**Inceptio**

09

**Introduction to Inceptio Provisioning Engine**

**ILM**

**Søren Granfeldt**

# Copyright

Copyright © 2009, Inceptio A/S. The Inceptio® name and Inceptio® logo are registered trade-marks and may not be used without prior consent. The name Inceptio will be used as a short form for Inceptio A/S throughout the rest of this and related documents.

Any immaterial rights to this work belongs to Inceptio, including, but not limited to, any rights in or to any inventions, ideas, designs, methodic, techniques or improvements included herein or that has been produced as part of Inceptio’s construction of this work, business secrets, copyrights, “droit moral” and any renewal or extension of such rights.

Information contained in this document represents the current view of Inceptio on the issues discussed as of the date of publication. The content is based on the knowledge available at the time of preparation of this document and as much of these findings are subjective and subject to change as these products and markets evolve, the content herein may be interpreted differently over time.

Information in this document is subject to change without prior notice and cannot be regarded as an agreement with Inceptio A/S. Inceptio A/S cannot be held liable for any errors that may be contained in this document.

Any software described in this document must be used only with appropriate license. Inceptio re-cognizes any 3rd party trademarks in this document.

All rights reserved.

# Introduction

The Inceptio Provisioning Engine (IPE) is a standalone extension for Microsoft Identity Lifecycle Manager 2007 (ILM). It allows for administrators to create advanced provisioning and deprovisioning rules for ILM without writing a single line of code.

By creating an XML document using a text editor, ILM administrators are able to build, change or remove provisioning and deprovisioning rules in ILM without recompiling. Basically, IPE enables codeless provisioning in ILM.

## IPE – the kit

IPE consist of a couple of files that needs to be present in the folder structure containing the ILM installation. This is typically in C:\Program Files\Microsoft Identity Integration Server directory (ILM root).

* Directory::Configuration – Under the ILM root, a directory called ‘Configuration’ must exist or be created and this directory must contain the XML rule file for IPE.
* Directory::Logs – Under the ILM root, a directory called ‘Logs’ must exist or be created. This directory will hold any logs from IPE (if logging is enabled)
* File::Inceptio.Engine.MVExtension.XML – This is the actual rule file and must be named ‘Inceptio.Engine.MVExtension.XML’. This file contains the provisioning and deprovisioning rules. A template file has been supplied with this kit.
* File:: Inceptio.Ilm.Logging.XML – This is the settings file for IPE logging. The specific settings are documented in this document. This file must be a valid XML document and must exist in the Configuration directory.
* File::Inceptio.Engine.MVExtension.DLL – This is the provisioning engine DLL file that performs the actual provisioning. This file must exist in the default Extension directory of the ILM installation. This is also the file that should be set in Identity Manager as the Rules Extension. Some implementations of ILM make use of MVRouter. This is supported with IPE, however, not documented in this introductory document.
* File::Inceptio.Ilm.Logging.DLL – this is the library containing code for creating logs. This file must exist in the default Extension directory of the ILM installation.

# Rules

A rule consists of the following –

1. A name and description
2. A source (metaverse object) and target object (connector space object)
3. The name of the Management Agent being targeted
4. Action (provision or deprovisioning)
5. Conditions for provision
6. Initial flows – if any

All rules should be written in valid XML and contained in the file Inceptio.Engine.MVExtension.XML (the rules file). The rules file can contain as many rules as needed.

Below is a sample rule

<Rule>

<Name>Provision Person ==> OIM</Name>

<Description></Description>

<TargetManagementAgentName

xsi:type="xsd:string">OIM MA</TargetManagementAgentName>

<Enabled>true</Enabled>

<SourceObject>person</SourceObject>

<TargetObject>Identities</TargetObject>

<Action>provision</Action>

<RenameDnFlow>

<Source>UserIDOIM</Source>

<Target>BNAME</Target>

<ReprovisionOnRename>true</ReprovisionOnRename>

</RenameDnFlow>

<Conditions>

<ConditionBase xsi:type="ConditionMatch">

<Description>Check if isSAPUser is Y</Description>

<MVAttribute>isSAPUser</MVAttribute>

<Pattern>^Y$</Pattern>

</ConditionBase>

</Conditions>

<InitialsFlows>

<AttributeFlowBase xsi:type="AttributeFlowGuid">

<Target>[DN]</Target>

</AttributeFlowBase>

<AttributeFlowBase xsi:type="AttributeFlowConstant">

<Constant>Primary</Constant>

<Target>IDENTITYTYPE</Target>

</AttributeFlowBase>

<AttributeFlowBase xsi:type="AttributeFlowConstant">

<Constant>Employee</Constant>

<Target>IDENTITYCATEGORY</Target>

</AttributeFlowBase>

</InitialsFlows>

</Rule>

## Rule definition

Each rule is contained between the tags <Rule> and </Rule> and within this tag the following are required or allowed.

### Tag – Name

Contains a descriptive name for the rule. Required.

### Tag – Description

Contains a more elaborate description of what the rule does. Not required.

### Tag – TargetManagementAgentName

The name of the Management Agent (MA) that is targeted by the rule. The name must match the name of the MA as seen in the Identity Manager

### Tag – Enabled

This allow for a rule to be enabled or disabled selectivily without disabling provisioning for all rules as you would normally do in Identity Manager. If set to false, the rule is not processed.

Allow value(s): true or false (case-sensitive)

### Tag – SourceObject

This is the class of the metaverse object that this rule pertains to, e.g. person or group

### Tag – TargetObject

This is the class of the connector space object that this rule pertains to, e.g. user, group, computer. If this is a provisioning rule, this is also the class that would be provisioned to the connector space if conditions are met.

### Tag – Action

This is the action that IPE should perform against the objects in the connector space if the conditions are met.

Allowed values: provision or deprovision (case-sensitive)

### Section – RenameDnFlow

This section allows for specifying information for performing renames in a connector space. Renames are typically used against LDAP source, such as Active Directory for moving objects between OUs.

Currently, the rename supports changing the DN or any other primary key. It is also possible to perform de-provisioning and (re-)provisioning instead of an actual rename. This is useful for systems, such as SAP, that does not support renaming.

#### Tag – RenameDnFlow::Source

This is the source attribute containing the new anchor value, e.g. distinguishedName could be used against Active Directory

Allowed values: the name of any metaverse attribute

#### Tag- RenameDnFlow::Target

This is the target attribute (anchor) that holds the value to be renamed. If you specify the target value [DN] including the surrounding brackets the built-in DN value in ILM will be used for the target; this is useful with the Active Directory Management Agent if you wish to move objects between OU’s.

Allowed values: [DN] or the name of a valid connector space object attribute

#### Tag – RenameDnFlow::ReprovisionOnRename

This is a Boolean specifying whether a de-provisioning and re-provisioning should be done upon rename (if values of source and target attributes are different). This is useful for systems that does not support rename, i.e. current versions of SAP.

Note: The current version has this restriction: If multiple provisioning rules target the same MA, only include the rename rule on one of the rules.

Allowed values: true or false (case-sensitive)

### Section – Conditions

This section contains all conditions that must match in order for the action specified in the rule to be performed against the object in the connector space. All specified conditions must evaluate to true to have the action performed. If just one evaluates to false, no action is taken. For provisioning rules, the conditions are only validated when no connected object exists in the connector space. This is similar to how join-rules are validated. The provisioning rule conditions are evaluated until a connection is established and thereafter the connection is locked.

All Conditions are derived from a single base condition tag called ConditionBase and all contain a condition type specified by the xsi:type attribute. All conditions have an optional tag called Description that allows for a more elaborate description of what the condition means.

#### Tag – Condition xsi:type=ConditionMatch

This condition can be used to check the value of a metaverse attribute against a regular expression pattern. The name of the metaverse attribute to check is specified in the MVAttribute tag (see sample below) and the regular expression pattern is specified in the Pattern tag (see sample below)

Sample condition

<ConditionBase xsi:type="ConditionMatch">

<Description>If uid starts with GSP-</Description>

<MVAttribute>uid</MVAttribute>

<Pattern>^GSP-</Pattern>

</ConditionBase>

#### Tag – xsi:type=ConditionAttributeIsPresent

This condition can be used to check whether a metaverse attribute is present (i.e. has not recalled or likewise). The name of the metaverse attribute to check is specified in the MVAttribute tag (see below)

Sample condition

<ConditionBase xsi:type="ConditionAttributeIsPresent">

<Description>Only if expectedDN is populated</Description>

<MVAttribute>expectedDN</MVAttribute>

</ConditionBase>

#### Tag – xsi:type=ConditionAttributeIsNotPresent

This condition can be used to check whether a metaverse attribute is missing (i.e. has been recalled or never populated). The name of the metaverse attribute to check is specified in the MVAttribute tag (see below)

Sample condition

<ConditionBase xsi:type="ConditionAttributeIsNotPresent">

<Description>..and the group not exists</Description>

<MVAttribute>exists910</MVAttribute>

</ConditionBase>

### Section – InitialsFlows

This section contains all settings regarding initial flows that are performed when new objects are provisioned. There are several options for creating or building initial flow.

All attribute flows derived from a single base attribute flow tag called AttributeFlowBase and all contain a type specified by the xsi:type attribute.

#### Tag – xsi:type=AttributeFlowGuid

This flow allows for generating a new GUID and flowing this to an attribute. In the example below a new GUID is flowed to the built-in DN (specified by the surrounding brackets), however the target attribute could any valid connector space object attribute.

Sample flow

<AttributeFlowBase xsi:type="AttributeFlowGuid">

<Target>[DN]</Target>

</AttributeFlowBase>

#### Tag – xsi:type=AttributeFlowConstant

This type of flow rule allows flowing a constant to an attribute. In the example below the constant ‘A’ in flowed to the connector space object attribute named ‘USERTYPE’.

Sample flow

<AttributeFlowBase xsi:type="AttributeFlowConstant">

<Constant>A</Constant>

<Target>USERTYPE</Target>

</AttributeFlowBase>

#### Tag – xsi:type= AttributeFlowAttribute

This type of flow rule allows for flowing a metaverse attribute value directly to a connector space object attribute. In the example below, the value of the metaverse attribute ‘userID’ is flowed directly to the connector space object attribute ‘BNAME’.

Sample flow

<AttributeFlowBase xsi:type="AttributeFlowAttribute">

<Source>userID</Source>

<Target>BNAME</Target>

</AttributeFlowBase>

#### Tag – xsi:type= AttributeFlowConcatenate

This type of flow rule allow for concatenating several values (metaverse values or constants) to one target value in the connector space. This flow rule option is often used for constructing initial password and such.

In the example below an number of SourceExpressionBase of different types are concatenated and flowed to the ‘export\_password’ attribute in the connector space object attribute. The example concatenates the value of the attribute ‘department’ from metaverse with the constant ‘.’ (note the type of the SourceExpressionBase being SourceExpressionConstant) and finally with the value of the attribute ‘userID’ from the metaverse.

Sample flow

<AttributeFlowBase xsi:type="AttributeFlowConcatenate">

<SourceExpressions>

<SourceExpressionBase xsi:type="SourceExpressionAttribute">

<Source>department</Source>

</SourceExpressionBase>

<SourceExpressionBase xsi:type="SourceExpressionConstant">

<Source>.</Source>

</SourceExpressionBase>

<SourceExpressionBase xsi:type="SourceExpressionAttribute">

<Source>userID</Source>

</SourceExpressionBase>

</SourceExpressions>

<Target>export\_password</Target>

</AttributeFlowBase>

# Logging

All parameters for logging are controlled through the logging XML file called Inceptio.Ilm.Logging.XML .

## Parameters

Below is a sample logging setup file.

<rules-extension-properties>

<logging>

<use-single-log>false</use-single-log>

<file-name>InceptioIlm.log</file-name>

<file-name-format>InceptioIlm-{0:yyyy-MM-dd-HH-mm}.log</file-name-format>

<logging-level>3</logging-level>

<log-errors-in-eventlog>false</log-errors-in-eventlog>

</logging>

</rules-extension-properties>

### Parameter – use-single-log

Controls whether new logs files are generated or all logging information is written to the same file. If set to true, the filename specified in the file-name tag is used for logging filename. If set to false, the format specified under file-name-format is used.

Allowed value(s): true or false (case-sensitive)

### Parameter – file-name

The name of the file when use-single-log is true.

### Parameter – file-name-format

If use-single-log is set to false, the format for logging filename specified in this parameter is used. Currently date and time values can be specified.

Allowed value(s): date and time specifiers – see sample or refer to .NET documentation for specifying date / time formats