuantity> ¹ , <BaseUnit> ² , <BomStatus> ³ , <BomGroup> ⁴ , <BomText> ⁵ , <AltText> ⁶ , <Laboratory> ⁷ , <AuthGroup> ⁸ , <CadIndicator> ⁹
Description	Adds the header of a bill of material to the internal memory for later calls to other material functions. Other BOM header values like material number, plant, EC number, etc. are also set automatically using the material's basic data values. So even if none of the arguments is to be set, this function must be called in order to execute BOM2 functions.
Remarks	This function works together with other BOM2 functions.
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, FERT, M, , 2 Material.AddBOMHeaderData2 , , 10, , #SmarTeam.CN_CHANGETEXT#,_ #SmarTeam.CN_CHANGETEXT2#, , , X

Name	Material.AddBOMHeaderDataExt
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...

Description	Sets fields of the bill of material header structure STKO_API01 which cannot be set using Material.AddBOMHeaderData .
Remarks	This function works together with other BOM functions. Fieldnames are not case-sensitive. All fields of SAP structure STKO_API01 can be taken as <Fieldname>.
Example	Material.AddBOMHeaderDataExt ZCustomerField:=#SmarTeam.CN_SECURITY_FLAG#

Name	Material.AddBOMHeaderDataExt2
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Sets fields of the bill of material header structure BICSK which cannot be set using Material.AddBOMHeaderData2 .
Remarks	This function works together with other BOM2 functions. Fieldnames are not case-sensitive. All fields of SAP structure BICSK can be taken as <Fieldname>.
Example	Material.AddBOMHeaderDataExt2 ZCustomerField:=#SmarTeam.CN_SECURITY_FLAG#

Name	Material.AddBOMSubItemData
Arguments	<BomItemCounter> ¹ , <SubItemNumber> ² , <InstallationPoint> ³ , <SubItemQuantity> ⁴ , <SubItemText> ⁵
Description	Adds sub items of a bill of material component to the internal memory for later calls to other material functions. If <BomItemCounter> is empty, the functions takes the last BOM component added as master for the sub item. If <SubItemNumber> is empty, the function adds 1 for each new sub item, starting at 1 for each new BOM component.
Remarks	This function works together with other BOM functions.
Example	

Name	Material.AddBOMSubItemData2
Arguments	<BomItemCounter> ¹ , <SubItemNumber> ² , <InstallationPoint> ³ , <SubItemQuantity> ⁴ , <SubItemText> ⁵
Description	Adds sub items of a bill of material component to the internal memory for later calls to other material functions. If <BomItemCounter> is empty, the functions takes the last BOM component added as master for the sub item. If <SubItemNumber> is empty, the function adds 1 for each new sub item, starting at 1 for each new BOM component.
Remarks	This function works together with other BOM2 functions.
Example	

Name	Material.AddClientData
------	-------------------------------

Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's client data to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: DelFlag, MatlGroup, OldMatNo, BaseUom, BaseUomIso, PoUnit, PoUnitIso, Document, DocType, DocVers, DocFormat, DocChgNo, PageNo, NoSheets, ProdMemo, Pageformat, SizeDim, BasicMatl, StdDescr, DsnOffice, PurValkey, NetWeight, UnitOfWt, UnitOfWtIso, Container, StorConds, TempConds, TransGrp, HazMatNo, Division, Competitor, QtyGrGi, ProcRule, SupSource, Season, LabelType, LabelForm, ProdHier, CadId, AllowedWt, PackWtUn, PackWtUnIso, AllwdVol, PackVoUn, PackVoUnIso, WtTolLt, VolTolLt, VarOrdUn, BatchMgmt, ShMatTyp, FillLevel, StackFact, MatGrpSm, Authoritygroup, QmProcmnt, Catprofile, Minremlife, ShelfLife, StorPct, PurStatus, SalStatus, Pvalidfrom, Svalidfrom, Envtrlvt, ProdAlloc, QualDik, ManuMat, MfrNo, InvMatNo, ManufProf, Hazmatprof, HighVisc, Looseorliq, ClosedBox, AppdBRec, Matcmpllvl, ParEff, RoundUpRuleExpirationDate, PeriodIndExpirationDate, ProdCompositionOnPackaging, ItemCat, HazMatNoExternal, HazMatNoGuid, HazMatNoVersion, InvMatNoExternal, InvMatNoGuid, InvMatNoVersion, MaterialFixed, CmRelevanceFlag
Example	Material.AddClientData MatlGroup:=001, Document:=#SmarTeam.CN_ID#, DocType:=VAB, DocVers:=#SmarTeam.REVISION#, PageNo:=000

Name	Material.AddDescription
Arguments	<Language> ¹ , <LanguageIso> ² , <Description> ³ , <DeleteFlag> ⁴
Description	Adds a material's description to the internal memory for a later call to function Material.Save.
Remarks	
Example	Material.AddDescription D, , #SmarTeam.CN_DESCRIPTION_DE# Material.AddDescription E, , #SmarTeam.CN_DESCRIPTION_EN#

Name	Material.AddExtension
Arguments	<Structure> ¹ , <StructureX> ² , <Valuepart1> ³ , <Valuepart1X> ⁴ , <Valuepart2> ⁵ , <Valuepart2X> ⁶ , <Valuepart3> ⁷ , <Valuepart3X> ⁸ , <Valuepart4> ⁹ , <Valuepart4X> ¹⁰
Description	Adds values for material master table extensions to internal memory for a later call to function Material.Save.
Remarks	
Example	

Name	Material.AddForecastParameters
------	---------------------------------------

Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's forecast parameters to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: Plant, ForeProf, ModelSi, ModelSp, ParamOpt, OptimLev, Initialize, ForeModel, AlphaFact, BetaFact, GammaFact, DeltaFact, Tracklimit, ForeDate, HistVals, InitPds, SeasonPds, ExpostPds, ForePds, FixedPds, WtgGroup
Example	Material.AddForecastParameters Plant:=1200

Name	Material.AddHeadData
Arguments	<Material> ¹ , <MatlType> ² , <IndSector> ³ , <BasicView> ⁴ , <SalesView> ⁵ , <PurchaseView> ⁶ , <MrpView> ⁷ , <ForecastView> ⁸ , <WorkSchedView> ⁹ , <PrtView> ¹⁰ , <StorageView> ¹¹ , <WarehouseView> ¹² , <QualityView> ¹³ , <AccountView> ¹⁴ , <CostView> ¹⁵ , <InpFldCheck> ¹⁶ , <MaterialExternal> ¹⁷ , <MaterialGuid> ¹⁸ , <MaterialVersion> ¹⁹
Description	Adds a material's head data to the internal memory for a later call to function Material.Save.
Remarks	For a long time, the material number in SAP was 18 characters long. With a setting in the General section of the configuration file, long material numbers with 40 characters can be switched on. Usually, the material number in field <Material> is converted automatically into SAP internal representation. This conversion can be suppressed by using a setting in the General section of the configuration file.
Example	Material.AddHeadData #SmarTeam.CN_PART_NUMBER#, HALB, M

Name	Material.AddLongtext
Arguments	<ApplObject> ¹ , <TextName> ² , <TextId> ³ , <Langu> ⁴ , <LanguIso> ⁵ , <Format> ⁶ , <TextLine> ⁷ , <DelFlag> ⁸
Description	Adds a material's longtext to the internal memory for a later call to function Material.Save.
Remarks	
Example	Material.AddLongtext Material, %MATNR%, IVER, E, , , Line 1 Material.AddLongtext Material, %MATNR%, IVER, E, , /, Line 2 Material.AddLongtext Material, %MATNR%, IVER, E, , /, Line 3

Name	Material.AddPlantData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...

Description	Adds a material's plant data to the internal memory for a later call to function Material.Save . In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: Plant, DelFlag, AbcId, CritPart, PurGroup, IssueUnit, IssueUnitIso, MrpProfile, MrpType, MrpCtrlr, PlndDelry, GrPrTime, PeriodInd, AssyScrap, LotSizeKey, ProcType, SpProcType, ReorderPt, SafetyStk, MinLotSize, MaxLotSize, FixedLot, RoundVal, MaxStock, OrdCosts, DepReqId, StorCosts, AltBomId, Discontin, EffODay, FollowUp, GrpReqmts, MixedMrp, SmKey, Backflush, ProductionScheduler, ProcTime, SetupTime, Interop, BaseQty, Inhseprod, Stgeperiod, StgePdUn, StgePdUnIso, OverTol, Unlimited, UnderTol, Replentime, ReplacePt, IndPostToInspStock, CtrlKey, DocReqd, Loadinggrp, BatchMgmt, Quotausage, ServLevel, SplitInd, Availcheck, FyVariant, CorrFact, SetupTime2, BaseQtyPlan, ShipProcTime, SupSource, AutoPOrd, Sourcelist, CommCode, Countryori, CountryorIso, Regionorig, CommCoUn, CommCoUnIso, Expimpgrp, ProfitCtr, PpcPICal, RepManuf, PITiFnce, Consummode, BwdCons, FwdCons, AlternativeBom, BOMUsage, Planlistgrp, Planlistcnt, LotSize, Specprocty, ProdUnit, ProdUnitIso, IssStLoc, MrpGroup, CompScrap, CertType, CycleTime, Covprofile, CcPhInv, VarianceKey, SernoProf, Repmanprof, NegStocks, QmRgmts, PlngCycle, RoundProf, Refmatcons, DToRefM, MultRefM, AutoReset, ExCertId, ExCertNoNew, ExCertDt, MilitId, InspInt, CoProduct, PlanStrgp, SlocExprc, BulkMat, CcFixed, DetermGrp, QmAuthgrp, TaskListType, PurStatus, Prodprof, SaftyTId, Safetytime, PlordCtrl, Batchentry, Pvalidfrom, Matfrgtgrp, Prodverscs, MatCfop, EuListNo, EuMatGrp, CasNo, ProdcomNo, CtrlCode, JitRelvt, MatGrpTrans, HandlgGrp, SupplyArea, FairShareRule, PushDistrib, DeployHoriz, MinLotSize2, MaxLotSize2, FixLotSize2, LotIncrement, ProdConvType, DistrProf, PeriodProfileSafetyTime, FxdPrice, AvailCheckAllProjSegments, Overallprf, MrpRelevancyDepRequirements, MinSafetyStk, NoCosting, UnitGroup, FollowUpExternal, FollowUpGuid, FollowUpVersion, RefmatconsExternal, RefmatconsGuid, RefmatconsVersion
Example	Material.AddPlantData Plant:=1200, PurGroup:=002, ProcType:=E, MrpType:=PD, AvailCheck:=KP

Name	Material.AddSalesData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's sales data to the internal memory for a later call to function Material.Save . In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: SalesOrg, DistrChan, DelFlag, MatlStats, RebateGrp, CommGroup, CashDisc, SalStatus, ValidFrom, MinOrder, MinDely, MinMto, DelyUnit, DelyUom, DelyUomIso, SalesUnit, SalesUnitIso, ItemCat, DelygPlnt, ProdHier, PrRefMat, MatPrGrp, AcctAssgt, MatlGrp1, MatlGrp2, MatlGrp3, MatlGrp4, MatlGrp5, ProdAtt1, ProdAtt2, ProdAtt3, ProdAtt4, ProdAtt5, ProdAtt6, ProdAtt7, ProdAtt8, ProdAtt9, ProdAtt10, RoundProf, VarSalesUn, UnitGroup, PrRefMatExternal, PrRefMatGuid, PrRefMatVersion
Example	Material.AddSalesData DelygPlnt:=HOME, MatPrGrp:=E, AcctAssgt:=01, ItemCat:=ZHW3, PriceCtrl:=S

Name	Material.AddStorageLocationData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's storage location data to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: Plant, StgeLoc, DelFlag, MrpInd, SpecProc, ReorderPt, ReplQty, StgeBin, PickgArea, InvCorrFac
Example	Material.AddStorageLocationData

Name	Material.AddStorageTypeData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's storage type data to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: WhseNo, StgeType, DelFlag, StgeBin, MaxSbQty, MinSbQty, CtrlQty, ReplenQty, PickArea, RoundQty
Example	Material.AddStorageTypeData

Name	Material.AddTaxClassifications
Arguments	< DepCountry> ¹ , <DepCountryIso> ² , <TaxType1> ³ , <Taxclass1> ⁴ , <TaxType2> ⁵ , <Taxclass2> ⁶ , <TaxType3> ⁷ , <Taxclass3> ⁸ , <TaxType4> ⁹ , <Taxclass4> ¹⁰ , <TaxType5> ¹¹ , <Taxclass5> ¹² , <TaxType6> ¹³ , <Taxclass6> ¹⁴ , <TaxType7> ¹⁵ , <Taxclass7> ¹⁶ , <TaxType8> ¹⁷ , <Taxclass8> ¹⁸ , <TaxType9> ¹⁹ , <Taxclass9> ²⁰ , <TaxInd> ²¹
Description	Adds a material's tax classifications to the internal memory for a later call to function Material.Save.
Remarks	
Example	Material.AddTaxClassifications DE, , MWST, 1 Material.AddTaxClassifications US, , UTXJ, 1

Name	Material.AddUnitOfMeasure
Arguments	<AltUnit> ¹ , <AltUnitIso> ² , <Numerator> ³ , <Denominator> ⁴ , <EanUpc> ⁵ , <EanCat> ⁶ , <Length> ⁷ , <Width> ⁸ , <Height> ⁹ , <UnitDim> ¹⁰ , <UnitDimIso> ¹¹ , <Volume> ¹² , <Volumeunit> ¹³ , <VolumeunitIso> ¹⁴ , <GrossWt> ¹⁵ , <UnitOfWt> ¹⁶ , <UnitOfWtIso> ¹⁷ , <DelFlag> ¹⁸ , <SubUom> ¹⁹ , <SubUomIso> ²⁰
Description	Adds a material's unit of measure to the internal memory for a later call to function Material.Save.

Remarks	
Example	Material.AddUnitOfMeasure ST, , , , , , , , , #SmarTeam.CN_VOLUME#, CCM, , #SmarTeam.CN_WEIGHT#, KG

Name	Material.AddValuationData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's valuation data to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: ValArea, ValType, DelFlag, PriceCtrl, MovingPr, StdPrice, PriceUnit, ValClass, PrCtrlPp, MovPrPp, StdPrPp, PrUnitPp, VclassPp, PrCtrlPy, MovPrPy, StdPrPy, VclassPy, PrUnitPy, ValCat, FuturePr, ValidFrom, Taxprice1, Commprice1, Taxprice3, Commprice3, PlndPrice, Plndprice1, Plndprice2, Plndprice3, Plndprdate1, Plndprdate2, Plndprdate3, LifoFifo, Poolnumber, Taxprice2, Commprice2, Devallnd, OrigGroup, OverheadGrp, QtyStruct, MIAActive, MISettle, OrigMat, VmSoStk, VmPStock, MatlUsage, MatOrigin, InHouse, TaxCmlUn
Example	Material.AddValuationData ValArea:=%ValArea%, ValClass:=%ValClass%, PriceCtrl:=V, OrigGroup:=1501

Name	Material.AddWarehouseNumberData
Arguments	<Fieldname1>:=<Fieldvalue1>, <Fieldname2>:=<Fieldvalue2>, <Fieldname3>:=<Fieldvalue3>, ...
Description	Adds a material's warehouse number data to the internal memory for a later call to function Material.Save. In order to create a certain view of the material master, it is not sufficient to set field values here. In addition, the desired view must be selected by setting the appropriate view selector in function Material.AddHeadData .
Remarks	Fieldnames are not case-sensitive. Possible fieldnames are: WhseNo, DelFlag, Stgesector, Placement, Withdrawal, LEquip1, LEquip2, LEquip3, LeqUnit1, LeqUnit1Iso, LeqUnit2, LeqUnit2Iso, LeqUnit3, LeqUnit3Iso, Unittyp1, Unittyp2, Unittyp3, WmUnit, WmUnitIso, AddToStk, BlockStge, MsgTolm, SpecMvmt, CopyUsage, ProcureUn, ProcureUnIso, StgeType, RefUnit, TwoStepPick
Example	Material.AddWarehousenumberData

Name	Material.AppendBOMData
Arguments	None
Description	Appends the data of a single line of the bill of material to internal memory.
Remarks	This function does not set any data of the line of the bill of material to be appended. To set data use functions Material.AddBOMData and/or Material.AddBOMDataExt .

Example	See function Material.AddBOMData
---------	--

Name	Material.AppendBOMData2
Arguments	None
Description	Appends the data of a single line of the bill of material to internal memory.
Remarks	This function works together with other BOM2 functions. It does not set any data of the line of the bill of material to be appended. To set data use functions Material.AddBOMData2 and/or Material.AddBOMDataExt2 .
Example	See function Material.AddBOMData2

Name	Material.ChangeDialog
Arguments	None
Description	Calls RFC_CHANGE_MATERIAL_MASTER with data taken from internal memory. I. e. the material master in memory can be modified using SAP's dialog transaction.
Remarks	
Example	Material.ChangeDialog

Name	Material.Clear
Arguments	None
Description	Clears the internal memory for material masters.
Remarks	To clear specific parts of the internal material memory use functions Material.ClearBasicData Material.ClearBIData Material.ClearBOMData Material.ClearBOMData2 Material.ClearClientData Material.ClearDescriptions Material.ClearExtensions Material.ClearForecastParameters Material.ClearHeadData Material.ClearLongtext Material.ClearPlantData Material.ClearSalesData Material.ClearStorageLocationData Material.ClearStorageTypeData Material.ClearTaxClassifications Material.ClearUnitsOfMeasure Material.ClearValuationData Material.ClearWarehousenumberData
Example	Material.Clear

Name	Material.CloseBOM
Arguments	None
Description	Calls CSAP_MAT_BOM_CLOSE closing an open bill of material in the SAP system and committing all changes. The BOM identification is taken from internal memory.
Remarks	
Example	See function Material.AddBOMDataEx

Name	Material.CreateDialog
Arguments	None
Description	Calls RFC_CREATE_MATERIAL_MASTER with data taken from internal memory. I. e. the material master in memory can be created using SAP's dialog transaction.
Remarks	
Example	Material.CreateDialog

Name	Material.CreateRevision
Arguments	None
Description	Calls CCAP_REV_LEVEL_MAINTAIN with data taken from internal memory to create a new revision of a material master.
Remarks	Another option is using Batch Input technique .
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, , , , , #SmarTeam.REVISION#, _ #SmarTeam.CN_ECNUMBER# Material.CreateRevision

Name	Material.DeleteBOM
Arguments	None
Description	Calls CSAP_MAT_BOM_DELETE deleting a bill of material in the SAP system. The BOM identification is taken from internal memory.
Remarks	
Example	See function Material.AddBOMData

Name	Material.Display
Arguments	None
Description	Calls RFC_DISPLAY_MATERIAL_MASTER with data taken from internal memory. I. e. the material master in memory can be viewed using SAP's dialog transaction.

Remarks	
Example	Material.Display

Name	Material.DisplayProductStructure
Arguments	None
Description	Calls RFC_EXPLODE_PRODUCT_STRUCTURE with material number taken from internal memory. I. e. SAP's product structure browser will be displayed for the material master in memory.
Remarks	
Example	Material.DisplayProductStructure

Name	Material.ExistenceCheck
Arguments	None
Description	Calls BAPI_MATERIAL_EXISTENCECHECK with the material number taken from internal memory.
Remarks	If the material exists, #Material.Exists# is equal to "X" and " " otherwise. If the material is marked for deletion, #Material.IsDeleted# is equal to "X" and " " otherwise.
Example	Material.ExistenceCheck

Name	Material.GetDetail
Arguments	None
Description	Calls BAPI_MATERIAL_GET_DETAIL with the material's key data taken from internal memory.
Remarks	
Example	Material.GetDetail

Name	Material.GetInternalNumber
Arguments	None
Description	Calls BAPI_MATERIAL_GETINTNUMBER with the material's type and industry sector data taken from internal memory.
Remarks	The material number reserved can be accessed through #Material.Material#
Example	Material.AddBasicData , HALB, M Material.GetInternalNumber SmarTeam.SetData CN_PART_NUMBER, #Material.Material#

Name	Material.GetInternalNumber2
Arguments	None
Description	Calls BAPI_STDMATERIAL_GETINTNUMBER with the material's type and industry sector data taken from internal memory.
Remarks	The material number reserved can be accessed through #Material.Material#
Example	Material.AddBasicData , HALB, M Material.GetInternalNumber2 Smarteam.SetData CN_PART_NUMBER, #Material.Material#

Name	Material.GetList
Arguments	<MatNrSelection> ¹ , <MaterialShortDescSel> ² , <PlantSelection> ³ , <StorageLocationSelect> ⁴ , <SalesOrganisationSelection> ⁵ , <DistributionChannelSelection> ⁶ , <ManufacturerPartNum> ⁷ , <MaxRows> ⁸
Description	Calls BAPI_MATERIAL_GETLIST with the search criteria specified and returns a list of materials satisfying the given criteria into internal memory.
Remarks	Looping on the resulting list of materials is provided by the "Loop Material.List"/"Next Material.List" statements.
Example	Material.GetList 40*, *Bulb*, , , , , 10 Loop Material.List ... Next Material.List Material.GetList from: 47* to: 48* Material.GetList not 40* Material.GetList not from: 47* to: 48*

Name	Material.ModifyBOMItem
Arguments	<Index>
Description	Calls CSAP_BOM_ITEM_MAINTAIN modifying a single line of a bill of material. The BOM identification is taken from internal memory. If <Index> is specified, BOM item number <Index> in internal memory is modified according to the function result.
Remarks	The BOM must have been opened using Material.OpenBOM and a single line must have been selected using Material.SelectBOMItem . Before applying Material.ModifyBOMItem, the selected item can be modified using Material.AddBOMData and/or Material.AddBOMDataEx .
Example	See function Material.AddBOMDataEx

Name	Material.OpenBOM
Arguments	None

Description	Calls CSAP_MAT_BOM_OPEN opening a bill of material for modification in the SAP system. The BOM identification is taken from internal memory.
Remarks	If the BOM exists, #Material.BOMExists# is equal to "X" and " " otherwise. In addition, the bill of material data is read into internal memory.
Example	See function Material.AddBOMDataEx

Name	Material.ReadBOM
Arguments	None
Description	Calls CSAP_MAT_BOM_READ reading a bill of material data from the SAP system. The BOM identification is taken from internal memory.
Remarks	If the BOM exists, #Material.BOMExists# is equal to "X" and " " otherwise.
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, , , 2, , , #System.Date#, 31.12.9999 Material.ReadBOM

Name	Material.ReadBOM2
Arguments	<DisplayFlag>
Description	Calls CAD_DISPLAY_BOM_WITH_SUBITEMS reading a bill of material data from the SAP system. The BOM identification is taken from internal memory and has to be set via Material.AddBasicData in conjunction with Material.AddBOMHeaderData2 . <DisplayFlag> must be usually left blank.
Remarks	If the BOM exists, #Material.BOMExists# is equal to "X" and " " otherwise.
Example	Material.AddBasicData #SmarTeam.CN_PART_NUMBER#, , , 2, , , #System.Date# Material.AddBOMHeaderData2 Material.ReadBOM2

Name	Material.Save
Arguments	None
Description	Calls BAPI_MATERIAL_SAVEDATA with the data stored in internal memory.
Remarks	
Example	Material.Save

Name	Material.SaveBI
Arguments	<CallMode> ¹ , <UpdateMode> ²
Description	Creates or changes the material master in internal memory via batch input without dialog.
Remarks	Before calling Material.Save, batch input records must be added using

	Material.AddData. Using this function might cause the need to adapt the configuration after an R/3 release upgrade or after a modification of the R/3 customizing because screen numbers and data fields available in screens might have changed.
Example	Material.SaveBIN, S

Name	Material.SaveBOM
Arguments	<BomCreate> ¹ , <NewItem> ² , <Complete> ³ , <CadIndicator> ⁴
Description	Calls CSAP_MAT_BOM_MAINTAIN and creates or changes the bill of material stored in internal memory. If <BomCreate> is set to "X", the BOM is created if it does not already exist. If <NewItem> is set to "X", all components stored in internal memory are added to the BOM as new items. And if <Complete> is equal "X", the systems considers the given BOM to be the complete BOM. Cad indicators on BOM items will be considered only if <CadIndicator> is set to "X".
Remarks	
Example	See function Material.AddBOMData

Name	Material.SaveBOM2
Arguments	<AutoPosNr>
Description	Calls CAD_CREATE_BOM_WITH_SUBITEMS or CAD_CHANGE_BOM_WITH_SUBITEMS in order to create or change the bill of material stored in internal memory. If <AutoPosNr> is set to "X", positioning numbers are created automatically.
Remarks	
Example	See function Material.AddBOMData2

Name	Material.SelectBOMItem
Arguments	<Index>
Description	If a bill of material is stored in internal memory, a single line of that bill of material can be activated for modification using Material.SelectBOMItem. <Index> is the line number of the desired line. Counting starts at 1, i. e. the first line of the bill of material has index 1.
Remarks	
Example	See function Material.AddBOMDataEx

Name	Material.SelectViaMatchcode
Arguments	None

Description	Calls RFC_SELECT_OBJ_VIA_MATCHCODE for materials allowing the user to select a material number using SAP's dialog matchcode selection.
Remarks	The material number selected by the user can be passed to ENOVIA SmarTeam using the Smarteam.SetData function.
Example	Material.SelectViaMatchcode Check #SAP.DialogStatus# <> A SmarTeam.SetData TDM_SAP_MAT_NUM, #Material.Material#

Name	Material.UpdateBOM
Arguments	<BomCreate> ¹ , <NewItem> ² , <Complete> ³ , <CadIndicator> ⁴ , <IdentifyAttributes> ⁵ , <UpdateAttributes> ⁶ , <PreserveAttributes> ⁷ , <UpdateConditions> ⁸
Description	This function is based on function modules CSAP_MAT_BOM_READ and CSAP_MAT_BOM_MAINTAIN and provides functionality to easily perform BOM updates while keeping detailed control on how the update is done. The first four parameters <BomCreate>, <NewItem>, <Complete> and <CadIndicator> are the same as described in Material.SaveBOM . The next four arguments <IdentifyAttributes>, <UpdateAttributes>, <PreserveAttributes> and <UpdateConditions> allow for controlling the update process in detail. For an explanation of these arguments see the complete example in sample configurations.
Remarks	
Example	See complete example in sample configurations.

Name	Material.UpdateBOMItem
Arguments	<UpdateAssignment> ¹ , <Condition1> ² , <Or Condition2> ³ , <Or Condition3> ⁴ , ...
Description	This function can only be called inside a "Loop Material.BOM/Next Material.BOM" loop. Inside such a loop it checks whether any of the given <Conditions> is true for the current table line. If so, it updates this line according to <UpdateAssignment> which must look like <Fieldname>:=<Fieldvalue>. A <Condition> will usually look like #Material.BOM.<Fieldname># = <Fieldvalue>.
Remarks	All <Fieldnames> of structure STPO_API03 can be used in conditions and in update assignment.
Example	Loop Material.BOM Material.UpdateBOMItem FIDelete:=X, #Material.BOM.CAD_IND# = X Next Material.BOM

Project Functions

Name	Project.AddBasicData
Arguments	<ProjectDefinition> ¹ , <Description> ² , <ProjectProfile> ³ , <NetworkProfile> ⁴ , <BudgetProfile> ⁵ , <IntProfile> ⁶ , <WbsSchedProfile> ⁷ , <CshBdgtProfile> ⁸ , <PlanProfile> ⁹ , <MaskId> ¹⁰ , <ResponsibleNo> ¹¹ , <ApplicantNo> ¹² , <CompCode> ¹³ ,

	<code><BusArea>¹⁴, <ControllingArea>¹⁵, <ProfitCtr>¹⁶, <ProjectCurrency>¹⁷, <ProjectCurrencyIso>¹⁸, <Start>¹⁹, <Finish>²⁰, <Plant>²¹, <Calendar>²², <PlanBasic>²³, <PlanFcst>²⁴, <TimeUnit>²⁵, <TimeUnitIso>²⁶, <ProjectStock>²⁷, <NetworkAssignment>²⁸, <Objectclass>²⁹, <Statistical>³⁰, <Taxjurcode>³¹, <JointVenture>³², <RecoveryInd>³³, <EquityType>³⁴, <JvObjectType>³⁵, <JvJibClass>³⁶, <JvJibSubClassA>³⁷, <ObjectclassExt>³⁸, <FuncArea>³⁹, <DeletionFlag>⁴⁰</code>
Description	Adds a projects's basic (definition) data to the internal memory for later calls to other project functions.
Remarks	
Example	Project.AddBasicData #SmarTeam.CN_PROJECT_ID#, #SmarTeam.CN_PROJECT_DESCRIPTION#,_0001

Name	Project.Clear
Arguments	None
Description	Clears the internal memory for projects.
Remarks	To clear specific parts of the internal project memory use function Project.ClearBasicData
Example	Project.Clear

Name	Project.Create
Arguments	None
Description	Calls BAPI_PROJECTDEF_CREATE with data taken from internal memory.
Remarks	
Example	Project.Create

Name	Project.ExistenceCheck
Arguments	None
Description	Calls BAPI_PROJECTDEF_EXISTENCECHECK with the project number taken from internal memory.
Remarks	If the project definition exists, #Project.Exists# is equal to "X" and " " otherwise.
Example	Project.ExistenceCheck

Name	Project.GetDetail
Arguments	None
Description	Calls BAPI_PROJECTDEF_GETDETAIL with the project number taken from internal

	memory.
Remarks	
Example	Project.GetDetail

Name	Project.GetList
Arguments	<ProjectDefinitionRange> ¹ , <ProjectDescriptionRange> ² , <MaxRows> ³
Description	Calls BAPI_PROJECTDEF_GETLIST with the search criteria specified and returns a list of projects satisfying the given criteria into internal memory.
Remarks	Looping on the resulting list of projects is provided by the "Loop Project.List"/"Next Project.List" statements.
Example	Project.GetList *P, , 15 Loop Project.List ... Next Project.List Project.GetList from: 47* to: 48*

Name	Project.Update
Arguments	None
Description	Calls BAPI_PROJECTDEF_UPDATE with data taken from internal memory.
Remarks	
Example	Project.Update

Smarteam Functions

Name	Smarteam.CopyFile
Arguments	<Directory> ¹ , <Filename> ² , <Permission> ³ , <ExecuteOnTree> ⁴
Description	Copies the ENOVIA SmarTeam object's file from vault to the given filename in the given directory. Permission specifies the permission which will be set for the file and can have values "ReadOnly" or "ReadWrite". If <ExecuteOnTree> is equal to "X", the copy operation is executed on the entire object tree.
Remarks	
Example	Smarteam.CopyFile C:\\Temp, tmp_#Smarteam.FILE_NAME#, ReadOnly Smarteam.CopyFile C:\\Temp, tmp_#Smarteam.FILE_NAME#, , X

Name	Smarteam.CopyFileExt
Arguments	<Directory> ¹ , <Filename> ² , <SourceVaultID> ³ , <SourceDirectory> ⁴ ,

	<SourceFileName> ⁵
Description	Copies a file from vault to the given filename in the given directory. If no <SourceVaultID> is given, the vault object ID from the current ENOVIA SmarTeam object is taken. If no <SourceDirectory> is given, the (vault) directory from the current ENOVIA SmarTeam object is taken. And if no <SourceFileName> is given, the (vault) filename from the current ENOVIA SmarTeam object is taken.
Remarks	
Example	Smarteam.CopyFileExt C:\\Temp, tmp_file.tif, , , #Smarteam.FILE_NAME#.tif

Name	Smarteam.CreateReportObject
Arguments	<ClassNameOrID> ¹ , <LinkClassNameOrID> ² , <ID attribute name> ³ , <ID value> ⁴ , <Description attribute name> ⁵ , <Description value> ⁶ , <Message attribute name> ⁷ , <Message text> ⁸
Description	Creates a ENOVIA SmarTeam object that can hold reporting information. The object is created in class <ClassNameOrID>, using <ID attribute name> for the object's key which is set to <ID value>, <Description attribute name> for the object's description which is set to <Description value> and <Message attribute name>, which must be of type Memo, for the <Message text>. The parent object passed to the ENOVIA SmarTeam/SAP Adaptor is linked to the report object as general link of link class <LinkClassNameOrID>.
Remarks	If BAPI messages or BAPI error messages are to be sent, #SAP.BapiMessages# or #SAP.BapiErrorMessages# should be passed as <Message text>.
Example	Smarteam.CreateReportObject SapError, SapError Document Links, _ CN_ID, #System.Date#_#Smarteam.User.Login#_#System.Time#, _ CN_DESCRIPTION, Error while processing object #Smarteam.CN_ID#,_ CN_MESSAGE, #SAP.BapiMessages#

Name	Smarteam.GetData
Arguments	None
Description	Retrieves all parent object's attribute values.
Remarks	
Example	Smarteam.GetData

Name	Smarteam.GetLastRevisionObject
Arguments	<RevisionFilter> ¹
Description	Retrieves the last-revision object of the parent object. <RevisionFilter> can be "", "Last", "Latest" or "LatestAvailable".
Remarks	Attribute values of the last-revision object can be accessed through #Smarteam.LastRevisionObject.<Attribute># .
Example	Smarteam.GetLastRevisionObject Last

	<pre> If #Smarteam.Object_ID# = #Smarteam.LastRevisionObject.Object_ID# ... Endif </pre>
--	--

Name	Smarteam.GetLinks
Arguments	<ClassNameOrID> ¹ , <LinkClassNameOrID> ² , <Direction> ³
Description	Retrieves all attribute values of all linked objects with class ID or class name <ClassNameOrID> which are linked through link class specified by <LinkClassNameOrID>. <Direction> specifies the direction of the link and can have the following values: AllDirections, NoDirection, FirstToSecond and SecondToFirst.
Remarks	The "Loop Smarteam.Links"/"Next Smarteam.Links" statements can be used to loop on the linked objects retrieved.
Example	<pre> Smarteam.GetLinks Items, Documents Items Relation, AllDirections Loop Smarteam.Links ... Next Smarteam.Links </pre>

Name	Smarteam.GetLinksPlus
Arguments	<ClassNameOrID> ¹ , <LinkClassNameOrID> ² , <Direction> ³
Description	Retrieves all attribute values of all linked objects with class ID or class name <ClassNameOrID> which are linked through link class specified by <LinkClassNameOrID> plus all attribute values of all link objects. <Direction> specifies the direction of the link and can have the following values: AllDirections, NoDirection, FirstToSecond and SecondToFirst.
Remarks	The "Loop Smarteam.Links"/"Next Smarteam.Links" statements can be used to loop on the linked objects retrieved.
Example	<pre> Smarteam.GetLinksPlus Items, Documents Items Relation, AllDirections Loop Smarteam.Links ... Next Smarteam.Links </pre>

Name	Smarteam.GetList
Arguments	<ClassIdOrName> ¹ , <Expression1> ² , <Expression2> ³ , ...
Description	<p>Performs a query in ENOVIA SmarTeam class given by <ClassIdOrName>. <Expressions> describe attributes to be supported in the result list, where conditions and order-by clauses. In more detail, an expression can be one of these:</p> <pre> <Attribute> Where <Attribute> <Operator> <Value> And <Attribute> <Operator> <Value> Or <Attribute> <Operator> <Value> Order by <Attribute> Order by <Attribute> Asc </pre>

	Order by <Attribute> Desc In these expressions, <Attribute> is the ENOVIA SmarTeam attribute name like CN_ID, <Operator> is a comparison operator (=, <>, <=, >=, like), and <Value> is the value searched for. If <Operator> equals "like", <Value> can contain wildcard characters "*".
Remarks	Attributes in where and order-by clauses are automatically included in the result list. The "Loop Smarteam.List"/"Next Smarteam.List" statements can be used to loop on the objects retrieved.
Example	Smarteam.GetList Documents, cn_item_no, where cn_id like *313*, and cn_description <> , order by revision Loop Smarteam.List ... Next Smarteam.List

Name	Smarteam.GetList2
Arguments	<SelectStatement>
Description	Performs a simple query in ENOVIA SmarTeam, the select statement being given as argument.
Remarks	Commas in the select statement have to be masked using \,. The "Loop Smarteam.List"/"Next Smarteam.List" statements can be used to loop on the objects retrieved.
Example	Smarteam.GetList2 select * from tn_documents where CN_ID like "%313%" order by revision Smarteam.GetList2 select cn_id\,cn_item\,cn_decription from tn_documents where CN_ID like "%313%" order by revision Loop Smarteam.List ... Next Smarteam.List

Name	Smarteam.GetStructure
Arguments	<ReplacePhantomCondition1>, <Or ReplacePhantomCondition2>, ...
Description	Retrieves all attribute values of all first level children of the parent object but no hierarchical link attributes. If a child satisfies any of the given <ReplacePhantomConditions>, the child is discarded from the children list and replaced by its own children which are themselves proved against the <ReplacePhantomConditions>. Each <ReplacePhantomCondition> must look like "<Attribute> <Operator> <Value>". <Attribute> is an attribute of the child, <Operator> is one of the operator that can be used in Goto conditions, and <Value> is the value that is to be compared with the attribute.
Remarks	The "Loop Smarteam.Structure"/"Next Smarteam.Structure" statements can be used to loop on the structure retrieved.
Example	Smarteam.GetStructure Loop Smarteam.Structure ... Next Smarteam.Structure Smarteam.GetStructure CN_ID ~ P_*, CLASS_ID = 755

Name	Smarteam.GetStructurePlus
Arguments	<ReplacePhantomCondition1>, <Or ReplacePhantomCondition2>, ...
Description	Retrieves all attribute values of all first level children of the parent object plus all attribute values of all hierarchical link objects. If a child satisfies any of the given <ReplacePhantomConditions>, the child is discarded from the children list and replaced by its own children which are themselves proved against the <ReplacePhantomConditions>. Each <ReplacePhantomCondition> must look like "<Attribute> <Operator> <Value>". <Attribute> is an attribute of the child, <Operator> is one of the operator that can be used in Goto conditions, and <Value> is the value that is to be compared with the attribute.
Remarks	The "Loop Smarteam.Structure"/"Next Smarteam.Structure" statements can be used to loop on the structure retrieved.
Example	Smarteam.GetStructurePlus Loop Smarteam.Structure ... Next Smarteam.Structure Smarteam.GetStructurePlus CN_ID ~ P_*, CLASS_ID = 755

Name	Smarteam.InitiateProcess
Arguments	<FlowClassIdOrName> ¹ , <FlowChartName> ² , <ResponseName> ³ , <NextNodeName> ⁴ , <Description> ⁵ , <Comment> ⁶ , <ExecutersNames> ⁷
Description	Initiates a ENOVIA SmarTeam flow process in class <FlowClassIdOrName> using flow chart <FlowChartName>. If no flow chart is given, the default flow chart of the process will be attached. The process is send to node <NextNodeName> via response <ResponseName>. The description of the flow object and the comment for the response are set to <Description> and <Comment>. If executers are specified, they are attached to the node as executers. Their names must be separated by semicolons.
Remarks	
Example	Smarteam.InitiateProcess ECO, ECO Short, Accept, Check ECO, ECO for object #Smarteam.CN_ID#, Initiated by smSAPif, joe;sue

Name	Smarteam.LifeCycleOperation
Arguments	<Operation> ¹ , <UseLifeCycleWindow> ² , <InvokeScripts> ³ , <ExecuteOnTree> ⁴
Description	Performs a life-cycle operation on the parent object. <Operation> can be "CHECKIN" or "RELEASE". If <UseLifeCycleWindow> is "X", ENOVIA SmarTeam's life-cycle window is displayed to perform the operation. Otherwise the operation is performed without dialog. If <InvokeScripts> is set to "X", scripts are invoked. If <ExecuteOnTree> is equal to "X", the life-cycle operation is executed on the entire object tree.
Remarks	
Example	Smarteam.LifeCycleOperation RELEASE, , , X

Name	Smarteam.Login
Arguments	<User> ¹ , <Password> ² , <Encrypted Password> ³
Description	Performs a login for <User> using <Password> or <Encrypted Password>.
Remarks	User this program to encrypt the password.
Example	Smarteam.Login transfer, secret

Name	Smarteam.Logout
Arguments	None
Description	Performs a logout for the user currently logged in.
Remarks	
Example	Smarteam.Logout

Name	Smarteam.RefreshWindow
Arguments	<RefreshAllWindows>
Description	Refreshes ENOVIA SmarTeam's active window(s) to display updated object data. If <RefreshAllWindows> is set to "X", all active windows of all open sessions are refreshed.
Remarks	
Example	Smarteam.RefreshWindow X

Name	Smarteam.Run
Arguments	<Database> ¹ , <ServerMode> ²
Description	Runs a new free threaded ENOVIA SmarTeam engine in server mode, creates a new session and connects this session to <Database>. If <ServerMode> is set to X, the ENOVIA SmarTeam engine is switched to server mode processing.
Remarks	
Example	Smarteam.Run SmDemo, X

Name	Smarteam.Save
Arguments	<InvokeScripts> ¹ , <CheckAuthorization> ²
Description	Updates the parent object in ENOVIA SmarTeam database without dialog. If <InvokeScripts> is set to "X", scripts are invoked. If <CheckAuthorization> is "X", authorization is checked.

Remarks	
Example	Smarteam.Save , X

Name	Smarteam.SendMessage
Arguments	<Subject> ¹ , <Body> ² , <ENOVIA SmarTeam user name1> ³ , <ENOVIA SmarTeam user name2> ⁴ , <ENOVIA SmarTeam user name3> ⁵ , ...
Description	Sends a SmartMessage without further user interaction. The parent object passed to the integration by ENOVIA SmarTeam is attached to the message.
Remarks	If BAPI messages or BAPI error messages are to be sent, #SAP.BapiMessages# or #SAP.BapiErrorMessages# should be passed as <Body>.
Example	Smarteam.SendMessage Error while processing object #Smarteam.CN_ID#, #SAP.BapiMessages#,_joe, jim, sue

Name	Smarteam.SetData
Arguments	<ENOVIA SmarTeam attribute1> ¹ , <Value1> ² , <ENOVIA SmarTeam attribute2> ³ , <Value2> ⁴ , ...
Description	Fills the specified ENOVIA SmarTeam variable of the parent object with the given values. Lookup fields and Reference-to-class fields can be set using "dot" spelling, for instance: CN_LOOKUP.DESCRPTION or CN_REF_ITEM.CN_ID. In this case the necessary query for retrieving the proper object is automatically performed.
Remarks	The ENOVIA SmarTeam attributes must be specified without prefix or postfix, i.e. without #Smarteam.___#. The "dot" spelling can cover more than one level like in CN_ITEM.CN_REF_FIELD.TDM_NAME but it must be made sure that the value found on each level is unique. If a value is not unique, the first entry found for this value is taken. Use settings "MessageLevelReferenceNotFound" and "MessageLevelReferenceNotUnique" in section [General] of the configuration file for controlling possible error situations.
Example	Smarteam.SetData CN_DESCRIPTION, #Document.Basicdata.Description#, TDM_SAP_MAT_NUM,_4711 Smarteam.SetData CN_QUANTITYUNIT.TDM_NAME, KG, CN_REF_ITEM.CN_ID, Part-4711 Smarteam.SetData CN_ITEM.CN_CHANGE_NUMBER.CN_ID, EC-4711

Name	Smarteam.SetObject
Arguments	<ClassIdOrName> ¹ , <ObjectId> ² , <SetDefaultValues> ³ , <IncrementSequences> ⁴
Description	If <ClassIdOrName> and <ObjectId> are both given, the parent object is set to the ENOVIA SmarTeam object thoroughly identified by the two parameters. If <ObjectId> is missing, the parent object is set to a new ENOVIA SmarTeam object in the class specified by <ClassIdOrName>. If <SetDefaultValues> is set to "X", default values are added to the newly created object. And if <IncrementSequences> is equal to "X", all attribute with underlying sequences are incremented.

Remarks	
Example	<pre>Smarteam.SetObject %Class_ID%, %Object_ID% Smarteam.SetObject %Class_ID%, , X, X</pre>

Name	Smarteam.Terminate
Arguments	None
Description	Closes the ENOVIA SmarTeam session currently open and terminates the ENOVIA SmarTeam engine running.
Remarks	
Example	Smarteam.Terminate

Control Functions

Name	AndIf/AndIfNot
Arguments	<Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	An AndIf or AndIfNot can only follow immediately an If or IfNot statement. In this case the given conditions are logically joined with the conditions in the preceding If/IfNot statement by means of an "and" conjunction.
Remarks	For more information cf. If/Else/Endif and IfNot/Else/Endif statements.
Example	<pre>Loop Smarteam.Structure If \Left1{#Smarteam.CN_ITEMNUMBER#} = - AndIf #Smarteam.CLASS_ID# = 755 Document.AddStructure %DOKAR%, #Smarteam.CN_PART_NUMBER#, %PART%, " -", #Smarteam.CN_QUANTITY# Endif Next Smarteam.Structure</pre>

Name	Check
Arguments	<Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	Checks if any given condition is true. In this case execution continues. If all conditions are false, execution terminates. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U~, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison.
Remarks	
Example	Check %STATUS% = IE, %MATNR%! <> !

Name	DoLoop
Arguments	<LoopVariable> ¹ , <FromValue> ² , <ToValue> ³ , <StepValue> ⁴
Description	A Do loop is performed varying <LoopVariable> from <FromValue> to <ToValue> in steps of <StepValue>.
Remarks	The loop has to be closed by the DoNext statement. <LoopVariable> is stored in memory as a definition and should therefore follow appropriate naming conventions for definitions.
Examples	<pre> ... FunctionModule.Call BAPI_DOCUMENT_GETDCLIST DoLoop %I%, 1, #FunctionModule.Tables.DATACARRIERLIST.Count# Set %VAL%, #FunctionModule.Tables.DATACARRIERLIST.%I%# ExitDoLoop %VAL% ~ PC DoNext </pre>

Name	DoNext
Arguments	None
Description	Indicates the end of a loop starting at the DoLoop statement.
Remarks	See DoLoop statement
Example	See DoLoop statement

Name	Exit
Arguments	<ExitCode> ¹ , <ResultString> ²
Description	Execution is terminated immediately. If <ExitCode> and/or <ResultString> are given, the return code and/or the result string of the operation sequence are set by function SetExitCode .
Remarks	
Example	Exit 1, Last used object had object ID #SmarTeam.OBJECT_ID#

Name	ExitDoLoop
Arguments	<Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	Exits the specified DoLoop , if any of the given conditions is true or if no conditions are given. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U=, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison.
Remarks	
Examples	See DoLoop statement

Name	ExitLoop
Arguments	<Table (Class.Characteristics, Class.CharacteristicValues, Classification.Allocations, Classification.Validations, Document.Descriptions, Document.CharacteristicValues, Document.ClassAllocations, Document.Files2, Documents.Links, Document.List, Document.StatusLog, Document.Structure, File.List, Material.BOM, Material.List, Smarteam.Links, Smarteam.List, Smarteam.Structure)>, <Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	Exits the specified loop, if any of the given conditions is true or if no conditions are given. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U=, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison..
Remarks	The <Table> name has to be the same as in the Loop and Next statements.
Examples	<pre> Loop Smarteam.Structure, %I% Document.AddStructure DRW, #Smarteam.CN_ID#, 000, #Smarteam.REVISION#, #Smarteam.CN_QUANTITY# ExitLoop Smarteam.Structure, %I% > 10 Next Smarteam.Structure </pre>

Name	Goto
Arguments	<Label to jump if true> ¹ , <Label to jump if false> ² , <Condition1> ³ , <Or Condition2> ⁴ , <Or Condition3> ⁵ , ...
Description	Jumps to the label passed as first argument if any given condition is true. Jumps to the label passed as second argument if all conditions result to false. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U=, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison.
Remarks	The label where to jump to has to be specified by the Label statement.
Example	<pre> Goto :SetStatus:, , %STATUS% = IE, %STATUS% = SP, %STATUS% = UG ... Label :SetStatus: ... Goto :Messages:, , #SAP.BapiStatusSummary# = E, #SAP.BapiStatusSummary# = A, #SAP.DialogStatus# = A Goto :Messages: ... Label :Messages: </pre>

Name	If/Elseif(Not)/Else/Endif
Arguments	<Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	If any of the given conditions is true, the statements between "If" and "Elseif(Not)/Else" will be executed. If all conditions are false, execution continues with the "Elseif(Not)/Else" statement. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U=, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator.

	is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison.
Remarks	
Example	<pre>If \Mid12{#SmarTeam.CN_ID#} ~ 8* Material.AddBIData SAPLMGMM, 3006, MARC-BESKZ, E, BDC_OKCODE, /00 Else Material.AddBIData SAPLMGMM, 3006, MARC-BESKZ, F, BDC_OKCODE, /00 Endif ... If #System.MessageResult# =U= Cancel Message Abort, Operations will be aborted Endif</pre>

Name	IfNot/ElseIf(Not)/Else/Endif
Arguments	<Condition1>, <Or Condition2>, <Or Condition3>, ...
Description	If all of the given conditions are false, the statements between "If" and "ElseIf(Not)/Else" will be executed. If any condition is true, execution continues with the "ElseIf(Not)/Else" statement. Operators usable in conditions are =, <>, >, < >=, <=, ~, =U=, <U>, >U>, <U<, >U=, <U=, ~U~, =N=, <N>, >N>, <N<, >N=, <N=. "~" is the "like" operator. In "U" operations the arguments are converted to upper case before comparison. "N" operators force a numerical comparison instead of the standard string comparison.
Remarks	
Example	<pre>... Document.Save IfNot #SAP.BapiStatus# = E, #SAP.BapiStatus# = A Message Information, Document "#Document.Documentnumber#" saved to SAP. Document.SetStatusOtherVersions , , , NV, , NotEqual Endif</pre>

Name	Include
Arguments	<SectionName> ¹
Description	Includes all executable lines of section <SectionName>.
Remarks	<SectionName> must be given without brackets. The Include statement can be used recursively. That means an included section can contain another Include statement.
Example	<pre>Include MoreLines ... [MoreLines] # Statements to be included above ...</pre>

Name	Label
Arguments	<LabelName>
Description	Indicates the label where to jump to in a Goto statement.
Remarks	Each <LabelName> must be unique in an action sequence.
Example	See Goto statement

Name	Loop
Arguments	<Table (Class.Characteristics, Class.CharacteristicValues, Classification.Allocations, Classification.Validations, Document.Descriptions, Document.CharacteristicValues, Document.ClassAllocations, Document.Files2, Documents.Links, Document.List, Document.StatusLog, Document.Structure, File.List, Material.BOM, Material.List, Smarteam.Links, Smarteam.List, Smarteam.Structure)> ¹ , <LoopVariable> ²
Description	A loop on the specified table in internal memory is performed.
Remarks	The loop has to be closed by the Next statement. If the loop counter shall be accessed inside the loop, a <LoopVariable> has to be specified. It will be stored in memory as a definition and should therefore follow appropriate naming conventions for definitions.
Examples	<pre> Loop Smarteam.Structure Document.AddStructure DRW, #Smarteam.Structure.CN_ID#, 000, #Smarteam.Structure.REVISION# Next Smarteam.Structure ... Loop Document.Links, %LinkLoopCounter% Material.Clear Material.AddData SAPLMGMM, 0060, RMMG1-MATNR, #Document.Links.ObjectKey# Material.AddData SAPLMGMM, 0070, MSICHTAUSW-KZSEL(01), X Material.AddData SAPLMGMM, 3005, MARA-ZEINR, #Smarteam.CN_PARTNUMBER#, BDC_OKCODE, =BU Material.Save Message Information, Material No. %LinkLoopCounter% saved Next Document.Links </pre>

Name	New
Arguments	<Type1>, <ObjectName1>, <Type2>, <ObjectName2>, <Type3>, <ObjectName3>, ...
Description	New objects of type <Type> and with names <ObjectName> are created. Possible types are: BATCHINPUT, CLASS, CLASSIFICATION, CLASSIFICATION2, DOCUMENT, ECMASER, EXCEL, FILE, FUNCTIONMODULE, GRID, MATERIAL, PROJECT, SMARTTEAM. The new objects can then be used according to documented functions and variables of that type.
Remarks	
Example	<pre> New Document, D, Material, M, Smarteam, S D.AddBasicData DRW, #S.CN_ID#, 000, #S.REVISION# M.AddBasicData #D.Documentnumber# </pre>

	New BatchInput, BI BI.CallTransaction MM01, X
--	--

Name	Next
Arguments	<Table (Class.Characteristics, Class.CharacteristicValues, Classification.Allocations, Classification.Validations, Document.Descriptions, Document.CharacteristicValues, Document.ClassAllocations, Document.Files2, Documents.Links, Document.List, Document.StatusLog, Document.Structure, File.List, Material.BOM, Material.List, Smarteam.Links, Smarteam.List, Smarteam.Structure)> ¹
Description	Indicates the end of a loop starting at the Loop statement.
Remarks	See Loop statement
Example	See Loop statement

Name	SetErrorHandler
Arguments	<SectionName> ¹
Description	Passes execution to section <SectionName> if an unrecoverable error occurs in the ENOVIA SmarTeam/SAP Adaptor.
Remarks	<SectionName> must be given without brackets. See section Error Handling for more information.
Example	SetErrorHandler OnError ... [OnError] SetExitCode 99, #System.ErrorDescription# Message Abort, Error: #System.ErrorDescription#

Name	Sleep
Arguments	<Milliseconds> ¹
Description	Execution is suspended for the given amount of milliseconds.
Remarks	
Example	Sleep 3000

Miscellaneous Functions

Name	# (Comment)
Arguments	Anything
Description	# indicates a comment line. That means a line having a # as its first character is not executed at all.

Remarks	The # has to be the first character of the line. At any other position #...# represents an SAP or other variable .
Example	# Any text as comment

Name	CallExternalFunction
Arguments	<Application.Class> ¹ , <Server> ² , <Function> ³ , <Argument> ⁴
Description	<p>Calls function <Function> of object <Application.Class> on server <Server>. <Application.Class> must have been compiled to an ActiveX dll with <Function> being a public function in a multi-use <Class>. <Argument> is passed as string to the function. The interface definition of <Function> must be equivalent to</p> <p>Public Function <Function>(ByVal Argument As String, ByVal TraceFile as String) As String</p> <p>The second argument passed to the function is the name of the trace file used by the ENOVIA SmarTeam/SAP Adaptor. The external function can write its own trace to that file. TraceFile is set to "", if no trace shall be written.</p>
Remarks	<p>The result of the function call can be accessed through variable #SYSTEM.FUNCTIONRESULT#.</p> <p>If the function shall return several values proceed as described in the sample configurations.</p>
Example	<pre>CallExternalFunction MyApp.Cls, , GetResult, ClassID=#SmarTeam.CLASS_ID# ObjectID=#SmarTeam.OBJECT_ID# If #System.FunctionResult# = 4711 ... Endif</pre>

Name	ClearBapiMessages
Arguments	None
Description	Clears all BAPI messages.
Remarks	
Example	ClearBapiMessages

Name	Commit
Arguments	<On Off>
Description	Turns the commit setting on or off. Default is On.
Remarks	The commit setting controls whether SAP transactions are committed (in case of no errors) or not. At the present point of time this setting only affects functions that require an explicit commit or rollback by means of function modules BAPI_TRANSACTION_COMMIT or BAPI_TRANSACTION_ROLLBACK. Actually these are all Document functions.

	The commit setting is reset to its default value "On" at the beginning of an activity section which is executed (and not only included).
Example	Commit Off

Name	Define
Arguments	<Key1>, <Value1>, <Key2>, <Value2>, <Key3>, <Value3>, ...
Description	Adds definitions to the internal memory.
Remarks	The definitions are available only during the action sequence in which they are issued. In opposition to the Set statement, <Values> are resolved when the definition is used, not at definition time. This means that different values may be substituted when a definition is used in different places. To learn more about definitions, follow this link.
Example	Define %DOKAR%, #SmarTeam.CN_DRAWINGTYPE# Define %CLASS%, DOC_#SmarTeam.CN_DOCCLASS# Define %SAPSTATUS%, \Left2{#SmarTeam.CN_SAPSTATUS#}, %VERSION%, \LeftDotRight2{ #SmarTeam.REVISION#}, %PART%, 000

Name	DisplayBapiMessages
Arguments	None
Description	Displays all BAPI messages of the current action sequence.
Remarks	
Example	DisplayBapiMessages

Name	ExtractString
Arguments	<String> ¹ , <Range1> ² , <Variable1> ³ , <Range2> ⁴ , <Variable2> ⁵ , ...
Description	Extracts parts of <String> defined by <Ranges> and adds the resulting strings as definitions to the internal memory using <Variables>. <Range> must be given as <from>-<to>. If <from> is missing, 1 is used. If <to> is missing, the length of <String> is used.
Remarks	The <Variables> should follow naming conventions for definitions .
Example	ExtractString #File.Line#, -10, %PART1%, 20-35, %PART2%, 36-, %REST%

Name	Message
Arguments	<MessageType> ¹ , <MessageText> ² , <Title> ³ , <Yes Button Text> ⁴ , <No Button

	Text> ⁵ , <Cancel Button Text> ⁶
Description	Displays a message box. <MessageText> is the text displayed. "\n" will be replaced by a new line character, and "\t" will insert a tab character. <MessageType> can be "Abort", "Error", "Warning", "Success", "Information", "Question", or "CancelQuestion". If <MessageType> is "Question", the message box shows buttons "Yes" and "No", and if <MessageType> is "CancelQuestion", the message box shows buttons "Yes", "No" and "Cancel". In all other cases it shows an "OK" button only. System defaults for the title and the button texts can be overwritten using parameters <Title>, <Yes Button Text>, <No Button Text>, <Cancel Button Text>.
Remarks	The user's reaction (the button pressed) can be accessed by the #SYSTEM.MESSAGERESULT# variable. The value is the caption (text) of the button pressed.
Example	Message Question, "Do you want to create the bill of material?", , Yes, NO If #System.MessageResult# = Yes ... Endif Message Success, Operation ended successfully

Name	Progress
Arguments	<Percentage> ¹ , <Message> ²
Description	Displays <Message> and adjusts the progress bar in the progress window.
Remarks	The percentage value should be between 0 and 100. If no Progress statements are issued, percentage is determined automatically using a statement counter. If no <Message> is passed, a standard message text is used.
Example	Progress 65 Progress 95, Nearly finished

Name	SaveBapiMessagesToFile
Arguments	<Filename>
Description	Saves all BAPI messages of the current action sequence and other information to <Filename>.
Remarks	The file is opened for append.
Example	SaveBapiMessagesToFile C:\\Temp\\BapiMsg_#System.Date#.txt

Name	SendMail
Arguments	<Subject> ¹ , <Body> ² , <Attachment> ³ , <FromAddress> ⁴ , <FromName> ⁵ , <ToAddress> ⁶ , <ToName> ⁷ , <SMTPServer> ⁸ , <UserName> ⁹ , <Password> ¹⁰
Description	Sends an SMTP mail via <SMTPServer>. If the server requires authorization, <UserName> and <Password> are used.
Remarks	If BAPI messages or BAPI error messages are to be sent, #SAP.BapiMessages# or #SAP.BapiErrorMessages# should be passed as <Body>.

Example	SendMail Error while processing object #Smarteam.CN_ID#, #SAP.BapiMessages#, , _ #Smarteam.User.USER_EMAIL#, #Smarteam.User.FIRST_NAME# #Smarteam.User.LAST_NAME#, _ sapadmin@mycompany.com, , SMTP.mycompany.com
---------	--

Name	Set
Arguments	<Key1>, <Value1>, <Key2>, <Value2>, <Key3>, <Value3>, ...
Description	Adds definitions to the internal memory.
Remarks	The definitions are available only during the action sequence in which they are issued. In opposition to the Define statement, the <Values> are resolved at definition time, not at run time. This means that their values are firm for the entire action sequence unless they are redefined by another Set or Define statement. To learn more about definitions, follow this link.
Example	Set %StartTime%, #System.Time#

Name	SetExitCode
Arguments	<ExitCode> ¹ , <ResultString> ²
Description	Sets the exit code returned to the calling ENOVIA SmarTeam script or to the calling application for the present action sequence. Possible values are "Err_None" ("0") or "Err_Gen" ("1"). Setting the exit code to "1" usually cancels a ENOVIA SmarTeam operation when called by the "before" hook. Furthermore a <ResultString> can be returned to the ENOVIA SmarTeam script or to the calling application.
Remarks	Run time errors of the integration software usually come to exit code "1". Passing an operation sequence as non-modal always returns exit code "0" and an empty return string. If a result has to be returned in <ResultString>, the ENOVIA SmarTeam/SAP Adaptor must be called modally.
Example	SetExitCode 1, Last used object has object ID #Smarteam.OBJECT_ID#

Name	SplitString
Arguments	<String> ¹ , <Split> ² , <Variable1> ³ , <Variable2> ⁴ , <Variable3> ⁵ , ...
Description	Splits <String> at <Split>. <Split> itself can be a string or a single character. The result of the split operation can be accessed through system variables #System.SplitString.1#, #System.SplitString.2#, ... The number of these fields is given by #System.SplitString.Count#. Furthermore, if variables <Variable1>, <Variable2>, ... are given, the result of the split operation is passed as definitions to internal memory. If there are more variables than result strings, all remaining variables that have no string assigned are defined as empty strings "". If there are less variables than result strings, the superfluous result strings are lost.
Remarks	The <Variables> should follow naming conventions for definitions .
Example	SplitString #File.Line#, ; , %PART1%, %PART2%, %REST%

	Message Information, %PART1% --- %PART2% --- %REST% SplitString #File.Line#, ; DoLoop %Item%, 1, #System.SplitString.Count# Message Information, Item %Item%: #System.SplitString.%Item%# DoNext
--	--

Name	SwitchSAPUser
Arguments	<UserName> ¹ , <ClearPassword> ² , <EncryptedPassword> ³ , <Language> ⁴ , <DialogMode> ⁵
Description	The function closes the current RFC connection and opens a new one using the given logon values. <DialogMode> can be "Dialog" or "NoDialog". If <UserName> is "Back", the function switches back to the previous logon.
Remarks	
Example	SwitchSAPUser JIM, , 7E0C0E0D1D79010A646B7F784B5F582A444B5F582A444B5F58, EN, Dialog Include SectionToBeExecutedInDialogMode SwitchSAPUser Back

Name	WaitOnCommit
Arguments	<On Off>
Description	Turns the wait-on-commit setting on or off. Default is On.
Remarks	The wait-on-commit parameter controls whether database commits in SAP are performed synchronously or asynchronously.
Example	WaitOnCommit Off

Available Inline Functions

Navigator

[\C](#) result is C if C is any of the following characters: \ , # { }

[\Charx{STRING}](#) result is the x-th character of STRING

[\Midx{STRING}](#) result is the part of STRING starting at position x

[\Leftx{STRING}](#) result are the x leftmost characters of STRING

[\Rightx{STRING}](#) result are the x rightmost characters of STRING

[\Trim{STRING}](#) result is STRING without leading and trailing space.

[\TrimLeftx{STRING}](#) result are the x leftmost characters of STRING but STRING is trimmed before

[\TrimRightx{STRING}](#) result are the x rightmost characters of STRING but STRING is trimmed before

[\LeftDotLeftx{STRING}](#) result are the x leftmost characters of the substring of STRING which is left of the dot "."

[\LeftDotRightx{STRING}](#) result are the x rightmost characters of the substring of STRING which is left of the dot "."

[\LeftSpaceLeftx{STRING}](#) result are the x leftmost characters of the substring of STRING which is left of the space " "

[\LeftSpaceRightx{STRING}](#) result are the x rightmost characters of the substring of STRING which is left of the space " "

[\DeleteC{STRING}](#) result is STRING without all occurrences of character C

[\MaskC{STRING}](#) result is STRING with all occurrences of C replaced by \C

[\UnmaskC{STRING}](#) result is STRING with all occurrences of \C replaced by C

[\LCASE{STRING}](#) result is STRING with all characters converted to lower case.

[\UCASE{STRING}](#) result is STRING with all characters converted to upper case.

[\LookUpX{KEY}](#) result is the line following line KEY in section [LookUpX]

[\Path{FILE}](#) result is the path of FILE

[\File{FILE}](#) result is the filename of FILE excluding path and excluding extension

[\FileExt{FILE}](#) result is the filename of FILE excluding path but including extension

[\Filename{FILE}](#) result is the filename of FILE including path but excluding extension

[\Extension{FILE}](#) result is the extension of FILE

[\ViewfileExtension{FILE_TYPE}](#) result is the (first) extension of the view file customized for the given ENOVIA SmarTeam file type

[\Eval{EXPRESSION}](#) result is the evaluated result of arithmetic EXPRESSION

[\Intx{NUMBER}](#) result is a string of length x containing the binary representation of NUMBER

[\Hex{NUMBER}](#) result is a string which hex representation is stored in NUMBER

[\Stringx{CHARACTER}](#) result is a string of length x containing CHARACTER only.

[\WinDate{SAPDATE}](#) result is a string containing a Windows short date built from SAPDATE string in SAP internal representation

[\WinDateX{SAPDATE}](#) result is a string containing a Windows short date built from SAPDATE string in SAP external representation

[\SapDate{WINDATE}](#) result is a string containing a SAP date in SAP internal representation built from WINDATE string

[\SapDateX{WINDATE}](#) result is a string containing a SAP date in SAP external representation built from WINDATE string

[\SwitchFloatSigns{NUMBER}](#) result is a string representing NUMBER but with decimal sign and thousand sign exchanged

[\SapInternalx{NUMBER}](#) result is NUMBER converted to internal SAP representation

[\SapExternal{NUMBER}](#) result is NUMBER converted to external SAP representation

[\SapInternalLanguage{ISOLANGUAGE}](#) result is ISOLANGUAGE converted to internal SAP language key

[\SapExternalLanguage{LANGUAGE}](#) result is LANGUAGE converted to external ISO language

key

[\DocumentKey{TYPE,NUMBER,PART,VERSION}](#) result is a string representing a correct document key build from the key fields separated by comma

[\ExpandDocumentKey{STRING}](#) result is a string representing the four document key fields separated by comma.

[\ExpandDocumentKey2{STRING}](#) result is a string representing the four document key fields separated by comma.

Name	\C
Description	Result is C if C is any of the following characters: \ , # { }. That means \ is an escape character.
Example	The result of "\\" is "\", the result of "\," is ",", the result of "\#" is "#", the result of "\{" is "{" and the result of "\}" is "}".

Name	\Charx{STRING}
Description	Result is the x-th character of STRING.
Example	The result of "\Char14{abcdefghijklmnopqrstuvwxyz}" is "n".

Name	\Midx{STRING}
Description	Result is the part of STRING starting at position x.
Example	The result of "\Mid14{abcdefghijklmnopqrstuvwxyz}" is "nopqrstuvwxyz".

Name	\Leftx{STRING}
Description	Result are the x leftmost characters of STRING.
Example	The result of "\Left6{abcdefghijklmnopqrstuvwxyz}" is "abcdef".

Name	\Rightx{STRING}
Description	Result are the x rightmost characters of STRING
Example	The result of "\Right3{abcdefghijklmnopqrstuvwxyz}" is "xyz".

Name	\Trim{STRING}
Description	Result is STRING without leading and trailing space.

Example	The result of "\Trim{ A b c 123 }" is "A b c 123".
---------	--

Name	\TrimLeftx{STRING}
Description	Result are the x leftmost characters of STRING but STRING is trimmed before.
Example	The result of "\LeftTrim6{ abcdefghijklmnopqrstuvwxyz }" is "abcdef"

Name	\TrimRightx{STRING}
Description	Result are the x rightmost characters of STRING but STRING is trimmed before.
Example	The result of "\TrimRight3{ abcdefghijklmnopqrstuvwxyz }" is "xyz".

Name	\LeftDotLeftx{STRING}
Description	\LeftDotLeftx{STRING} result are the x leftmost characters of the substring of STRING which is left of the dot "."
Example	The result of "\LeftDotLeft1{AB.3}" is "A".

Name	\LeftDotRightx{STRING}
Description	Result are the x rightmost characters of the substring of STRING which is left of the dot "."
Example	The result of "\LeftDotRight2{ABC.8}" is "BC".

Name	\LeftSpaceLeftx{STRING}
Description	\LeftSpaceLeftx{STRING} result are the x leftmost characters of the substring of STRING which is left of the dot " "
Example	The result of "\LeftSpaceLeft1{AB 3}" is "A".

Name	\LeftSpaceRightx{STRING}
Description	Result are the x rightmost characters of the substring of STRING which is left of the dot " "
Example	The result of "\LeftSpaceRight2{ABC 8}" is "BC".

Name	\DeleteC{STRING}
Description	Result is STRING without all occurrences of character C.
Example	The result of "\Delete {My home is my castle.}" is "Myhomeismycastle".

Name	\MaskC{STRING}
Description	Result is STRING with all occurrences of C replaced by \C.
Example	The result of "\Mask,{A, B, or C}" is "A\, B\, or C".

Name	\UnmaskC{STRING}
Description	Result is STRING with all occurrences of \C replaced by C.
Example	The result of "\Unmask,{A\, B\, or C}" is "A, B, or C".

Name	\LCASE{STRING}
Description	Result is STRING with all characters converted to lower case.
Example	The result of "\LCASE{AbCdEfGhI}" is "abcdefghi".

Name	\UCASE{STRING}
Description	Result is STRING with all characters converted to upper case.
Example	The result of "\UCASE{AbCdEfGhI}" is "ABCDEFGHI".

Name	\LookUpX{KEY}
Description	Result is the line following line KEY in section [LookUpX].
Example	<p>If smSAPif.ini file contains the following section:</p> <pre>[LookUp27] Line1 Line2 Key Result Line5 Line6</pre> <p>the result of "\LookUp27{Key}" is "Result".</p>

Name	\Path{FILE}
Description	Result is the path of FILE.
Example	The result of "\Path{C:\WorkDir\4711.tif}" is "C:\WorkDir".

Name	\File{FILE}
Description	Result is the filename of FILE excluding path and excluding extension.
Example	The result of "\File{C:\WorkDir\4711.tif}" is "4711".

Name	\FileExt{FILE}
Description	Result is the filename of FILE excluding path but including extension.
Example	The result of "\FileExt{C:\WorkDir\4711.tif}" is "4711.tif".

Name	\Filename{FILE}
Description	Result is the filename of FILE including path but excluding extension.
Example	The result of "\Filename{C:\WorkDir\4711.tif}" is "C:\WorkDir\4711".

Name	\Extension{FILE}
Description	Result is the extension of FILE.
Example	The result of "\Extension{C:\WorkDir\4711.tif}" is ".tif".

Name	\ViewfileExtension{FILE_TYPE}
Description	Result is the (first) extension of the view file customized for the given ENOVIA SmarTeam file type.
Example	The result of "\ViewfileExtension{48}" is, for instance, ".CATDrawing.pdf" depending on the customization of the ENOVIA SmarTeam application tool setup.

Name	\Eval{EXPRESSION}
Description	Result is the evaluated result of EXPRESSION which must be a valid arithmetic expression containing numbers, arithmetic operators "*", "/", "\", "+", "-" and brackets

	"(", ")" only.
Example	The result of "\Eval{(4*(3+1))\3}" is "5".

Name	\Intx{NUMBER}
Description	Result is a string of length x containing the binary representation of NUMBER.
Example	The result of "\Int4{9710}" is "i%□□" or - in hex representation - "EE250000".

Name	\Hex{NUMBER}
Description	Result is a string which hex representation is stored in NUMBER.
Example	The result of "\Hex{616263}" is "abc".

Name	\Stringx{CHARACTER}
Description	Result is a string of length x containing CHARACTER only.
Example	The result of "\String10{ }" is " ".

Name	\WinDate{SAPDATE}
Description	Result is a string containing a Windows short date built from SAPDATE which must be formatted as YYYYMMDD (SAP internal representation).
Example	The result of "\WinDate{20043112}" is "31.12.2004" or "31/12/2004" or "12/31/2004" depending on Windows locale setting.

Name	\WinDateX{SAPDATE}
Description	Result is a string containing a Windows short date built from SAPDATE in external representation.
Example	For instance, the result of "\WinDate{31.12.2004}" is "12/31/2004" if SAP language setting is German and Windows locale setting is American English.

Name	\SapDate{WINDATE}
Description	Result is a string containing a date formatted in SAP internal representation (YYYYMMDD) built from WINDATE.
Example	The result of "\SapDate{31.12.2004}" (or "31/12/2004" or "12/31/2004" depending on

	Windows locale setting) is 20043112.
--	--------------------------------------

Name	\SapDateX{WINDATE}
Description	Result is a string containing a date formatted in SAP external representation built from WINDATE.
Example	For instance, the result of "\SapDate{12/31/2004}" is "31.12.2004" if SAP language setting is German and Windows locale setting is American English..

Name	\SwitchFloatSigns{NUMBER}
Description	Result is a string representing NUMBER but with decimal sign and thousand sign exchanged.
Example	The result of "\SwitchFloatSigns{1.234,56}" is "1,234.56" and the result of "\SwitchFloatSigns{1,234.56}" is "1.234,56".

Name	\SapInternalx{NUMBER}
Description	Result is NUMBER converted to internal SAP representation. x specifies the number of digits in result.
Example	The result of "\SapInternal18{4711}" is "000000000000004711".

Name	\SapExternal{NUMBER}
Description	Result is NUMBER converted to external SAP representation.
Example	The result of "\SapExternal{000000000000004711}" is "4711".

Name	\SapInternalLanguage{ISOLANGUAGE}
Description	Result is ISOLANGUAGE converted to internal SAP language key.
Example	The result of "\SapInternalLanguage{SL}" is "5".

Name	\SapExternalLanguage{LANGUAGE}
Description	Result is LANGUAGE converted to external ISO language key.
Example	The result of "\SapExternalLanguage{5}" is "SL".

Name	\DocumentKey{DOCTYPE,DOCNUMBER,DOCPART,DOCVERSION}
Description	Result is a string representing a correct document key build from the key fields separated by comma.
Example	The result of "\DocumentKey{DRW,ST4711,001,A}" is "DRWST4711 A 001".

Name	\ExpandDocumentKey{STRING}
Description	Result is a string representing the four document key fields separated by comma.
Example	The result of "\ExpandDocumentKey{CAD10000000185 AA000}" is "CAD,000000000000000010000000185,000,AA".

Name	\ExpandDocumentKey2{STRING}
Description	Result is a string representing the four document key fields separated by comma. The difference to inline function ExpandDocumentKey is that here the document key fields in STRING are separated by blank while there document key fields are not separated at all.
Example	The result of "\ExpandDocumentKey2{CAD 10000000185 AA 000}" is "CAD,000000000000000010000000185,000,AA".

Available Variables and Definitions

Preliminary and Navigator

All variables - either ENOVIA SmarTeam or SAP or others - can be accessed using Hash characters # #. For instance, the ID of a ENOVIA SmarTeam object is referenced by #SMARTEAM.OBJECT.CN_ID# and the value of field "Documentnumber" of SAP object "Document" is given by #DOCUMENT.DOCUMENTNUMBER#. Spelling of variable names is not case sensitive, and if a variable is not defined, a warning message is displayed.

This file shows the "generic" variable names that have to be adopted to the actual object definitions. If, for instance, an activity section contains a statement like "New Document, D, Smarteam, S", the document number of document object D can be accessed using #D.DocumentNumber# and the ID of Smarteam object S is given by #S.OBJECT.CN_ID#.

ENOVIA SmarTeam Data

SAP Data

[General Data](#)

[Class Data](#)

[Classification Data](#)

[Document Data](#)

[EC Master Data](#)

[Function Module Call Data](#)

[Material Data](#)

[Project Data](#)

Other Data

[Excel Data](#)

[File Data](#)

[Grid Data](#)

[System Data](#)

Definitions

ENOVIA SmarTeam Data

#SMARTEAM.OBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside the SMARTEAM object

#SMARTEAM.<ATTRIBUTE># Abbreviation for #SMARTEAM.OBJECT.<ATTRIBUTE>#

#SMARTEAM.LASTREVISIONOBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside the last revision object

#SMARTEAM.ATTRIBUTES.COUNT# Number of attributes of the SMARTEAM object

#SMARTEAM.ATTRIBUTES.NAME# The name of an object's attribute inside a loop

#SMARTEAM.ATTRIBUTES.EXTERNALNAME# The external name of an object's attribute inside a loop

#SMARTEAM.ATTRIBUTES.TYPE# The type of an object's attribute inside a loop(Possible values are: Blob, Boolean, Char, Date, Double, EffectiveDateFrom, EffectiveDateUntil, HTML, Integer, Memo, Money, Object, ObjectIdentifier(LookUp), ObjectIdentifier(RefToClass), Pointer, RelTimeStamp, SmallInt, Time, TimeStamp and URL)

#SMARTEAM.ATTRIBUTES.SIZE# The size of an object's attribute inside a loop

#SMARTEAM.ATTRIBUTES.VALUE# The value of an object's attribute inside a loop

#SMARTEAM.CLASS.CLASSID# The class ID of the object's class

#SMARTEAM.CLASS.SUPERCLASSID# The super class ID of the object's class

#SMARTEAM.CLASS.NAME# The name of the object's class

#SMARTEAM.CLASS.EXTERNALNAME# The external name of the object's class

#SMARTEAM.CLASS.INTERNALNAME# The internal name of the object's class

#SMARTEAM.CLASS.TABLERNAME# The name of the object's class table

#SMARTEAM.STRUCTURE.COUNT# Number of children of the SMARTEAM object

#SMARTEAM.STRUCTURE.OBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside a child object

#SMARTEAM.STRUCTURE.LINKOBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside the link object to a child

#SMARTEAM.STRUCTURE.<ATTRIBUTE># Abbreviation for

#SMARTEAM.STRUCTURE.OBJECT.<ATTRIBUTE># or

#SMARTEAM.STRUCTURE.LINKOBJECT.<ATTRIBUTE># if the first does not exist

#SMARTEAM.LINKS.COUNT# Number of objects linked to the SMARTEAM object

#SMARTEAM.LINKS.OBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside a linked object

#SMARTEAM.LINKS.LINKOBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> inside the link object to the linked object

#SMARTEAM.LINKS.<ATTRIBUTE># Abbreviation for

#SMARTEAM.LINKS.OBJECT.<ATTRIBUTE># or

#SMARTEAM.LINKS.LINKOBJECT.<ATTRIBUTE># if the first does not exist

#SMARTEAM.LIST.COUNT# Number of objects in result list of SMARTEAM query or simple query

#SMARTEAM.LIST.OBJECT.<ATTRIBUTE># Value of <ATTRIBUTE> in result list of SMARTEAM query or simple query

#SMARTEAM.LIST.<ATTRIBUTE># Abbreviation for

#SMARTEAM.LIST.OBJECT.<ATTRIBUTE>#

#SMARTEAM.USER.<ATTRIBUTE># Value of <ATTRIBUTE> Inside the SMARTEAM login user object

#SMARTEAM.LOCALCONFIG.<VARIABLE># Value of local configuration variable <VARIABLE>

#SMARTEAM.ADMINCONFIG.<VARIABLE># Value of admin configuration variable <VARIABLE>

#SMARTEAM.DATABASE.NAME# The database name

#SMARTEAM.DATABASE.VERSION# The database version

Attribute values of children can be accessed only inside "Loop Smarteam.Structure"/"Next Smarteam.Structure" loops. Attribute values of linked objects can be accessed only inside "Loop Smarteam.Links"/"Next Smarteam.Links" loops. Attribute values of objects in result lists of

queries or simple queries can be accessed only inside "Loop Smarteam.List"/"Next Smarteam.List" loops. Attribute properties can be accessed only inside "Loop Smarteam.Attributes"/"Next Smarteam.Attributes" loops.

If <ATTRIBUTE> is an object identifier (i. e. lookup or reference to class), values from the lookup or from the referenced object can be retrieved by means of more "dots". For instance, CN_REFITEM.CN_DESCRIPTION will give an item's description, if CN_REFITEM contains the reference to an item object. CN_REFITEM.STATE.DESCRPTION will give the description of the referenced item's state. And CN_REFITEM.USER_ID_MOD.LOGIN will give the login name of the user who last modified the referenced item. Special cases are lookup classes and login names: To simplify access to the description of a lookup or the login name of a user, the .DESCRIPTION and .LOGIN extensions need not be specified. So the last two examples can be abbreviated to CN_REFITEM.STATE and CN_REFITEM.USER_ID_MOD.

If <ATTRIBUTE> is an object identifier, the value of the SMARTEAM projection associated with that field can also be retrieved using <ATTRIBUTE>.PROJECTIONVALUE. If, for instance, a ENOVIA SmarTeam object contains a field CN_REFITEM which refers to an item, the projection value displayed in ENOVIA SmarTeam's GUI can be accessed by means of #SMARTEAM.CN_REFITEM.PROJECTIONVALUE#.

SAP Data

General Data

```
#SAP.DESTINATION# The destination as specified as name of the destination section
#SAP.LANGUAGE# The language used for login to the R/3 system
#SAP.CLIENT# The client used for login to the R/3 system
#SAP.USER# The user name used for login to the R/3 system
#SAP.PASSWORD# The password used for login to the R/3 system
#SAP.RELEASE# The release of the R/3 system
#SAP.DECIMALSIGN# The decimal sign used by SAP
#SAP.DATEFORMAT# The date format used by SAP
#SAP.BAPISTATUS# The message type returned by the last BAPI call
                  ("A", "E", "W", "I", "S" or " ")
#SAP.BAPISTATUSSUMMARY# The most severe message type of all BAPI calls
                       ("A", "E", "W", "I", "S" or " ")
#SAP.LASTBAPIMESSAGE# The message text returned by the last BAPI call
#SAP.BAPIMESSAGES# All BAPI messages as string
#SAP.BAPIERRORMESSAGES# All BAPI error and abort messages as string
#SAP.DIALOGSTATUS# The message type returned by the last dialog step
                   ("A", "E", "W", "I", "S" or " ")
```

Class Data

```
#CLASS.CLASSNUM# Class name
#CLASS.CLASSTYPE# Class type, e. g. "001", "017"
```

Available inside a "Loop Class.Characteristics"/"Next Class.Characteristics" loop only:

```
#CLASS.CHARACTERISTICS.COUNT# Number of table lines
#CLASS.CHARACTERISTICS.NAMECHAR#
```

```
#CLASS.CHARACTERISTICS.DESCRCHAR#
#CLASS.CHARACTERISTICS.ENTRYOBLIGATORY#
#CLASS.CHARACTERISTICS.ADDITIONALVAL#
#CLASS.CHARACTERISTICS.UNIT#
#CLASS.CHARACTERISTICS.UNITTEXT#
#CLASS.CHARACTERISTICS.BASEUOMISO#
#CLASS.CHARACTERISTICS.CURRENCY#
#CLASS.CHARACTERISTICS.CURRENCYISO#
#CLASS.CHARACTERISTICS.DATATYPE#
#CLASS.CHARACTERISTICS.NUMBERDIGITS#
#CLASS.CHARACTERISTICS.NUMBERDECIMALS#
#CLASS.CHARACTERISTICS.TEMPLATE#
#CLASS.CHARACTERISTICS.DINKEY#
#CLASS.CHARACTERISTICS.DEPARTMENTVIEW#
#CLASS.CHARACTERISTICS.SIGN#
#CLASS.CHARACTERISTICS.SINGLEVALUE#
#CLASS.CHARACTERISTICS.INTERVALSALLOWED#
#CLASS.CHARACTERISTICS.CASESENSITIVE#
#CLASS.CHARACTERISTICS.OBJECTTABLE#
#CLASS.CHARACTERISTICS.TABLEFIELD#
#CLASS.CHARACTERISTICS.CHARINHERITED#
```

See SAP structure BAPI_CHAR for details.

Available inside a "Loop Class.CharacteristicValues"/"Next Class.CharacteristicValues" loop only:

```
#CLASS.CHARACTERISTICVALUES.COUNT# Number of table lines
#CLASS.CHARACTERISTICVALUES.NAMECHAR#
#CLASS.CHARACTERISTICVALUES.CHARVALUE#
#CLASS.CHARACTERISTICVALUES.DESCRVAL#
#CLASS.CHARACTERISTICVALUES.NUMVALFM#
#CLASS.CHARACTERISTICVALUES.NUMVALTO#
#CLASS.CHARACTERISTICVALUES.CURRVALFM#
#CLASS.CHARACTERISTICVALUES.CURRVALTO#
#CLASS.CHARACTERISTICVALUES.VALRELATN#
```

See SAP structure BAPI_CHAR_VALUES for details.

Available inside a "Loop Class.SelectedObjects"/"Next Class.SelectedObjects" loop only:

```
#CLASS.SELECTEDOBJECTS.COUNT# Number of table lines
#CLASS.SELECTEDOBJECTS.OBJECT#
#CLASS.SELECTEDOBJECTS.OBJECTTEXT#
```

See SAP structure BAPI_OBJECTS_VALUES for details.

Classification Data

```
#CLASSIFICATION.CLASSNUM# Class name
#CLASSIFICATION.CLASSTYPE# Class type, e. g. "001", "017"
#CLASSIFICATION.OBJECTTYPE# Object type, e. g. "DRAW", "MARA", "PRPS"
```

Available inside a "Loop Classification.Allocations"/"Next Classification.Allocations" loop only:

```
#CLASSIFICATION.ALLOCATIONS.COUNT# Number of table lines
#CLASSIFICATION.ALLOCATIONS.CLASS#
#CLASSIFICATION.ALLOCATIONS.CATCHWORD#
#CLASSIFICATION.ALLOCATIONS.STATUS#
#CLASSIFICATION.ALLOCATIONS.STATDESCR#
#CLASSIFICATION.ALLOCATIONS.STANDARDCL#
```

See SAP structure API_ALLOC for details.

Available inside a "Loop Classification.Validations"/"Next Classification.Validations" loop only:

```
#CLASSIFICATION.VALIDATIONS.COUNT# Number of table lines
#CLASSIFICATION.VALIDATIONS.CHARACT#
#CLASSIFICATION.VALIDATIONS.VALUE#
#CLASSIFICATION.VALIDATIONS.VALUENEUTRAL#
#CLASSIFICATION.VALIDATIONS.INHERITED#
#CLASSIFICATION.VALIDATIONS.VALASSIGN#
#CLASSIFICATION.VALIDATIONS.INSTANCE#
```

See SAP structure API_VAL_R for details.

To get the external and internal value of a certain characteristic use:

```
#CLASSIFICATION.CHARACTERISTICVALUE.<NAME># for external representation
#CLASSIFICATION.CHARACTERISTICVALUENEUTRAL.<NAME># for internal representation
```

where <NAME> is the internal SAP name of the characteristic, not the display name.

Document Data

```
#DOCUMENT.DOCUMENTTYPE# Document type
#DOCUMENT.DOCUMENTNUMBER# Document number
#DOCUMENT.DOCUMENTVERSION# Document version
#DOCUMENT.DOCUMENTPART# Document part
#DOCUMENT.STATUSEXTERN# Document status in external representation
#DOCUMENT.STATUSINTERN# Document status in internal representation
#DOCUMENT.LATESTVERSION# Document version returned by Document.GetLatest
#DOCUMENT.EXISTS# Equals "X" if document exists and space if it does not
#DOCUMENT.ISDELETED# Equals "X" if document is marked for deletion and space if it is not
#DOCUMENT.FRONTENDTYPE# the frontend type returned by Document.GetFrontendType
#DOCUMENT.DATACARRIER# Resulting SAP data carrier if function
                        Document.EncodeDataCarrier was called
#DOCUMENT.DOCFILE# Resulting SAP document filename if function
                        Document.EncodeDataCarrier was called
#DOCUMENT.DOCPATH# Resulting path of SAP data carrier if function
                        Document.DecodeDataCarrier was called
```

For details of the following structure fields see SAP structure BAPI_DOC_DRAW.

```
#DOCUMENT.BASICDATA.DOCUMENTTYPE#
#DOCUMENT.BASICDATA.DOCUMENTNUMBER#
#DOCUMENT.BASICDATA.DOCUMENTVERSION#
#DOCUMENT.BASICDATA.DOCUMENTPART#
#DOCUMENT.BASICDATA.DESCRPTION#
#DOCUMENT.BASICDATA.USERNAME#
#DOCUMENT.BASICDATA.STATUSEXTERN#
#DOCUMENT.BASICDATA.STATUSINTERN#
#DOCUMENT.BASICDATA.LABORATORY#
#DOCUMENT.BASICDATA.ECNUMBER#
#DOCUMENT.BASICDATA.AUTHORITYGROUP#
```

For details of the following structure fields see SAP structure BAPI_DOC_DRAW2.

```
#DOCUMENT.BASICDATA2.DOCUMENTTYPE#
#DOCUMENT.BASICDATA2.DOCUMENTNUMBER#
#DOCUMENT.BASICDATA2.DOCUMENTVERSION#
#DOCUMENT.BASICDATA2.DOCUMENTPART#
#DOCUMENT.BASICDATA2.DESCRPTION#
#DOCUMENT.BASICDATA2.USERNAME#
#DOCUMENT.BASICDATA2.STATUSEXTERN#
#DOCUMENT.BASICDATA2.STATUSINTERN#
#DOCUMENT.BASICDATA2.LABORATORY#
#DOCUMENT.BASICDATA2.ECNUMBER#
#DOCUMENT.BASICDATA2.AUTHORITYGROUP#
#DOCUMENT.BASICDATA2.REVLEVEL#
```

Available inside a "Loop Document.Descriptions"/"Next Document.Descriptions" loop only:

```
#DOCUMENT.DESCRPTIONS.COUNT# Number of table lines
#DOCUMENT.DESCRPTIONS.DESCRPTION#
#DOCUMENT.DESCRPTIONS.LANGUAGE#
#DOCUMENT.DESCRPTIONS.LANGUAGEISO#
#DOCUMENT.DESCRPTIONS.TEXTINDICATOR#
```

See SAP structure BAPI_DOC_DRAT for details.

Available inside a "Loop Document.CharacteristicValues"/"Next Document.CharacteristicValues" loop only:

```
#DOCUMENT.CHARACTERISTICVALUES.COUNT# Number of table lines
#DOCUMENT.CHARACTERISTICVALUES.CLASSTYPE#
#DOCUMENT.CHARACTERISTICVALUES.CLASSNAME#
#DOCUMENT.CHARACTERISTICVALUES.CHARNAME#
#DOCUMENT.CHARACTERISTICVALUES.CHARVALUE#
#DOCUMENT.CHARACTERISTICVALUES.DELETEVALUE#
```

See SAP structure BAPI_CHARACTERISTIC_VALUES for details.

To directly access the value of a certain characteristic use:

```
#DOCUMENT.CHARACTERISTICVALUE.<NAME>#
```

where <NAME> is the internal SAP name of the characteristic, not the display name.

Available inside a "Loop Document.ClassAllocations"/"Next Document.ClassAllocations" loop only:

```
#DOCUMENT.CLASSALLOCATIONS.COUNT# Number of table lines
#DOCUMENT.CLASSALLOCATIONS.CLASSTYPE#
#DOCUMENT.CLASSALLOCATIONS.CLASSNAME#
#DOCUMENT.CLASSALLOCATIONS.STATUS#
#DOCUMENT.CLASSALLOCATIONS.STANDARDCLASS
#DOCUMENT.CLASSALLOCATIONS.DELETEALLOCATION#
#DOCUMENT.CLASSALLOCATIONS.ECNUMBER#
```

See SAP structure BAPI_CLASS_ALLOCATIONS for details.

Available inside a "Loop Document.Files2"/"Next Document.Files2" loop only:

```
#DOCUMENT.FILES2.COUNT# Number of table lines
#DOCUMENT.FILES2.ACTIVEVERSION#
#DOCUMENT.FILES2.APPLICATIONID#
#DOCUMENT.FILES2.CHANGEDAT#
#DOCUMENT.FILES2.CHECKEDIN#
#DOCUMENT.FILES2.CREATEDAT#
#DOCUMENT.FILES2.DELETEVALUE#
#DOCUMENT.FILES2.DESRIPTION#
#DOCUMENT.FILES2.DOCFILE#
#DOCUMENT.FILES2.DOCPATH#
#DOCUMENT.FILES2.DOCUMENTNUMBER#
#DOCUMENT.FILES2.DOCUMENTPART#
#DOCUMENT.FILES2.DOCUMENTTYPE#
#DOCUMENT.FILES2.DOCUMENTVERSION#
#DOCUMENT.FILES2.FILEID#
#DOCUMENT.FILES2.LANGUAGE#
#DOCUMENT.FILES2.ORIGINALTYPE#
#DOCUMENT.FILES2.SOURCEDATACARRIER#
#DOCUMENT.FILES2.STATUSEXTERN#
#DOCUMENT.FILES2.STATUSINTERN#
#DOCUMENT.FILES2.STATUSLOG#
#DOCUMENT.FILES2.STORAGECATEGORY#
#DOCUMENT.FILES2.WSAPPLICATION#
```

See SAP structure BAPI_DOC_FILES2 for details.

Available inside a "Loop Document.Links"/"Next Document.Links" loop only:

```
#DOCUMENT.LINKS.COUNT# Number of table lines
#DOCUMENT.LINKS.DELETEVALUE#
#DOCUMENT.LINKS.OBJECTTYPE#
#DOCUMENT.LINKS.OBJECTKEY#
#DOCUMENT.LINKS.DOCUMENTDIRECTION#
#DOCUMENT.LINKS.OBJECTDESCRIPTION#
```

See SAP structure BAPI_DOC_DRAD for details.

Available inside a "Loop Document.List"/"Next Document.List" loop only:

```
#DOCUMENT.LIST.COUNT# Number of table lines
#DOCUMENT.LIST.DOCUMENTTYPE#
#DOCUMENT.LIST.DOCUMENTNUMBER#
#DOCUMENT.LIST.DOCUMENTVERSION#
#DOCUMENT.LIST.DOCUMENTPART#
#DOCUMENT.LIST.DESCRPTION#
#DOCUMENT.LIST.USERNAME#
#DOCUMENT.LIST.STATUSEXTERN#
#DOCUMENT.LIST.STATUSINTERN#
#DOCUMENT.LIST.LABORATORY#
#DOCUMENT.LIST.ECNUMBER#
#DOCUMENT.LIST.AUTHORITYGROUP#
```

See SAP structure BAPI_DOC_DRAW for details.

Available inside a "Loop Document.Structure"/"Next Document.Structure" loop only:

```
#DOCUMENT.STRUCTURE.COUNT# Number of table lines
#DOCUMENT.STRUCTURE.DELETEVALUE#
#DOCUMENT.STRUCTURE.DOCUMENTTYPE#
#DOCUMENT.STRUCTURE.DOCUMENTNUMBER#
#DOCUMENT.STRUCTURE.DOCUMENTPART#
#DOCUMENT.STRUCTURE.DOCUMENTVERSION#
#DOCUMENT.STRUCTURE.QUANTITY#
```

See SAP structure BAPI_DOC_STRUCTURE for details.

Available inside a "Loop Document.StatusLog"/"Next Document.StatusLog" loop only:

```
#DOCUMENT.STATUSLOG.COUNT# Number of table lines
#DOCUMENT.STATUSLOG.AUDITFLAG1#
#DOCUMENT.STATUSLOG.AUDITFLAG2#
#DOCUMENT.STATUSLOG.LOGDATE#
#DOCUMENT.STATUSLOG.LOGFIELD#
#DOCUMENT.STATUSLOG.LOGTIME#
#DOCUMENT.STATUSLOG.STATUSEXTERN#
#DOCUMENT.STATUSLOG.STATUSINTERN#
#DOCUMENT.STATUSLOG.USERNAME#
```

See SAP structure BAPI_DOC_DRAP for details.

EC Master Data

```
#ECMASTER.ECNUMBER# Engineering change number
#ECMASTER.EXISTS# Equals "X" if EC master exists and space if it does not
```

For details of the following structure fields see SAP structure AENR_API01.

```
#ECMASTER.ECHEADER.CHANGENO#
#ECMASTER.ECHEADER.STATUS#
#ECMASTER.ECHEADER.AUTHGROUP#
#ECMASTER.ECHEADER.VALIDFROM#
#ECMASTER.ECHEADER.DESCRPT#
```

```
#ECMASTER.ECHEADER.REASONCHG#
#ECMASTER.ECHEADER.DELETIONMARK#
#ECMASTER.ECHEADER.INDATERULE#
#ECMASTER.ECHEADER.OUTDATERULE#
#ECMASTER.ECHEADER.FUNCTION#
#ECMASTER.ECHEADER.CHANGELEADER#
#ECMASTER.ECHEADER.EFFECTIVITYTYPE#
#ECMASTER.ECHEADER.OVERRIDINGMARK#
#ECMASTER.ECHEADER.RANK#
#ECMASTER.ECHEADER.RELEASEKEY#
#ECMASTER.ECHEADER.STATUSPROFILE#
#ECMASTER.ECHEADER.TECHREL#
#ECMASTER.ECHEADER.BASICCHANGE#
```

Function Module Call Data

```
#FUNCTIONMODULE.RFCRC# Return code of RFC call (0 means OK)
#FUNCTIONMODULE.ERRORINFO# Error message returned by the RFC environment in
                           case RFCRC <> 0
#FUNCTIONMODULE.EXCEPTION# Exception returned by the function module in
                           case RFCRC = 2
#FUNCTIONMODULE.EXPORTS.<NAME># Value of export parameter <NAME>
#FUNCTIONMODULE.IMPORTS.<NAME># Value of import parameter <NAME>
#FUNCTIONMODULE.CHANGES.<NAME># Value of changing parameter <NAME>
#FUNCTIONMODULE.TABLES.<NAME>.COUNT# Number of lines in table <NAME>
#FUNCTIONMODULE.TABLES.<NAME>.<NUMBER># Line number <NUMBER>
                                     of table <NAME>
```

Available inside a "Loop FunctionModule.Tables.<Name>"/"Next FunctionModule.Tables.<Name>" loop only:

```
#FUNCTIONMODULE.TABLES.<NAME># The respective line of table <NAME> inside the loop
```

Available inside a "Loop FunctionModule.<Name>"/"Next FunctionModule.<Name>" loop only:

```
#FUNCTIONMODULE.<NAME># The respective line of table <NAME> inside the loop
```

This loop is an abbreviation of the longer form mentioned above.

Material Data

```
#MATERIAL.MATERIAL# Material number
#MATERIAL.MATERIALEXTERNAL# External material number
#MATERIAL.MATERIALVERSION# Material version
#MATERIAL.MATERIALGUID# GUID of material
#MATERIAL.MATERIALTYPE# Material type, e. g. "HALB", "FERT"
#MATERIAL.INDUSTRYSECTOR# Industry sector, e. g. "A", "M"
#MATERIAL.EXISTS# Equals "X" if material exists and space if it does not
#MATERIAL.ISDELETED# Equals "X" if material is marked for deletion and space if it is not
#MATERIAL.BOMEXISTS# Equals "X" if material has a BOM (set by Material.OpenBOM and
                     Material.ReadBOM)
```

For details of the following structure fields see SAP structure BAPIMATDOA.

```
#MATERIAL.BASICDATA.MATLDESC#
#MATERIAL.BASICDATA.OLDMATNO#
#MATERIAL.BASICDATA.MATLTYPE#
#MATERIAL.BASICDATA.INDSECTOR#
#MATERIAL.BASICDATA.DIVISION#
#MATERIAL.BASICDATA.MATLGROUP#
#MATERIAL.BASICDATA.PRODHIER#
#MATERIAL.BASICDATA.BASICMATL#
#MATERIAL.BASICDATA.STDDESCR#
#MATERIAL.BASICDATA.LABDESIGN#
#MATERIAL.BASICDATA.PRODMEMO#
#MATERIAL.BASICDATA.PAGEFORMAT#
#MATERIAL.BASICDATA.CONTAINER#
#MATERIAL.BASICDATA.STORCONDS#
#MATERIAL.BASICDATA.TEMPCONDS#
#MATERIAL.BASICDATA.BASEUOM#
#MATERIAL.BASICDATA.EANUPC#
#MATERIAL.BASICDATA.EANCAT#
#MATERIAL.BASICDATA.SIZEDIM#
#MATERIAL.BASICDATA.GROSSWT#
#MATERIAL.BASICDATA.NETWEIGHT#
#MATERIAL.BASICDATA.UNITOFWT#
#MATERIAL.BASICDATA.VOLUME#
#MATERIAL.BASICDATA.VOLUMEUNIT#
#MATERIAL.BASICDATA.LENGTH#
#MATERIAL.BASICDATA.WIDTH#
#MATERIAL.BASICDATA.HEIGHT#
#MATERIAL.BASICDATA.UNITDIM#
#MATERIAL.BASICDATA.MANUMAT#
#MATERIAL.BASICDATA.MFRNO#
#MATERIAL.BASICDATA.BASEUOMISO#
#MATERIAL.BASICDATA.UNITOFWTISO#
#MATERIAL.BASICDATA.VOLUMEUNITISO#
#MATERIAL.BASICDATA.UNITDIMISO#
#MATERIAL.BASICDATA.CREATEDON#
#MATERIAL.BASICDATA.CREATEDBY#
#MATERIAL.BASICDATA.LASTCHNGE#
#MATERIAL.BASICDATA.CHANGEDBY#
#MATERIAL.BASICDATA.MATLCAT#
#MATERIAL.BASICDATA.EMPTIESBOM#
#MATERIAL.BASICDATA.BASICMATLNEW#
```

For details of the following structure fields see SAP structure BAPIMATDOC.

```
#MATERIAL.PLANTDATA.ISSUEUNIT#
#MATERIAL.PLANTDATA.PURGROUP#
```

For details of the following structure fields see SAP structure BAPIMATDOBEW.

```
#MATERIAL.VALUATIONDATA.CURRENCY#
#MATERIAL.VALUATIONDATA.CURRENCYISO#
#MATERIAL.VALUATIONDATA.MOVINGPR#
#MATERIAL.VALUATIONDATA.PRICECTRL#
```

```
#MATERIAL.VALUATIONDATA.PRICEUNIT#
#MATERIAL.VALUATIONDATA.STDPRICE#
```

Available inside a "Loop Material.Descriptions"/"Next Material.Descriptions " loop only:

```
#MATERIAL.DESCRPTIONS.COUNT# Number of table lines
#MATERIAL.DESCRPTIONS.LANGUAGE#
#MATERIAL.DESCRPTIONS.LANGUAGEISO#
#MATERIAL.DESCRPTIONS.DESCRPTION#
```

See SAP structure BAPIMAKT for details.

For details of the following structure fields see SAP structure STKO_API02.

```
#MATERIAL.BOMHEADER.ALTTEXT#
#MATERIAL.BOMHEADER.AUTHGROUP#
#MATERIAL.BOMHEADER.BASEQUAN#
#MATERIAL.BOMHEADER.BASEUNIT#
#MATERIAL.BOMHEADER.BOMGROUP#
#MATERIAL.BOMHEADER.BOMSTATUS#
#MATERIAL.BOMHEADER.BOMTEXT#
#MATERIAL.BOMHEADER.DELETEIND#
#MATERIAL.BOMHEADER.LABORATORY#
#MATERIAL.BOMHEADER.VALUE# A string holding the complete structure.
```

Available inside a "Loop Material.BOM"/"Next Material.BOM" loop only:

```
#MATERIAL.BOM.COUNT# Number of table lines
#MATERIAL.BOM.ITEMCATEG#
#MATERIAL.BOM.ITEMNO#
#MATERIAL.BOM.COMPONENT#
#MATERIAL.BOM.COMPQTY#
#MATERIAL.BOM.COMPUNIT#
#MATERIAL.BOM.FIXEDQTY#
#MATERIAL.BOM.ITEMTEXT1#
#MATERIAL.BOM.ITEMTEXT2#
#MATERIAL.BOM.SORTSTRING#
#MATERIAL.BOM.RELCOST#
#MATERIAL.BOM.RELENGIN#
#MATERIAL.BOM.RELPMAINT#
#MATERIAL.BOM.RELPROD#
#MATERIAL.BOM.RELSALES#
#MATERIAL.BOM.SPAREPART#
#MATERIAL.BOM.MATPROVIS#
#MATERIAL.BOM.BULKMAT#
#MATERIAL.BOM.RECALLOWD#
#MATERIAL.BOM.VALUE# A string holding the complete table line.
```

See SAP structure STPO_API02 for details.

Available inside a "Loop Material.List"/"Next Material.List" loop only:

```
#MATERIAL.LIST.COUNT# Number of table lines
#MATERIAL.LIST.MATERIAL#
```

```
#MATERIAL.LIST.MATERIALEXTERNAL#
#MATERIAL.LIST.MATERIALGUID#
#MATERIAL.LIST.MATERIALVERSION#
#MATERIAL.LIST.MATLDESC#
```

See SAP structure BAPIMATLST for details.

Project Data

```
#PROJECT.PROJECT# Project number (external)
#PROJECT.EXISTS# Equals "X" if project exists and space if it does not
```

For details of the following structure fields see SAP structure BAPI_PROJECT_DEFINITION.

```
#PROJECT.BASICDATA.PROJECTDEF#
#PROJECT.BASICDATA.DESRIPTION#
#PROJECT.BASICDATA.MASKID#
#PROJECT.BASICDATA.RESPONSIBLENO#
#PROJECT.BASICDATA.APPLICANTNO#
#PROJECT.BASICDATA.COMPCODE#
#PROJECT.BASICDATA.BUSAREA#
#PROJECT.BASICDATA.CONTROLLINGAREA#
#PROJECT.BASICDATA.PROFITCTR#
#PROJECT.BASICDATA.PROJECTCURRENCY#
#PROJECT.BASICDATA.PROJECTCURRENCYISO#
#PROJECT.BASICDATA.NETWORKASSIGNMENT#
#PROJECT.BASICDATA.START#
#PROJECT.BASICDATA.FINISH#
#PROJECT.BASICDATA.PLANT#
#PROJECT.BASICDATA.CALENDAR#
#PROJECT.BASICDATA.PLANBASIC#
#PROJECT.BASICDATA.PLANFCST#
#PROJECT.BASICDATA.TIMEUNIT#
#PROJECT.BASICDATA.TIMEUNITISO#
#PROJECT.BASICDATA.NETWORKPROFILE#
#PROJECT.BASICDATA.PROJECTPROFILE#
#PROJECT.BASICDATA.BUDGETPROFILE#
#PROJECT.BASICDATA.PROJECTSTOCK#
#PROJECT.BASICDATA.OBJECTCLASS#
#PROJECT.BASICDATA.STATISTICAL#
#PROJECT.BASICDATA.TAXJURCODE#
#PROJECT.BASICDATA.INTPROFILE#
#PROJECT.BASICDATA.WBSSCHEDPROFILE#
#PROJECT.BASICDATA.CSHBDGTPROFILE#
#PROJECT.BASICDATA.PLANPROFILE#
#PROJECT.BASICDATA.JOINTVENTURE#
#PROJECT.BASICDATA.RECOVERYIND#
#PROJECT.BASICDATA.EQUITYTYPE#
#PROJECT.BASICDATA.JVOBJECTTYPE#
#PROJECT.BASICDATA.JVJIBCLASS#
#PROJECT.BASICDATA.JVJIBSUBCLASSA#
#PROJECT.BASICDATA.OBJECTCLASSEXT#
#PROJECT.BASICDATA.FUNCAREA#
```

Available inside a "Loop Project.List"/"Next Project.List" loop only:

```
#PROJECT.LIST.COUNT# Number of table lines
#PROJECT.LIST.PROJECTDEFINITION#
#PROJECT.LIST.DESCRPTION#
```

See SAP structure BAPIPREXP for details.

Available inside a "Loop Project.WbsElements"/"Next Project.WbsElements" loop only:

```
#PROJECT.WBSELEMENTS.COUNT# Number of table lines
#PROJECT.WBSELEMENTS.WBSELEMENT#
#PROJECT.WBSELEMENTS.PROJECTDEFINITION#
#PROJECT.WBSELEMENTS.DESCRPTION#
#PROJECT.WBSELEMENTS.SHORTID#
#PROJECT.WBSELEMENTS.RESPONSIBLENO#
#PROJECT.WBSELEMENTS.APPLICANTNO#
#PROJECT.WBSELEMENTS.COMPCODE#
#PROJECT.WBSELEMENTS.BUSAREA#
#PROJECT.WBSELEMENTS.COAREA#
#PROJECT.WBSELEMENTS.PROFITCTR#
#PROJECT.WBSELEMENTS.PROJTYPE#
#PROJECT.WBSELEMENTS.NETWORKASSIGNMENT#
#PROJECT.WBSELEMENTS.COSTINGSHEET#
#PROJECT.WBSELEMENTS.OVERHEADKEY#
#PROJECT.WBSELEMENTS.CALENDAR#
#PROJECT.WBSELEMENTS.PRIORITY#
#PROJECT.WBSELEMENTS.EQUIPMENT#
#PROJECT.WBSELEMENTS.FUNCTIONALLOCATION#
#PROJECT.WBSELEMENTS.CURRENCY#
#PROJECT.WBSELEMENTS.CURRENCYISO#
#PROJECT.WBSELEMENTS.PLANT#
#PROJECT.WBSELEMENTS.USERFIELDKEY#
#PROJECT.WBSELEMENTS.USERFIELDCHAR201#
#PROJECT.WBSELEMENTS.USERFIELDCHAR202#
#PROJECT.WBSELEMENTS.USERFIELDCHAR101#
#PROJECT.WBSELEMENTS.USERFIELDCHAR102#
#PROJECT.WBSELEMENTS.USERFIELDQUAN1#
#PROJECT.WBSELEMENTS.USERFIELDUNIT1#
#PROJECT.WBSELEMENTS.USERFIELDUNIT1ISO#
#PROJECT.WBSELEMENTS.USERFIELDQUAN2#
#PROJECT.WBSELEMENTS.USERFIELDUNIT2#
#PROJECT.WBSELEMENTS.USERFIELDUNIT2ISO#
#PROJECT.WBSELEMENTS.USERFIELDCURR1#
#PROJECT.WBSELEMENTS.USERFIELDCHUKY1#
#PROJECT.WBSELEMENTS.USERFIELDCHUKY1ISO#
#PROJECT.WBSELEMENTS.USERFIELDCURR2#
#PROJECT.WBSELEMENTS.USERFIELDCHUKY2#
#PROJECT.WBSELEMENTS.USERFIELDCHUKY2ISO#
#PROJECT.WBSELEMENTS.USERFIELDDATE1#
#PROJECT.WBSELEMENTS.USERFIELDDATE2#
#PROJECT.WBSELEMENTS.USERFIELDFLAG1#
#PROJECT.WBSELEMENTS.USERFIELDFLAG2#
#PROJECT.WBSELEMENTS.OBJECTCLASS#
#PROJECT.WBSELEMENTS.STATISTICAL#
#PROJECT.WBSELEMENTS.TAXJURCODE#
```

```
#PROJECT.WBSELEMENTS.INTPROFILE#
#PROJECT.WBSELEMENTS.JOINTVENTURE#
#PROJECT.WBSELEMENTS.RECOVERYIND#
#PROJECT.WBSELEMENTS.EQUITYTYPE#
#PROJECT.WBSELEMENTS.JVOBJECTTYPE#
#PROJECT.WBSELEMENTS.JVJIBCLASS#
#PROJECT.WBSELEMENTS.JVJIBSUBCLASSA#
#PROJECT.WBSELEMENTS.DELETIONFLAG#
#PROJECT.WBSELEMENTS.OBJECTCLASSEX#
#PROJECT.WBSELEMENTS.WBSPLANNINGELEMENT#
#PROJECT.WBSELEMENTS.WBSACCOUNTASSIGNMENTELEMENT#
#PROJECT.WBSELEMENTS.WBSBILLINGELEMENT#
#PROJECT.WBSELEMENTS.RESPSBLCCTR#
#PROJECT.WBSELEMENTS.RESPSBLCCTRCONTROLLINGAREA#
#PROJECT.WBSELEMENTS.REQUESTCCTR#
#PROJECT.WBSELEMENTS.REQUESTCOMPCODE#
#PROJECT.WBSELEMENTS.REQUESTCCTRCONTROLLINGAREA#
#PROJECT.WBSELEMENTS.LOCATION#
#PROJECT.WBSELEMENTS.CHANGENO#
#PROJECT.WBSELEMENTS.INVESTPROFILE#
#PROJECT.WBSELEMENTS.RESANALKEY#
#PROJECT.WBSELEMENTS.WBSCCTRPOSTEDACTUAL#
#PROJECT.WBSELEMENTS.WBSBASICSTARTDATE#
#PROJECT.WBSELEMENTS.WBSBASICFINISHDATE#
#PROJECT.WBSELEMENTS.WBSFORECASTSTARTDATE#
#PROJECT.WBSELEMENTS.WBSFORECASTFINISHDATE#
#PROJECT.WBSELEMENTS.WBSACTUALSTARTDATE#
#PROJECT.WBSELEMENTS.WBSACTUALFINISHDATE#
#PROJECT.WBSELEMENTS.WBSBASICDURATION#
#PROJECT.WBSELEMENTS.WBSBASICDURUNIT#
#PROJECT.WBSELEMENTS.WBSBASICDURUNITISO#
#PROJECT.WBSELEMENTS.WBSFORECASTDURATION#
#PROJECT.WBSELEMENTS.WBSFORCASTDURUNIT#
#PROJECT.WBSELEMENTS.WBSFORECASTDURUNITISO#
#PROJECT.WBSELEMENTS.WBSACTUALDURATION#
#PROJECT.WBSELEMENTS.WBSACTUALDURUNIT#
#PROJECT.WBSELEMENTS.WBSACTUALDURUNITISO#
#PROJECT.WBSELEMENTS.WBSSCDBASICSTARTDATE#
#PROJECT.WBSELEMENTS.WBSSCDBASICFINISHDATE#
#PROJECT.WBSELEMENTS.WBSSCDFORECASTSTARTDATE#
#PROJECT.WBSELEMENTS.WBSSCDFORECASTFINISHDATE#
#PROJECT.WBSELEMENTS.WBSSCDACTUALSTARTDATE#
#PROJECT.WBSELEMENTS.WBSSCDACTUALFINISHDATE#
#PROJECT.WBSELEMENTS.FUNCAREA#
```

See SAP structure [BAPI_WBS_ELEMENT_EXP](#) for details.

Available inside a "Loop Project.WbsHierarchie"/"Next Project.WbsHierarchie" loop only:

```
#PROJECT.WBSHIERARCHIE.COUNT# Number of table lines
#PROJECT.WBSHIERARCHIE.WBSELEMENT#
#PROJECT.WBSHIERARCHIE.PROJECTDEFINITION#
#PROJECT.WBSHIERARCHIE.UP#
#PROJECT.WBSHIERARCHIE.DOWN#
#PROJECT.WBSHIERARCHIE.LEFT#
#PROJECT.WBSHIERARCHIE.RIGHT#
```

See SAP structure BAPI_WBS_HIERARCHIE for details.

Available inside a "Loop Project.WbsMilestones"/"Next Project.WbsMilestones" loop only:

```
#PROJECT.WBSMILESTONES.COUNT# Number of table lines
#PROJECT.WBSMILESTONES.MILESTONENUMBER#
#PROJECT.WBSMILESTONES.WBSELEMENT#
#PROJECT.WBSMILESTONES.MILESTONEUSAGE#
#PROJECT.WBSMILESTONES.DESCRPTION#
#PROJECT.WBSMILESTONES.SCHEDMILESTONEDATEBASIC#
#PROJECT.WBSMILESTONES.SCHEDMILESTONETIMEBASIC#
#PROJECT.WBSMILESTONES.SCHEDMILESTONEDATEFORECAST#
#PROJECT.WBSMILESTONES.SCHEDMILESTONETIMEFORECAST#
#PROJECT.WBSMILESTONES.FIXEDMILESTONEDATEBASIC#
#PROJECT.WBSMILESTONES.FIXEDMILESTONETIMEBASIC#
#PROJECT.WBSMILESTONES.FIXEDMILESTONEDATEFORECAST#
#PROJECT.WBSMILESTONES.FIXEDMILESTONETIMEFORECAST#
#PROJECT.WBSMILESTONES.ACTUALDATE#
#PROJECT.WBSMILESTONES.ACTUALTIME#
#PROJECT.WBSMILESTONES.LATESTDATESINDICATOR#
#PROJECT.WBSMILESTONES.OFFSETSTARTENDINDICATOR#
#PROJECT.WBSMILESTONES.OFFSETMILESTONEDATE#
#PROJECT.WBSMILESTONES.OFFSETMILESTONEDATEUNIT#
#PROJECT.WBSMILESTONES.OFFSETMILESTONEDATEUNITISO#
#PROJECT.WBSMILESTONES.OFFSETPERCENTAGE#
#PROJECT.WBSMILESTONES.TRENDANALYSISINDICATOR#
#PROJECT.WBSMILESTONES.EARNEDVALUEINDICATOR#
#PROJECT.WBSMILESTONES.PERCENTAGEOFCOMPLETION#
#PROJECT.WBSMILESTONES.SALESDOCDATEINIDICATOR#
#PROJECT.WBSMILESTONES.INVOICEPERCENTAGE#
#PROJECT.WBSMILESTONES.DELETIONFLAG#
```

See SAP structure BAPI_WBS_MILESTONE_EXP for details.

Available inside a "Loop Project.Networkactivities"/"Next Project.Networkactivities" loop only:

```
#PROJECT.NETWORKACTIVITIES.COUNT# Number of table lines
#PROJECT.NETWORKACTIVITIES.NETWORK#
#PROJECT.NETWORKACTIVITIES.ACTIVITY#
#PROJECT.NETWORKACTIVITIES.CONTROLKEY#
#PROJECT.NETWORKACTIVITIES.WORKCNTR#
#PROJECT.NETWORKACTIVITIES.PLANT#
#PROJECT.NETWORKACTIVITIES.DESCRPTION#
#PROJECT.NETWORKACTIVITIES.VENDORNO#
#PROJECT.NETWORKACTIVITIES.PRICE#
#PROJECT.NETWORKACTIVITIES.PRICEUNIT#
#PROJECT.NETWORKACTIVITIES.COSTELEM#
#PROJECT.NETWORKACTIVITIES.CURRENCY#
#PROJECT.NETWORKACTIVITIES.CURRENCYISO#
#PROJECT.NETWORKACTIVITIES.INFOREC#
#PROJECT.NETWORKACTIVITIES.PURCHORG#
#PROJECT.NETWORKACTIVITIES.PURGROUP#
#PROJECT.NETWORKACTIVITIES.MATLGROUP#
#PROJECT.NETWORKACTIVITIES.FLEXIBLEDURATION#
#PROJECT.NETWORKACTIVITIES.NUMBEROFCAPACITIES#
#PROJECT.NETWORKACTIVITIES.PERCENTOFWORK#
```

```
#PROJECT.NETWORKACTIVITIES.MILESTONE#
#PROJECT.NETWORKACTIVITIES.ACTTYPE#
#PROJECT.NETWORKACTIVITIES.ACTIVITYCOSTS#
#PROJECT.NETWORKACTIVITIES.PROJECTDEFINITION#
#PROJECT.NETWORKACTIVITIES.WBSELEMENT#
#PROJECT.NETWORKACTIVITIES.FACTORYCALENDAR#
#PROJECT.NETWORKACTIVITIES.DISTRIBUTINGKEY#
#PROJECT.NETWORKACTIVITIES.PRIORITY#
#PROJECT.NETWORKACTIVITIES.TAXJURCODE#
#PROJECT.NETWORKACTIVITIES.OBJECTCLASS#
#PROJECT.NETWORKACTIVITIES.PROFITCTR#
#PROJECT.NETWORKACTIVITIES.NOTMRPAPPLICABLE#
#PROJECT.NETWORKACTIVITIES.PROJECTSUMMARIZATION#
#PROJECT.NETWORKACTIVITIES.OPERATIONMEASUREUNIT#
#PROJECT.NETWORKACTIVITIES.OPERATIONMEASUREUNITISO#
#PROJECT.NETWORKACTIVITIES.PLNDDELRY#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMAL#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMALUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMALUNITISO#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUM#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUMUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUMUNITISO#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTTYPESTART#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTTYPEFINISH#
#PROJECT.NETWORKACTIVITIES.WORKACTIVITY#
#PROJECT.NETWORKACTIVITIES.UNWORK#
#PROJECT.NETWORKACTIVITIES.UNWORKISO#
#PROJECT.NETWORKACTIVITIES.EARLYSTARTDATE#
#PROJECT.NETWORKACTIVITIES.EARLYSTARTTIME#
#PROJECT.NETWORKACTIVITIES.EARLYFINISHDATE#
#PROJECT.NETWORKACTIVITIES.EARLYFINISHTIME#
#PROJECT.NETWORKACTIVITIES.LATESTSTARTDATE#
#PROJECT.NETWORKACTIVITIES.LATESTSTARTTIME#
#PROJECT.NETWORKACTIVITIES.LATESTFINISHDATE#
#PROJECT.NETWORKACTIVITIES.LATESTFINISHTIME#
#PROJECT.NETWORKACTIVITIES.FLOATFREE#
#PROJECT.NETWORKACTIVITIES.FLOATTOTAL#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTSTARTDATE#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTSTARTTIME#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTFINISHDATE#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTFINISHTIME#
#PROJECT.NETWORKACTIVITIES.USERFIELDKEY#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHAR201#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHAR202#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHAR101#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHAR102#
#PROJECT.NETWORKACTIVITIES.USERFIELDQUAN1#
#PROJECT.NETWORKACTIVITIES.USERFIELDUNIT1#
#PROJECT.NETWORKACTIVITIES.USERFIELDUNIT1ISO#
#PROJECT.NETWORKACTIVITIES.USERFIELDQUAN2#
#PROJECT.NETWORKACTIVITIES.USERFIELDUNIT2#
#PROJECT.NETWORKACTIVITIES.USERFIELDUNIT2ISO#
#PROJECT.NETWORKACTIVITIES.USERFIELDCURR1#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHUKY1#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHUKY1ISO#
#PROJECT.NETWORKACTIVITIES.USERFIELDCURR2#
#PROJECT.NETWORKACTIVITIES.USERFIELDCHUKY2#
```

```
#PROJECT.NETWORKACTIVITIES.USERFIELDCKY2ISO#
#PROJECT.NETWORKACTIVITIES.USERFIELDDATE1#
#PROJECT.NETWORKACTIVITIES.USERFIELDDATE2#
#PROJECT.NETWORKACTIVITIES.USERFIELDFLAG1#
#PROJECT.NETWORKACTIVITIES.USERFIELDFLAG2#
#PROJECT.NETWORKACTIVITIES.DELETIONFLAG#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMALFC#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMALFCUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONNORMALFCUNITISO#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUMFC#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUMFCUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONMINIMUMFCUNITISO#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTTYPEFINISHFC#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTTYPESTARTFC#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTSTARTDATEFC#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTSTARTTIMEFC#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTFINISHDATEFC#
#PROJECT.NETWORKACTIVITIES.CONSTRAINTFINISHTIMEFC#
#PROJECT.NETWORKACTIVITIES.EARLYSTARTDATEFC#
#PROJECT.NETWORKACTIVITIES.EARLYSTARTTIMEFC#
#PROJECT.NETWORKACTIVITIES.EARLYFINISHDATEFC#
#PROJECT.NETWORKACTIVITIES.EARLYFINISHTIMEFC#
#PROJECT.NETWORKACTIVITIES.LATESTSTARTDATEFC#
#PROJECT.NETWORKACTIVITIES.LATESTSTARTTIMEFC#
#PROJECT.NETWORKACTIVITIES.LATESTFINISHDATEFC#
#PROJECT.NETWORKACTIVITIES.LATESTFINISHTIMEFC#
#PROJECT.NETWORKACTIVITIES.FLOATFREEFC#
#PROJECT.NETWORKACTIVITIES.FLOATTOTALFC#
#PROJECT.NETWORKACTIVITIES.WORKFORECAST#
#PROJECT.NETWORKACTIVITIES.WORKACTUAL#
#PROJECT.NETWORKACTIVITIES.DURATIONSCHEДУLED#
#PROJECT.NETWORKACTIVITIES.DURATIONSCHEДУLEDFC#
#PROJECT.NETWORKACTIVITIES.DURATIONSCHEДУLEDUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONSCHEДУLEDUNITISO#
#PROJECT.NETWORKACTIVITIES.DURATIONCONFIRMFC#
#PROJECT.NETWORKACTIVITIES.DURATIONCONFIRMFCUNIT#
#PROJECT.NETWORKACTIVITIES.DURATIONCONFIRMFCUNITISO#
#PROJECT.NETWORKACTIVITIES.OBJECTCLASSEXТ#
#PROJECT.NETWORKACTIVITIES.FUNCAREA#
```

See SAP structure BAPI_NETWORK_ACTIVITY_EXP for details.

Other Data

Excel Data

```
#EXCEL.COL# The current column of the current sheet
#EXCEL.FILENAME# The filename of the current workbook without path
#EXCEL.FULLFILENAME# The filename of the current workbook with full path
#EXCEL.ROW# The current row of the current sheet
#EXCEL.SHEETNAME# The name of the current sheet
#EXCEL.SHEETS.COUNT# The number of sheets of the current workbook
```

#EXCEL.VALUE# The value of the cell in row #Excel.Row# and column #Excel.Col#
#EXCEL.VALUE (R,C) # The value of the cell in row R and column C

File Data

#FILE.EOF# Equals "X" if the end of file was reached and space otherwise
#FILE.LINE# The last line read by File.GetLine
#FILE.LINENUMBER# The line number of the last line read
#FILE.LIST.COUNT# The number of entries in a file list
#FILE.LIST.DIRECTORY# The directory from which the file list was created

Available inside a "Loop File.List"/"Next File.List" loop only:

#FILE.LIST.FILENAME# The name of the active file in a file list

Grid Data

#GRID.RESULT# The return value of the grid window ("OK" or "CANCEL") most recently closed
 #GRID.COL# The number of the column the cursor is placed on (runs from 1 to #Grid.Cols#)
 #GRID.COLS# The number of columns in the grid
 #GRID.COLSEL# If the user marked an area, this is beside #Grid.Col# the other area boundary
 #GRID.ROW# The number of the row the cursor is placed on (runs from 1 to #Grid.Rows#)
 #GRID.ROWS# The number of rows in the grid
 #GRID.ROWSEL# If the user marked an area, this is beside #Grid.Row# the other area boundary
 #GRID.VALUE# The value of the cell the cursor is placed on
 #GRID.VALUE (R,C) # The value of the cell in row R and column C.
 R runs from 1 to #Grid.Rows#, C from 1 to #Grid.Cols#

System Data

#SYSTEM.DATE# The present date (taken from operating system)
 #SYSTEM.TIME# The present time (taken from operating system)
 #SYSTEM.DECIMALSIGN# The decimal sign used by the operating system
 #SYSTEM.DATEFORMAT# The date format used by the operating system
 #SYSTEM.ENV.<NAME># The value of environment variable <NAME>
 #SYSTEM.ERRORDESCRIPTION# The error description in case of a fatal, non-recoverable error
 #SYSTEM.ERRORNUMBER# The error number in case of a fatal, non-recoverable error
 #SYSTEM.ERRORSOURCE# The error source in case of a fatal, non-recoverable error
 #SYSTEM.FUNCTIONRESULT# The return value of the function most recently called by
 CallExternalFunction
 #SYSTEM.HOMEDIR# The installation directory of the ENOVIA SmarTeam/SAP Adaptor exe
 files
 #SYSTEM.MESSAGERESULT# The return value of the last message box
 ("OK", "CANCEL", "ABORT", "YES" or "NO")
 #SYSTEM.TEMPORARYFILENAME# A temporary filename, unique for an entire action sequence
 (activity section)
 #SYSTEM.SPLITSTRING.COUNT# The number of items resulting from the last SplitString
 operation
 #SYSTEM.SPLITSTRING.<NUMBER># Item <NUMBER> resulting from the last SplitString
 operation. <NUMBER> runs from 1 to #System.Splitstring.Count#

Definitions

Definitions are self defined abbreviations for strings. Their names are not case-sensitive. Definitions can be created using the [Define](#) or [Set](#) function. The difference is that Set evaluates its arguments directly while Define stores a definition as it is without evaluation. Thus the action sequence

```
Define %TimeDef%, #System.Time#
Set %TimeSet%, #System.Time#
```

```
Sleep 1000
Message Information, Defined: %TimeDef% - Set: %TimeSet%
```

will give two time values with at least one second difference. The reason is that the arguments of the Set statement are evaluated when Set is executed whereas the arguments of the Define statement are only evaluated when the Message statement is processed which is - due to the "Sleep 1000" statement - at least one second later.

Definitions are inserted into the program code by string substitution. Any occurrence of the defined name will be substituted but if a name is not defined it will not be replaced by anything, not even by an empty string "". This can lead to unexpected results or error messages. Look at the following code for instance:

```
Define DATA, 500, CN_ID, CN_DESCRIPTION
Smarteam.GetData
Document.AddBasicData DRW, #Smarteam.Object.CN_ID#, 000, _
                        #Smarteam.Object.REVISION#
Material.AddBasicData %MATNUM%, HALB, M
Material.Save
```

If you run this code, you get the following error message: "Function 'Smarteam.Get500' not defined" because even string "Data" inside the function name is substituted. After having solved this problem, you run into a more serious one because no error is reported at first sight. But instead of CN_ID being set as SAP document number, CN_DESCRIPTION is used because CN_ID is substituted by CN_DESCRIPTION before the function is called and even before variables are inserted. Facing this problems it is recommended to choose a unique way of naming definitions. For instance, they can be named like %NAME%, _NAME_, %%_NAME or __NAME.

The next problem arises because %MATNUM% is not defined and therefore not replaced by anything, not even by an empty string. This means when performing the Material.Save, %MATNUM% is taken as material number and an SAP material named %MATNUM% will be created. This problem occurs mainly if it is not clear whether a definition is executed or not. This is for instance the case for Define or Set statements inside a loop. If the loop has no values to loop at, the Define or Set will not be reached. In this case it is recommended to use an Inside-the-loop marker like this:

```
Loop Smarteam.Structure
  Set %InsideLoop%, True
  Set %MATNUM%, #Smarteam.Structure.CN_PART_NUMBER#
  ...
Next Smarteam.Structure
If %InsideLoop% = True
```

```

...
Material.AddBasicData %MATNUM%
...
Endif

```

Generally spoken, the Define statement should be used as often as possible, and the Set statement should be used only if a value must be stored immediately. Thus the preferred task of the Set statement is to pick up values from inside a loop or to freeze time stamps. Define statements usually cannot be used inside loops (while definitions surely can) because they do not evaluate their arguments immediately. Thus,

```

Loop Smarteam.Structure
  Define %MATNUM%, #Smarteam.Structure.CN_ID#
  ...
Next Smarteam.Structure

```

will not yield the desired result because the Define statement will only store the definition but not pick up the CN_ID from the structure object. A correct use case, for instance, is

```

Define %MATNUM%, #Smarteam.Structure.CN_ID#
Loop Smarteam.Structure
  Material.AddBOMData L, %ITEMNO%, %MATNUM%, ...
  ...
Next Smarteam.Structure

```

Sample Configurations

Navigator and Preliminary

Like in other object-oriented programming languages all "objects" in a Adaptor script must be declared and created with function "New". All [activity functions](#) and [variables](#) of the object can then be accessed through the object's name. In order to increase the readability of the sample configurations all "New" statements have been left out and class names are used instead of object names. Thus the sample configurations show a "generic" code that has to be adopted to actually created objects.

SAP Documents, SAP Vaults and Storage Categories, Accessing ENOVIA SmarTeam Files

[Save document with classification to SAP and set other document versions to SAP status NV \(not valid\)](#)

[Create SAP document with internal number assignment SAP and store SAP document number to ENOVIA SmarTeam](#)

[Store file in SAP vault](#)

[Split ENOVIA SmarTeam filename into data carrier and residual filename before storing file in SAP vault](#)

[Change SAP document in dialog and transfer SAP description back to ENOVIA SmarTeam object](#)

[Display SAP product structure of SAP document](#)

[Create single level SAP document structure \(document BOM\) from ENOVIA SmarTeam structure](#)

[Copy all SAP object links to material from version "-" to the present version](#)

[Create an SAP object link between a document and a material master](#)

[Perform an ENOVIA SmarTeam life-cycle operation on certain SAP status](#)

[Complete document example with description, classification, object link, file and structure](#)

SAP Materials

[Select Material in SAP, read its details from SAP and save some data to the ENOVIA SmarTeam object](#)

[Store some material classification data in ENOVIA SmarTeam object](#)

[Save material classification](#)

[Write document description and document key to SAP material master using batch input](#)

[Write document descriptions, document key, and purchase group to SAP material master using function module](#)

[Add non-basic data like plant data, valuation data, tax classifications, units of measurement, long texts to a material master](#)

[Write ENOVIA SmarTeam memo field to material long text](#)

[Create a material master with internal SAP number assignment](#)

[Create material revision using batch input](#)

[Create material revision using function module](#)

SAP Bill of Material

[Create or update a BOM with item numbers; old BOM is deleted first](#)
[Delete all BOM items with item numbers greater 500](#)
[Update all CAD items in a BOM by means of CSAP_MAT_BOM... function modules \(1\)](#)
[Update all CAD items in a BOM by means of CSAP_MAT_BOM... function modules \(2\)](#)
[Update BOM items by means of CSAP_MAT_BOM... function modules \(3\)](#)
[Update all CAD items in a BOM by means of CAD_CREATE/CHANGE_BOM... function modules](#)
[Modify BOM in dialog mode](#)

SAP EC Masters

[Create or change EC master including long text using profile via batch input](#)
[Create or change EC master including long text via function module](#)

SAP Projects

[Create an SAP project](#)
[Search for an SAP project and read project definition details into ENOVIA SmarTeam object](#)
[Show all WBS elements of an SAP project](#)
[Show the structure of the WBS elements of an SAP project](#)

ENOVIA SmarTeam Objects, Structures, Queries

[Retrieve complete ENOVIA SmarTeam filename if file is stored in a ENOVIA SmarTeam vault](#)
[Copy a file from ENOVIA SmarTeam vault to ENOVIA SmarTeam work directory](#)
[Get the first three objects of class "Item" that are linked by a general link in link class "Documents Items Relation"](#)
[Running a ENOVIA SmarTeam query by means of an SmQuery](#)
[Running a ENOVIA SmarTeam query by means of an SmSimpleQuery](#)

Miscellaneous

[Call an external function which returns multiple values](#)
[Pass multiple arguments to the ENOVIA SmarTeam/SAP Adaptor through single argument ArgString](#)
[Call SAP function module; read values directly from SAP table](#)
[Select plant and valuation area from SAP table](#)
[Retrieve list of materials and write material information to file](#)
[Process all lines of an input file](#)
[Write some information to a temporary file, execute a program on the file, and delete the file when the program has terminated](#)
[Display values in a grid](#)
[Read a "BOM" from an Excel sheet](#)

SAP Documents, SAP Vaults and Storage Categories, Accessing ENOVIA SmarTeam Files

Save document with classification to SAP and set other document versions to SAP status NV (not valid)

```

Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%, #SmarTeam.TDM_DESCRIPTION#, , _
                        #SmarTeam.CN_SAPSTATUS.DESCRPTION#
Document.AddClassallocation 017, CL_CAD
Document.AddCharacteristicvalue 017, CL_CAD, CAD_DRAWINGNUMBER, %DOCNUM%
Document.AddCharacteristicvalue 017, CL_CAD, CAD_DESCRIPTION, #SmarTeam.TDM_DESCRIPTION#
Document.Save
    If #SAP.BapiStatusSummary# = E, #SAP.BapiStatusSummary# = A
        DisplayBapiMessages
    Else
        Message Information, Document "#Document.Documentnumber#" saved to SAP.
        Document.SetStatusOtherVersions , , , NV, Set by SMARTEAM, NotEqual
    Endif

```

Create SAP document with internal number assignment SAP and store SAP document number to ENOVIA SmarTeam

```

Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData DRW, *, 000, %VERSION%, #SmarTeam.TDM_DESCRIPTION#, , _
                        #SmarTeam.CN_SAPSTATUS.DESCRPTION#
Document.Create
If #SAP.BapiStatusSummary# = E, #SAP.BapiStatusSummary# = A
    DisplayBapiMessages
Else
    SmarTeam.SetData TDM_ID, #Document.Documentnumber#
    SmarTeam.Save
    SmarTeam.RefreshWindow
Endif

```

Store file in SAP vault

```

Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
Define %FULLFILENAME%, #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME#
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
Document.AddFile DRW, %DOCNUM%, 000, %VERSION%, , , SAPVault, \Lookup1{#SmarTeam.FILE_TYPE#}, _
                %FULLFILENAME%
Document.Checkin

[Lookup1]
6
DOC
7
XLS
8
PPT

```

9
ACD
10
TXT
11
AVI
12
BMP
16
A13

Remarks: 1) If the file already exists in the vault, use Document.CheckinReplace. The use of Document.Change is also possible. 2) For an explanation of the Lookup-Section see inline function [\Lookup{}](#).

Split ENOVIA SmarTeam filename into data carrier and residual filename before storing file in SAP vault

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
Define %FULLFILENAME%, #SmarTeam.DIRECTORY#\#SmarTeam.FILE_NAME#
SmarTeam.GetData
Document.GetFrontendtype #System.Env.HOSTNAME#
Document.EncodeDataCarrier %FULLFILENAME%
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
Document.AddFile DRW, %DOCNUM%, 000, %VERSION%, , #Document.DataCarrier#, SAPVault, TIF, _
#Document.DocFile#
Document.Checkin
```

Due to SAP requirements, an environment variable HOSTNAME must have been set to the local computer's name. Otherwise SAP data carriers cannot be resolved to paths and paths cannot be encoded to data carriers.

Change SAP document in dialog and transfer SAP description back to ENOVIA SmarTeam object

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
Document.ChangeDialog
If #SAP.DialogStatus# <> A
    SmarTeam.SetData TDM_DESCRIPTION, #Document.Basicdata.Description#
    SmarTeam.Save
Endif
```

Remark: Must be call with SAPGUI.

Display SAP product structure of SAP document

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
Document.DisplayProductStructure
```

Remark: Must be call with SAPGUI.

Create single level SAP document structure (document BOM) from ENOVIA SmarTeam structure

```
Define %DOCNUM_PAR%, ST#SmarTeam.TDM_ID#
Define %DOCNUM_SON%, ST#SmarTeam.Structure.TDM_ID#
Define %VERSION_PAR%, \LeftDotLeft2{#SmarTeam.REVISION#}
Define %VERSION_SON%, \LeftDotLeft2{#SmarTeam.Structure.REVISION#}
SmarTeam.GetData
SmarTeam.GetStructure
Document.AddBasicData DRW, %DOCNUM_PAR%, 000, %VERSION_PAR%
Loop SmarTeam.Structure
    Document.AddStructure DRW, %DOCNUM_SON%, 000, %VERSION_SON%
Next SmarTeam.Structure
Document.Save
```

Remark: If an attribute of the link object has to be accessed (for instance #SmarTeam.Structure.LinkObject.CN_QUANTITY# or #SmarTeam.Structure.LinkObject.CN_ITEMNUMBER#), function SmarTeam.GetStructure must be replaced by SmarTeam.GetStructurePlus.

Copy all SAP object links to materials from version "- " to the present version

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
Document.AddBasicData DRW, %DOCNUM%, 000, "- "
Document.GetObjectLinks MARA
Document.AddBasicData , , , %VERSION%
Document.Save
```

Create an SAP object link between a document and a material master

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
```

```

Define %MATNUM%, #SmarTeam.TDM_SAP_MAT_NUM#
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
If %MATNUM%! <> !
    Document.AddObjectLink MARA, %MATNUM%
    Document.Save
Endif

```

Perform an ENOVIA SmarTeam life-cycle operation on certain SAP status

```

Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData DRW, %DOCNUM%, 000, %VERSION%
Document.GetStatus
SmarTeam.SetData CN_SAP_STATUS, \LookUp1{#Document.StatusIntern#}
SmarTeam.Save
SmarTeam.RefreshWindow
If #Document.StatusIntern# = FR
    SmarTeam.LifeCycleOperation RELEASE
    SmarTeam.RefreshWindow
Endif

```

```

[LookUp1]
AA
1
IA
2
SP
3
FR
4

```

Remarks: 1) This operation can, for instance, be attached to an On-Enter-Screen event. The SmarTeam.RefreshWindow function updates the screen information after the new status has been saved to ENOVIA SmarTeam and after the life-cycle operation has been performed. 2) In this sample configuration CN_SAP_STATUS is considered as being a lookup so that a translation has to be done from SAP's status to ENOVIA SmarTeam's lookup object ID. This translation is done by function [\Lookup{}](#).

Complete document example with description, classification, object link, file and structure

```

Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Document.AddBasicData2 DRW, %DOCNUM%, 000, %VERSION%, #SmarTeam.TDM_DESCRIPTION#, , _
    #SmarTeam.CN_SAPSTATUS.DESCRPTION#

```

```

Document.AddDescription , EN, #SmarTeam.CN_DESCRIPTION_EN#
Document.AddDescription , DE, #SmarTeam.CN_DESCRIPTION_DE#

Document.AddClassallocation 017, CL_CAD
Document.AddCharacteristicvalue 017, CL_CAD, CAD_DRAWINGNUMBER, %DOCNUM%
Document.AddCharacteristicvalue 017, CL_CAD, CAD_DESCRIPTION , #SmarTeam.TDM_DESCRIPTION#

If #SmarTeam.TDMX_RELATED_ITEM_ID#! <> !
    Document.AddObjectLink MARA, #SmarTeam.TDMX_RELATED_ITEM_ID#
Endif

Set %FILENAME%, #SmarTeam.FILE_NAME#.pdf
SmarTeam.CopyFileExt C:\Temp\ , %FILENAME%, , , %FILENAME%
Document.AddFile2 , , , , , SAP-SYSTEM, PDF, C:\Temp\ , %FILENAME%
If #SmarTeam.Class.InternalName# = CATIA Part
    SmarTeam.GetStructure
    Loop SmarTeam.Structure
        Document.AddStructure DRW, ST#SmarTeam.Structure.TDM_ID#, 000, _
            \LeftDotLeft2{#SmarTeam.Structure.REVISION#}
    Next SmarTeam.Structure
Endif

Document.Save2
If #SAP.BapiStatusSummary# = E, #SAP.BapiStatusSummary# = A
    DisplayBapiMessages
Endif

```

SAP Materials

Select Material in SAP, read its details from SAP and save some data to the ENOVIA SmarTeam object

```

Material.SelectViaMatchcode
Check #SAP.DialogStatus# <> A
Material.AddBasicData , , , 1200, , , , , 1200
Material.GetDetail
SmarTeam.SetData TDM_SAP_MAT_NUM, #Material.Material#, TDM_DESCRIPTION, _
    #Material.Basicdata.MatlDesc#, CN_PRICE, #Material.Valuationdata.StdPrice#

SmarTeam.Save
SmarTeam.RefreshWindow

```

Remark: Must be call with SAPGUI.

Store some material classification data in ENOVIA SmarTeam object

```

Classification.AddData MARA, 001
Classification.AddIdentifier MATNR, #SmarTeam.TDM_SAP_MAT_NUM#
Classification.GetAllocations
Loop Classification.Allocations, %AllocCounter%

```

```

If %AllocCounter% = 1
  Set %MAT_CLASS%, \Trim{#Classification.Allocations.Class#}
  Set %HAS_CLASS%, True
Endif
Next Classification.Allocations
If %HAS_CLASS% = True
  Smarteam.SetData CN_MAT_CLASS, %MAT_CLASS%
  Classification.AddData MARA, 001, , , , E
  Classification.GetValidations
  Smarteam.SetData CN_SAP_DESCR_EN, #Classification.CharacteristicValue.DESIGNATION#
  Classification.AddData MARA, 001, , , , D
  Classification.GetValidations
  Smarteam.SetData CN_SAP_DESCR_DE, #Classification.CharacteristicValue.DESIGNATION#
  Smarteam.Save
Endif

```

Remark: To understand why the "Set %HAS_CLASS%, True" statement is used inside the loop, see the explanation about problems when using [definitions](#).

Save material classification

```

Classification.AddData MARA, 001, MATERIAL_CLASS_NAME
Classification.AddIdentifier MATNR, #Smarteam.TDM_SAP_MAT_NUM#
Classification.AddValidation MAT_CHARACT_1, #Smarteam.CN_ATTRIBUTE_1#
Classification.AddValidation MAT_CHARACT_2, #Smarteam.CN_ATTRIBUTE_2#
Classification.AddValidation MAT_CHARACT_3, #Smarteam.CN_ATTRIBUTE_3#
Classification.SaveValidation

```

Write document description and document key to SAP material master using batch input

```

Define %DOCNUM%, ST#Smarteam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#Smarteam.REVISION#}
Smarteam.GetData
Material.AddBasicData #Smarteam.TDM_SAP_MAT_NUM#
Material.AddBIData SAPLMGMM, 0060, RMMG1-MATNR, #Smarteam.TDM_SAP_MAT_NUM#, RMMG1-MBRSH, _
M, RMMG1-MTART, HALB
Material.AddBIData SAPLMGMM, 0070, MSICHTAUSW-KZSEL(01), X, MSICHTAUSW-KZSEL(02), X
Material.AddBIData SAPLMGMM, 4004, MAKT-MAKTX, #Smarteam.TDM_DESCRIPTION#, BDC_OKCODE, _
=SP02
Material.AddBIData SAPLMGMM, 4004, MARA-ZEINR, %DOCNUM%, MARA-ZEIAR, DRW, MARA-ZEIVR, _
%VERSION%, MARA-BLATT, 000, BDC_OKCODE, =BU
Material.SaveBI N, A

```

Remark: The code is valid for a standard 4.6C system. For a 4.5B system, screen numbers 4004 have to be replaced by 4000. For other or non-standard R/3 systems the batch input may be different.

Write document descriptions, document key, and purchase group to SAP material master using function module

```

Define %DOCNUM%, ST#SmarTeam.TDM_ID#
Define %VERSION%, \LeftDotLeft2{#SmarTeam.REVISION#}
SmarTeam.GetData
Material.AddHeadData #SmarTeam.TDM_SAP_MAT_NUM#
Material.AddClientData Document:=%DOCNUM%, DocType:=DRW, DocVers:=%VERSION%, PageNo:=000
Material.AddPlantData Plant:=1200, PurGroup:=002
Material.AddDescription E, , #SmarTeam.CN_DESCRIPTION_EN#
Material.AddDescription D, , #SmarTeam.CN_DESCRIPTION_DE#
Material.Save

```

Add non-basic data like plant, valuation and sales data, tax classifications, units of measure and long texts to a material master

```

Material.AddHeadData #SmarTeam.TDM_SAP_MAT_NUM#, HALB, M, X, X, X, X, , , , , X
Material.AddClientData BaseUom:=#SmarTeam.CN_BASEUOM#, Division:=01, CadId:=X, _
NetWeight:=#SmarTeam.CN_NETWEIGHT#, UnitOfWt:=#SmarTeam.CN_UNITOFWT#
Material.AddPlantData ProcType:=E, MrpType:=PD, SmKey:=001, PurGroup:=D01, LotSizeKey:=EX, _
ProductionScheduler:=150, AvailCheck:=ZA, PlanStrgp:=21, DepReqId:=1, _
MrpCtrlr:=MNA, MrpGroup:=MIN1
Material.AddValuationData ValArea:=0100, ValType:= , ValClass:=0200, PriceCtrl:=S, PriceUnit:=1, _
StdPrice:=0.00, QtyStruct:=X, OrigMat:=X
Material.AddSalesData MatPrGrp:=E, AcctAssgt:=01, ItemCat:=ZHW3, Matlstats:=1, _
SalesOrg:=0001, DistrChan:=01
Material.AddTaxClassifications CH, , MWST, 1
Material.AddTaxClassifications US, , UTXJ, 1
Material.AddUnitOfMeasure #SmarTeam.CN_BASEUOM#, , , , , , , , #SmarTeam.CN_VOLUME#, _
#SmarTeam.CN_UNITOFVOL#, , #SmarTeam.CN_GROSSWEIGHT#, _
#SmarTeam.CN_UNITOFWT#

Material.AddLongtext Material, #SmarTeam.TDM_SAP_MAT_NUM#, IVER, E, , /, Supplier
Material.AddLongtext Material, #SmarTeam.TDM_SAP_MAT_NUM#, IVER, E, , /, #SmarTeam.CN_SUPPLIER#

```

Write ENOVIA SmarTeam memo field to material long text

```

SplitString #S.CN_LONGTEXT#, \Hex{0D}\Hex{0A}
Set %Sep%, " "
DoLoop %Item%, 1, #System.SplitString.Count#
    M.AddLongtext Material, #S.TDM_SAP_MAT_NUM#, GRUN, I, , %Sep%, \Unmask, _
    {#System.SplitString.%Item%#}
    Set %Sep%, /
DoNext

```

Remarks: 1) Each line of an SAP long text has to be defined individually. Therefore it is not possible to transfer an entire memo field at once. The memo field has to be split into individual lines at "Carriage Return"/"Line Feed" characters before. "Carriage Return"/"Line Feed" characters can be accessed by \Hex{0D}\Hex{0A}. 2) Users have to take care that an individual

line does not reach maximum size allowed in SAP (132) because lines are truncated otherwise.
 3) See documentation of function module BAPI_MATERIAL_SAVEDATA for correct values of parameters <ApplObject>, <TextName> and <TextId>.

Create a material master with internal SAP number assignment

```
Material.AddBasicData    , HALB, M
Material.GetInternalNumber
Smarteam.SetData    TDM_SAP_MAT_NUM, #Material.Material#
Material.AddHeadData    #Material.Material#, HALB, M
Material.AddClientData    BaseUom:=ST, MatlGroup:=99, Division:=01
Material.AddDescription    , DE, (Nummer reserviert)
Material.AddDescription    , EN, (Number reserved)
Material.Save
```

Create material revision using batch input

```
Smarteam.GetData
BatchInput.AddRecord    SAPLCCRL, 0100, X
BatchInput.AddRecord    , , , RC29A-MATNR, #Smarteam.TDM_SAP_MAT_NUM#
BatchInput.AddRecord    SAPLCCRL, 0110, X
BatchInput.AddRecord    , , , RC29A-AENNR, #Smarteam.CN_CHANGE_NUMBER#
BatchInput.AddRecord    , , , RC29A-REVLV, \Left2{#Smarteam.REVISION#}
BatchInput.AddRecord    , , , CSDATA-XFELD, X
BatchInput.AddRecord    , , , BDC_OKCODE, =BUCH
BatchInput.AddRecord    SAPLCCCN, 0301, X
BatchInput.AddRecord    , , , RC29A-OITXT, #Smarteam.CN_CHANGE_TEXT#
BatchInput.AddRecord    , , , BDC_OKCODE, =CLWI
BatchInput.CallTransaction    CC11, , N, S
```

Create material revision using function module

```
Material.AddBasicData    #Smarteam.TDM_SAP_MAT_NUM#, , , , , \Left2{#Smarteam.REVISION#}, _
                        #Smarteam.CN_CHANGE_NUMBER#
Material.CreateRevision
```

SAP Bill of Material

Create or update a BOM with item numbers; old BOM is deleted first

```
Material.AddBasicData    #Smarteam.TDM_SAP_MAT_NUM#, , , , 2
Material.DeleteBOM
Loop    Smarteam.Structure
```

```

Set %ITEMNO%, \Eval{%ITEMNO% + 10}
Material.AddBOMData L, %ITEMNO%, #SmarTeam.Structure.TDM_ID#, #SmarTeam.Structure.CN_QUANTITY#, _
                    #SmarTeam.Structure.CN_QUANTITYUNIT#
Material.AppendBOMData
Next SmarTeam.Structure
Material.SaveBOM X, X

```

Delete all BOM items with item numbers greater 500

```

Material.AddBasicData #SmarTeam.TDM_SAP_MAT_NUM#, , , 2 , , , #System.Date#
Material.OpenBOM
Loop Material.BOM, %ItemCounter%
If #Material.BOM.ItemNo# >N> 500
    Material.SelectBOMItem %ItemCounter%
    Material.AddBOMDataExt FIDelete:=X
    Material.ModifyBOMItem
Endif
Next Material.BOM
Material.CloseBOM

```

Remark: In the If condition (#Material.BOM.ItemNo# >N> 500) a numerical comparison (>N>) has to be forced because SAP always returns a 4 character string as item number, and in this case, 0800 is not greater than 500 using string comparison.

Update all CAD items in a BOM by means of CSAP_MAT_BOM... function modules (1)

```

SmarTeam.GetData
SmarTeam.GetStructurePlus
Check #SmarTeam.Structure.Count# >N> 0
# When working without change number, Material.ReadBOM needs both, valid-from and valid-to dates
Material.AddBasicData #SmarTeam.TDM_SAP_MAT_NUM#, , , 2 , , , #System.Date#, 31.12.9999
# When working with change number, leave valid-from and valid-to dates empty
Material.AddBasicData #SmarTeam.TDM_SAP_MAT_NUM#, , , 2 , , , #SmarTeam.CN_EC_NUMBER#
Material.ReadBOM
If #SAP.BapiStatusSummary# = E
    DisplayBapiMessages
    Exit 1
Endif
Loop Material.BOM
    Material.UpdateBOMItem FIDelete:=X, #Material.BOM.CAD_IND# = X
Next Material.BOM
Loop SmarTeam.Structure
    Material.AddBOMData L, #SmarTeam.Structure.CN_ITEMNO#, #SmarTeam.Structure.TDM_SAP_MAT_NUM#, _
                    #SmarTeam.Structure.CN_QUANTITY#, ST
    Material.AddBOMDataExt CAD_IND:=X
    Material.AppendBOMData
Next SmarTeam.Structure
Material.SaveBOM X, X, , X

```

Remarks:

- 1) This approach scans the BOM item table read into internal memory just before. It marks all BOM items with CAD indicator for deletion. Then it adds all ENOVIA SmarTeam structure items to the BOM item table and calls CSAP_MAT_BOM_MAINTAIN updating the entire BOM in a single call.
- 2) If you need to work with a change number, customize the change number in Material.AddBasicData but leave the valid-from date empty.
- 3) If you discover problems with this approach to update BOMs in SAP, use the ABAP program "z_test_csap_bom_maintain.txt" in the SAP Adaptor for direct testing inside SAP or move to the other approach based on CAD_CREATE/CHANGE_BOM... function modules.

Update all CAD items in a BOM by means of CSAP_MAT_BOM... function modules (2)

```

SmarTeam.GetData
SmarTeam.GetStructurePlus
Check #SmarTeam.Structure.Count# >N> 0
# When working without change number, Material.OpenBOM needs valid-from date only
Material.AddBasicData #SmarTeam.TDM_SAP_MAT_NUM# , , , 2 , , , #System.Date#
# When working with change number, leave valid-from date empty
Material.AddBasicData #SmarTeam.TDM_SAP_MAT_NUM# , , , 2 , , , #SmarTeam.CN_EC_NUMBER#
Material.OpenBOM
If #SAP.BapiStatusSummary# = E
    DisplayBapiMessages
    Exit 1
Endif
Loop Material.BOM, %ItemCounter%
If #Material.BOM.CAD_IND# = X
    Material.SelectBOMItem %ItemCounter%
    Material.AddBOMDataExt FIDelete:=X
    Material.ModifyBOMItem
Endif
Next Material.BOM
Material.CloseBOM
Material.ClearBOMData
Loop SmarTeam.Structure
    Material.AddBOMData L, #SmarTeam.Structure.CN_ITEMNO#, #SmarTeam.Structure.TDM_SAP_MAT_NUM#, _
        #SmarTeam.Structure.CN_QUANTITY#, ST
    Material.AddBOMDataExt CAD_IND:=X
    Material.AppendBOMData
Next SmarTeam.Structure
Material.SaveBOM X, X, , X

```

Remarks:

- 1) Unlike the previous approach the BOM is opened for update. Then all BOM items with CAD indicator are deleted from this BOM using repetitive calls to function CSAP_BOM_ITEM_MAINTAIN. After these changes have been committed, the internal BOM tables are cleared, all ENOVIA SmarTeam structure items are added and the new BOM items are saved with a single call of CSAP_MAT_BOM_MAINTAIN.
- 2) If you need to work with a change number, customize the change number in Material.AddBasicData but leave the valid-from date empty.

3) If you discover problems with this approach to update BOMs in SAP, use the ABAP program `z_test_csap_bom_maintain.txt` in the SAP Adaptor for direct testing inside SAP or move to the other approach based on `CAD_CREATE/CHANGE_BOM...` function modules.

Update BOM items by means of `CSAP_MAT_BOM...` function modules (3)

```

Smarteam.GetData
Smarteam.GetStructurePlus
# When working without change number, we need valid-from and valid-to dates
Material.AddBasicData #Smarteam.TDM_SAP_MAT_NUM#, , , 2, , , #System.Date#, 31.12.9999
# When working with change number, leave valid-from and valid-to dates empty
Material.AddBasicData #Smarteam.TDM_SAP_MAT_NUM#, , , 2, , , #Smarteam.CN_EC_NUMBER#
Material.ReadBOM
If #SAP.BapiStatusSummary# = E
    DisplayBapiMessages
    Exit 1
Endif
Loop Smarteam.Structure
    Material.AddBOMData L, #Smarteam.Structure.CN_ITEMNO#, #Smarteam.Structure.TDM_SAP_MAT_NUM#, _
        #Smarteam.Structure.CN_QUANTITY#, ST
    Material.AddBOMDataExt CAD_IND:=X
    Material.AppendBOMData
Next Smarteam.Structure
Material.UpdateBOM X, X, , X, COMPONENT, COMP_QTY, ITEM_NO;ITEM_TEXT1, ITEM_NO <N< 500

```

Explanation: If we compare the documentation of function `Material.UpdateBOM`, we find `<IdentifyAttributes>=COMPONENT`, `<UpdateAttributes>=COMP_QTY;ITEM_TEXT1`, `<PreserveAttributes>=ITEM_NO;ITEM_TEXT2` and `<UpdateConditions>=ITEM_NO <N< 500`. Parameter `<IdentifyAttributes>` determines whether a BOM item is equal to an already existing one. In this example it means that if a BOM item has the same value in field `COMPONENT` than an existing item, this item will update the existing one. New items will be added to the BOM while existing BOM items that are not found in the new BOM are deleted if they satisfy `<UpdateConditions>`. That means parameter `<UpdateConditions>` controls which existing BOM items are considered as non-CAD or non-Smarteam item that should not be touched by SAP Toolkit. In this example items with item number greater or equal 500 are considered as manually maintained items that should not be deleted if they are not found in the new CAD or Smarteam BOM. Parameter `<UpdateAttributes>` controls whether an item that is marked for update (because the new item is already contained in the existing BOM) is actually updated. Only if at least one field mentioned in `<UpdateAttributes>` is different, the item is actually updated. In this example the BOM item is only updated if the quantity has changed. If an item is to be updated, parameter `<PreserveAttributes>` controls which fields of the existing item are going to be preserved. In this case the item number and the first item text are not updated but taken from the existing component.

Let's consider the following BOMs. "Existing BOM before" is the BOM as found in SAP before update. "New BOM" is the BOM coming from CAD/Smarteam. And "BOM after update" is the BOM in SAP after update.

BOM	Item no.	Component	Quantity	Item text 1	Item text 2
Existing BOM before, 1st child	0010	PART-1	1	text-1 part-1	text-2 part-1
Existing BOM before, 2nd child	0020	PART-2	1	text-1 part-2	text-2 part-2
Existing BOM before, 3rd child	0030	PART-3	5	text-1 part-3	text-2 part-3
Existing BOM before, 4th child	1000	PART-4	1	text-1 part-4	text-2 part-4
New BOM, 1st child	0010	PART-1	1	new text-1 part-1	new text-2 part-1
New BOM, 2nd child	0020	PART-3	2	new text-1 part-3	new text-2 part-3
New BOM, 3rd child	0050	PART-5	10	new text-1 part-5	new text-2 part-5
BOM after update, 1st child	0010	PART-1	1	text-1 part-1	text-2 part-1
BOM after update, 2nd child	0030	PART-3	2	text-1 part-3	new text-2 part-3
BOM after update, 3rd child	0050	PART-5	10	new text-1 part-5	new text-2 part-5
BOM after update, 4th child	1000	PART-4	1	text-1 part-4	text-2 part-4

PART-1 is contained in old and new BOMs and is therefore a candidate for update. Actually, it remains unchanged because update attribute COMP_QTY did not change.
PART-2 and PART-4 are not contained in new BOM but only PART-2 is deleted because only PART-2 fulfills update condition "item number 0020 < 0500". PART-4 does not fulfill the update condition and is therefore not considered as CAD/Smarteam part.
PART-3 is contained in old and new BOMs and is therefore a candidate for update. And in fact, it is updated because update attribute COMP_QTY has changed. During the update, fields item number and item text 1 remain unchanged because they are mentioned as preserved attributes.
PART-5 finally is added to the BOM as new child.

Remarks: 1) This procedure also works with change numbers. 2) BOM sub items are supported too.

Update all CAD items in a BOM by means of CAD_CREATE/CHANGE_BOM... function modules

```
Smarteam.GetData
Smarteam.GetStructurePlus
Check  #Smarteam.Structure.Count# >N> 0
# When working without change number
```

```

Material.AddBasicData  #Smarteam.TDM_SAP_MAT_NUM#,,,2,,,#System.Date#
# When working with change number, leave valid-from date empty
Material.AddBasicData  #Smarteam.TDM_SAP_MAT_NUM#,,,2,,,#Smarteam.CN_EC_NUMBER#
Material.AddBOMHeaderData2
Material.ReadBOM2
Material.ClearBOMData2
Material.AddBOMHeaderData2  ,,,,,,X
Set  %POSNR%, 500
Loop  Smarteam.Structure
    Material.AddBOMData2  L, %POSNR%, #Smarteam.Structure.TDM_SAP_MAT_NUM#, _
                        #Smarteam.Structure.CN_QUANTITY#, ST,,,,,,X
    Material.AppendBOMData2
    Set  %POSNR%, \Eval{%POSNR% + 10}
Next  Smarteam.Structure
Material.SaveBOM2

```

Remarks:

- 1) In this approach the actual update of the BOM is done by function module **CAD_CHANGE_BOM_WITH_SUB_ITEMS** so that ENOVIA SmarTeam structure items are just collected and then passed to that function.
- 2) The CAD Indicator for BOM items must be active in this scenario. You find it in SAP customizing transaction SPRO under "Production->Basic Data->Bill of Material->Control Data for Bills of Material->Define Modification Parameters". Look at checkbox "CAD actice" at the bottom of the screen.
- 3) If you need to work with a change number, customize the change number in **Material.AddBasicData** but leave the valid-from date empty.
- 4) If you discover problems with this approach to update BOMs in SAP, use the ABAP program "z_test_cad_bom_sub_items.txt" in the Adaptor for direct testing inside SAP or move to the other approach based on **CSAP_MAT_BOM...** function modules.

Modify BOM in dialog mode

```

Smarteam.GetData
BatchInput.AddSetGetParameters  MAT, #Smarteam.TDM_SAP_MAT_NUM#, CSV, 2
BatchInput.CallTransaction  CS02, X

```

SAP EC Masters

Create or change EC master including long text using profile via batch input

```

Smarteam.GetData
BatchInput.AddRecord  SAPMC29C, 0100, X
BatchInput.AddRecord  ,,, RC29A-AENNR, #Smarteam.CN_CHANGE_NUMBER#
BatchInput.AddRecord  ,,, RC29A-AEPRO, CAD
BatchInput.AddRecord  SAPMC29C, 0010, X
BatchInput.AddRecord  ,,, RC29A-AETXT, Design Change via Smarteam
BatchInput.AddRecord  ,,, RC29A-AEGRU, Design Change via Smarteam
BatchInput.AddRecord  ,,, RC29A-AENST, 01
BatchInput.AddRecord  ,,, RC29A-DATUV, \Left10{#Smarteam.EFFECTIVE_FROM#}

```

```

BatchInput.AddRecord      , , , BDC_OKCODE, "LTSS"
BatchInput.AddRecord      SAPLSTXX, 1100, X
SplitString               #Smarteam.TDM_DESCRIPTION#, \Hex{0D}\Hex{0A}
Set                       %N%, #System.SplitString.Count#
DoLoop                    %I%, 1, %N%
    BatchInput.AddRecord  , , , RSTXT-TXPARGRAPH(\Eval{%I% + 1}), /
    BatchInput.AddRecord  , , , RSTXT-TXLINE(\Eval{%I% + 1}), #System.SplitString.%I%#
DoNext
Set                       %N%, \Eval{#System.SplitString.Count# + 2}
DoLoop                    %I%, %N%, 20
    BatchInput.AddRecord  , , , RSTXT-TXPARGRAPH(%I%),
    BatchInput.AddRecord  , , , RSTXT-TXLINE(%I%),
DoNext
BatchInput.AddRecord      , , , BDC_OKCODE, "TXVB"
BatchInput.AddRecord      SAPLSTXX, 1100, X
BatchInput.AddRecord      , , , BDC_OKCODE, "TXBA"
BatchInput.AddRecord      SAPMC29C, 0010, X
BatchInput.AddRecord      , , , BDC_OKCODE, "FCBU"
ECMaster.AddHeaderData    #Smarteam.CN_CHANGE_NUMBER#
ECMaster.ExistenceCheck
If #ECMaster.Exists# = X
    BatchInput.CallTransaction CC02, , N, S
Else
    BatchInput.CallTransaction CC01, , N, S
Endif

```

Create or change EC master including long text via function module

```

Smarteam.GetData
ECMaster.AddHeaderData    #Smarteam.CN_CHANGE_NUMBER#, 01, , #Smarteam.EFFECTIVE_FROM#, _
                        Design Change via Smarteam, Design Change via Smarteam
ECMaster.AddObjectType    BOM, X, X, X
ECMaster.AddObjectType    DOC, X, X, X
ECMaster.AddObjectType    MAT, X, X, X
SplitString               #Smarteam.TDM_DESCRIPTION#, \Hex{0D}\Hex{0A}
Set                       %N%, #System.SplitString.Count#
DoLoop                    %I%, 1, %N%
    ECMaster.AddTextLine /, #System.SplitString.%I%#
DoNext
ECMaster.SaveHeader

```

SAP Projects

Create an SAP project

```

Project.AddBasicData      #Smarteam.TDM_ID#, #Smarteam.TDM_DESCRIPTION#, 0001
Project.Create

```

Search for an SAP project and read project definition details into ENOVIA SmarTeam object

```

Project.GetList #SmarTeam.TDM_ID#, , 15
If #Project.List.Count# = 0
    Message E, No projects matching search string "#SmarTeam.TDM_ID#" found.
    Exit
Endif
Loop Project.List
    Grid.AddLine #Project.List.ProjectDefinition#, #Project.List.Description#
Next Project.List
Grid.SetColFontBold 1
Grid.SetColWidth 2000, 3000
Grid.SetProperties 6000, 5500, 0, 0, X
Grid.Sort 1, Ascending
Grid.Show Select Project, Please choose a project by double clicking it., X
Check #Grid.Result# = OK
Set %R%, #Grid.Row#
Project.AddBasicData #Grid.Value(%R%,1)#
Project.GetDetail
SmarTeam.Setdata TDM_ID, #Grid.Value(%R%,1)#, TDM_DESCRIPTION, #Grid.Value(%R%,2)#, _
    CN_START_DATE, \WinDateFromSapDate{#Project.BasicData.Start#}, _
    CN_TARGET_DATE, \WinDateFromSapDate{#Project.BasicData.Finish#}, _
    CN_MANAGER, #Project.BasicData.ResponsibleNo#

```

Remark: In order to keep the execution time and the result list small, it is assumed that the user enters a search pattern like *4711/R* into the TDM_ID field before pressing the search-project button. After execution this field is filled with the project ID found.

Show all WBS elements of an SAP project

```

Project.AddBasicData #SmarTeam.TDM_ID#
Project.GetDetailEx
Grid.AddLine WBS Element, Description, Type, Costing Sheet, Requesting Cost Center
Loop Project.WbsElements
    Grid.Addline #Project.WbsElements.WBSELEMENT#, #Project.WbsElements.DESCRPTION#, _
        #Project.WbsElements.PROJTYPE#, #Project.WbsElements.COSTINGSHEET#, _
        #Project.WbsElements.REQUESTCCTR#
Next Project.WbsElements
Grid.SetColFontBold 1
Grid.SetRowFontBold 1
Grid.SetColWidth 2000, 3000, 500, 2000, 2000
Grid.SetProperties 6000, 9500, 1, 1, X
Grid.Show WBS Elements, SAP System: #SAP.Destination#\, Client: #SAP.Client# _
    \n\nWBS Elements of Project #SmarTeam.TDM_ID#, , , X, X

```

Show the structure of the WBS elements of an SAP project

```
Project.AddBasicData #SmarTeam.TDM_ID#
Project.GetDetailEx
Grid.AddLine WBS Element, Up, Down, Left, Right
Loop Project.WbsHierarchie
  Grid.Addline #Project.WbsHierarchie.WBSELEMENT#, _
               #Project.WbsHierarchie.UP#, #Project.WbsHierarchie.DOWN#, _
               #Project.WbsHierarchie.LEFT#, #Project.WbsHierarchie.RIGHT#
Next Project.WbsHierarchie
Grid.SetColFontBold 1
Grid.SetRowFontBold 1
Grid.SetColWidth 2000, 2000, 2000, 2000, 2000
Grid.SetProperties 6000, 10000, 1, 1, X
Grid.Show WBS Hierarchie, SAP System: #SAP.Destination#, Client: #SAP.Client# _
          \n\nWBS Hierarchie of Project #SmarTeam.TDM_ID#, , , X, X
```

ENOVIA SmarTeam Objects, Structures, Queries

Retrieve complete ENOVIA SmarTeam filename if file is stored in a ENOVIA SmarTeam vault

```
Define %FILENAME%, _
  \#SmarTeam.VAULT_OBJECT_ID.SRV_OBJECT_ID.SRV_SHARE_UNIT##SmarTeam.VAULT_OBJECT_ID.ROOT_
  DIR_ON_SRV#\#SmarTeam.FILE_NAME#
```

Remark: After attribute evaluation, %FILENAME% will be something like "`\\server\path\filename.ext`". Due to the fact that "\" is the escape character for masking special characters, "`\\server\path\filename.ext`" would be passed to the according function. To restore the eliminated backslash, another backslash has to be inserted in front of the string. Naturally this backslash (and the one between pathname and filename) has to be masked itself, because the sequence "\#" would otherwise be interpreted as a masked Hash sign.

Copy a file from ENOVIA SmarTeam vault to ENOVIA SmarTeam work directory

```
SmarTeam.CopyFile #SmarTeam.LocalConfig.USER_DIR#, #SmarTeam.FILE_NAME#, ReadOnly
```

Get the first three objects of class "Item" that are linked by a general link in link class "Documents Items Relation"

```
Define %DOCNUM%, ST#SmarTeam.TDM_ID#
SmarTeam.GetLinks Item, Documents Items Relation
Loop SmarTeam.Links, %SLCounter%
  If %SLCounter% <= 3
    Message Information, Linked object No. %SLCounter% - _
            Class ID: #SmarTeam.Links.CLASS_ID# - Object ID: #SmarTeam.Links.OBJECT_ID#
  Endif
```

```

Set      %FOUND%, True
Next  Smarteam.Links
If  %FOUND% <> True
    Message  Error, No object of class "Item" linked to document %DOCNUM%.
Endif

```

Running a ENOVIA SmarTeam query by means of an SmQuery

```

Smarteam.GetList Document , CN_ITEM, where TDM_ID like *313*, and TDM_DESCRIPTION <> , _
                                order by REVISION
Loop  Smarteam.List
    Message I, #Smarteam.List.TDM_ID# --- #Smarteam.List.TDM_DESCRIPTION# --- #Smarteam.List.CN_ITEM#
Next  Smarteam.list

```

Running a ENOVIA SmarTeam query by means of an SmSimpleQuery

```

Smarteam.GetList2  select * from TN_DOCUMENTS where TDM_ID like "%313%" and TDM_DESCRIPTION <> "" _
                                order by REVISION
Loop Smarteam.List
    Message I, #Smarteam.List.TDM_ID# --- #Smarteam.List.TDM_DESCRIPTION# --- #Smarteam.List.CN_ITEM#
Next Smarteam.list

```

Miscellaneous

Call an external function which returns multiple values

```

CallExternalFunction  MyApplication.Cls, , GetResult, #Smarteam.TDM_ID#
#System.FunctionResult#

```

Explanation: Suppose that function GetResult returns a string like this: "Define %RET1%, Val1, %RET2%, Val2, %RET3%, Val3". In this case #System.FunctionResult# is automatically defined as "Define %RET1%, Val1, %RET2%, Val2, %RET3%, Val3" after the function call. Therefore statement #System.FunctionResult# evaluates to "Define %RET1%, Val1, %RET2%, Val2, %RET3%, Val3", so that - after this Define has been executed - %RET1%, %RET2%, and %RET3% have values Val1, Val2, and Val3 respectively.

Pass multiple arguments to the ENOVIA SmarTeam/SAP Adaptor through single argument ArgString

```
%ARGSTRING%
```

Explanation: Suppose that the value of ArgString - the argument passed to the ENOVIA SmarTeam/SAP Adaptor - is "Define %VAR1%, Val1, %VAR2%, Val2, %VAR3%, Val3". In this

case statement %ARGSTRING% evaluates to "Define %VAR1%, Val1, %VAR2%, Val2, %VAR3%, Val3", so that - after this Define has been executed - %VAR1%, %VAR2%, and %VAR3% have values Val1, Val2, and Val3 respectively.

Call SAP function module; read values directly from SAP table

```
FunctionModule.AddExport  QUERY_TABLE, Char, 30, MAKT
FunctionModule.AddExport  ROWCOUNT, Int, 4, 2
FunctionModule.AddTable   DATA, Char, 512
FunctionModule.AddTable   OPTIONS, Char, 72
FunctionModule.AddTableLine  OPTIONS, MATNR = '#SmarTeam.TDM_SAP_MAT_NUM#'
FunctionModule.Call        RFC_READ_TABLE
If  #FunctionModule.RfcRc# <> 0
    Message E, #FunctionModule.ErrorInfo#
Else
    Loop  FunctionModule.DATA
        Message  I, \Left40{\Mid23{#FunctionModule.DATA#}}
    Next  FunctionModule.DATA
Endif
```

Select plant and valuation area from SAP table

```
FunctionModule.AddExport  QUERY_TABLE, Char, 30, T001W
FunctionModule.AddExport  ROWCOUNT, Int, 4, 100
FunctionModule.AddExport  DELIMITER, CHAR, 1, .
FunctionModule.AddTable   OPTIONS, Char, 72
FunctionModule.AddTable   DATA, Char, 512
FunctionModule.AddTable   FIELDS, Char, 103
FunctionModule.AddTableLine  FIELDS, WERKS
FunctionModule.AddTableLine  FIELDS, NAME1
FunctionModule.AddTableLine  FIELDS, BWKEY
FunctionModule.Call        RFC_READ_TABLE
If  #FunctionModule.RfcRc# <> 0
    Message E, #FunctionModule.ErrorInfo#
Else
    Grid.Addline  Plant, Description, Valuation\Area
    Grid.SetColWidth  1200, 2400, 1200
    Grid.SetProperties  6000, 5500, 0, 1, X
    Grid.SetRowFontBold  1
    Loop  FunctionModule.DATA
        Grid.AddLine  \LeftDotLeft4{#FunctionModule.DATA#}, _
                      \LeftDotLeft30{\TrimRight35{#FunctionModule.DATA#}}, _
                      \Right4{\Trim{#FunctionModule.DATA#}}
    Next  FunctionModule.DATA
Endif
Grid.Show  Select Plant, Please choose a Plant by double clicking it., X
Check      #Grid.Result# = OK
Set        %R%, #Grid.Row#
Message    I, Plant=#Grid.Value(%R%,1)#\, Plant Name=#Grid.Value(%R%,2)#\, _
           Valuation Area=#Grid.Value(%R%,3)#
```

Process all lines of an input file

```
Define %FILENAME%, C:\\Temp\\test.txt
File.OpenForInput %FILENAME%
Label :Continue:
  Check #File.EOF# <> X
  File.GetLine
  SplitString #File.Line#, \\Hex{09}, %1%, %2%, %3%
  ExtractString #File.Line#, 7-, %Extract%
  Message Information, This is line number #File.Line.Number#: #File.Line# _
    \\Its first element is "%1%". _
    \\The extracted part is "%Extract%".
Goto :Continue:
```

Remark: \\Hex{09} gives the tab character which - in this example - separates the line elements.

Retrieve list of materials and write material information to file

```
Material.GetList from: 47* to: 60*,,,,,,10
File.OpenForOutput C:\\Temp\\MatList.txt
Loop Material.List
  Material.AddBasicData #Material.List.Material#
  Material.GetDetail
  File.PutLine #Material.List.Material#;#Material.List.Matldesc#;#Material.BasicData.Baseuom#
Next Material.List
```

Write some information to a temporary file, execute a program on the file, and delete the file when the program has terminated

```
Define %TEMPFILE%, C:\\Temp\\#System.TemporaryFilename#.txt
File.OpenForOutput %TEMPFILE%
File.PutLine TDM_SAP_MAT_NUM = #SmarTeam.TDM_SAP_MAT_NUM# \\, TDM_ID = #SmarTeam.TDM_ID#
File.Close
File.Execute notepad.exe %TEMPFILE%, 1
File.WaitForTask
File.Delete %TEMPFILE%
```

Remark: The comma in the third line has to be masked by a backslash character. Otherwise "TDM_ID = #SmarTeam.TDM_ID#" would be interpreted as the second argument of File.PutLine.

Display values in a grid

```

Grid.AddLine   Field Name, Field Value
Grid.AddLine   Material Number, #MATERIAL.MATERIAL#
Grid.AddLine   Description, #MATERIAL.BASICDATA.MATLDESC#
Grid.AddLine   Unit of Measurement, #MATERIAL.BASICDATA.BASEUOM#
Grid.AddLine   Net Weight, #MATERIAL.BASICDATA.NETWEIGHT#
Grid.AddLine   Unit of Weight, #MATERIAL.BASICDATA.UNITOFWT#
Grid.SetColFontBold 1
Grid.SetRowFontBold 1
Grid.SetColWidth 2000, 3000
Grid.SetProperties 6000, 5500, 1, 1, X
Grid.Show      Material Values, SAP System: #SAP.Destination# _
               \nClient: #SAP.Client# _
               \nDetails of Material #MATERIAL.MATERIAL#, X, , , X, X

```

Read a "BOM" from an Excel sheet

```

Excel.Launch
Excel.OpenWorkBook C:\\Temp\\BOM.xls
Excel.SelectSheet 1
Excel.FindPosition 1, 30, 1, 1, Item No.
Excel.SetPosition \Eval{#Excel.Row# + 1}
DoLoop %I%, 1, 1000, 1
  Excel.GetData %ItemNo%, %Item%, %Quantity%
  If %ItemNo%! = !
    Excel.Terminate
    ExitDoLoop
  Endif
  Message I, Item number: %ItemNo%\nItem:\t %Item%\nQuantity: %Quantity%
DoNext

```