

# 3DSmartDocCreator Server– TS9

## Server Administration Guide

---

**BPA Delivery 8 for V5R20 (V 5.8)**  
**Version 1.8**

# Table of Contents





---

<b>3DSMARTDOCCREATOR .....</b>	<b>1</b>
<b>SERVER– TS9.....</b>	<b>1</b>
BPA Delivery 8 for V5R20 (V 5.8) .....	1
<b>Table of Contents .....</b>	<b>2</b>
<b>3DSmartDocCreator Server Administration - Introduction.....</b>	<b>3</b>
A. Related Documentation .....	3
<b>CreateJobFile.....</b>	<b>4</b>
<b>The operations file (JOD).....</b>	<b>6</b>
<b>Appendix 1 .....</b>	<b>15</b>
3DSmartDocCreator Server - Available Activity Functions.....	15
<b>Appendix 2 .....</b>	<b>35</b>
3DSmartDocCreator Server - Available Composer related Functions.....	35
<b>Appendix 3 .....</b>	<b>37</b>
3DSmartDocCreator Server - Available Inline Functions.....	37
<b>Appendix 4 .....</b>	<b>42</b>
3DSmartDocCreator Server - Available Variables and Definitions .....	42

# 3DSmartDocCreator Server Administration - Introduction

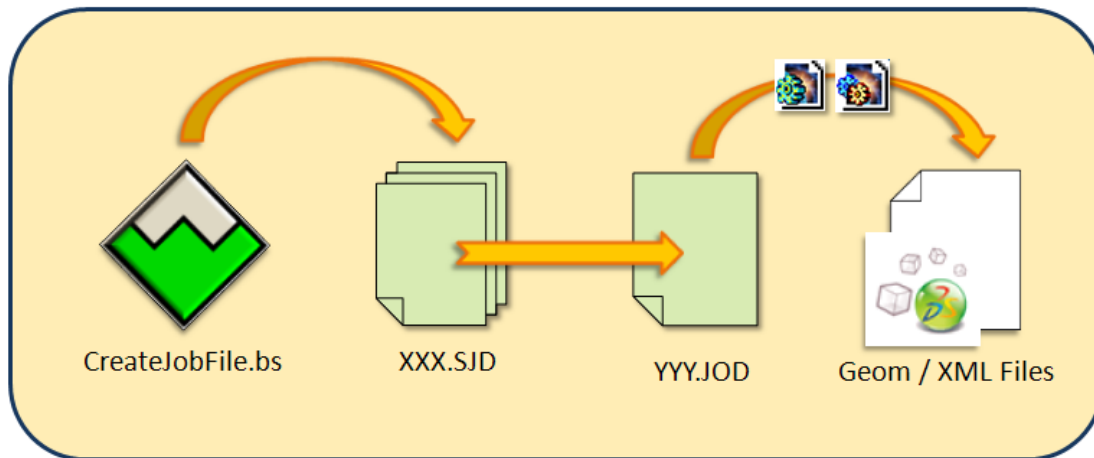
This documentation describes the the sequence of the automation process. It also contains the details of the structure of the operation file, which eventually can be customized to obtain the desired result.

## A. Related Documentation

-  TS9 Activity Functions – Appendix 1
-  TS9 Composer Functions – Appendix 2
-  TS9 Inline Functions.htm – Appendix 3
-  TS9 Variable Functions.htm – Appendix 4

The server side process consists of creating a job file (SJD). This job file is then processed as per the operation laid down in the JOD (Job operation definition) file. On successful completion the desired output is created.

A typical server side process can be depicted from the following figure.



# CreateJobFile

This script file contains all the functions needed to create the automated jobs for the 3DSmartDocCreator Server. The associated functions listed carry the operation of creating the job files need to be fed to the operations file which then create the output required.

The content of the script file are as follows.

```
' CONST Definitions has to be set according to used Database and infrastructure |****
Const ID_FIELD = "TDMX_ID" ' Document Superclass sequenced identifier
Const PENDING = "\\BPASERVER\PENDING\"
Const DESCRIPTION = "TDM_DESCRIPTION" ' Document Superclass Description Attribute
Const PENDING = "C:\JS\PENDING\" ' pending Directory of 3DSmartDocCreator Server
```

This section deals with the constat definitions.

Const\_ID\_FILED is the document ID

Const\_PENDING is the location on the server where the job files are located before being processed.

Const\_DESCRIPTION is the document description attribute

```
*****
' Function : CreateGeomFile
' Description : Creates a job to convert selected Assembly/Part and store the XML and Geom in the Server vault
' Class : Design
' Hook : After Checkin / used defined
' Author : DASSAULT SYSTEMES DEUTSCHLAND AG
' last change : 19.2.2009
*****
Function CreateGeomFile(ApplHnd1 As Long, Sstr As String, FirstPar As Long, SecondPar As Long, ThirdPar As Long ) As Integer
    Dim Session As SmApplic.SmSession
    Dim FirstRec As Object
    Dim classid As Integer
    Dim objectid As Long
    Dim path As String ' JobServerPath für Job-Datei
    Dim filename As String
    Dim desc As String

    'Converting ApplHnd1 to SmSession
    Set Session = SCREXT_ObjectForInterface(ApplHnd1)

    'converting record list into COM SmRecordList object
    CONV_RecordListToComRecordList FirstPar, FirstRec
    path = PENDING

    'Check for same Class in the recordlist
    For i = 0 To FirstRec.RecordCount - 1
        classid = FirstRec.ValueAsInteger("CLASS_ID", i)
        objectid = FirstRec.ValueAsInteger("OBJECT_ID", i)
        filename = path & ltrim(str(objectid)) & ".sjd"
        desc = firstrec.valueasstring(DESCRIPTION, i)

        open filename For Output As #1
        Print #1, "[JOB]" & Chr(13)
        Print #1, "OPERATION=CREATECOMPOSERFILES" & Chr(13)
        Print #1, "ABC=TEST" & Chr(13)
        Print #1, "OBJECT_ID=" & ltrim(str(objectid)) & Chr(13)
        Print #1, "CLASS_ID=" & ltrim(str(classid)) & Chr(13)
        Print #1, "DESCRIPTION=" & desc & Chr(13)

        Close #1
    Next i
End Function
```

This section deals with the function CreateGeomfile. The purpose of this function is to create the job file which interacts with the operation file and converts the selected CAD files (Assembly and Part) into the desired XML and Geom format and stores them in the server location defined.

The output is an ASCII file with the following content as per the print instruction in the script. This file is located in the Pending folder of the server location.

```
[JOB]
OPERATION=CREATECOMPOSERFILES
ABC=TEST
OBJECT_ID=19643
CLASS_ID=100
DESCRIPTION=Sub_assembly
```

The Other functions in the script file are to create the view objects (using publishing features of the 3DVia Sync.)

```
*****
Function      : CreateViewObjects
Description   : Creates a Job to generate Jpeg's for each view from the Composer Project
Class        : Composer Project
Hook         : After CheckIn
Author       : DASSAULT SYSTEMES DEUTSCHLAND AG
Last Change  : 19.2.2009
*****
Function CreateViewObjects(ApplHndl As Long,Sstr As String,FirstPar As Long,SecondPar As Long,ThirdPar As Long ) As Integer
    Dim Session As SmApplic.SmSession
    Dim FirstRec As Object
    Dim classid As Integer
    Dim objectid As Long
    Dim path As String
    Dim filename As String
    Dim desc As String
    Dim viewFile As String
    Dim Id As String

    'Converting ApplHndl to SmSession
    Set Session = SCREXT_ObjectForInterface(ApplHndl)

    'Converting record list into COM SmRecordList object
    CONV_ReclstToComRecordList FirstPar,FirstRec
    path = PENDING

    'check for same Class in the recordlist
    For i = 0 To FirstRec.RecordCount -1
        Classid = FirstRec.ValueAsInteger("CLASS_ID",i)
        objectid = FirstRec.ValueAsInteger("OBJECT_ID",i)
        filename = path & ltrim(str(objectid)) & ".sjd"
        desc = firstrec.valueasstring(DESCRIPTION,i)
        id= firstrec.valueasstring(id_field,i)
        viewFile = id & "\" & id & ".smgview"

        Open filename For Output As #1
        Print #1, "[JOB]" & Chr(13)
        Print #1, "OPERATION=CREATECOMPOSERDOCUMENTS" & Chr(13)
        Print #1, "ABC=TEST" & Chr(13)
        Print #1, "OBJECT_ID=" & ltrim(str(objectid)) & Chr(13)
        Print #1, "CLASS_ID=" & ltrim(str(classid)) & Chr(13)
        Print #1, "DESCRIPTION=" & desc & Chr(13)
        Print #1, "viewFile=" & viewFile & Chr(13)
        Close #1
    Next i
End Function
```

# The operations file (JOD)

The job file created by the script is stored in the pending folder which is then processed by the operations files with an extension JOD. The jod files are located in the folder Operations in the server.

A JOD file mainly consists for 4 sections as shown below.

## Section 1: Messages

```
[Messages]

1.I=Starting SMARTEAM
1.S=Success
1.E=Error starting SMARTEAM

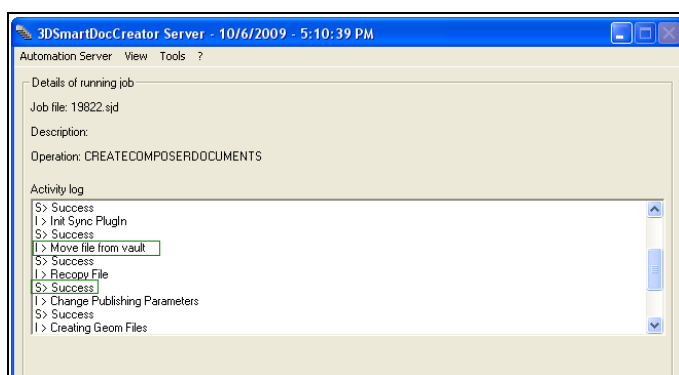
2.I=Connecting User
2.S=Success
2.E=Error connecting User

3.I=Creating SMARTEAM object
3.S=Success
3.E=Error

4.I=Move file from vault
4.S=Success
4.E=Error

5.I=Recopy File
5.S=Success
5.E=Error
```

This section is responsible for the messages those need to be flashed in the Automation Server UI. These messages can be customized to suit the user's understanding and language.



## Section 2: Operations

```

[CREATECOMPOSERFILES]

SetExitHandler      OnSuccess, OnError
New                 Smarteam, STJOB
New                 File, DoCopy
Set                 %Home%, C:\JS\

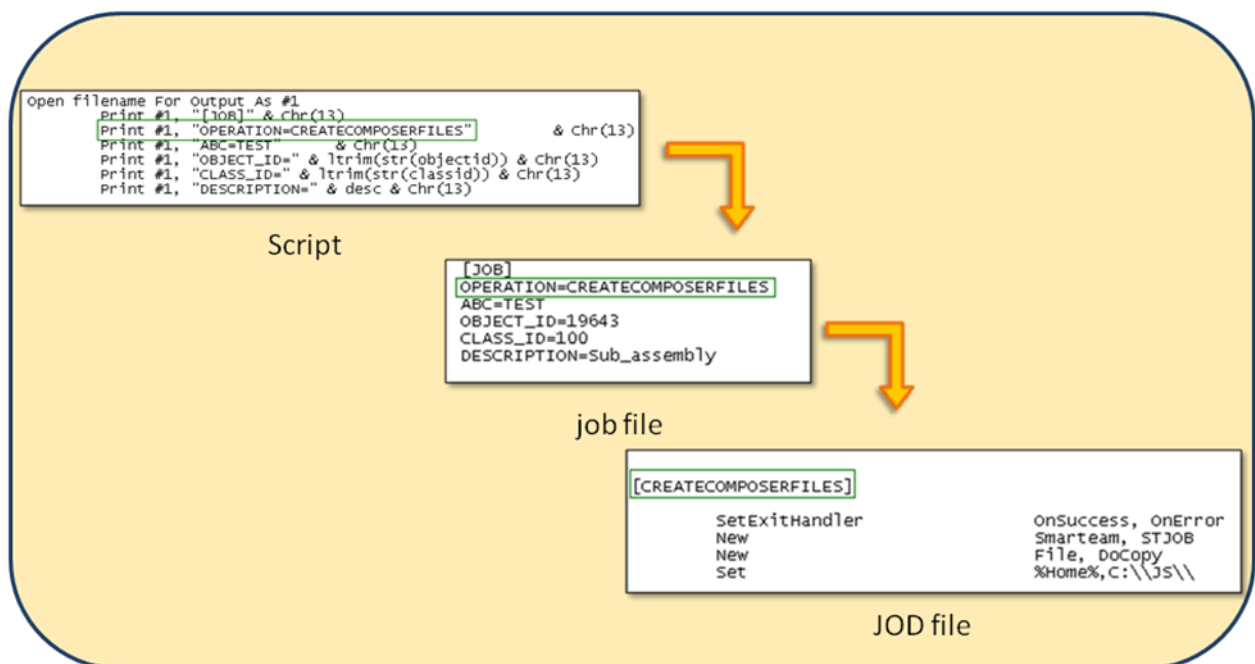
set                 %SyncExe%, "G:\Program Files\Dassault Systemes\3DVIAComposer\6.4\Bin\3dviaconverter.exe"
set                 %syncXML%, "c:\js\operations\convertshattered.xml"
  
```

This section deals with the operation those can be carried sequentially to obtain the desired output.

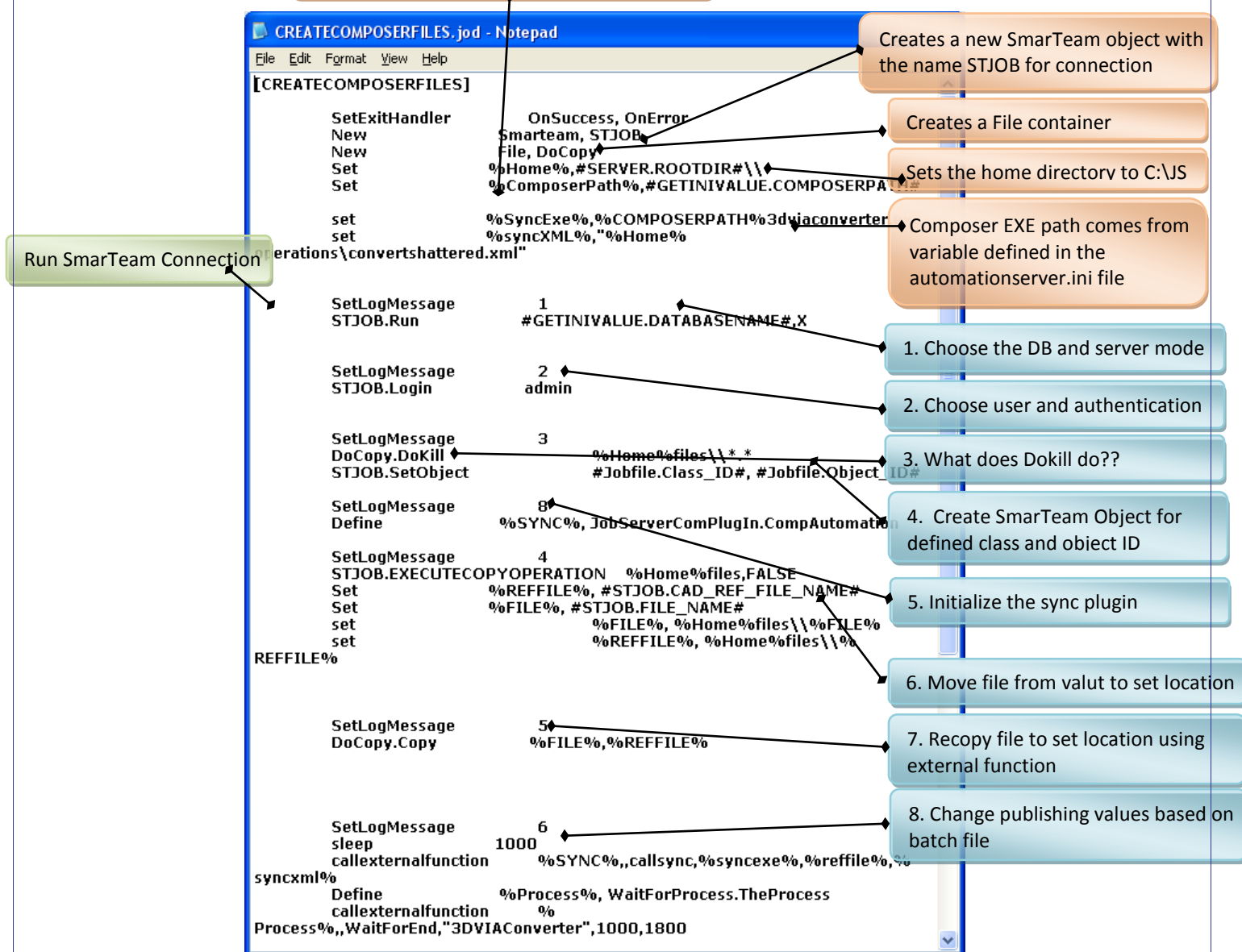
For instance the CREATECOMPOSERFILES operation is responsible to create a job to convert the Assembly / Part and store the XML and Geom in the server Vault. The name of the operation file has to be the value of the parameter operation in the script file for the particular function.

Here the value of OPERATION=CREATECOMPERFILES, hence the name of the JOD file is CREATECOMPOSERFILES.jod.

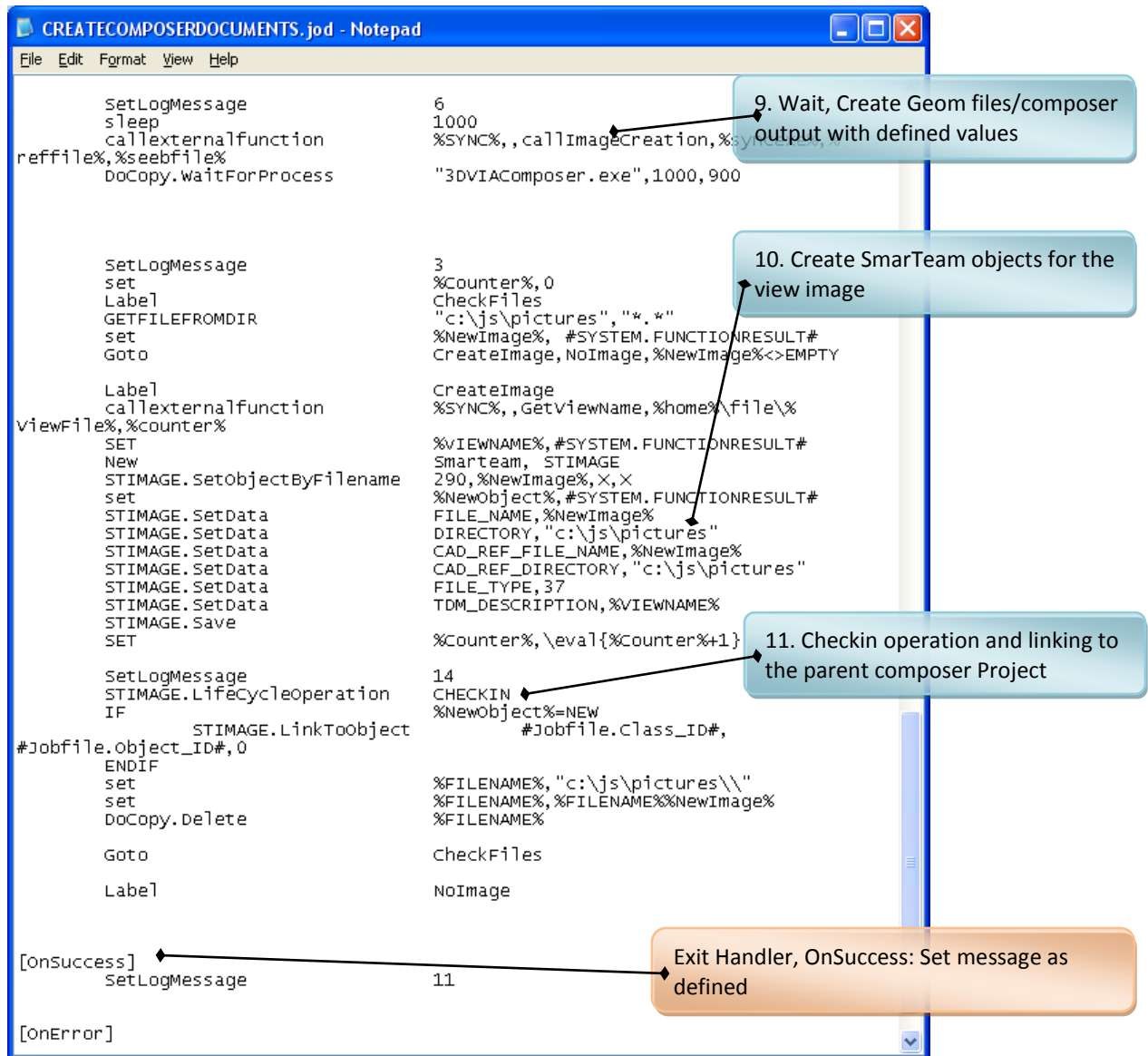
Attention: The name of the main section has to fit to the JOD filename



Sets the path for the batch file responsible for creating and storing view images







### 1. SmarTeam Function : Choose the data base and the server

Name	Smarteam.Run
Arguments	<Databasename>, <Mode>
Description	Creates a SMARTEAM engine. If <Mode> is set to "X" then the engine FreeThreaded (Servermode)
Remarks	
Example	Smarteam.Run SmDeno , X

### 2. SmarTeam Function : Choose user and authentication

Name	Smarteam.Login
Arguments	<Login>, <Password>,<Not use encryption as : X>
Description	Performs a login to SMARTEAM and creates a valid User-Session
Remarks	The password has to be encrypted with the encrypt software delivered with the JobServer. If the Password is clear Test you have to set the third parameter to : X
Example	<p>Smarteam.Login joe (Login for joe if joe has no password)</p> <p>Smarteam.Login joe, 56WERT45, X (login for joe and the not encrypted password)</p> <p>Smarteam Login joe, -.q&lt;j@&lt;f (login for joe and the encrypted password)</p>

### 3. File Container

### 4. SmarTeam Function: Create SmarTeam Object for defined class and object ID

Name	Smarteam.SetObject
Arguments	<ClassIdOrName>, <ObjectId>, <SetDefaultValues>, <IncrementSequences>
Description	If <ClassIdOrName> and <ObjectId> are both given, the parent object is set to the SMARTEAM object thoroughly identified by the two parameters. If <ObjectId> is missing, the parent object is set to a new 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	<p>Smarteam.SetObject %Class_ID%, %Object_ID%</p> <p>Smarteam.SetObject %Class_ID%, , X, X</p>

### 5. Composer Function: Initialize the sync Plugin

<b>Name</b>	<b>CallSync</b>
<b>Arguments</b>	<SyncExe> -> Full Path of Converter <Design File> -> Design File to be converted <seeb XML> -> Convertrules defined in xml file
<b>Description</b>	Exexutes the Sync Integration with the defined values
<b>Remarks</b>	
<b>Example</b>	set %SyncExe%, "C:\Programme\Dassault Systemes\3DVIAComposer\6.3\Bin\3dviaconverter.exe" set %syncXML%, "c:\js\operations\convertshattered63.xml" Set %REFFILE%, #STJOB.CAD_REF_FILE_NAME# Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,callsync,%syncexe%,%reffile%,%syncxml% DoCopy.WaitForProcess "3DVIAConverter.exe",1000,900

#### 6. SmarTeam function: Move file from valut to set location

<b>Name</b>	Smarteam.MoveFileFromVault
<b>Arguments</b>	-
<b>Description</b>	Copies the Object-File to the Jobserver-Files directory in read/wrire mode
<b>Remarks</b>	
<b>Example</b>	Smarteam.MoveFileFromVault

#### 7. Miscellaneous Function: Recopy file to set location, composer function used:Unzip

<b>Name</b>	CallExternalFunction
<b>Arguments</b>	<Application.Class>, <Server>, <Function>, <Argument>, <ResultVariable>
<b>Description</b>	<p>Calls function &lt;Function&gt; of object &lt;Application.Class&gt; on server &lt;Server&gt;. &lt;Application.Class&gt; must have been compiled to an ActiveX dll with &lt;Function&gt; being a public function in a multi-use &lt;Class&gt;. &lt;Argument&gt; is passed as string to the function. &lt;ResultVariable&gt; is the name of a definition which is defined as the function result in the same way <b>Define</b> does. The interface definition of &lt;Function&gt; must be equivalent to</p> <p>Public Function &lt;Function&gt;(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 SMARTEAM SAP Integration Toolkit. The external function can write its own trace to that file. TraceFile is set to "", if no trace shall be written.</p>
<b>Remarks</b>	If the function shall return several values proceed as described in the <b>sample configurations</b> .
<b>Example</b>	CallExternalFunction MyApp.Cls, , GetResult, ClassID=#Smarteam.CLASS_ID# ObjectID=#Smarteam.OBJECT_ID#, %MYRESULT%.

<b>Name</b>	<b>UnZip</b>
Arguments	<Zipname> -> Project that has to be unzipped <Destination> -> Destination Path for unzipping
Description	Unzips a Composer Project to a named Folder
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,Unzip,%file%,%Home%files

8. Miscellaneous Function: Change publishing values based on batch file, composer function used:ChangeXMLValue

<b>Name</b>	<b>ChangeXMLValue</b>
Arguments	<Filename> -> XML Filename <Identifier> -> Node descriptor <Value> the new value that will be written to that node
Description	Changes a value in a "Seeb" file
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,ChangeXMLValue,%seebfile%,Batch.IOSmgViewFile,%home%\file%\ViewFile%

9. Miscellaneous Function : Wait, Create Geom files/composer output with defined values, composer function used: CallImageCreation

<b>Name</b>	<b>CallImageCreation</b>
Arguments	<SyncExe> -> Full Path of Composer <Reffile> -> Composer Project File that has to be worked on <seeb XML> -> Converter rules defined in xml file
Description	Executes the Composer Output creation with the defined values
Remarks	
Example	set %SeebFile%,%Home%\Operations\createviewpictures.xml set %SyncExe%,"C:\Programme\Dassault Systemes\3DVIAComposer\6.2\Bin\3dviacomposer.exe" Set %REFFILE%, #STJOB.CAD_REF_FILE_NAME# Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,callImageCreation,%syncexe%,%reffile%,%seebfile% DoCopy.WaitForProcess "3DVIAComposer.exe",1000,900

10. Miscellaneous Function : Create SmartTeam objects for the view image, composer function used: GetViewname

<b>Name</b>	<b>GetViewName</b>
<b>Arguments</b>	<Viewfile> -> XML Composer View file <Number> -> Number of selected View inside View File
<b>Description</b>	Sets the "System Function Result Value" to the View name of the View File
<b>Remarks</b>	
<b>Example</b>	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,GetViewName,%home%\file\%ViewFile%,%counter% SET %VIEWNAME%,#SYSTEM.FUNCTIONRESULT#

#### 11. Checkin operation and linking to the parent composer Project

<b>Name</b>	<b>Smarteam.LifeCycleOperation</b>
<b>Arguments</b>	<Operation>, <InvokeScripts>
<b>Description</b>	Performs a life-cycle operation on the parent object. <Operation> can be "CHECKIN" or "RELEASE". If <InvokeScripts> is set to "X", scripts are invoked. Dependencies will not be lifecycled.
<b>Remarks</b>	
<b>Example</b>	Smarteam.LifeCycleOperation RELEASE, X

<b>Name</b>	<b>Smarteam.LinkToObject</b>
<b>Arguments</b>	<ClassId>, <ObjectId>,<LinkClassId oder LinkClassName>
<b>Description</b>	Performs a general link to an Object. if <LinkClassId> = "" the default linkid for this will be taken.
<b>Remarks</b>	Scripts are invoked
<b>Example</b>	Smarteam.LinkToObject 20,234,0

## Section 3 and Section 4: Exit handler

The Exit handles also forms the next two section of the operations file.

There are two arguments associated with the Exit handlers are OnSuccess and OnError.

They are represented as follows.

[OnSuccess] SetLogMessage	11
[OnError]	

The control is shifted to these exit handlers depending on the success or failure in the operation section and then a set of output is obtained based on the instructions in the exit handler section. Here in the above example, the OnSuccess exit handler will give a message 11 set in the message section at the start of the document.

# Appendix 1

---

## **3DSmartDocCreator Server - Available Activity Functions**

### **File Functions**

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

### **Smarteam Functions**

[CopyFile](#) [CreateReportObject](#) [Delete](#) [GetData](#) [GetLinks](#) [GetLinksPlus](#)  
[GetList](#) [GetList2](#) [GetStructure](#) [GetStructurePlus](#) [InitiateProcess](#)  
[LifeCycleOperation](#) [LinkToObject](#) [LinkToParent](#) [Login](#) [Logout](#)  
[RefreshWindow](#) [RetrieveMapping](#) [Run](#) [Save](#) [SendMessage](#) [SetData](#)  
[SetObject](#) [Terminate](#)  
[ExecuteCopyOperation](#) [ExecuteLFOperation](#) [CopyViewFile](#)  
[MoveFileFromVault](#) [MoveBack](#) [MOVEADDFILESTOVAULT](#)  
[MoveBackFileFromDir](#)

### **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](#) [Sleep](#) [StopRedo](#) [StartRedo](#)

### **Miscellaneous Functions**

[# \(Comment\)](#) [CallExternalFunction](#) [Define](#) [ExtractString](#) [Message](#)  
[Progress](#) [PutLog](#) [ResetPrinter](#) [SendMail](#) [Set](#) [SetExitHandler](#) [SetJobDescription](#)  
[SetLogMessage](#) [SetPrinter](#) [SplitString](#) [WaitOnCommit](#) [DoPrintCommand](#)  
[ResetJobTimer](#) [Replace](#)

### **Workflow Functions**

[InitByObjectId](#) [AcceptResponse](#) [RejectResponse](#) [OBJECT](#)  
[GetAttachedDocuments](#)

## 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>, <Target>
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>, <WindowStyle>
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.Move
Arguments	<Source>, <Target>
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 <b>File.PutLine</b> and <b>File.Close</b> 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>
Description	The file given as <Name> is opened for input. Functions <b>File.GetLine</b> and <b>File.Close</b> refer to this file without mentioning the filename again.
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 <b>File.PutLine</b> and <b>File.Close</b> 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 <b>File.OpenForAppend</b> or <b>File.OpenForOutput</b> .

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

Name	File.WaitForProcess
Arguments	Name of Process without Path, Sleep intervall for next check in milliseconds , Repeat value
Description	Jobserver is waiting until a process in windows is terminated.
Remarks	The example looks for the end of the process named PNJob5.exe. The jobserver is looking every second and for 20 seconds long.
Example	new FILE, PrintProcess PrintProcess.WaitForProcess PNJob5.exe, 1000 , 20

## Smarteam Functions

Name	Smarteam.CopyFile
Arguments	<Directory>, <Filename>, <Permission>
Description	Copies the SMARTTEAM 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".
Remarks	
Example	Smarteam.CopyFile C:\\Temp, tmp_#Smarteam.FILE_NAME#, ReadOnly

Name	Smarteam.CreateReportObject
Arguments	<ClassNameOrID>, <LinkClassNameOrID>, <ID attribute name>, <ID value>, <Description attribute name>, <Description value>, <Message attribute name>, <Message text>
Description	Creates a SMARTTEAM 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 SMARTTEAM SAP Integration Toolkit 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, #Logoninfo.BapiMessages# or #Logoninfo.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, #Logoninfo.BapiMessages#
--	--

Name	Smarteam.Delete
Arguments	<X=Invokescripts>,<X=Checkauthorisation>
Description	Deletes a SMARTEAM-Object from database
Remarks	
Example	Smarteam.GetData

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

Name	Smarteam.GetLinks
Arguments	<ClassNameOrID>, <LinkClassNameOrID>
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>.
Remarks	The "Loop Smarteam.Links"/"Next Smarteam.Links" statements can be used to loop on the linked objects retrieved.
Example	Smarteam.GetLinks Items, Documents Items Relation Loop Smarteam.Links ... Next Smarteam.Links

Name	Smarteam.GetLinksPlus
Arguments	<ClassNameOrID>, <LinkClassNameOrID>
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.
Remarks	The "Loop Smarteam.Links"/"Next Smarteam.Links" statements can be used to loop on the linked objects retrieved.
Example	Smarteam.GetLinksPlus Items, Documents Items Relation Loop Smarteam.Links ... Next Smarteam.Links

Name	Smarteam.GetList
Arguments	<ClassIdOrName>, <Expression1>, <Expression2>, ...
Description	<p>Performs a query in SMARTEAM class given by &lt;ClassIdOrName&gt;. &lt;Expressions&gt; 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> <p>&lt;Attribute&gt;  Where &lt;Attribute&gt; &lt;Operator&gt; &lt;Value&gt;  And &lt;Attribute&gt; &lt;Operator&gt; &lt;Value&gt;  Or &lt;Attribute&gt; &lt;Operator&gt; &lt;Value&gt;  Order by &lt;Attribute&gt;</p> <p>In these expressions, &lt;Attribute&gt; is the SMARTEAM attribute name like CN_ID, &lt;Operator&gt; is a comparison operator (=, &lt;, &lt;=, &gt;=, like), and &lt;Value&gt; is the value searched for. If &lt;Operator&gt; equals "like", &lt;Value&gt; can contain wildcard characters "*" and ".".</p>
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	<pre>Smarteam.GetList Documents, cn_item_no, where cn_id like *313*, and cn_description &lt;&gt; , order by revision Loop Smarteam.List ... Next Smarteam.List</pre>

Name	Smarteam.GetList2
Arguments	<SelectStatement>
Description	Performs a simple query in 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	<pre>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</pre>

Name	Smarteam.GetStructure
Arguments	None
Description	Retrieves all attribute values of all first level children of the parent object but no hierarchical link attributes.
Remarks	The "Loop Smarteam.Structure"/"Next Smarteam.Structure" statements can be used to loop on the structure retrieved.
Example	<pre>Smarteam.GetStructure Loop Smarteam.Structure ... Next Smarteam.Structure</pre>

Name	Smarteam.GetStructurePlus
Arguments	None
Description	Retrieves all attribute values of all first level children of the parent object plus all attribute values of all hierarchical link objects.
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

Name	Smarteam.InitiateProcess
Arguments	<FlowClassIdOrName>, <FlowChartName>, <ResponseName>, <NextNodeName>, <Description>, <Comment>, <ExecutersNames>
Description	Initiates a SMARTTEAM 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>, <InvokeScripts>
Description	Performs a life-cycle operation on the parent object. <Operation> can be "CHECKIN" or "RELEASE". If <InvokeScripts> is set to "X", scripts are invoked. Dependencies will not be lifecycled.
Remarks	
Example	Smarteam.LifeCycleOperation RELEASE, X

Name	Smarteam.LinkToObject
Arguments	<ClassId>, <ObjectId>,<LinkClassId oder LinkClassName>
Description	Performs a general link to an Object. if <LinkClassId> = "" the default linkid for this will be taken.
Remarks	Scripts are invoked
Example	Smarteam.LinkToObject 20,234,0

Name	Smarteam.LinkToParent
Arguments	<ClassId>, <ObjectId>,<LinkClassId oder LinkClassName>
Description	Performs a link to a parent Object. if <LinkClassId> = "" the default linkid for this superclass will be taken.

Remarks	Scripts are invoked
Example	Smarteam.LinkToParent 20,234,0

Name	Smarteam.Login
Arguments	<Login>, <Password>,<Not use encryption as : X>
Description	Performs a login to SMARTEAM and creates a valid User-Session
Remarks	The password has to be encrypted with the encrypt software delivered with the JobServer. If the Password is clear Test you have to set the third parameter to : X
Example	Smarteam.Login joe (Login for joe if joe has no password)  Smarteam.Login joe, 56WERT45, X (login for joe and the not encrypted password)  Smarteam Login joe, -.q<j@<f (login for joe and the encrypted password)

Name	Smarteam.RefreshWindow
Arguments	<RefreshAllWindows>
Description	Refreshes 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	<Databasename>, <Mode>
Description	Creates a SMARTEAM engine. If <Mode> is set to "X" then the engine FreeThreaded (Servermode)
Remarks	
Example	Smarteam.Run SmDeno , X

Name	Smarteam.Save
Arguments	<InvokeScripts>, <CheckAuthorization>
Description	Updates the parent object in 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>, <Body>, <SMARTEAM user name1>, <SMARTEAM user name2>,<SMARTEAM user name3>, ...

Description	Sends a SmartMessage without further user interaction. The parent object passed to the integration by SMARTEAM is attached to the message.
Remarks	If BAPI messages or BAPI error messages are to be sent, #Logoninfo.BapiMessages# or #Logoninfo.BapiErrorMessages# should be passed as <Body>.
Example	Smarteam.SendMessage Error while processing object #Smarteam.CN_ID#, #Logoninfo.BapiMessages#, joe, jim, sue

Name	Smarteam.SetData
Arguments	<SMARTEAM attribute1>, <Value1>, <SMARTEAM attribute2>, <Value2>, ...
Description	Fills the specified SMARTEAM variable of the parent object with the given values.
Remarks	The SMARTEAM attributes must be specified without prefix or postfix, i. e. without #Smarteam.__#.
Example	Smarteam.SetData CN_DESCRIPTION, #Document.Basicdata.Description#, TDM_SAP_MAT_NUM, 4711

Name	Smarteam.SetObject
Arguments	<ClassIdOrName>, <ObjectId>, <SetDefaultValues>, <IncrementSequences>
Description	If <ClassIdOrName> and <ObjectId> are both given, the parent object is set to the SMARTEAM object thoroughly identified by the two parameters. If <ObjectId> is missing, the parent object is set to a new 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	Smarteam.SetObject %Class_ID%, %Object_ID% Smarteam.SetObject %Class_ID%, , X, X

Name	Smarteam.ExecuteCopyOperation
Arguments	optional <Copy Folder> (without ending "\"), optional TRUE for propagate -> default = false
Description	Copies the Object-File and all dependencies to the SMARTEAM-Workdirectory. Reacts to the current Lifecyclerules for the COPY Operation. If the Copy Folder is set the main document is switched to read/write mode. Default propagation = false
Remarks	Be sure that the Work-Directory for the SMARTEAM-JobServer User is set to the JobServers File Directory
Example	Smarteam.ExecuteCopyOperation C:\js\files, TRUE

Name	Smarteam.ExecuteLFOperation
Arguments	<Operation String>
Description	Does the same LF-Operation on a selected Object like the command on a Tree-Object in SMARTEAM. Operations are : APPROVE, CHECKIN, OBSOLETE, CHECKOUT. Dependencies will be lifecycled as defined in lifecycle-rule-setup

Remarks	Be sure that the Work-Directory for the SMARTTEAM-JobServer User is set to the JobServers File Directory
Example	Smarteam.ExecuteLFOperation APPROVE

Name	Smarteam.CopyViewFile
Arguments	<Copy Folder> (without ending "\"), optional <WriteMode>
Description	Copies the Object-File and all ViewFiles defined in the Application-Setup to the <Copy Folder>. If the <WriteMode> is set to "TRUE" the main document is switched to read/write mode
Remarks	
Example	Smarteam.CopyViewFile C:\js\files , TRUE

Name	Smarteam.MoveFileFromVault
Arguments	-
Description	Copies the Object-File to the Jobserver-Files directory in read/wrire mode
Remarks	
Example	Smarteam.MoveFileFromVault

Name	Smarteam.MoveBack
Arguments	-
Description	Copies the Object-File back to his Vault
Remarks	
Example	Smarteam.MoveBack

Name	Smarteam.MOVEADDFILESTOVAULT
Arguments	<Filename> (without directory)
Description	Copies the File to the vault of the SMARTTEAM-Object
Remarks	The File must exist in the JobServer-Files directory
Example	Smarteam.MoveBackFile example.tif

Name	Smarteam.MoveBackFileFromDir
Arguments	<Filename> (without directory), <SourceDir> (without ending "\" )
Description	Copies the File to the vault of the SMARTTEAM-Object from the SourceDir
Remarks	The File must exist in the SourceDir
Example	Smarteam.MoveBackFileFromDir example.tif , c:\printdir



## 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>, <FromValue>, <ToValue>, <StepValue>
Description	A Do loop is performed varying <LoopVariable> from <FromValue> to <ToValue> in steps of <StepValue>.
Remarks	<p>The loop has to be closed by the DoNext statement.</p> <p>&lt;LoopVariable&gt; is stored in memory as a <b>definition</b> and should therefore follow appropriate naming conventions for definitions.</p>
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>, <ResultString>
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 <b>SetExitCode</b> .
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 <b>DoLoop</b> , 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 (Classification.Allocations, Classification.Validations, Document.Descriptions, Documents.Links, Document.List, Document.StatusLog, Document.Structure, Material.BOM, Smarteam.Links, 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 <b>Loop</b> and <b>Next</b> statements.
Examples	Loop Smarteam.Structure, %I% Document.AddStructure DRW, #Smarteam.CN_ID#, 000, #Smarteam.REVISION#, #Smarteam.CN_QUANTITY# ExitLoop Smarteam.Structure, %I% > 10 Next Smarteam.Structure

Name	Goto
Arguments	<Label to jump if true>, <Label to jump if false>, <Condition1>, <Or Condition2>, <Or Condition3>, ...
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:, , #LogonInfo.BapiStatusSummary# = E, #LogonInfo.BapiStatusSummary# = A, #LogonInfo.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. 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.AddBIDData SAPLMGMM, 3006, MARC-BESKZ, E, BDC_OKCODE, /00 Else Material.AddBIDData 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 #Logondata.BapiStatus# = E, #Logondata.BapiStatus# = A Message Information, Document "#Document.Documentnumber#" saved to SAP. Document.SetStatusOtherVersions , , , NV, , NotEqual Endif</pre>

Name	Include
Arguments	<SectionName>
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	Include MoreLines ... [MoreLines] # Statements to be included above ...

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 (Classification.Allocations, Classification.Validations, Document.Descriptions, Documents.Links, Document.List, Document.StatusLog, Document.Structure, Material.BOM, Smarteam.Links, Smarteam.List, Smarteam.Structure)>, <LoopVariable>
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, ECMASTER, FILE, FUNCTIONMODULE, MATERIAL, SMARTEAM. The new objects can then be used according to documented functions and variables of that type.
Remarks	
Example	New Document, D, Material, M, Smarteam, S D.AddBasicData DRW, #S.CN_ID#, 000, #S.REVISION# M.AddBasicData #D.Documentnumber# New BatchInput, BI BI.CallTransaction MM01, X

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

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

Name	StopRedo
Arguments	None
Description	In case of failure JobServer will not try to redo the Job. This is the default behavior
Remarks	
Example	StopRedo

Name	StartRedo
Arguments	None
Description	In case of failure JobServer will try to redo the Job.
Remarks	
Example	Startredo

## 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 <b>variable</b> .
Example	# Any text as comment

Name	CallExternalFunction
Arguments	<Application.Class>, <Server>, <Function>, <Argument>, <ResultVariable>
Description	<p>Calls function &lt;Function&gt; of object &lt;Application.Class&gt; on server &lt;Server&gt;. &lt;Application.Class&gt; must have been compiled to an ActiveX dll with &lt;Function&gt; being a public function in a multi-use &lt;Class&gt;. &lt;Argument&gt; is passed as string to the function. &lt;ResultVariable&gt; is the name of a definition which is defined as the function result in the same way <b>Define</b> does. The interface definition of &lt;Function&gt; must be equivalent to</p> <p>Public Function &lt;Function&gt;(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 SMARTTEAM SAP Integration Toolkit. The external function can write its own trace to that file. TraceFile is set to "", if no trace shall be written.</p>
Remarks	If the function shall return several values proceed as described in the <b>sample configurations</b> .
Example	CallExternalFunction MyApp.Cls, , GetResult, ClassID=#Smarteam.CLASS_ID# ObjectID=#Smarteam.OBJECT_ID#, %MYRESULT%.

Name	Define
Arguments	<Key1>, <Value1>, <Key2>, <Value2>, <Key3>, <Value3>, ...
Description	Adds definitions to the internal memory.
Remarks	<p>The definitions are available only during the action sequence in which they are issued.</p> <p>In opposition to the <b>Set</b> statement, &lt;Values&gt; 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.</p> <p>To learn more about definitions, follow <a href="#">this link</a>.</p>
Example	<pre>Define %DOKAR%, \LookUp1{#Smarteam.CLASS_ID#}#Smarteam.CN_DRAWINGTYPE# Define %CLASS%, DOC_#Smarteam.CN_DRAWINGTYPE# Define %SAPSTATUS%, \Left2{#Smarteam.CN_SAPSTATUS#}, %VERSION%, \LeftDotRight2{ #Smarteam.REVISION#}, %PART%, 000</pre>

Name	ExtractString
Arguments	<String>, <Range1>, <Variable1>, <Range2>, <Variable2>, ...
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 <b>definitions</b> .
Example	ExtractString #File.Line#, -10, %PART1%, 20-35, %PART2%, 36-, %REST%

Name	Message
Arguments	<MessageType>, <MessageText>
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.
Remarks	The user's reaction (the button pressed) can be accessed by the MESSEAGERESULT variable. Possible values are: "OK", "YES", "NO" or "CANCEL".
Example	Message "Question", "Do you want to create the bill of material?" Goto :End:, , MessageResult =U= No, MessageResult =U= Cancel ... Label :End: Message Success, Operation ended successfully

Name	SendMail
Arguments	<Subject>, <Body>, <Attachment>, <FromAddress>, <FromName>, <ToAddress>, <ToName>, <SMTPServer>, <UserName>, <Password>
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, #Lgoninfo.BapiMessages# or #Lgoninfo.BapiErrorMessages# should be passed as <Body>.
Example	SendMail Error while processing object #Smarteam.CN_ID#, #Lgoninfo.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 <b>Define</b> 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. <b>To learn more about definitions, follow this link.</b>
Example	Set %StartTime%, #System.Time#

Name	SetPrinter
Arguments	<String Printername>
Description	Switches the Windows default Printer to the given Printername
Remarks	Please be sure that there is NO NOT RESPONDING Application in the Taskmanager. Otherwise the JobServer will hang and not respond anymore
Example	SetPrinter OnReleaseA4

Name	DoPrintCommand
Arguments	<String Printername>,<String Command>, <optional String AddCommandSigns, <string optional NoEndDoc>
Description	Does a Printer.Print command with a following printer.enddoc()
Remarks	Printername to use Command that should be executed or be printed : acbk: Bookmark1 if AddCommandSigns = X then the command is putted in %% : %%acbk : Bookmark1%% if NoEndDoc = X then there will be no printer.enddoc() executed
Example	DoPrintCommand PDF,acbk : Bookmark1,X

Name	ResetJobTimer
Arguments	No Arguments
Description	Resets the watching Job-Timer to zero to enlarge the Time for executing the job
Remarks	
Example	ResetJobTimer

Name	Replace
Arguments	<String ResultValue>,<String Source-String>, <String Searchstring>, <String Replacementstring>
Description	Replaces a string in a value
Remarks	You should use a new definition for your result
Example	Set %StringOld%,ThisIsTheOldString Replace %StringNew%,%StringOld%,Old,New  Result : %StringNew% = ThisIsTheNewString



Name	SplitString
Arguments	<String>, <Split>, <Variable1>, <Variable2>, <Variable3>, ...
Description	Splits <String> at <Split> and adds the resulting strings as definitions to the internal memory using <Variables>. <Split> itself can be a string or a single character. 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 <b>definitions</b> .
Example	SplitString #File.Line#, ; , %PART1%, %PART2%, %REST%

## Workflow Functions (FLOW Object)

Name	InitByObjectId
Arguments	<String Classid of Process>, <String Objectid of Process>, <Optional String Seconds try to find the current node in the JobServers InBox>
Description	Get the SmFlowProcess of the given classid and objectid
Remarks	In the Example below, the Jobserver is looking for 120 seconds in the JobServers-Inbox to find a node for the given Process.
Example	new FLOW, Process Process.InitByObjectId #Jobfile.ProcessClass_ID#, #Jobfile.ProcessObject_ID#, 120

Name	AcceptResponse
Arguments	<Optional String, Name of AcceptResponse>
Description	Executes the default <Accept Response> for all current nodes of the initialized process if there is no argument. If there is a AcceptResponse specified executes the named response
Remarks	
Example	new FLOW, Process Process.InitByObjectId #Jobfile.ProcessClass_ID#, #Jobfile.ProcessObject_ID# Process.AcceptResponse Accept

	RejectResponse
Arguments	<Optional String, Name of RejectResponse>
Description	Executes the default <Reject Response> for all current nodes of the initialized process if there is no argument. If there is a RejectResponse specified executes the named response
Remarks	
Example	new FLOW, Process Process.InitByObjectId #Jobfile.ProcessClass_ID#, #Jobfile.ProcessObject_ID#

	Process.RejectResponse	Reject

	OBJECT	
Arguments	<String Fieldname> Fieldname from the SMARTEAM Process Object	
Description	Get's the values from the SMARTEAM Process Object (SmApplic.SmObject)	
Remarks	Be sure to spell the fieldname in the right way. Take care of capital letters	
Example	<pre> new          FLOW, Process Process.InitByObjectId #Jobfile.ProcessClass_ID#, #Jobfile.ProcessObject_ID# set          %ProcessName%,#Process.OBJECT.TDM_NAME# </pre>	

	GetAttachedDocuments	
Arguments	<Optional String SuperClassId> Superclassid of the Documents-Superclass. If not named the Automationserver will take : 5	
Description	Get's a list of all linked documents to the process	
Remarks	Be sure to deliver the right superclassid. For the SmDemo or all derived databases it's : 5	
Example	<pre> New Flow,Process Process.InitByObjectId #Jobfile.PROCESS_CLASS_ID#, #Jobfile.PROCESS_OBJECT_ID#, 60 Process.GetAttachedDocuments 5   Loop Process.Attachments     new SMARTEAM,STJOB     STJOB.SetObject #Process.Attachments.Class_id#, #Process.Attachments.Object_id#   Next </pre>	

## Appendix 2

### 3DSmartDocCreator Server - Available Composer related Functions

**Composer Functions** (These functions require Microsoft Office 2003 or higher)

[CallSync](#)   [CallImageCreation](#)   [CallMapper](#)   [Unzip](#)   [Zip](#)   [ChangeXMLValue](#)  
[GetViewName](#)

#### Composer Functions

Name	CallSync
Arguments	<SyncExe> -> Full Path of Converter <Design File> -> Design File to be converted <seeb XML> -> Convertrules defined in xml file
Description	Exexutes the Sync Integration with the defined values
Remarks	
Example	set %SyncExe%, "C:\Programme\Dassault Systemes\3DVIAComposer\6.3\Bin\3dviaconverter.exe" set %syncXML%, "c:\js\operations\convertshattered63.xml" Set %REFFILE%, #STJOB.CAD_REF_FILE_NAME# Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%, callsync, %syncexe%, %reffile%, %syncxml% DoCopy.WaitForProcess "3DVIACConverter.exe", 1000, 900

Name	CallImageCreation
Arguments	<SyncExe> -> Full Path of Composer <Reffile> -> Composer Poject File thathas to be worked on <seeb XML> -> Convertrules defined in xml file
Description	Exexutes the Composer Output creation with the defined values
Remarks	
Example	set %SeebFile%, %Home%\Operations\createviewpictures.xml set %SyncExe%, "C:\Programme\Dassault Systemes\3DVIAComposer\6.2\Bin\3dviacomposer.exe" Set %REFFILE%, #STJOB.CAD_REF_FILE_NAME# Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%, callImageCreation, %syncexe%, %reffile%, %seebfile% DoCopy.WaitForProcess "3DVIAComposer.exe", 1000, 900

Name	CallMapper
Arguments	<CLASSID> -> Class Id of SmarTeam Design Object <OBJECTID> -> Object Id of SmarTeam Design Object

	<CAD System> -> Integration Rules from this CAD-System will be used
Description	Collects all 3D Data and all Item Meta data and push them to the smgXML file of the selected Design Object
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,callMapper,#Jobfile.Class_ID#, #Jobfile.Object_ID#, "CATIA"

<b>Name</b>	<b>UnZip</b>
Arguments	<Zipname> -> Project that has to be unzipped <Destination> -> Destination Path for unzipping
Description	Unzips a Compoer Project to a named Folder
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,Unzip,%file%,%Home%files

<b>Name</b>	<b>Zip</b>
Arguments	<Zipname> -> Project that has to be zipped <Destination> -> Destination Filename for zipping
Description	Zips a Compoer Project to only one File
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,Zip,%file%,%Home%files

<b>Name</b>	<b>ChangeXMLValue</b>
Arguments	<Filename> -> XML Filename <Identifier> -> Node descriptor <Value> the new value that will be written to that node
Description	Changes a value in a "Seeb" file
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,ChangeXMLValue,%seebfile%,Batch.IOSmgViewFile,%home%\file%\ViewFile%

<b>Name</b>	<b>GetViewName</b>
Arguments	<Viewfile> -> XML Composer View file <Number> -> Number of selected View inside View File
Description	Sets the "System Function Result Value" to the View name of the View File
Remarks	
Example	Define %SYNC%, JobServerComPlugIn.CompAutomation callexternalfunction %SYNC%,,GetViewName,%home%\file%\ViewFile%,%counter% SET %VIEWNAME%,#SYSTEM.FUNCTIONRESULT#

## Appendix 3

### 3DSmartDocCreator Server - Available Inline Functions

\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

\Filename{FILE} result is the filename of FILE (including path)

\Extension{FILE} result is the extension of FILE

\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

\WinDate{SAPDATE} result is a string containing a Windows short date built from SAPDATE string

\SapDate{WINDATE} result is a string containing a SAP date 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

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

<b>Name</b>	<b>\C</b>
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 ", and the result of "\"# is "#.

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

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

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

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

<b>Name</b>	<b>\Trim{STRING}</b>
Description	Result is STRING without leading and trailing space.
Example	The result of "\Trim{ A b c 123 }" is "A b c 123".

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

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

<b>Name</b>	<b>\LeftDotLeftx{STRING}</b>
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".

<b>Name</b>	<b>\LeftDotRightx{STRING}</b>
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".

<b>Name</b>	<b>\LeftSpaceLeftx{STRING}</b>
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".

<b>Name</b>	<b>\LeftSpaceRightx{STRING}</b>
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".

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

<b>Name</b>	<b>\MaskC{STRING}</b>
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".

<b>Name</b>	<b>\UnmaskC{STRING}</b>
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".

<b>Name</b>	<b>\LCase{STRING}</b>
Description	Result is STRING with all characters converted to lower case.
Example	The result of "\LCase{AbCdEfGhI}" is "abcdefghi".

<b>Name</b>	<b>\UCase{STRING}</b>
-------------	-----------------------

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

<b>Name</b>	<b>\LookUpx{KEY}</b>
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>

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

<b>Name</b>	<b>\Filename{FILE}</b>
Description	Result is the filename of FILE (including path).
Example	The result of "\Filename{C:\WorkDir\4711.tif}" is "C:\WorkDir\4711".

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

<b>Name</b>	<b>\Eval{EXPRESSION}</b>
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".

<b>Name</b>	<b>\Intx{NUMBER}</b>
Description	Result is a string of length x containing the binary representation of NUMBER.
Example	The result of "\Int4{9710}" is "%04x" or - in hex representation - "EE250000".



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

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

<b>Name</b>	<b>\SapDate{WINDATE}</b>
Description	Result is a string containing a SAP date (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.

<b>Name</b>	<b>\SwitchFloatSigns{NUMBER}</b>
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".

<b>Name</b>	<b>\SapInternalx{NUMBER}</b>
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".

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

<b>Name</b>	<b>\DocumentKey{DOCTYPE,DOCNUMBER,DOCPART,DOCVERSION}</b>
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 "DRWST4711001" <span style="float: right;">A</span>





#JOBSERVER.WORKDIR# The working directory of the job server  
#JOBSERVER.LASTERROR# Description of the last existing Error during processing job

#### Job File Data

#JOBFILE.NAME# The name of the job file being processed  
#JOBFILE.<ENTRYKEY># The value of a job file entry with key <ENTRYKEY>

#### System Data

#SYSTEM.DATE# The present date  
#SYSTEM.TIME# The present time  
#SYSTEM.ENV.<NAME># The value of environment variable <NAME>  
#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)

#### 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.LINE.NUMBER# The line number of the last line read

[Automationserver.ini]

##### Directory Settings:

- Open the <Homedirectory>\bin\AutomationServer.ini with a Text Editor
- Change the marked rows with proper path information

[RUNTIME]

**COMPOSERPATH**=C:\Programme\Dassault Systemes\3DVIAComposer\6.4\Bin\  
SHAREDCOMPOSERDIR=E:\composer\geomfiles\  
**DATABASENAME**=PLM DB ST

[Explanations]

You can access the runtime values inside your operation like:

Set %ComposerPath%,#GETINIValue.COMPOSERPATH#  
Set %GeomPath%,#GETINIValue.SHAREDCOMPOSERDIR#

## [CREATECOMPOSERFILES]

### Directory Settings:

- Open the CREATECOMPOSERFILES.jod File located in the Homedirectory>\Operations directory
- Change the marked rows with proper Path informations:

### Operation definition \*.JOD):

```
Set %Home%,#SERVER.ROOTDIR#\
Set %ComposerPath%,#GETINIValue.COMPOSERPATH#
Set %SyncExe%,%COMPOSERPATH%3dviaconverter.exe
Set %syncXML%, "%Home%\operations\convertshattered.xml"
```

```
DoCopy.DoKill %Home%\files\*.*
```

```
SetLogMessage 1
STJOB.Run #GETINIValue.DATABASENAME#,X
```

```
STJOB.Login admin
```

## 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
File.PutLine #Smarteam.Object.CN_ID#
File.PutLine %DOCNUM%
```

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 printed to the file, 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 %DOCNUM% is not defined and therefore not replaced by anything, not even by an empty string. This means when performing the File.PutLine, %DOCNUM% is printed to the file and not the desired document number. 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  %DOCNUM%, #Smarteam.Structure.CN_DOCUMENT_NUMBER#
  ...
Next Smarteam.Structure
If  %InsideLoop% = True
  ...
  File.PutLine %DOCNUM%
  ...
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.