Tech Note 1013
Using a COM Object for Supply Chain Connector Imports and Exports

Topic#: 002845
Created: February 2014

Introduction

For specialized data transfers in Supply Chain Connector (SCC) that require more complex functionality than a straight importing of table records, or the exporting of a query result to a file, you can write a method that is invoked via COM for greater flexibility.

Application Versions

- **MES Supply Chain Connector (SCC) version 4.5**

Invoking a Method of a COM-Visible Object from an SCC Schedule

Triggering an import schedule with the Import Data Type Import from COM object or an export schedule with the Transport Type Call COM object causes MES Service (or MES Supervisor, in the case when the **Test** button is pressed on the Summary tab) to:

- Create an instance of the object specified in the COM Object field using late binding.
- Call the method specified in the Method to Call on COM Object field.

The contents of the COM Object field must be a fully qualified name that is the format of **Namespace.Class.Version**.

For an import, the method must conform to the following signature:

- For Visual Basic: **Public Method xxxxxx** (ByVal session_id as long, ByVal sched_id as string, Optional ByVal context as string) as long.
- For C#: **public long xxxxxx** (long session_id, string sched_id, string context).

For an export, the method must conform to the following signature:

- For Visual Basic: **Public Method xxxxxx** (ByVal session_id as long, ByVal sched_id as string, ByVal rs as ADODB.recordset, Optional ByVal context as string) as long.
- For C#: **public long xxxxxx** (long session_id, string sched_id, ADODB.Recordset rs, string context).

The method utilizes the parameters passed to it to know what it is to do in each particular case:

- The **session_id** parameter is the session ID of the MES Service (or MES Supervisor) that triggered the import, and is used as a parameter in most stateless API methods.

- The **sched_id** parameter identifies the SCC schedule that triggered the import, and allows the COM object to extract additional configuration information from the dx_sched table (and other) if required, or to use the sched_id if creating additional entries in the SCC log table (dx_log), where it is a required field.
- In the case of a method called from an export schedule, the **rs** parameter is the recordset returned by the query associated with the export schedule.
- The optional context parameter allows passing of context information to the COM object. For example, context information can be passed to allow multiple schedules to call the same COM object yet have it respond differently, by passing different context values it can use to conditionalize different behaviors. It is always a string even though it is passed as a variant in Visual Basic. It is only included in the method call if it is not null and not an empty string.

The method should return a (long) **0** if the import was handled successfully or a (long) **-1** if an error occurred.

Although SCC will create entries in its log if it is unable to invoke the method or the method caused an unhandled exception, it otherwise does not know if the intended processing succeeded. Therefore, it is up to the method to make entries in the log (the **dx_log** table) if its internal processing was not successful, to provide users a record of what failed and, to the extent possible, why. Note that there are currently no stateless API methods to access the **dx_*** tables, so this must be done via a direct connection to the MES Middleware.

It is up to the method called by an input schedule to actually insert records into the MES database, preferably by using the stateless API. A method called by an export schedule can either use the ADODB recordset passed to it, or it could do its own query. However, if all the required information to be exported is in the recordset, a separate query is not required.

C# COM-Visible Example Code

The following C# code is an example of a COM object that contains an import method and an export method:

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Runtime.InteropServices;
namespace SCCCOMObjNamespace

{
    [GuidAttribute("B25215A6-CDB0-47A6-A3FD-DBDE69749198")]
    public interface ISCCCOMObjClass

    {
        long ImportMethod(long session_id, string sched_id,
        string context);
        long ExportMethod(long session_id, string sched_id,
        ADODB.Recordset rs, string context);
        void Close();
    }

    [ComDefaultInterface(typeof(ISCCCOMObjClass))]
    [ClassInterface(ClassInterfaceType.None)]
```

```
        [ComVisible(true)]
        [GuidAttribute("22C7350C-15CF-43E0-97C2-DE1A3150D4EE")]
        [ProgId("SCCCOMObjNamespace.SCCCOMObjClass.1")]
        public class SCCCOMObjClass : ISCCCOMObjClass, IDisposable

        {
            public long ImportMethod(long session_id, string sched_id, string context)

            {
                // Add processing code here
                return 0;
            }

            public long ExportMethod(long sessionId, string sched_id, ADODB.Recordset rs, string context)

            {
                // Following 3 lines show an easy way to create a dataset ds with the contents of the ADODB recordset rs

                OleDbDataAdapter da = new OleDbDataAdapter();
                DataSet ds = new DataSet();
                da.Fill(ds, rs, "ATableName");

                // Add processing code here
                return 0;
            }

            public void Close()

            {

            Dispose();

            }

            public void Dispose()

            {

            GC.SuppressFinalize(this);

            }
        }
    }
```

The two GUIDs in the GuidAttributes decorating the interface and class are created using the Create GUID choice in Visual Studio's Tools menu and will not be the same as those shown in this example. The DLL that will result from this code needs to be registered in the GAC using **REGASM** (found in the latest version folder beneath the Microsoft.NET\Framework folder in the Windows folder.

For example, **C:\Windows\ Microsoft.NET\Framework\v4.0.3019** using the **/codebase** option.

In this case the entries in the Data tab for the import schedule would be:

- Import Data Type: **Import from COM object**
- COM object: **SCCCOMObjNamespace.SCCCOMObjClass.1**
- Method to Call on COM Object: **ImportMethod**
- Method Context Parameter: <as needed>

The entries on the Transport tab for the export schedule would be:

- Transport Type: **Call COM object**
- COM object: **SCCCOMObjNamespace.SCCCOMObjClass.1**
- Method to Call on COM Object: **ExportMethod**
- Method Context Parameter: <as needed>

J. Stella

Tech Notes are published occasionally by Wonderware Technical Support. Publisher: Invensys Systems, Inc., 26561 Rancho Parkway South, Lake Forest, CA 92630. There is also technical information on our software products at **Wonderware Technical Support.**

For technical support questions, send an e-mail to **wwsupport@invensys.com**.

Back to top