IDoc Connector for XML Component (BC-FES-AIT)

HELP.BCFESIDOCXML

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

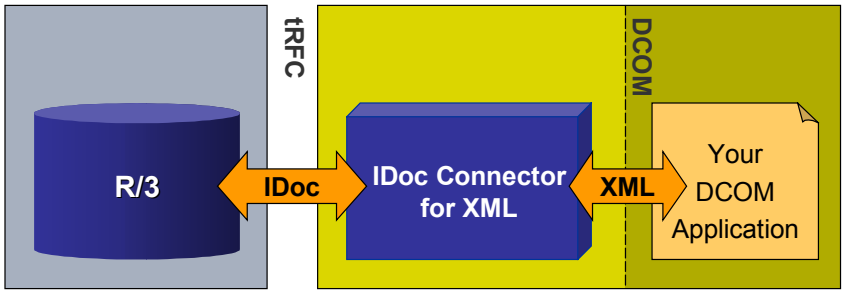
IDoc Connector for XML Component (BC-FES-AIT)

IDoc Connector for XML Component (BC-FES-AIT)

Purpose

The *IDoc Connector for XML* is a COM component that provides an XML interface for working with IDoc documents.

The *IDoc Connector for XML* component accepts your XML documents and sends them as inbound IDocs to a specified R/3 system. Conversely, it can receive outbound IDocs and it translate them into XML documents for your application to use.



One advantage of using XML documents instead of IDocs is that an XML document is easier to write, read and interpret. The IDoc protocol is very particular, for example, about the placement of data fields and spaces in a document. In addition, XML is a standard language. Working with XML documents instead of IDocs allows you to take advantage of available standard parsers.

Features

The *IDoc Connector for XML* component lets you develop DCOM applications that can:

- wait for one or more incoming outbound IDoc(s), receiving them as XML document(s) and also saving them as XML file(s)
- send one or more XML document(s) to an R/3 system as inbound IDoc(s)

  The XML document can be created by your program or it can be taken from an existing XML file

The *IDoc Connector for XML* component does the translation to and from XML automatically.

Implementation Considerations

The *IDoc Connector for XML* is an ActiveX Server. It is only available for Microsoft Windows systems (Windows 95 and up and Windows NT), for DCOM-compliant programs.

The *IDoc Connector for XML* works with R/3 releases 3.1 and higher.

Integration

The *IDoc Connector for XML* component can reside on the same system as your application, or it can reside on a separate system. It communicates with your application through Microsoft DCOM.

IDoc Connector for XML Component (BC-FES-AIT)

The *IDoc Connector for XML* component works with the SAP DCOM Connector [Ext.], allowing you to use *Destination* entries that had been defined in the *SAP DCOM Connector* for logon information (See Using the Component for Creating Inbound IDocs [Page 20]).

The *IDoc Connector for XML* component also works with the DCOM Connector Logon Component [Ext.] (which is a part of the SAP Automation suite of tools and components [Ext.]) to get logon information from an end user (See Using the Component for Creating Inbound IDocs [Page 20]).

The *IDoc Connector for XML* component communicates with the R/3 system through the Transactional RFC (tRFC) protocol. To accept outbound IDocs the component must be set up to be an RFC server.

The R/3 system with which you exchange documents must be set up to send and receive IDocs. This involves configuring an ALE infrastructure and configuring the IDoc interface (See Prerequisite Setup [Page 9]).

Constraints

When sending inbound IDoc documents, the *IDoc Connector for XML* only verifies the syntax of the XML tags in the XML document you specify. It does not verify the resulting IDoc, or the data in it.

Prerequisites for Using the Component

Prerequisite Knowledge

Although you are sending or receiving XML documents when using the *IDoc Connector for XML* component, you are actually sending or receiving IDocs to or from an R/3 system.

This means that when sending an XML document it must convert to an appropriately structured and valid IDoc, and when receiving an XML document you should know how to interpret it as an IDoc.

You should:

- Be familiar with the fundamentals of the IDoc protocol and the general structure of IDocs
- Be able to construct a valid IDoc for the relevant R/3 application

In addition, you or your administrator should be able to set up the ALE and IDoc infrastructure. See Prerequisite Setup [Page 9].

Prerequisite Setup

Although you send or receive XML documents when using the *IDoc Connector for XML* component, you are actually sending or receiving IDocs to or from an R/3 system.

Therefore both the R/3 system and the external system using the *IDoc Connector for XML* component must be set up to send or receive IDocs.

ALE/IDoc Setup on the R/3 System

To allow the R/3 system to either send or receive IDocs, your administrator must set up the R/3 system with the appropriate ALE infrastructure and IDoc settings.

This ALE/IDoc setup involves many tasks, including (but not limited to) setting up an RFC destination, setting up a port for communication, and defining partner profiles. Port number and partner information, for example, are included in the header of the IDocs that you either send or receive.

The complete list of ALE/IDoc setup tasks and their details are outside the scope of this document. For More details, see the following documentation:

- IDoc Interface / Electronic Data Interchange [Ext.]
- ALE Introduction and Administration [Ext.]

ALE/IDoc Setup for the *IDoc Connector for XML* Component

Note the following selections your administrator should make when setting up the ALE/IDoc infrastructure to work with the *IDoc Connector for XML* component:

Defining a Port (Inbound and Outbound IDocs)

When defining a Port (Transaction WE21) for communicating with the *IDoc Connector for XML* component, your administrator should define it as a *Transactional RFC* (*tRFC*) port type.

Defining an RFC Destination for Outbound IDocs

When using outbound IDocs, the *IDoc Connector for XML* acts as the server for RFC Function calls from the R/3 system.

Therefore, for using outbound IDocs with the component your administrator should define an RFC Destination (Transaction SM59) to point to the *IDoc Connector for XML* component.

This RFC Destination should use Connection type *T — TCP/IP connection*.

External System Setup for Outbound IDocs

The other side of preparing the *IDoc Connector for XML* to act as an RFC server is to set up the following on the system on which the *IDoc Connector for XML* component is installed:

- The system must contain a *saprfc.ini* file. The *saprfc.ini* file must contain an entry for the connection with the sending R/3 system:

  The *TYPE* field should be *R.*

  The *GWHOST* and *GWSERV* fields should point to the sending R/3 system.

  The *DEST* field should be the name of the *RFC Destination* as it was defined in Transaction *SM59* on the R/3 system.

Prerequisite Setup

The *PROGID* field should be the *Program ID* entered in Transaction *SM59* on the R/3 system.

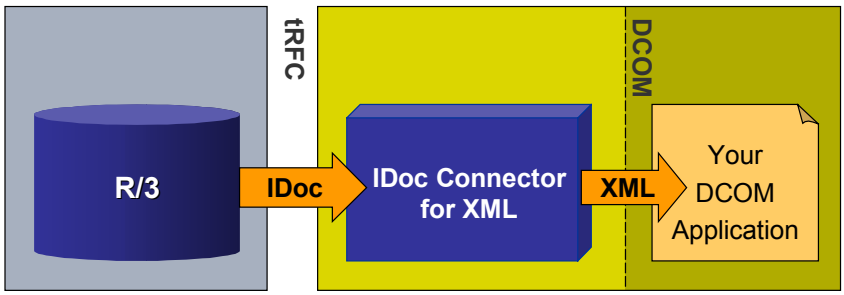
- A Windows environment variable — *RFC_INI* — must point to this *saprfc.ini* file (The value of the variable should contain the full path and file name).

Using the Component for Receiving Outbound IDocs

Purpose

Using the *IDoc Connector for XML* component your application can receive outbound IDoc(s) as XML document(s). The component does the translation from IDoc to an XML document.

Using the Component for Outbound IDocs



When receiving outbound IDocs, the *IDoc Connector for XML* component acts as an RFC server to the R/3 system that is sending the IDoc (Meaning that it receives RFC function calls from the R/3 system).

Prerequisites

- The ALE infrastructure must be set up in the source R/3 system (to allow for sending IDocs).
- The system on which the *IDoc Connector for XML* component is installed must be set up for transactional RFC.

See the details in the Prerequisite Setup [Page 9] topic and in the ALE and IDoc documentation mentioned there.

Process Flow

To use this feature of receiving outbound IDocs as XML documents you use the SAPIDocProcessorOutbound object [Page 14] in your application.

1. Create an instance of the *SAPIDocProcessorOutbound* object. Subscribe to all of its events.
2. Use the Listen method [Page 16] of the *SAPIDocProcessorOutbound* object to wait for the a single IDoc.

   The *Listen* method is a blocking call, meaning that it waits for the IDoc indefinitely. We recommend that you use *Listen* within a separate process or thread, if you wish to not interrupt other processing in your application.

   Once an IDoc is received, the *Listen* method terminates. If you wish to receive multiple IDocs, place the *Listen* method in a loop.
3. After receiving an IDoc, you can choose one of the following courses of action:

Action	How to Perform

Get a document ID for the document, and then get the XML document later. Use this process, for example, if you are getting multiple outbound documents, and you wish to get their count before actually getting the documents.	1. Handle the *IDocReceived* event to get the document ID 2. Use the *PickIDoc* method to get the XML document
Immediately get the resulting XML document	Handle the *IDocReceived2* event. This performs the same action as the two separate steps above.

Result

In addition to supplying your program with the XML document(s), the component also saves the document(s) as XML file(s) in the system *temp* directory.

Visual Basic Example

The following example waits for IDocs in a loop, and then gets the resulting XML document(s).

```
'Declare the outbound object,
'   subscribing to all its events:
Dim WithEvents idoc As SAPIDOCPROCLib.SAPIDocProcessorOutbound
Private Sub GetDoc_Click()
'Create the instance of the outbound object:
Set idoc = New SAPIDOCPROCLib.SAPIDocProcessorOutbound
On Error GoTo bad_news
' Run Listen in a loop to receive multiple IDoc documents.
'...
    idoc.Listen ("IDOCSYS")
                  'IDOCSYS is an RFC Destination
'...
Exit Sub
bad_news:
MsgBox Err.Description & "  " & Err.Source, vbCritical, Err.Number
Exit Sub
End Sub
' VB automatically creates the subroutines
'   for handling all of the events of the object.
'If you choose to handle this routine (for the IdocReceived2 event)
'   you get the XML document as soon as the IDoc document is received:
Private Sub idoc_IDocReceived2(ByVal idocxml As String)
            'VB appends the string idoc_
            '    to create the name of the event handling routine
   MsgBox idocxml
   ' Your program should do something more meaningful
   '   with the XML document
End Sub
'As an alternative,
'   If you chose to handle the IDocReceived event,
'      you would get only document ID,
'      and then you can get the XML document in a separate step:
'Private Sub idoc_IDocReceived(ByVal Id As Long)
```

```
'    MsgBox idoc.PickIDoc(Id)
'End Sub
```

SAPIDocProcessorOutbound Object Reference

Use

The *SAPIDocProcessorOutbound* object allows you to wait for one or more outbound IDoc(s) (IDoc documents sent from an R/3 system) and then receive them as XML document(s).

The *SAPIDocProcessorOutbound* object allows you to either receive the XML document immediately after the IDoc is received, or it allows you to get a document ID with which you can get the XML document at a later time.

For details of the process of receiving outbound documents with the *SAPIDocProcessorOutbound* object, see Using the Component for Receiving Outbound IDocs [Page 11].

Syntax

```
SAPIDocProcLib.SAPIDocProcessorOutbound
```

Property Summary

Name	Type	Description
Trace	ITrace	Use for tracing errors with using the *IDoc Connector for XML*. See Tracing Errors and Messages [Page 32].

Method Summary

Name	Description
Syntax (In Visual Basic)	
Listen [Page 16]	A blocking call that waits for the next IDoc to be sent from the R/3 system
`Sub Listen(rfcdest As String)`	
PickIDoc [Page 17]	Gets an XML document resulting from a single outbound IDoc based on document ID
`Function PickIDoc(Id As Long) As String`	

Event Summary

Name	Description
Syntax (In Visual Basic)	
IDocReceived [Page 18]	Occurs when the outbound IDoc is received, and returns the ID of the XML document. You can later use the *PickIDoc* method to get the actual XML document.
`Sub idoc_IDocReceived(ByVal Id As Long)`	

IdocReceived2 [Page 19]	Occurs when the outbound IDoc is received, and returns the XML document.
`Sub idoc_IDocReceived2(ByVal idocxml As String)`	

See Also

Using the Component for Receiving Outbound IDocs [Page 11]

Listen Method

Use

A blocking call waiting to for single outbound IDoc.

Syntax

```
idocobj.Listen (rfcdest)
```

Parameters

Name	Type	Description
rfcdest	String	The name of the RFC destination. It should be the value of the *DEST* entry in your *saprfc.ini* file, and it should also match the name of the *RFC destination* as entered in Transaction *SM59* on the R/3 system.

Comments

Because it is a blocking call, we recommend that you use it within a separate process or thread, if you wish to not interrupt the other processing in your application.

Once an IDoc is received, the *Listen* method terminates. If you wish to receive multiple IDocs, place the *Listen* method in a loop.

Visual Basic Example

The following section of code waits for an IDoc.

```
idoc.Listen ("IDOCSYS")
```

IDOCSYS is the name of the *RFC destination* in R/3 for the system on which the *IDoc Connector for XML* component is installed. It is also the *DEST* entry in the *saprfc.ini* file on the system on which the *IDoc Connector for XML* component is installed.

See Also

See the code example in Using the Component for Receiving Outbound IDocs [Page 11]. Also see SAPIDocProcessorOutbound object [Page 14], IDocReceived [Page 18], IDocReceived2 [Page 19], and PickIDoc [Page 17]

PickIDoc Method

Use

Gets the XML document resulting from an outbound IDoc. Gets the XML document specified by the ID parameter.

Syntax

```
idocobj.PickIDoc(DocId)
```

Parameters

Name	Type	Description
DocId	Long	The ID of the XML document

Data Type

String

Return Value

Returns the XML document

Comments

You must have first obtained the document ID by using the IDocReceived [Page 18] event.

VB Example

The following section of code first gets the document ID for the received IDoc, and then it uses PickIDoc to get the XML document. It displays the XML text in a message box (your program should provide a more sophisticated handling of the text of the resulting XML document).

```
Private Sub idoc_IDocReceived(ByVal Id As Long)
' Some other processing...
   MsgBox idoc.PickIDoc(Id)
' Displays the XML document in a message box
' (which only displays the end of the document,
'   if the document is long)
End Sub
```

See Also

For a more complete code for receiving outbound documents, see Using the Component for Receiving Outbound IDoc Documents [Page 11].

IDocReceived Event

Use

Occurs when the outbound IDoc is received. Translates the IDoc to an XML document, and saves the XML document as an XML file in the system *temp* directory.

Returns the ID of the XML document.

Syntax

```
IDocReceived_routine(Id)
```

Parameters

Name	Type	Description
Id	Long	The ID of the XML document

Comments

To get the actual XML document use the PickIDoc method [Page 17].

Note that the ID of the document is known within the context of the routine or code handling the *IDocReceived* event. Either use *PickIDoc* within this code, or save the ID into some variable that is available to *PickIDoc*.

VB Example

The following section of code shows the routine for handling *IDocReceived*. It uses *PickIDoc* to get the XML document.

Note that Visual Basic automatically creates the subroutines for handling all events of an object you create. In the following code, VB named the routine *idoc_IDocReceived*.

```
Private Sub idoc_IDocReceived(ByVal Id As Long)
    MsgBox idoc.PickIDoc(Id)
End Sub
```

For a more complete example of using the *SAPIDocProcessorOutbound* object see Using the Component for Receiving Outbound IDocs [Page 11] [Page 11].

See Also

PickIDoc method [Page 17], SAPIDocProcessorOutbound object [Page 14], Using the Component for Receiving Outbound IDocs [Page 11]

IDocReceived2 Event

Use

Occurs when the outbound IDoc is received. Translates the IDoc to an XML document, returns the XML document, and also saves the XML document as an XML file in the system *temp* directory.

Syntax

```
IDocReceived2_routine(idocxml)
```

Parameters

Name	Type	Description
idocxml	String	The actual XML document

VB Example

The following section of code shows the routine for handling *IDocReceived2*. It displays the XML document in a message box. Your program should do something more meaningful with the resulting XML document.

Note that Visual Basic automatically creates the subroutines for handling all events of an object you create. In the following code, VB named the routine *idoc_IDocReceived2*.

```
Private Sub idoc_IDocReceived2(ByVal idocxml As String)
   MsgBox idocxml
End Sub
```

For a more complete example of using the *SAPIDocProcessorOutbound* object see Using the Component for Receiving Outbound IDocs [Page 11] [Page 11].

See Also

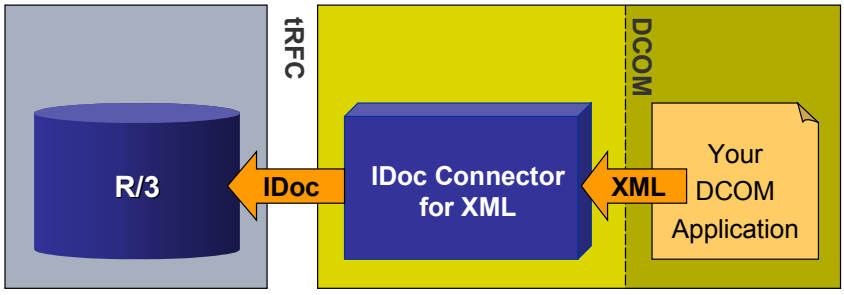
SAPIDocProcessorOutbound object [Page 14], Using the Component for Receiving Outbound IDocs [Page 11] [Page 11]

Using the Component for Creating Inbound IDoc Documents

Purpose

The *IDoc Connector for XML* component can accept one or more XML document(s) or file(s) and send them as inbound IDoc(s) to a specified R/3 system.

Using the Component for Inbound IDocs



Prerequisites

- The receiving R/3 system must be set up to accept inbound IDocs. This involves setting up the ALE and IDoc infrastructure. See the details in the Prerequisite Setup [Page 9] topic and in the ALE and IDoc documentation mentioned there.
- An XML document containing the tags for creating a well formed IDoc (meaning a valid IDoc that is appropriate for the receiving R/3 application) must be available. The XML document can exist in a file or you can create the XML document in your client program.

  Note that when sending inbound IDoc documents, the *IDoc Connector for XML* only verifies the syntax of the XML tags in the XML document you provide. It does not verify the resulting IDoc, or the data in it.

  You can obtain a skeleton XML file for a specific IDoc type by using the SAP Assistant product [Ext.] (which is a part of SAP Automation [Ext.]). See the topic Creating the XML Document for Inbound IDocs [Page 22].

Process Flow

To use this feature, of sending inbound IDocs from XML documents, you use the SAPIDocProcessorInbound object [Page 24] in your application.

1. Create an instance of the *SAPIDocProcessorInbound* object.
2. Provide connection information for logging into the receiving R/3 system.

   You can do so in one of the following ways:

Method	Details

By using an existing *Destination* entry from the SAP DCOM Connector product [Ext.]	Use the SetConnectionParameters method [Page 30] of the SAPIDocProcessorInbound object [Page 24], specifying the Destination name (from the *SAP DCOM Connector*) as a parameter. See the documentation of the SAP DCOM Connector product [Ext.] for how to set up destinations. Note that the complete documentation for the product is available after you install the product.
By allowing an end use to enter connection information through a logon dialog	1. Use the DCOM Connector Logon Component [Ext.] (from the SAP Automation suite of tools [Ext.]) to display a Logon dialog to the user. 2. When the user finishes entering logon information, the *DCOM Connector Logon Component* makes the logon information available through its *Connection* object. 3. Assign this *Connection* object to the Connection property [Page 26] of the SAPIDocProcessorInbound object [Page 24].

3. Send the document to R/3 either synchronously or asynchronously. Do so by calling the SendIDocToR3 method [Page 28] of the SAPIDocProcessorInbound object [Page 24].

   This connects to the destination system using the connection information you have set up in the previous step. It converts the specified XML document into an IDoc, and sends it to the receiving R/3 system.

Visual Basic Example

The following example uses a destination name that had been defined in the SAP DCOM Connector (*SYS1*). It then uses the XML file the end user specifies to send an inbound IDoc to R/3.

```
Private Sub SendDoc_Click()
'Instantiate the inbound object:
Dim idoc As New SAPIDOCPROCLib.SAPIDocProcessorInbound
On Error GoTo bad_news
idoc.SetConnectionParameters ("SYS1")
                        'SYS1 is an existing destination entry
                          'in SAP DCOM Connector
'
'Send the IDoc synchronously.
'   Let the user specify the XML file path in an edit box (Text1):
Call idoc.SendIDocToR3(INBOUND_IDOC_PROCESS, Text1.Text)
MsgBox "IDoc processed and sent to R/3", vbInformation, "Cheers."
Exit Sub
bad_news:
    MsgBox Err.Description & "  " & Err.Source, vbCritical, Err.Number
    Exit Sub
End Sub
```

Creating the XML Document for Inbound IDocs

Use

When using the *IDoc Connector for XML* to send inbound IDocs, you provide an XML document, and the *IDoc Connector for XML* component translates it to an IDoc.

While the *IDoc Connector for XML* component checks the syntax of the XML document, it does not check or validate the resulting IDoc.

It is your responsibility to provide an XML document that contains:

- all the relevant segments and fields for the type of IDoc you are sending
- values for all the relevant fields

The Contents of the XML Document

The XML document is built as a hierarchy of all the IDoc segments and their fields.

The XML tags represent either an IDoc segment or field name.

Field values are included within each tag.

The order of the XML tags for the segments and field should be in their order of appearance in the resulting IDoc.

Example

The following is a section from the beginning of an XML document for IDoc type *ORDERS01*:

```
<?xml version="1.0"?>
<ORDERS01>
   <IDOC BEGIN="1">
      <EDI_DC40 SEGMENT="1">
         <TABNAM>EDI_DC</TABNAM>
         <MESTYP>ORDERS</MESTYP>
         <DIRECT>2</DIRECT>
         <SNDPRN>IDOCCLASS</SNDPRN>
         <SNDPRT>LS</SNDPRT>
         <SNDPOR>A000000009</SNDPOR>
         <STDMES>ORDERS</STDMES>
         <IDOCTYP>ORDERS01</IDOCTYP>
      </EDI_DC40>
      <E1EDK01 SEGMENT="1">
         <SEGNAM>E2EDK01</SEGNAM>
         <CURCY>DEM</CURCY>
         <WKURS>1.00000</WKURS>
         <ZTERM>0001</ZTERM>
         <BSART>NB</BSART>
         <BELNR>4500000329</BELNR>
         <RECIPNT_NO>WTBLIEF</RECIPNT_NO>
      </E1EDK01>
```

Using SAP Assistant to Start the XML Document

Using the SAP Assistant product [Ext.] (which is a part of SAP Automation [Ext.]) you can get a blank XML file (containing tags without values) for a specific IDoc type. You can then use this XML file as a skeleton for creating the complete XML document.

Prerequisites

1. The SAP Assistant must be Installed on your computer.
2. A *Destination* entry must be defined for the receiving R/3 system. See the documentation of the SAP Assistant product [Ext.] for how to define destinations.

Procedure

1. Invoke the SAP Assistant product.
2. Log into the receiving R/3 system.
3. Choose *File → Blank XML doc. for IDoc Type*.
4. Enter a directory name for the resulting XML file (Target directory), and enter the IDoc type.
5. SAP Assistant produces a blank XML file for the specified IDoc type.

   You can now use this skeleton to build the complete XML document. You can do so with an XML editor or you can do so programmatically.

   Regardless of whether you are building the XML document interactively or through your program, your goal is to:

   - Copy segments and fields for repeating items, for example, repeat the segments and fields representing line items in a Sales Order document.
   - Delete unnecessary segments and fields.
   - Enter values for IDoc fields.

See the SAP Assistant documentation [Ext.] for general instructions on using the product.

SAPIDocProcessorInbound Object Reference

Use

The *SAPIDocProcessorInbound* object allows you to create and send one or more inbound IDoc(s) from XML document(s).

You can specify that an IDoc be sent to the R/3 system synchronously or asynchronously.

Syntax

```
SAPIDocProcLib.SAPIDocProcessorInbound
```

Visual Basic Example

```
Dim idoc As New SAPIDocProcLib.SAPIDocProcessorInbound
```

Property Summary

Name	Description
Connection [Page 26]	Returns or sets a DCOM-Connector-Logon [Ext.] type object containing the necessary parameters for connecting to the destination R/3 system
Trace	Use for tracing errors with using the *IDoc Connector for XML*. See Tracing Errors and Messages [Page 32].

Method Summary

Name	Description
Syntax (In Visual Basic)	
SendIDocToR3 [Page 28]	Converts a specified XML document into an IDoc and sends it to the receiving R/3 system.
`SendIDocToR3(rfcfunction As IDOC_RFC_NAME, xmlsource As String)`	
SetConnectionParameters [Page 30]	Specifies the connection information for logging onto the receiving R/3 system when sending an inbound IDoc. Uses information from a *Destination* entry in the SAP DCOM Connector [Ext.], allowing you to supplement or override the information in it.
`SetConnectionParameters(Destination As String, [UserID As String], [Password As String], [Client As String], [Language As String])`	

See Also

For details of the process of sending inbound documents with the *SAPIDocProcessorInbound* object, see Using the Component for Creating Inbound IDoc Documents [Page 20].

Connection Property

Use

Provides the necessary information for logging onto the receiving R/3 system when sending an inbound IDoc through the *IDoc Connector for XML* component.

You can set the *Connection* property by assigning a connection object to the *Connection* property. You can obtain a connection object by using the DCOM Connector Logon Component [Ext.] (which is a part of the SAP Automation suite of tools).

Obtaining a Connection Object

Use the DCOM Connector Logon Component [Ext.] to get a *Connection* object for the *Connection* property:

1. Instantiate the *Connection* object of the *DCOM Connector Logon Component*.
2. Use the *Logon* method of the *DCOM Connector Logon Component* to display a Logon dialog to the end user.
3. When the user enters the information and dismisses the dialog (when the call to *Logon* returns successfully), the *Connection* object of the *DCOM Connector Logon Component* contains the logon information.
4. Assign the *Connection* object of the *DCOM Connector Logon Component* to the *Connection* property of the *SAPIDocProcessorInbound* object.

Syntax

***IdocProcOutboundObj*.Connection**

Data Type

Object

VB Example

The following example lets an end user enter the logon information. It uses this information to set the *Connection* property of the *SAPIDocProcessorInbound* object. It then sends an inbound IDoc from an XML file.

```
Private Sub Inbound_Click()
'Instantiate the inbound object:
Dim idoc As New SAPIDOCPROCLib.SAPIDocProcessorInbound
'Instantiate the connection object of the DCOM Connector Logon
Component:
Dim conn As New Connection
On Error GoTo bad_news
'Use the Logon method of the DCOM Connector Logon Component
'   to display a dialog box to the user
'   for entering connection information.
'   If the call succeeded,
'   the user had entered the information.
'   Assign the resulting connection object
'   to the Connection property of the inbound object.
```

```
If conn.Logon() then idoc.Connection = conn
' Send the inbound IDoc document
'   from myXML XML file:
Call idoc.SendIDocToR3(INBOUND_IDOC_PROCESS, "myXML.XML")
MsgBox "IDoc processed and sent to R/3", vbInformation, "Cheers"
Exit Sub

bad_news:
    MsgBox Err.Description & "  " & Err.Source, vbCritical, Err.Number
    Exit Sub
End Sub
```

See Also

As an alternative to setting the *Connection* property, you can provide the necessary connection information by specifying an existing *Destination* entry from the SAP DCOM Connector [Ext.]. See the SetConnectionParameters method [Page 30] for details.

SendIDocToR3 Method

Use

Converts the specified XML document into an IDoc, and then sends it to the R/3 system as an inbound IDoc.

The specified XML document can come from a file or you can create it in your client application.

Syntax

```
SendIDocToR3(rfcfunction, xmlsource)
```

Parameters

Name	Type	Description	
rfcfunction	IDOC_RFC_NAME	The name of the RFC function to use for sending the IDoc to R/3. `IDOC_RFC_NAME` is an enumerated type that can take one of the two following values:	
		`IDOC_INBOUND_ASYNCHRONOUS`	Send the IDoc asynchronously
		`INBOUND_IDOC_PROCESS`	Send the IDoc synchronously
xmlsource	**String**	The XML source to use for converting into an inbound IDoc. The string of this parameter can contain either one of the following: • the full path to an XML file • the actual contents of the XML document	

VB Example

The following example uses the *myXML.XML* file in the root directory on the *C:* drive as the source XML. It sends an IDoc synchronously to R/3:

```
Call idoc.SendIDocToR3(INBOUND_IDOC_PROCESS, "C:\myXML.XML")
```

For a more complete example of sending an inbound IDoc with the *IDoc Connector for XML* component, see Using the Component for Creating Inbound IDoc Documents [Page 20].

Comments

- The source XML file or actual text (as specified in the second parameter) must be a well formed XML document. It should contain the appropriate data and tags for creating a valid IDoc document. However, the *SendIDocToR3* method only check the syntax of the source XML document. It does not check the validity of the IDoc resulting from this XML document.

- Before sending the document, make sure that you supply the necessary connection information to the *SAPIDocProcessorInbound* object. You can do so by either using the SetConnectionParameters method [Page 30], or by setting the Connection property [Page 26]. See the details in these topics.

See Also

Using the Component for Creating Inbound IDoc Documents [Page 20], SetConnectionParameters method [Page 30], Connection Property [Page 26]

SetConnectionParameters Method

Use

Specifies connection information for logging onto the receiving R/3 system when sending an inbound IDoc through the *IDoc Connector for XML* component.

The method takes most or all of the connection information from an existing *Destination* entry in the SAP DCOM Connector [Ext.].

If the *Destination* entry in the *SAP DCOM Connector* does not include all the information, your program can specify the missing connection information as parameters when calling this method.

You can also override some of the information in that *Destination* entry in the call to this method. For example, you can specify a different logon language than the language specified in the *Destination* entry.

Syntax

```
SetConnectionParameters(Destination, [UserID], [Password], [Client],
[Language])
```

Parameters

Name	Type	Description
Destination	String	The name of the *Destination* entry as specified in the SAP DCOM Connector [Ext.].

Optional Parameters

You can use the following parameters to supplement or override the information in the *Destination* entry:

Name	Type	Description
UserID	String	User ID (also called Username) for logging onto the destination R/3 system.
Password	String	Password used by the user specified in UserID for logging onto the destination system
Client	String	Client number to use in the destination system
Language	String	Language to use when logging onto the destination system

VB Example

The following example uses the Destination entry *SYS1* from the SAP DCOM Connector [Ext.]:

```
SetConnectionParameters("SYS1")
```

The following example uses the *Destination* entry *SYS1*, but it overrides the *Language* to be English:

```
SetConnectionParameters("SYS1",,,, "EN")
```

For a more complete example of sending an inbound IDoc with the *IDoc Connector for XML* component (after specifying the connection information with SetConnectionParameters) see Using the Component for Creating Inbound IDoc Documents [Page 20].

Comments

- The destination system entry must exist in the SAP DCOM Connector product [Ext.]. The *SAP DCOM Connector* creates an entry in the Windows Registry for the destination system. You can create a destination system entry using the *SAP DCOM Connector* product.

  You can find the *SAP DCOM Connector product* in the Windows *Start* menu under *SAP Frontend* or under *SAP Automation*.

- You can also define destination entries (or allow an end user to define destination entries) by using the DCOM Connector Logon Component [Ext.] (from SAP Automation [Ext.]).

- The *IDoc Connector for XML* component does not write any information back to the *SAP DCOM Connector*'s *Destination* entry. Any connection parameters you specify in this method apply only to the SAPIDocProcessorInbound object [Page 24] of this component.

See Also

As an alternative to using the *SetConnectionParameters* method, you can provide the necessary connection information by using the DCOM Connector Logon Component [Ext.] (from SAP Automation [Ext.]), and then setting the Connection property [Page 26] of the *SAPIDocProcessorInbound* object. This allows you to get the information from an end user.

Tracing Errors and Messages

Use

To create a log file that traces errors that occur when you use the *IDoc Connector for XML* component with your application, you use the *ITrace* object.

You can use this log file to communicate problems to SAP support regarding the use of the component.

You can also log errors that occur within your application into the same file.

Procedure

To log IDoc Connector for XML errors:

1. Instantiate the *ITrace* object.
2. Set the *TraceLevel* property of the *ITrace* object.

   The *TraceLevel* property uses an enumerated type, with the following values:

Trace Level	Constant	Meaning
0	NOMESG	Nothing is traced
1	ERRMESG	Only errors are traced
2	MESG	Both errors and informational messages are traced

   Notice that using trace level 2 includes both messages and errors.
3. Open the log file with the *OpenFile* method of the *ITrace* object, specifying the full path of the log file.
4. Assign the *ITrace* object to the *Trace* property of the SAPIDocProcessorOutbound object [Page 14].

To log errors within your application to the same log file:

1. Get the name of the log file using the *filename* property of the *ITrace* object.
2. Set the *TraceLevel* property to the desired level.
3. Depending on the trace level you have set, when error or message occurs, use the appropriate Write method to log the error or message to the file:

If you specify trace level…	You can use…
1 or 2	either WriteErrMsg (without a timestamp) or WriteErrMsgTime (with a timestamp)
2	either WriteMsg or WriteMsgTime

   Notice that if you specify trace level 2, both errors and messages are traced, and therefore you can use both WriteErrMsg and WriteMsg to log both errors and messages. However, when you specify trace level 1, only errors are traced, so using WriteMsg or WriteMsgTime does not result in any log entry.

Example

The following example logs both errors and messages that occur when using the *IDoc Connector for XML* component:

```
Dim oTrace As New ITrace
oTrace.FileOpen("C:\temp\log1.txt")
oTrace.TraceLevel = 2
Set idoc.Trace = oTrace
```