Aras Innovator 11

Data Synchronization Service Sample Package Guide

Document #: 11.0.02018110501

Last Modified: *11/16/2018*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Phone: 978-806-9400
Fax: 978-794-9826

E-mail: Support@aras.com

Website: https://www.aras.com/

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
Support@aras.com
Subject: Aras Innovator Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Innovator Documentation

Or,

FAX:
978-794-9826
Attn: Aras Innovator Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support

Document Conventions

The following table highlights the document conventions used in the document:

Table 1: Document Conventions

Convention	Description
Bold	This shows the names of menu items, dialog boxes, dialog box elements, and commands. Example: Click **OK**.
`Code`	Code examples appear in `courier` font. It may represent text you type or data you read.
`Yellow highlight`	Code highlighted in yellow draws attention to the code that is being indicated in the content.
`Yellow highlight with red text`	Red text highlighted in yellow indicates the code parameter that needs to be changed or replaced.
Italics	Reference to other documents.
Note:	Notes contain additional useful information.
Warning	Warnings contain important information. Pay special attention to information highlighted this way.
Successive menu choices	Successive menu choices may appear with a greater than sign (-->) between the items that you will select consecutively. Example: Navigate to **File** --> **Save** --> **OK**.

1 Terminology

The following table describes the terms used in the Administrator Guide.

Term	Definition
DSS	Data Synchronization Services. The set of APIs introduced in 11.0 SP15 that enable one way data synchronization.
Source System	The source Aras Innovator instance *from* which the data is being pushed.
Destination System	The destination Aras Innovator instance *to* which the data from the Source System is being pushed.
Synchronization Scope	The set of Items that are being synchronized between the Source and Destination Systems.
Synchronization Map	The Query Definition(s) that determines which set of ItemTypes and Relationships are to be synchronized.

2 Overview

The Data Synchronization Service (DSS) Sample Package provides an example of how to use the DSS APIs introduced in 11.0 SP15 to create an environment where data from one Aras Innovator instance can be synchronized with another instance of Aras Innovator. This package consists of a set of AML packages that need to be imported to the Source and Destination systems. Once you install these packages and follow the configuration steps described in this guide, the sample environment allows you to:

- Add, update, and delete Items that are to be synchronized between the Source and Destination Systems.
- Synchronize Items using the Synchronize action.
- Monitor Items that are in the Synchronization Scope but have not been synchronized yet using the Show Synchronization Scope action.
- View synchronized items on the Destination Systems.

Note: Please see the *Data Synchronization Service Programmer's Guide* for API details.

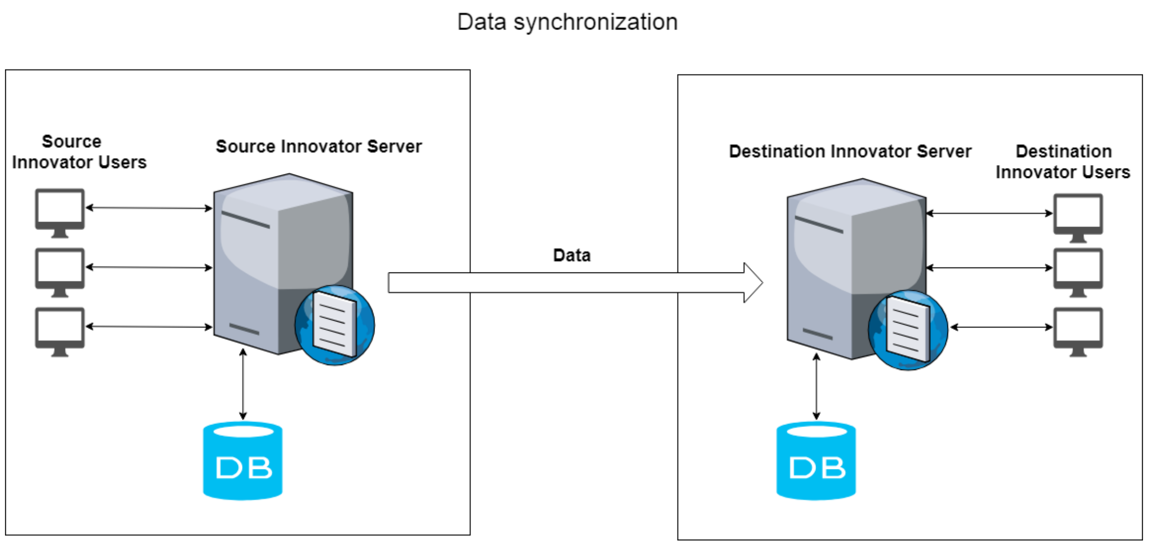


Figure 1.

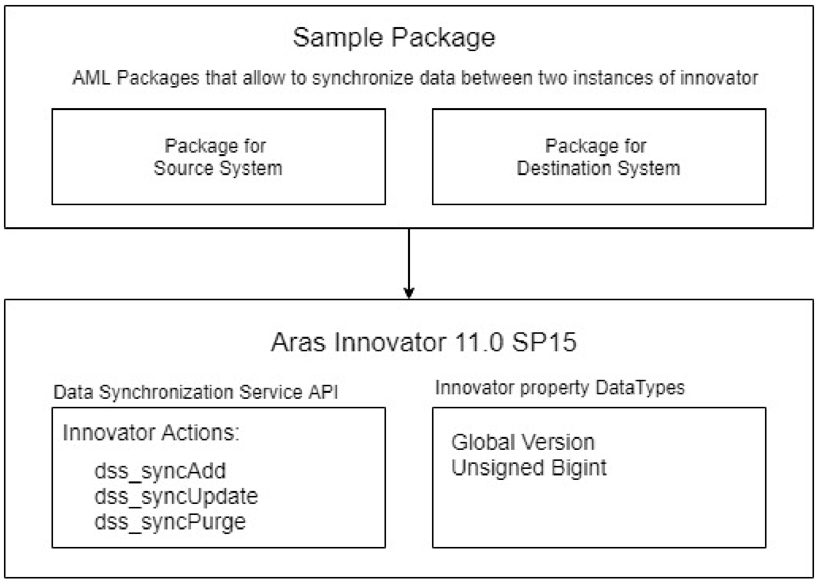

Sample Package
AML Packages that allow to synchronize data between two instances of innovator
Package for Source System
Package for Destination System
Aras Innovator 11.0 SP15
Data Synchronization Service API
Innovator Actions:
dss_syncAdd
dss_syncUpdate
dss_syncPurge
Innovator property DataTypes
Global Version
Unsigned Bigint

Figure 2.

2.1 Limitations

The sample package has the following restrictions and limitations. Please see section 6 Potential Solution Risks for an additional list of potential risks associated with using DSS and this sample package.

- The Destination system is intended only for reading information, so the expectation is that users on the Destination system will not lock/update/delete the synchronized elements.
- The ItemType metadata on both the Source and Destination systems is the same to make sure that the synchronized data from the Source system is saved and correctly interpreted by the Destination system. However, permissions are not synchronized and *may* be different in the Source and Destination systems.
- The Source system knows about the credentials necessary for data synchronization (for example, a special user). This user/identity has access to add/update/delete instances of ItemTypes that are intended to be synchronized.
- This example synchronizes data using the Innovator Admin user/identity, so the expectation is that the Destination system has the same user with the default password
- The sample package is not intended to synchronize more than 1000 items (this number includes Sync Root, Part, CAD, Files, and the relationship items between them). The focus of the sample package is to show possible configuration and performance issues that were not addressed in the first release.

3 Setup

3.1 Installing Innovator Server

For our example, we need two instances of Innovator Server (version 11.0 SP15). They will serve as our Source and Destination systems.

To install Innovator server instances refer to the **Aras Innovator 11.0 Installation Guide** (https://www.aras.com/support/documentation/11.0%20SP12/Installation%20and%20Configuration/Aras%20Innovator%2011.0%20-%20Installation%20Guide.pdf).

For the Source Innovator instance, you should also install the Conversion Server and Agent Service.

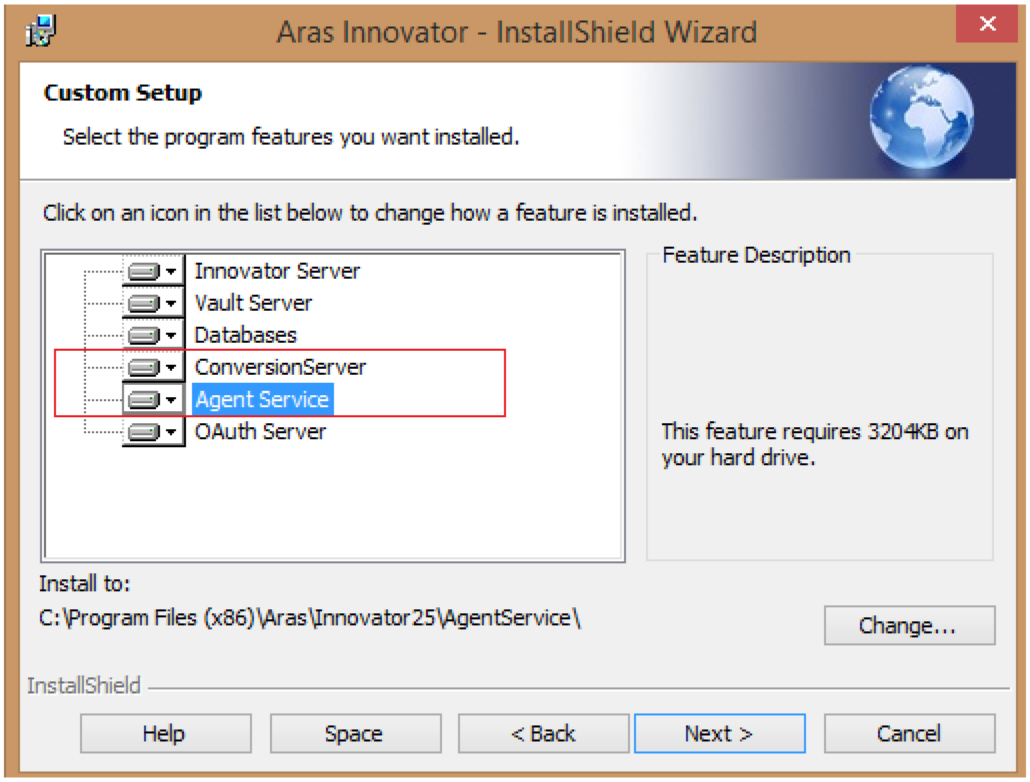

Aras Innovator - InstallShield Wizard
Custom Setup
Select the program features you want installed.
Click on an icon in the list below to change how a feature is installed.
Innovator Server
Vault Server
Databases
ConversionServer
Agent Service
OAuth Server
Feature Description
This feature requires 3204KB on your hard drive.
Install to:
C:\Program Files (x86)\Aras\Innovator25\AgentService\
Change...
InstallShield
Help
Space
< Back
Next >
Cancel

Figure 3.

3.2 Importing the AML Packages

You need to import different AML packages for the Source and Destination Innovator Instances. You can find them in the Source and Destination folders. Use the Aras Innovator Import tool to import them. Use **super user** as the login user name.

3.3 Setting up the Conversion and Agent Services on the Source System

The synchronization process uses the standard Conversion and Agent Services. In this section, you configure the Conversion Server and Agent Service for the Source System:

1. Open **\AgentService\conversion.config** for editing.
2. Add a cycle to ***<ProcessingCycles>***:

```
<cycle DB="DB_name_of_source_build" User="admin" Interval="20000" MaxBatch="5" />
```

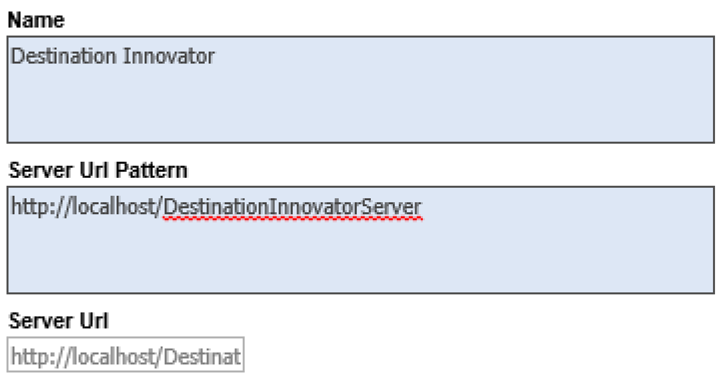
3. Edit **\AgentService\Aras.Server.Agent.Service.exe.config**.
4. Set the correct value for the Innovator server in ***<appSetting>***:

```
<add key="InnovatorServer" value="http://name_of
server_machine/web_alias_of_source_build/Server/InnovatorServer.aspx"></add>
```

5. Restart the Agent Service on the Source server.

3.4 Setting up the Source System

1. Log into the Source System.
2. Go to **Administration/File Handling/Conversion tasks**. Either confirm that there are no tasks with rule = LauncherSynchronizationRule, or delete them.
3. Go to **Administration/Data Synchronization/Innovator Servers**.
4. Open the existing item and set the correct Destination Server URL Pattern, as shown in the following example:



Name
Destination Innovator

Server Url Pattern
http://localhost/DestinationInnovatorServer

Server Url
http://localhost/Destinat

Figure 4.

5. Go to **Administrations/Data Synchronization/Synchronization Rules**. Confirm that the item exists.
6. Open the rule for editing and set the following values:

   a. **Destination database = Database** for the Destination Aras Innovator. This drop-down updates when you set or change a value for the Destination system.
   b. **Files Checkout Path = Folder** for the installed **Source** Aras Innovator or another folder (confirming that the permissions for the folder are correct).

1. Click **Save**, **Unlock** and **Close**.
2. Go to **Methods**. Run the **ConversionTaskCreate** method.

4 Using the Sample Package

The sample package is constructed to support the synchronization of the following structure:

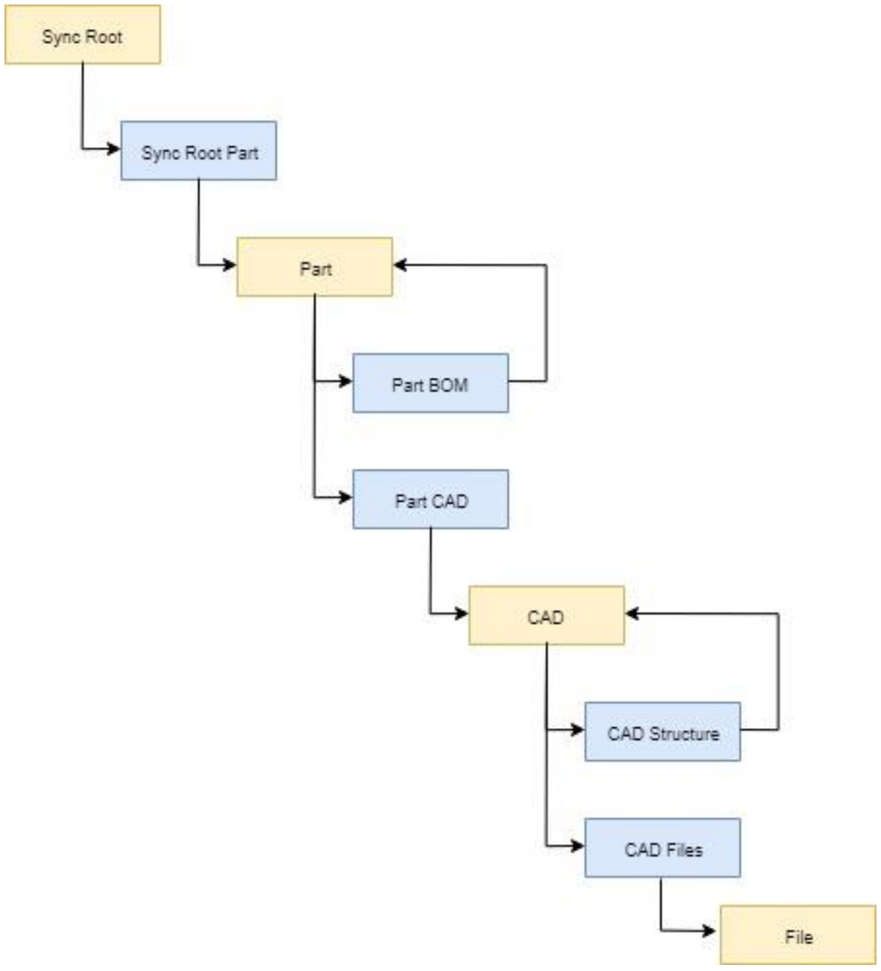


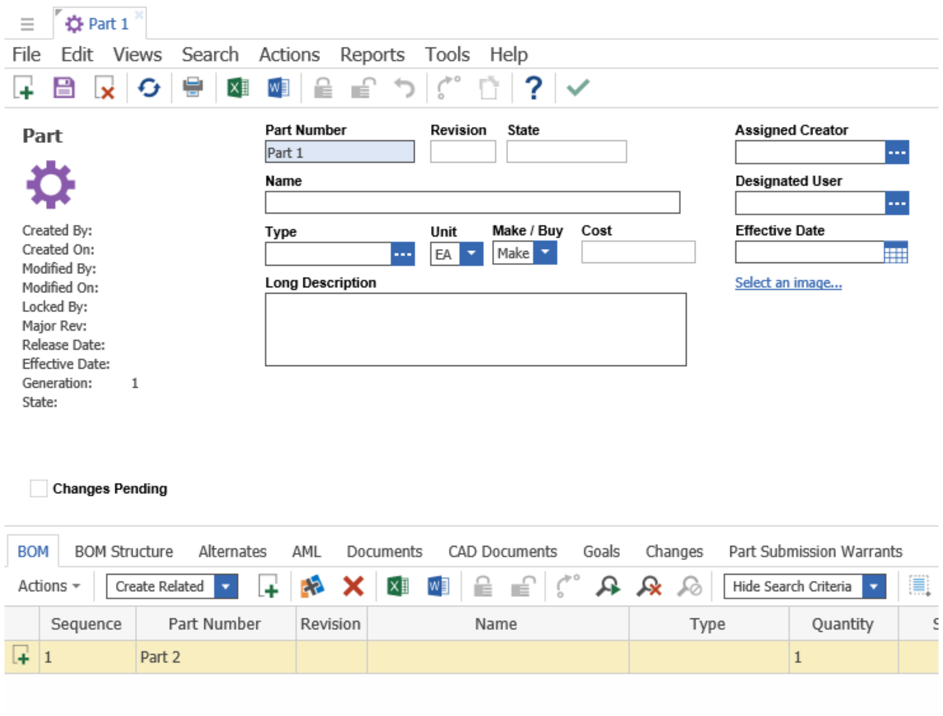
Figure 5.

This section describes how to synchronize all Sync Root items with nested Parts and CAD documents.

Sync Root is a simple ItemType, provided in the sample package, with one property “name”. It is used for the aggregation of **Parts** that are synchronized. **Sync Root** is intended to represent a Partner, Vendor, Customer, or Supplier with whom we share Part/CAD information and make it possible to read the data on the Destination System. You can only synchronize Parts that you add to a Sync Root. The custom property **Internal** is added to Part ItemType. If this property is checked on a Part, it is not synchronized even if the Part is added to a Sync Root.

4.1 Creating Sample Data

1. Create a Part (**Part 1**).
2. Create **Part 2** and create a relationship by adding it to the BOM tab.



Part 1

File Edit Views Search Actions Reports Tools Help

Part

Created By:
Created On:
Modified By:
Modified On:
Locked By:
Major Rev:
Release Date:
Effective Date:
Generation: 1
State:

Part Number	Part 1
Revision	
State	
Assigned Creator	
Name	
Designated User	
Type	
Unit	EA
Make / Buy	Make
Cost	
Effective Date	
Long Description	

Select an image...

Changes Pending

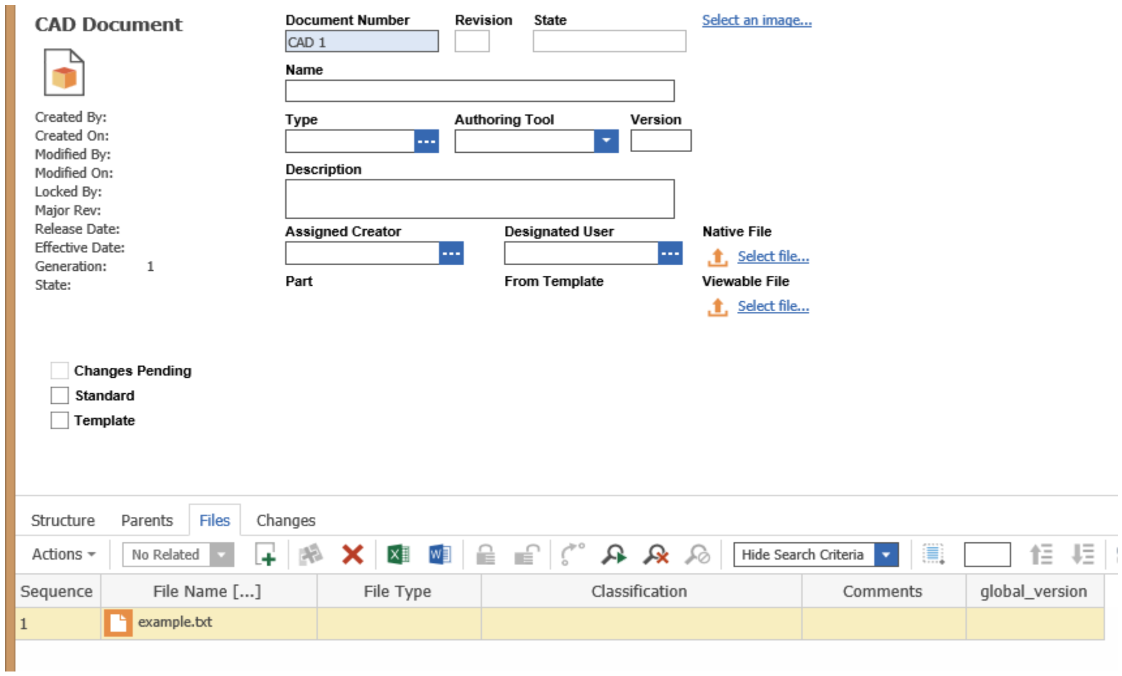
BOM | BOM Structure | Alternates | AML | Documents | CAD Documents | Goals | Changes | Part Submission Warrants

Actions | Create Related | Hide Search Criteria

Sequence	Part Number	Revision	Name	Type	Quantity
1	Part 2				1

Figure 6.

3. Create a CAD Document (**CAD 1**) and create a relationship by adding a file to the **Files** tab.



CAD Document

Created By:
Created On:
Modified By:
Modified On:
Locked By:
Major Rev:
Release Date:
Effective Date:
Generation: 1
State:

Document Number	CAD 1
Revision	
State	
Name	
Type	
Authoring Tool	
Version	
Description	
Assigned Creator	
Designated User	
Native File	Select file...
Part	
From Template	
Viewable File	Select file...

Select an image...

Changes Pending
Standard
Template

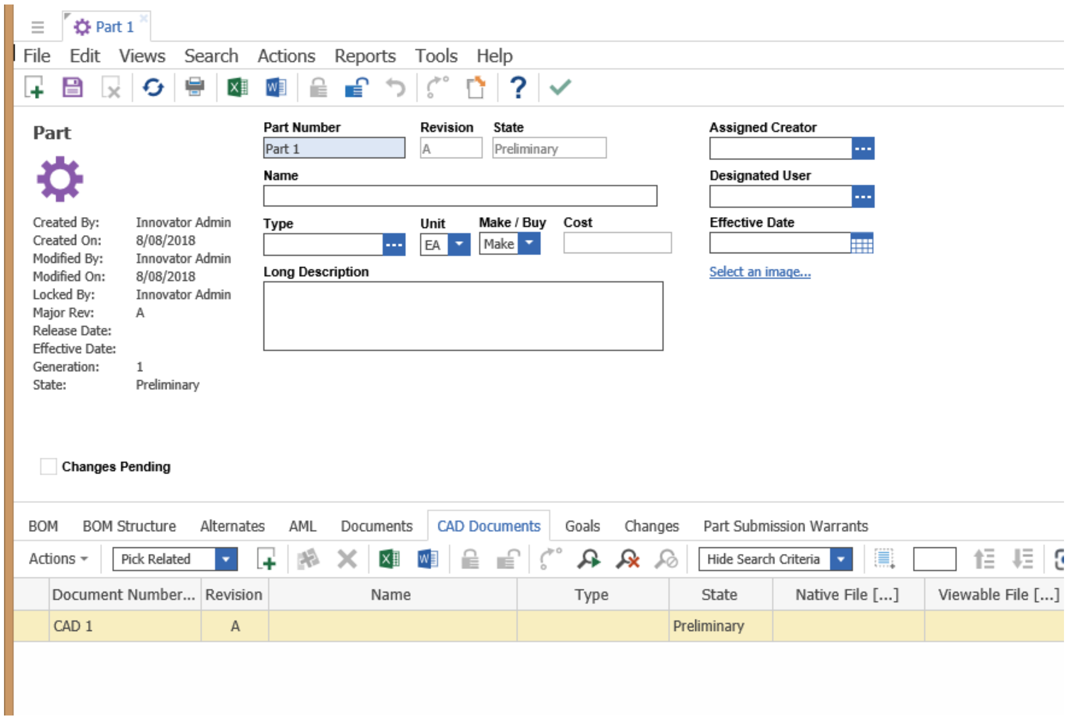
Structure | Parents | Files | Changes

Actions | No Related | Hide Search Criteria

Sequence	File Name [...]	File Type	Classification	Comments	global_version
1	example.txt				

Figure 7.

3. Attach **CAD 1** to **Part 1** using the **CAD Documents** relationship as shown in the following screenshot.



Part Number	Part 1
Revision	A
State	Preliminary
Name	
Type	
Unit	EA
Make / Buy	Make
Cost	
Long Description	
Assigned Creator	
Designated User	
Effective Date	

Select an image...

Created By:	Innovator Admin
Created On:	8/08/2018
Modified By:	Innovator Admin
Modified On:	8/08/2018
Locked By:	Innovator Admin
Major Rev:	A
Release Date:	
Effective Date:	
Generation:	1
State:	Preliminary

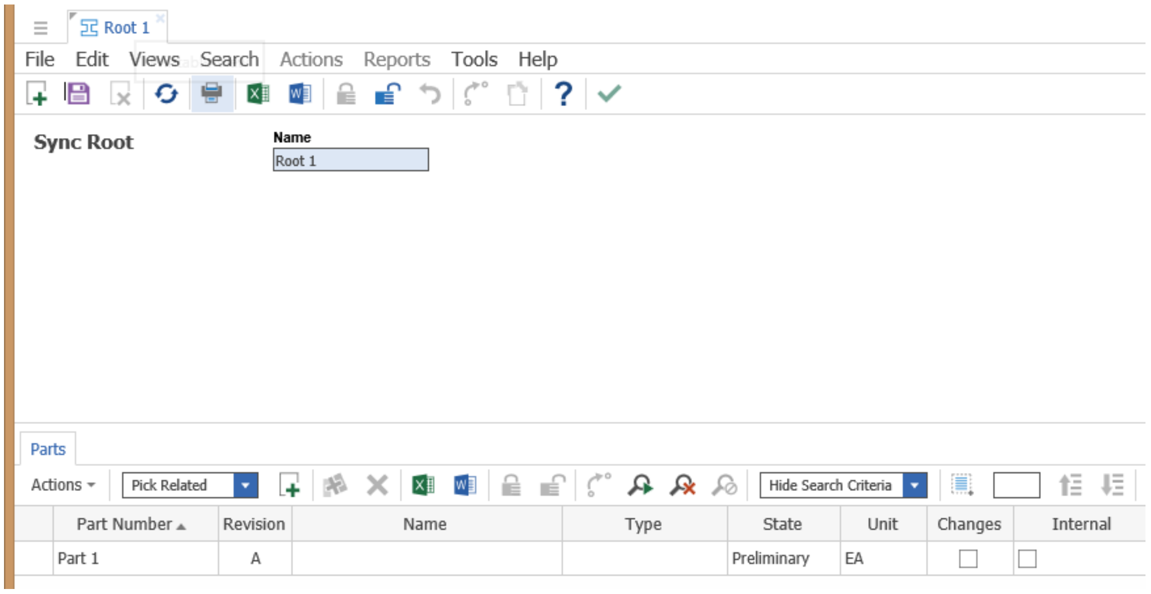
Changes Pending

BOM BOM Structure Alternates AML Documents CAD Documents Goals Changes Part Submission Warrants
Actions Pick Related Hide Search Criteria

Document Number...	Revision	Name	Type	State	Native File [...]	Viewable File [...]
CAD 1	A			Preliminary		

Figure 8.

4. Navigate to **Administration>Data Synchronization>Sync Roots**.
5. Create a new Sync Root Item (**Root 1**) and attach **Part 1** by adding it to the **Parts** tab.


Root 1
File Edit Views Search Actions Reports Tools Help
Sync Root
Parts
Actions Pick Related Hide Search Criteria

Name: Root 1

Part Number ▲	Revision	Name	Type	State	Unit	Changes	Internal
Part 1	A			Preliminary	EA	☐	☐

Figure 9.

You have created a simple structure: **Sync Root > Part > CAD > File**.

4.2 Running Show Sync Scope

Use the following procedure to run the **Show Sync Scope** action to show the scope of items designated for synchronization with the Destination Server:

1. Select **Administration>Data Synchronization>Sync Roots** from the TOC**.**
2. Select **Actions>Show Sync Scope**.

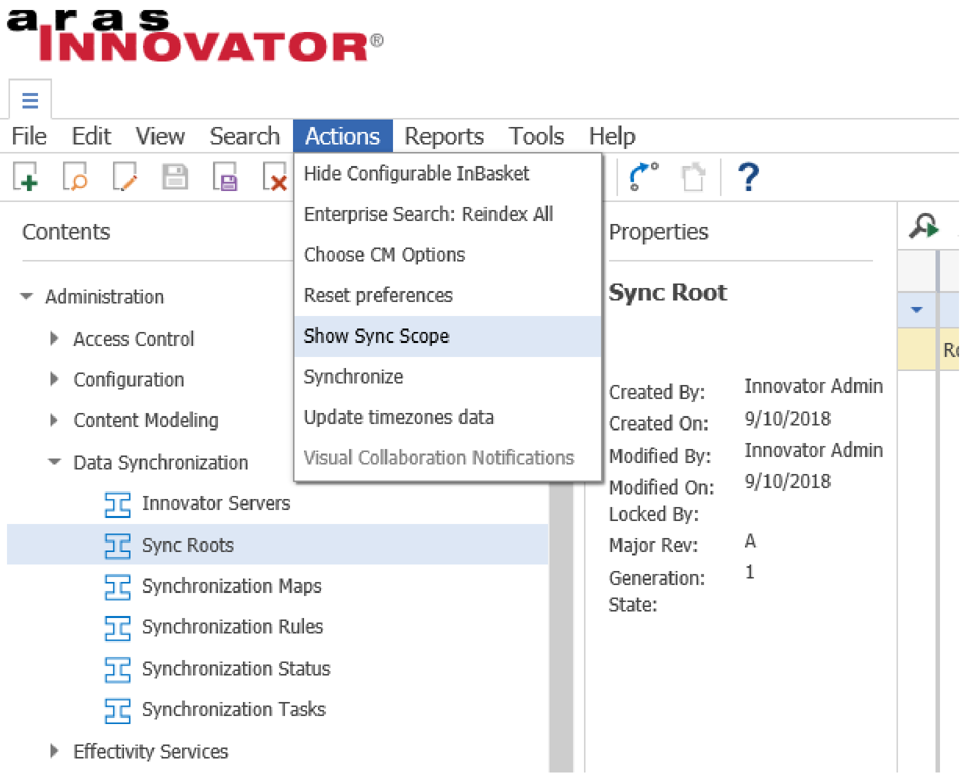
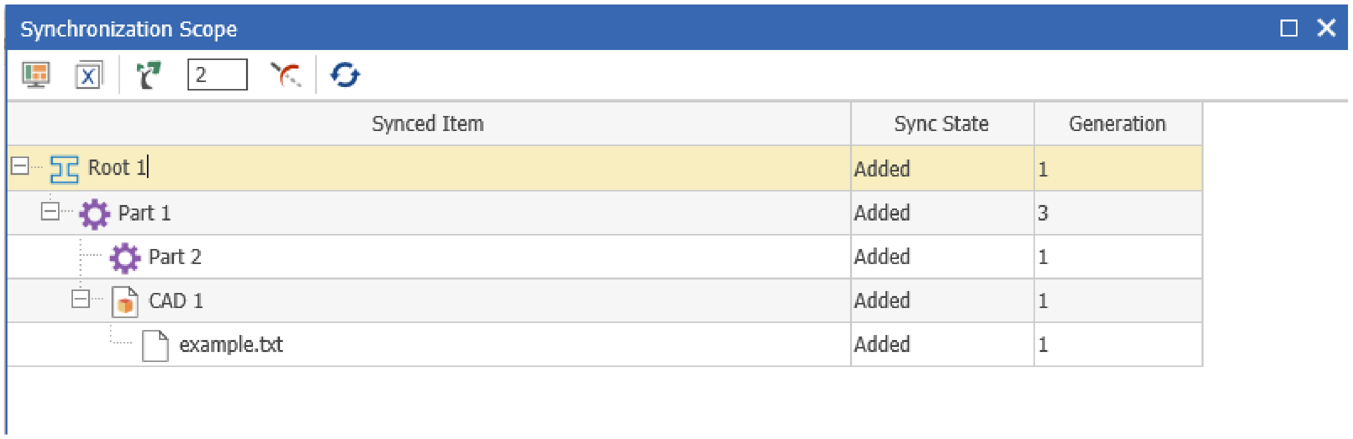


Figure 10.

A Tree Grid View similar to the following appears:



Synced Item	Sync State	Generation
Root 1	Added	1
Part 1	Added	3
Part 2	Added	1
CAD 1	Added	1
example.txt	Added	1

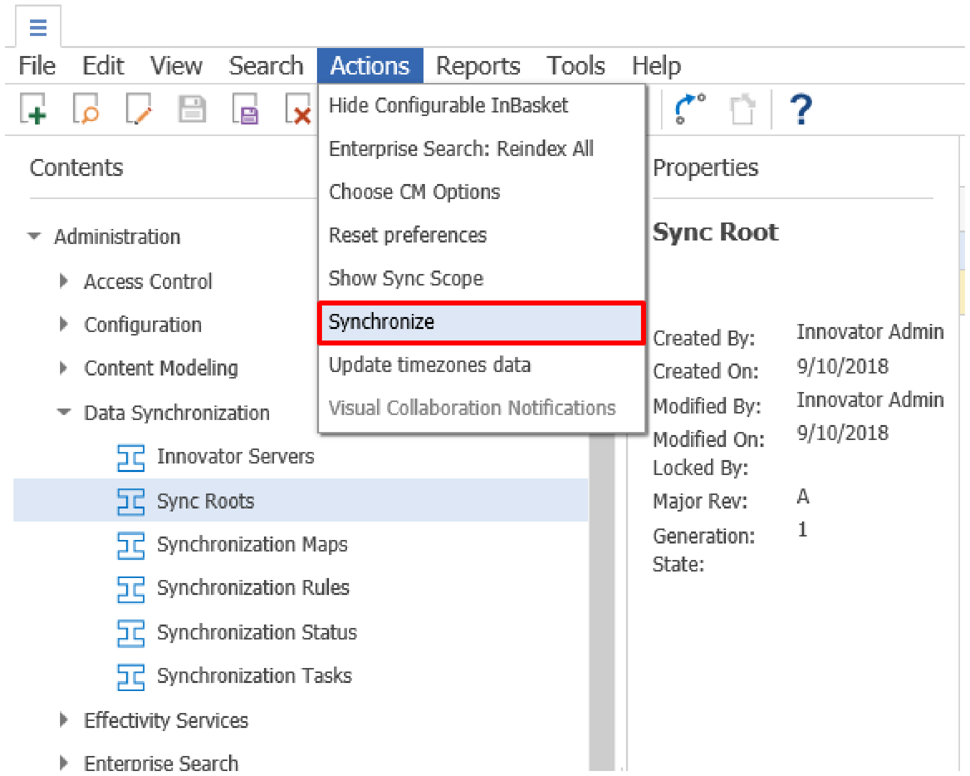
Figure 11.

The Synced Item column displays the items that have been added, updated or Not Modified since the last Synchronize action was performed. The Sync State column displays the current state of the associated item. Added and updated items are synchronized to the Destination system during the next **Synchronize** action.

Note: Please keep in mind that this view doesn't display items that were synchronized to the Destination system during the last synchronization event but have subsequently been deleted in the Source system or excluded from the Sync Roots. These items will be deleted from the Destination system during the next synchronize action.

4.2.1 Creating a Synchronization Task

Before you can create a synchronization task, you need to select the synchronization action as shown in the following screenshot:

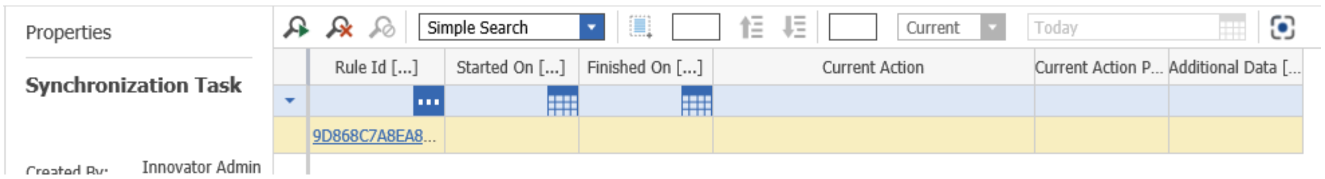


Created By:	Innovator Admin
Created On:	9/10/2018
Modified By:	Innovator Admin
Modified On:	9/10/2018
Locked By:	
Major Rev:	A
Generation:	1
State:	

Figure 12.

A new Synchronization Task is created.

3. Select **TOC>Administration>Data Synchronization>Synchronization Tasks** to see the task status.

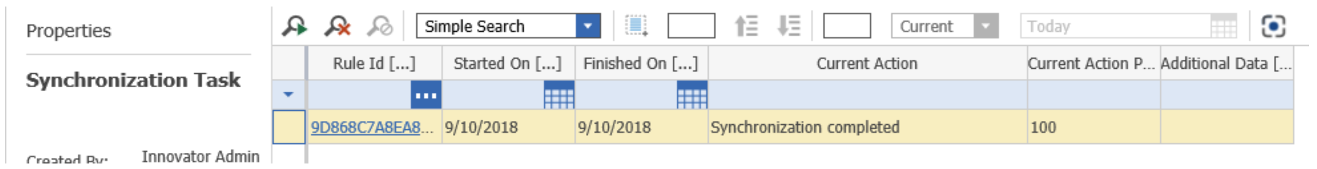


Rule Id [...]	Started On [...]	Finished On [...]	Current Action	Current Action P...	Additional Data [...
9D868C7A8EA8...					

Created By: Innovator Admin

Figure 13.

The task has not yet started. As soon as the Conversion Server takes it to process, the status changes and the data is synchronized.



Rule Id [...]	Started On [...]	Finished On [...]	Current Action	Current Action P...	Additional Data [...
9D868C7A8EA8...	9/10/2018	9/10/2018	Synchronization completed	100	

Created By: Innovator Admin

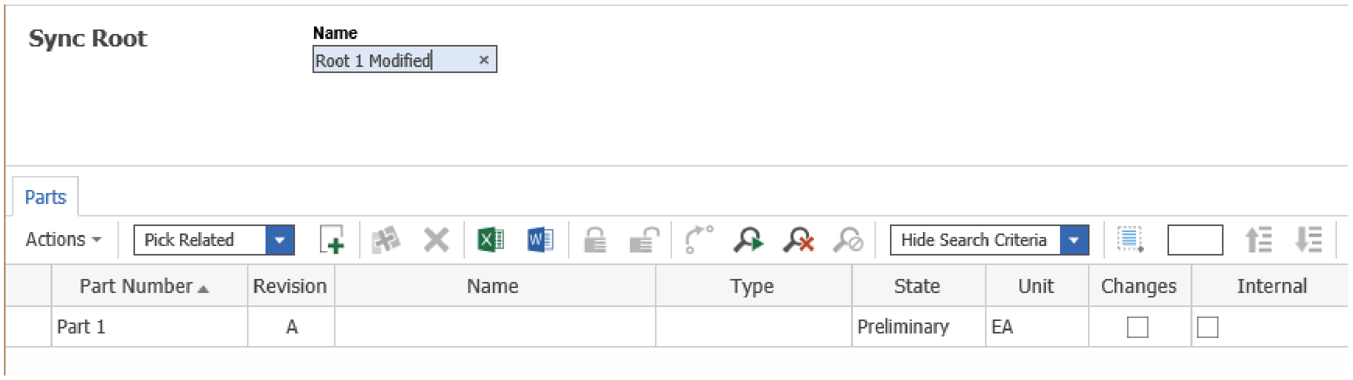
Figure 14.

4. Once the Conversion Server processes the task, you can go to the Destination Server and confirm the synchronization.

4.2.2 Updating a Synchronization Task

Use the following procedure to modify Sync Root by changing its name:

1. Enter **Root 1 Modified** in the Name field as shown.

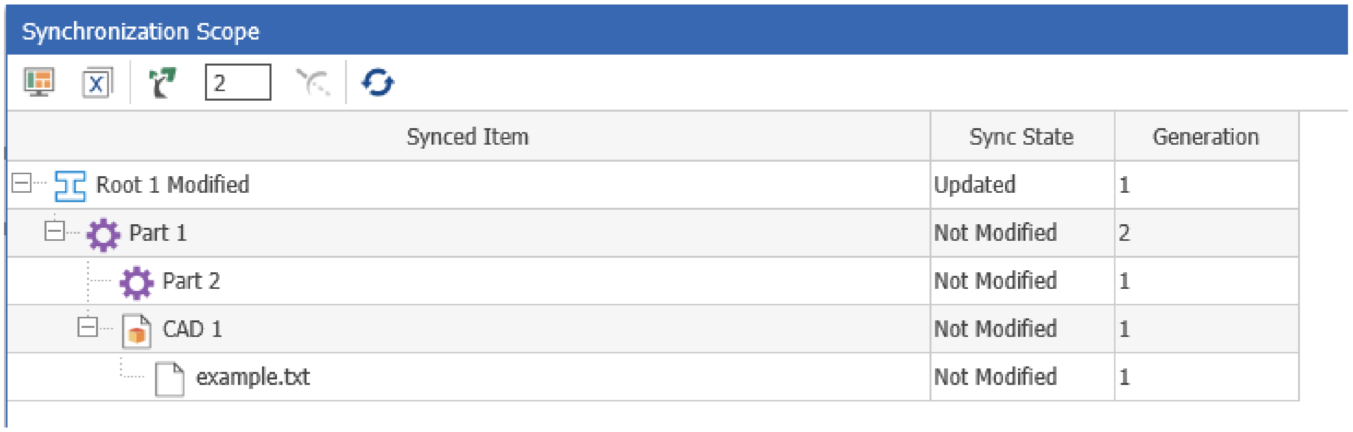

Sync Root

Name: Root 1 Modified

	Part Number	Revision	Name	Type	State	Unit	Changes	Internal
	Part 1	A			Preliminary	EA	☐	☐

Figure 15.

2. Select **Actions>Show Sync Scope**. The updated Tree Grid View appears.

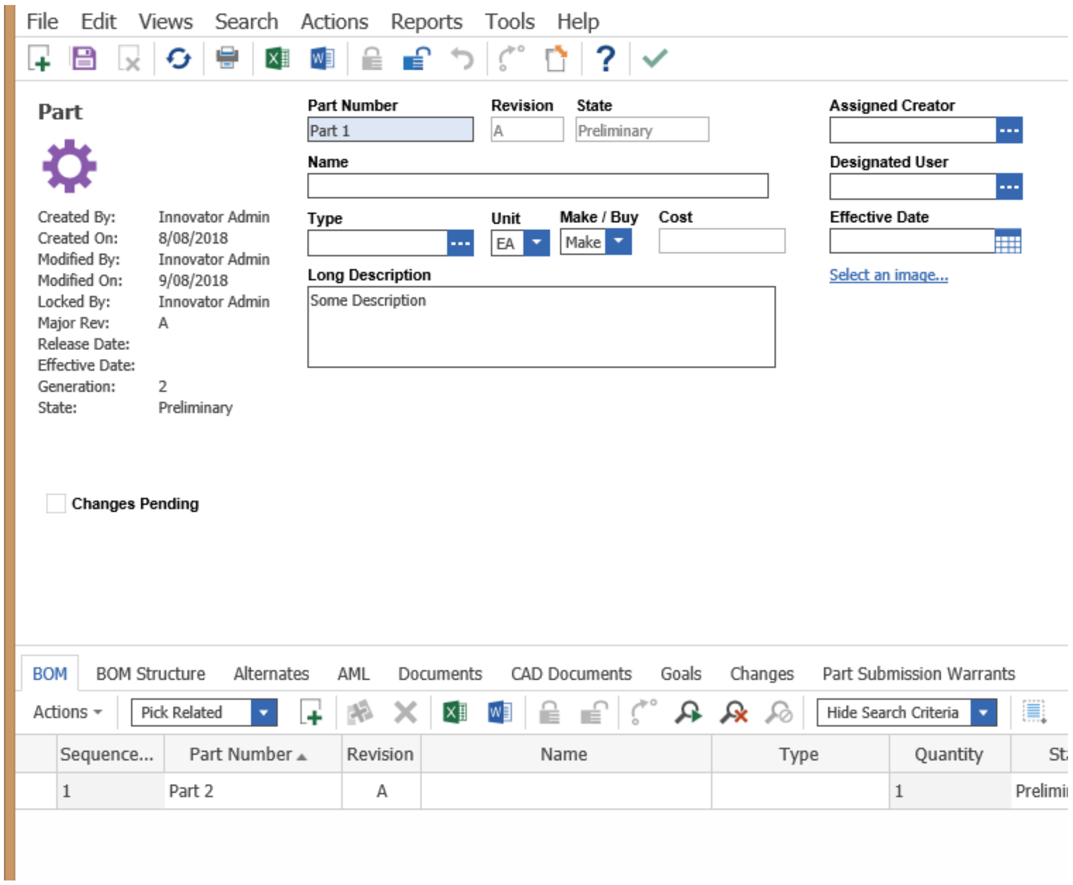

Synchronization Scope

Synced Item	Sync State	Generation
Root 1 Modified	Updated	1
Part 1	Not Modified	2
Part 2	Not Modified	1
CAD 1	Not Modified	1
example.txt	Not Modified	1

Figure 16.

Sync Root is updated and will be synchronized during the next synchronization event. Other items in the view were not modified.

3. Modify **Part 1** by adding a description as shown in the following screenshot:



Part

Created By:	Innovator Admin
Created On:	8/08/2018
Modified By:	Innovator Admin
Modified On:	9/08/2018
Locked By:	Innovator Admin
Major Rev:	A
Release Date:	
Effective Date:	
Generation:	2
State:	Preliminary

Part Number: Part 1
Revision: A
State: Preliminary
Name:
Type:
Unit: EA
Make / Buy: Make
Cost:
Long Description: Some Description
Assigned Creator:
Designated User:
Effective Date:
Select an image...

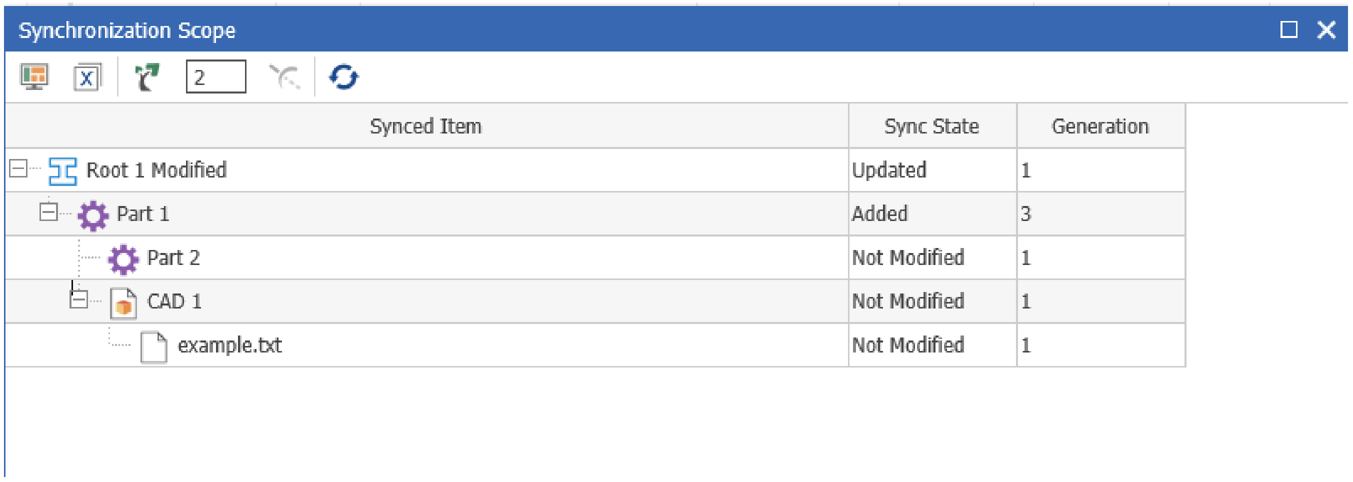
Changes Pending

BOM BOM Structure Alternates AML Documents CAD Documents Goals Changes Part Submission Warrants
Actions Pick Related Hide Search Criteria

Sequence...	Part Number	Revision	Name	Type	Quantity	St
1	Part 2	A			1	Prelimi

Figure 17.

4. Open the Tree Grid View again and look at synchronization scope.



Synced Item	Sync State	Generation
Root 1 Modified	Updated	1
Part 1	Added	3
Part 2	Not Modified	1
CAD 1	Not Modified	1
example.txt	Not Modified	1

Figure 18.

Part 1 is added on the next synchronization event. This is because Part is a versionable ItemType, which means that when you lock and modify it, a new generation of Part is created in the DB. Synchronization is configured in such a way that only the last version of Part is synchronized. That's why the Sync State shows **Added** for Part 1.

5. Select **Action>Synchronize** to synchronize your changes.

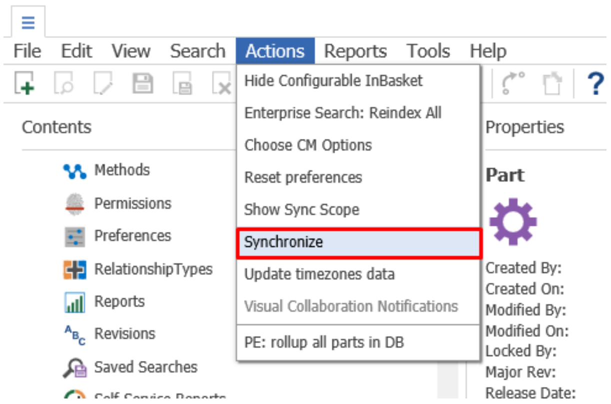
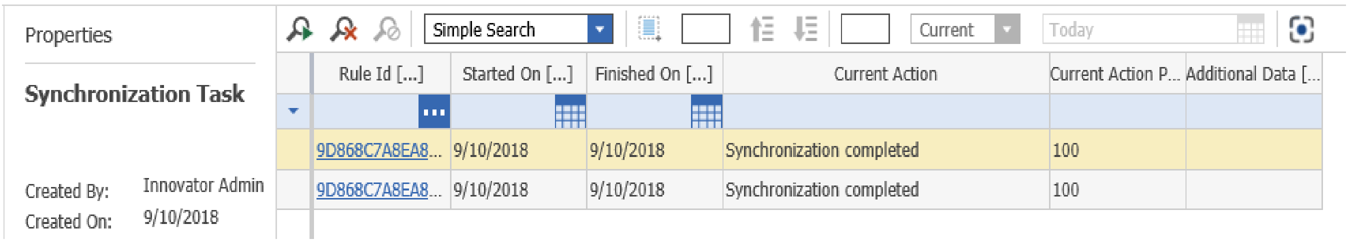


Figure 19.

6. Check that a new Synchronization Task was created and finished.



Created By: Innovator Admin
Created On: 9/10/2018

Rule Id [...]	Started On [...]	Finished On [...]	Current Action	Current Action P...	Additional Data [...
9D868C7A8EA8...	9/10/2018	9/10/2018	Synchronization completed	100	
9D868C7A8EA8...	9/10/2018	9/10/2018	Synchronization completed	100	

Figure 20.

7. Go to the Destination system and check that everything is synchronized.

5 Sample Package Technical Overview

5.1 DSS APIs

During synchronization, the Source system adds new Items from the Synchronization Scope and updates already synchronized Items on the Destination system.

Unfortunately, you cannot use the existing Item actions **add** and **update** for these operations for the following reasons:

- Synchronization may require overwriting some fields that are automatically calculated in add/update actions (e.g. id, config_id, state, generation, etc.).
- The **add/update/delete** actions on ItemTypes may have additional logic using the Server Events that may not be necessary during synchronization. For example:
    - Data validation in onBefore events – the data should be considered as already verified by the Source system.
    - Field pre-calculation in onBefore events – the data should be considered as already calculated by the Source system and the result is being passed to the Destination system for saving.
    - Sequence field value calculation.
- The **add** and **update** actions also handle infrastructural processing (LifeCycle handling, workflows start, history tracking) which may not be necessary or may be handled in a different way for synchronized Items.
- The standard **add** action treats received data as the initial generation of an Item which is not registered in the database, while during synchronization the received data may correspond to a subsequent generation and it should be written directly into the database.
- The Standard **update** action also does additional generation handling for versionable items, while for synchronization it is only required to update the record with received data.

The following Item actions were introduced in Innovator 11.0 SP15 for synchronization purposes:

- **dss_syncAdd**
- **dss_syncUpdate**
- **dss_syncPurge**

Synchronization actions perform the required Server Events *onBefore-/onAfter- Add/Update/Delete* configured for the ItemType (i.e. with `@is_required=1`). In addition, each of the specialized synchronization actions comes with a couple of corresponding *onBeforeSync-/onAfterSync-* Server Events. Where possible, the new synchronization Server Events provide the means for more efficient processing of Item sets.

Note: Please see the *Data Synchronization Service Programmer's Guide* for details.

Synchronization API actions have some common restrictions:

- Synchronization API actions can be called only by the **dss_SyncReceiver** identity (which is also introduced in Innovator 11.0 SP15).
- Synchronization API actions are supported for Table ItemTypes, where custom OnAdd, OnUpdate and OnDelete events are not defined.
- Synchronization API actions produce an error if the target Item is locked.
- Synchronization API actions produce an error in calls for Items associated with ItemTypes representing Innovator metadata (ItemType, RelationshipType, Property, Method, SystemEvent, LifeCycle, Variable, Locale, Language, Permission, Workflow, WorkflowProcess, etc.).

Warning It is assumed that permissions on Source and Destination systems are controlled separately and so it is not possible to synchronize permission items or set custom permissions on an Item via synchronization. As a result of the *dss_syncAdd* or *dss_syncUpdate* operations an Item should receive the permissions that are configured on the Destination system.

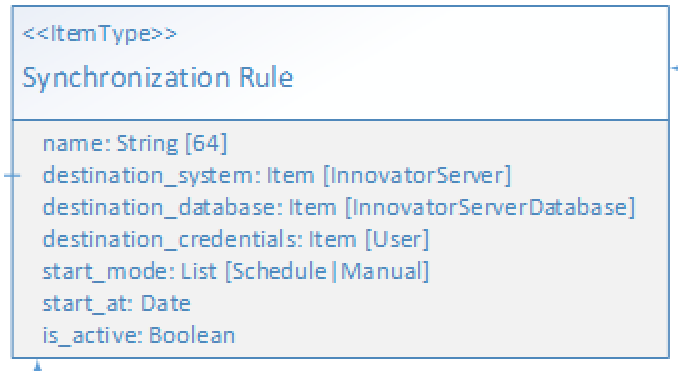
Important! Files are not immutable in Innovator. Use CheckinManager to add files and their content. The usual add action is enough for the synchronization of files. Synchronization actions do not support the File ItemType. However, ItemTypes that reference files and play the role of File Containers should be synchronized using synchronization actions.

5.2 Sample Package Data Model

5.2.1 Creating the Synchronization Rule ItemType

You need to create a Synchronization Rule ItemType that contains information about:

- What you would like to synchronize (item scope).
- Where you would like to synchronize (information about the destination instance of innovator).
- When you would like to synchronize (set up automatic synchronization, i.e. daily, weekly, etc.).

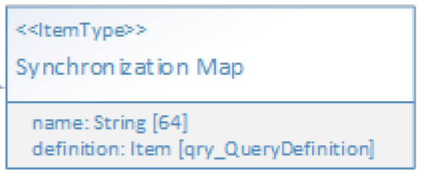


<<ItemType>>
Synchronization Rule
name: String [64]
destination_system: Item [InnovatorServer]
destination_database: Item [InnovatorServerDatabase]
destination_credentials: Item [User]
start_mode: List [Schedule|Manual]
start_at: Date
is_active: Boolean

Figure 21.

5.2.2 Using the Synchronization Map ItemType

You need to define a scope of items that you would like to synchronize. For this purpose, you can use Query Builder. It enables you to create a relationship structure of items based on customized conditions.


<<ItemType>>
Synchronization Map

name: String [64]
definition: Item [qry_QueryDefinition]

Figure 22.

The **Synchronization Map** ItemType contains a reference to the following Query Definition.

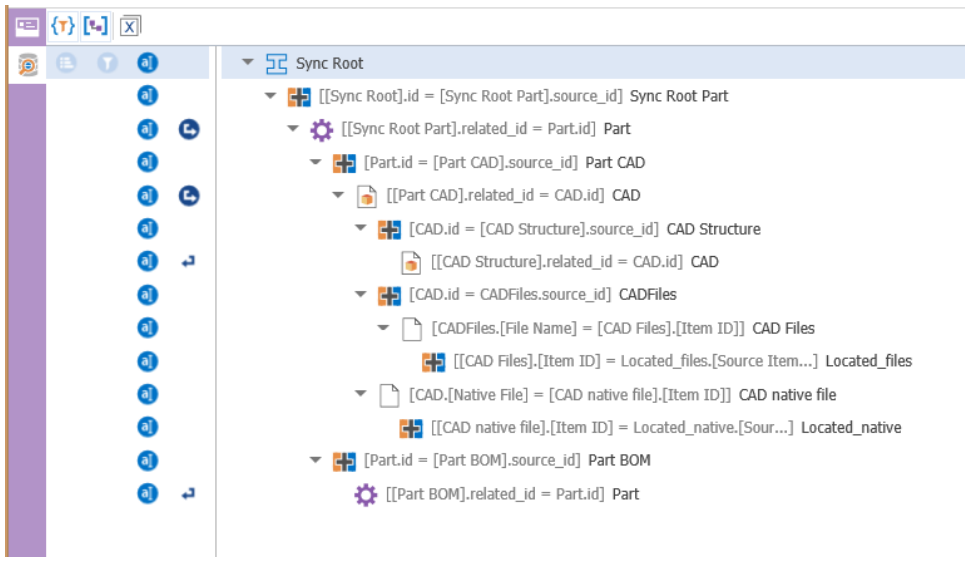
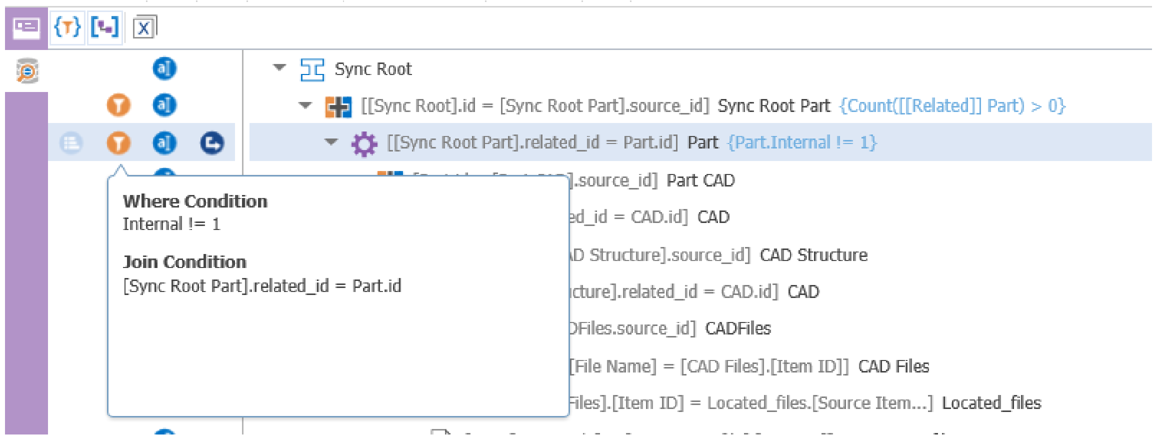

Sync Root
[[Sync Root].id = [Sync Root Part].source_id] Sync Root Part
[[Sync Root Part].related_id = Part.id] Part
[Part.id = [Part CAD].source_id] Part CAD
[[Part CAD].related_id = CAD.id] CAD
[CAD.id = [CAD Structure].source_id] CAD Structure
[[CAD Structure].related_id = CAD.id] CAD
[CAD.id = CADFiles.source_id] CADFiles
[CADFiles.[File Name] = [CAD Files].[Item ID]] CAD Files
[[CAD Files].[Item ID] = Located_files.[Source Item...] Located_files
[CAD.[Native File] = [CAD native file].[Item ID]] CAD native file
[[CAD native file].[Item ID] = Located_native.[Sour...] Located_native
[Part.id = [Part BOM].source_id] Part BOM
[[Part BOM].related_id = Part.id] Part

Figure 23.

Please notice the condition for the Internal property on Part.

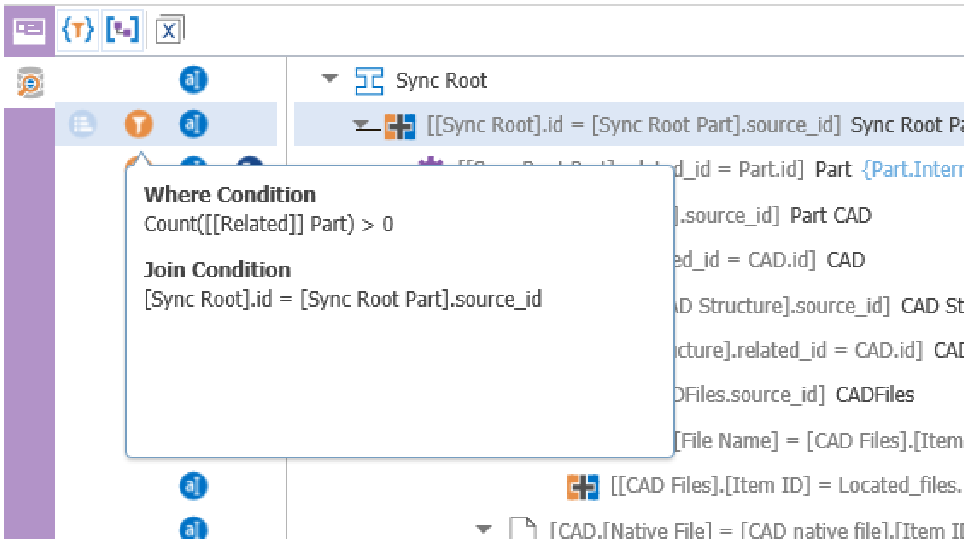

Sync Root
[[Sync Root].id = [Sync Root Part].source_id] Sync Root Part {Count([[Related]] Part) > 0}
[[Sync Root Part].related_id = Part.id] Part {Part.Internal != 1}
].source_id] Part CAD
ed_id = CAD.id] CAD
D Structure].source_id] CAD Structure
cture].related_id = CAD.id] CAD
DFiles.source_id] CADFiles
[File Name] = [CAD Files].[Item ID]] CAD Files
Files].[Item ID] = Located_files.[Source Item...] Located_files

Where Condition
Internal != 1

Join Condition
[Sync Root Part].related_id = Part.id

Figure 24.

There are also additional conditions for Count on a Sync Root Part and Part BOM nodes as shown in the following screenshots.



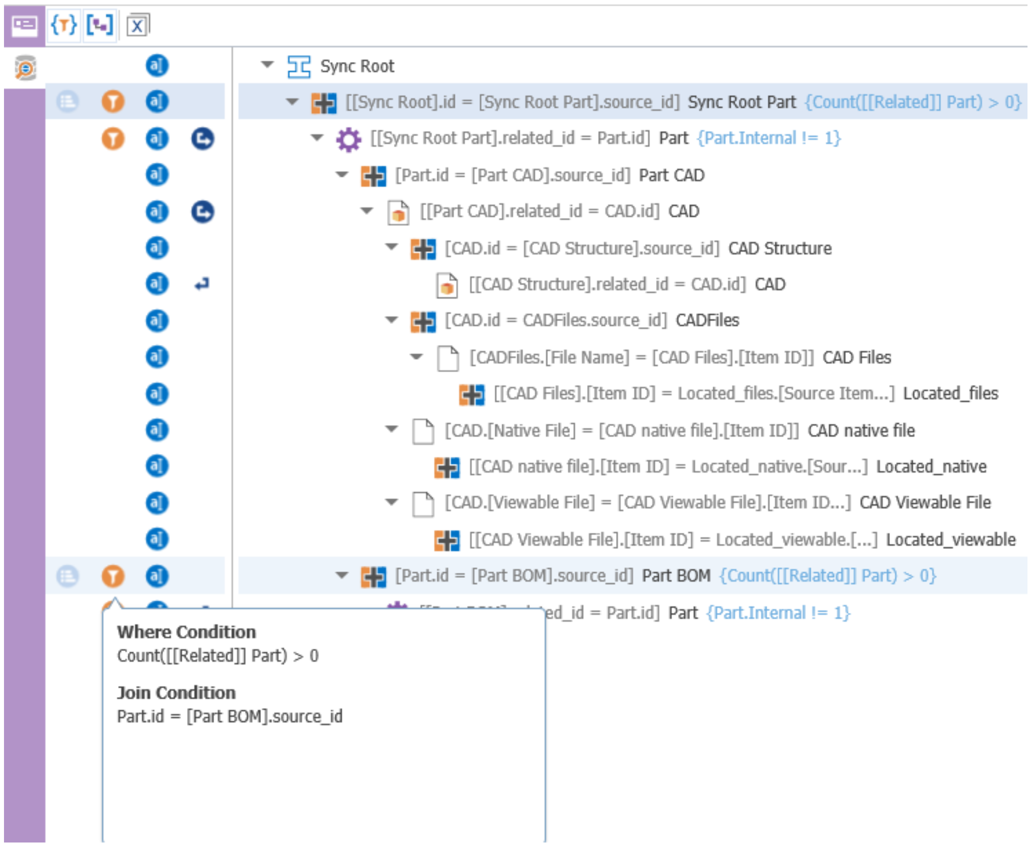
Sync Root

[[Sync Root].id = [Sync Root Part].source_id] Sync Root P

Where Condition
Count([[Related]] Part) > 0

Join Condition
[Sync Root].id = [Sync Root Part].source_id

Figure 25.



Sync Root
[[Sync Root].id = [Sync Root Part].source_id] Sync Root Part {Count([[Related]] Part) > 0}
[[Sync Root Part].related_id = Part.id] Part {Part.Internal != 1}
[Part.id = [Part CAD].source_id] Part CAD
[[Part CAD].related_id = CAD.id] CAD
[CAD.id = [CAD Structure].source_id] CAD Structure
[[CAD Structure].related_id = CAD.id] CAD
[CAD.id = CADFiles.source_id] CADFiles
[CADFiles.[File Name] = [CAD Files].[Item ID]] CAD Files
[[CAD Files].[Item ID] = Located_files.[Source Item...] Located_files
[CAD.[Native File] = [CAD native file].[Item ID]] CAD native file
[[CAD native file].[Item ID] = Located_native.[Sour...] Located_native
[CAD.[Viewable File] = [CAD Viewable File].[Item ID...] CAD Viewable File
[[CAD Viewable File].[Item ID] = Located_viewable.[...] Located_viewable
[Part.id = [Part BOM].source_id] Part BOM {Count([[Related]] Part) > 0}
ed_id = Part.id] Part {Part.Internal != 1}

Where Condition
Count([[Related]] Part) > 0

Join Condition
Part.id = [Part BOM].source_id

Figure 26.

Without these conditions, the relationship Items would be included in the execution results of the Query Definition and as a consequence in the Sync Scope, while the related Part marked as Internal would not be selected for synchronization. This could result in an error during the synchronization process when the system tries to add to the Destination relationships without the related Part. The additional conditions on the relationship nodes in the Query Definition prevents these types of errors.

5.2.3 Synchronization Scope ItemType

A Query Definition is restricted to having only one root ItemType. However, there can be situations when you need to create several Query Definitions to define a scope of items that you would like to synchronize.

For this purpose, a Synchronization Rule has a related Synchronization Scope that enables you to create several Synchronization Map Items.

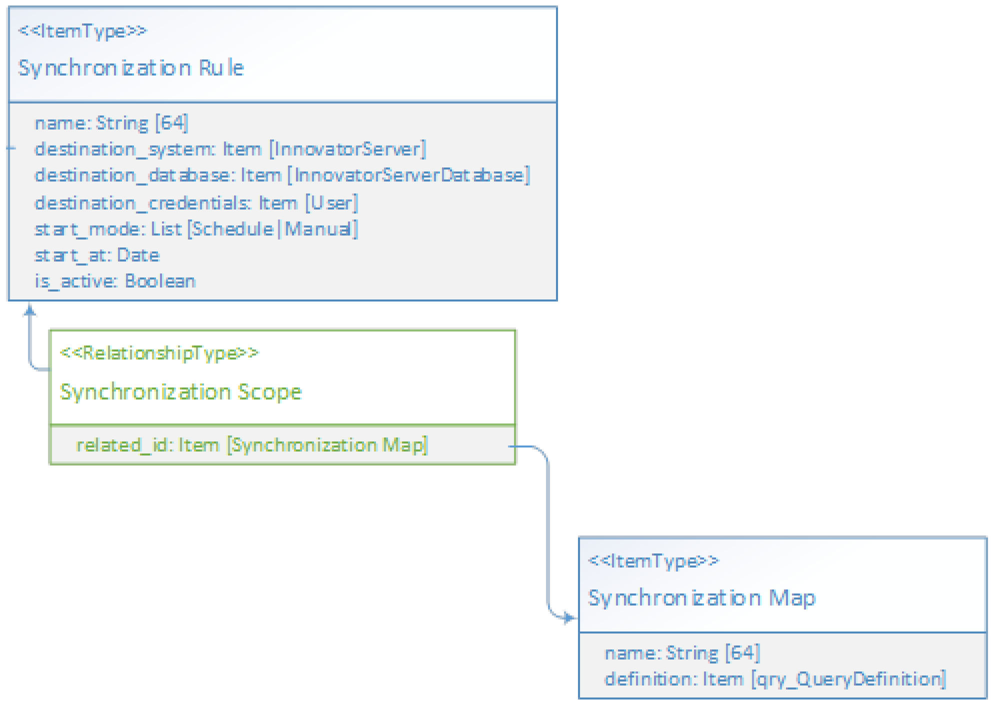

<<ItemType>>
Synchronization Rule
name: String [64]
destination_system: Item [InnovatorServer]
destination_database: Item [InnovatorServerDatabase]
destination_credentials: Item [User]
start_mode: List [Schedule|Manual]
start_at: Date
is_active: Boolean
<<RelationshipType>>
Synchronization Scope
related_id: Item [Synchronization Map]
<<ItemType>>
Synchronization Map
name: String [64]
definition: Item [qry_QueryDefinition]

Figure 27.

5.2.4 Innovator Server and Innovator Server Database ItemTypes

You need to store information about the Destination server and the database that you would like to use to synchronize data. For this purpose, you need to create the following:

- *InnovatorServer:* contains the name and URL of the destination Innovator server.
- *InnovatorServerDatabase:* provides information about the databases located on the destination server.

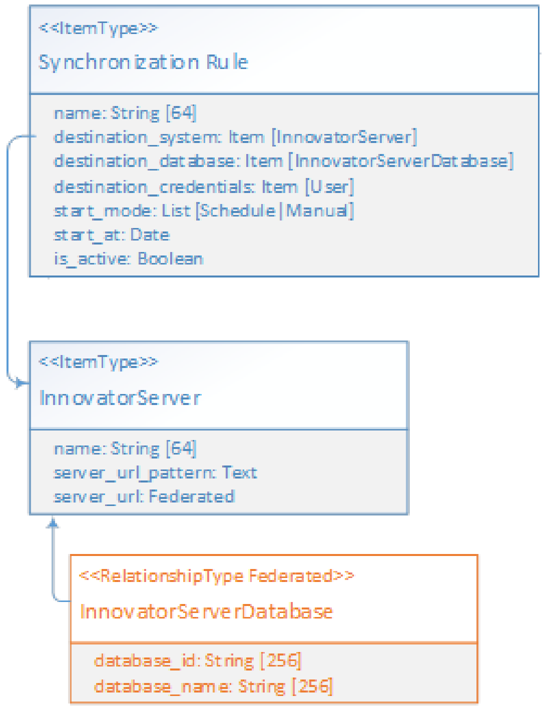

<<ItemType>>
Synchronization Rule
name: String [64]
destination_system: Item [InnovatorServer]
destination_database: Item [InnovatorServerDatabase]
destination_credentials: Item [User]
start_mode: List [Schedule|Manual]
start_at: Date
is_active: Boolean
<<ItemType>>
InnovatorServer
name: String [64]
server_url_pattern: Text
server_url: Federated
<<RelationshipType Federated>>
InnovatorServerDatabase
database_id: String [256]
database_name: String [256]

Figure 28.

5.2.5 Using the Synchronization Task ItemType

You need to store information about the synchronization process to show that it was started, processed, and finished. Use the Synchronization Task ItemType for this purpose. It contains references to the Synchronization Rule and information about the start date, finished date, and current status of the task.

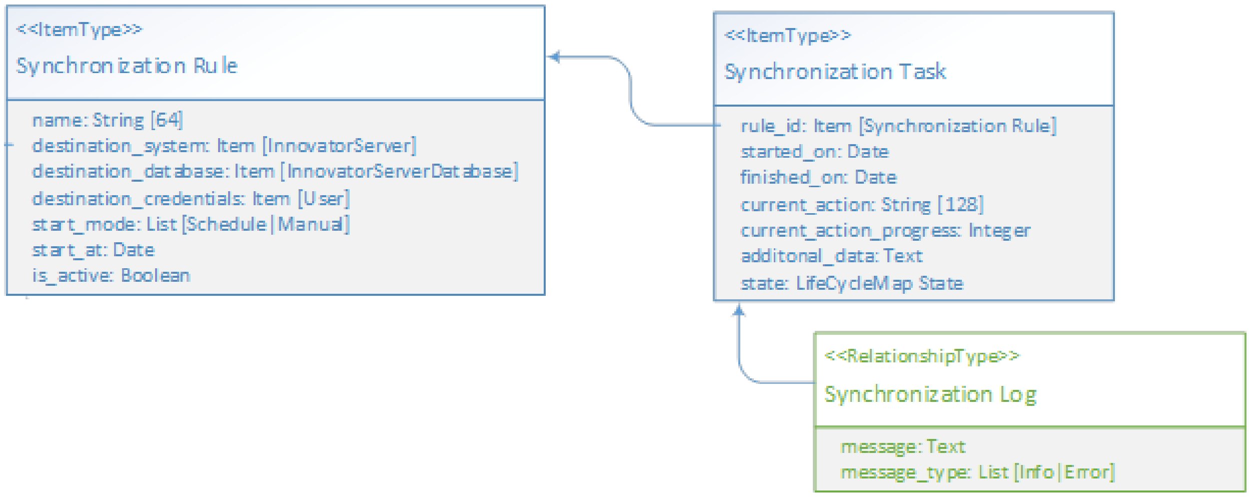

<<ItemType>>
Synchronization Rule
name: String [64]
destination_system: Item [InnovatorServer]
destination_database: Item [InnovatorServerDatabase]
destination_credentials: Item [User]
start_mode: List [Schedule|Manual]
start_at: Date
is_active: Boolean
<<ItemType>>
Synchronization Task
rule_id: Item [Synchronization Rule]
started_on: Date
finished_on: Date
current_action: String [128]
current_action_progress: Integer
additonal_data: Text
state: LifeCycleMap State
<<RelationshipType>>
Synchronization Log
message: Text
message_type: List [Info|Error]

Figure 29.

In order to have a log of particular synchronization steps, you need to create the Synchronization Log relationship.

5.2.6 Synchronization Task Processing Using Conversion Tasks

Synchronization is based on the Conversion Service. You need to create a new Conversion Rule that describes the methods to use and the business logic to perform. After installing the destination system, the admin runs a method that creates a Conversion Task that is linked to a Conversion Rule.

As soon as the Conversion Service starts working, it takes your Conversion Rule and all unprocessed Conversion tasks (at the beginning you only have one) for this rule. While processing the Conversion Task, the Conversion Server runs the **OnConvert** server event that you defined to call your Synchronization method. This method is responsible for synchronizing items by user request or by using the scheduler.

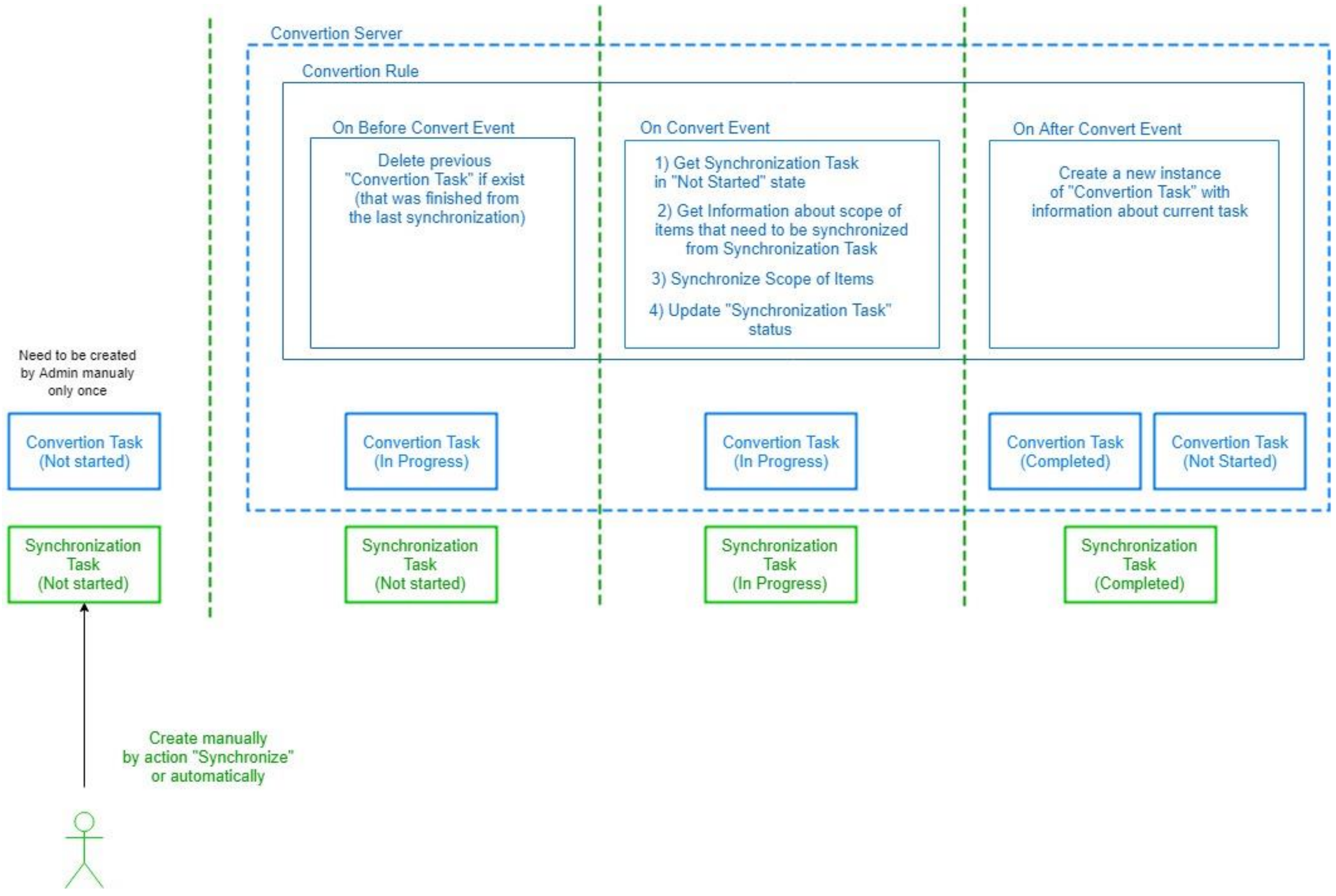

Convertion Server
Convertion Rule
On Before Convert Event
Delete previous "Convertion Task" if exist (that was finished from the last synchronization)
On Convert Event
1) Get Synchronization Task in "Not Started" state
2) Get Information about scope of items that need to be synchronized from Synchronization Task
3) Synchronize Scope of Items
4) Update "Synchronization Task" status
On After Convert Event
Create a new instance of "Convertion Task" with information about current task
Need to be created by Admin manualy only once
Convertion Task (Not started)
Convertion Task (In Progress)
Convertion Task (In Progress)
Convertion Task (Completed)
Convertion Task (Not Started)
Synchronization Task (Not started)
Synchronization Task (Not started)
Synchronization Task (In Progress)
Synchronization Task (Completed)
Create manually by action "Synchronize" or automatically

Figure 30.

5.2.7 Using the Synchronization Method

The Synchronization Method (Synchronization.xml) is responsible for the synchronization of items to the Destination system.

The method implements the following algorithm:

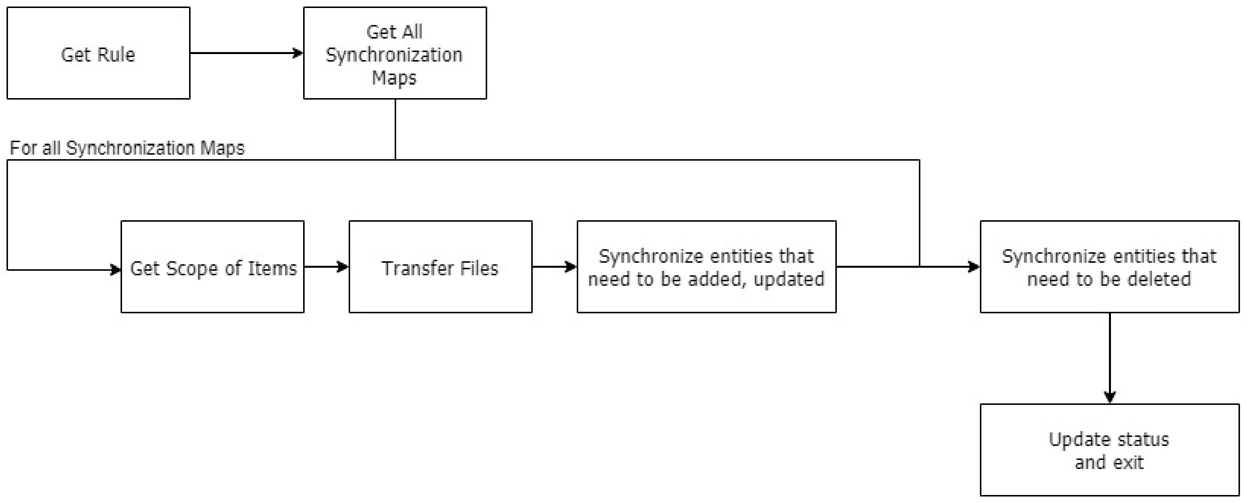

Get Rule
Get All Synchronization Maps
For all Synchronization Maps
Get Scope of Items
Transfer Files
Synchronize entities that need to be added, updated
Synchronize entities that need to be deleted
Update status and exit

Figure 31.

5.2.7.1 *Synchronization Rule and Synchronization Task information*

The input parameter for the main synchronization method is an item containing information about the Synchronization Rule and Synchronization Task (properties *syncRuleId*, *syncTaskId*). Based on this data, the method gets the particular Synchronization Rule (to get all Synchronization Maps described in the next section) and the particular Synchronization Task (to be able to update the status of this task and add logs about the synchronization steps).

5.2.7.2 *Getting all Synchronization Maps*

Based on the Synchronization Rule described in the previous section, the method gets all Synchronization Maps (Synchronization Scope Relationship). Each Synchronization Map contains a link to a Query Definition that defines a scope of items that can be synchronized to the destination system.

5.2.7.3 *Getting Full Scope of Items*

Based on a particular Synchronization Map, run the linked Query Definition and get the scope of Items. This doesn’t mean that you will add/update all of them to the Destination system (at this stage). You need to define which items need to be added/updated/deleted and which items do not need to be updated.

5.2.7.4 *Defining items that should be added, updated, deleted*

A new Data Type, Global Version, is introduced in Innovator 11.0 SP15.

The new Data Type serves as a unique identifier for a row in the database. When you change a row (ItemType instance is changed), this column (property) is automatically incremented and is unique in the database.

You can use the property of Global Version Data Type in your synchronization to understand if an instance was changed or not.

Use the Synchronization Status ItemType to store the information about Items that were synchronized. For each synchronized Item, this ItemType contains information about the Item’s type,id, and the value of the global_version property from the last synchronization (synchronized version).

When the synchronization algorithm tries to understand what Items should be added, updated or deleted, it accesses this table and checks each item to see if it has a record for the item's type, ID and synchronized version:

- If the item exists and the global version property is the same as the synchronized version, then the item is not changed and there is no need to synchronize it.
- If the item exists and the global version property has a value other than the synchronized version then the item was modified and needs to be synchronized (update).
- If the item doesn't exist in the table, it means you need to add it to the destination system.
- If an item exists in the table, but it is not in scope from Synchronization Map, it means that you need to remove it from the destination system.

Note: The global_version property is controlled by SQL and cannot be modified. Use the new Data Type, Unsigned BigInt, to store the value of this property in the Synchronization Status table.

XProperty synchronization is not supported.

5.2.7.5 *Transfer Files*

Transfer files using the default Innovator mechanism CheckIn/CheckOut managers. (`Aras.IOME.CheckinManager`, `Aras.IOME.CheckoutManager`).

Synchronization of files occurs before the synchronization of other Innovator items.

5.2.7.6 *Item synchronization*

Once Files are synchronized and you know the exact scope of items that should be added, updated, or deleted, the method generates separate requests for special add, update, and delete requests using the following actions from the Sync API introduced in Innovator 11.0 SP15:

- **dss_syncAdd** adds items to the destination system.
- **dss_syncUpdate** updates items in the destination system. This action is slightly different from the default edit action. It is more suitable for synchronization.
- **dss_syncPurge** removes items from the destination system.

In this sample, the Innovator Admin user is used to connect to the Destination server. The Innovator Admin user is added to the *dss_syncReceiver* identity on the Destination server. Since Innovator Admin is included in the Administrators group, this user has the required permissions to update the synchronized Items. For alternative ways to configure access to synced Items on the Destination system, please refer to the Data Synchronization Services Programmer's Guide

5.2.7.7 *Updating Synchronization Tasks*

As soon as the synchronization completes (successfully or not), the Synchronization Task that has been processed is updated. It contains the synchronization details.

The synchronization procedure should create Task Log entries ("Synchronization Log") similar to the following, which describe the main events and the resulting summary.

```
·    Info | Synchronization completed. Items in Scope (N). Items Processed: Added (N), Failed
to Add (N), Updated (N), Failed to Update (N), Deleted (N), Failed to Delete (N).

·    Error | Failed to add <Item Type> item ID <ID>. Reason: <E.g. Error from server>
```

6 Potential Solution Risks

This solution package serves as an example. You can modify it to create custom solutions. However, it is important to note the potential risks of using the associated data model and the DSS APIs.

6.1 Query Definition Performance

While the Query Definition is a very elegant way to describe the Synchronization Scope, the solution depends on the query evaluation mechanisms. In cases where a large number of ItemTypes need to be synchronized, the queries in Synchronization Maps are complex and a lot of Items can match these queries. Real applications should consider different approaches to bypass these limitations.

In general, alternative solutions for the Source include using plain AML **get** requests to retrieve Items for Sync Scope.

6.2 Order of Related Items and Relationships Synchronization

The order in which Items and their associated relationships are synchronized is important. The synchronization process should be able to handle cases where an Item that is being synchronized references Items that have not been synchronized yet. For example:

- Circular dependencies in Relationships (for example, Part – Part BOM – Part).
- An Item is being synchronized earlier than another Item referenced by one of the Item's properties.
- A Relationship Item is being synchronized earlier than the source Item.
- A Relationship Item is being synchronized earlier than the related Item.
- An Item referenced in an Item property does not match the query in the Synchronization Map and won't be synchronized at all.
- A source Item in a Relationship does not match the query in the Synchronization Map and therefore the Relationship is not complete.
- A related Item in a Relationship does not match the query in the Synchronization Map and therefore the Relationship is not complete.

There are different approaches to bypass these limitations.

Synchronization configurators can manually prepare several Synchronization Maps so the query definitions contained in them do not include recursion and describe synchronization of relationships separately. These maps can be included in a single Sync Rule in the appropriate order. An example of such a separation was described in 5.2.2 Using the Synchronization Map ItemType.

The synchronization algorithm ideally should be able to perform topological sorting over the Items in the Synchronization Scope. However, the whole Synchronization Scope may be too big and this operation may be too time consuming. Therefore, the synchronization algorithm should combine different techniques to process the Synchronization Scope more effectively. It may try to apply topological sorting at least when flattening relationships of a particular Item to the Synchronization Scope. Then it may make several attempts to synchronize the Incremental Synchronization Scope, each time skipping those Items that cannot be synchronized in the current step and returning to them on the next iteration (obviously, the maximum number of iterations either per the whole Sync Scope or per a concrete Item will be defined to avoid endless loops).

The way AML instructions with synchronization actions are sent to the Destination server (either in a one-by-one or all-in-one request, with or without a reference to a source Item) may lead to violation of the Min/Max Occurs on relationship and Dependent requirement settings of an ItemType because they are checked in the same way as *add/update/delete* actions. As a workaround for potential errors, it is enough to clear these settings on the Destination System only. In this case, the Source system is responsible for passing all the required Items to the Destination to guarantee data consistency as soon as the synchronization is completed.

6.3 Special Handling of some Property Types

6.3.1 File Synchronization for Image Properties

Synchronization of Image properties is not supported.

6.3.2 Synchronization of Sequence Properties

Sequence property Values are generated just once at the moment of Item creation. The generated string value is then saved to the Item for which it was created. The Sequence is also updated when its current value is incremented.

During the synchronization an Item is sent to the Destination System with the already generated string value for the Sequence property and is written as is. However, in this case the corresponding Sequence is not updated on the Destination System. In theory, a similar situation may also occur if an Administrator on the Destination System resets the current value for the Sequence manually.

There should be no conflict in cases where a Sequence is being used by one ItemType that contains synchronized items, as long as the synchronization is done in one direction and the items in the Destination system are read-only.

If the Sequence is shared between both synchronized and unsynchronized ItemTypes there should be no conflict unless there is a business need to have unique property values for different ItemTypes. To keep everything consistent, once an Item with a Sequence property is added to the Destination system during synchronization, the property value could be parsed in oAfterSyncAdd/Update server events to extract the Sequence value. If this value is greater than the current Sequence value, it should be set to this new value. By default, dss_syncAdd and dss_syncUpdate do not support the adjustment of a Sequence’s current value.

6.3.3 Synchronization of Multilingual Properties

Items containing multilingual properties on the Source system should be loaded for synchronization with all available languages (`@language=”*”`). Otherwise, only the translation for the current locale will be loaded and transferred to the Destination system.

6.3.4 LifeCycle State Processing for Synchronized Items on the Destination System

An Item has the following properties referencing its current State:

- *current_state* - ID of the current LifeCycleState Item.
- *state* – The name of the current state.

The assumption that Items have preset common metadata in the Source and Destination systems means that they have identical LifeCycle state names and state IDs. By default, the Destination system ignores the *state* property and validates the *current_state* property against the LifeCycle found in the Destination system.

Another use case is that an Item being synchronized may have some historical LifeCycle state data (e.g. a released Item with a reference to an old LifeCycle). If such a LifeCycle state is also found in the Destination System, it will be used.

If the Source and Destination systems have different LifeCycle metadata for the same ItemType, you need to use the *onBeforeSyncAdd*/*onBeforeSyncUpdate* server event handlers to customize the correct data mapping for the *current_state* and *state* properties to make sure the data is valid for the Destination system.

LifeCycles and ItemType configurations may change over time on the Source System. If consumers of DSS support these changes, they have to be reflected in the metadata on the Destination system. DSS does not validate consistency between LifeCycle states and related fields (*classification, is_released*, etc.) on the Destination system in synchronization actions. If required, specific custom validation can be implemented using the *onBeforeSyncAdd/onBeforeSyncUpdate* or *onAfterSyncAdd/onAfterSyncUpdate* Server Event handlers.

6.3.5 Validation of generation and config_id Properties

In Aras Innovator, the system core is the only source of values for the *config_id* and *generation* properties associated with an Item. It is responsible for the initialization and consistency of these properties. It guarantees the proper sequence of values for the *generation* property based on the value of the previous item generation at the moment of Item versioning. You cannot modify the property values using other external means.

The synchronization algorithm enables you to synchronize a specific generation of an Item. It also enables you to send the generations out of sequence.

In circumstances like this, the Source system is the place that controls the initialization of values for the *generation* and *config_id* properties.

Unfortunately, the Destination system cannot guarantee the consistency of the values on the synchronized Items. It can only accept these property values from the Source system and verify that:

- the provided *generation* is a positive number.
- the *id* and *config_id* match the rules for the ID value.
- it is the first *generation* (1), and the *id* and *config_id* are equal.
- the combination of *config_id* and *generation* is unique.

The last check may potentially lead to issues with performance because with each *syncAdd* action the Item table is scanned to find duplicates. To avoid these kinds of issues, this check is omitted in the dss_syncAdd action. If required, it is possible to use the *onBeforeSyncAdd* event to add it for custom solutions.

The Source system may be not necessarily be Aras Innovator. It may be another PLM system that performs synchronization using a custom implementation that sends AML requests with actions of Sync PI. If it is necessary to emulate Aras Innovator's work, this custom implementation of the non-Innovator Source system has to generate consistent values for the *id*, *config_id* and *generation* properties. Alternatively, the values for the properties can be generated in the *onBeforeSyncAdd* event handler in the Destination system. The proper initialization of values is the responsibility of the custom solution.

6.3.6 Performance Issues for Processing Large Datasets

When attempting to synchronize a large number of Items in the Sync Scope, the synchronization method can take a long time to process on the server. The synchronization process can also create a large number of internal AML requests for synchronization operations and updates in the Sync Status. By default, the Aras Innovator server has both a timeout limit for server requests and a max request length. To avoid potential issues, set the following parameters:

1. Open **Innovator\Server\web.config**.
2. Find the **<system.web>** section.
3. Set the **maxRequestLength** attribute to 4194304 in the **< httpRuntime >** property and adjust the executionTimeout attribute to 3600 (1hour):

   ```
   <httpRuntime maxRequestLength="4194304" executionTimeout="3600"
   targetFramework="4.5.2" />
   ```

4. Add the following to the **<system.webServer>** section:

   ```
   <security>
      <requestFiltering>
         <requestLimits maxAllowedContentLength="4294967295" />
      </requestFiltering>
   </security>
   ```