Upgrade Guide

/ ForgeRock Identity Management 7

Latest update: 7.0.4

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

This guide shows you how to upgrade an existing deployment to the latest ForgeRock® Identity Management release.

Table of Contents

Overview

This guide shows you how to upgrade an existing deployment to the latest ForgeRock Identity Management release.

Quick Start

 Migrate Configuration Migrate an existing IDM configuration to IDM 7.	 Update Repository Update an existing repository or install a new repository for IDM 7.
 Migrate Data Move the data in an existing IDM repository to an updated deployment.	 Migrate Workflows Migrate existing process instances to IDM 7.

The upgrade process is largely dependent on your deployment and on the extent to which you have customized IDM. Engage ForgeRock Support Services for help in upgrading an existing deployment. Also, read the Release Notes before you start an upgrade; specifically, "*Incompatible Changes*".

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

Chapter 1

About Upgrades

The automated update process available with previous IDM versions is no longer supported. This guide describes the manual process required to update an existing deployment to IDM 7. At a high level, the manual update process involves the following steps:

1. Install IDM 7.
2. Optionally, place your existing server in maintenance mode.
3. Migrate your existing IDM configuration to the new installation.
4. Update your repository.
5. Test that your scripts and customizations work as expected.
6. Migrate existing data to the new installation.
7. Migrate workflows.

Important

If you are upgrading from IDM 7.0.0 to IDM 7.0.1, see "*Update to a Maintenance Release*".

Supported Upgrade Paths

The following table contains information about the supported upgrade paths to IDM 7:

Upgrade Paths

Version	Upgrade Supported to IDM 7
IDM 6.5.x	✔
IDM 6.0.x	✔
IDM 5.5.x	✔
IDM 5.0.x	✔
IDM 4.5.x	✔
IDM 4.0.x	✔

Important

An upgrade from version 6.5.x to version 7 introduces no incompatible changes to the IDM configuration. Depending on how you have customized your deployment, there might be incompatible configuration changes when you upgrade from versions prior to 6.5.x. Read the upgrade documentation for each interim release and apply all required script and configuration changes.

Chapter 2
Before You Upgrade

Fulfill these requirements before you upgrade IDM, especially before upgrading the software in a production environment. Also see the requirements listed in "*Before You Install*" and the changes listed in "*Incompatible Changes*" in the *Release Notes*.

Before you start, verify that you have a supported Java version installed:

Supported Java Versions

Vendor	Versions
OpenJDK, including OpenJDK-based distributions: • AdoptOpenJDK/Eclipse Adoptium • Amazon Corretto • Azul Zulu • Red Hat OpenJDK ForgeRock tests most extensively with AdoptOpenJDK/Eclipse Adoptium.	11
Oracle Java	11

If the server uses an older version that is no longer supported, install a newer Java version before you update, and follow the instructions in "*Check Your Java Installation*" in the *Installation Guide*.

Then, follow these steps:

1. Back up your existing deployment by archiving the `openidm` directory and creating a backup of the repository and all other applicable databases.

   Note

   If you use workflow, you must manually dump the workflow database tables, and then import them *before* you start the new instance of IDM for the first time. The workflow database tables start with the prefix `ACT_`. For information on how to dump/import individual tables, see the documentation for your database.

2. To save a record of the audit logs from your existing IDM installation, manually copy the log files from the `/path/to/openidm/audit/` directory, before you start the upgrade.

3. Download and extract `IDM-7.0.4.zip` from the ForgeRock BackStage download site.

Chapter 3

Place a Server in Maintenance Mode

The maintenance service disables non-essential services of a running IDM instance, in preparation for an update to a later version. When maintenance mode is enabled, services such as recon, sync, scheduling, and workflow are disabled. The complete list of disabled services is output to the log file.

The router remains functional and requests to the `maintenance` endpoint continue to be serviced. Requests to endpoints that are serviced by a disabled component return the following response:

```
404 Resource endpoint-name not found
```

Before you enable maintenance mode, temporarily suspend scheduled tasks.

Manage maintenance mode over REST as follows:

+ *Enable maintenance mode*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/maintenance?_action=enable"
{
  "maintenanceEnabled": true
}
```

+ *Disable maintenance mode*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/maintenance?_action=disable"
{
  "maintenanceEnabled": false
}
```

+ *Check if a server is in maintenance mode*

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
"http://localhost:8080/openidm/maintenance?_action=status"
{
  "maintenanceEnabled": false
}
```

If the server is in maintenance mode, the command returns `"maintenanceEnabled": true`; otherwise it returns `"maintenanceEnabled": false`.

Chapter 4

Migrate Your Configuration

This chapter covers the steps required to migrate your IDM configuration to IDM 7.

There is no automated way to migrate a customized configuration to IDM 7, so you need to migrate customized configuration files manually. Assuming you are upgrading from IDM 6.5, there are three ways to do this:

- Use the new IDM 7 configuration files as a base, and copy any customizations you have made to the new files.

  This is the preferred option, particularly if you have used version control on your configuration and can determine the exact changes you have applied.

- Use your existing configuration files as a base, and add any new IDM 7 configuration to your existing files.

- Use your existing configuration "as is" with no IDM 7 changes.

In most cases, a customized IDM 6.5 configuration will work without further modification on IDM 7.

Migrate Configuration Files

For customized files in your project's `conf/` directory, check that the customizations are compatible with the changes outlined in "*Incompatible Changes*" in the *Release Notes*. If there are no incompatible changes, either copy your old configuration files to your IDM 7 installation, or copy any customization into the corresponding new configuration files.

Migrate `boot.properties`

On the IDM 7 installation, edit the `resolver/boot.properties` file to match any customizations that you made on your IDM 6.5 server. Specifically, check the following elements:

- The HTTP, HTTPS, and mutual authentication ports.

  If you changed the default ports in your IDM 6.5 deployment, make those same changes in the new `boot.properties` file.

- Check that the keystore and truststore passwords match the current passwords for the keystore and truststore of your existing IDM deployment.

- Check that the Changes to boot.properties are compatible with your customizations.

Migrate `logging.properties`

Check that the Changes to logging.properties are compatible with your customizations before you migrate your `conf/logging.properties` file.

Migrate Security Settings

Copy the contents of your IDM 6.5 `security/` folder to the IDM 7 installation. By default, the IDM 6.5 `security/` folder contains the following files:

- `keystore.jceks`
- `truststore`

The default IDM 7 `security/` folder includes two new files (`keystorepass` and `storepass`), which contain the passwords to the keystore and truststore. Update these files with the passwords that you have set for your keystore and truststore.

Warning

If you do not copy your old truststore and keystore files to your new instance, you will be unable to decrypt anything that was encrypted by your old instance of IDM.

Migrate Custom Aliases

Creating custom aliases for default keys is no longer supported. If your old deployment used custom aliases for the default secret keys, migrate them using one of the following methods:

- Generate new keys with the same custom aliases:

1. After you have upgraded IDM, but *before restarting the server*, turn off the auto-generation of the default keys by setting `"populateDefaults": false` in your `conf/secrets.json` file.
2. Generate new keys, using the same aliases you used in your previous deployment.
3. Make sure that your `conf/secrets.json` file shows the custom aliases.

- Use your old keys in the new IDM deployment:

1. After you have upgraded IDM, but *before restarting the server*, turn off the auto-generation of the default keys by setting `"populateDefaults": false` in your `conf/secrets.json` file.
2. Export the custom keys from the old keystore to the new keystore.
3. Make sure that your `conf/secrets.json` file shows the custom aliases.

Migrate Custom Scripts

Migrate any custom scripts or default scripts *that you have modified* to the `script` directory of your IDM 7 instance. In general, custom and customized scripts should be located in the `openidm/script` directory of your existing IDM deployment.

For custom scripts, review "*Incompatible Changes*" in the *Release Notes*. If you are confident that the scripts will work as intended on IDM 7, copy these scripts to the new instance.

If you modified a default IDM script, compare the default versions of the IDM 6.5 and IDM 7 scripts. If nothing has changed between the default versions, review your customizations against "*Incompatible Changes*" in the *Release Notes*. If you are confident that your changes will work as intended on the new version, copy the customized scripts to the new `script` directory.

If a default script has changed since the IDM 6.5 release, test that your customizations work with the new default script before porting your changes to that new script.

Note

The `bin/defaults/script/auth/amSessionCheck.js` script has been removed in IDM 7. The only supported method of authentication through AM is by using AM bearer tokens and the `rsFilter` authentication module. For information on configuring an integrated deployment, see the Platform Setup Guide.

If you modify any shell scripts, such as `startup.sh`, you must migrate your changes manually to the new version of the script.

Migrate Provisioner Files

Change any customized provisioner configurations in your existing deployment to point to the connectors that are provided with IDM 7. Specifically, make sure that the `connectorRef` properties reflect the new connector versions, where applicable. For example:

```
"connectorRef" : {
    "bundleName": "org.forgerock.openicf.connectors.ldap-connector",
    "bundleVersion": "[1.4.0.0,1.6.0.0)",
    "connectorName": "org.identityconnectors.ldap.LdapConnector"
},
```

Alternatively, copy the connector .jar files from your existing deployment into the `openidm/connectors` directory of the new installation.

Migrate UI Customizations

If you have customized the Admin UI, review any custom UI files from your IDM 6.5 deployment (generally in the `openidm/ui/admin/extension` directory), and compare them against the corresponding IDM 7 files.

For each customized file, copy the corresponding default IDM 7 UI files to a `openidm/ui/admin/extension` directory on the new instance.

Apply your customizations to files in the new `openidm/ui/admin/extension` directory.

Chapter 5

Update the Repository

When you have migrated your configuration to the new IDM installation, you need to handle the data that is stored in your repository. There are two options to update a repository:

- Upgrade your existing IDM 6.5 repository.
- Create a new IDM 7 repository, then migrate your data to the new repository.

When you have upgraded the repository, or created a new repository, start the IDM server and test that all your scripts are working as expected, before migrating your data.

Upgrade an Existing Repository

Upgrading an existing repository means that you do not need to migrate data. However, you need to run a series of scripts that modify the repository, to use the new features in IDM 7.

Note

Upgrading an existing repository is `not` supported if you use a DS repository. If you're upgrading from a previous IDM release with a DS repository, you must create a new repository, then migrate your data to the new repository.

Prepare an existing repository for IDM 7 as follows:

1. Clear all `configobjects` related tables. For example, in MySQL run:

   ```
   DELETE FROM openidm.configobjects;
   DELETE FROM openidm.configobjectproperties;
   ```

2. Increase the column length of the `objectid` column in the `locks` table.

   Queued synchronization creates locks when it acquires the mappings to process on an IDM node. The length of the `objectid` column in the `locks` table in versions prior to IDM 7 is 38 characters. Because the lock `_id` is set to the mapping name, it can easily exceed 38 characters. Increase the length of this column to 255 characters.

3. Delete existing `openidm-authorized` role relationships.

   Previous IDM releases created a relationship to the `openidm-authorized` for every new managed user. IDM 7 handles the default authorization differently, setting a `defaultUserRoles` property for users on authentication. For more information, see XREF.

In existing deployments, you can delete these existing relationships from the repository as follows:

```
DELETE FROM openidm.relationships
WHERE firstResourceCollection = 'internal/role' AND firstResourceId = 'openidm-authorized'
OR secondResourceCollection = 'internal/role' AND secondResourceId = 'openidm-authorized';
```

4. From your IDM 7 installation, run the schema update scripts for your database type.

   These scripts are located in the `openidm/db/database-type/scripts/updates` directory:

   `00-relationshipresources.sql`

   This script adds a new `relationshipresources` table, and loads the table with a distinct list of the resources that have existing relationships linked between them. Loading the data might take some time, if the relationship table is large. You will also need to update your `repo.jdbc.json` file, adding the new mapping to your `explicitMappings` configuration:

   ```
   "relationshipresources": {
     "table": "relationshipresources",
     "objectToColumn": {
       "_id": {
         "column": "id",
         "isNotNull": true
       },
       "originResourceCollection": {
         "column": "originresourcecollection",
         "isNotNull": true
       },
       "originProperty": {
         "column": "originproperty",
         "isNotNull": true
       },
       "refResourceCollection": {
         "column": "refresourcecollection",
         "isNotNull": true
       },
       "originFirst": {
         "column": "originfirst",
         "isNotNull": true,
         "type": "BOOLEAN"
       },
       "reverseProperty": "reverseproperty"
     }
   }
   ```

   `01-syncqueue.sql`

   Removes the `remainingRetries` property from the queued synchronization object. IDM 7 lets you configure an infinite number of queued synchronization retries.

   Remove the `remainingRetries` code block from `conf/repo.jdbc.json` file. Example code block to delete:

```
"remainingRetries": {
  "column": "remainingRetries",
  "type": "NUMBER"
}
```

`02-importobjects.sql`

Adds support for bulk import.

`03-reconassoc.sql`

Adds recon association tables to your repository.

You will also need to update your `repo.jdbc.json` to include the new `recon/assoc` mappings to your `explicitMapping` configuration:

```
"recon/assoc" : {
    "table" : "reconassoc",
    "objectToColumn" : {
        "_id" : "objectid",
        "_rev" : "rev",
        "mapping" : "mapping",
        "sourceResourceCollection" : "sourceResourceCollection",
        "targetResourceCollection" : "targetResourceCollection",
        "isAnalysis" : "isAnalysis",
        "finishTime" : "finishTime"
    }
},
"recon/assoc/entry" : {
    "table" : "reconassocentry",
    "objectToColumn" : {
        "_id" : "objectid",
        "_rev" : "rev",
        "reconId" : "reconId",
        "situation" : "situation",
        "action" : "action",
        "phase" : "phase",
        "linkQualifier" : "linkQualifier",
        "sourceObjectId" : "sourceObjectId",
        "targetObjectId" : "targetObjectId",
        "status" : "status",
        "exception" : "exception",
        "message" : "message",
        "messageDetail" : {"column" : "messagedetail", "type" : "JSON_MAP"},
        "ambiguousTargetObjectIds" : "ambiguousTargetObjectIds"
    }
},
"recon/assoc/entry/view" : {
    "table" : "reconassocentryview",
    "objectToColumn" : {
        "_id" : "objectid",
        "_rev" : "rev",
        "mapping" : "mapping",
        "reconId" : "reconId",
        "situation" : "situation",
        "action" : "action",
```

```
        "linkQualifier" : "linkQualifier",
        "sourceObjectId" : "sourceObjectId",
        "targetObjectId" : "targetObjectId",
        "sourceResourceCollection" : "sourceResourceCollection",
        "targetResourceCollection" : "targetResourceCollection",
        "status" : "status",
        "exception" : "exception",
        "message" : "message",
        "messageDetail" : "messageDetail",
        "ambiguousTargetObjectIds" : "ambiguousTargetObjectIds"
    }
}
```

If you use a Microsoft SQL Server repository, run the following additional scripts:

- `04-alter_ntext_openidm.sql`
- `05-alter_ntext_audit.sql`

These convert uses of `ntext` to `nvarchar(max)`, because Microsoft is deprecating `ntext`.

Important

For a managed relational database service such as Amazon RDS, be aware that some update scripts might require root level access to the system tables in the underlying database.

Specifically, certain PostgreSQL update scripts require access to the `pg_attribute` table. Because the database service super user is not the same as the PostgreSQL root user, such scripts might fail with a permissions error. In this case, investigate the failing script, and use an `ALTER TABLE` command on the specific IDM table instead.

5. Launch IDM and run the following Groovy script to clear the `reconprogressstate` data in your repository:

```
def result = openidm.query(
  "repo/reconprogressstate", [ "_queryFilter" : "true", "_fields" : "_id" ]).result;
for ( item in result ) {
  openidm.delete("repo/reconprogressstate/" + item["_id"], null);
}
return result.size() + " reconprogressstate records deleted";
```

This script will work regardless of the type of repository, and can be sent as a REST call. For example:

```
curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Content-Type: application/json" \
 --request POST \
 --data '{
   "type":"groovy",
   "source":"def result = openidm.query(\"repo/reconprogressstate\", [ \"_queryFilter\" : \"true\",
 \"_fields\" : \"_id\" ]).result; for ( item in result ) { openidm.delete(\"repo/reconprogressstate/\"
 + item[\"_id\"], null); }; return result.size() + \" reconprogressstate records deleted\";"
 }' \
 "http://localhost:8080/openidm/script?_action=eval"
"1 reconprogressstate records deleted"
```

6. Verify that all scripts and functions behave as expected.

Create a New Repository

Set up a new repository, following the steps in "*Select a Repository*" in the *Installation Guide*. A new repository is already configured for all the new capabilities in IDM, but does require migrating existing data to that repository.

If you create a new repository, you must still update your configuration files to use the new features.

After you have set up the new repository, migrate your data to that repository.

Chapter 6
Migrate Data

The data migration service helps you move information stored in an IDM repository to a new deployment. This service is off by default. To enable it, copy `migration.json` from `samples/example-configurations/conf/` into your `conf/` directory, and set `"enabled": true`.

Migration is run from your new installation through IDM's recon service, using your previous deployment as a data source. The data migration service supports importing information from IDM instances back to version 4. If you are migrating from a version of IDM earlier than that, you will need to follow previous update instructions to get your deployment into a state where it can be migrated using this service.

Note

Because the migration service migrates information that may be encrypted, such as passwords, you must make sure you have copied the `truststore` and `keystore` files from your previous deployment *before* you start the migration.

+ *Default Data Imported by the Migration Service*

- Internal Roles
- Internal Users
- Internal User Metadata
- Managed Roles
- Managed Users
- Managed Assignments
- Links and Relationships
- Scheduler jobs

Note

If you are migrating scheduler jobs from IDM 4.0 or 4.5, you will need to modify the entry in `migration.json` to be:

```
{
    "source" : "scheduler",
    "target" : "scheduler/job"
}
```

If you have additional object types (for example, managed devices), modify `migration.json` to include these objects.

- "Configure the Migration Service"
- "Run the Data Migration"
- "Delete Orphaned Meta Entries"

Configure the Migration Service

The data migration service is configured through `migration.json`. The default file assumes a default schema; modify the file if you have added custom managed data. The `migration.json` file can have the following properties:

enabled

Boolean, `true` or `false`. Enables the migration service.

connection

Configures the connection to the source IDM instance you are migrating from. Available properties:

instanceUrl

The URI for the source IDM instance.

authType

The authentication mechanism to the source IDM instance. Can be `basic` (username/password) or `bearer` (authentication using AM bearer tokens).

userName

Used for authenticating to the source IDM instance, if the `authType` is `basic`.

password

Used for authenticating to the source IDM instance, if the `authType` is `basic`.

clientId

Used for authenticating to the source IDM instance, if the `authType` is `bearer`.

clientSecret

Used for authenticating to the source IDM instance, if the `authType` is `bearer`.

tokenEndpoint

Used for authenticating to the source IDM instance, if the `authType` is `bearer`.

scope (optional)

List of OAuth scopes.

scopeDelimiter (optional)

Delimiter for the list of OAuth scopes.

tlsVersion (optional)

Lets you override the default TLS version.

connectionTimeout (optional)

Timeout for connecting to the source IDM instance (defaults to `10s`).

reuseConnections (optional)

Lets you override the default setting (defaults to `true`).

retryRequests (optional)

Lets you override the default setting (defaults to `true`).

hostnameVerifier (optional)

The SSL hostname verification policy. Specifies whether the host name presented by the remote server certificate is verified upon establishing new SSL connections. Possible values:

- `STRICT`: Requires that the host name match the host name presented in the certificate. Wildcards only match a single domain.
- `ALLOW_ALL`: Accepts any host name (disables host name verification).

Defaults to `STRICT`.

maxConnections (optional)

Lets you override the default maximum number of connections (default is `64`).

proxy (optional)

Lets you specify connection through a proxy server. Includes the following properties:

`proxyUri`

The proxy host and port to which IDM should connect.

`userName`

The user account to connect to the remote proxy.

`password`

The password of the proxy user.

socketTimeout

The TCP socket timeout, when waiting for HTTP responses. If you do not set a duration, the default is no timeout.

Example valid duration values:

- 4 days
- 59 minutes and 1 millisecond
- 1 minute and 10 seconds
- 42 millis
- unlimited
- none
- zero

mappings

A list of the endpoints that will be migrated from your old IDM instance to your new instance, expressed as mappings between the old and new instances. The complete list of mapping properties is the same as any regular synchronization mapping. Properties with particular significance for data migration include the following:

source

This is the only property that is *required* for data migration. The source should be the path to the resource within the repo; for example, `repo/managed/user`.

target

The path to the resource within the target repository. By default, this will be the same as the source path.

runTargetPhase

Specifies whether the migration should run the target phase of reconciliation. By default, this is set to `false`, as there is no data in the target repository.

sourceQuery

The query on the source system, used to find all objects to be migrated. Defaults to `"_queryFilter" : "true&fields=_id"`, which returns the IDs of all source objects.

You can improve migration performance by returning the whole source entry (setting the `sourceQuery` to `"_queryFilter" : "true"`).

sourceQueryFullEntry

(Optional). Specifies whether the defined source query returns full object data (`true`) or IDs only (`false`). Defaults to `true`.

If you do not set this parameter, IDM attempts to detect whether the full object is returned, based on the query results.

reconSourceQueryPaging

Specifies whether the migration service should use paging when querying the source IDM instance. By default, this is set to `false`. Turn paging on if you have a large data set and are concerned about memory usage.

For large data sets, you might be able to improve migration performance by turning paging on and increasing the query page size (using `reconSourceQueryPageSize`). The most effective page size will vary, depending on the available resources.

reconSourceQueryPageSize

Specifies the number of results to return per page, if paging is turned on. By default, 1000 results per page are returned.

allowEmptySourceSet

Specifies whether the migration service should continue if it encounters an empty source mapping. This is enabled by default.

properties

An array of properties you wish to perform additional actions on, such as modifying the contents of a property during the migration. (This follows the pattern you would find in a standard reconciliation. For more information about transforming data during a reconciliation, see "Transform Attributes in a Mapping" in the *Synchronization Guide*.)

policies

An array of policies you wish to apply to the data being migrated.

onCreate

The script used by the migration service for creating the data that is being migrated to the new installation. By default, this points to a Groovy script: `update/mapLegacyObject.groovy`.

onUpdate

The script used by the migration service for updating the data that is being migrated in the new installation. By default, this points to a Groovy script: `update/mapLegacyObject.groovy`.

correlationQuery

You can specify a custom correlation query. By default, this is:

```
"var map = {'_queryFilter': '_id eq \"' + source._id + '\"'}; map;"
```

For more information about writing correlation queries, see "*Correlating Source Objects With Existing Target Objects*" in the *Synchronization Guide*.

validSource

You can specify a script to validate the source object prior to migration. By default, this property is empty.

endpoint

By default, the migration service endpoint is `migration`. You can use the `endpoint` property to change this if needed.

Note

Because the data migration service performs a reconciliation between your old installation and your new installation, the general reconciliation optimizations also apply to the data migration service. For more information about reconciliation optimization, see "*Tuning Reconciliation Performance*" in the *Synchronization Guide*.

Run the Data Migration

Before you run your migration, make sure that you have done the following:

- Paused any scheduled jobs on the source deployment.
- Configured your `conf/migration.json` and `update/mapLegacyObject.groovy` files on the new IDM installation.
- Moved your configuration files from the old deployment to the new one.
- If you use workflow, you must manually dump the workflow database tables, and then import them *before* you start the new instance of IDM for the first time. The workflow database tables start with

the prefix `ACT_`. For information on how to dump/import individual tables, see the documentation for your database.

When you launch the new IDM installation, a new `migration` endpoint should be available. This endpoint supports the following actions:

- `migrate`: Triggers a migration of all legacy objects from the remote system. Optionally takes a `mapping` parameter in order to specify a specific mapping to migrate. For example:

```
curl \
 --header "X-OpenIDM-Username: openidm-admin" \
 --header "X-OpenIDM-Password: openidm-admin" \
 --header "Accept-API-Version: resource=1.0" \
 --request POST \
 "http://localhost:8080/openidm/migration?_action=migrate&mapping=repoManagedUser_repoManagedUser"
```

- `status`: Returns the last status for all reconciliations triggered by the migration service.
- `mappingConfigurations`: Returns the full list of migration mapping configurations.
- `mappingNames`: Returns the list of migration mapping names.

The period of time a migration takes will depend on the amount of information being migrated. Migrated data will retain the same object IDs they had in the previous deployment.

Delete Orphaned Meta Entries

Due to an issue in releases prior to IDM 7.0, if you used a PUT request to update an existing managed object that included metadata, the request would create an additional, orphaned meta object before the create failed. This might have resulted in a number of orphaned meta objects in your deployment.

After you have migrated your data, delete these orphaned meta objects as follows:

1. Update the following script with the credentials of your IDM administrative user and host system:

```
/path/to/openidm/bin/update/scripts/remove-orphan-meta.sh
```

2. Start IDM, if it is not running, then launch the script.

Chapter 7
Migrate Workflows

Migrating workflows for IDM 7 involves the following steps:

1. Manually dump the workflow database tables, and then import them before you start the new IDM instance for the first time.

   The workflow database tables start with the prefix `ACT_`. For information about dumping and importing individual tables, see the documentation for your database.

   Note

   Workflows are not supported with a DS repository.

2. Update your BPMN 2.0 workflow definitions, and any associated resource files, such as forms. Workflow forms in IDM 7 must use Vue.JS and updated Vue.JS libraries.

   The end-user validation mechanism has been upgraded (from VeeValidate 2.x to 3.x). If you are migrating workfows from IDM 6.5, follow the VeeValidate migration documentation to update your workflows accordingly.

3. Migrate process instances that are "in progress".

   a. When you have successfully migrated your IDM system and workflow definitions, query the old and updated process definitions. For example:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "accept: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/workflow/processdefinition?
key=contractorOnboarding&_queryId=filtered-query"
"result": [
    {
        "_id": "contractorOnboarding:1:5",
        ...
        "key": "contractorOnboarding",
        ...
        "version": 1
    },
    {
        "_id": "contractorOnboarding:2:105",
        ...
        "ioSpecification": null,
        "key": "contractorOnboarding",
        ...
        "version": 2
    }
]
```

b. Query existing process instances that use the old version of the process definition:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "accept: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request GET \
"http://localhost:8080/openidm/workflow/processinstance?processDefinitionId=contractorOnboarding
%3A1%3A5&_queryId=filtered-query"
"result": [
    {
        "_id": "6",
        ...
        "processDefinitionId": "contractorOnboarding:1:5",
        "processDefinitionKey": "contractorOnboarding",
        "processDefinitionName": "Contractor onboarding process",
        "processDefinitionVersion": 1,
        "processInstanceId": "6",
        ...
    }
]
```

c. Migrate the process instance to the new version of the process definition:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "accept: application/json" \
--header "Accept-API-Version: resource=1.0" \
--request POST \
--data '{
  "processDefinitionId": "contractorOnboarding:2:105"
}' \
"http://localhost:8080/openidm/workflow/processinstance/6?_action=migrate"
```

Chapter 8

Upgrade a Clustered Deployment

Follow these general steps when you are updating servers in a cluster:

- Redirect client traffic to a different IDM system or cluster.
- Shut down every node in the cluster.
- Update one node in the cluster.
- Clone the first node to the other nodes in that cluster.

Chapter 9
Update to a Maintenance Release

"Maintenance Releases" in the *Release Notes* incorporate a collection of fixes and minor RFEs. IDM 7.0.4 is the latest maintenance release for IDM 7. To upgrade an existing IDM 7 deployment to IDM 7.0.4, follow these steps:

1. Download and extract the IDM 7.0.4 binary from the ForgeRock BackStage download site.
2. Copy any customized configuration files, scripts, or workflow definitions from your existing deployment to the comparable directory in your 7.0.4 deployment.
3. Copy the `conf/authentication.json` file from your existing deployment to the `conf` directory in your 7.0.4 deployment.
4. Copy the keystore and truststore from your existing deployment to the 7.0.4 deployment. For example:

   ```
   cp -r /path/to/openidm7/security /path/to/openidm704
   ```

5. Configure the IDM 7.0.4 server to point to your existing repository:
   a. If you are using an external DS repository, make sure that the `conf/repo.ds.json` file in your new deployment is accurate.
   b. If you are using a JDBC repository, make sure that the `conf/repo.jdbc.json` and `conf/datasource.jdbc-default.json` files in your new deployment are accurate, *and* that your `resolve/boot.properties` file has the correct values for `openidm.repo.host` and `openidm.repo.port`.
6. The Flowable embedded workflow engine has been upgraded to version 6.6.0 in IDM 7.0.4.

   If you have existing workflow data in your IDM 7 instance, run the schema update scripts for your repository type (in the `openidm/db/database-type/scripts/updates` directory).

   Note

   Flowable can perform the schema updates automatically, but *only* if the database user account has permission to alter schemas. This might not be the case in your deployment.

7. Shut down your existing IDM 7 server.
8. Start up your IDM 7.0.4 server.

IDM Glossary

correlation query	A correlation query specifies an expression that matches existing entries in a source repository to one or more entries in a target repository. A correlation query might be built with a script, but it is not the same as a correlation script. For more information, see "*Correlating Source Objects With Existing Target Objects*" in the *Synchronization Guide*.
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a `correlation query`, a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JCE	Java Cryptographic Extension, which is part of the Java Cryptography Architecture, provides a framework for encryption, key generation, and digital signatures.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.
JSON Pointer	A JSON Pointer defines a string syntax for identifying a specific value within a JSON document. For information about JSON Pointer syntax, see the JSON Pointer RFC.

JWT	JSON Web Token. As noted in the JSON Web Token draft IETF Memo, "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For IDM, the JWT is associated with the `JWT_SESSION` authentication module.
managed object	An object that represents the identity-related data managed by IDM. Managed objects are configurable, JSON-based data structures that IDM stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For more information, see What is OSGi? Currently, only the Apache Felix container is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	IDM distinguishes between two distinct role types - provisioning roles and authorization roles. For more information, see "Managed Roles" in the *Object Modeling Guide*.
source object	In the context of reconciliation, a source object is a data object on the source system, that IDM scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, IDM then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.

system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in IDM for the period during which IDM requires access to that entry. System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that IDM scans after locating its corresponding object on the source system. Depending on the defined mapping, IDM then adjusts the target object to match the corresponding source object.