

Provisioning with Microsoft Identity Integration Server 2003

A Technical Overview Document by Oxford Computer Group

Introduction

The ability to automatically create new user accounts in Network Operating System (NOS) directories, email servers, LDAP directories, databases, or any other accessible data source is one of the major benefits of an Identity Management strategy. This creation process is known as *provisioning*, and its converse, deletion of accounts, *deprovisioning*.

The benefits of having automated provisioning and deprovisioning are immediate and compelling. A reduction in administration overheads ensues when new accounts are entered manually into one system and then automatically flowed into many others. Enhanced security follows from the ability to automatically delete or suspend user accounts across the enterprise when an individual leaves the organization, or loses rights to resources.

Microsoft Identity Information Server 2003 (MIIS 2003), with its predecessor MMS 2.2, is the backbone of the Identity Management infrastructure in many organizations due to its ability to aggregate, manage and synchronize identity data across many different data stores. MIIS is also capable of automatically provisioning and deprovisioning accounts in any connected data stores, including Active Directory and Microsoft Exchange. For example, if a new user is added into an HR database running on SQL server, MIIS can be configured to automatically create a new Active Directory account for that user – along with an Exchange mailbox if desired.

This paper is for new users of MIIS wanting to understand more about provisioning; specifically how to configure MIIS to add and delete accounts in connected directories. As such, the paper assumes that readers have at least a basic understanding of metadirectories, MIIS, and such MIIS terminology as *Metaverse*, *connector space*, and *management agent*. Readers who are not familiar with such terminology should read our introductory white paper “MIIS 2003” available from the Oxford Computer Group website (<http://www.oxfordcomputergroup.com>).

Table of Contents

Introduction	ii
Provisioning with MIIS	1
Provisioning Concepts	1
Enabling Provisioning	2
Rules Extensions	3
Types of Rules Extension	3
Developing a Metaverse Rules Extension for Provisioning	4
Creating an Extension	4
Metaverse Extension Methods	4
Basic Steps in Provisioning	5
Provisioning Scenarios	7
Simple Provisioning to a File-Based MA	7
Provisioning to Active Directory	10
Provisioning to specific AD OUs	13
Deprovisioning	15
Metaverse Object Deletion	15
Rules Extensions for Metaverse Deletion	16
Configure Deprovisioning	16
Oxford Computer Group	18
Service Offerings:	18
Modus Operandi:	18
Glossary	19

Table of Figures

Figure 1 – MIIS Provisioning Concepts	1
Figure 2 – Basic Provisioning Steps	5
Figure 3 – Simple Provisioning Scenario	7
Figure 4 – Active Directory Provisioning Scenario	10

Provisioning with MIIS

Provisioning Concepts

From an MIIS perspective, provisioning is simply the action of creating a new connector in the connector space of a Management Agent (MA), ready for export. During the next export run of that MA, a new account, or new object will be created that corresponds to the connector space object. Similarly, deprovisioning is the marking of an object for deletion in the connector space ready for export to the connected data source, where the actual account deletion will take place.

Typically (but not exclusively), there will be a Management Agent connected to a data store that is authoritative for new account creation, most often the HR database, or perhaps a customer billing system. When new entries are created in this data store, they are imported into the Metaverse. With provisioning enabled, the new Metaverse object causes corresponding pending exports to be created in other MAs, as required. During subsequent synchronization, individual attributes can be provided by any of these data sources.

The following diagram shows this concept:

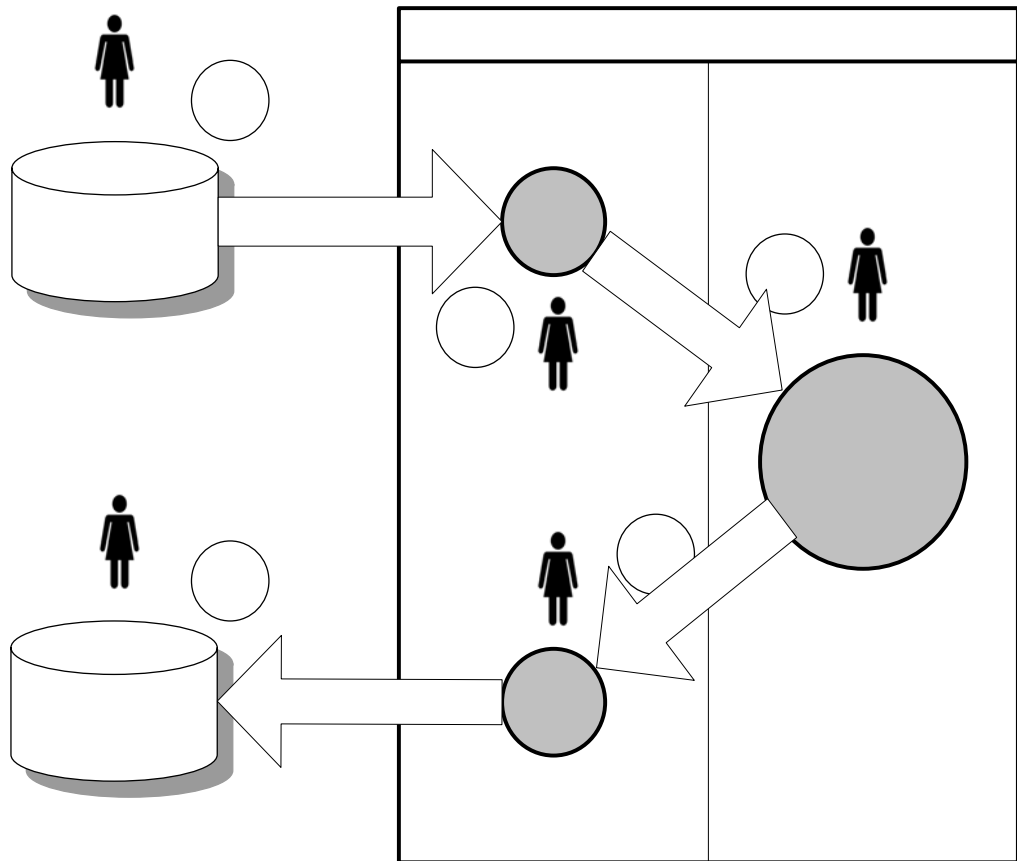


Figure 1 – MIIS Provisioning Concepts

1. A new user entry is entered manually into the HR database.
2. The HR database MA runs and discovers the new entry. A new connector is created in the HR MA connector space.
3. As the HR MA is configured to project new connectors into the Metaverse, a new Metaverse object is created that corresponds to the new connector.
4. As MIIS is configured for provisioning, the provisioning code runs during the import and a new pending export is created in the Active Directory MA connector space.
5. A new Active Directory user account is created the next time the Active Directory MA runs in Export mode.

1

HR Database

Management Agent

Deprovisioning would simply be the opposite sequence, starting with either the manual deletion of the HR database entry, or perhaps just the setting of a status attribute to indicate termination. However, depending upon the business rules in place, the Active Directory user account need not simply be deleted. It is possible instead to disable the account, or move it to another OU within Active Directory.

Another scenario is a central provisioning management system, where accounts are authorized, and MIIS, responding to this, actually provisions accounts in various Connected Directories. So rather than MIIS following easily programmable business rules to create and manage new accounts, they could be authorized "manually" through an application storing data in a specific directory such as and ADAM or SQLServer. Microsoft has produced Active Directory in Application Mode (ADAM) – a sort of cut down Active Directory – with just this (amongst other things) in mind. ADAM should perhaps be the subject of another paper!

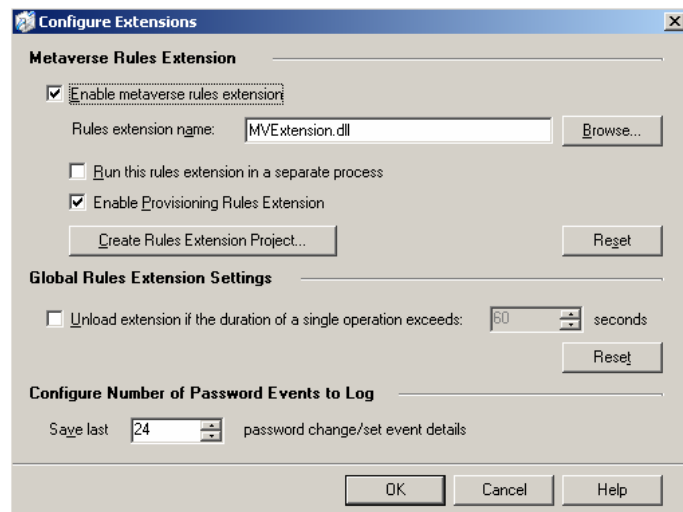
Yet another scenario is where no single MA is authoritative, but each is authoritative for its own metaverse entries. This sort of scenario is common where, for instance, multiple email systems are using MIIS for address book synchronization.

Enabling Provisioning

It is important to realise that provisioning is not enabled by default in MIIS. It is possible to create and configure MAs that project new Metaverse entries, and flow attributes between joined data sources, without the provisioning of new accounts into connected data sources. As long as there is a join between connected directory entries and Metaverse entries, attribute flow will take place in both directions: export and import. Enabling provisioning requires the deliberate intervention by an administrator via the Identity Manager console, and the creation of a Metaverse rules extension – in other words, you need to write code to control the provisioning process.

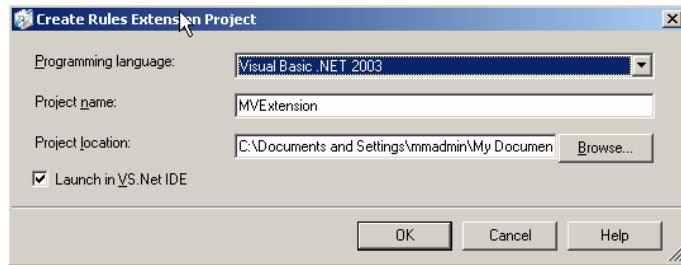
The following shows how to enable provisioning, but we strongly recommend that you don't do this until you have finished reading the paper and have a thorough understanding of rules extensions.

In Identity Manager, from the **Tools** menus, select the **Configure Extensions** option. This brings up the following dialog.



To enable provisioning:

1. Check the **Enable Metaverse rules extension** check-box.
2. Click the **Create Provisioning Rules Extension**.
3. The following dialog will be displayed, giving the option to choose between Visual Basic .NET or Visual C# .NET



4. If you have Visual Studio .NET installed, it will be invoked and a new Metaverse rules project framework will be built.
5. Edit the rules extension (discussed below) as required, and build the project. Close Visual Studio .NET and the new rules extension name "MVExtension.dll" will appear in the **Rules extension name** window.
6. Check that **Enable Provisioning Rules Extension** is checked, and select **OK**.

Rules Extensions

Rules extensions are dynamic link libraries (.dlls) used with MAs and the Metaverse to extend functionality. For example, a rule extension may be used to combine data from two source attributes from a connected directory (such as `sn` and `givenName`) and flow them to one Metaverse attribute (such as `displayName`). In this case, this would be a MA rules extension and would run appropriately when the MA runs. Rules extensions are created by using programming languages such as Microsoft Visual Basic .NET or C#.

Types of Rules Extension

Management Agent: A MA rules extension is applied at various points during a synchronization run. Each MA has one rules extension. MA rules include:

- Connector filter rules
- Join rules
- Projection rules
- Attribute flow rules
- Deprovisioning rules

Metaverse: A Metaverse rules extension is associated with Metaverse object deletion or provisioning, the latter code being run whenever a Metaverse object is processed (e.g. whenever it is created or modified). There can be only one Metaverse rules extension for each MIIS installation (although this extension can call out to code in other assemblies if they are appropriately referenced).

The following section provides more details on developing a Metaverse rules extension for provisioning.

Developing a Metaverse Rules Extension for Provisioning

Creating an Extension

To create a Metaverse rules extension, follow the instructions in the previous section. If you are using Visual Studio .NET, a new project will be created and a framework for the rules extension will be presented, as shown below:

```
Imports Microsoft.MetadirectoryServices

Public Class MVExtensionObject
    Implements IMVSynchronization

    Public Sub Initialize() Implements IMVSynchronization.Initialize
        ' TODO: Add initialization code here
    End Sub

    Public Sub Terminate() Implements IMVSynchronization.Terminate
        ' TODO: Add termination code here
    End Sub

    Public Sub Provision(ByVal mventry As MVEntry) Implements
        IMVSynchronization.Provision
        ' TODO: Remove this throw statement if you implement this method
        Throw New EntryPointNotImplementedException()
    End Sub

    Public Function ShouldDeleteFromMV(ByVal centry As CSEntry, ByVal mventry
        As MVEntry) As Boolean Implements IMVSynchronization.ShouldDeleteFromMV
        ' TODO: Add MV deletion code here
        Throw New EntryPointNotImplementedException()
    End Function
End Class
```

Metaverse Extension Methods

As can be seen above, the Metaverse rules extension supports 4 methods:

- **IMVSynchronization.Initialize.** The first time that the rules extension DLL is invoked, this Initialize method is called. It is used to initialize the environment in which the other methods will run repeatedly for each object. For example, it can be used to set up parameters, open a link to another database, or read a file.
- **IMVSynchronization.Terminate.** The Terminate method is called when the rules extension object is no longer needed, typically 5 minutes after the last time the DLL is called. This method is used to free resources owned by the rules extension – typically these will be those allocated by the Initialize method.
- **IMVSynchronization.Provision.** The Provision method is executed in response to a change to a Metaverse object. It is passed a single parameter, the Metaverse object that has changed, and has no return values. Note that although the method is called “Provision”, it may also be responsible for a wide variety of other activities (beyond Provisioning) which are required when an object changes. These may include Renaming, Moving, Disconnecting, and Deleting the Metaverse object.
- **IMVSynchronization.ShouldDeleteFromMV.** This method has nothing to do with provisioning, but allows a custom rule to be specified for Metaverse object deletion. It is called when a connector space entry is disconnected during an import operation. This method determines if the Metaverse entry connected to the disconnecting connector space entry should be deleted.

Basic Steps in Provisioning

Provisioning code can be fairly complex, but the following diagram illustrates some basic steps in a simple scenario that we will build on to create a simple provisioning rules extension code.

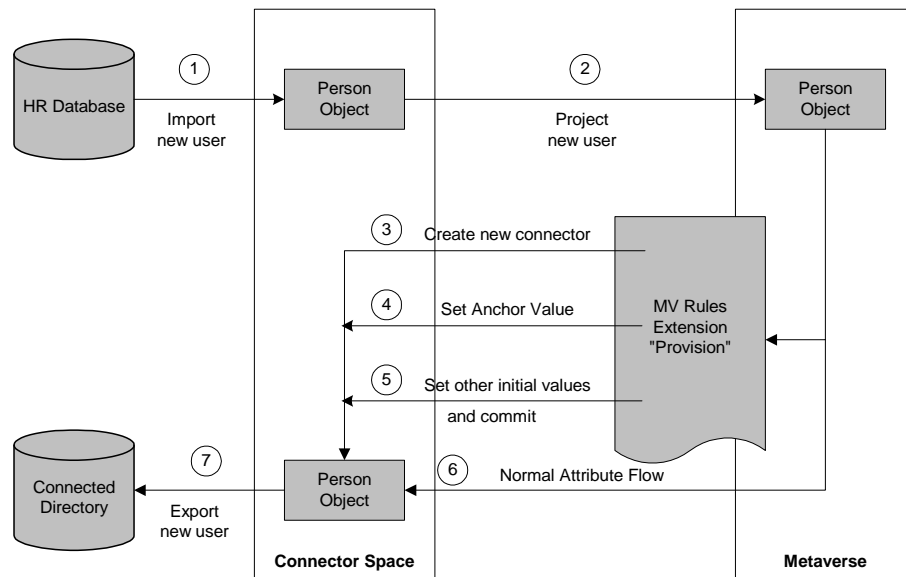


Figure 2 – Basic Provisioning Steps

1. The HR Database MA runs an import profile and imports a new user into the connector space as a "person" object.
2. When the HR MA is run with a synchronization profile, the new user is projected into the Metaverse. The MV rules extension is also invoked during the synchronization process.
3. The first step in provisioning is to define which connected directory we are provisioning. We identify the Management Agent which manages that connected directory (in this example, an MA called "Telephone TXT MA")

```
TargetMA = mentry.ConnectedMAs("Telephone TXT MA")
```

Next, we create a new connector object in the connector space of the identified management agent. This must be a connector of the appropriate type - in this case a "person" object class is used. In VB .NET, this would be expressed as

```
csentry = TargetMA.StartNewConnector("person")
```

4. The next step would be to set an anchor value for the new connector. This value will vary depending upon the connected directory type. For example, in Active Directory, it would be the DN, and for a SQLServer database it would be your chosen unique ID attribute. Again, in VB .NET this would be expressed as

```
csentry("Anchor").Value = [InitValue]
```

or

```
csentry.DN = [Generated DN]
```

5. You may need to set other attributes as new objects are created. For example, in AD you might set an initial password or some Exchange attributes. The new connector is then committed to the MIIS connector space database table. If the anchor value is not unique, the commit will fail.
6. After the provisioning code has completed, normal attribute flow rules configured for that MA will be applied, populating the new connector with Metaverse attribute values.

7. The management agent is run with an Export step, and this newly created connector space object is created in the connected directory.

It should be stressed that this is a very simplified view of things. In general you will have to foresee potential errors and handle them – for example, what if the anchor you have chosen turns out not to be unique, or if an attribute value on which you are relying is not present?

Provisioning Scenarios

This section uses two common scenarios to illustrate how a Metaverse rules extension may be developed for provisioning. In the first scenario, a very basic extension is developed to enable provisioning of new accounts to a file-based MA. The second scenario shows how to provision to Active Directory, including some more advanced DN construction based on Metaverse attribute values.

Simple Provisioning to a File-Based MA

In the following scenario, MIIS is being used to provision new accounts from a SQL server HR database into a file-based MA being used to populate a telephone system. The following diagram illustrates the scenario MIIS architecture:

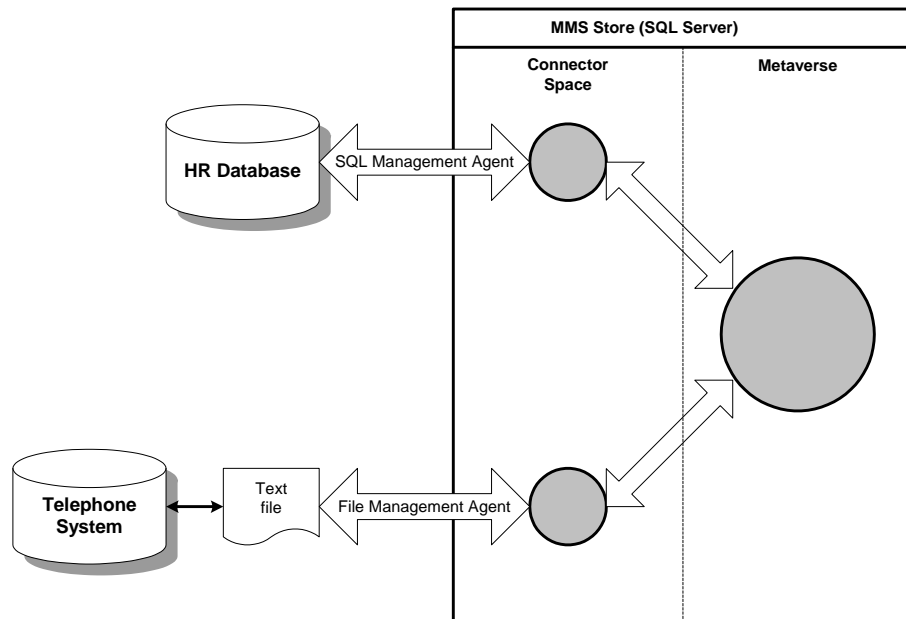


Figure 3 – Simple Provisioning Scenario

In the scenario pictured above, new Metaverse person objects are projected into the Metaverse from the HR database, and then provisioned to the telephone text MA. The following table shows the attribute flow in the HR SQL MA into the Metaverse. Note that the Metaverse attributes displayName and cn are constructed from the SQL MA rules extension *not* the Metaverse rules extension (although cn isn't used in this example, in general it would be useful for provisioning an LDAP CD).

Data Source Attribute	Metaverse Attribute	Type	Flow Nulls
Object Type: person			
JobTitle	title	Direct	
FirstName,LastName	displayName	Rules Extension - displayNa...	
Status	employeeStatus	Direct	
ReportsTo	manager	Direct	
OfficeLocation	location	Direct	
LastName	sn	Direct	
homePhone	homePhone	Direct	
FirstName	givenName	Direct	
employeeID	employeeID	Direct	
email	mail	Direct	
Department	department	Direct	
Mobile	mobile	Direct	
FirstName,LastName	cn	Rules Extension - cn	

The Telephone Text MA is configured to take entries from a fixed-width file and join them to the Metaverse person objects projected from the HR SQL MA, flowing certain attributes *into* the Metaverse. The join takes place between the EMPID attribute in the text file and the employeeID Metaverse attribute. The following table shows the attribute flow from the Telephone Text MA to the Metaverse:

Data Source Attribute	Metaverse Attribute	Type	Flow Nulls
Object Type: person	Object Type: person		
MOBILE	mobile	Direct	
FAX	facsimileTelephoneNumber	Direct	
PAGER	pager	Direct	
TELEPHONE	telephoneNumber	Direct	

To create new entries to be exported to the Telephone system, provisioning must be enabled and a Metaverse rules extension created. The following code shows a Metaverse rules extension suitable written in VB .NET for such a job.

```
Public Sub Provision(ByVal mventry As MVEntry) Implements _
    IMVSynchronization.Provision

    ' Provision new MV entries into the Telephone Text MA
    ' First, the telephone system only takes person objects

    If mventry.ObjectType.Equals("person") Then

        ' Dimension and assign some variables
        Dim centry As CEntry
        Dim TelephoneMA As ConnectedMA
        TelephoneMA = mventry.ConnectedMAs("Telephone TXT MA")

        ' If there is no connector present, add a new telephone MA connector
        If TelephoneMA.Connectors.Count = 0 Then

            centry = TelephoneMA.Connectors.StartNewConnector("person")

            ' Set the initial values at the same time and commit
            centry("EMPID").Values.Add(mventry("employeeID").Value)
            centry("NAME").Values.Add(mventry("displayName").Value)
            centry.CommitNewConnector()

        ElseIf TelephoneMA.Connectors.Count = 1 Then
            ' Nothing to do, connector already exists for this MV object

        Else
            ' Shouldn't be multiple connectors
            Throw New UnexpectedDataException("multiple connectors:" +
                TelephoneMA.Connectors.Count.ToString)

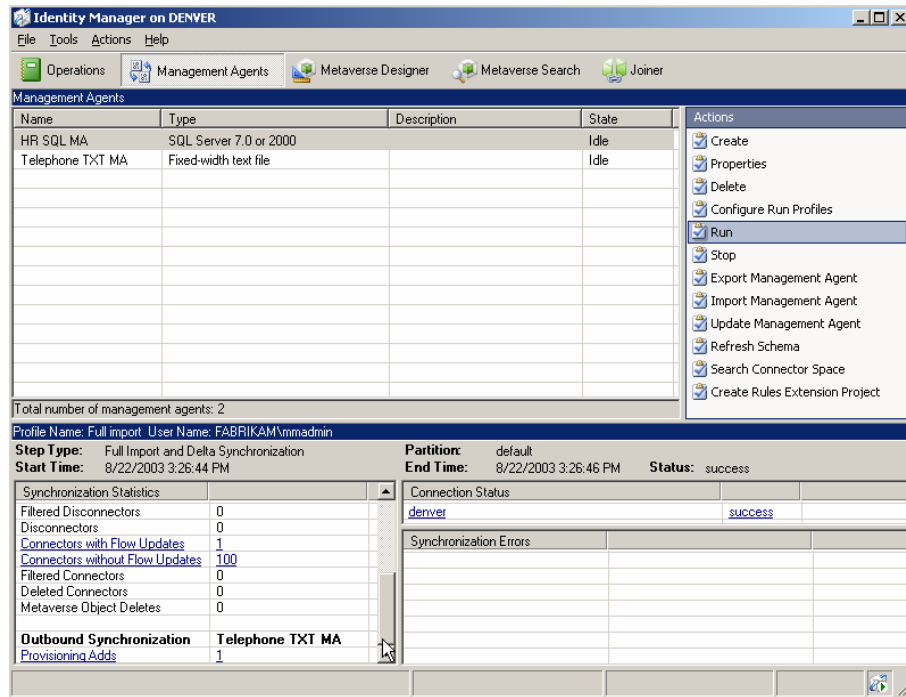
        EndIf
    EndIf
End Sub
```

Now, whenever a new Metaverse object is projected, a new connector will be created in the Telephone MA connector space, with the initial attributes "EMPID" and "NAME" set to the Metaverse "employeeID" and "displayName" attribute values. As noted previously, it is lucky for us that we can rely on the EMPID attribute to be unique, as the code above doesn't contain any way of checking this, or of suggesting alternatives if EMPID turns out not to be unique.

For example, in our scenario, a new user "Andrew Flintoff" is added to the HR SQL Database as shown below:

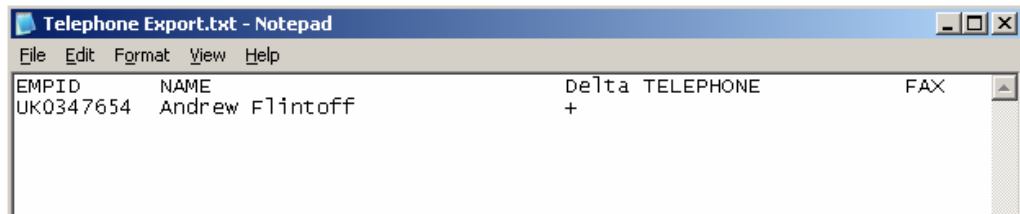
employeeID	FirstName	LastName	JobTitle	Status	WorkTelNo	WorkFax
UK0347654	Andrew	Flintoff	Director	Active	<NULL>	<NULL>

This new user is discovered and projected into the Metaverse during the next HR SQL MA import and synchronization run.



It is important to understand that the new user has been provisioned to the Telephone TXT MA connector space, even though the Telephone TXT MA has not run. This is because the Metaverse rules extension (containing the provisioning code) is triggered automatically by the creation of a new Metaverse object.

In order to export the user, the Telephone TXT MA must now be run in Export mode. The new user appears in a flat file in the appropriate format, with the initial values set.



Finally, the file becomes an import file for our telephone system. The Exports are held in a "in progress" state until a subsequent import from the telephone system shows that it has the new accounts, and MIIS can return to a normal state.

Provisioning to Active Directory

In this scenario, an Active Directory MA is added to the MIIS architecture in the previous example. The architecture now looks like this:

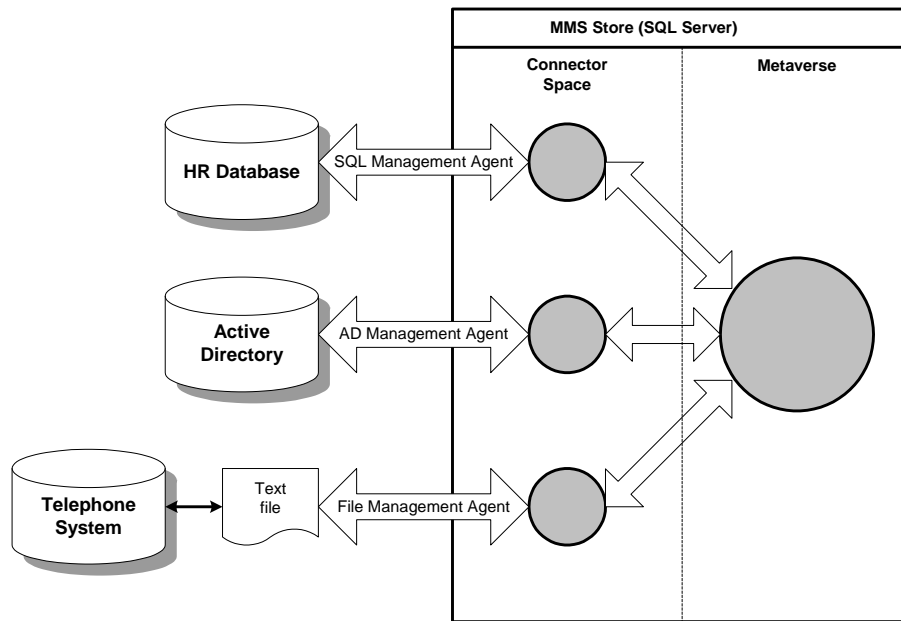
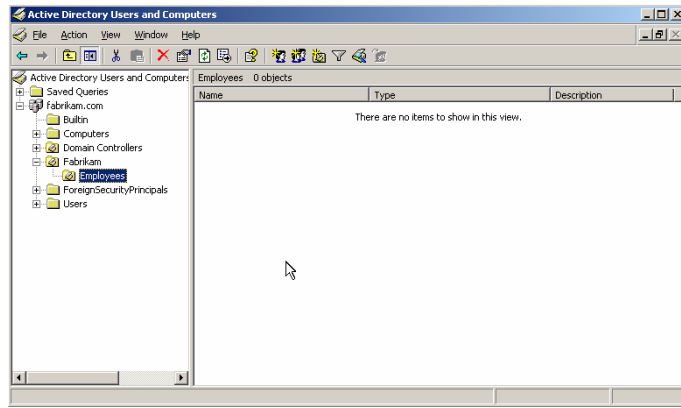


Figure 4 – Active Directory Provisioning Scenario

To prepare for the AD provisioning, the HR SQL MA attribute flow has been extended to include attribute flow rules for useful AD attributes: UserPrincipalName (which must be unique within a forest) and SamAccountName (which is the pre-Windows 2000 logon and must be unique within its domain). Both of these AD attributes will be constructed using the HR SQL MA rules extension. SamAccountName will be constructed in the form "FirstName.LastName@fabrikam.com" then the MV attribute "uid" will be constructed in the form "firstinitialofFirstNameLastName" and flowed to UserPrincipal name on export.

Data Source Attribute	Metaverse Attribute	Type	Flow Nu
homePhone	homePhone	Direct	
FirstName	givenName	Direct	
employeeID	employeeID	Direct	
email	mail	Direct	
Department	department	Direct	
Mobile	mobile	Direct	
FirstName,LastName	cn	Rules Exten...	
FirstName,LastName	uid	Rules Exten...	
FirstName,LastName	samAccountName	Rules Exten...	

The AD MA is configured to import a container within Active Directory called ou=Employees, ou=Fabrikam, dc=fabrikam, dc=com. All new users from the HR SQL MA will be provisioned to this container. You have to import the container structure before you can provision into it.



In this case, the AD MA will only ever export users and attributes from the Metaverse to Active Directory; therefore the attribute flow is configured to flow that way too. However, in practice, attributes will very often flow both ways.

Data Source Attribute		Metaverse Attribute	Type	Flow Nulls
Object Type: user		Object Type: person		
cn	←	cn	Direct	
displayName	←	displayName	Direct	
givenName	←	givenName	Direct	
manager	←	manager	Direct	
mobile	←	mobile	Direct	
sAMAccountName	←	samAccountName	Direct	
sn	←	sn	Direct	
telephoneNumber	←	telephoneNumber	Direct	
userPrincipalName	←	uid	Direct	

The following code for AD is equivalent to the earlier file-based MA example and will create a new connector in the AD MA connector space, ready for export:

```

Public Sub Provision(ByVal mventry As MEntry) Implements _
    IMVSynchronization.Provision

    If mventry("cn").IsPresent Then
        Dim centry As CEntry
        Dim dn As ReferenceValue
        Dim rdn As String
        Dim ParentContainer As String = _
            "OU=employees,OU=Fabrikam,DC=fabrikam,DC=com"
        Dim Connected_AD_MA As ConnectedMA

        Connected_AD_MA = mventry.ConnectedMAs("Fabrikam AD MA")
        'Construct the dn
        rdn = "CN=" + mventry("cn").Value
        dn = Connected_AD_MA.EscapeDNComponent(rdn).Concat(ParentContainer)

        ' If there is no connector present, add a new AD connector
        ' and add a password

        If Connected_AD_MA.Connectors.Count = 0 Then
            centry = Connected_AD_MA.Connectors.StartNewConnector("user")
            centry.DN = dn
            centry("unicodepwd").Values.Add("PA$$w0rd")
            centry.CommitNewConnector()

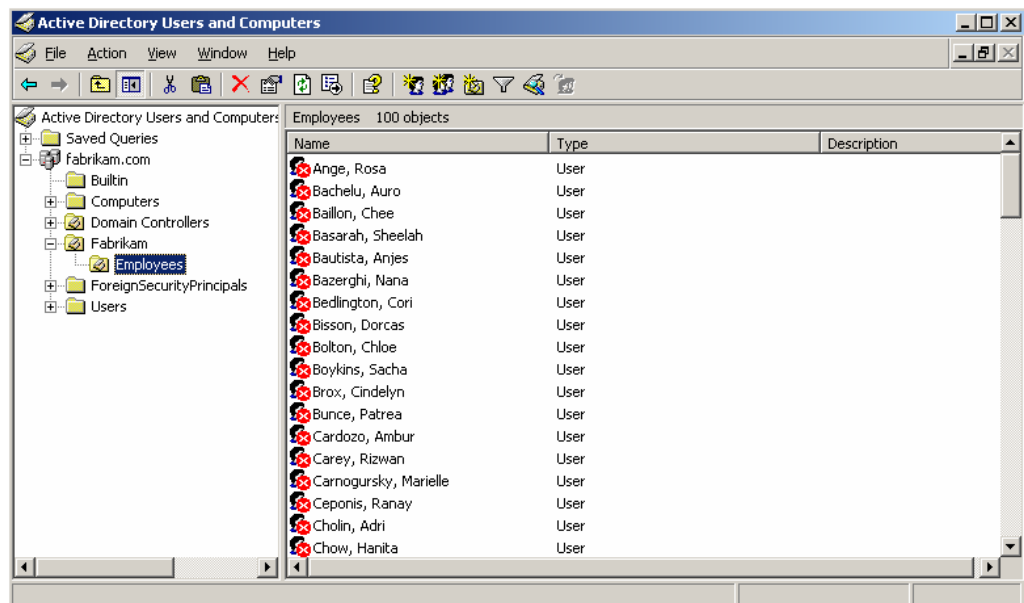
        ElseIf Connected_AD_MA.Connectors.Count = 1 Then
            'Grab the existing connector and reset the dn (it might have changed)
            centry = Connected_AD_MA.Connectors.ByIndex(0)
            centry.DN = dn

        Else
            Throw New UnexpectedDataException("multiple AD connectors:" +
                Connected_AD_MA.Connectors.Count.ToString)

        End If
    End If
End Sub

```

Once saved and rebuilt this simple code will cause new connectors to be added in the AD MA connector space as pending exports for any existing Metaverse objects whenever a full synchronization run profile is run on any MA. Note, though, that for simplicity, no attempt has been made to foresee and cope with data problems such as missing attribute values or the DN turning out not to be unique – in a real case these would have to be considered. Running an export profile on the AD MA causes these pending adds to be exported into Active Directory where the new user accounts are created.



Notice that the above new entries have all been created disabled. This is because no value for the AD attribute "userAccountControl" was exported, so AD used its default policy. We could

have set the userAccountControl attribute so as to enable these (and could continue to set this) according to some Metaverse attribute.

Provisioning to specific AD OUs

In the above example, all new accounts were created in a generic "Employees" OU. However, by altering our code, we could arrange for new accounts could be added to specific OUs based upon a Metaverse attribute such as "location" or "department", and for them to be subsequently moved in response to any change.

In our example, all employees added to the HR SQL DB have an attribute called "OfficeLocation" that is projected into the Metaverse as the Metaverse attribute "location". To provision new employees into an Active Directory that matches this attribute the following code will be required.

First, we define variables to hold the OUs. Alternatively we could set up parameters in an XML configuration file and then use the IMVSynchronization.Initialize method to read these in, avoiding the need to recompile whenever these change:

```
Public Class MVExtensionObject
    Implements IMVSynchronization

    ' The container to be used in AD
    Dim AD_LContainer As String = "OU=London,OU=Fabrikam,DC=fabrikam,DC=com"
    Dim AD_TContainer As String = "OU=Teddington,OU=Fabrikam,DC=fabrikam,DC=com"
    Dim AD_CContainer As String = "OU=Cambridge,OU=Fabrikam,DC=fabrikam,DC=com"
    Dim AD_DContainer As String = "OU=Didcot,OU=Fabrikam,DC=fabrikam,DC=com"
    Dim AD_BContainer As String = "OU=Bracknell,OU=Fabrikam,DC=fabrikam,DC=com"
```

These can then be used in an obvious Case structure when setting the dn, as shown below. In reality, this code could be made a good bit more elegant (e.g. by arranging things to make the attribute part of the container name, but it has been left "raw" for transparency). The provision sub-routine now looks like this:

```
Public Sub Provision(ByVal mventry As MVEntry) Implements _
    IMVSynchronization.Provision

    If mventry("cn").IsPresent Then
        Dim centry As CSEntry
        Dim dn As ReferenceValue
        Dim rdn As String
        Dim AD_Container As String
        Dim Connected_AD_MA As ConnectedMA

        Connected_AD_MA = mventry.ConnectedMAs("Fabrikam AD MA")
        'Construct the dn
        rdn = "CN=" + mventry("cn").Value

        ' Decide which container to use on a Case basis
        Select Case mventry("Location").Value
            Case "London"
                AD_Container = AD_LContainer
            Case "Teddington"
                AD_Container = AD_TContainer
            Case "Cambridge"
                AD_Container = AD_CContainer
            Case "Didcot"
                AD_Container = AD_DContainer
            Case "Bracknell"
                AD_Container = AD_BContainer
        End Select

        dn = Connected_AD_MA.EscapeDNComponent(rdn).Concat(AD_Container)

        ' If there is no connector present, add a new AD connector
```

The following code gets rid of the need to declare the containers as constants and the case method altogether:

```

Public Sub Provision(ByVal mventry As MVENTry) Implements
IMVSynchronization.Provision

    If mventry("cn").IsPresent Then
        Dim centry As CSEntry
        Dim dn As ReferenceValue
        Dim rdN As String
        Dim AD_Container As String
        Dim Connected_AD_MA As ConnectedMA

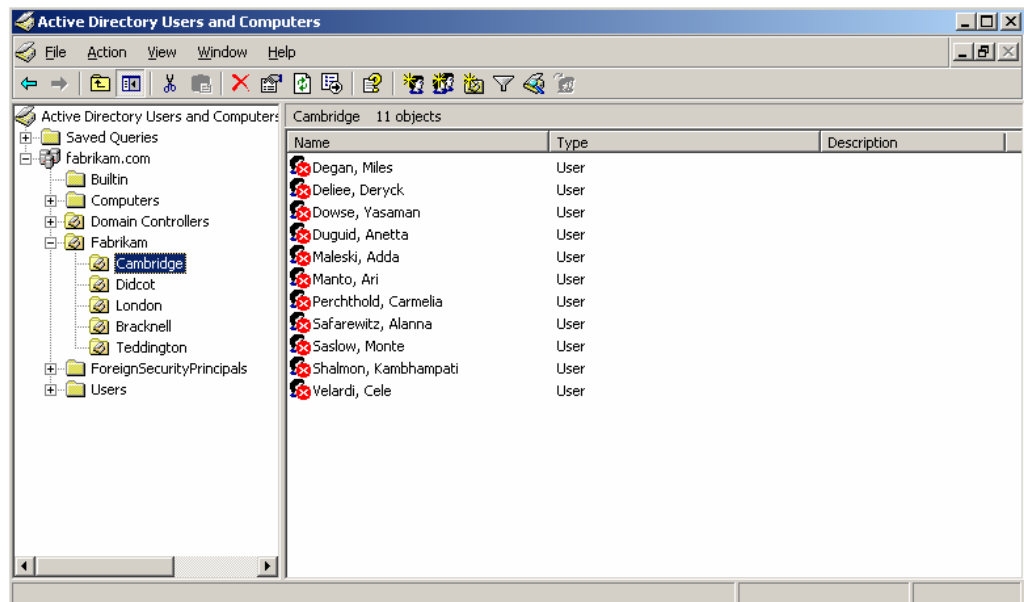
        Connected_AD_MA = mventry.ConnectedMAs("Fabrikam AD MA")
        'Construct the dn
        rdN = "CN=" + mventry("cn").Value

        AD_Container = String.Format("OU={0},OU=Fabrikam,DC=fabrikam,DC=com",
            mventry("Location").Value)

        dn = Connected_AD_MA.EscapeDNComponent(rdN).Concat(AD_Container)
    End If
End Sub

```

The results of this provisioning code are shown below: each new AD user has been provisioned to the correct OU.

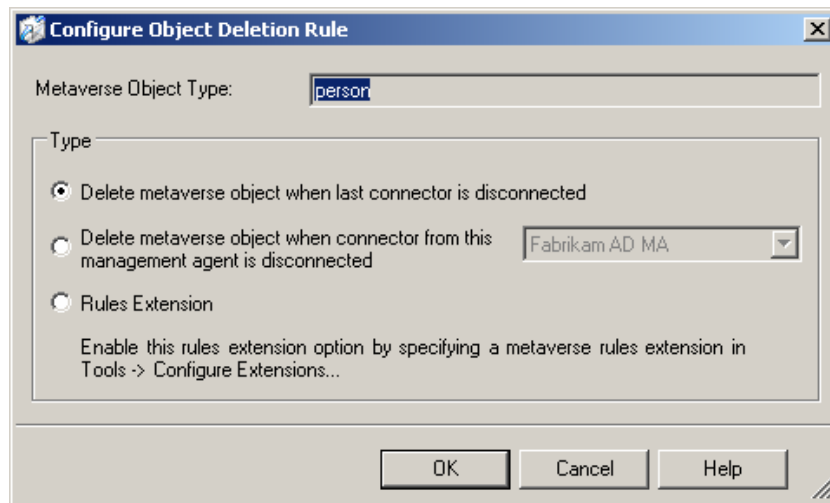


Deprovisioning

As stated earlier, deprovisioning is the marking of an object for deletion in the connector space ready for export to the connected data source, where the actual account deletion will take place. Note that the object is not necessarily deleted in the Metaverse, only in the connector space of the MA to which the deletion needs to be exported. However, in order to understand deprovisioning, it is necessary first to understand the rules for deleting objects from the Metaverse.

Metaverse Object Deletion

Once an object has been projected into the Metaverse, it cannot simply be deleted by hand: there is no “delete” function for Metaverse objects in the Identity Manager UI. Objects are deleted from the Metaverse based on the Metaverse object deletion rule for that object type. The object deletion rule is set via the Metaverse Designer as shown below:



There are three possible deletion rules:

- Delete Metaverse object when last connector is disconnected.** This is the default deletion rule for Metaverse objects and means that Metaverse objects will not be deleted until the last connector joined to that object is deleted or disconnected. It is this deletion rule that causes much scratching of heads amongst people attempting deprovisioning for the first time; they delete an object in a connected directory, it is deleted in the connector space of that MA, but the Metaverse object remains, and the deletion is not exported to other CDs.
- Delete Metaverse object when connector from this management agent is disconnected.** This rule is very useful when there is one MA is authoritative for all deletions. If an object is deleted in the connected directory, the deletion is passed to the Metaverse and automatically causes connectors to this Metaverse object in all other connector space to become disconnected. What happens after the disconnection is discussed further below.
- Rules Extension.** It is possible to trigger the Metaverse deletion based upon a custom rules extension. This is useful when there is no single authoritative MA, but that new additions and deletions come from many MAs. The most common rules extension would be to declare “delete the Metaverse object when deleted from the MA that created it”.

So, Metaverse object deletion occurs when a connector has been disconnected from the Metaverse *and* that disconnection triggers one of the above rules. The object will not be deleted from the Metaverse unless the rule in force is satisfied.

It is also important to remember that it is not just deletion in the connected directory that can cause disconnection from the Metaverse. A connection filter (responding to changing data) may cause disconnection.

Rules Extensions for Metaverse Deletion

If "Rules Extension" is selected as the Metaverse deletion rule, coding for this rule is contained within the Metaverse rules extension, within the "ShouldDeleteFromMV" method. The following code could be used for the example given above, deleting an object based upon its deletion from the MA that created it.

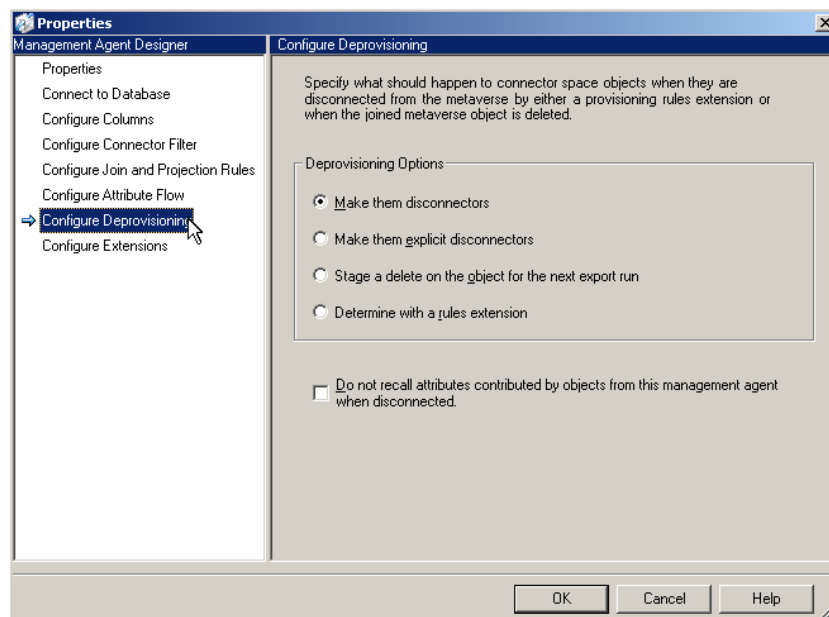
In this example, two AD MAs provision new users into an LDIF MA ready for export. Both are authoritative for users created in their respective connected directory. Therefore if a deletion occurs in one, it should trigger a Metaverse deletion and thereby cause any connectors in the LDIF MA to be disconnected. The code accomplishes this by checking to see if the MA doing the disconnection is the same one that set a certain attribute (in this case "uid"). If it is, then it must be the MA that created this connector, and the Metaverse object can be deleted.

```
Public Function ShouldDeleteFromMV(ByVal centry As CSEntry, ByVal mventry As
MVEEntry) As Boolean Implements IMVSynchronization.ShouldDeleteFromMV

    If centry.MA.Name = mventry("uid").LastContributingMA.Name Then
        ShouldDeleteFromMV = True
    Else
        ShouldDeleteFromMV = False
    End If
```

Configure Deprovisioning

Deleting Metaverse objects triggers deprovisioning, but what happens to the disconnector left behind in the connector space of other MAs after the Metaverse object is deleted? This is controlled by the *Configure Deprovisioning* rule, which is part of each MA's configuration, as shown below:



The Configure Deprovisioning rule asks a question: what should happen to connector space objects (for this MA only) when they are disconnected from the Metaverse? There are four possible answers:

- **Make them disconnectors.** These will be normal disconnectors and will attempt to join back to the Metaverse object, should it reappear.
- **Make them explicit disconnectors.** Explicit disconnectors will remain in the connector space, but won't attempt to join again.
- **Stage a delete on the object for the next export run.** This will export the deletion to the connected directory the next time the MA runs in export mode.
- **Determine with a rules extension.** Apply a rules extension to determine what to do:

Of the four examples above, the bottom two are the most commonly used when doing "true" deprovisioning, ie deleting an object in a connected directory that has previously been created by MIIS. Staging a delete will do just that – remove the object in the connected directory, but can be a bit of blunt instrument. In some cases, it may be preferable to apply a rules extension to the process to apply a more subtle approach. For example, if deprovisioning to AD, the rules extension could just set the UserAccountControl attribute to change the state of the user account from "Active" to "Inactive". Or, instead of deletion, the object could simply be disabled and moved to a "deleted" container so that the object details are not lost forever.

Oxford Computer Group

Oxford Computer Group (OCG) is an IT services company who specialise in Identity Management. Established in 1983 and recently reformed, our services include: strategic and functional consulting; system integration; development; and project management. Skill development and skills transfer usually form an integral part of our projects, depending on our client's desire to take a full and active participation in the project or to extend the solution.

Understanding and deploying MIIS is a task which requires a high degree of commitment, planning and expertise, but the rewards for deployment far outweigh the risk, especially if you minimize this risk by partnering with someone who knows the product inside-out. There are a number of good reasons you should talk to us if you have identity management needs or are considering an MIIS project:

- **OCG understand Identity Management and its potential value to your business.** We also understand the issues and challenges involved.
- **OCG have extensive experience of MIIS 2003.** We have expertise available that extends from the original Zoomit Via, through early beta versions right through to the final release. We can leverage this experience to make your MIIS deployment as smooth as possible.
- **OCG have over 20 years in the business of strategic consulting, project management, infrastructure implementation, knowledge transfer and development.** Our consultants are all highly skilled and experienced professionals, as comfortable in front of the Board as they are in front of a computer – both skill sets you will need to make your MIIS deployment go as planned.
- **Microsoft selected OCG to produce and deliver the first training courses on MIIS 2003.** We are the experts and have trained the rest of the market – including some of Microsoft's own consultants.

Service Offerings:

To assist you with your MIIS/IdM project, we offer the following services:

- **Strategic and functional consulting.** We can work alone or alongside your team through all stages of your MIIS deployment, making sure our expertise and experience works to your benefit.
- **Skill development and skills transfer.** Providing the necessary capability and skills is key to reducing the long term cost of ownership of an implementation and guaranteeing success. We can provide formal training, transfer skills by working alongside your team, or provide a managed service to support the solution post implementation.
- **Custom Development.** If you have the skills for most of your MIIS project in house, but would prefer to outsource component development, we are at your disposal.
- **Project management.** A typical MIIS project involves co-ordinating multiple departments, systems and development activities – some serial, some parallel. We can provide the strong controls and communication you need.

Modus Operandi:

The OCG philosophy is to deliver real business value, provide success, and reduce risk and cost. We enjoy what we do and it is important to us that our services are valued. The pressure on our customers is to provide clear and measurable returns from their IT investments in short time-frames. We like to realise this and in addition build long term relationships based on trust and achievement. Our business model is to provide highly skilled people with low overheads, delivering an exceptional return on investment. You pay for the capability and service, not the organisation.

Glossary

AD	- Active Directory
AD/AM	- Active Directory/Application Mode (AD "Lite")
API	- Application Programming Interface
Connector Space	- A staging area in the MIIS database used by management agents to move data into and out of a connected data source
CRM	- Customer Relationship Management
DCOM	- Distributed Component Object Model
DLL	- Distributed Link Library
ERP	- Enterprise Resource Planning
GAL	- Global Address List
GUI	- Graphical User Interface
HR	- Human Resources
IdM	- Identity Management
LDAP	- Lightweight Directory Access Protocol (RFC 1777)
LDIF	- LDAP Data Interchange Format
HTTP	- Hypertext Transfer Protocol
MA	- Management Agent
Metaverse	- A set of tables in the MIIS database that contain the integrated identity data from multiple connected sources
MMS	- Microsoft Metadirectory Services
MIIS	- Microsoft Identity Integration Server
NDS	- Novell Directory Services
NOS	- Network Operating System
OU	- Organizational Unit
PABX	- Private Automatic Branch Exchange
PKI	- Public Key Infrastructure
RDBMS	- Relational Database Management System
ROI	- Return on Investment
SQL	- Structured Query Language
WMI	- Windows Management Interface
XML	- Extensible Markup Language
X.500	- ISO Standard for Global Directory Services