Pro Exchange

Microsoft Identity Integration Server
Setup and Configuration Guide
Exchange 5.5 to Active Directory
Document version 1.0
March 2, 2005

Introduction

There are situations where Exchange 5.5 will need to coexist with other directories for extended amounts of time. It is estimated that over 20 million Exchange 5.5 seats will remain in that environment for years to come. This is a troubling thought for Microsoft and indeed for those procrastinating companies since Exchange 5.5 support is over.

Having said that, the estimate remains and so does the problem of coexistence. In this example, we have an Exchange 5.5 organization that will need to stay synchronized with several other Active Directory environments running Exchange 2000 and Exchange 2003. To support this requirement, we need to ensure we have selected the fields and formats needed to allow message flow to work across different system types.

Project Objectives

The first thing we need to do is map out the directory requirements:

From the AD Domain to Exchange 5.5

a) Mail-enabled Contacts --> Exchange 5.5 custom recipients
   Name, Address, Company, title and Phone Number Fields
   SMTP, X.500 and X.400 addresses

b) Mailbox-enabled Users --> Exchange 5.5 custom recipients
   Name, Address, Company, title and Phone Number Fields
   SMTP, X.500 and X.400 addresses

c) Mail-enabled groups --> Exchange 5.5 custom recipients
   SMTP, X.500 and X.400 addresses

From Exchange 5.5 to the AD Domain

a) Exchange 5.5 custom recipients --> Mail-enabled Contacts
   Name, Address, Company, title and Phone Number Fields
   SMTP, X.500 and X.400 addresses

b) Exchange 5.5 custom recipients --> Mailbox-enabled Users
   Name, Address, Company, title and Phone Number Fields
   SMTP, X.500 and X.400 addresses

c) Exchange Distribution Lists --> Mail-enabled Contacts
   SMTP, X.500 and X.400 addresses
   Ownership and Support

While building the system, we found many more useful fields and attributes that should be added and we created intelligent discovery and join rules as well as highly detailed projection rules. This basic list you see above expanded to include hundreds of fields.

Product Choice

As you can imagine, there are many products available to perform this type of directory synchronization. HP offers a product called LDSU and less-expensive offerings such as SimpleSync are also available. We chose Microsoft's Identity Integration Server 2003 because of its support for AD, Exchange and ability to synchronize with Lotus Notes, SQL and DBMS.

Test Lab Environment

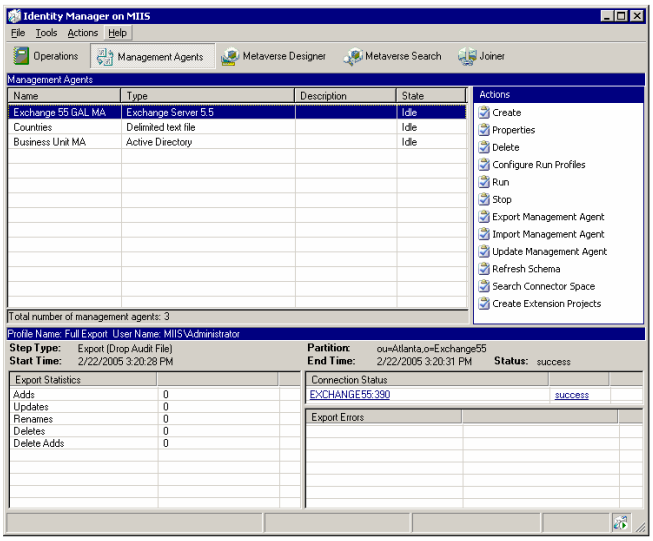
The most important aspect of MIIS is your test environment. It is from this environment that you test new scripts, attribute flows, join rules, etc. For us, we have built an entire lab environment that includes MIIS, Exchange 5.5 and Exchange 2003 (on Windows 2000 Server) in virtual images that can be transported, copied and distributed to those who need a better understanding of the systems.

Servers and Configuration

There are three servers in this test environment. All are patched to the current date and homogenized with test data.

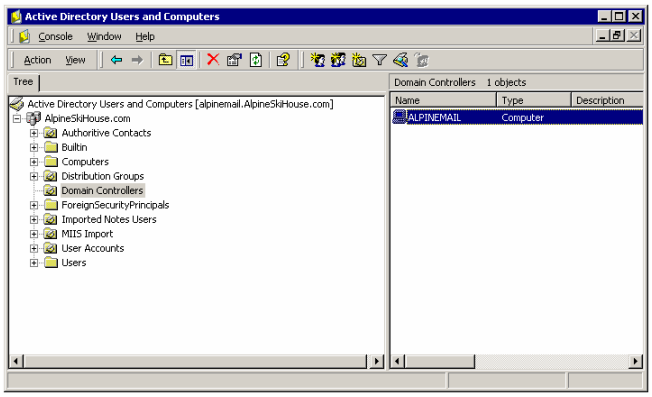
MIIS

The MIIS virtual machine has all the required components installed locally to fully manage and run the MIIS environment. SQL Server 2000 Enterprise, SP3 is installed and fully patched. Microsoft Identity Integration Server 2003, SP1 is also installed. In order to manage and maintain MIIS, Microsoft Visual Studio .NET 2003 is installed as well.



Computer name	MIIS
WorkGroup	WorkGroup
IP	10.10.20.233/24
Administrator	administrator
Password	MSEvent.123

Active Directory



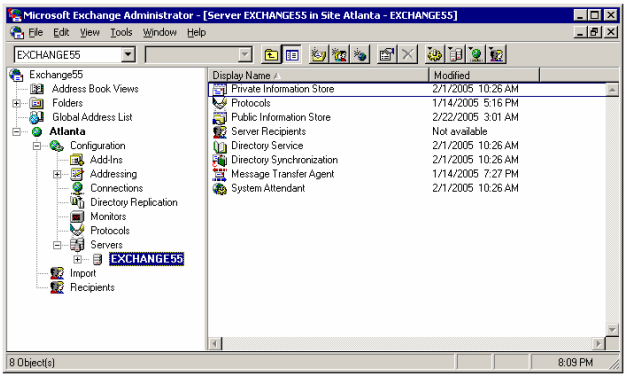
On this image, we have Exchange Server 2003 as well as the domain services for the Alpineskihouse.com domain. Windows Server 2000 in installed as is the Support Tools (ADSIEdit is handy for this type of work)

Computer Name	ALPINEMAIL
Domain	alpineskihouse.com
IP	10.10.20.231/24
Administrator	administrator
Password	MSEvent.123

Exchange 5.5

The MIIS virtual machine is running Windows 2000 Server and Exchange 5.5. It is a domain controller as well as the global catalog server for EX5.com

Computer name	EXCHANGE55
Domain	EX5.COM
IP	10.10.20.232/24
Administrator	administrator
Password	MSEvent.123



Setting up the MIIS Environment

Permissions

To help reduce the risk of applying changes back to the source systems, it is very important that each environment establish a working account for the MIIS system to use. In our examples, we have created an account in both domains called MIIS. These accounts should not be in the administrators or domain administrators group.

MIIS Account name	MIIS
Password	MSEvent.123

Active Directory

It is important that this account not be given too much access to the Active Directory.

- In the root of the Active Directory domain, give the MIIS account the following permissions:
  - Read
  - Replicating Directory Changes
  - Replication Synchronization
- In the container or OU you wish to use to import metaverse data, give the MIIS account the following permissions:
  - Full Control
    - These rights could probably be tuned down finer in order to restrict group policy and some Exchange attributes, but this setting is the bare minimum for this container

Exchange Server 2003 (or 2000)

Some mail attributes require read permissions from the Exchange 2003 organization. It is a good idea to assign the same MIIS account the following permissions to the Exchange organization:

- Exchange View Only Administrator

Exchange 5.5

In order for MIIS to write to the Exchange directory, we repeat a similar process. As before, an account needs to be created. In our lab, we created a domain account named MIIS and made sure the account was not added to any administrative group.
Then from the Exchange Admin application, give the account the following rights:

- Search Permissions to the Organization
- View Only Admin to the Site you will connect to for directory writes
- Admin to the container the MIIS server will use for directory writes

Note: In some cases, Search permissions will not work against the organizational such as when the DS Site Configuration for the site is incorrect or when the Anonymous account has been disabled or deleted. In these instances, it you may need to give the MIIS account greater access to the organization (this does not automatically give the account permissions to user objects as rights in the org-level are not inherited down to the site and container levels). The Admin role is sufficient in all cases to perform searches against the global address list. Verify that the account does not have permissions to the recipient's container.

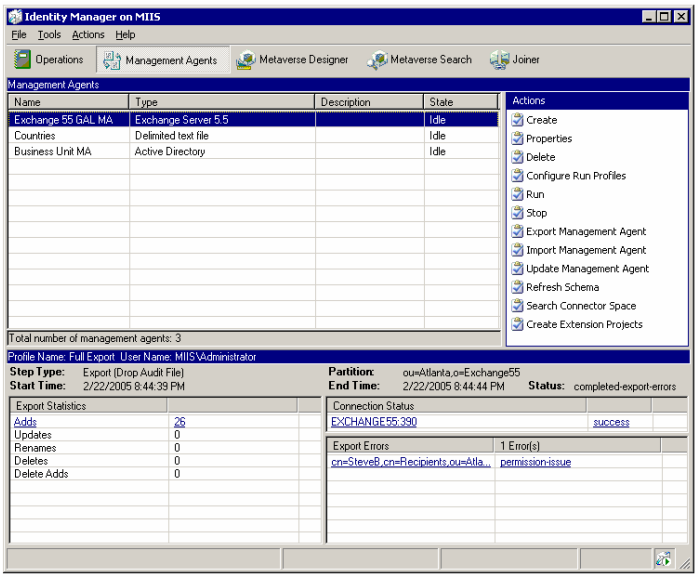
Management Agents

In the initial setup, there are two management agents. One MA connects to the Legacy Exchange 5.5 organization while the other connects to the product CRM Active Directory. As business break out of the Exchange 5.5 organization, they will need their own Management Agent added to initialize the connection and maintain directory Sync. These management agents control attribute flow, connection settings and are called (by name) from the custom provisioning code that has been written for this example. A third management agent has been added for country-code lookups, but it is not fully integrated and optional.

Management Window

The Identity Manager is where most of the work is done for MIIS. It is from this program that we configure the connections, attribute flow, replication schedule and monitor the system for errors and problems. It can be launched from the Start Menu and has several selections at the top of the screen. We will focus on the Management Agents screen for now.

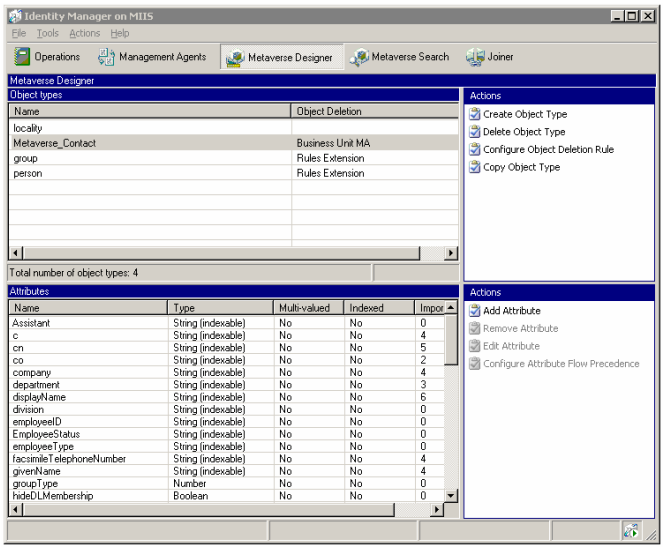
From this screen, we can perform exports of the current configurations (should do this prior to ANY change made to the agent) and can force directory updates. We can also see a quick summary of the last action and what was done. In this example, 25 additions were made to the Exchange 5.5 directory and an error occurred when MIIS tried to write to an object in another container (one where specific rights were not assigned) In our current configuration, MIIS only creates contacts in the target system. This attempt to write to the source object is will need to be addressed with a few lines of additional code in order to further clean the process.



Metaverse Object

Before we get into the management agents, we need to get a better understanding of what we are doing with MIIS and how we plan to do it! We will be importing Mailbox, DL and contact information from Exchange 5.5 and placing that information in a SQL database. This Metaverse will contain a global directory of names imported from Exchange 5.5 as well as names imported from other Active Directory domains. We have defined how these objects will look in the Metaverse by creating a generic MV object called a Metaverse_Contact.

This object type was created from scratch and contains specific fields needed for our immediate needs and some fields we think will be needed later for additional functionality.



Here are the attributes you need to add as well as the characteristics of the attribute:

Assistant	String	(indexable)	Not Multi-valued	
c	String	(indexable)	Not Multi-valued	
cn	String	(indexable)	Not Multi-valued	
co	String	(indexable)	Not Multi-valued	
company	String	(indexable)	Not Multi-valued	
department	String	(indexable)	Not Multi-valued	
displayName	String	(indexable)	Not Multi-valued	
division	String	(indexable)	Not Multi-valued	
employeeID	String	(indexable)	Not Multi-valued	
EmployeeStatus	String	(indexable)	Not Multi-valued	
employeeType	String	(indexable)	Not Multi-valued	
facsimileTelephoneNumber	String	(indexable)	Not Multi-valued	
givenName	String	(indexable)	Not Multi-valued	
groupType	Number		Not Multi-valued	
hideDLMembership	Boolean		Not Multi-valued	
homeMTA	Reference (DN)		Not Multi-valued	
homePhone	String	(indexable)	Not Multi-valued	
info	String	(indexable)	Not Multi-valued	
initials	String	(indexable)	Not Multi-valued	
l	String	(indexable)	Not Multi-valued	
legacyExchangeDN	String	(indexable)	Multi-valued	Indexed
locality	String	(indexable)	Not Multi-valued	Indexed
mail	String	(indexable)	Not Multi-valued	Indexed
mailNickname	String	(indexable)	Not Multi-valued	
manager	Reference (DN)		Not Multi-valued	
MapiRecipient	Boolean		Not Multi-valued	
mobile	String	(indexable)	Not Multi-valued	

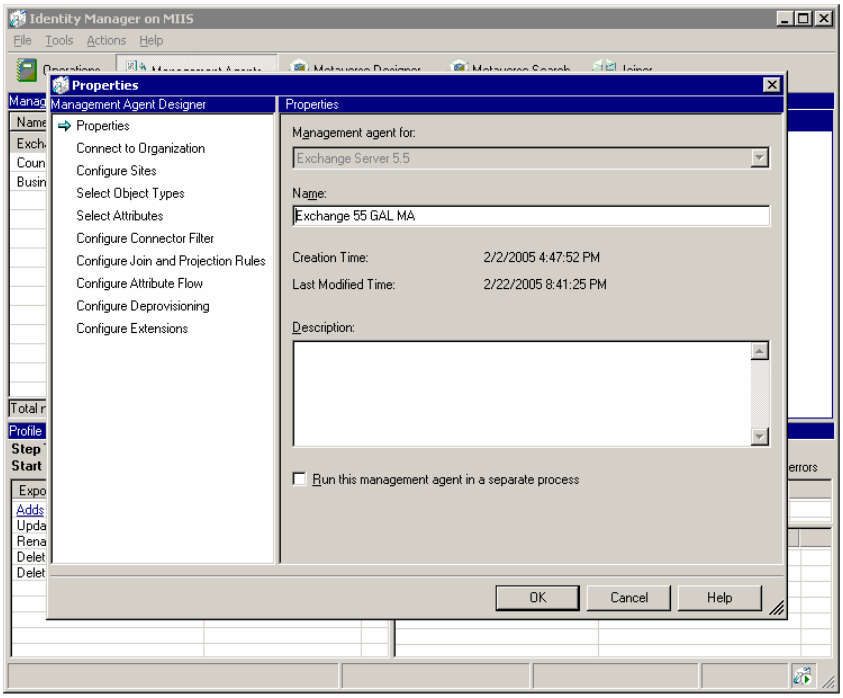
msExchAssistantName	String	(indexable)	Not Multi-valued	
msExchExpansionServerName	String	(non-indexable)	Not Multi-valued	
msExchHideFromAddressLists	Boolean		Not Multi-valued	
msExchHomeServerName	String	(indexable)	Not Multi-valued	
msExchMasterAccountSid	Binary	(indexable)	Not Multi-valued	
msExchOriginatingForest	String	(indexable)	Not Multi-valued	
nTGroupMembers	Binary	(indexable)	Multi-valued	Indexed
o	String	(indexable)	Not Multi-valued	
otherHomePhone	String	(indexable)	Multi-valued	Indexed
otherTelephone	String	(indexable)	Multi-valued	Indexed
pager	String	(indexable)	Not Multi-valued	
physicalDeliveryOfficeName	String	(indexable)	Not Multi-valued	
postalCode	String	(indexable)	Not Multi-valued	
proxyAddresses	String	(indexable)	Multi-valued	Indexed
Rfc822Mailbox	String	(indexable)	Not Multi-valued	
sn	String	(indexable)	Not Multi-valued	
st	String	(indexable)	Not Multi-valued	
streetaddress	String	(indexable)	Not Multi-valued	
targetAddress	String	(indexable)	Not Multi-valued	Indexed
telephoneAssistant	String	(indexable)	Not Multi-valued	
telephoneNumber	String	(indexable)	Not Multi-valued	
textEncodedORAddress	String	(indexable)	Not Multi-valued	
title	String	(indexable)	Not Multi-valued	
uid	String	(non-indexable)	Not Multi-valued	

It is possible to clean up these entries in order to further trim down unnecessary attributes. In some instances, you may want to create additional attributes for specific tasks for example. You could create an attribute to denote whether an item should be written to the target directory and perform a filter against that field. In addition, you could create “work” fields to hold any type of data you wish to key upon in code.

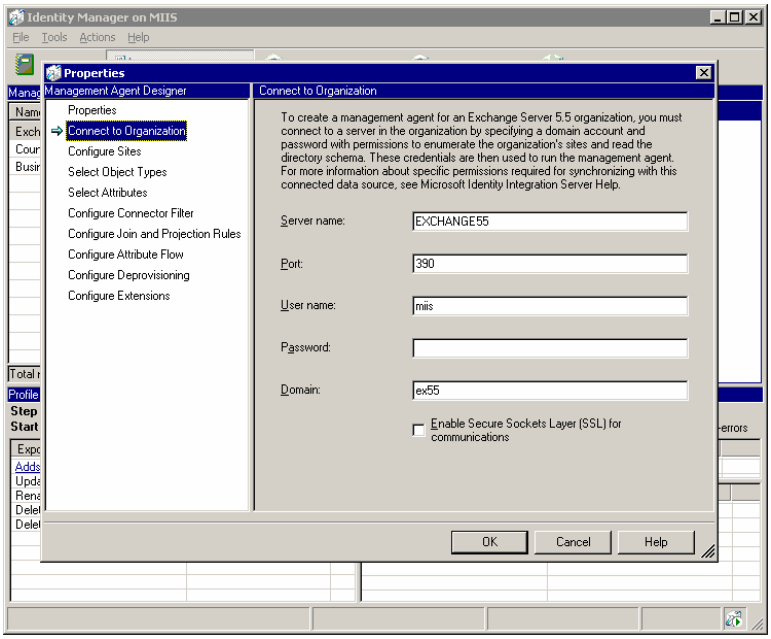
Exchange 55 GAL MA

OK, so now that we know the Metaverse (MV) object we are going to target, we can take a deeper look at the agents that collect and distribute the data. Each MA contains connection information to a specific directory. As a result, each MA is also known as a Collector Space (CS). To view the settings on the Exchange 55 GAL MA, we need only to double click the agent from the main agent screen.

This page shows us the base Management Agent (Exchange Server 5.5) and the name we have provided to identify the Agent. This name is used within the source code to identify the Connector Space.



Connection to Organization

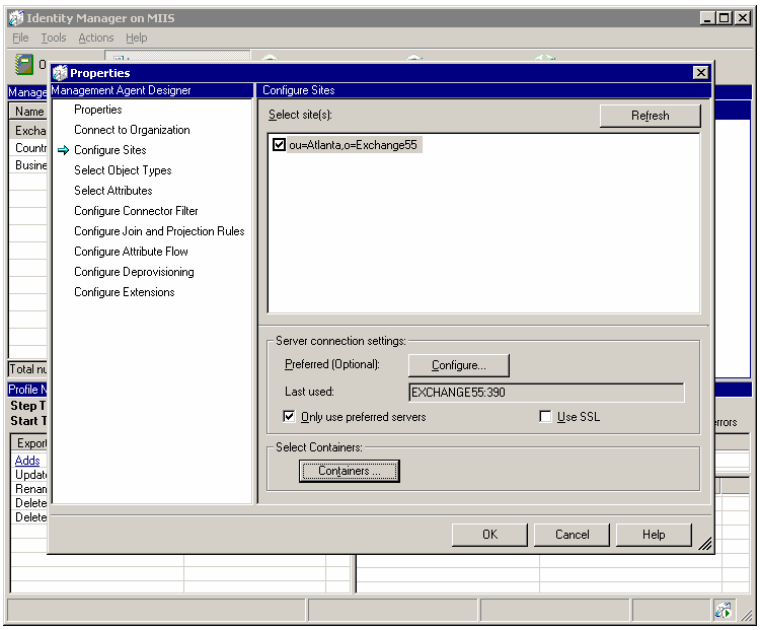


In this page, we identify the Exchange 5.5 server we will connect to as well as the credentials we will use. You can see that we have specified the MIIS user account in order to restrict our permissions to the Exchange environment. Moreover, you should notice that we are not using the standard port of 389 for our LDAP connections. This server is also a Windows 2000 Server running Domain Controller services so we had to change the Exchange 5.5 port for LDAP to port 390.

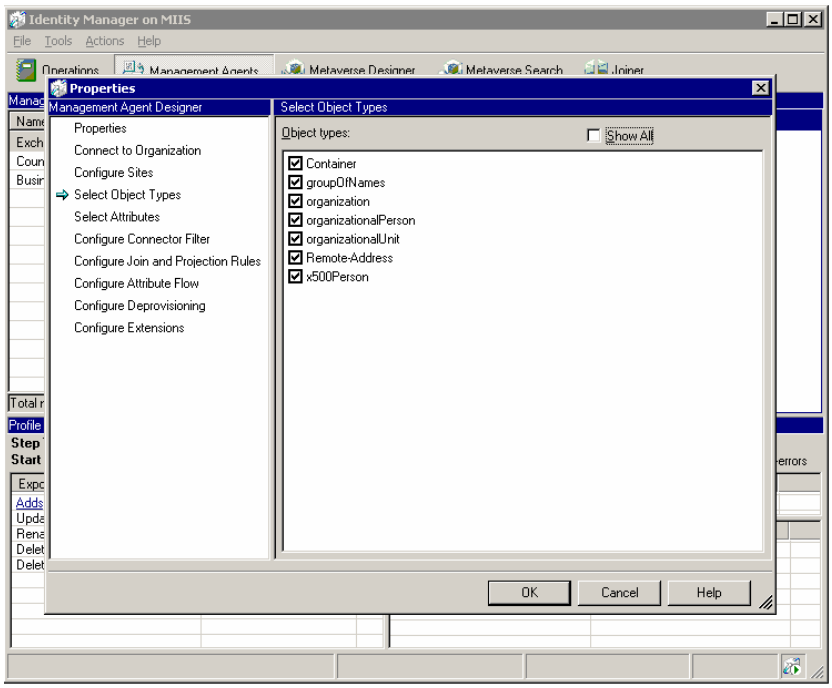
Configure Sites

It is from this screen that we identify the sites that contain the objects we want to synchronize (copy to) with the AD. From this screen, we can identify any other servers we want to target (not required for this installation) as well as the containers we want to choose.

Note: It is important that you select both the source containers and target containers from the Containers window. Deselected containers will not be used for either reading or writing with MIIS.



Select Object Types

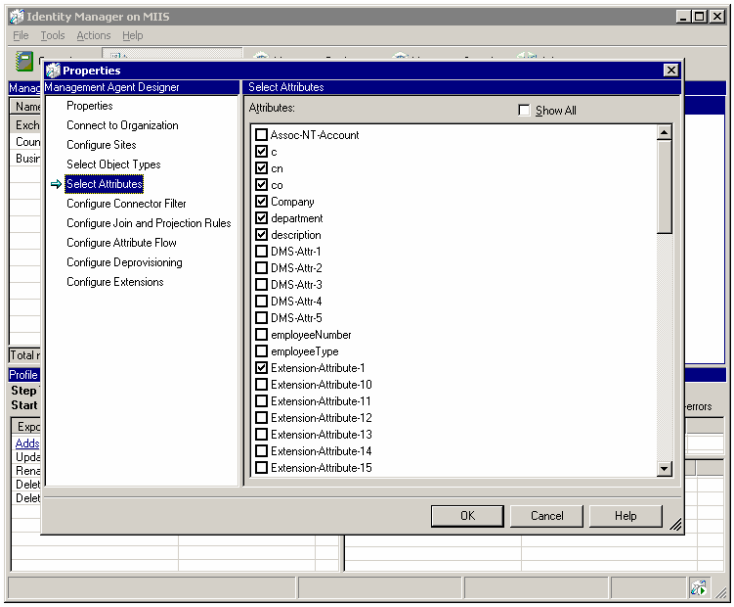


We need to expose several different types of information from the Exchange 5.5 environment. You will probably need to use the Show All checkbox to select each of these object types.

Select Attributes

In our case, we are selecting a few more attributes than required per our list of requirements, but the additional fields we have identified will allow us to better connect the directories now and for later projects.

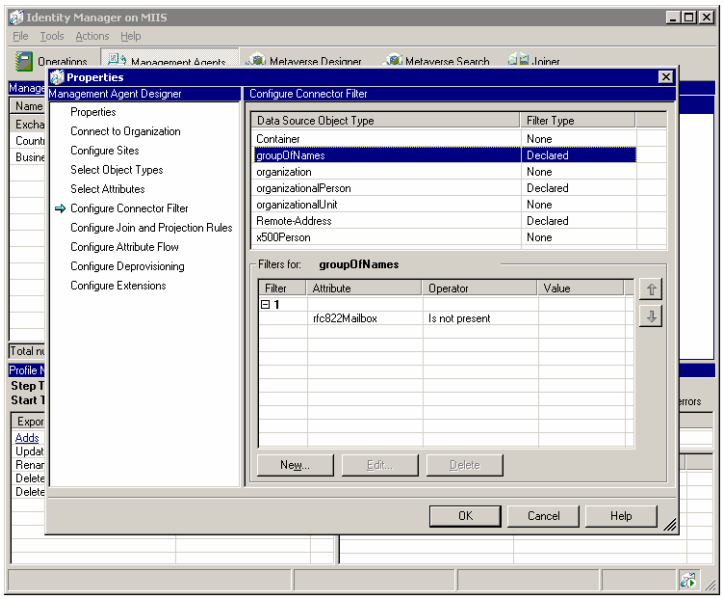
You will certainly need to select the Show All checkbox to get to all of these attributes:



C
Cn
Co
Company
Department
Description
Extension-Attribute-1
facsimileTelephoneNumber
giveName
Hide-From-Address-Book
Initials
L
Mail
MAPI-Recipient
Mobile
otherMailbox
pager
physicalDeliveryOfficeName
postalAddress
postalCode
Proxy-Addresses
Rdn
Report-To-Originator
Report-To-Owner
Rfc822Mailbox
Sn
St
Street
Target-Address
telephoneNumber
textEncodedORaddress
title
uid

Configure Connector Filter

The connector filter is used to specify things we do not want to replicate. In our environment, we have chosen not to replicate anything that does not have an Internet Address. We could also choose not to replicate hidden objects or objects with certain key words in the Custom Attribute field or similar. The object types you see listed in this window are specific to Exchange 5.5.



groupOfNames –distribution list
organizationalPerson -mailbox
Remote-Address –custom recipient

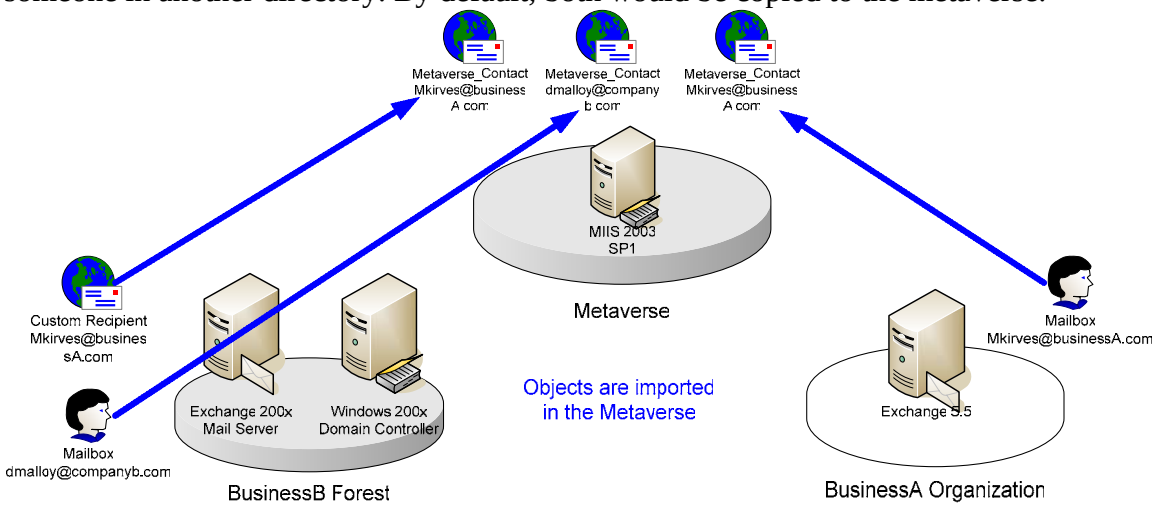
These are the only objects you really need to focus on when building filters in this Exchange management agent.

Configure Join and Projection Rules

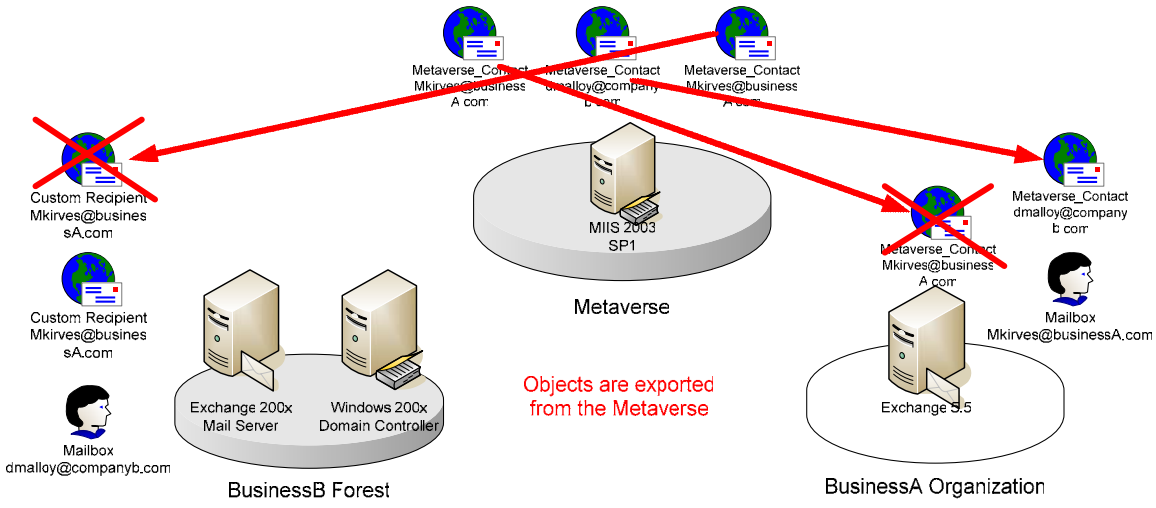
This is where things can get a little confusing with MIIS. It is important that each person in your organization only exist in the directories once. This sounds pretty basic, but if there are more than one of you, then mail flow could break. The problem is that in many cases, a company has a contact or custom recipient in one system that "points" to a mailbox in another system. Because of that, you may be in two directories at once. Now, if we should try to combine those directories, we need to make sure there is some logic that does not allow duplicate entries to be created.

Join Rules

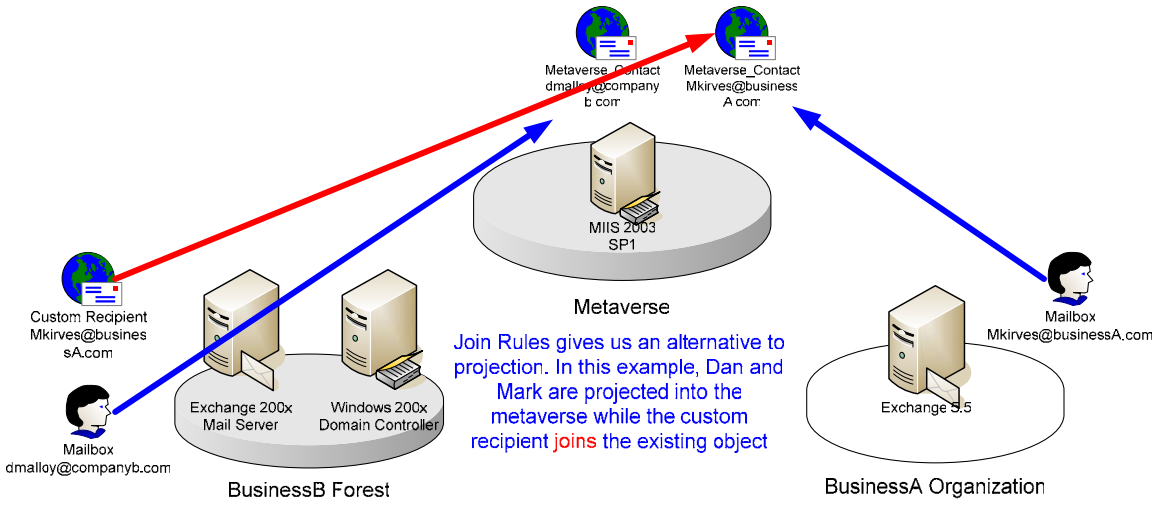
The purpose of a join rules is to try to match objects before creating them. For example, let us assume that someone in one directory has created n custom contact or recipient for someone in another directory. By default, both would be copied to the metaverse.



This part in of itself does not create a problem. It is when that objects are written back that we have issues as duplicate items will be written which will certainly cause a disruption in mail flow within that system.



To prevent this from happening, we use join rules to “connect” duplicates in the metaverse.



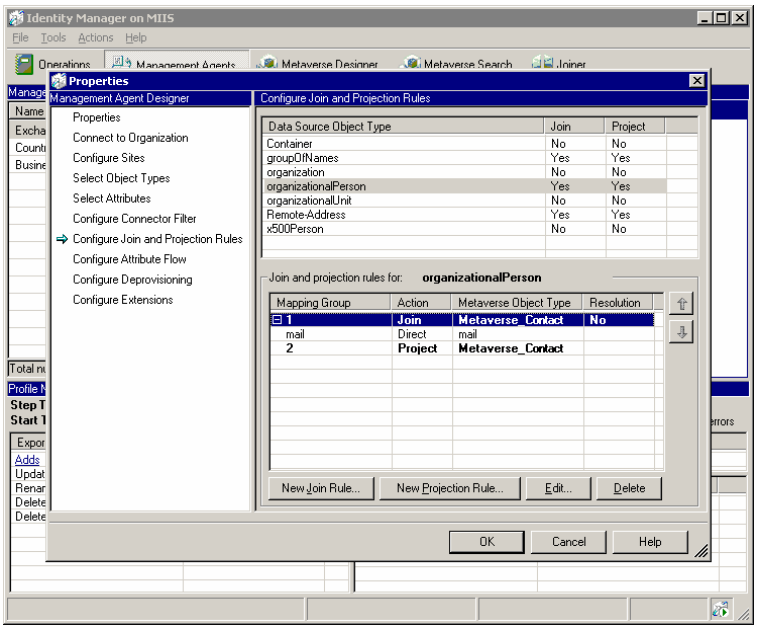
You can create very specific join rules to follow your business logic For example, you may have items in an HR system that should take precedent over other source systems in which case you could join based on a specific field type or format.

In this example, we are joining objects if the mail attribute is matched. In other words, if an object using the mail attribute: mkirves@businessa.com is in the metaverse, another object using the same mail attribute will “join” the existing one. This prevents the mail address from being used more than once in the environment. In fact, we are using two rules for each object type:

- Join if mail attribute is matched
  - If a contact, group or user is imported into the metaverse with a specific mail attribute, no other other object may overwrite it. This means that user objects may “join” existing contact objects if the custom recipient already exists for that user. The drawback to this example would be that the entry could have incomplete directory information and that the real mailbox would not be authorative for the object.
- Project is mail attribute is not matched
  - If the RFC822mailbox field is populated and the there is no match for the mail attribute, then the object will be automatically projected into the Metaverse

Configuring the basic join rules I have described is a fairly easy task and should take no longer than a few short minutes to apply. Open the **Configure Join and Projection Rules** selection from the Exchange 55 GAL MA agent.

You will need to add two rules to each of the object types you wish to synchronize. For this example, we will add the following:



groupOfNames

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

organizationalPerson

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

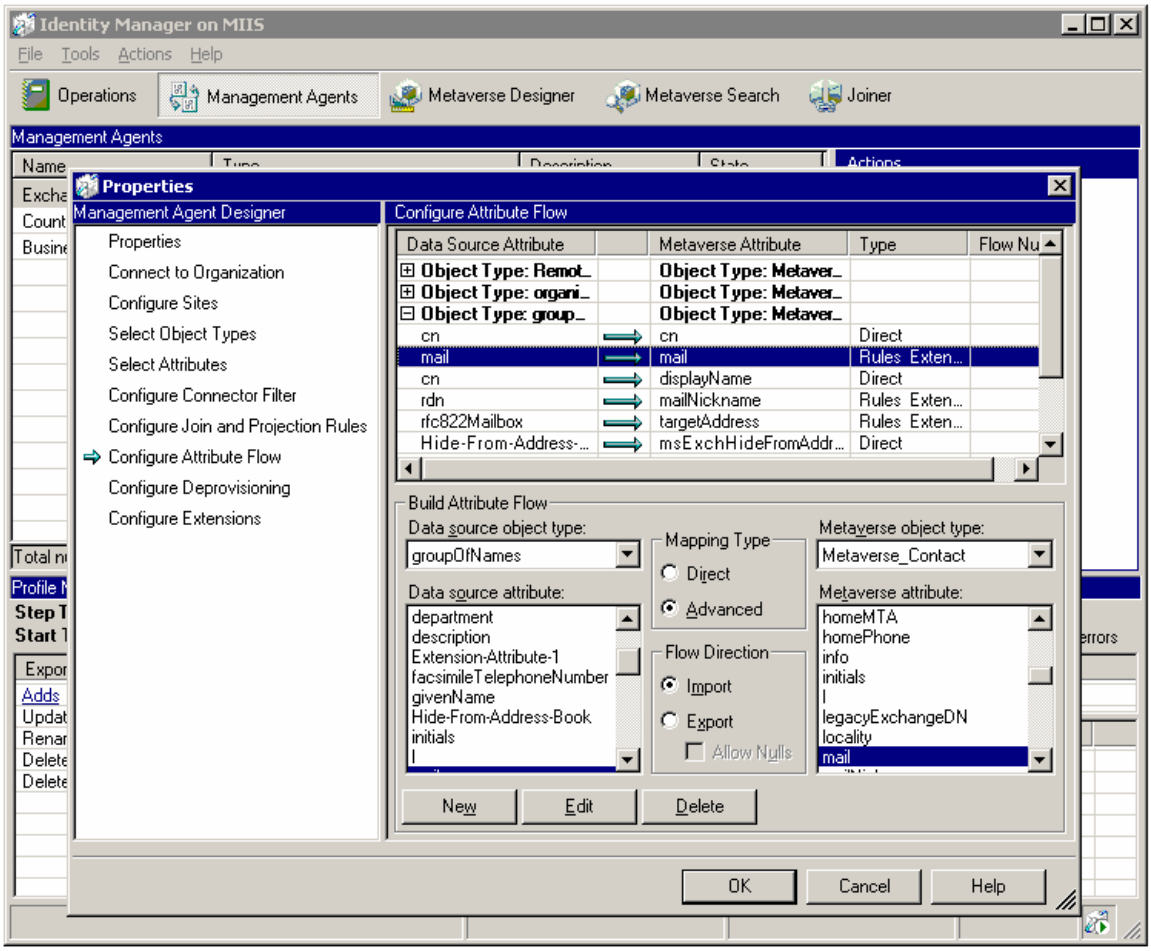
Remote-Address

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

You should notice that the projection rules follow the join rules. You can create multiple join rules for each object type and you can specify custom rules to handle joins. For our purposes, the standard mail mapping fields will suffice.

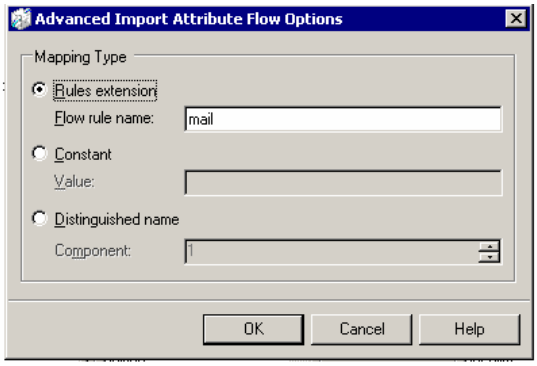
Configure Attribute Flow

This step will likely take the most time and you should pay close attention to the detail. As we map fields from Exchange to the metaverse and visa-versa we have to pay close attention to the flow direction as well as any advanced rules we need to apply. As you will find out, standard mapping rules will allow us to meet our objectives. From the Configure Attribute Flow screen, you can see that we are configuring flow for all three object types we wish to synchronize.



To make things simple for this installation, we are importing each of the Exchange settings into a single object type called Metaverse_contact. Moreover, we are only exporting objects into the Exchange environment as Remote-Address (custom recipients) and only into a specific, pre-determined container. When specifying attribute flow, you choose the source, the target and the fields you wish to map. In addition, you can specify any advanced mapping type you wish to use. When using the advanced features, we can select more than one object type as a source and the source code we wish to use to handle the business logic. Let's go over an example. You should notice that the mail field highlighted in the picture shows an Advanced mapping rule is in place. If we click Edit, we can see which rule is being identified.

From the advanced flow options window, we can see that a specific flow rule named “mail” is identified. This tells MIIS that when performing an import on this object, it should reference the extension DLL identified in this MA (we will get to that later in the document) and that the logic in that code should be applied to this flow.



Inside our extension DLL, we have a sub called:

```
  Public Sub MapAttributesForImport(ByVal FlowRuleName As String, ByVal
csentry As Microsoft.MetadirectoryServices.CSEntry, ByVal mventry As
Microsoft.MetadirectoryServices.MVEntry) Implements
Microsoft.MetadirectoryServices.IMASynchronization.MapAttributesForImpo
rt
```

Within this sub, there are several import rules defined. This is the one named mail:

```
Case "mail"

                If csentry(FlowRuleName).IsPresent Then
                    Dim work As String = csentry(FlowRuleName).Value
                    work = Replace(work, "/", "")
                    work = Replace(work, "\", "")
                    mventry(FlowRuleName).Value = work

                End If
```

The purpose of this code is to cleanout illegal characters such as the “/” and “\” from the mail address (SMTP Address). An illegal address of Atlanta/sales@companya.com would be changed to atlantasales@companya.com. While the illegal characters would remain in the Exchange system, they would be clean in the metaverse and then legal to push to AD.

Export Attribute Flow (Send to Exchange from Metaverse)

Here are the current flow settings for the Exchange MA. You can see from this table, that we did not require specialized code to manipulate the data. Instead, we elected to clean the information during the import process so that data in the metaverse is “normalized”.

Export Attribute Flow				
Data Source Attribute	**To**	**Metaverse Attribute**	**Mapping Type**	**Allow Nulls**
Remote-Address	-	**Metaverse_Contact**	-	-
cn	<—	cn	Direct	-
givenName	<—	givenName	Direct	Allow
mail	<—	mail	Direct	-
uid	<—	uid	Direct	Allow
MAPI-Recipient	<—	MapiRecipient	Direct	Allow
otherMailbox	<—	proxyAddresses	Direct	-
sn	<—	sn	Direct	Allow

department	<—	department	Direct	Allow
Company	<—	company	Direct	Allow
initials	<—	initials	Direct	Allow
pager	<—	pager	Direct	Allow
title	<—	title	Direct	Allow
st	<—	st	Direct	Allow
postalCode	<—	postalCode	Direct	Allow
postalAddress	<—	streetAddress	Direct	Allow
l	<—	l	Direct	Allow
facsimileTelephoneNumber	<—	facsimileTelephoneNumber	Direct	Allow
mobile	<—	mobile	Direct	Allow
telephoneNumber	<—	telephoneNumber	Direct	Allow
physicalDeliveryOfficeName	<—	physicalDeliveryOfficeName	Direct	Allow
rdn	<—	uid	Direct	-
co	<—	c	Direct	Allow
Target-Address	<—	targetAddress	Direct	Allow

Import Attribute Flow (Send to Metaverse from Exchange)

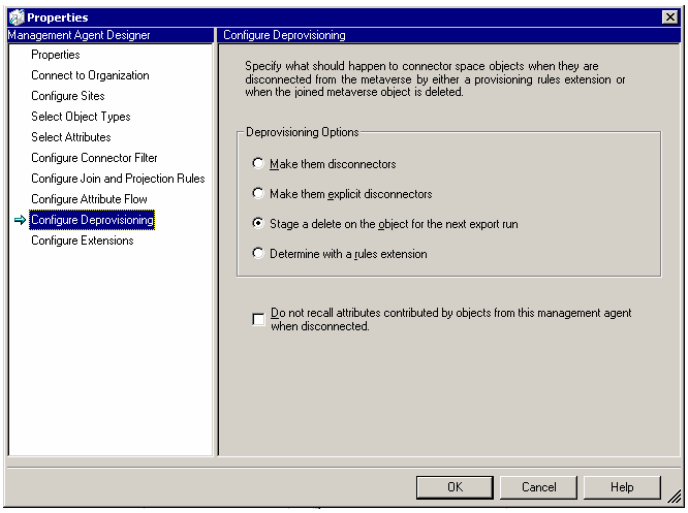
Importing Exchange information into the metaverse required much more work in that we needed to reformat, clean and construct new fields based on information collected from other fields. In order to correctly populate the proxyaddresses attribute, we had to contruct the field based on three Exchange 5.5 fields; otherMailbox, rfc822Mailbox and textEncodedORaddress. Custom code was created to combine these strings into a collection in order to populate the multi-value field in the MV. To see the source code for the specific rules extensions, refer to the code section at the end of this document.

Import Attribute Flow				
Data Source Attribute	**To**	**Metaverse Attribute**	**Mapping Type**	**Precedence Order**
organizationalPerson	-	**Metaverse_Contact**	-	-
cn	—>	cn	Direct	2
groupOfNames	-	**Metaverse_Contact**	-	-
cn	—>	cn	Direct	3
Remote-Address	-	**Metaverse_Contact**	-	-
cn	—>	cn	Direct	4
organizationalPerson	-	**Metaverse_Contact**	-	-
Company	—>	company	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
Company	—>	company	Direct	3
department	—>	department	Direct	2
mail	—>	mail	Rules Extension - mail	1
groupOfNames	-	**Metaverse_Contact**	-	-
mail	—>	mail	Rules Extension - mail	2
organizationalPerson	-	**Metaverse_Contact**	-	-
mail	—>	mail	Direct	3
co	—>	co	Direct	2
givenName	—>	givenName	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
givenName	—>	givenName	Direct	3
initials	—>	initials	Direct	2

organizationalPerson	-	**Metaverse_Contact**	-	-
mobile	—>	mobile	Direct	2
title	—>	title	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
title	—>	title	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
cn	—>	displayName	Direct	3
groupOfNames	-	**Metaverse_Contact**	-	-
cn	—>	displayName	Direct	4
Remote-Address	-	**Metaverse_Contact**	-	-
cn	—>	displayName	Direct	6
organizationalPerson	-	**Metaverse_Contact**	-	-
sn	—>	sn	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
sn	—>	sn	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
facsimileTelephoneNumber	—>	facsimileTelephoneNumber	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
facsimileTelephoneNumber	—>	facsimileTelephoneNumber	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
rdn	—>	mailNickname	Direct	3
Remote-Address	-	**Metaverse_Contact**	-	-
rdn	—>	mailNickname	Direct	5
groupOfNames	-	**Metaverse_Contact**	-	-
rdn	—>	mailNickname	Rules Extension - mailNickname	6
organizationalPerson	-	**Metaverse_Contact**	-	-
pager	—>	pager	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
pager	—>	pager	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
postalCode	—>	postalCode	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
postalCode	—>	postalCode	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
st	—>	st	Direct	2
rfc822Mailbox	—>	targetAddress	Rules Extension - targetAddress	1
groupOfNames	-	**Metaverse_Contact**	-	-
rfc822Mailbox	—>	targetAddress	Rules Extension - targetAddress	2
Remote-Address	-	**Metaverse_Contact**	-	-
rfc822Mailbox	—>	targetAddress	Rules Extension - targetAddress	3
organizationalPerson	-	**Metaverse_Contact**	-	-
Hide-From-Address-Book	—>	msExchHideFromAddressLists	Direct	2
groupOfNames	-	**Metaverse_Contact**	-	-
Hide-From-Address-Book	—>	msExchHideFromAddressLists	Direct	4
Remote-Address	-	**Metaverse_Contact**	-	-
Hide-From-Address-Book	—>	msExchHideFromAddressLists	Direct	6
organizationalPerson	-	**Metaverse_Contact**	-	-
postalAddress	—>	streetAddress	Direct	4
Remote-Address	-	**Metaverse_Contact**	-	-
postalAddress	—>	streetAddress	Direct	5
organizationalPerson	-	**Metaverse_Contact**	-	-
telephoneNumber	—>	telephoneNumber	Direct	1

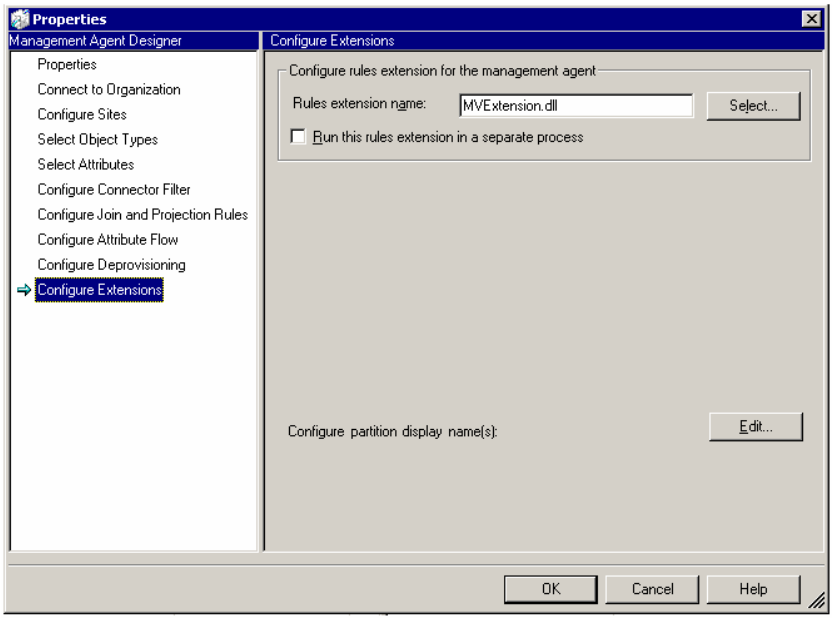
Remote-Address	-	**Metaverse_Contact**	-	-
telephoneNumber	—>	telephoneNumber	Direct	3
uid	—>	uid	Direct	3
rdn	—>	uid	Direct	4
groupOfNames	-	**Metaverse_Contact**	-	-
uid	—>	uid	Rules Extension - uid	6
Remote-Address	-	**Metaverse_Contact**	-	-
otherMailbox,rfc822Mailbox,textEncodedORaddress	—>	proxyAddresses	Rules Extension - proxyaddresses	1
organizationalPerson	-	**Metaverse_Contact**	-	-
otherMailbox,rfc822Mailbox,textEncodedORaddress	—>	proxyAddresses	Rules Extension - proxyaddresses	2
groupOfNames	-	**Metaverse_Contact**	-	-
otherMailbox,rfc822Mailbox,textEncodedORaddress	—>	proxyAddresses	Rules Extension - proxyaddresses	3
textEncodedORaddress	—>	textEncodedORAddress	Direct	2
Remote-Address	-	**Metaverse_Contact**	-	-
textEncodedORaddress	—>	textEncodedORAddress	Direct	5
organizationalPerson	-	**Metaverse_Contact**	-	-
MAPI-Recipient	—>	MapiRecipient	Direct	1
Remote-Address	-	**Metaverse_Contact**	-	-
MAPI-Recipient	—>	MapiRecipient	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
l	—>	l	Direct	1
Remote-Address	-	**Metaverse_Contact**	-	-
l	—>	l	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
physicalDeliveryOfficeName	—>	physicalDeliveryOfficeName	Direct	1
Remote-Address	-	**Metaverse_Contact**	-	-
physicalDeliveryOfficeName	—>	physicalDeliveryOfficeName	Direct	3
organizationalPerson	-	**Metaverse_Contact**	-	-
co	—>	c	Direct	3
Remote-Address	-	**Metaverse_Contact**	-	-
co	—>	c	Direct	4
rfc822Mailbox	—>	Rfc822Mailbox	Rules Extension - Rfc822Mailbox	1
groupOfNames	-	**Metaverse_Contact**	-	-
rfc822Mailbox	—>	Rfc822Mailbox	Rules Extension - Rfc822Mailbox	2

Configure Deprovisioning



When a source item is deleted, a process called Deprovisioning is used to remove the object from the metaverse and from the target directories. As you can imagine, considerable time and energy can be spent in developing the appropriate business logic to follow when deprovisioning items.

At this time, we did not create a specialized rules extension to control how items are removed. We feel that the standard deletion on export rule can be used to remove the target entry is the source is deleted. Because of the granular rights assignment and specific containers used for exports, we do not believe additional coding will be required. If additional coding work is required, this is the screen that you would use to specify the change.

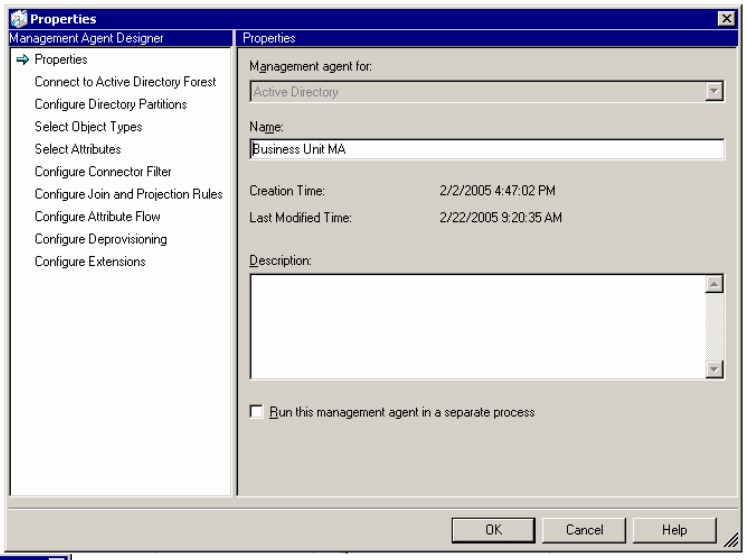


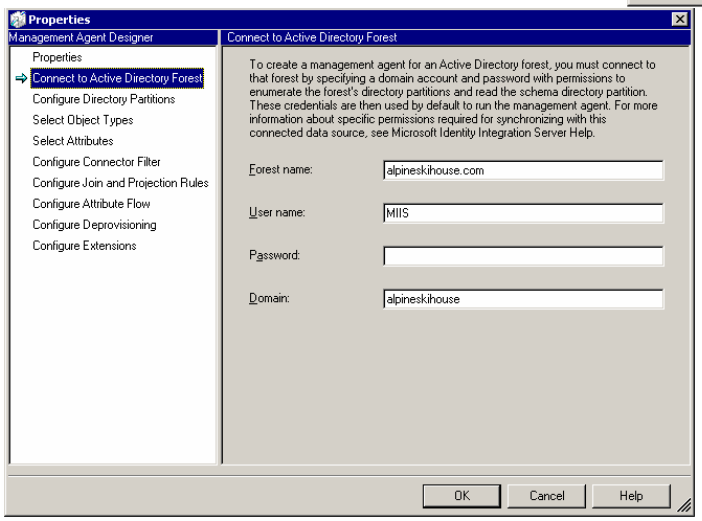
Configure Extensions

This last configuration screen is used to specify which DLL is used to control the logic for this MA. Currently, there is a single extension DLL used for this MIIS environment. While this basic design works, it is not considered best-practice. Each MA should use its own extension.DLL when possible. To learn more about this concept, see the Developer Reference help file on the MIIS server or search for “Creating a Rules Extension from Multiple Sources.”

Business Unit MA

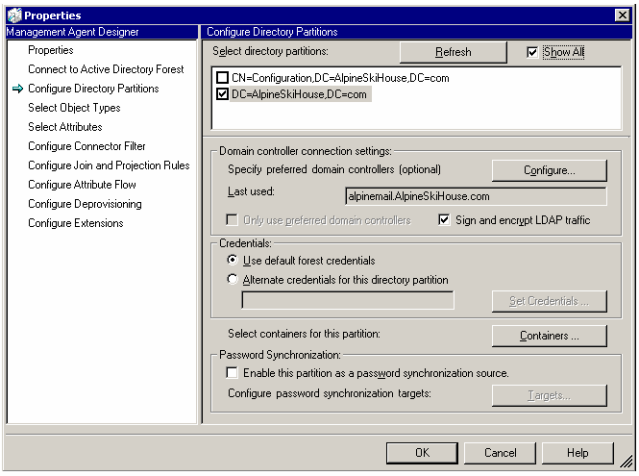
Just as with the Exchange MA, this page shows us the base Management Agent (Active Directory) and the name we have provided to identify the Agent. This name is used within the source code to identify the Connector Space.





Connection to Active Directory Forest

In this page, we identify the Active Directory Forest we will connect to as well as the credentials we will use. You can see that we have specified the MIIS user account in order to restrict our permissions to the AD environment.

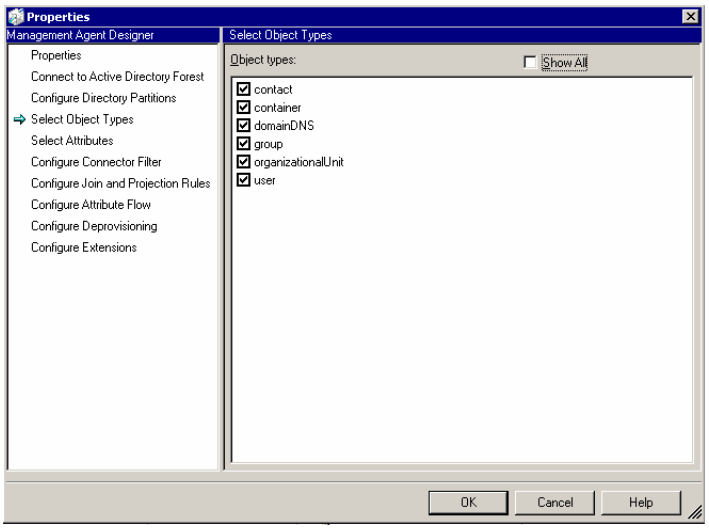


Configure Directory Partitions

It is from this screen that we identify the directory partitions that contain the objects we want to synchronize (copy to) with the Metaverse. From this screen, we can identify any other credentials we want to use (for other partitions for example) as well as the containers we want to choose.

Note: It is important that you select both the source containers and target containers from the Containers window. Deselected containers will not be used for either reading or writing with MIIS.

Select Object Types

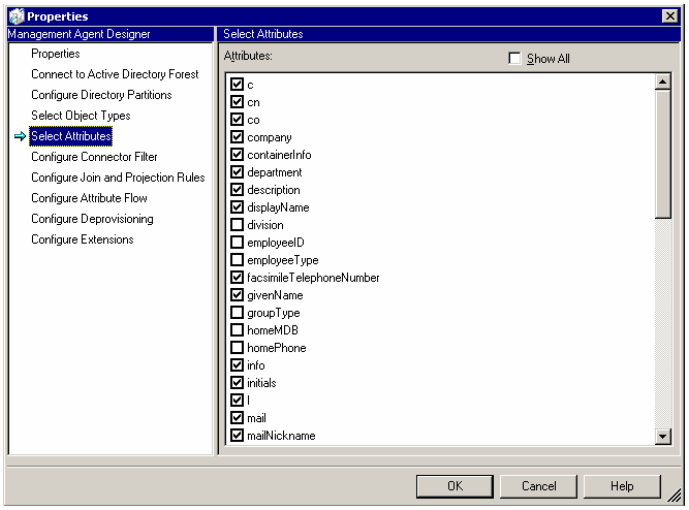


We need to expose several different types of information from the Active Directory environment. You will probably need to use the Show All checkbox to select each of these object types.

Select Attributes

In our case, we are selecting a few more attributes than required per our list of requirements, but the additional fields we have identified will allow us to better connect the directories now and for later projects.

You will certainly need to select the Show All checkbox to get to all of these attributes:



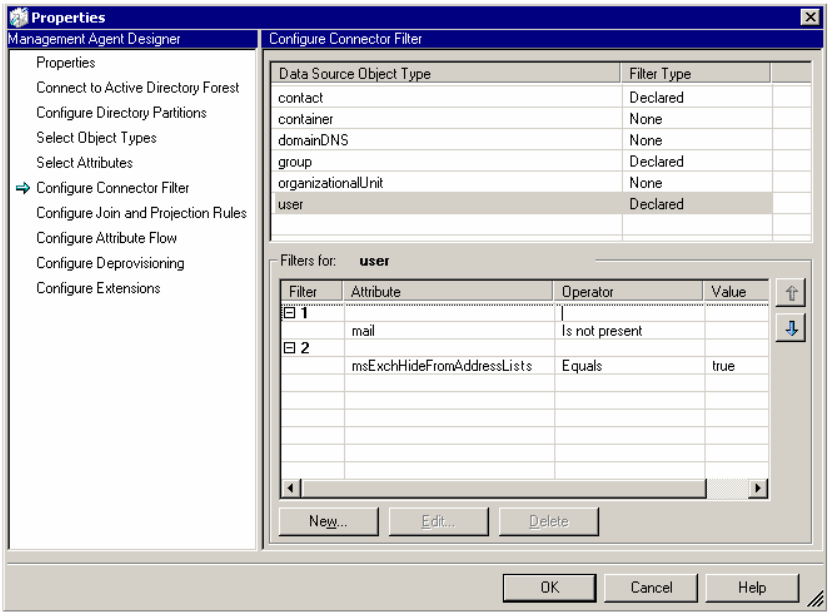
c
cn
co
company
containerInfo
department
description
displayName
facsimileTelephoneNumber
givenName
info
initials
l
mail
mailNickname
mAPIRecipient
middleName
mobile
msExchHideFromAddressLists
msExchPoliciesExcluded
name
nTSecurityDescriptor
o
otherTelephone
ou
pager
physicalDeliveryOfficeName
postalAddress
postalCode
proxyAddresses
sAMAccountName
sn
st
street
streetAddress
targetAddress
telephoneNumber
textEncodedORAddress
title

Configure Connector Filter

The connector filter is used to specify things we do not want to replicate. In our environment, we have chosen not to replicate anything that does not have mail address. We have also chosen not to replicate hidden objects. We could have just as easily filtered based on certain key words in the Custom Attribute field or similar. The object types you see listed in this window are specific to Active Directory.

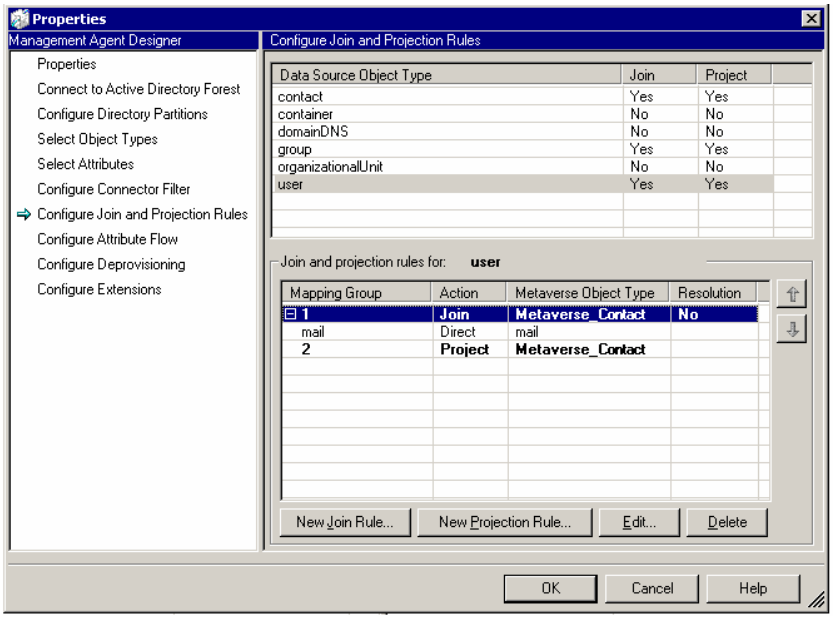
group –distribution list
user-mailbox
contact –custom recipient

These are the only objects you really need to focus on when building filters in this AD management agent.



Configure Join and Projection Rules

You will need to add two rules to each of the object types you wish to synchronize. In this example, we added the following:



contact

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

Group

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

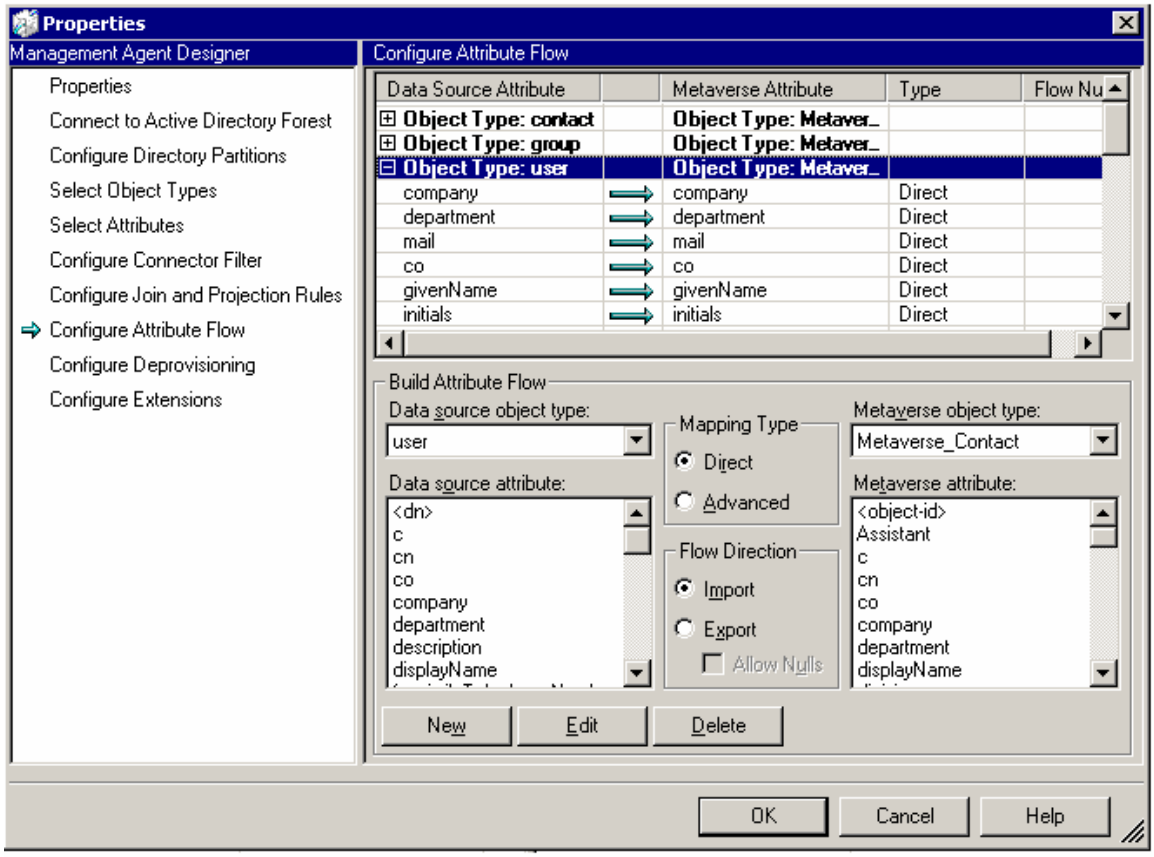
user

- Declared Projection Rule based on the Metaverse_Contact object type.
- Direct Join rule based on the mail field between the “mail” data source attribute and the “mail” attribute on “Metaverse_Contact.

You should notice that the projection rules follow the join rules. You can create multiple join rules for each object type and you can specify custom rules to handle joins. For our purposes, the standard mail mapping fields will suffice.

Configure Attribute Flow

This step will likely take the most time and you should pay close attention to the detail. As we map fields from AD to the metaverse and visa-versa we have to pay close attention to the flow direction as well as any advanced rules we need to apply. As you will find out, standard mapping rules will allow us to meet our objectives. From the Configure Attribute Flow screen, you can see that we are configuring flow for all three object types we wish to synchronize.



Export Attribute Flow (Send to AD from Metaverse)

Here are the current flow settings for the AD MA. You can see from this table, that we required some specialized code to manipulate the data. We are forcing an entry to the msExchPoliciesExcluded field which well disable RUS manipulation of our entry. We are also formatting the country codes.

Export Attribute Flow				
Data Source Attribute	**To**	**Metaverse Attribute**	**Mapping Type**	**Allow Nulls**
contact	-	**Metaverse_Contact**	-	-
cn	<—	cn	Direct	-
company	<—	company	Direct	-
department	<—	department	Direct	-
displayName	<—	displayName	Direct	-
facsimileTelephoneNumber	<—	facsimileTelephoneNumber	Direct	-

givenName	<—	givenName	Direct	-
initials	<—	initials	Direct	-
mail	<—	mail	Direct	-
mailNickname	<—	mailNickname	Direct	-
mobile	<—	mobile	Direct	-
pager	<—	pager	Direct	-
postalCode	<—	postalCode	Direct	-
sn	<—	sn	Direct	-
streetAddress	<—	streetAddress	Direct	-
st	<—	st	Direct	-
telephoneNumber	<—	telephoneNumber	Direct	-
targetAddress	<—	targetAddress	Direct	-
title	<—	title	Direct	-
textEncodedORAddress	<—	textEncodedORAddress	Direct	-
mAPIRecipient	<—	MapiRecipient	Direct	-
l	<—	l	Direct	-
msExchHideFromAddressLists	<—	msExchHideFromAddressLists	Direct	Allow
o	<—	o	Direct	-
postalAddress	<—	streetAddress	Direct	-
physicalDeliveryOfficeName	<—	physicalDeliveryOfficeName	Direct	-
c	<—	c	Rules Extension - cd.contact:c<- mv.Metaverse_Contact:c	Allow
msExchPoliciesExcluded	<—	-	Constant - {26491CFC-9E50-4857-861B-0CB8DF22B5D7}	-
proxyAddresses	<—	proxyAddresses	Direct	-

Import Attribute Flow (Send to Metaverse from AD)

Is with Exchange, importing AD information into the metaverse requirs much more work in that we needed to reformat, clean and construct new fields based on information collected from other fields. While there are many mapping rules here, only three custom mapping rules were required as AD synchronizes with the MV much easier than Exchange 5.5.

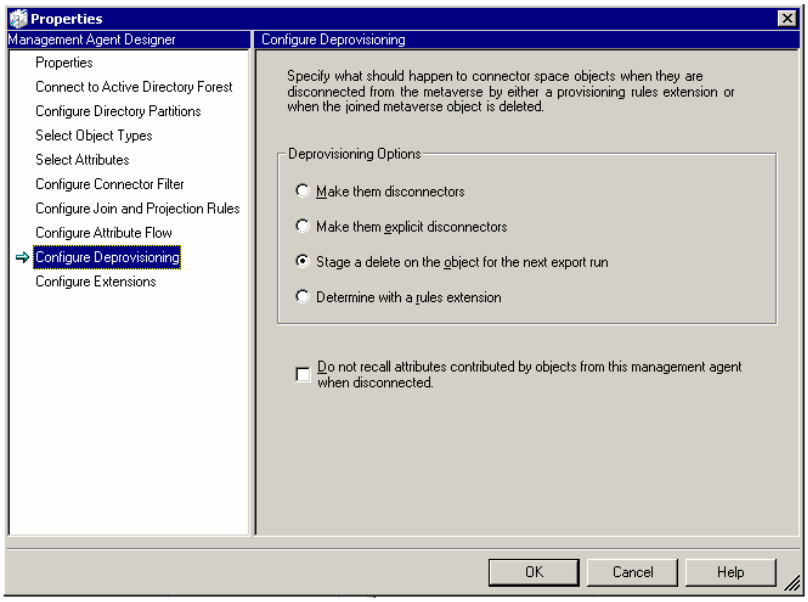
Import Attribute Flow				
Data Source Attribute	**To**	**Metaverse Attribute**	**Mapping Type**	**Precedence Order**
group	-	**Metaverse_Contact**	-	-
cn	—>	cn	Direct	1
contact	-	**Metaverse_Contact**	-	-
cn	—>	cn	Direct	5
user	-	**Metaverse_Contact**	-	-
company	—>	company	Direct	1
contact	-	**Metaverse_Contact**	-	-
company	—>	company	Direct	4
user	-	**Metaverse_Contact**	-	-
department	—>	department	Direct	1
contact	-	**Metaverse_Contact**	-	-
department	—>	department	Direct	3
mail	—>	mail	Direct	4
user	-	**Metaverse_Contact**	-	-
mail	—>	mail	Direct	5
group	-	**Metaverse_Contact**	-	-
mail	—>	mail	Direct	6

user	-	**Metaverse_Contact**	-	-
co	—>	co	Direct	1
givenName	—>	givenName	Direct	1
contact	-	**Metaverse_Contact**	-	-
givenName	—>	givenName	Direct	4
user	-	**Metaverse_Contact**	-	-
initials	—>	initials	Direct	1
contact	-	**Metaverse_Contact**	-	-
initials	—>	initials	Direct	3
user	-	**Metaverse_Contact**	-	-
mobile	—>	mobile	Direct	1
contact	-	**Metaverse_Contact**	-	-
mobile	—>	mobile	Direct	3
user	-	**Metaverse_Contact**	-	-
title	—>	title	Direct	1
contact	-	**Metaverse_Contact**	-	-
title	—>	title	Direct	4
user	-	**Metaverse_Contact**	-	-
displayName	—>	displayName	Direct	1
group	-	**Metaverse_Contact**	-	-
displayName	—>	displayName	Direct	2
contact	-	**Metaverse_Contact**	-	-
displayName	—>	displayName	Direct	5
user	-	**Metaverse_Contact**	-	-
sn	—>	sn	Direct	1
contact	-	**Metaverse_Contact**	-	-
sn	—>	sn	Direct	4
user	-	**Metaverse_Contact**	-	-
facsimileTelephoneNumber	—>	facsimileTelephoneNumber	Direct	1
contact	-	**Metaverse_Contact**	-	-
facsimileTelephoneNumber	—>	facsimileTelephoneNumber	Direct	4
user	-	**Metaverse_Contact**	-	-
info	—>	info	Direct	1
mailNickname	—>	mailNickname	Direct	1
group	-	**Metaverse_Contact**	-	-
mailNickname	—>	mailNickname	Direct	2
contact	-	**Metaverse_Contact**	-	-
mailNickname	—>	mailNickname	Direct	4
user	-	**Metaverse_Contact**	-	-
o	—>	o	Direct	1
contact	-	**Metaverse_Contact**	-	-
o	—>	o	Direct	2
user	-	**Metaverse_Contact**	-	-
pager	—>	pager	Direct	1
contact	-	**Metaverse_Contact**	-	-
pager	—>	pager	Direct	4
user	-	**Metaverse_Contact**	-	-
postalCode	—>	postalCode	Direct	1
contact	-	**Metaverse_Contact**	-	-
postalCode	—>	postalCode	Direct	4
user	-	**Metaverse_Contact**	-	-
st	—>	st	Direct	1
contact	-	**Metaverse_Contact**	-	-
st	—>	st	Direct	3
targetAddress	—>	targetAddress	Direct	4

group	-	**Metaverse_Contact**	-	-
mail	—>	targetAddress	Rules Extension - ADtargetaddress	5
user	-	**Metaverse_Contact**	-	-
mail	—>	targetAddress	Rules Extension - ADtargetaddress	6
group	-	**Metaverse_Contact**	-	-
msExchHideFromAddressLists	—>	msExchHideFromAddressLists	Direct	1
user	-	**Metaverse_Contact**	-	-
msExchHideFromAddressLists	—>	msExchHideFromAddressLists	Direct	3
contact	-	**Metaverse_Contact**	-	-
msExchHideFromAddressLists	—>	msExchHideFromAddressLists	Direct	5
user	-	**Metaverse_Contact**	-	-
postalAddress	—>	streetAddress	Direct	1
street	—>	streetAddress	Direct	2
streetAddress	—>	streetAddress	Direct	3
contact	-	**Metaverse_Contact**	-	-
streetAddress	—>	streetAddress	Direct	6
postalAddress	—>	streetAddress	Direct	7
user	-	**Metaverse_Contact**	-	-
telephoneNumber	—>	telephoneNumber	Direct	2
contact	-	**Metaverse_Contact**	-	-
telephoneNumber	—>	telephoneNumber	Direct	4
user	-	**Metaverse_Contact**	-	-
sAMAccountName	—>	uid	Direct	1
group	-	**Metaverse_Contact**	-	-
displayName	—>	uid	Direct	2
contact	-	**Metaverse_Contact**	-	-
mailNickname	—>	uid	Direct	5
user	-	**Metaverse_Contact**	-	-
proxyAddresses	—>	proxyAddresses	Direct	4
group	-	**Metaverse_Contact**	-	-
proxyAddresses	—>	proxyAddresses	Direct	5
contact	-	**Metaverse_Contact**	-	-
proxyAddresses	—>	proxyAddresses	Direct	6
user	-	**Metaverse_Contact**	-	-
textEncodedORAddress	—>	textEncodedORAddress	Direct	1
group	-	**Metaverse_Contact**	-	-
textEncodedORAddress	—>	textEncodedORAddress	Direct	3
contact	-	**Metaverse_Contact**	-	-
textEncodedORAddress	—>	textEncodedORAddress	Direct	4
user	-	**Metaverse_Contact**	-	-
mAPIRecipient	—>	MapiRecipient	Direct	2
contact	-	**Metaverse_Contact**	-	-
mAPIRecipient	—>	MapiRecipient	Direct	4
user	-	**Metaverse_Contact**	-	-
l	—>	l	Direct	2
contact	-	**Metaverse_Contact**	-	-
l	—>	l	Direct	4
user	-	**Metaverse_Contact**	-	-
physicalDeliveryOfficeName	—>	physicalDeliveryOfficeName	Direct	2
contact	-	**Metaverse_Contact**	-	-
physicalDeliveryOfficeName	—>	physicalDeliveryOfficeName	Direct	4
user	-	**Metaverse_Contact**	-	-
c	—>	c	Direct	1

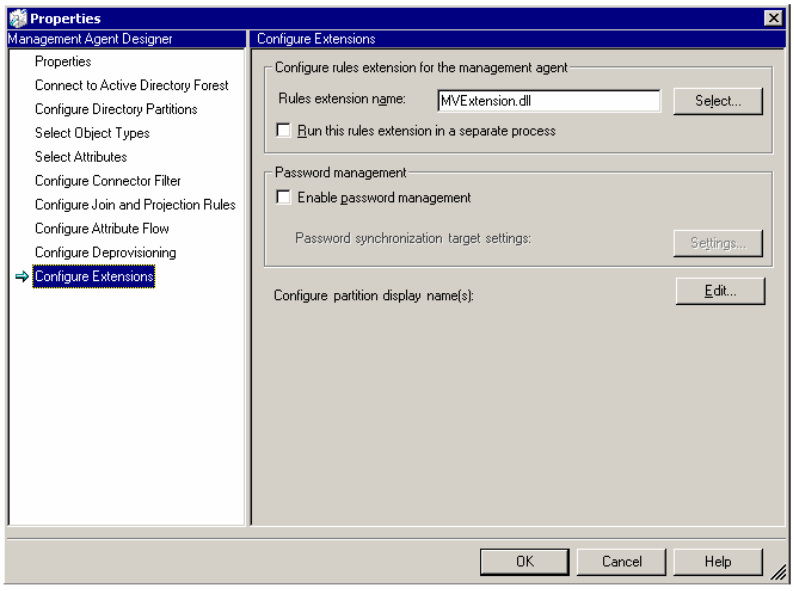
contact	-	**Metaverse_Contact**	-	-
c	—>	c	Direct	2

Configure Deprovisioning



As with Exchange, we need to select a deprovisioning method to use for our disconnected items. We did not create a specialized rules extension to control how items are removed. We feel that the standard deletion on export rule can be used to remove the target entry is the source is deleted. Because of the granular rights assignment and specific containers used for exports, we do not believe additional coding will be required. If additional coding work is required, this is the screen that you would use to specify the change.

Configure Extensions



This last configuration screen is used to specify which DLL is used to control the logic for this MA. Currently, there is a single extension DLL used for this MIIS environment. While this basic design works, it is not considered best-practice. Each MA should use its own extension.DLL when possible. To learn more about this concept, see the Developer Reference help file on the MIIS server or search for "Creating a Rules Extension from Multiple Sources."

This is a single agent designed to synchronize an Active Directory domain with the metaverse.

Production Environment

MIIS Location

Still to be Determined

Support Personnel

Still to be Determined

Synchronization Schedule

Still to be Determined

Additional Tasks

Processes like directory synchronization can always be improved. It may be necessary to begin replicating additional directory fields or perhaps force some field strings. As more Active Directories are brought onboard, some code-work will be needed and this is a good opportunity to make those “tweaks” necessary for additional services or simply to hard-code certain system settings.

Free and Busy Replication

As we mentioned earlier, public folder replication and free/busy data replicates easily within the same organization. Between organizations, additional processes are required in order to replicate the data or provide access to scheduling availability.

Exchange Server Inter-Organization Replication

The Exchange Server Inter-Organization Replication tool is a free download from Microsoft and provides the software and documentation needed to replicate free and busy information as well as public folder content between different Exchange organizations (as needed). By adding this component to Exchange environments that have synched directories, you will be able to restore public folder replication including free/busy data between disjointed Exchange organizations. The tool consists of two programs: the Replication Configuration program (exscfg.exe), and the Replication service (exssrv.exe). The Replication Configuration program creates a configuration file for setting the replication frequency, logging options, folders to be replicated, and accounts to be used. The Replication service continuously updates information from one server (designated as the Publisher) to one or more Exchange servers (designated as Subscribers).

This tool can be downloaded directory from Microsoft
http://www.microsoft.com/downloads/details.aspx?familyid=e7a951d7-1559-4f8f-b400-488b0c52430e&displaylang=en

Internet Free/Busy

Outlook 2000, 2002 and 2003 users can also take advantage of a feature called Internet Free/Busy. In this scenario, the user’s Outlook program is hard-coded to publish the user’s free/busy information to a specific web location such as ftp://Contactserver/Freebusy/MKirves.vfb or http://Contactserver/Freebusy/MKirves.vfb. Remember that no real data is published there, just blocks of time and a single digit to denote whether the person is free or busy during that time.

For lookups, this location is added to the user’s contact or mailbox information in the address list that way other Outlook clients know where to go to find the Free/Busy information for that user. This is also a field that could potentially be incorporated and forced with MIIS.

Here are some links that offer some guidance on using the Internet Free/Busy with Outlook:
http://support.microsoft.com/kb/196484
http://www.saas.nsw.edu.au/solutions/outlook/free-busy.html
http://www.outlookexchange.com/articles/joycetang/article5a.asp#2

Coding Optimization

Those with a developer background will notice that the code used in the appendix is not the cleanest. While the code functions and meets the requirements of the project, it could certainly be optimized. Development projects can always be fine-tuned, optimized and structured better. This environment is no different. In addition to optimization, there are several optional functions that could be implemented:

- Better Exception code could be handled. In other words, emails could be sent programmatically to alert the administrator if a disconnect or delete event is triggered.
- Exception code could also be added to write log files or send an email to alert the administrator of coding violations or MIIS errors.
- The country code information between AD and Exchange is much different. Currently, the country code format that must be used in Exchange is the two-letter designation as specified in the ISO standards. For example, US for United States and GB for the United Kingdom. A much more sophisticated lookup table could be converted to allow for different standards to be used in Exchange (and compatible with AD) such as United States or U.S.A.
- The Hide From Address Book flag could be synchronized so that hidden objects could be replicated in the same hidden state
- The MAPIRecipient attribute could be replicated to force rich text between the systems.
- A much more sophisticated deprovisioning logic could be put into place to provide “tombstoning” to occur within the MV for deleted source objects.
- Logic could be added to the code to block rights to objects that are not in the target container. In some instances, a contact in a target system could be “joined” by another object and an attempted write can occur. This is remedied by setting permissions on the containers, but it would be cleaner to implement some logic during the write that verifies the object’s DN matches the folder specified in the provisioning code

Provisioning Code (Extension DLL)

```
'Updated February 23, 2005
'Steve Bryant
Imports Microsoft.MetadirectoryServices
'Imports Mms_Metaverse.MVExtensionObject



Public Class MVExtensionObject
    Implements IMVSynchronization
    Implements IMASynchronization

    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

        Dim BusinessUnit1 As ConnectedMA  ' Management agent object
        Dim Legacy55 As ConnectedMA  ' Management agent object
        Dim Connectors As Integer           ' Management agent
connectors
        Dim DN As ReferenceValue            ' Distinguished name
attribute
        Dim Container As String             ' Container name
        Dim RDN As String                   ' Relative distinguished
name strings
        Dim csentry As CSEntry              ' Connector space entry
object
        Dim SMTPAddress As String         ' Building the SMTP target
address
        Dim TargetAddress As String         ' Building the SMTP target
address
        Dim displayname As String           ' Identify the display name
        Dim textEncodedORAddress As String
        Dim mailnickname As String

        If mventry.ObjectType = "locality" Then Exit Sub


        Legacy55 = mventry.ConnectedMAs("Exchange 55 GAL MA")
        BusinessUnit1 = mventry.ConnectedMAs("Business Unit MA")
        'BusinessUnit2 = mventry.connectedMAs("Second Business Unit MA)


        'Begin code for First Business Unit AD

        If BusinessUnit1.Connectors.Count = 0 Then
            BusinessUnit1 = mventry.ConnectedMAs("Business Unit MA")
            Connectors = BusinessUnit1.Connectors.Count
```

```
            ' Insert the OU for the business Unit here
            mailnickname = mventry("mailNickname").Value
            displayname = mventry("DisplayName").Value
            SMTPAddress = mventry("mail").Value
            TargetAddress = mventry("mail").Value
            Container = "OU=MIIS Import,DC=AlpineSkiHouse,DC=com"
            RDN = "CN=" & mventry("cn").Value
            DN = BusinessUnit1.EscapeDNComponent(RDN).Concat(Container)
            ' This line below denotes the target object class. We are
using custom recipients only
            'csentry =
BusinessUnit1.Connectors.StartNewConnector("contact")
            csentry =
ExchangeUtils.CreateMailEnabledContact(BusinessUnit1, DN, mailnickname,
TargetAddress)
            csentry.DN = DN
            csentry.CommitNewConnector()

        End If
        'Begin code for First Business Unit AD

        'Begin code for Legacy Exchange 5.5 Org
        If Legacy55.Connectors.Count = 0 Then

            'Provisioning based upon metaverse object type person
            If "Metaverse_Contact" = mventry.ObjectType Then

                Legacy55 = mventry.ConnectedMAs("Exchange 55 GAL MA")
                mailnickname = mventry("mailNickname").Value
                displayname = mventry("DisplayName").Value
                SMTPAddress = mventry("mail").Value
                ' Construct the textEncodedORAddress if needed
                textEncodedORAddress =
mventry("TextEncodedORAddress").Value
                ' If there is a g= missing in the TextEncodedORAddress,
then rem out the line above and allow these to execute
                'Dim strvar1 As String
                'strvar1 = mventry("TextEncodedORAddress").Value
                'textEncodedORAddress =
(StrReverse(Left(StrReverse(strvar1), InStr(2, StrReverse(strvar1),
";") - 1) & "=g;" & Right(StrReverse(strvar1),
(Len(StrReverse(strvar1)) - InStr(2, StrReverse(strvar1), ";")))))

                ' Construct the distinguished name
                DN = Legacy55.EscapeDNComponent("CN=" +
mventry("uid").Value).Concat("cn=import,ou=Atlanta,o=Exchange55")
                csentry =
ExchangeUtils.Create55CustomRecipient(Legacy55, DN, mailnickname,
displayname, SMTPAddress, textEncodedORAddress)

            End If

        End If
        'End code for Legacy Exchange 5.5 Org

    End Sub
```

```
    Public Function FilterForDisconnection(ByVal csentry As
Microsoft.MetadirectoryServices.CSEntry) As Boolean Implements
Microsoft.MetadirectoryServices.IMASynchronization.FilterForDisconnecti
on

    End Function

    Public Sub MapAttributesForExport(ByVal FlowRuleName As String,
ByVal mventry As MVEntry, ByVal csentry As CSEntry) Implements
IMASynchronization.MapAttributesForExport

        Select Case FlowRuleName.ToLower

            Case "cd.remote-address:textencodedoraddress<-
mv.metaverse_contact:textencodedoraddress"
                Dim strvar1 As String =
mventry("TextEncodedORAddress").Value
                csentry("TextEncodedORAddress").Value =
(StrReverse(Left(StrReverse(strvar1), InStr(2, StrReverse(strvar1),
";") - 1) & "=g;" & Right(StrReverse(strvar1),
(Len(StrReverse(strvar1)) - InStr(2, StrReverse(strvar1), ";"))))

            Case "cd.contact:c<-mv.metaverse_contact:c"
                csentry("c").Value = Left(mventry("c").Value, 2)

            Case "proxyaddresses"
                If mventry("proxyAddresses").IsPresent Then
                    Dim entry As Value
                    Dim work As String

                    Dim zapSMTP As New System.collections.ArrayList

                    ' discover existing csentry SMTP and smtp keys for
removal
                    If csentry("proxyAddresses").IsPresent Then
                        For Each entry In
csentry("proxyAddresses").Values

                            work = Left(entry.ToString, 4)
                            Select Case work
                                Case "SMTP", "smtp"
                                    zapSMTP.Add(entry.ToString)
                            End Select
                        Next
                    End If

                    'remove existing SMTP and smtp keys discovered
above
                    For Each work In zapSMTP
                        csentry("proxyAddresses").Values.Remove(work)
                    Next

                    'add mventry SMTP,smtp and X500 keys to connected
system
                    For Each entry In mventry("proxyAddresses").Values

                        work = Left(entry.ToString, 4)
```

```
                        Select Case work
                            Case "SMTP", "X500", "smtp"

csentry("proxyAddresses").Values.Add(entry.ToString)
                        End Select
                    Next
                End If

            Case "msexchpoliciesexcluded"
                ' Add this bit in there somewhere to identify the
target folder
                ' We would need similar code to identify the Exchange
5.5 system
                'Const PROVISIONED_OU =
"ou=mms,dc=testeurope,dc=testfs,dc=fujitsu,dc=local"
                'If
csentry.DN.ToString.ToLower.EndsWith(PROVISIONED_OU) Then
                csentry("msExchPoliciesExcluded").Value = "{26491CFC-
9E50-4857-861B-0CB8DF22B5D7}"
                'End If
        End Select

    End Sub


    Public Sub MapAttributesForImport(ByVal FlowRuleName As String,
ByVal csentry As Microsoft.MetadirectoryServices.CSEntry, ByVal mventry
As Microsoft.MetadirectoryServices.MVEntry) Implements
Microsoft.MetadirectoryServices.IMASynchronization.MapAttributesForImpo
rt

        Select Case FlowRuleName.ToLower

            Case "cd.organizationalperson:co->mv.metaverse_contact:c"

                If csentry("co").IsPresent Then
                    mventry("c").Value = Left(csentry("co").Value, 2)
                End If

            Case "mail", "rfc822mailbox"
                fixString(csentry, FlowRuleName, mventry, FlowRuleName)

            Case "mailnickname"
                fixString(csentry, "RDN", mventry, FlowRuleName)

            Case "uid"
                fixString(csentry, FlowRuleName, mventry, FlowRuleName)

            Case "cd.remote-address:target-address-
>mv.metaverse_contact:targetaddress"
                If csentry("target-address").IsPresent Then
                    mventry("targetaddress").Value =
Mid(csentry("target-address").Value, 6)
                End If

            Case "adtargetaddress"
```

```
                fixString(csentry, "mail", mventry, "targetaddress",
"SMTP:")

            Case "targetaddress"
                fixString(csentry, "rfc822mailbox", mventry,
FlowRuleName, "SMTP:")

            Case "proxyaddresses"
                Dim tempVal As Object
                If csentry("otherMailbox").IsPresent Then
                    For Each tempVal In csentry("otherMailbox").Values

mventry("proxyAddresses").Values.Add(tempVal.ToString)
                    Next
                End If

                If csentry("rfc822mailbox").IsPresent Then
                    mventry("proxyAddresses").Values.Add("SMTP:" &
csentry("rfc822mailbox").Value)
                End If

                If csentry("textEncodedORAddress").IsPresent Then
                    mventry("proxyAddresses").Values.Add("X400:" &
csentry("textEncodedORAddress").Value)
                End If

        End Select

    End Sub

    Public Sub MapAttributesForJoin(ByVal FlowRuleName As String, ByVal
csentry As Microsoft.MetadirectoryServices.CSEntry, ByRef values As
Microsoft.MetadirectoryServices.ValueCollection) Implements
Microsoft.MetadirectoryServices.IMASynchronization.MapAttributesForJoin

    End Sub

    Public Function ResolveJoinSearch(ByVal joinCriteriaName As String,
ByVal csentry As Microsoft.MetadirectoryServices.CSEntry, ByVal
rgmventry() As Microsoft.MetadirectoryServices.MVEntry, ByRef imventry
As Integer, ByRef MVObjectType As String) As Boolean Implements
Microsoft.MetadirectoryServices.IMASynchronization.ResolveJoinSearch

    End Function

    Public Function ShouldProjectToMV(ByVal csentry As
Microsoft.MetadirectoryServices.CSEntry, ByRef MVObjectType As String)
As Boolean Implements
Microsoft.MetadirectoryServices.IMASynchronization.ShouldProjectToMV

    End Function

    Public Sub Terminate1() Implements
Microsoft.MetadirectoryServices.IMASynchronization.Terminate

    End Sub
```

```
    Public Function ShouldDeleteFromMV(ByVal csentry As
Microsoft.MetadirectoryServices.CSEntry, ByVal mventry As
Microsoft.MetadirectoryServices.MVEntry) As Boolean Implements
Microsoft.MetadirectoryServices.IMVSynchronization.ShouldDeleteFromMV

    End Function

    Public Function Deprovision(ByVal csentry As
Microsoft.MetadirectoryServices.CSEntry) As
Microsoft.MetadirectoryServices.DeprovisionAction Implements
Microsoft.MetadirectoryServices.IMASynchronization.Deprovision

    End Function

    Public Sub Initialize1() Implements
Microsoft.MetadirectoryServices.IMASynchronization.Initialize

    End Sub
    Private Sub fixString(ByVal csentry As CSEntry, ByVal csAttrName As
String, ByVal mventry As MVEntry, ByVal mvAttrName As String, Optional
ByVal preFix As String = "")
        ' takes the csentry csAttrName and removes illegal characters
        ' and prepends prefix if provided e.g. SMTP:

        If csentry(csAttrName).IsPresent Then
            Dim work As String = csentry(csAttrName).Value
            work = Replace(work, "/", "")
            work = Replace(work, "\", "")
            work = Replace(work, "?", "")
            mventry(mvAttrName).Value = preFix & work
        End If
    End Sub
End Class
```