Aras Innovator 30

Data Synchronization Service

Programmer's Guide

Document #: *D-008101*

Last Modified: *11/15/2023*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Phone: 978-691-8900

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:
TechDocs@aras.com
Subject: Aras Product Documentation

Or,

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

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

Table 2 describes the terms used in the Programmer's Guide.

Table 1: Terms

Term	Definition
DSS	Data Synchronization Service. 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 which will are being synchronized between the Source and Destination systems.

2 Overview

The Data Synchronization Service (DSS) provides Item Actions and Events that you can use to create an environment where data from one Aras Innovator instance can be synchronized with another Aras Innovator instance. These actions and events target the Destination system and are used to add, update, or delete data passed from the Source system to the Destination system. This document contains a detailed API guide for the Item Actions and Events.

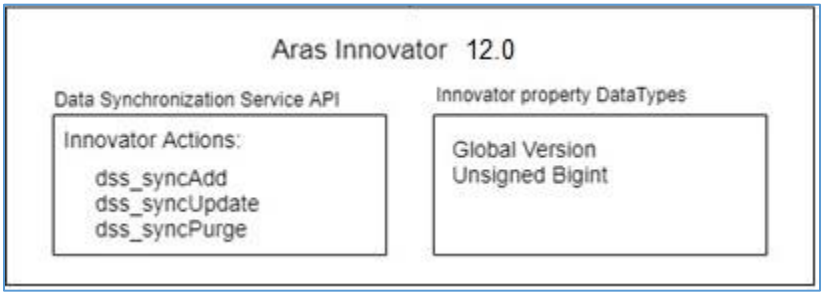

Figure 1.

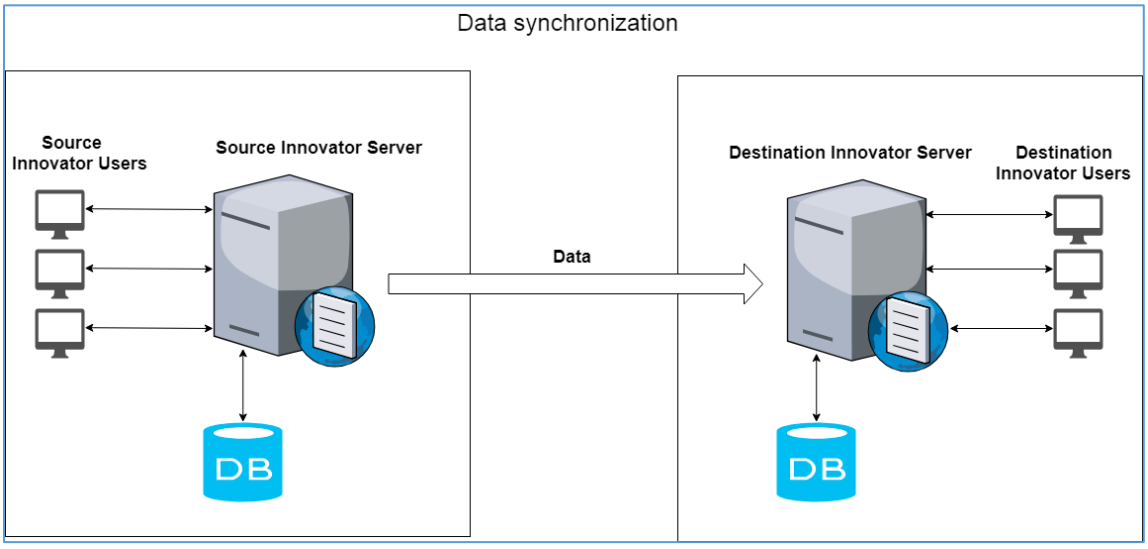

Figure 2.

3 API Guide

3.1 Overview

During synchronization, the Source system adds new Items from a 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 (for example: *id*, *config_id*, *state*, *generation*, etc.).
- The **add/update/delete** actions on ItemTypes may contain additional logic using 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. The result is 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 be unnecessary or 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. During synchronization the received data may correspond to a higher generation and should be written directly into the database.
- The Standard **update** action also performs additional generation handling for versionable Items, while for synchronization it is only necessary to update the record with the received data.

The following Item Actions were introduced in Aras 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 come with a couple of corresponding *onBeforeSync-/onAfterSync-* Server Events (see New Server Events for Synchronization). Where possible, the new synchronization Server Events provide the means for more efficient processing of Item sets.

In addition, two new Data Types were introduced to enable tracking of item changes on the source system:

- Global Version
- Unsigned BigInt

Synchronization API actions have the following common restrictions:

- They can only be called by the **dss_SyncReceiver** Identity (which is also introduced in Aras Innovator 11.0 SP15)

- They are supported for **Table** ItemTypes, where custom **OnAdd**, **OnUpdate**, and **OnDelete** events are not defined.
- They return an error if the target Item is locked.
- They return an error in calls for Items associated with ItemTypes representing Aras 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 therefore it is not possible to synchronize permission items or set custom permissions on an Item using synchronization. As 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 immutable in Aras Innovator. Use the *CheckInManager* to add files and their associated content. The **add** action is sufficient for the synchronization of Files. The File ItemType is not supported by synchronization actions. However, ItemTypes that reference Files and play the role of File Containers should be synchronized using synchronization actions.

3.2 Data Types

There are two new Aras Innovator Data Types that were introduced in 11.0 SP15 that enable the tracking of changes to items on a Source system. These Data Types are very important in order to monitor and maintain synchronization between the Source and Destination systems.

3.2.1 Global Version

The Global Version is a unique value across the entire database that should be attached to every ItemType that is intended to be synchronized. ItemTypes that will be synched should have the optional global_version property attached that will be incremented any time the item is updated. The value of a global_version property is handled by the system and cannot be modified by an administrator. In order to ensure that the value is unique, the Data Type supports values between 0 and 2^{64}-1.

3.2.2 Unsigned BigInt

The Global Version is a unique Data Type that is only used for the global_version property. In order to store and compare the current global_version value with a previously stored version, the Unsigned BigInt Data Type is introduced in order to be able to track the Global Version of an item. Since the value of a global_version property can range from 0 to 2^{64}-1, an Integer will not support values large enough to store and compare a global_version value, requiring the support for an Unsigned BigInt.

3.3 Identities

3.3.1 dss_SyncReceiver

Data synchronization is a low-level, specialized process. Therefore, permissions to configure and run it should be granted carefully. The new **dss_SyncReceiver** Identity can receive synchronization requests and perform Item synchronization on the Destination system.

When applying synchronization requests, you must have create/update/delete permissions for Items associated with ItemTypes that come from a particular Source system. By default, **dss_syncReceiver** does not have these types of permissions. When preparing the Destination system for synchronization the Administrators of that system should give corresponding permissions either to the entire **dss_SyncReceiver** Identity or to a particular User Identity that will be used for synchronization requests from a particular Source system. Although this type of approach requires some time for manual configuration, it has the following advantages:

- It controls the amount of data that can be changed using the Synchronization API. It is only possible to modify Items associated with ItemTypes that have been specially configured for that.
- It is possible to receive synchronization requests from different Source systems on the same Destination System and ensure that they will modify different sets of Items.

3.3.2 Examples of Authentication and Permission Set up on Destination System for Synchronization Requests

Figure 3 shows a One-Source system where **dss_SyncReceiver** is an Alias Identity for sync request authentication User.

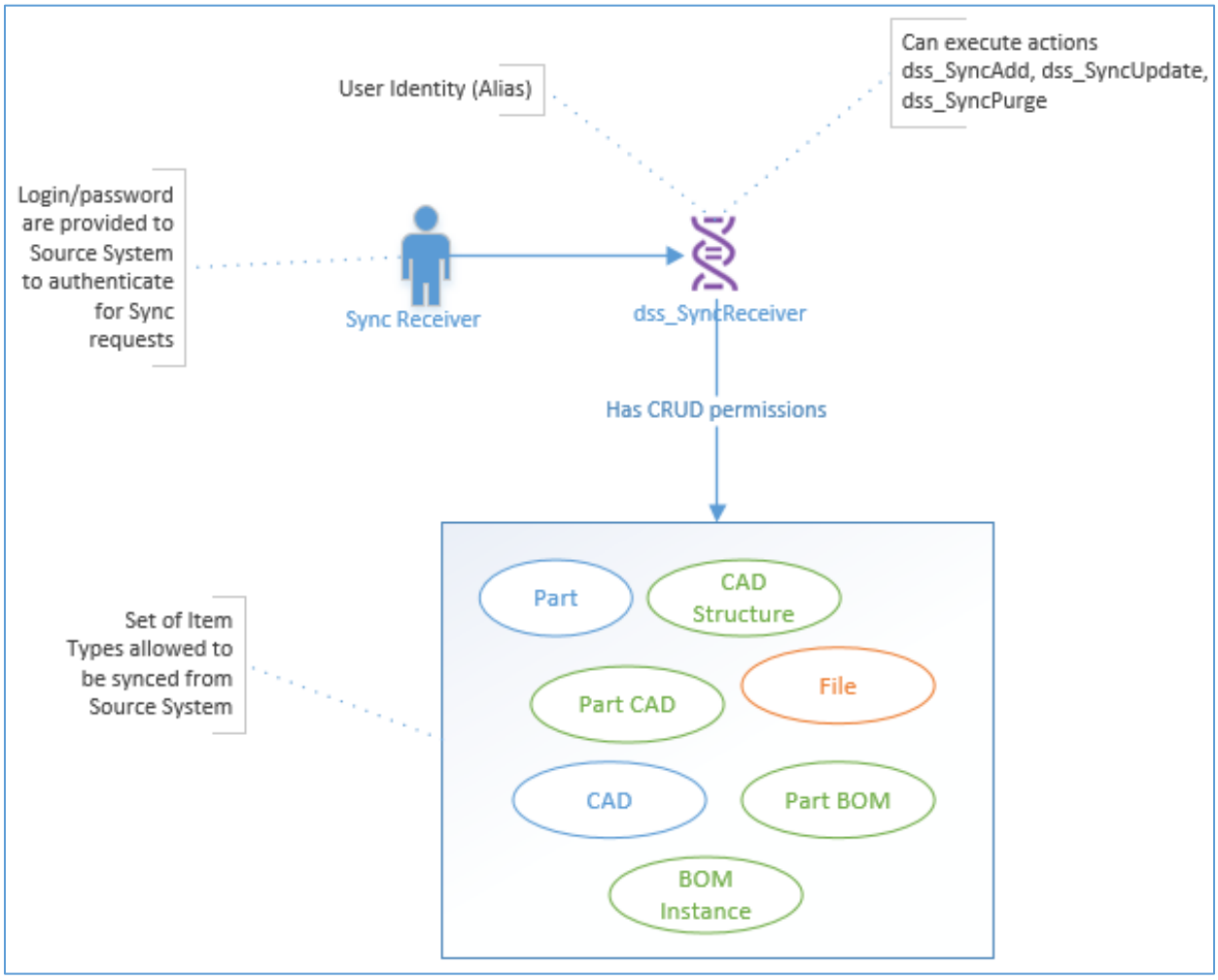

Figure 3.

Figure 4 shows a One Source system where Sync request authentication User Identity is a member of **dss_SyncReceiver**.

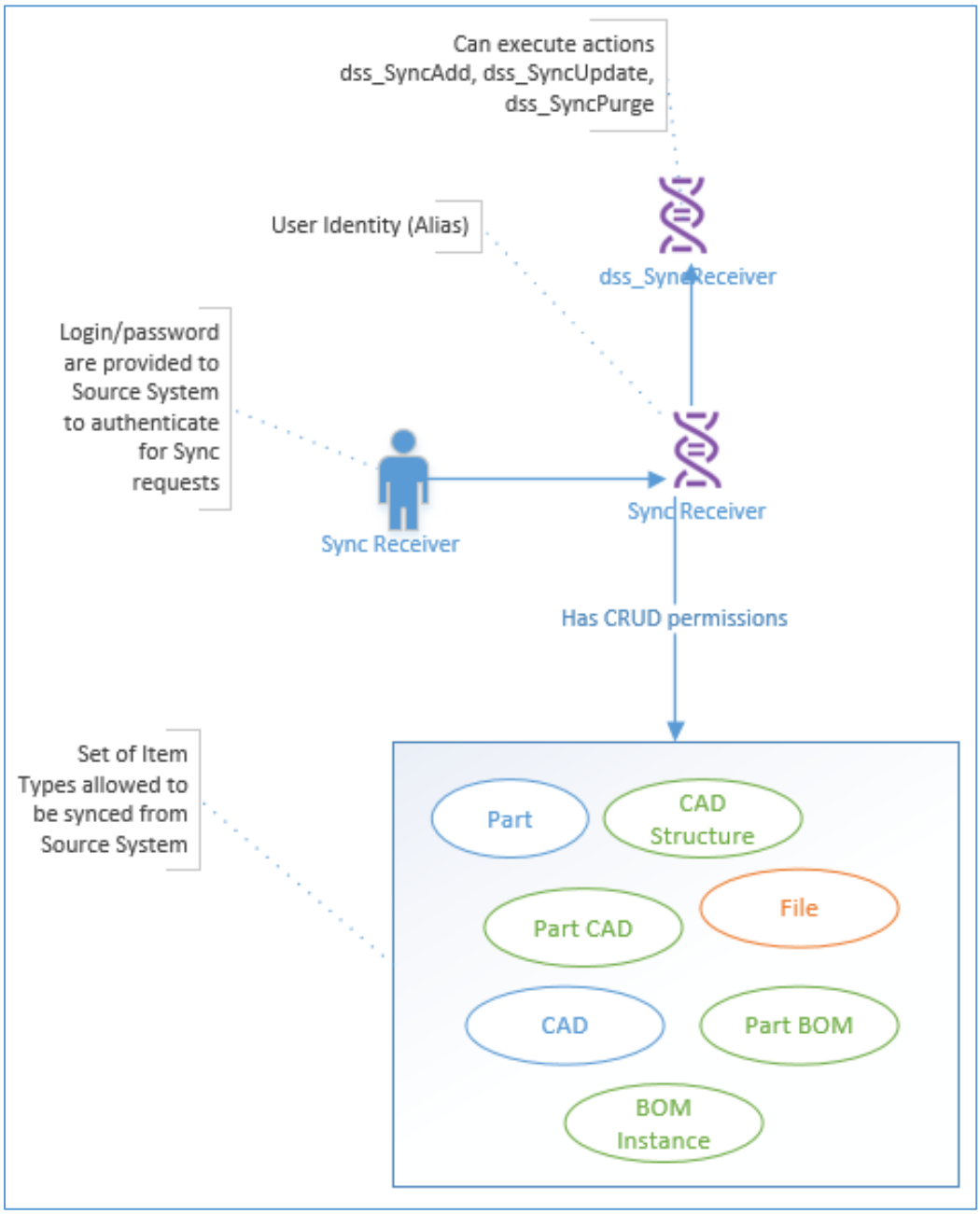

Figure 4.

Figure 5 shows two Source systems with different sets of synced Item types and two sync request authentication Users.

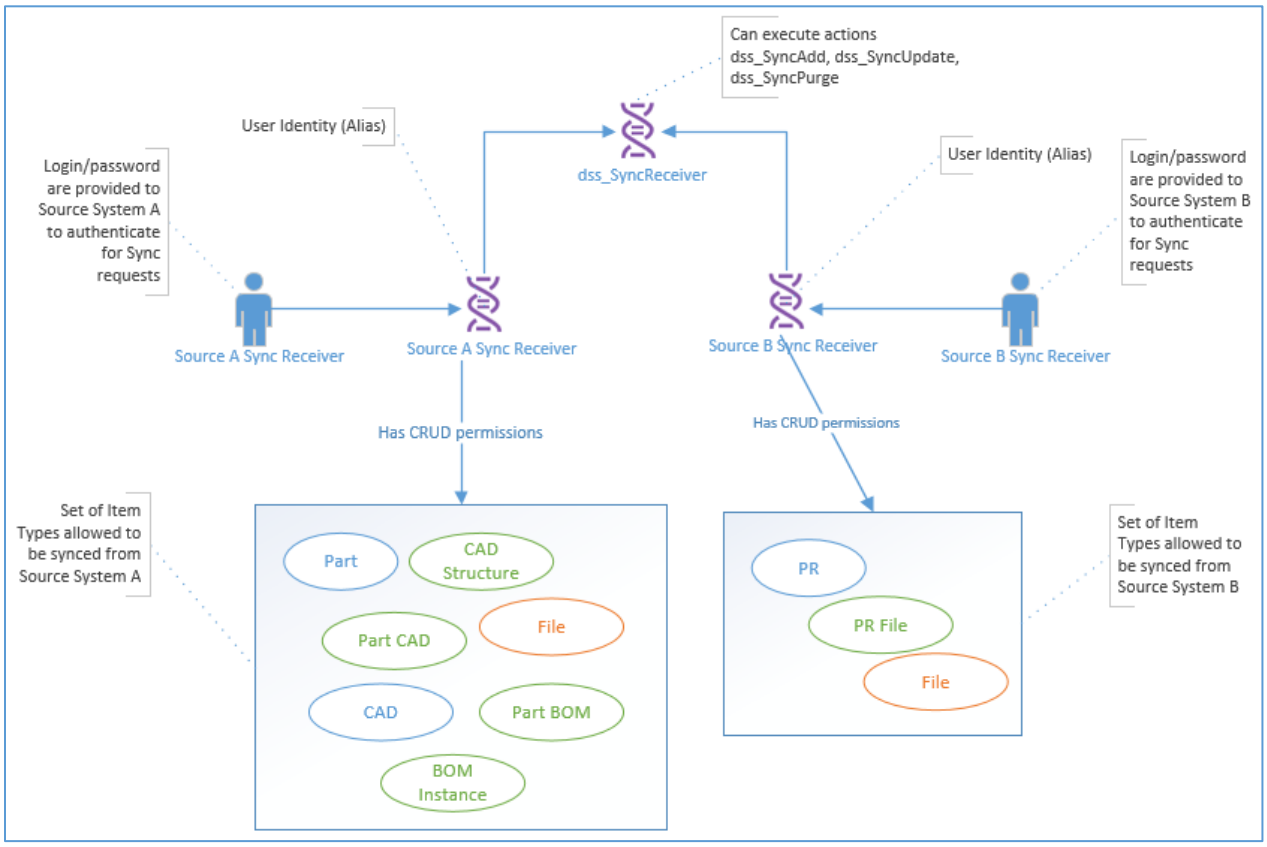

Figure 5.

3.4 Item Actions

3.4.1 dss_syncAdd

AML

```
<Item type='Part' id='7073098207234317BBC2CA865413CAD5' action='dss_syncAdd'>
  <config_id>7073098207234317BBC2CA865413CAD5</config_id>
  <generation>1</generation>
  <major_rev>A</major_rev>

  <!-- Item properties with sync data-->
  <item_number>PA-1586-0</item_number>
  <name>Engine</name>
  <!-- … the rest of Item props to be synced -->
</Item>
```

Implementation Details

The main differences between the **add** action and **dss_syncAdd** are **dss_syncAdd**:

- Requires **id**, **generation**, and **config_id** values to be provided for the Item. **dss_syncAdd** does not generate an ID.
- Skips determination of the initial LifeCycle State.
- Allows the setting of system properties such as **created_on**, **modified_on**, etc.
- The default **onBeforeAdd/onAfterAdd** events are not called. Use the new server events **onBeforeSyncAdd/onAfterSyncAdd** instead.

Generally, **dss_syncAdd** directly saves the provided properties to the database.

Table 3 describes the processing of system properties in the **dss_syncAdd** action.

Table 2: The **dss_syncAdd** action processing of system properties

System Property	Required *	Validation in syncAdd	Comments
id	Yes**	Is not null. Has valid ID format.	Saved as is. Since synchronization is run per Item generation, it is required that properties **id**, **config_id**, and **generation** are provided from the Source system or at least set inside the **onBeforeSyncAdd** event handler. If any of these properties are missing, an error occurs and the Item is not saved.
config_id	Yes	Is not null. Has valid ID format. If **generation** is **1**, **config_id** should be the same as **id**.	
generation	Yes	Is not null. Is a positive number.	

System Property	Required *	Validation in syncAdd	Comments
major_rev	Yes	Is not null. If the ItemType has a Revision List, the property value should match its values; otherwise, it should be empty (to match the current logic of **major_rev** initialization in such cases)	Saved as is.
locked_by_id	No	-	Value provided from Source is ignored. Because the initial DSS implementation only supports read-only data being passed to the Destination system, the **locked_by_id** property on the synchronized Item is set to **null** as a result of the synchronization action.
is_current	No	-	Value provided from Source is ignored. The value for the property is set by the Destination system: • For a non-Versionable ItemType the value is set to 1 (as there can only be one generation per Item). • For a Versionable ItemType the **is_current** property is adjusted on the whole sequence of generations sharing the same **config_id**. **is_current** is set to 1 only on the last **generation**.

System Property	Required *	Validation in syncAdd	Comments
new_version	**No**	-	Value provided from Source is ignored. The **new_version** property is an internal system property required for proper update handling for versionable Items. For the **add** action, it is closely coupled with the value for the current item lock state and depending on the versioning method, may dictate if a new Item version should be created to save changes to the Item. For **dss_syncAdd** any passed value for this property from Source is ignored. The value is set on the Destination system. The resulting value in this property after **dss_syncAdd** is calculated in the same way that it is done for the *add* action with the difference being that **dss_syncAdd** does not lock the item.
classification	**No**	If provided, should match class structure declared on an ItemType	Saved as is.
current_state	No	If value is not null, it should correspond to the ID of the lifecycle state from the Destination system.	If valid, saved as is. If not provided, should be set to **null**.
state	No		Since *state* contains the name of an LCS referenced via the **current_state** property, the value will be overridden: If **current_state** is null (and this is valid), *state* property is also set to null Otherwise, it gets the name of the LC state corresponding to **current_state** on the Destination system.

System Property	Required *	Validation in syncAdd	Comments
is_released	**No**	-	Saved as is. If not provided, set to **False** = **0**.
released_date, effective_date	**No**	If provided for a released Versionable Item it should not be null and should be in a valid date format	Saved as is.
permission_id	**No**	-	Value provided from Source is ignored. If required, it is possible to add the **permission_id** property to the Item in the **onBeforeSyncAdd** event handler on the Destination system.
keyed_name	No	-	If the **keyed_name** is provided and is not null or empty, it is used as is. Otherwise it is calculated in the same way it is done for the **add** action. Assuming that the Source and Destination systems have the same metadata, the result should be the same.
created_by_id, modified_by_id	No	-	Saved as is if not null or empty, otherwise should be initialized using the same logic as the **add** action. Source values for the obligatory **modified_by_id**, **created_by_id** and optional **owned_by_id**, **managed_by_id**, **team_id** may not correspond to the user/team items in the Destination database. It may break permissions calculation for dynamic roles and links in the UI. However, if required, the values from the Source system can be replaced with local Destination values in the **onBeforeSyncAdd** event handler.

System Property	Required *	Validation in syncAdd	Comments
created_on, modified_on	No	If provided, should be a valid date.	Saved as is if not null or empty. Otherwise, it is initialized using the same logic as the **add** action.
sort_order	No	-	The processing of **sort_order** is the same as the **add** action: • If passed from Source, **sort_order** on Relationship Items is written on Destination as is. • If it is not provided, the value should be calculated.

* - Required properties should be provided with an Item, otherwise an error is returned.
** - For compatibility with the format of the standard **add** action, the **id** property is expected to be provided for an Item as an attribute in the AML request. The `<id/>` property tag is ignored.

3.4.2 dss_syncUpdate

AML

```
<Item type="Part" id="70730982072343l7BBC2CA865413CAD5" action="dss_syncUpdate">
  <!-- Item properties with sync data-->
  <item_number>PA-1586-0</item_number>
  <name>Main Engine</name>
  <!-- … the rest of Item props to be synced -->
</Item>
```

Implementation Details

The main differences from **update** are that **dss_syncUpdate**:

- Allows the setting of system properties like **modified_on**, **modified_by_id**.
- Does not handle the float behavior of Relationships.
- The default **onBeforeUpdate**/**onAfterUpdate** events are not called. Instead, the new server events **onBeforeSyncUpdate**/**onAfterSyncUpdate** should be used.
- Does not handle versioning for Versionable Items.

Generally, **dss_syncUpdate** directly saves the provided properties to the database on the Item generation corresponding to the provided Item ID.

Table 4 describes the processing of system properties in the **dss_syncUpdate** action.

Table 3: The **dss_syncUpdate** action processing of system properties

System Property	Required *	Validation in syncUpdate	Comments
config_id **generation** **itemtype** **created_on** **created_by_id** **relationship_id** **is_current** **source_id**	No	-	Are ignored.
major_rev	No	If provided, validated that: It is not null. If ItemType has Revision List, then the property value should match its values. Otherwise, it should be empty (to match the current logic of **major_rev** initialization in such cases)	Saved as is.
locked_by_id	No	-	Value provided from Source is ignored. Since the initial DSS implementation only supports read-only data being passed to the Destination system, the **locked_by_id** property on the synchronized Item is set to **null** as a result of the synchronization action.
is_current	No	-	Value provided from Source is ignored.
new_version	No	-	Value provided from Source is ignored. **new_version** is left unchanged on an Item.

System Property	Required *	Validation in syncUpdate	Comments
classification	No	If provided, should match class structure declared on an Item Type.	Saved as is.
current_state	No	If value is provided and it is not null, it should correspond to the ID of the lifecycle state from the Destination System. See more in LifeCycle state processing for a synced Item on Destination	If valid, saved as is.
state	No		Since **state** contains a name of a LCS referenced via the **current_state** property, the value will be overridden: If **current_state** is null (and this is valid), *state* property is also set to null Otherwise, it gets the name of the LC state corresponding to **current_state** on the Destination system.
is_released	No	-	Saved as is.
released_date **effective_date**	No	If provided for a released Versionable Item it should not null be and should be in a valid date format	Saved as is.
permission_id	No	-	Value provided from Source is ignored. If required, it is possible to add the **permission_id** property to the Item in the **onBeforeSyncUpdate** event handler on the Destination system.

System Property	Required *	Validation in syncUpdate	Comments
keyed_name	No	-	If the **keyed_name** is provided and is not null or empty, it is used as is. Otherwise it is calculated in the same way it is done for the **update** action. Assuming that the Source and Destination systems have the same metadata, the result should be the same.
modified_by_id	No	-	Saved as is if not null or empty. Otherwise, it is initialized using the same logic as the **update** action.
modified_on	No	If provided, should be valid date	
sort_order	No	-	If passed from Source, **sort_order** on Relationship Items is written on Destination as is.

* - Required properties should be provided with an Item, otherwise an error is returned.

3.4.3 dss_syncPurge

AML

```
<Item type="Part" id="7073098207234317BBC2CA865413CAD5" action="dss_syncPurge" />
```

Implementation Details

In Aras Innovator 11.0 SP15 this item action is implemented as a wrapper for the **purge** action.

In future releases, it is planned to introduce custom **onBeforeSyncDelete**/**onAfterSyncDelete** events and switch off calls for **onBeforeDelete**/**onAfterDelete** which are not marked as required.

3.5 Server Events

3.5.1 onBeforeSyncAdd

Overview

The **onBeforeSyncAdd** server event:

- Runs before an Item is added to the database during synchronization.
- Is called per Item.
- Can be used to populate some data with values that have not been passed from the Source System or to add local values, e.g., values for fields **created_by_id** and **managed_by_id**.
- Runs before required **onBeforeAdd** events (with `@is_required=1`).
- The **dss_syncAdd** request may be rejected completely by returning an error from this event handler.

Input/Output arguments

This event is called per Item. The request Item with the assigned ID is passed to the event handler as a Context Item (inDom). No additional event arguments are passed to the handler.

The context Item of **onBeforeSyncAdd** event handlers with all the changes applied to it inside the handlers is passed further as a Context Item to the **onBeforeAdd** event handlers.

The return results of **onBeforeSyncAdd** event handlers are ignored, unless there is an error. In this case the **dss_syncAdd** is stopped and the error is returned in the action response.

3.5.2 onAfterSyncAdd

Overview

The **onAfterSyncAdd** server event:

- Runs after an item has been added to the database during synchronization
- Runs after required **onAfterAdd** events (with `@is_required=1`).
- Is called per Item.
- Can be used to propagate the synchronized Items to external systems if required, e.g., for Mixed Federated Items.

Input/Output arguments

Unlike the **onAfterAdd** Server Event, **onAfterSyncAdd** Server Event handlers don't get the created Item as the Context Item. The event handler's Context Item is empty. The details about the created Item are provided in event arguments of type **OnAfterSyncAddEventArgs**:

```
public class OnAfterSyncAddEventArgs
{
    public string TypeId { get; }
    public string Id { get; }
}
```

If required, the event handlers can load the item from the DB themselves using the type and ID.

The return results of **onAfterSyncAdd** event handlers should be ignored, unless there is an error. In this case the **dss_syncAdd** should be stopped and the error should be returned in the action response.

3.5.3 onBeforeSyncUpdate

Overview

The **onBeforeSyncUpdate** event:

- Runs before an Item is updated in the database during synchronization.
- Runs before the required **onBeforeUpdate** event (`@is_required=1`).
- Can be used to populate some data with values that have not been passed from the Source System or should be adjusted to local values, e.g., values for fields **created_by_id** and **managed_by_id**.
- Is called per Item.
- The request may be rejected by returning an error.

Input/Output arguments

If the request Item contains a **where** condition or **idlist**, the condition is resolved to a list of Item IDs and the event handlers are called per each Item ID.

The request Item with the condition replaced with the Item ID is passed to the event handler as a Context Item (inDom). Table 5 shows an example.

Table 4: A request Item example

Request AML	OnBeforeUpdate Context Items
<Item type="SyncedItem" idlist="50DD6F18D7E14791965914314B1E156C, 5459D4256BAB4639AF564186A5174E93" action='dss_SyncUpdate'> <description>New Description</description> </Item>	<Item type='SyncedItem' id='50DD6F18D7E14791965914314B1E156C' action='dss_SyncUpdate'> <description>New Description</description> </Item>
	<Item type='SyncedItem' id='5459D4256BAB4639AF564186A5174E93' action='dss_SyncUpdate'> <description>New Description</description> </Item>

The context Item for **onBeforeSyncUpdate** event handlers with all the changes applied to it inside the handlers are passed further as a Context Item to the **onBeforeUpdate** event handlers.

In addition to the Context Items, the **onBeforeSyncUpdate** event handlers receive event arguments of type **OnBeforeSyncUpdateEventArgs** that contain a list of all the IDs in the entire as well as the ItemType ID:

```
public class OnBeforeSyncUpdateEventArgs
{
    public string TypeId { get; }
    public IEnumerable<string> IdList { get; }
}
```

The return results for **onBeforeSyncUpdate** event handlers are ignored, unless there is an error. In this case that **dss_syncUpdate** is stopped and the error is returned in the action response.

3.5.4 onAfterSyncUpdate

Overview

The **onAfterSyncUpdate** event:

- Runs after an item has been updated in the database during synchronization.
- Runs after a required **onAfterUpdate** event (`@is_required=1`).
- Runs once for the whole Item set in the request.
- Can be used to propagate the synchronized Items to external systems if required, e.g., for Mixed Federated Items.

Input/Output arguments

The event handler's Context Item is empty. The updated Items aren't loaded. The details about the updated Items are provided in the event arguments of type **OnAfterSyncUpdateEventArgs**:

```
public class OnAfterSyncUpdateEventArgs
{
    public string TypeId { get; }
    public IEnumerable<string> IdList { get; }
}
```

If required, the event handlers can load the items from the database themselves using the type and ID list.

The return results of **onAfterSyncUpdate** event handlers are ignored, unless there is an error. In the case of an error, the **dss_syncUpdate** is stopped, the request transaction is rolled back, and the error is returned in the action response.