Repository Services Component (BC-FES-AIT)

HELP.BCFESDEH

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Content

Repository Services Component (BC-FES-AIT)

Repository Services Component (BC-FES-AIT)

Purpose

The Repository Services Component provides read access to SAP R/3 business object and RFC function module metadata to external programs and applications.

The Repository Services Component provides a standard access interface to R/3 objects and encapsulates the mechanism required to physically access the underlying R/3 system and database.

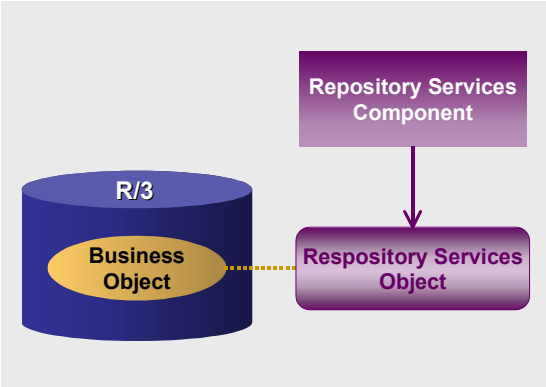
Implementation Considerations

You can use the Repository Services component in programs that follow the Microsoft Component Object Model (COM).

Integration

The Repository Services component allows you to create a Repository Services object on a client computer, which holds the metadata of an R/3 business object or a function module.

The following diagram illustrates, as an example, a Repository Services object representing a SAP business object.



The details of the metadata of the SAP business object or of the RFC function module are contained in a hierarchy of objects under the Repository Services object [Page 12].

Features

The Repository Services component allows you read SAP business object and SAP function module metadata.

The Repository Services component allows you to access this metadata using a live connection to an R/3 system. We refer to this as working online [Page 14].

The Repository Services component also allows you to save metadata in a local repository database [Page 18]. This enables access to the metadata offline, that is, without a connection to the R/3 system.

What's New in Release 4.6A?

Error handling has changed in Release 4.6A to use the COM error handling. As a result, the various methods of the Repository Services component now return either True or False, instead of Success or Failure. In addition, the LastError property of IRepositoryServices is no longer in use.

What's New in Release 4.6B?

The Repository Services Component now works with R/3 Releases 3.0 A and greater.

System Requirements

Development Requirements

To create applications using the Repository Services component you need the following:

- Windows NT 4.0, Windows 95, or the Windows 98 operating system
- The SAP DCOM Connector product installed, which in turn requires:
  - Visual C++ version 5.0 or higher
  - Windows NT 4.0 Option Pack including the MTS development environment
- SAP R/3 System Release 3.0 A or higher.

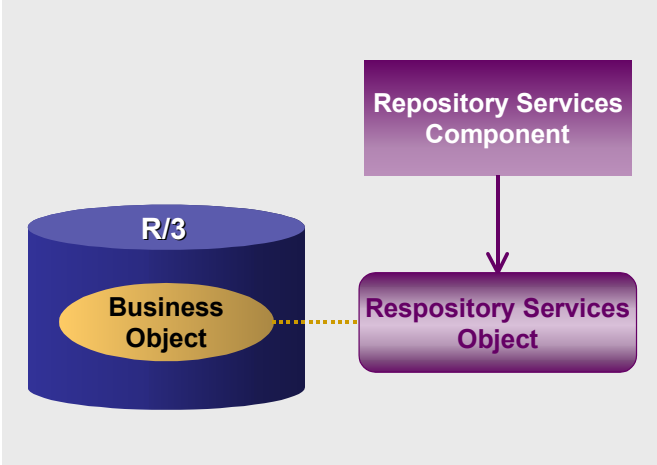
Run-time Requirements

The end user of an applications using the Repository Services component needs the same components as listed above.

Repository Services Object Hierarchy

Use

The Repository Services component allows you to create a Repository Services object, representing the metadata of an SAP business object or of an RFC function module.

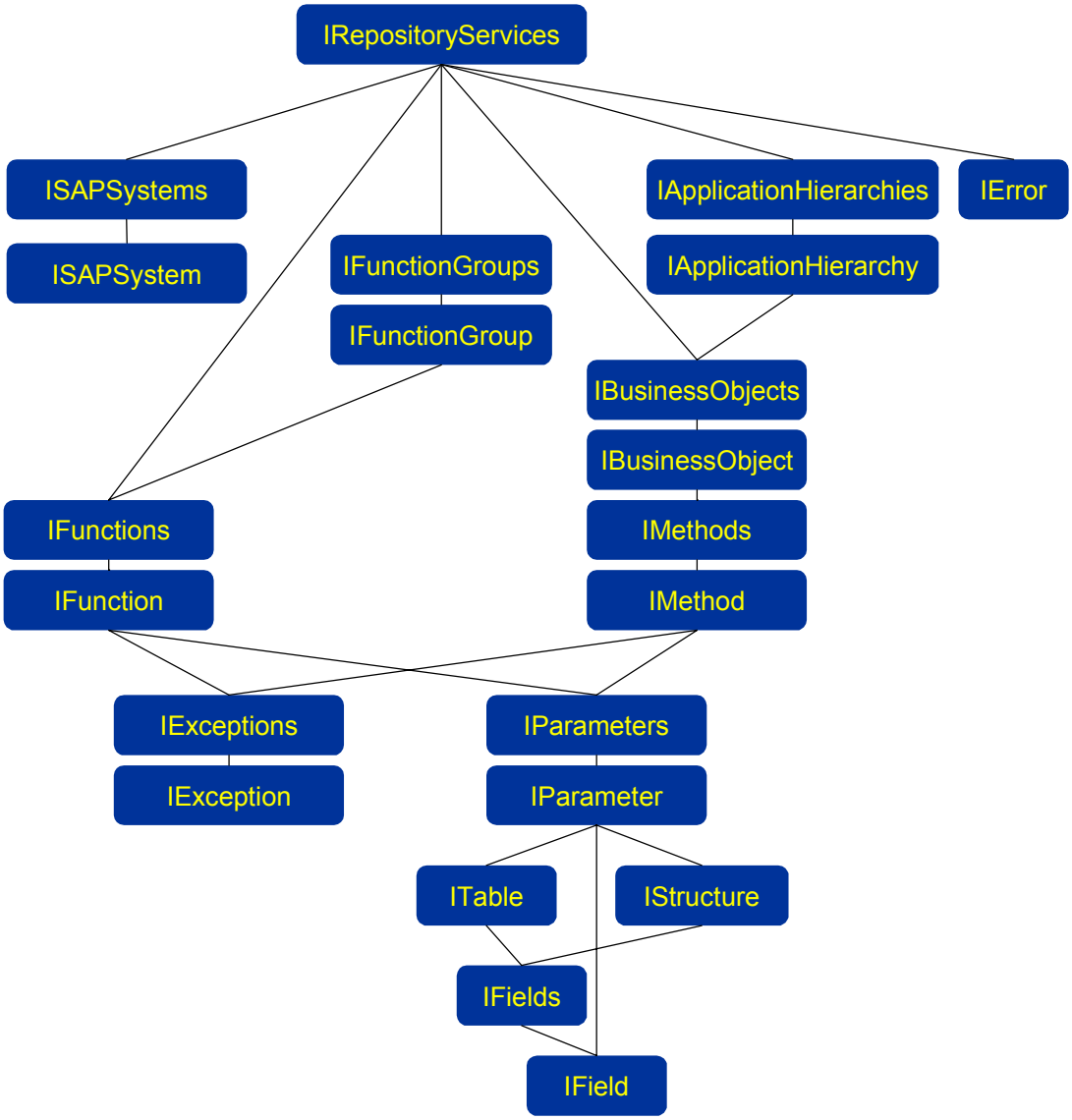


The Repository Services component also allows you to create other objects, all related to the Repository Services object, to hold the details of the metadata of the SAP business object or function module.

Using the various methods and properties of these objects you can find out the exact details of the parameters of a function module, for example, including their data type and documentation. For a business object, as another example, you can find out what are its methods (BAPIs), and for each method, what are its parameters.

Structure

The following diagram shows the hierarchy relationship among the Repository Services objects.


IRepositoryServices
ISAPSystems
ISAPSystem
IFunctionGroups
IFunctionGroup
IApplicationHierarchies
IApplicationHierarchy
IError
IBusinessObjects
IBusinessObject
IFunctions
IFunction
IMethods
IMethod
IExceptions
IException
IParameters
IParameter
ITable
IStructure
IFields
IField

Working Online

Use

Using the Repository Services component you can work in online mode, that is, you can read metadata information using a live connection to an R/3 system.

Prerequisites

Before you can access any metadata in online mode through the Repository Services, you must set up the necessary connection information in the Repository Services object.

Why You Need to Set up The Connection Property?

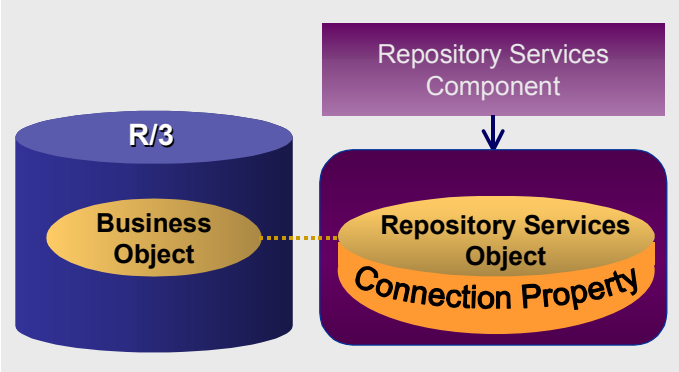
The Repository Services object represents an SAP business object or an RFC function module.

When working online, the Repository Services component needs to be connected the R/3 system containing the SAP business object in order to get its metadata.

The Repository Services component uses services from the SAP DCOM Connector product to handle connections to an R/3 system when a connection is necessary. It does not actively establish a connection.

To allow for a connection to be established when needed, each object requiring a connection must contain connection information in their properties.

The Repository Services object therefore contains a Connection property to hold this required information.

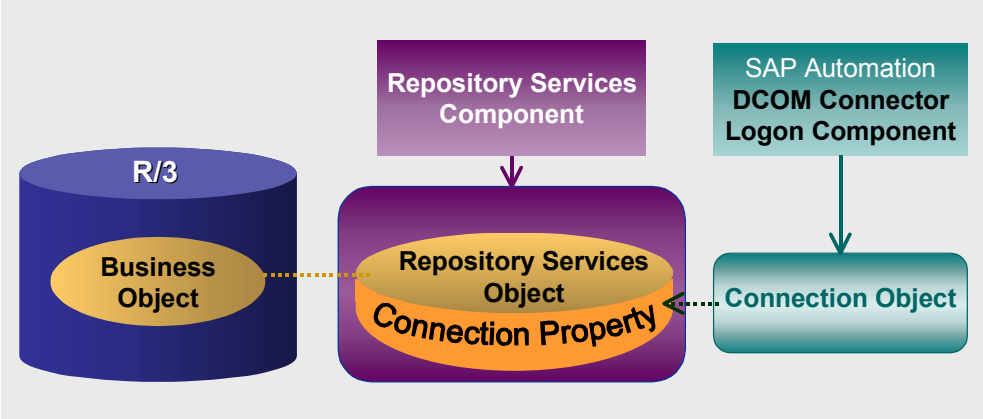


The Connection property points to a Connection object, whose properties may hold the various parameters of logon information.

Programs using the Repository Services component can use the SAP Automation DCOM Connector Logon Component [Ext.] to create a Connection object.

With the SAP Automation DCOM Connector Logon Component you can present a Logon dialog to your end user, which then fills out the various properties of the Connection object.

You then assign this connection object to the Connection property of the Repository Services object.



Procedure

1. Set up at least one destination system with the SAP DCOM Connector Destination editor.

   This creates a destination entry in the Windows Registry. This destination entry may include client, user ID, and even password.

   See the SAP DCOM Connector notes, or the online help for the SAP Automation DCOM Connector Logon Component [Ext.].

2. Create a Connection object with the SAP Automation DCOM Connector Logon Component [Ext.].

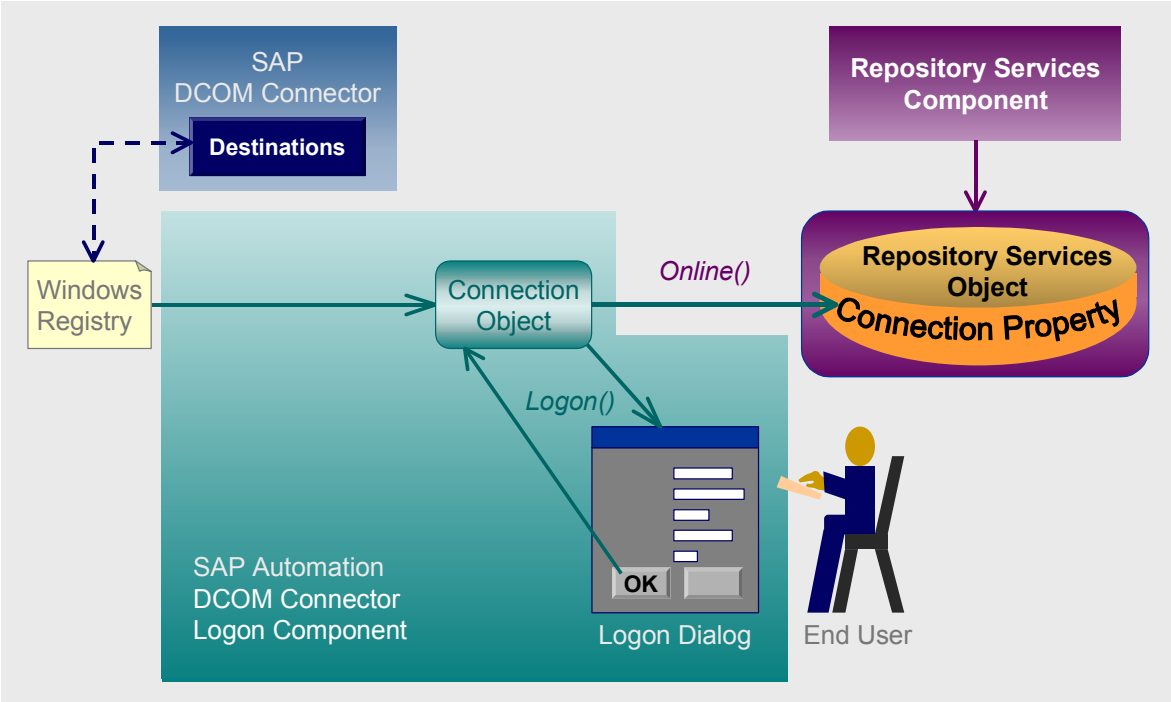
3. Set up the properties of the Connection object to the values of the system you wish to use.

   You can set up the values programmatically, or you can use the Logon method of the SAP Automation DCOM Connector Logon Component to display a Logon dialog to your end user. When the user chooses OK at this dialog, the various properties of the Connection object are filled with the values entered at the dialog.

   If you set up the properties of the Connection object, then their values override any logon values that exist in the Windows Registry when the connection is established.

4. Assign the Connection object to the Connection property of the Repository Services object by either setting the Connection property [Page 34] or by using the Online method [Page 47].

The following illustration summarizes the process of setting up an online mode.



Result

After setting the Connection property of the Repository Services object you are ready to use the various Repository Services objects, their methods and properties.

Whenever an object or method you are using requires a connection to an R/3 system, the connection is established automatically for you.

Example

The following example is a VB function for setting up the connection parameters. It uses the SAP Automation DCOM Connector Logon Component to create a Connection object, and to display a Logon dialog to the end user.

```
Private Function R3Logon() As Object
    Dim oConn As Connection
    ' Create the Connection object
    Set oConn = New SAPLogonLib.Connection
    ' use the Logon method of the
    '         SAP Automation DCOM Connector Logon Component
    '         to display a Logon dialog to the user
    oConn.Logon
    Set R3Logon = oConn
    Set oConn = Nothing
End Function


Dim oRep As New RepositoryServices
Dim oConn As Object
```

```
' Log onto R/3 using the R3Logon function shown above
Set oConn = R3Logon()
oRep.online (oConn)
```

Note that although the example above does not actively establish a connection to R/3 (a call to a BAPI or an RFC that requires a connection automatically establishes such a connection), it is called R3Logon for compatibility with older sample code, which was using the Logon Control for establishing a connection.

Previous ersions of the Repository Services component (for R/3 release 4.5A and earlier) used the SAP Automation Logon Control. The following code for the R3Logon function uses the Logon Control to create the Connection object. It Also actively establishes a connection to R/3. Use this code if you are using earlier versions of the Repository Services component

```
Private Function R3Logon() As Object
    Dim oLogonControl As Object
    Dim oLogon As Object
    Set oLogonControl = CreateObject("SAP.LogonControl.1")
    If Not oLogonControl Is Nothing Then
        Set oLogon = oLogonControl.NewConnection
        If oLogon.logon() Then _
            Set R3Logon = oLogon
    End If
    Set oLogonControl = Nothing
    Set oLogon = Nothing
End Function
```

You can use the sample code in other sections of this Help document without any changes regardless of whether you use the code using the SAP Automation Logon Control or the code using the SAP Automation DCOM Connector Logon Component.

See Also

What's New? [Page 10], SAP Automation DCOM Connector Logon Component [Ext.], Working offline with a local repository [Page 18]

The Local Repository: Working Offline

Use

The Repository Services product allows you to copy metadata you obtain from an R/3 system into a local repository. You can then work offline, that is, you can access the desired metadata from the local repository, instead of from the original R/3 system.

Working offline allows you to work with the metadata when the R/3 system is not available, or when connecting to the R/3 system would be too slow.

Since metadata usually does not change very often, working offline is a good alternative to connecting to a live R/3 system for retrieving metadata.

Integration

The local repository is a Microsoft Access database. Its file type is MDB.

Features

You can store metadata from one or more R/3 systems in a single local repository.

You can create one or more local repositories, but you always work with one of them at a time.

Repository Services provides special methods for working with local repositories:

- Connecting to a local repository
- Writing data into the local repository
- Deleting all the data that belongs to a specific R/3 system from the local repository

Reading data from the local repository is done with the same methods used for reading data from an R/3 database. The Repository Services automatically uses the local repository if you are working in an offline mode.

Activities

You start using the local repository by using the Offline method, in which you specify the name of the local repository.

The Repository Services creates the local repository database automatically the first time you use the Offline method with a database, if that database does not exist yet.

The following table lists the tasks you can perform when working with the local repository. It lists the method or methods of the IRepositoryServices [Page 30] you can use to perform each of these tasks.

Task	IRepositoryServices Method(s)

Connect to the local repository and work in offline mode. Note that if you are also connected to a live R/3 database, that connection takes precedence, meaning that if you use a method for reading data, the Repository Services reads that data from the R/3 database, not from the local repository.	Offline [Page 46]
Store metadata, which you have read from an R/3 system, into the local repository	The various *Write* methods: WriteAppHierarchies [Page 50], WriteAppHierarchy [Page 51], WriteBO [Page 53], WriteBOTree [Page 54], WriteFunctionGroup [Page 56], WriteFunctionGroups [Page 58], WriteRFC [Page 59]
Get the list of SAP R/3 systems whose objects exist in the local repository	SAPSystems [Page 48]
Specify which R/3 system objects to work with, when reading from the local repository	SetSAPSystem [Page 49]
Read metadata from the local repository	ApplicationHierarchies [Page 39], BusinessObjects [Page 40], FunctionGroups [Page 44], Functions [Page 45] (The same functions that are used for reading from an R/3 system)
Delete all the data belonging to the R/3 system you specify from the repository.	DeleteSAPSystem [Page 43]

Reading from the Local Repository

Use

Reading data from the local repository allows you to retrieve metadata offline, that is, retrieve metadata without being connected to the R/3 system containing that metadata.

Procedure

When reading metadata with the Repository Services you access one R/3 system at a time. When reading data in offline mode you also need to work with objects that belong to one R/3 system at a time.

Since metadata originating from different systems can be stored in the same local repository, you need to specify which R/3 system's objects you are going to work with.

The following table summarizes the steps for reading data from the local repository, and the methods to use for performing these steps.

Step	Method(s)	
1.	Connect to the local repository.	Offline [Page 46]
2.	(Optional) Get the list of SAP R/3 systems contained in the local repository. This is the list of all the systems to which the objects in the local repository belong. You do not need to perform this step if you know the R/3 system to which the objects you wish to work with belong.	SAPSystems [Page 48]
3.	Specify which of these systems to work with.	SetSAPSystem [Page 49]
4.	Read metadata from the local repository using the same functions that are used for reading from an R/3 system. Note that if you are also connected to a live R/3 database, the live R/3 connection takes precedence, meaning that the Repository Services reads the data from the R/3 database, not from the local repository.	ApplicationHierarchies [Page 39], BusinessObjects [Page 40], FunctionGroups [Page 44], Functions [Page 45]

Example

The following VB code example looks for the objects belonging to a specific SAP system in the local repository, and then gets the list of business objects belonging to that system:

```
Dim oRep As New RepositoryServices
Dim oAppHiers As IApplicationHierarchies
Dim oAppHier As IApplicationHierarchy
Dim oBusObjs As IBusinessObjects
Dim oBusObj As IBusinessObject
Dim oSyss As ISapSystems
Dim Count As Integer

If oRep.offline("c:\sapreps.mdb") Then
```

```
    Set oSyss = oRep.sapsystems()
    For Count = 1 To oSyss.Count
        ' (Your code for determining which system to look at)
        If StrComp(oSyss(Count).Name, "ESS", vbtextcomprae) = 0 Then _
        oRep.SetSAPSystem (oSyss(Count))
    Next
End If
'Getting list of ApplicationHierarchy objects
Set oAppHiers = oRep.ApplicationHierarchies()
For Count = 1 To oAppHiers.Count
    Set oAppHier = oAppHier.Item(Count)
    ' Your code for using the application hierarchy object,
    '    for example:
    'Msgbox oapphier.description
    'Get the list of Business Objects
    Set oBusObjs = oAppHier.BusinessObjects()
    For Count = 1 To oBusObjs.Count
        Set oBusObj = oBusObjs.Item(Count)
        ' Your code for using the business object
        '   for example:
        'Msgbox oBusObj.name
    Next
Next
```

Copying Data from R/3 to the Local Repository

Use

To work with a local repository you must first populate the local repository database with metadata from the R/3 database.

You can copy all of the metadata of all business objects and RFC functions from an R/3 system, or you can copy a subset of this metadata.

Prerequisites

To copy data from an R/3 system to the local repository you must be connected to both the R/3 system and the local repository.

To connect to the R/3 system from which you wish to copy data you use the SAP Logon Control outside of the Repository Services product.

Procedure

1. Use the Online [Page 47] method to point the IRepositoryServices object to the R/3 connection you have established through the Logon Control. The Online method assigns the logon object to the IRepositoryServices you are working with.

   You can also use the Connection property [Page 34] of the IRepositoryServices object to perform this task.

2. Use the Offline [Page 46] method to connect to the local repository, specifying the full path of the local database location and name.

   If the local repository exists, the Repository Services opens it. If the local repository database does not exist, the Repository Services creates it first.

   The Repository Services also verifies the integrity of the local repository database, to check that it was not manipulated manually.

3. Get the desired metadata from the R/3 system. Use the read methods, such as ApplicationHierarchies [Page 39], BusinessObjects [Page 40], FunctionGroups [Page 44], and Functions [Page 45].

   Since the connection to the R/3 takes precedence, the Repository Services reads the data from R/3 even though you are connected to both the local repository and the R/3 database at the same time.

4. Use the CreateSAPSystem [Page 42] method to create the ISAPSystem object (representing the R/3 system you are copying from) in the local repository.

Use the CreateSAPSystem even if the ISAPSystem object for that R/3 system exists. If that ISAPSystem already exists, the method returns a pointer to it, allowing you to use it in the methods for writing the data into the local repository, as in the next step.

5. Write the object or collection of objects to the local repository using the appropriate Write method, such as WriteAppHierarchies [Page 50], WriteAppHierarchy [Page 51], WriteBO [Page 53], WriteBOTree [Page 54], WriteFunctionGroup [Page 56], WriteFunctionGroups [Page 58], WriteRFC [Page 59].

   For example, if you retrieved a business object from the R/3 system with the BusinessObjects method, then you can add that business object to the local repository with the WriteBO method. To add all the children objects of that business object to the local repository, as well as the business object itself, use the WriteBOTree method.

Examples

The following example copies business objects from an R/3 system into the local repository. It shows two alternative codes for selecting business objects: one section of the code shows how to save all of the application hierarchies and their child objects, and one section of the code shows how to save only a specific application hierarchy, and its child objects:

```
Dim oRep As New RepositoryServices
Dim oAppHiers As IApplicationHierarchies
Dim oAppHier As IApplicationHierarchy
Dim oSys As ISAPSystem
Dim oConn As Object
Dim Count As Integer
Dim Count1 As Integer

' Log onto R/3 using the function shown in the example
'           for the Working Online topic
Set oConn = R3Logon()

If Not oConn Is Nothing Then
    oRep.online (oConn)
' Connect offline
'     This creates the local database file
'     if it does not exist
    oRep.offline ("c:\sapreps.mdb")
    Set oSys = oRep.CreateSAPSystem(oConn.destination, _
               oConn.HostName, oConn.System, oConn.systemnumber, _
               oConn.applicationserver, oConn.saprelease)
    'Get a list of ApplicationHierarchy objects
    Set oAppHiers = oRep.ApplicationHierarchies()
    'Save all the application hierarchy objects
    '     and their child objects,
    '     such as BusinessObjects, their methods, parameters etc
    oRep.writeapphierarchies oSys, oAppHiers
    For Count = 1 To oAppHiers.Count
        Set oAppHier = oAppHier.Item(Count)
        ' Your code for using the application hierarchy object
        '    for example: Msgbox oapphier.description
        ' As an alternative to
        '   saving all of the application hierarchies,
        '   as in the code above
```

```
        '   you can save only a single application hierarchy object
        '   (the current application hierarchy object),
        '   and all of its child objects, such as
        '   BusinessObjects, their methods, parameters etc.
        '   Use the following code:
        'oRep.writeapphierarchies oSys, oAppHier
        Set oBusObjs = oAppHier.BusinessObjects
        For Count1 = 1 To oBusObjs.Count
            Set oBusObj = oBusObjs.Item(Count1)
            ' Your code for using the current business object
            'Msgbox oBusObj.name
        Next
    Next
End If
```

The following example copies all of the business objects of a particular R/3 system into the local repository:

```
Dim oRep As New RepositoryServices
Dim oBusObjs As IBusinessObjects
Dim oBusObj As IBusinessObject
Dim oSys As ISAPSystem
Dim oConn As Object
Dim Count As Integer

' Log onto R/3 using the function shown in the example
'           for the Working Online topic
Set oConn = R3Logon()

If Not oConn Is Nothing Then
    oRep.online (oConn)
'   Connect offline
'     This creates the local database file
'     if it does not exist
    oRep.offline ("c:\sapreps.mdb")
    Set oSys = oRep.CreateSAPSystem(oConn.destination, _
               oConn.HostName, oConn.System, oConn.systemnumber, _
               oConn.applicationserver, oConn.saprelease)
    'Get the list of all the Business objects
    Set oBusObjs = oRep.BusinessObjects
    For Count = 1 To oBusObjs.Count
        Set oBusObj = oBusObjs.Item(Count)
        ' Your code for using the business object, for example:
        'Msgbox oBusObj.name
        'To save the current Business Object object,
        '     and all of its child objects
        '     such as methods, parameters etc:
        oRep.writeBOTree oSys, oBusObj
        'To save only the current business object
        '  without its child objects:
        'oRep.writeBO oSys, oBusObj
    Next
```

```
End If
```

See Also

Working Online [Page 14].

You may need to refresh the metadata in the local repository [Page 27] occasionally, if the metadata the R/3 database changes after you have copied it.

Deleting Objects from the Local Repository

Use

You can delete all the objects originating from a single R/3 system from the local repository.

You cannot delete individual objects from the local repository.

Procedure

1. Connect to the local repository with the Offline method [Page 46], specifying the local repository database from which you wish to delete objects.
2. (Optional) Get a list of the SAP Systems in the local repository, by calling SAPSystems. Identify the SAP system to be deleted.
3. Use the DeleteSAPSystem method [Page 43] to delete all of the objects belonging to the ISAPSystem object you specify.

Result

The objects are deleted from the Microsoft Access database file (MDB).

The size of the MDB file does not change after deleting objects. If you delete objects often, you may want to release the space on your hard disk by using the Microsoft Acces command: *Tools→Compact Database*.

Refreshing Local Repository Data

Use

If your program or users read data from the local repository, and if the business objects and RFC functions metadata in your R/3 system changes, you should update the local repository with the new version of the metadata from R/3.

The frequency of updating the local repository depends on how often the metadata in your R/3 system changes.

Procedure

To refresh the local copy of the metadata of a particular R/3 system:

1. Delete the data belonging to that R/3 system from the local repository database [Page 26].
2. Copy the desired metadata from the R/3 system to the new local repository [Page 22].

Since you can create and work with more than one local repository database, as an alternative, you can first copy the desired metadata into a new local repository database, and only then delete the old copy of the data from the existing local repository.

1. Create a new local repository database by using the Offline method [Page 46], and specifying a new database name.
2. Copy the desired data from R/3 to the new local repository [Page 22].
3. Delete the data from the existing local repository database [Page 26].

Do not simply re-write the same objects into the local repository without deleting the old data first.

If an object you are writing into the local repository already exists in the local repository, the Repository Services does not write it again. If you are trying to write a newer version of the objects, which you have just copied from a live R/3 system, then the object is not updated in the local repository.

Component Interface Reference

How to Use this Documentation

This reference documentation describes the various interfaces to the objects in the Repository Services hierarchy.

The properties and methods of each of the interfaces is listed in the interface description. A separate topic describes each property or methods in more details.

The properties and methods that are unique to each interface appears under that interface. Properties and methods that are common to multiple interfaces are grouped together under the Common Properties and Common Methods titles, respectively. To see the full description of a property or method, follow the link from the list in the interface description.

Syntax Conventions

The format used for describing the syntax of the various properties in this reference documentation is as follows:

```
PropertyName : PropertyType = InitialValue
```

For example, the following is the syntax for the *Count* property:

```
Count : Integer = 0
```

The format used for describing method syntax is as follows:

```
MethodName () : MethodType
```

For example, the following is the syntax for the *Fields* method:

```
Fields () : IFields
```

When a property or a method has one or more parameters, the parameters and their type is included in the syntax. For example, the two parameters of the WriteAppHierarchy method and their type are described in the following syntax:

```
WriteAppHierarchy(aSAPSystem : ISAPSystem, aApplicationHierarchy :
IApplicationHierarchy) : Boolean
```

IRepositoryServices

Purpose

An Interface for a repository services object. It retrieves the metadata of R/3 objects and manages the connection to R/3 system and the local repository. This is the entry point of the Repository Services component.

Properties

Connection	Sets [Page 34] or gets [Page 33] connection to an R/3 System
IsOffline [Page 35]	Finds out if a user connects to the local repository
IsOnline [Page 36]	Finds out if the user is connected to the R/3 system
IsR3Connected [Page 37]	Finds out if a user's attempt to connect to R/3 System has worked

Methods

ApplicationHierarchies [Page 39]	Gets an application hierarchy collection containing all application hierarchies
BusinessObjects [Page 40]	Gets a business object collection containing one or more business objects as specified by the search criteria
CompareSAPSystems [Page 41]	Compares the system information between the current R/3 System and other R/3 System
CreateSAPSystem [Page 42]	Create a new ISAPSystem object in the local repository whose properties are the parameters you specify
DeleteSAPSystem [Page 43]	Deletes all the R/3 objects associated with the R/3 system you specify from the local repository
FunctionGroups [Page 44]	Gets one or more function groups as specified by the search criteria
Functions [Page 45]	Gets a function collection containing one or more functions as specified by the search criteria
Offline [Page 46]	Sets the connection to the local repository
Online [Page 47]	Sets the connection to a R/3 System
SAPSystems [Page 48]	Gets a list of R/3 Systems, which are stored in the local repository, to which a user has the authorization to log on
SetSAPSystem [Page 49]	Connects to one R/3 System in the local repository
WriteAppHierarchies [Page 50]	Downloads an application hierarchies collection into the local repository
WriteAppHierarchy [Page 51]	Downloads an application hierarchy object into the local repository
WriteBO [Page 53]	Downloads a business object into the local repository

WriteBOTree [Page 54]	Downloads a business object and all of its children objects into the local repository
WriteFunctionGroup [Page 56]	Downloads a function group object into the local repository
WriteFunctionGroups [Page 58]	Downloads a function group collection into the local repository
WriteRFC [Page 59]	Downloads a function object into the local repository

Unique Properties

Connection (Get)

Purpose

Returns the Connection object associated with Connection property of the Repository Services object.

This is for online use only.

Syntax

```
Connection : Object = Null
```

Parameters

None.

Return Value

Returns a connection object, or null if it fails.

See Also

Online [Page 47], Connection (Set) [Page 34], Working online [Page 14]

Connection (Set)

Purpose

Assigns the Connection object from the SAP Automation DCOM Connector Logon Component (containing all the necessary R/3 connection information) to the IRepositoryServices object.

This is for online use only.

Syntax

```
Connection(aConnection : Object) : HRESULT
```

Parameters

aConnection	A Connection object from the SAP logon control

Return Value

Returns HRESULT.

Comments

Using the connection property to set the connection object is the same as using the Online method [Page 47] of IRepositoryServices.

See Also

Online [Page 47], Connection (Get) [Page 33], Working online [Page 14]

IsOffline

Purpose

Finds out if a user connects to the local repository.

Syntax

```
IsOffline : Boolean = False
```

Parameters

None.

Return Value

Returns true, if the user connects to the local repository, and false if the user does not connect.

IsOnline

Purpose

Finds out if the user is connected to the R/3 system.

Syntax

```
IsOnline : Boolean = False
```

Parameters

None

Return Value

Returns true if the user is connected to the R/3 system, and false if not.

IsR3Connected

Purpose

Indicates whether the user is in online mode.

Syntax

```
IsR3Connected : Boolean = False
```

Parameters

None.

Comments

You can obtain the same information with the IsOnline property [Page 36]. The IsR3Connected is kept for compatibility with previous versions of the Repository Services component. In previous versions of Repository Services this property indicated if a user's attempt to connect to the R/3 system has succeeded.

Return Value

Returns true, if the user connected to the R/3 system, and false if the user does not connect.

Unique Methods

ApplicationHierarchies

Purpose

Gets all application hierarchies of a particular R/3 system.

- When working online, objects are always retrieved directly from R/3 system.
- When working offline, objects are retrieved from the local repository.

Syntax

```
ApplicationHierarchies () : IApplicationHierarchies
```

Parameters

None.

Return Value

Returns an application hierarchy object collection containing all the application hierarchy objects, or null, if failure.

BusinessObjects

Purpose

Gets one or more business objects as specified by the search criteria.

- When working online, business objects are always retrieved directly from R/3 system.
- When working offline, business objects are retrieved from the local repository.

Syntax

```
BusinessObjects (aSearchCriterion : String) : IBusinessObjects
```

Parameters

aSearchCriterion	A search criterion to retrieve Business objects

If the parameter is null, then all business objects are retrieved. If it is not null, then the requested business object is retrieved, based on the search criterion.

Return Value

Returns a business object collection containing all the business objects that satisfy the selection criteria, or null, if failure.

CompareSAPSystems

Purpose

Compares the system information between the current SAP R/3 system and another SAP R/3 system.

Syntax

```
CompareSAPSystems (aSAPSystem : ISAPSystem) : Boolean
```

Parameters

aSAPSystem	a ISAPSystem object

Return Value

Returns true, if the two SAP R/3 systems are the same, and false if they are different.

CreateSAPSystem

Purpose

Create a new ISAPSystem object whose properties are the parameters you specify.

You can use this ISAPSystem object to create objects in the local repository that belong to the R/3 system identified by the ISAPSystem.

If the ISAPSystem already exists, this method returns a pointer to it.

Syntax

```
CreateSAPSystem (aDestination : String, aHostName : String, aSystem
:String, aSystemNumber : String, aApplicationServer : String,
aSAPRelease : String) : ISAPSystem
```

Parameters

aDestination	the SAP R/3 system name
aHostName	the hostname of the SAP R/3 system
aSystem	SAP R/3 system description
aSystemNumber	the system number of the SAP R/3 system
aApplicationServer	the SAP R/3 application server name
aSAPRelease	the SAP R/3 system release

Return Value

Returns an ISAPSystem object, or null if failure.

DeleteSAPSystem

Purpose

Deletes all the R/3 objects associated with the R/3 system you specify from the local repository.

Syntax

```
DeleteSAPSystem (aSAPSystem : ISAPSystem) : Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the objects belongs

Return Value

Returns True if the deletion succeeds, and False if it fails.

See Also

CreateSAPSystem [Page 42]

FunctionGroups

Purpose

Gets one or more function groups as specified by the search criteria.

- When working online, function group objects are always retrieved directly from R/3 System.
- When working offline, the function group objects are retrieved from the local repository.

Syntax

```
FunctionGroups (aSearchCriterion : String) : IFunctionGroups
```

Parameters

aSearchCriterion	Search criterion to retrieve the function groups

If the parameter is null, then all function groups are retrieved. If it is not null, then the requested function groups based upon the search criterion are retrieved. The wildcard expression can be used in the aSearchCriterion parameter.

Return Value

Returns a function group object collection containing all the function group objects that satisfy the selection criteria, or null, if failure.

Functions

Purpose

Gets one or more functions as specified by the search criteria.

- When working online, function objects are always retrieved directly from R/3 System.
- When working offline, the function objects are retrieved from the local repostiroy.

Syntax

```
Functions (aSearchCriterion : String) : IFunctions
```

Parameters

aSearchCriterion	A search criterion to retrieve the functions

If the parameter is null, then all functions are retrieved. If it is not null, then the requested functions are retrieved based upon the search criterion. A wildcard expression can be used in the *aSearchCriterion* parameter.

Return Value

Returns a function object collection containing all the function objects that satisfy the selection criteria, or null, if failure.

Offline

Purpose

Sets a connection to the local repository [Page 18].

Note that if you are connected to both the local repository and to a live R/3 database, the live R/3 connection takes precedence. That means that if you use a method for reading data, the Repository Services reads that data from the R/3 database, not from the local repository.

Syntax

```
Offline (aDatabase : String) : Boolean
```

Parameters

aDatabase	Full path of the local database location and name

Result

- If the local repository database exists, this method opens it.
- If the local repository database file does not exist, the method first creates it, and then opens it.
- If the local database file is invalid, meaning that it does not follow the correct structure for the Repository Services (for example, if it was created manually), then the method fails.

Return Value

Returns true if the local database has been opened successfully, or false if failure.

Online

Purpose

Assigns the SAP Automation DCOM Connector Logon Component's Connection object to the Connection property of the IRepositoryServices you are working with.

Syntax

```
Online (aConnection : Object) : Boolean
```

Parameters

aConnection	A connection object of the SAP logon control

Comments

Using the Online method is the same as using the Connection property [Page 34] of IRepositoryServices.

Note that if you are connected to both the local repository [Page 18] and to a live R/3 database, the live R/3 connection takes precedence. That means that if you use a method for reading data, the Repository Services reads that data from the R/3 database, not from the local repository.

Return Value

Returns true if successful, or false if failure.

See Also

Working Online [Page 14]

SAPSystems

Purpose

Gets a list of SAP R/3 Systems:

- In online mode, this method returns a single ISAPSystem object, representing the SAP R/3 system to which the user is currently logged on.
- In offline mode, this method gets the list of ISAPSystem objects stored in the local repository

Syntax

```
SAPSystems () : ISAPSystems
```

Parameters

None.

Return Value

Returns a collection of SAP R/3 Systems or null, if failure.

SetSAPSystem

Purpose

Specifies the SAP system whose objects you are going to access when working with the local repository.

Syntax

```
SetSAPSystem (aSAPSystem : ISAPSystem) : Boolean
```

Parameters

aSAPSystem	An *ISAPSystem* object

Return Value

Return True if successful, or False if failure.

WriteAppHierarchies

Purpose

Downloads an application hierarchies collection into the local repository.

Syntax

```
WriteAppHierarchies(aSAPSystem : ISAPSystem, aApplicationHierarchies :
IApplicationHierarchies) : Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the application hierarchies collection belongs
aApplicationHierarchies	The IApplicationHierarchies object that will be stored in the local repository

Return Value

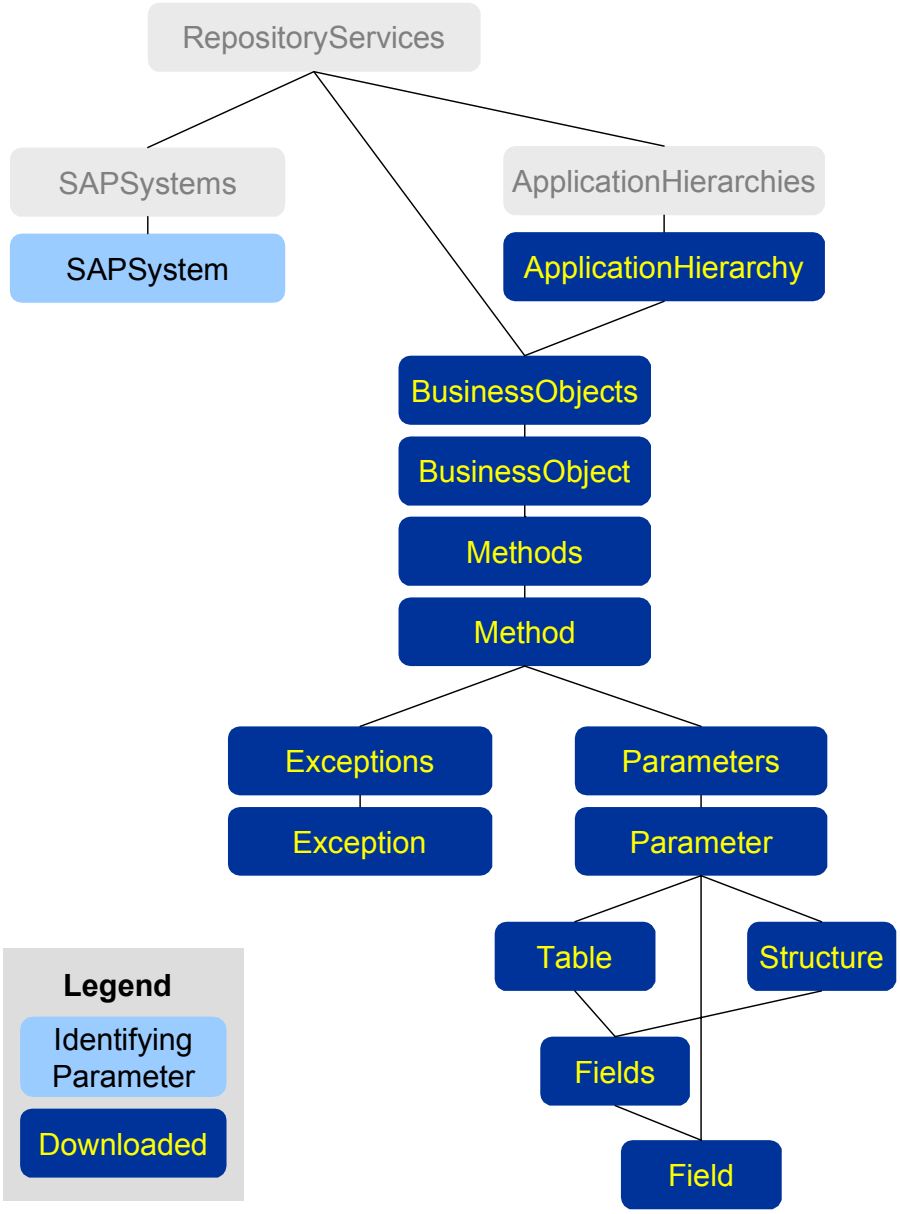
Return True if successful, or False if failure.

WriteAppHierarchy

Purpose

Downloads an application hierarchy and all of its children objects into the local repository.

The following diagram shows the objects that are downloaded into the local repository.



WriteAppHierarchy

Syntax

```
WriteAppHierarchy(aSAPSystem : ISAPSystem, aApplicationHierarchy :
IApplicationHierarchy) : Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the application hierarchy object belongs
aApplicationHierarchy	The IApplicationHierarchy object that will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

WriteAppHierarchies [Page 50]

WriteBO

Purpose

Downloads a business object into the local repository.

Syntax

```
WriteBO(aSAPSystem : ISAPSystem, aBusinessObject : IBusinessObject) : Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the business object belongs
aBusinessObject	The IBusinessObject object that will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

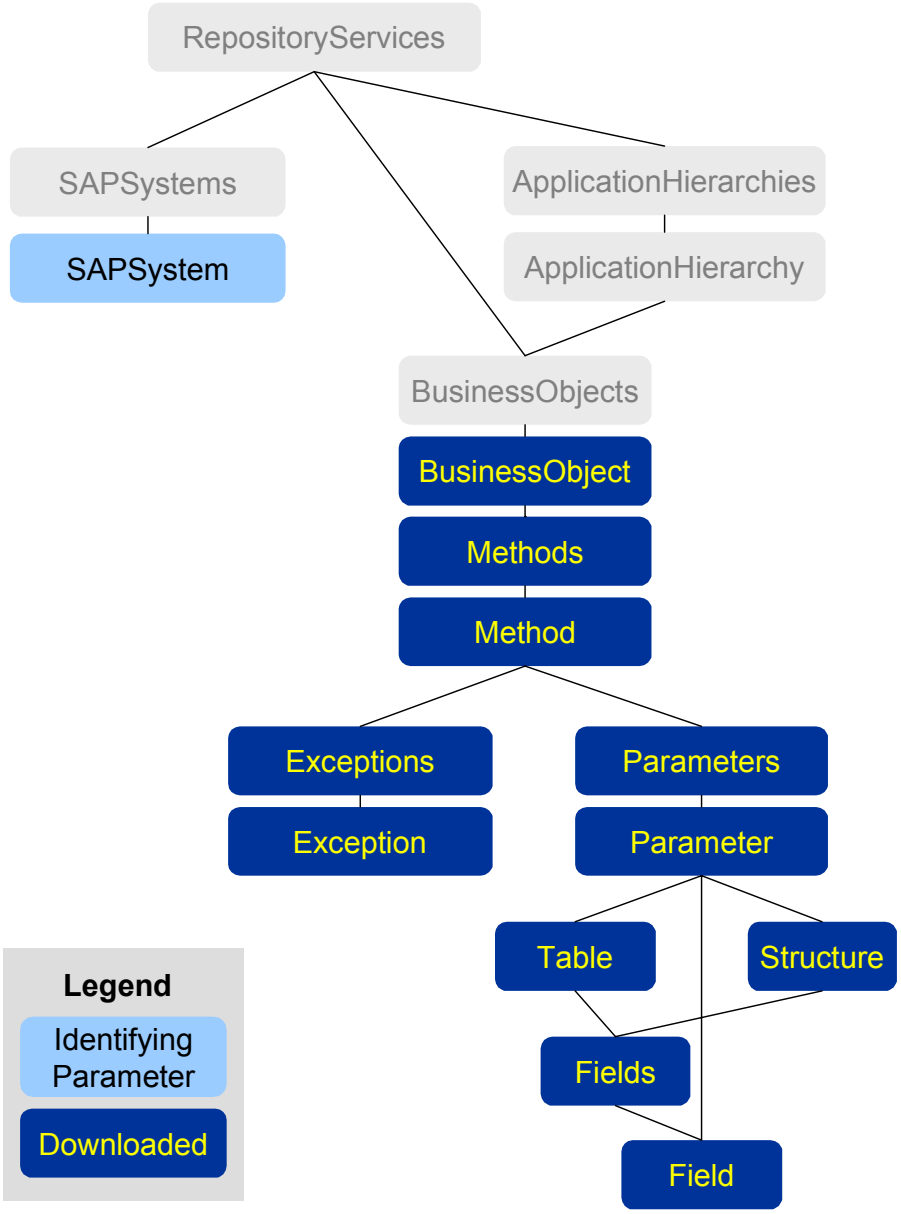
WriteBOTree [Page 54]

WriteBOTree

Purpose

Downloads a business object and all of its children objects into the local repository.

The following diagram shows the objects that are downloaded as part of a business object's tree.



Syntax

```
WriteBOTree(aSAPSystem : ISAPSystem, aBusinessObject : IBusinessObject) :
Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the business object tree belongs
aBusinessObject	The IBusinessObject object whose tree will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

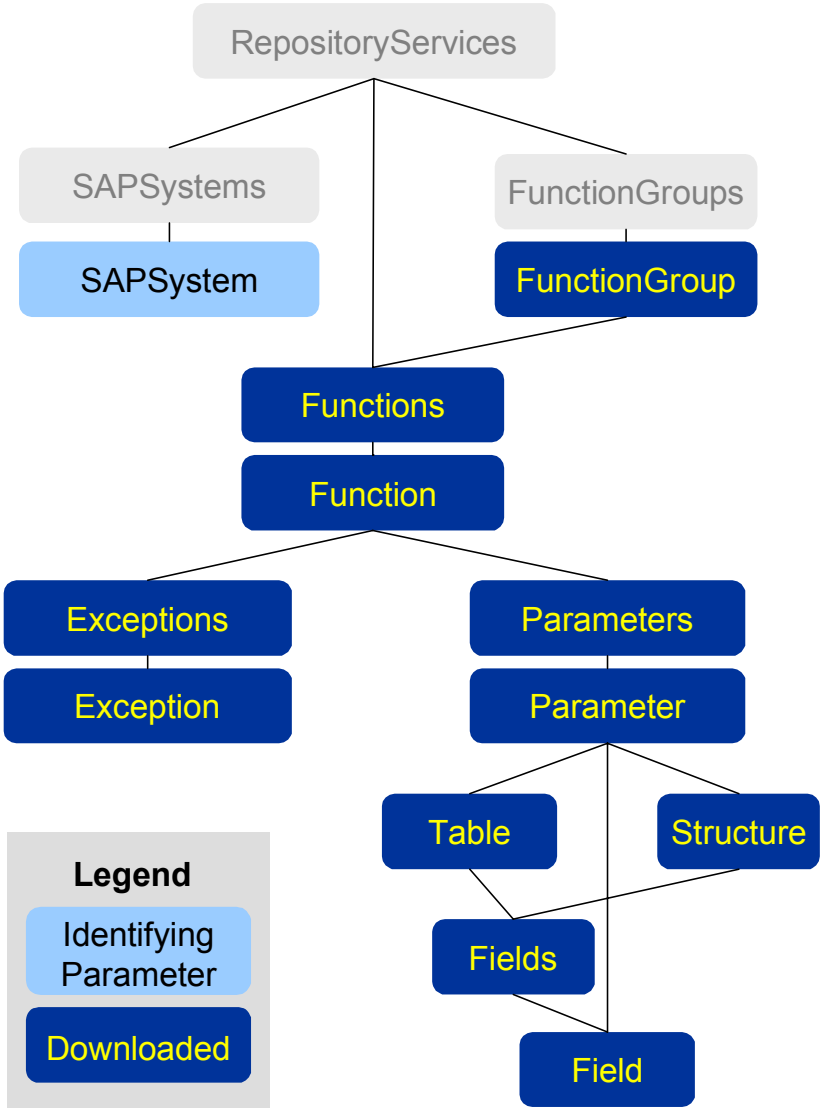
WriteBO [Page 53]

WriteFunctionGroup

Purpose

Downloads a function group object and all of its children objects into the local repository.

The following diagram shows the objects that are downloaded into the local repository.



Syntax

```
WriteFunctionGroups(aSAPSystem : ISAPSystem, aFunctionGroup : IFunctionGroup)
: Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the function group belongs
aFunctionGroup	The IFunctionGroup object that will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

WriteFunctionGroups [Page 58], WriteRFC [Page 59]

WriteFunctionGroups

Purpose

Downloads a function group collection into the local repository.

Syntax

```
WriteFunctionGroups(aSAPSystem : ISAPSystem, aFunctionGroups :
IFunctionGroups) : Boolean
```

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the business object belongs
aFunctionGroups	The IFunctionGroups object that will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

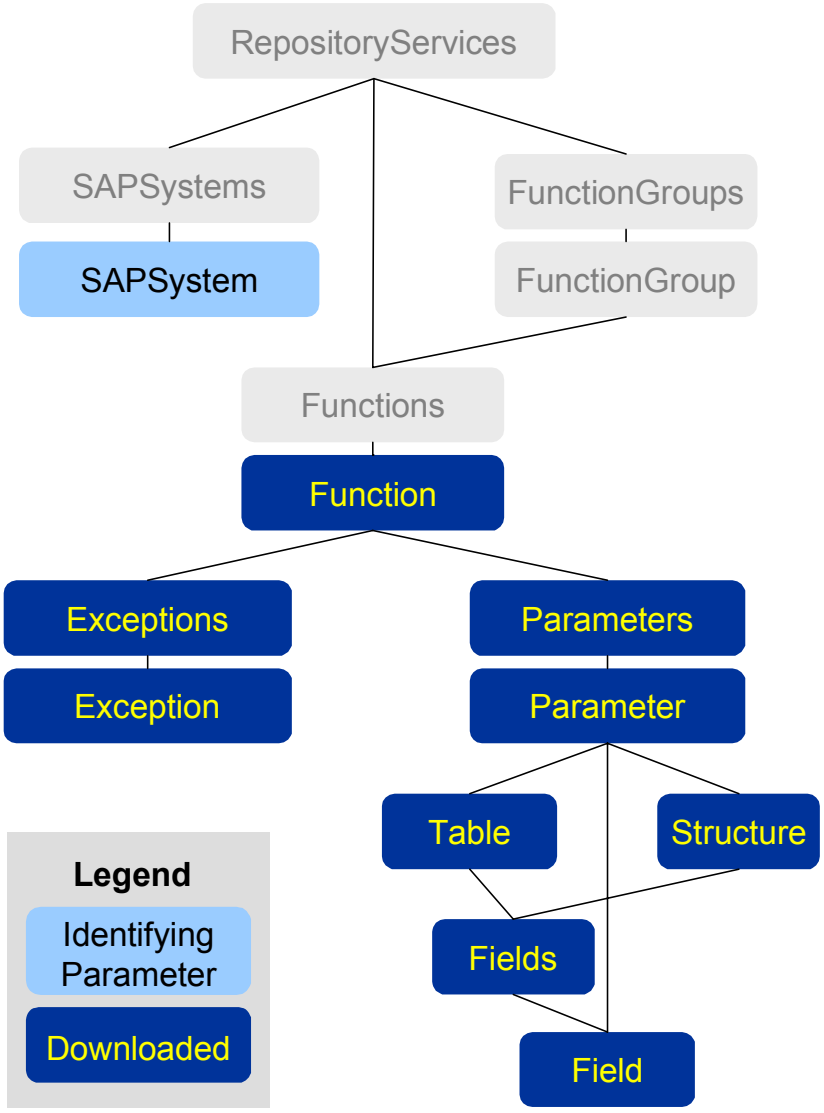
WriteFunctionGroup [Page 56], WriteRFC [Page 59]

WriteRFC

Purpose

Downloads a function object and all of its children into the local repository.

The following diagram shows the objects that are downloaded into the local repository.



Syntax

```
WriteRFC(aSAPSystem : ISAPSystem, aFunction : Ifunction) : Boolean
```

WriteRFC

Parameters

aSAPSystem	An ISAPSystem object, identifying the R/3 system to which the function object belongs
aFunction	The IFunction object that will be stored in the local repository

Return Value

Return True if successful, or False if failure.

See Also

WriteFunctionGroup [Page 56], WriteFunctionGroups [Page 58]

ISAPSystems

Purpose

An Interface for a SAP R/3 system collection that contains the SAP R/3 system objects.

Properties

Count [Page 152]	Gets the number of items in a SAP R/3 system collection
Item [Page 157]	Given an index or a name, returns an item in the collection
ItemByIndex [Page 158]	Given an index, returns an item in the collection
ItemByName [Page 159]	Given a name, returns an item in the collection
Parent [Page 162]	Gets the parent of a SAP R/3 system collection
ParentType [Page 163]	Gets the parent type of a SAP R/3 system collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the ISAPSystems' hierarchy

ISAPSystem

Purpose

An Interface for an SAP R/3 system that contains the metadata of an R/3 System.

Properties

ApplicationServer [Page 64]	Gets the SAP R/3 application server name
HostName [Page 65]	Gets the hostname of the SAP R/3 system
Name [Page 160]	Gets the SAP R/3 system name
Parent [Page 162]	Gets the parent of the SAP R/3 system object
ParentType [Page 163]	Gets the parent type of a SAP R/3 system object
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the ISAPSystem's hierarchy
SAPRelease [Page 66]	Gets the SAP R/3 system release
System [Page 67]	Gets SAP R/3 system description
SystemNumber [Page 68]	Gets the system number of the SAP R/3 system

Unique Properties

ApplicationServer

Purpose

Gets the SAP R/3 application server name.

Syntax

```
ApplicationServer : String = Null
```

Parameters

None.

Return Value

Returns the application server name, or null if failure.

HostName

Purpose

Gets the hostname of the SAP R/3 system.

Syntax

```
HostName : String = Null
```

Parameters

None.

Return Value

Returns the host name, or null if failure.

SAPRelease

Purpose

Gets the SAP R/3 system release.

Syntax

```
SAPRelease : String = Null
```

Parameters

None.

Return Value

Returns the SAP R/3 system release, or null if failure.

System

Purpose

Gets an SAP R/3 system description.

Syntax

```
System : String = Null
```

Parameters

None.

Return Value

Returns the SAP R/3 system description, or null if failure.

SystemNumber

Purpose

Gets the system number of the SAP R/3 system.

Syntax

```
SystemNumber : Long = 0
```

Parameters

None.

Return Value

Returns the system number.

IApplicationHierarchies

Purpose

An Interface for an application hierarchy collection that contains application hierarchy objects.

Properties

Count [Page 152]	Gets the number of items in an application hierarchy collection
Item [Page 157]	Given an index or a name, return an item in the collection
ItemByIndex [Page 158]	Given an index, return an item in the collection
ItemByName [Page 159]	Given a name, return an item in the collection
Parent [Page 162]	Gets the parent of an application hierarchy collection
ParentType [Page 163]	Gets the parent type of an application hierarchy collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IApplicationHierarchies' hierarchy

IApplicationHierarchy

Purpose

An Interface for an application hierarchy that contains metadata of an application hierarchy.

Properties

ApplicationArea [Page 151]	Gets the application area of an application hierarchy
ApplicationAreaDescription [Page 72]	Gets the application description of an application hierarchy
Parent [Page 162]	Gets the parent of an application hierarchy
ParentType [Page 163]	Gets the parent type of an application hierarchy
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IApplicationHierarchy's hierarchy

Methods

BusinessObjects [Page 74]	Gets a business object collection containing the business objects of an application hierarchy

Unique Properties

ApplicationAreaDescription

Purpose

Gets the application description of an application hierarchy.

Syntax

```
ApplicationAreaDescription : String = Null
```

Parameters

None.

Return Value

Returns the application description, or null if failure.

Unique Methods

BusinessObjects

Purpose

Gets the business objects of an application hierarchy.

Syntax

```
BusinessObjects () : IBusinessObjects
```

Parameters

None.

Return Value

Returns a business object collection, or null if failure.

IBusinessObjects

Purpose

An Interface for a business object collection that contains business objects.

Properties

Count [Page 152]	Gets the number of items in a business object collection
Item [Page 157]	Given an index or a name, return an item in the collection
ItemByIndex [Page 158]	Given an index, return an item in the collection
ItemByName [Page 159]	Given a name, return an item in the collection
Parent [Page 162]	Gets the parent of a business object collection
ParentType [Page 163]	Gets the parent type of a business object collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IbusinessObjects' hierarchy

IBusinessObject

Purpose

An Interface for a business object that contains metadata of a business object.

Properties

ApplicationArea [Page 151]	Gets the application area of a business object
Descirption [Page 153]	Gets the description of a business object
DevelopmentClass [Page 78]	Gets the development class a business object
Documentation [Page 155]	Gets the documentation of a business object
InternalName [Page 156]	Gets the object type of a business object
LastChangeDate [Page 79]	Gets the date on which the business object is changed
LastChangeRelease [Page 80]	Gets the R/3 release of the last change on a business object
LastChangeTime [Page 81]	Gets the time at which a business object is changed
Name [Page 160]	Gets the name of a business object
Obsolete [Page 161]	Gets release as of which the object type has been marked as obsolete
Parent [Page 162]	Gets the parent of a business object
ParentType [Page 163]	Gets the parent type of a business object
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IBusinessObject's hierarchy

Methods

KeyFields [Page 83]	Gets the key fields of a business object
Methods [Page 84]	Gets the methods of a business object

Unique Properties

DevelopmentClass

Purpose

Gets the development class of a business object.

Syntax

```
DevelopmentClass : String = Null
```

Parameters

None.

Return Value

Returns the development class or null, if failure.

LastChangeDate

Purpose

Gets the date on which the business object has last changed.

Syntax

```
LastChangeDate : String = Null
```

Parameters

None.

Return Value

Returns the changed date or null, if failure.

LastChangeRelease

Purpose

Gets the R/3 release of the last change to a business object.

Syntax

```
LastChangeRelease : String = Null
```

Parameters

None.

Return Value

Returns the R/3 release of the last change or null, if failure.

LastChangeTime

Purpose

Gets the time at which the business object has last changed.

Syntax

```
LastChangeTime : String = Null
```

Parameters

None.

Return Value

Returns the changed time or null, if failure.

Unique Methods

KeyFields

Purpose

Gets the key fields of a business object.

Syntax

```
KeyFields () : IFields
```

Parameters

None.

Return Value

Returns a collection of fields or null, if not found.

Methods

Purpose

Gets the methods of a business object.

Syntax

```
Methods () : IMethods
```

Parameters

None.

Return Value

Returns a collection of methods or null, if not found.

IMethods

Purpose

An Interface for a method collection that contains method objects.

Properties

Count [Page 152]	Gets the number of items in a method collection
Item [Page 157]	Given an index or a name, returns an item in the collection
ItemByIndex [Page 158]	Given an index, returns an item in the collection
ItemByName [Page 159]	Given a name, returns an item in the collection
Parent [Page 162]	Gets the parent of a method collection
ParentType [Page 163]	Gets the parent type of a method collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IMethods' hierarchy

IMethod

Purpose

An Interface for a method that contains metadata of a method.

Properties

Description [Page 153]	Gets the description of a method
Documentation [Page 155]	Gets the documentation of a method
InternalName [Page 156]	Gets the object type of a method
IsClassMethod [Page 88]	Determines if the method is a class method
IsFactoryMethod [Page 89]	Decides if the method is a factory method
IsFrozen [Page 90]	Determines if the DDIC structure is frozen
IsModelOnly [Page 91]	Determines if the method is modeled only
IsSynchronousMethod [Page 92]	Decides if the method is a synchronous method
MethodType [Page 93]	Gets the type of a method
Name [Page 160]	Gets the name of a method
Obsolete [Page 161]	Gets the release as of which the object type has been marked as obsolete
Parent [Page 162]	Gets the parent of a method
ParentType [Page 163]	Gets the parent type of a method
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IMethod's hierarchy

Methods

Exceptions [Page 167]	Gets the exceptions of a method
Parameters [Page 169]	Gets the parameters of a method

Unique Properties

IsClassMethod

Purpose

Determines if the method is a class method.

Syntax

```
IsClassMethod : Boolean = False
```

Parameters

None.

Return Value

Returns true if it is a class method, and false if it is not.

Description

If a method is neither a class nor a factory, then it is an instance.

IsFactoryMethod

Purpose

Determines if the method is a factory method.

Syntax

```
IsFactoryMethod : Boolean = False
```

Parameters

None.

Return Value

Returns true if it is a factory method, and false if it is not.

Remarks

If a method is neither a class nor a factory, then it is an instance.

IsFrozen

Purpose

Determines if the DDIC structure is frozen.

Syntax

```
IsFrozen : Boolean = False
```

Parameters

None.

Return Value

Returns true if it is, and false if it is not.

Remarks

For 4.0A release and higher

.

IsModelOnly

Purpose

Determines if the method is modeled only.

Syntax

```
IsModelOnly : Boolean = False
```

Parameters

None.

Return Value

Returns true if it is modeled, and false if it is not.

IsSynchronousMethod

Purpose

Decides if the method is a synchronous method.

Syntax

```
IsSynchronousMethod : Boolean = False
```

Parameters

None.

Return Value

Returns true if the method is synchronous, and false if it is not.

MethodType

Purpose

Gets the type of a method.

Syntax

```
MethodType : BO_METHOD_TYPE = UNKNOWN
```

Parameters

None.

Return Value

See the BO_METHOD_TYPE [Page 171] topic for a list of its values.

Remarks

If a method is neither a class nor a factory, then it is an instance.

IFunctionGroups

Purpose

An Interface, which is collection that contains function group objects.

Properties

Count [Page 152]	Gets the number of items in a function group collection
Item [Page 157]	Given an index or a name, returns an item in the collection
ItemByIndex [Page 158]	Given an index, returns an item in the collection
ItemByName [Page 159]	Given a name, returns an item in the collection
Parent [Page 162]	Gets the parent of a function group collection
ParentType [Page 163]	Gets the parent type of a function group collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IFunctionGroups' hierarchy

IFunctionGroup

Purpose

An Interface for a function group and contains the metadata of a function group.

Properties

Description [Page 153]	Gets the description of a function group
Documentation [Page 155]	Gets the documentation of a function group
Name [Page 160]	Gets the name of a function group
Parent [Page 162]	Gets the parent of a function group
ParentType [Page 163]	Gets the parent type of a function group
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IFunctionGroup's hierarchy

Methods

Functions [Page 97]	Gets the functions of a function group

Unique Methods

Functions

Purpose

Gets the functions of a function group.

Syntax

```
Functions () : IFunctions
```

Parameters

None.

Return Value

Returns a function collection, or null if failure.

IFunctions

Purpose

An Interface for a function collection that contains function objects.

Properties

Count [Page 152]	Gets the number of items in a function collection
Item [Page 157]	Given an index or a name, returns an item in the collection
ItemByIndex [Page 158]	Given an index, returns an item in the collection
ItemByName [Page 159]	Given a name, returns an item in the collection
Parent [Page 162]	Gets the parent of a function collection
ParentType [Page 163]	Gets the parent type of a function collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IFunctions' hierarchy

IFunction

Purpose

An Interface for a function that contains the metadata of a function.

Properties

Documentation [Page 155]	Gets the documentation of a function
GroupName [Page 101]	Gets the group name of a function
Name [Page 160]	Gets the name of a function
Parent [Page 162]	Gets the parent of a function
ParentType [Page 163]	Gets the parent type of a function
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IFunction's hierarchy

Methods

Exceptions [Page 167]	Gets all exception messages of a function
Parameters [Page 169]	Gets the parameters of a function

Unique Properties

GroupName

Purpose

Gets the group name of a function.

Syntax

```
GroupName : String = Null
```

Parameters

None.

Return Value

Returns the function group name, or null if failure.

IExceptions

Purpose

An Interface for an exception collection that contains exception objects.

Properties

Count [Page 152]	Gets the number of items in an exception collection
Item [Page 157]	Given an index or a name, return an item in the collection
ItemByIndex [Page 158]	Given an index, return an item in the collection
ItemByName [Page 159]	Given a name, return an item in the collection
Parent [Page 162]	Gets the parent of an exception collection
ParentType [Page 163]	Gets the parent type an exception collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IExceptions' hierarchy

IException

Purpose

An Interface for an exception that contains the metadata of an exception.

Properties

Description [Page 153]	Gets the description of an exception
ID [Page 105]	Gets the ID of an exception
Parent [Page 162]	Gets the parent of an exception
ParentType [Page 163]	Gets the parent type of an exception
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IException's hierarchy

Unique Properties

ID

Purpose

Gets the ID of an exception.

Syntax

```
ID : String = Null
```

Parameters

None.

Return Value

Returns the exception ID, or null if failure.

IParameters

Purpose

An Interface for a parameter collection that contains parameter objects.

Properties

Count [Page 152]	Gets the number of items in a parameter collection
Item [Page 157]	Given an index or a name, returns an item in the collection
ItemByIndex [Page 158]	Given an index, returns an item in the collection
ItemByName [Page 159]	Given a name, returns an item in the collection
Parent [Page 162]	Gets the parent of a parameter collection
ParentType [Page 163]	Gets the parent type of a parameter collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IParameters' hierarchy

IParameter

Purpose

An Interface for a parameter that contains the metadata of a parameter.

Properties

ABAPName [Page 110]	Gets the ABAP name of a parameter, which is not applicable to RFC
ABAPType [Page 150]	Gets the ABAP data type of a parameter
Default [Page 111]	Gets the default value of a parameter
Description [Page 153]	Gets the description of a parameter
DictionaryType [Page 154]	Gets the screen data type of a parameter for the Screen Painter tool
Documentation [Page 155]	Gets the documentation of a parameter
InternalName [Page 156]	Gets the reference object type of a parameter
IsExport [Page 113]	Determines if the parameter is an export parameter
IsField [Page 114]	Determines if the parameter is a field parameter
IsImport [Page 115]	Determines if the parameter is an import parameter
IsMandatory [Page 116]	Determines if the parameter is mandatory
IsStructure [Page 112]	Determines if the parameter is a structure parameter
IsTable [Page 117]	Determines if the parameter is a table parameter
Name [Page 160]	Gets the name of a parameter
Parent [Page 162]	Gets the parent of a parameter
ParentType [Page 163]	Gets the parent type of a parameter
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IParameter's hierarchy

Methods

Field [Page 119]	Gets the field of a parameter
Structure [Page 120]	Gets the structure of a parameter

IParameter

Table [Page 121]	Gets the table of a parameter

Unique Properties

ABAPName

Purpose

Gets the ABAP name of a parameter, which is not applicable to RFC.

Syntax

```
ABAPName : String = Null
```

Parameters

None.

Return Value

Returns one of the following values:

- The ABAP name, if the parameter is from BAPI.
- Null, if the parameter is from RFC.
- Null, if failure.

Default

Purpose

Gets the default value of a parameter.

Applies To

This property is only for a simple parameter (field) and does not apply to Table or Structure.

Syntax

```
Default : String = Null
```

Parameters

None.

Return Value

Returns the default value, or null if failure.

IsStructure

Purpose

Determines if the parameter is a structure parameter.

Syntax

```
IsStructure : Boolean = False
```

Parameters

None.

Return Value

Returns true if the parameter is a structure, and false if it is not.

IsExport

Purpose

Determines if the parameter is an export parameter.

Syntax

```
IsExport : Boolean = False
```

Parameters

None.

Return Value

Returns true, if the parameter is an export parameter, and false if it is not.

IsField

Purpose

Determines if the parameter is a field parameter.

Syntax

```
IsField : Boolean = false
```

Parameters

None.

Return Value

Returns true if the parameter is a field, and false if it is not.

IsImport

Purpose

Determines if the parameter is an import parameter.

Syntax

```
IsImport : Boolean = False
```

Parameters

None.

Return Value

Returns true, if the parameter is an import parameter, and false if it is not.

IsMandatory

Purpose

Determines if the parameter is mandatory.

Syntax

```
IsMandatory : Boolean = False
```

Parameters

None.

Return Value

Returns true if the parameter is mandatory, and false if it is not.

IsTable

Purpose

Determines if the parameter is a table parameter.

Syntax

```
IsTable : Boolean = False
```

Parameters

None.

Return Value

Returns true, if the parameter is a table, and false if it is not.

Unique Methods

Field

Purpose

Gets the field of a parameter.

Syntax

```
Field () : IField
```

Parameters

None.

Return Value

Returns the field, or null if failure.

Structure

Purpose

Gets the structure of a parameter.

Syntax

```
Structure () : IStructure
```

Parameters

None.

Return Value

Returns the structure, or null if failure.

Table

Purpose

Gets the table of a parameter.

Syntax

```
Table () : ITable
```

Parameters

None.

Return Value

Returns the table, or null if failure.

IStructure

Purpose

An Interface for a structure that contains the metadata of a structure.

Properties

Name [Page 160]	Gets the name of a structure
Parent [Page 162]	Gets the parent of a structure
ParentType [Page 163]	Gets the parent type of a structure
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IStructure's hierarchy
RowLength [Page 165]	Gets the length of a structure

Methods

Fields [Page 168]	Gets the fields of a structure

ITable

Purpose

An Interface for a table that contains the metadata of a table.

Properties

Name [Page 160]	Gets the name of a table
Parent [Page 162]	Gets the parent of a table
ParentType [Page 163]	Gets the parent type of a table
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the ITable's hierarchy
RowLength [Page 165]	Gets the row length of a table

Methods

Fields [Page 168]	Gets the fields in a table

IFields

Purpose

An Interface for a field collection that contains field objects.

Properties

Count [Page 152]	Gets the number of items in a field collection
Item [Page 157]	Given an index or a name, return an item in the collection
ItemByIndex [Page 158]	Given an index, return an item in the collection
ItemByName [Page 159]	Given a name, return an item in the collection
Parent [Page 162]	Gets the parent of a field collection
ParentType [Page 163]	Gets the parent type of a field collection
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IFields' hierarchy

IField

An Interface for a field that contains the metadata of a field.

Properties

ABAPType [Page 150]	Gets the ABAP data type of a field
CheckTable [Page 128]	Gets the check table of a field
DataElement [Page 129]	Gets the data element (semantic domain) of a field
Decimals [Page 130]	Gets the number of decimals of a field
Description [Page 153]	Gets the description of a field
DictionaryType [Page 154]	Gets the screen data type of a field for the Screen Painter tool
Domain [Page 131]	Gets the domain of a field
HasFixedValues [Page 132]	Determines if the field contains fixed values
HelpValues [Page 133]	Given an index, return an item in the help value collection
HelpValuesCount [Page 134]	Gets the number of help values of a field
InternalLength [Page 135]	Gets the internal length of a field in bytes
InternalName [Page 156]	Gets the type name of a field
IsCaseSensitive [Page 136]	Gets the case sensitivity status of a field
IsConvertionExit [Page 137]	Finds the field's conversion exit status
Length [Page 138]	Gets the width of a field
LongScreenText [Page 139]	Gets the long screen text of a field
MediumScreenText [Page 140]	Gets the medium screen text of a field
Name [Page 160]	Gets the name of a field
Offset [Page 141]	Gets the internal start address of a field
Parent [Page 162]	Gets the parent of a field
ParentType [Page 163]	Gets the parent type of a field
Position [Page 142]	Gets the position of a field in a table
ReferenceField [Page 143]	Gets the reference field for currency and quantity fields

IField

ReferenceTable [Page 144]	Gets the reference table of a field
ReportHeader [Page 145]	Gets the report header of a field
Root [Page 164]	Gets the *IRepositoryServices* object at the root of the IField's hierarchy
ShortScreenText [Page 146]	Gets the short screen text of a field
ValuesForField [Page 147]	Given an index, return an item in the values for field collection
ValuesForFieldCount [Page 148]	Gets the number of values of a field

Unique Properties

CheckTable

Purpose

Gets the check table of a field.

Syntax

```
CheckTable : String = Null
```

Parameters

None.

Return Value

Returns the field's check table, or null if failure.

DataElement

Purpose

Gets the data element (semantic domain) of a field.

Syntax

```
DataElement : String = Null
```

Parameters

None.

Return Value

Returns the field data element (semantic domain), or null if failure.

Decimals

Purpose

Gets the number of decimals of a field.

Syntax

```
Decimals : Long = 0
```

Parameters

None.

Return Value

Returns the number of decimals in the field.

Domain

Purpose

Gets the domain of a field.

Syntax

```
Domain : String = Null
```

Parameters

None.

Return Value

Returns the field's domain, or null if failure.

HasFixedValues

Purpose

Determines if the field contains fixed values.

Syntax

```
HasFixedValues : Boolean = False
```

Parameters

None.

Return Value

Returns true, if there are fixed values, and false if there are no fixed values.

HelpValues

Purpose

Given an index, returns an item in the help value collection.

Syntax

```
HelpValues(aIndex : Long) : String = Null
```

Parameters

aIndex	An index

Return Value

Returns the requested help value, or null if failure.

See also:

HelpValuesCount [Page 134]()

HelpValuesCount

Purpose

Gets the number of help values of a field.

Syntax

```
HelpValuesCount : Long = 0
```

Parameters

None.

Return Value

Returns the number of field help values.

InternalLength

Purpose

Gets the internal length of a field in bytes.

Syntax

```
InternalLength : Long = 0
```

Parameters

None.

Return Value

Returns the field's internal length.

IsCaseSensitive

Purpose

Gets the case sensitivity status of a field.

Syntax

```
IsCaseSensitive : Boolean = False
```

Parameters

None.

Return Value

Returns true, if the field is case sensitive and false, if it is not.

IsConversionExit

Purpose

Gets the field's conversion exit status.

Syntax

```
sConversionExit : Boolean = False
```

Parameters

None.

Return Value

Returns true, if there is a conversion exit, or false if there is no exit.

Length

Purpose

Gets the width of a field.

Syntax

```
Length : Long = 0
```

Parameters

None.

Return Value

Returns the field's width.

LongScreenText

Purpose

Gets the long screen text of a field.

Syntax

```
LongScreenText : String = Null
```

Parameters

None.

Return Value

Returns the long screen text, or null if failure.

MediumScreenText

Purpose

Gets the medium screen text of a field.

Syntax

```
MediumScreenText : String = Null
```

Parameters

None.

Return Value

Returns the medium screen text, or null if failure.

Offset

Purpose

Gets the internal start address of a field.

Syntax

```
Offset : Long = 0
```

Parameters

None.

Return Value

Returns the field's internal start address.

Position

Purpose

Gets the position of a field in a table.

Syntax

```
Position : Long = 0
```

Parameters

None.

Return Value

Returns the field's position in a table.

ReferenceField

Purpose

Gets the reference field for currency and quantity fields.

Syntax

```
ReferenceField : String = Null
```

Parameters

None.

Return Value

Returns the field's reference field, or null if failure.

ReferenceTable

Purpose

Gets the reference table of a field.

Syntax

```
ReferenceTable : String = Null
```

Parameters

None.

Return Value

Returns the field reference table, or null if failure.

ReportHeader

Purpose

Gets the report header of a field.

Syntax

```
ReportHeader : String = Null
```

Parameters

None.

Return Value

Returns the field report header, or null if failure.

ShortScreenText

Purpose

Gets the short screen text of a field.

Syntax

```
ShortScreenText : String = Null
```

Parameters

None.

Return Value

Returns the short screen text, or null if failure.

ValuesForField

Purpose

Given an index, returns an item in the values for field collection.

Syntax

```
ValuesForField(aIndex : Long) : String = Null
```

Parameters

aIndex	An index

Return Value

Returns the requested value for field, or null if failure.

See also:

ValuesForFieldCount [Page 148]()

ValuesForFieldCount

Purpose

Gets the number of values of a field.

Syntax

```
ValuesForFieldCount : Long = 0
```

Parameters

None.

Return Value

Returns the number of values for a field.

Common Properties

ABAPType

Purpose

Gets the ABAP data type of a field or a parameter.

Applies To

IField [Page 125], IParameter [Page 107]

Syntax

```
ABAPType : String = Null
```

Parameters

None.

Return Value

Returns the ABAP data type of the field or parameter, or null if failure.

ApplicationArea

Purpose

Gets the application area of an application hierarchy or a business object.

Applies To

IApplicationHierarchy [Page 70], IBusinessObject [Page 76]

Syntax

```
ApplicationArea : String = Null
```

Parameters

None.

Return Value

Returns the application area, or null if failure.

Count

Purpose

Gets the number of items in a collection of objects.

Applies To

IApplicationHierarchies [Page 69], IBusinessObjects [Page 75], IExceptions [Page 102], IFields [Page 124], IFunctions [Page 98], IFunctionGroups [Page 94], IMethods [Page 85], IParameters [Page 106], and ISAPSystems [Page 61] collections.

Syntax

```
Count : Integer = 0
```

Parameters

None.

Return Value

Returns the number of objects in the collection.

Description

Purpose

Gets the description of the object.

Applies To

IFunctionGroup [Page 95], IBusinessObject [Page 76], IMethod [Page 86], IParameter [Page 107], IException [Page 103], IField [Page 125]

Syntax

```
Description : String = Null
```

Parameters

None.

Return Value

Returns the object's description, or null if failure.

DictionaryType

Purpose

Gets the screen data type of the object for the Screen Painter tool.

Applies To

IField [Page 125], IParameter [Page 107].

Syntax

```
DictionaryType : String = Null
```

Parameters

None.

Return Value

Returns the field's data type, or null if failure.

Documentation

Purpose

Gets the documentation of the object.

Applies To

IBusinessObject [Page 76], IFunction [Page 99], IFunctionGroup [Page 95], IMethod [Page 86], IParameter [Page 107].

Syntax

```
Documentation : String = Null
```

Parameters

None.

Return Value

Returns the object's documentation, or null if failure.

InternalName

Purpose

Gets the object type of a business object, a method, a parameter, or a field.

Applies To

IBusinessObject [Page 76], IMethod [Page 86], IParameter [Page 107], IField [Page 125].

Syntax

```
InternalName : String = Null
```

Parameters

None.

Return Value

Returns the object type or null, if failure.

Item

Purpose

Given an index or a name, returns an item in the collection.

Applies To

IApplicationHierarchies [Page 69], IBusinessObjects [Page 75], IExceptions [Page 102], IFields [Page 124], IFunctions [Page 98], IFunctionGroups [Page 94], IMethods [Page 85], IParameters [Page 106], and ISAPSystems [Page 61] collections.

Syntax

```
Item (Index : Variant) : Object = Null
```

Parameters

Index	A name or an index

Return Value

Returns the requested object, or null if failure.

The type of the returned object (*Object* in the above syntax) depends on the collection. For an ISAPSystems collection for example, the returned object is an ISAPSystem object.

See Also

Count() [Page 152]

ItemByIndex

Purpose

Given an index, returns an item in the collection.

Applies To

IApplicationHierarchies [Page 69], IBusinessObjects [Page 75], IExceptions [Page 102], IFields [Page 124], IFunctions [Page 98], IFunctionGroups [Page 94], IMethods [Page 85], IParameters [Page 106], and ISAPSystems [Page 61] collections.

Syntax

```
ItemByIndex (aIndex : Integer) : Obeject = Null
```

Where *Object* is the item in the collection, for example, for an ISAPSystems collection the syntax is:

```
ItemByIndex (aIndex : Integer) : ISAPSystem = Null
```

Parameters

aIndex	An index

Return Value

Returns the requested object, or null if failure.

The type of the returned object (*Object* in the above syntax) depends on the collection. For an ISAPSystems collection for example, the returned object is an ISAPSystem object.

See also

Count() [Page 152]

ItemByName

Purpose

Given an object name, returns the item in the collection. For example, given a SAP R/3 system name, it returns an item in the ISAPSystems collection.

Applies To

IApplicationHierarchies [Page 69], IBusinessObjects [Page 75], IExceptions [Page 102], IFields [Page 124], IFunctions [Page 98], IFunctionGroups [Page 94], IMethods [Page 85], IParameters [Page 106], and ISAPSystems [Page 61] collections.

Syntax

```
ItemByName (aName : String) : Object = Null
```

Parameters

aName	Object name, for example, an SAP R/3 system name

Return Value

Returns the requested SAP R/3 system, or null if failure.

The type of the returned object (*Object* in the above syntax) depends on the collection. For an ISAPSystems collection for example, the returned object is an ISAPSystem object.

Name

Purpose

Gets the name of the object.

Applies To

IFunction [Page 99], IFunctionGroup [Page 95], IBusinessObject [Page 76], IMethod [Page 86], IParameter [Page 107], ISAPSystem [Page 62], IStructure [Page 122], ITable [Page 123], IField [Page 125].

Syntax

```
Name : String = Null
```

Parameters

None.

Return Value

Returns the object's name, or null if failure.

Obsolete

Purpose

Gets the release as of which the object type has been marked as obsolete.

Applies To

IBusinessObject [Page 76], IMethod [Page 86].

Syntax

```
Obsolete : String = Null
```

Parameters

None.

Return Value

Returns the release as of which the object type has been marked as obsolete, or null if failure.

Remarks

For Release 4.0A and higher.

Parent

Purpose

Gets the parent of the collection or object.

Prerequisites

Before using this property, use the ParentType [Page 163]() property to find out the parent type.

Applies To

All interfaces, except IRepositoryServices.

Syntax

```
Parent : Object = Null
```

Parameters

None.

Return Value

Returns the parent object of the object or collection, or null if the parent is not found.

Examples

For an ISAPSystems collection, this property returns one of the following values:

- An *IRepositoryService* object, if the parent is a repository services
- Null, if the parent is not found

For ISAPSystem object, this property returns one of the following values:

- An *ISAPSystems* object, if the parent is a SAP R/3 system collection
- Null, if the parent is not found

For an IFields collection, this property returns one of the following values:

- An *ITable* object, if the parent is a table
- An *ISturcture* object, if the parent is a structure
- An *IBusinessObject* object, if the parent is a business object
- Null, if the parent is not found

See Also

ParentType [Page 163]()

ParentType

Purpose

Gets the parent type of a collection or object. Use this property before using the Parent [Page 162]() property.

Applies To

All interfaces, except IRepositoryServices.

Syntax

```
ParentType : OBJECT_TYPE = CUNKNOWN
```

Parameters

None.

Return Value

See OBJECT_TYPE [Page 172] for the list of values that may be returned. The returned value may either be CUNKNOWN, if the parent is not found, or it is the type of the parent object.

For example, for a field collection, this property returns one of the following values:

- CTABLE, if the parent is a table.
- CSTRUCTURE, if the parent is a structure.
- CBUSINESSOBJECT, if the parent is a business object.
- CUNKNOWN, if the parent is not found.

Root

Purpose

Gets the *IRepositoryServices* object at the root of the object's hierarchy.

Applies To

All interfaces, except IRepositoryServices itself.

Syntax

```
Root : IRepositoryServices = Null
```

Parameters

None.

Return Value

Returns an *IRepositoryServices* object, or null if failure.

RowLength

Purpose

Gets the length of a structure or one row in a table.

Applies To

IStructure [Page 122], ITable [Page 123].

Syntax

```
RowLength : Long = 0
```

Parameters

None.

Return Value

Returns the length of the structure or the row in the table.

Common Methods

Exceptions

Purpose

Gets the exception messages of a function or a method.

Applies To

IFunction [Page 99], IMethod [Page 86].

Syntax

```
Exceptions () : IExceptions
```

Parameters

None.

Return Value

Returns a set of messages, or null if failure.

Fields

Purpose

Gets all of the fields in a table or a structure.

Applies To

ITable [Page 123], IStructure [Page 122].

Syntax

```
Fields () : IFields
```

Parameters

None.

Return Value

Returns a field collection, or null if failure.

Parameters

Purpose

Gets the parameters of a function or a method.

Applies To

IFunction [Page 99], IMethod [Page 86].

Syntax

```
Parameters () : IParameters
```

Parameters

None.

Return Value

Returns a parameter collection, or null if there is no parameter.

Common Returned Values

BO_METHOD_TYPE

The BO_METHOD_TYPE is a returned value of an enumeration type.

Attribute and Value	Indicates that the Method is…
UNKNOWN = -1	unknown
CLASSM = 0	a class method
INSTANCEM = 1	an instance method
FACTORYM = 2	a factory method

OBJECT_TYPE

The Object_Type returned value is of an enumeration type.

Attribute and Value	Indicates that the Object is…
CUNKNOWN = -1	unknown
CREPOSITORYSERVICES = 0	a repostiroy services object
CBUSINESSOBJECTS = 1	a business object collection
CBUSINESSOBJECT = 2	a business object
CMETHODS = 3	a method collection
CMETHOD = 4	a method
CPARAMETERS = 5	a parameter collection
CPARAMETER = 6	a parameter
CSTRUCTURE = 7	a structure
CTABLE = 8	a table
CFIELDS = 9	a field collection
CFIELD = 10	a field
CAPPLICATIONHIERARCHIES = 11	an application hierarchy collection
CAPPLICATIONHIERARCHY = 12	an application hierarchy
CFUNCTIONGROUPS = 13	a function group collection
CFUNCTIONGROUP = 14	a function group
CFUNCTIONS = 15	a function collection
CFUNCTION = 16	a function
CEXCEPTIONS = 17	an exception collection
CEXCEPTION = 18	an exception
CSAPSYSTEMS = 25	a SAP system collection
CSAPSYSTEM = 26	a SAP system