

# ServiceNow CMDB Sync Only Configuration

- 1. [Using this guide](#)
- 2. [ServiceNow CMDB Sync Basic Deployment Step-By-Step Walkthrough](#)
- 3. [Configuring the ServiceNow CMDB User](#)
- 4. [ServiceNow configuration for the CMDB](#)
  - [4.1. Create the VMwareTVS data\\_source](#)
  - [4.2. Create the CI Class Identification Rules](#)
  - [4.3. Create the CI Class Dependent Relationship](#)
  - [4.4. Create the Reconciliation Rules](#)
- 5. [Appendices](#)
  - [5.1. Appendix A: What is the Config File](#)
  - [5.2. Appendix B: CMDB Sync Config File fields](#)
  - [5.3. Appendix C: CMDB Sync Config File Example](#)


## 1. Using this guide

This guide is intended to walk through portions of the ServiceNow management pack. Please refer to the [landing page](#) for an overview and terminology.


## 2. ServiceNow CMDB Sync Basic Deployment Step-By-Step Walkthrough

The list below will give you an idea of a basic deployment of the ServiceNow Management Pack in steps. This will be a typical engagement.


1. Provide the [User Requirements](#) to the customer

 This is the ServiceNow User. This will create an account with the minimum permissions necessary, which we will then use in the adapter instance configuration later. The role and acl's listed are all necessary to read and modify the CMDB


2. Provide the customer the documentation to setup the [Initial configuration of CMDB Sync](#). This can be copy and pasted with little modifications.

 This step is modifying the customers' CMDB in ServiceNow. Due to the ServiceNow CMDB being so customizable, we need to organize the structure in a way where we can reference it.


2a. Create the [VMwareTVS data\\_source](#)

 The Data Source will allow ServiceNow to listen to the adapter instance. This step is letting ServiceNow trust and communicate with the adapter to populate the CMDB with data.


2b. Create the [CI Class Identification rules](#)

 The CI Class Identification rules are being modified so only the items that belong in each category, are put in there. It is organizing the configuration items which means the objects we expect to update in the certain classes, will be there.

2c. Create the [CI Class Dependent relationship](#)

 Dependent Relationships are defining the structure and relationships between configuration items. In this step we are organizing the structure in a way that the adapter instance will expect.

2d. Create the [Reconciliation rules](#)

 The Reconciliation Rules are allowing the Data Source to update the configuration items we choose. Configuring these will allow the Data Source we have set up previously to work on the specific configuration items we will be modifying.

3. OPTIONAL: Verify with the customer over a call that all of the configuration has been completed.

**i** This is optional but recommended. There are several steps to set up the ServiceNow environment and sometimes going through it again finds problems that will hold you up later during testing and verification. Depending on how well the customer understand the requests, and how the changes are being assigned it may be useful or even necessary to review the modifications. If you wait until the end, it may cause them to lose trust in the product.

4. Edit one of the MOID '[combined](#)' templates that identifies the customers alert type (incidents, events or alerts). A typical config used is: 'moid\_incidents\_and\_populate\_only.json'. You will notice 'incidents' in the name to identify the alert type used, populate only means it will not remove CMDB items when removed from Aria Operations. Below is a [config file from our environment](#) as an example that you can use if necessary.

**i** There are several options for templates that you can use to make configuring the config file easy. It is important however to use a template that is either incident, alert or event based on the customer environment as those change slightly.

5. Save the config file you are editing into the 'work' directory on the Aria Operations collector, with a name that is appropriate. Make sure to place it on any collector/cloud proxy in the group the adapter instance is assigned to.

**i** This step will be done through SSH exclusively. It will need to exist in the 'work' directory on everything in the collector group as it does not propagate automatically. It is also recommend the customer has a back up.

6. In the ServiceNow adapter instance, use the filename for the config file you have created.

**i** All done with the set up. Now configure the adapter instance with the user, config file name and any other option that the customer will need located in the 'Advanced' settings of the Adapter Instance. At this point you will be ready to test. A appropriate test would be to check the ServiceNow Aria Operations dashboards. Finally you could check some of the configuration items (CI's) in ServiceNow to verify data is being brought in from the adapter.

## 3. Configuring the ServiceNow CMDB User

1. Create a user then assign it the "itil" role. You can use this same user for the alerts as well.

2. Create a new role and assign the new role to the user. Then add the ACL permissions (under System Security -> Access Control List) for the tables below to the new role, use read permissions:

**Important:** To make changes to the ACL, you have to elevate the permissions of your account. To do that, click on your username in the top right, and select 'Elevate Roles'

- cmdb\_metadata\_hosting
- cmdb\_metadata\_reference
- cmdb\_metadata\_containment
- sys\_db\_object
- cmdb\_reconciliation\_definition
- sys\_choice
- sys\_choice.\*
- cmdb\_rel\_type
- sys\_user\_has\_role
- security\_acl\_detail
- sys\_security\_operation
- sys\_user
- sys\_dictionary
- sys\_dictionary.\*
- sys\_glide\_object

Each table below should have "read" and "edit\_ci\_relations":

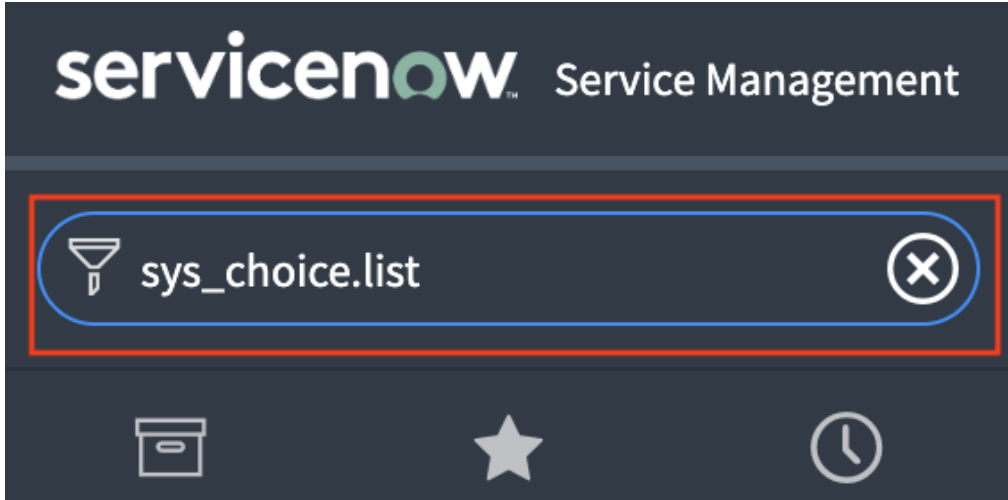
- cmdb\_ci\_vmware\_instance
- cmdb\_ci\_vcenter
- cmdb\_ci\_esx\_server
- cmdb\_ci\_vcenter\_datastore
- cmdb\_ci\_vcenter\_datacenter

- cmdb\_ci\_vcenter\_cluster

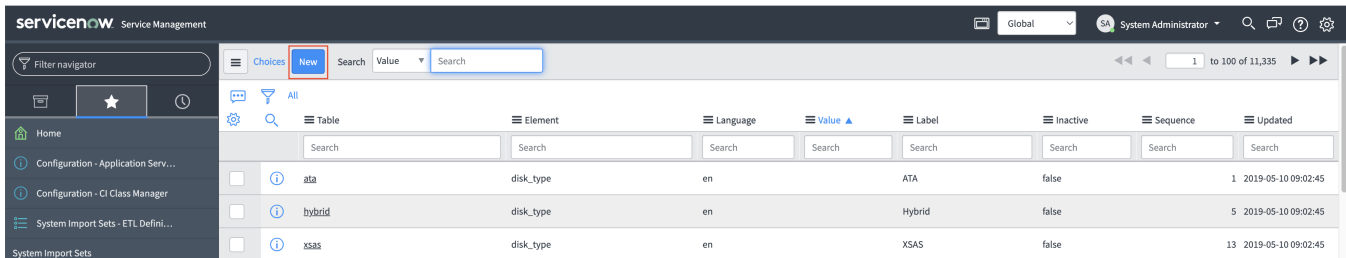
## 4. ServiceNow configuration for the CMDB

### 4.1. Create the VMwareTVS data\_source

1. Open the Choices table (enter sys\_choices.list in the Filter navigator)



2. Click on the New button



3. Fill in the following fields

Table: Configuration Item [cmdb\_ci]

Element: discovery\_source

Label: VMwareTVS

Value: VMwareTVS

Click Submit

Choice - New record

\* Table: Configuration Item [cmdb\_ci]      Sequence:

\* Element: discovery\_source      Inactive:

\* Language: en

\* Label: VMwareTVS

\* Value: VMwareTVS

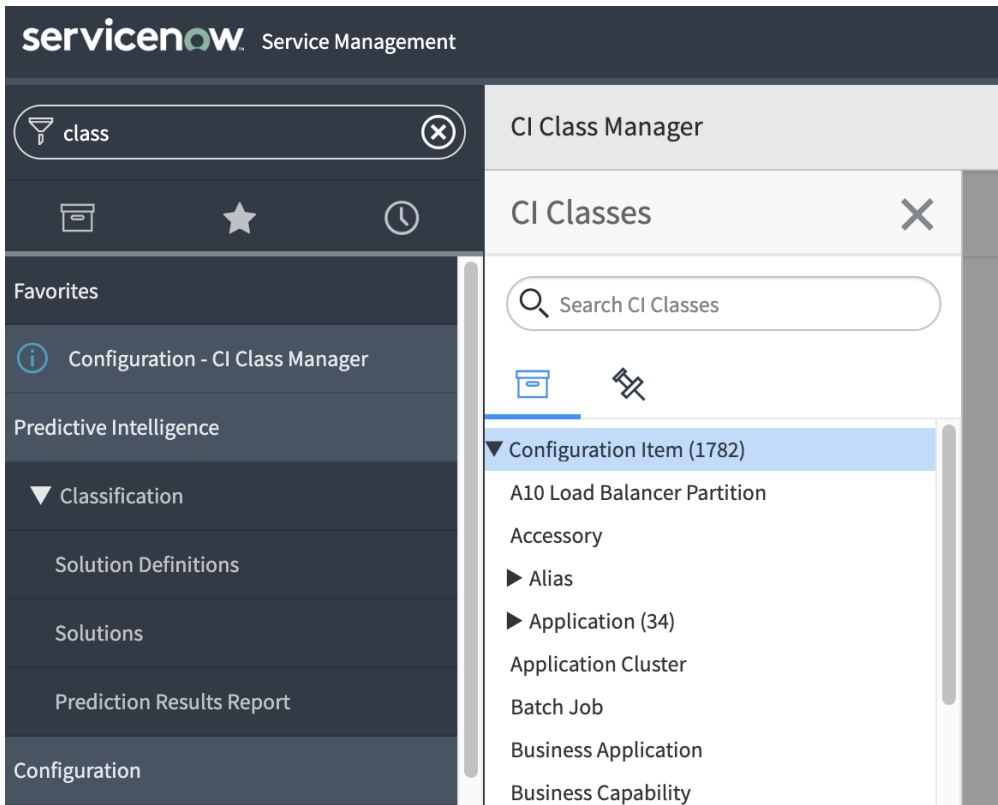
Dependent value:

Hint:

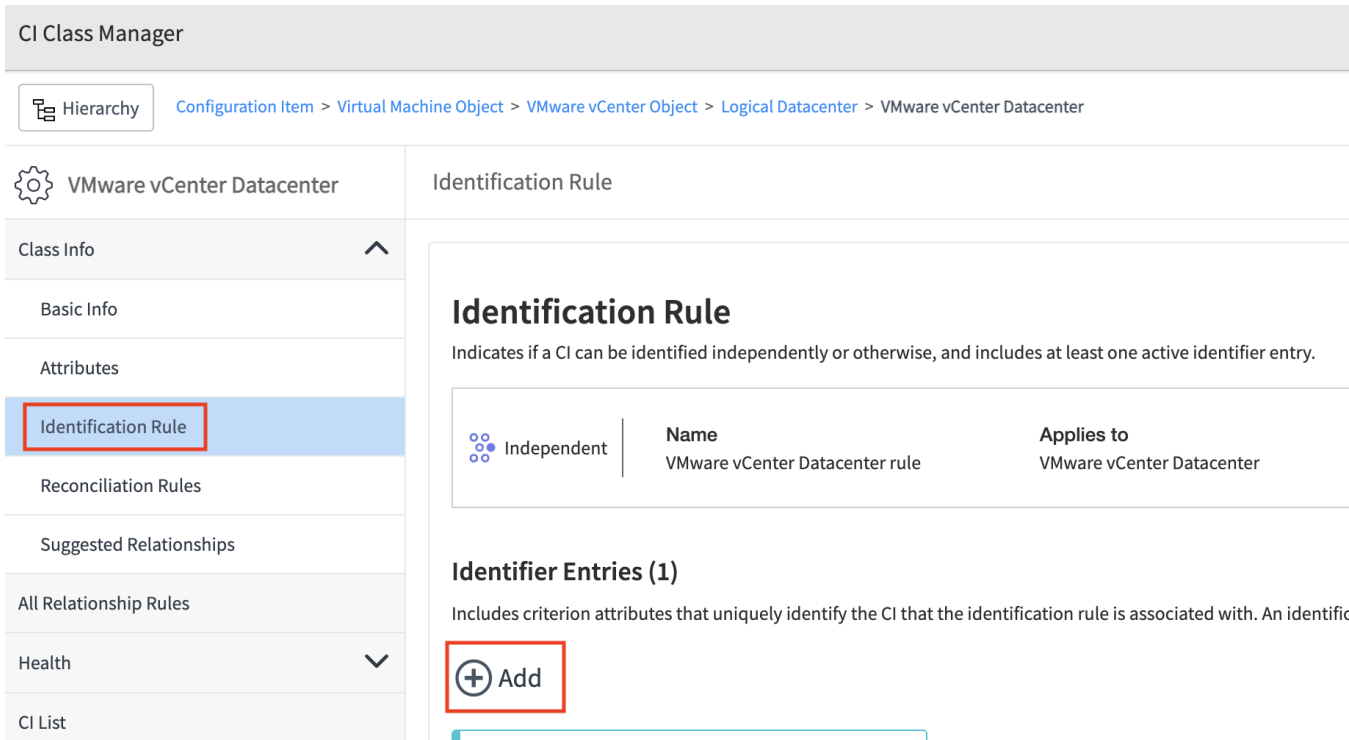
Submit

## 4.2. Create the CI Class Identification Rules

1. Open the CI Class Manager Hierarchy



2. Search for the appropriate CI (see list below) and click on it
3. Click on Identification Rule under Class Info, then click on Add under Identifier Entries



4. Select "Use attributes from main table ..." and click Next

### New Identifier Entry ✕

An identifier entry uses criterion attributes to uniquely identify this class. Which table would you like to choose attributes from?

Use attributes from main table 'VMware vCenter Instance'  
 Use attributes from another table (Lookup table)  
 Use attributes from main and another table (Hybrid)

5. Choose any value for Priority (100 recommended)

6. Select the appropriate attributes (see list below)

### Create Identifier Entry ✕

Active

\* Search On Table

\* Priority

\* Criterion Attributes

Available	Selected
Purchased	Object ID
Region	vCenter Instance UUID
Requires verification	
Schedule	
Serial number	
Server	
Skip sync	
Start date	
Status	
Subcategory	
Support group	
Supported by	
Top level folder for hosts	
Top level folder for VMs	
vCenter Reference	
Vendor	
Warranty expiration	

Allow null attributes

7. Click Save

#### CI Classes requiring an Identification Rule

- cmdb\_ci\_vcenter\_datacenter (VMware vCenter Datacenter)
  - o Add a rule that uses object\_id and vcenter\_uuid together

Search on Table  
**VMware vCenter Datacenter**  
Priority  
**100**  
Attributes (2)  
**Object ID, vCenter Instance UUID**  
  
Active

· cmdb\_ci\_esx\_server (ESX Server)

- o Add a rule that uses object\_id and vcenter\_uuid together

Search on Table ✕  
**ESX Server**  
Priority  
**50**  
Attributes (2)  
**Object ID, vCenter Instance UUID**  
  
Active

· cmdb\_ci\_vcenter (VMware vCenter Instance)

- o Add a rule that uses instance\_uuid

Search on Table  
**VMware vCenter Instance**  
Priority  
**100**  
Attributes (1)  
**Instance UUID**  
  
Active

### 4.3. Create the CI Class Dependent Relationship

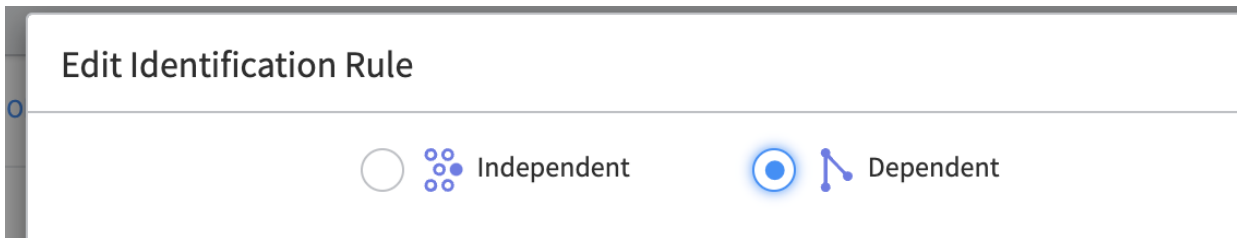
1. Open the CI Class Manager Hierarchy
2. Search for the cmdb\_ci\_vccenter\_datastore (VMware vCenter Datastore) CI and click on it
3. Click on Identification Rule under Class Info
4. Ensure the Identification Rule is Dependent

#### Identification Rule

Indicates if a CI can be identified independently or otherwise, and includes at least one active identifier entry.

Dependent	Name	Applies to	Description	Derived	Replace
<input checked="" type="checkbox"/>	VM Object	Virtual Machine Object		<input type="checkbox"/>	<input type="checkbox"/>

- a. If the Identification Rule is Independent, click on Edit and change the value to Dependent



5. Click on Dependent Relationships under Class Info

CI Class Manager

Hierarchy Configuration Item > Virtual Machine Object > VMware vCenter Object > Datastore >

VMware vCenter Datastore

Dependent Relationships

Class Info ^

Basic Info

Attributes

Identification Rule

**Dependent Relationships**

Reconciliation Rules

Suggested Relationships

All Relationship Rules

Dependent Relationships

Create dependent relationship rules (hosting ; rules) to help identify dependent CIs.

6. Click Add dependency
7. Fill in the following fields:

Rule Type: Containment

Relationship: Contained by (Contains::Contained by)

Target Class: VMware vCenter Datacenter

### Add Dependent Relationship Rule

Rule Type

Hosting  
Hosting rules can only be one level deep, and always involve resources, typically physical or virtual hardware.

Containment  
Containment rules are chained to each other in a group, with a CI type that is the top-level (root) parent of the group

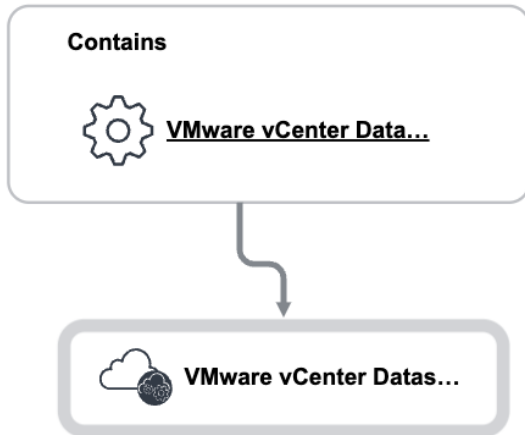
This Class: VMware vCenter Datastore  
Child class in the dependent relationship rule

\* Relationship: Contained by (Contains::Contained by) ▼

\* Target Class: VMware vCenter Datacenter ▼  
Parent class in the dependent relationship rule

Cancel Save

8. Click Save



9. Note: Viewing the relationship will show the relationship as "Contains", not "Contained by". That is expected.

#### 4.4. Create the Reconciliation Rules

1. Open the CI Class Manager Hierarchy
2. Search for the appropriate CI (see list below) and click on it
3. Click on Reconciliation Rules under Class Info, then click on Add under Reconciliation Rules



# CI Class Manager

Hierarchy

Configuration Item > Virtual Machine Object > VMware vCenter Object > Datastore > VMware vCenter Datastore

VMware vCenter Datastore

Reconciliation Rules

Class Info

Basic Info

Attributes

Identification Rule

Dependent Relationships

Reconciliation Rules

Suggested Relationships

All Relationship Rules

Health

CI List

## Reconciliation Rules

Determine which discovery sources are allowed to populate attributes and in which order.

Created (1)

+ Add

Attributes  
All  
Discovery Source & Priority  
VMwareTVS[100]  
Applies to  
VMware vCenter Datastore  
Active

Derived (1)

4. Fill in the following fields and click Next:

Discovery Source: VMwareTVS

Choose any value for Priority (100 recommended)

### Create Reconciliation Rule

Add Discovery Sources & Prioritize      Select Attributes      Set Filter Condition

Add discovery sources authorized for the class and assign priorities.

Active

\* Discovery Source: VMwareTVS      \* Priority: 100

Back      Next

5. Select "Apply to all attributes" and click Next

### Create Reconciliation Rule

Add Discovery Sources & Prioritize ✓
Select Attributes
Set Filter Condition

Select attributes that can be updated by the authorized discovery sources.

Apply to all attributes

Update with null

Available	Selected
Accessible	
Approval group	
Asset	
Asset tag	
Assigned	
Assigned to	
Attestation Score	
Attestation Status	
Attested	
Attested By	
Attested Date	
Attributes	
Business Unit	
Can Print	
Capacity (GB)	
Category	
Change Group	

6. Leave the next tab blank and click Save

### Edit Reconciliation Rule

Add Discovery Sources & Prioritize ✓
Select Attributes
Set Filter Condition

Authorized discovery sources can update only those CIs that meet the following conditions.

Filter Condition All of these conditions must be met

7. The new Reconciliation Rule will look like this

## Attributes

### All

#### Discovery Source & Priority

### VMwareTVS[100]

#### Applies to

### VMware vCenter Instance

Active

CI Class requiring a Reconciliation Rule

- cmdb\_ci\_vmware\_instance (VMware Virtual Machine Instance)
- cmdb\_ci\_vcenter (VMware vCenter Instance)
- cmdb\_ci\_esx\_server (ESX Server)
- cmdb\_ci\_vcenter\_datastore (VMware vCenter Datastore)
- cmdb\_ci\_vcenter\_datacenter (VMware vCenter Datacenter)
- cmdb\_ci\_vcenter\_cluster (VMware vCenter Cluster)

## 5. Appendices

### 5.1. Appendix A: What is the Config File

The ServiceNow adapter instance uses a config file to define all of the configuration that describes the customers ServiceNow environment. The config file does not exist by default. You can either copy a template over or create a new one. The config file can go on the collector that the adapter instance is running on, if using a group it is best to push the config file to the correct directory on all collectors in the group.

The config file will go here: `/usr/lib/vmware-vcops/user/plugins/inbound/servicenow_adapter3/work`

The templates are located at: `/usr/lib/vmware-vcops/user/plugins/inbound/servicenow_adapter3/conf/config_samples/`

When the config file is placed in the work directory, you can enter just the filename in the adapter instance configuration screen.

### 5.2. Appendix B: CMDB Sync Config File fields

The CMDB Sync feature will use the config file to map Aria Operations objects to their ServiceNow counterpart.

The config file can include the following, each of which will be described in greater detail.

```
"cmdbSync": {
  "cmdbSyncMethod": "IRE_API",
  "syncMode": "POPULATE_AND_DELETE_WHEN_NOT_EXISTING",
  "objectIdentifierSource": "MOID",
  "builtInTreeEnabled": true,
  "customTreesEnabled": false,
  "additionalColumns": {},
  "nameFilter": {}
}
```

If you have multiple adapter instances and need to reference different custom trees, the parameter `customTreeDirName` can be added to the `cmdbSync` configuration. This maps to a string which defines the name of the custom tree directory in `work`.

Additionally, using CMDB sync requires the user to add a Data Source to the identification rule for each resource type, otherwise columns cannot be updated via IRE.

There are a few CI Class Identification Rules that must be addressed as well.

- `cmdb_ci_vcenter_datacenter`:
  - Add a rule that uses `object_id` and `vcenter_uuid` together to uniquely identify the datacenter
- `cmdb_ci_vcenter_datastore`
  - Add a dependent containment relationship to the parent Datacenter (`cmdb_ci_vcenter_datastore` Contains::Contained By `cmdb_ci_vcenter_datacenter`)
- `cmdb_ci_esx_server`
  - Add a rule that uses `object_id` and `vcenter_uuid` together to uniquely identify the datacenter
  - If Discovery is being used, you may also need to disable the `correlation_id` rule to avoid duplicate objects.
- `cmdb_ci_vcenter`
  - Add a rule that uses `instance_uuid` as the unique identifier

#### 5.2.1.1.1. Data Source

In order to sync data, the following must be done on ServiceNow:

- Add `data_source VMwareTVS` to the `Choice List` (you can search `sys_choice.list` in the UI or query `sys_choice` in the REST API) with the `Element (element in REST)` attribute set to `discovery_source` and the `Value (value)` and `Label (label)` attributes set to `VMwareTVS`. The `Table (table)` must be set to `cmdb_ci`.
- Add a Reconciliation Rule for your new `data_source (VMwareTVS)` on each of the CI classes in Synced Resources

If this is not done, IRE will not be able to update the resource, but the API still returns a successful response so we are not able to alert/fail collection due to this.

#### 5.2.1.1.2. Synced Resources

The following resources from the VMWARE adapter will be synced to the CMDB if CMDB Sync is enabled:

Aria Ops Type	CI Class
Datastore	<code>cmdb_ci_vcenter_datastore</code>
Datacenter	<code>cmdb_ci_vcenter_datacenter</code>
HostSystem	<code>cmdb_ci_esx_server</code>
ClusterComputeResource	<code>cmdb_ci_vcenter_cluster</code>
VMwareAdapter Instance	<code>cmdb_ci_vcenter</code>
VirtualMachine	<code>cmdb_ci_vmware_instance</code>

#### 5.2.1.1.3. CMDB Sync Method

Currently, the only supported sync method is the IRE API (`IRE_API`). This is the default value and specifying it is not required. `IRE_API` uses ServiceNow's [Identification and Reconciliation Engine](#) to resolve Aria Ops resources to the appropriate ServiceNow resources. For this to work properly, the identification rules for the ci classes this adapter populates should not be altered from what ServiceNow includes by default. If rules are edited or removed, we cannot guarantee how the IRE will resolve resources that this adapter sends it.

Note: The `TableAPI` is also used for methods the IRE API cannot accomplish such as adding/deleting relationships that have changed which would affect identification and deleting resources if the sync mode is set to allow deletion.

#### 5.2.1.1.4. Sync Mode

This specifies how we should sync CIs to the CMDB.

There are 3 options for `syncMode`:

- `POPULATE_ONLY`: Populates the CMDB, but does not remove CIs. If a resource is Not Existing, it sets the operational status to '6' (retire).
- `POPULATE_AND_DELETE_WHEN_NOT_EXISTING`: Populates the CMDB and removes CIs that have been marked as `Not Existing` in Aria Ops
- `POPULATE_AND_DELETE_WHEN_REMOVED`: Populates the CMDB and removes CIs that have been removed from Aria Ops. If a resource is Not Existing, it sets the operational status to '6' (retire).

#### 5.2.1.1.5. Object Identifier Source

This is used to select which identifier should be used for objects which have both a UUID and a MOID. Currently, this only applies to Virtual Machine resources. If you are populating your CMDB for the first time, we recommend using MOID. If you are using this adapter to add data to a CMDB that has already been populated, check the format of the identifier in the `object_id` column for a `cmdb_ci_vmware_instance` object and choose the option below that matches it.

There are 2 options for Object Identifier Source:

- MOID (default): Also called MoRef ID, this is a required identifier in Aria Ops and looks like `vm-111`. It includes a prefix and a number which is generated by a counter.
- UUID: This is an optional identifier in Aria Ops and looks like a standard UUID. In most cases, this should be present, but there is a chance that it will not be. We have a few VMs in our test environment which are in bad states that don't have UUIDs, but they do have MOIDs.

For further info about identifiers in vCenter, see

- [Identification Overview Part 1](#)
- [Identification Overview Part 2](#)

#### 5.2.1.1.6. Name Filter

A map from a built-in ServiceNow CI Class Identifier to a regex filter. Every resource with a name that matches this regex will not be synced to the CMDB. This can be omitted if you intend to include all resources. The following built-in classes can have filters applied to them:

Identifier	Aria Ops Resource Type	ServiceNow CI Class
builtInCluster	ClusterComputeResource	cmdb_ci_vcenter_cluster
builtInDatacenter	Datacenter	cmdb_ci_vcenter_datacenter
builtInDatastore	Datastore	cmdb_ci_vcenter_datastore
builtInHostSystem	HostSystem	cmdb_ci_esx_server
builtInVCenter	VMwareAdapter Instance	cmdb_ci_vcenter
builtInVirtualMachine	VirtualMachine	cmdb_ci_vmware_instance

It will look like this if you wish to ignore all VMs that end in the string `dev`:

```
{
  "builtInVirtualMachine": "^.*dev$"
}
```

Note: VM templates are automatically filtered out for `builtInVirtualMachine` so there is no need to add a name filter for VM templates.

#### 5.2.1.1.7. Enabling Trees

By default, built in trees are enabled and custom trees are disabled. If you need to change the identification of the built-in classes, or map these resources to different CIs, you should disable the built-in tree and create your own custom tree to represent vCenter resources. If you just want to add some columns to the existing tree, use `additionalColumns` rather than defining a new tree. If you want to map another tree representing another technology that Aria Ops can monitor, such as AWS, but also use the built-in tree, enable custom trees and leave the built-in tree also enabled. Defining custom trees is described in [Custom Trees](#).

#### 5.2.1.1.8. Additional Columns

To add properties to the CMDB that are not defined by the built-in tree definition, use the 'additionalColumns' parameter.

This is a mapping from class identifier (See table in [Name Filter](#)) to a list of additional columns. An additional column has the following fields:

- `cmdbColumn`: The key of the column that the Aria Ops field will be send to on the CMDB CI Class
- `vropsType`: Where to get this data from in Aria Ops. See the type table [below](#) for a list of options
- `vropsField`: Represents the field to sync. See the type table [below](#) for a list of options

This example syncs the Aria Ops property 'cpu|cpuModel' to the CMDB column 'cpu\_name' on all Host System resources.

```
"additionalColumns": {
  "builtInHostSystem": [
    {
      "cmdbColumn": "cpu_name",
      "vropsType": "PROPERTY",
      "vropsField": "cpu|cpuModel"
    }
  ]
}
```

#### 5.2.1.1.9. Aria Ops Types and Fields

When defining a column, this describes available 'vropsType' enums and what should be provided as the 'vropsField' when that type is selected

Type	Applicable Field
------	------------------

PROPERTY	The key of the Aria Ops property to sync such as 'cpu cpuModel'
IDENTIFIER	The key of the Aria Ops Identifier to sync sync as 'VMEntityVCID'
METRIC	The key of the Aria Ops property to sync such as 'cpu demand stress'
NAME	This should be left empty. The name of the resource will be used.
UUID	This should be left empty. The UUID assigned by Aria Ops for the resource will be used. This is unique to each Aria Ops instances so if the same resource is monitored by multiple Aria Ops instances, this is not an ideal metric to use
STATIC_STRING	The value you want assigned to the column. For instance, you could set 'Synced From Aria Operations' to a description column and the column would be set to that value for all resources of the CI Class that we sync.

#### 5.2.1.1.10. Custom Trees

To use custom trees, you must enable custom trees in the [CMDB Sync Configuration](#). You must define each custom tree in its own json file under 'work /custom\_trees'. When you make the directory 'custom\_trees', make sure your Aria Ops user has permission to read from that directory.

Format:

```
{
  "treeName": "Sample Tree Name",
  "syncClasses": []
}
```

##### 5.2.1.1.10.1. Tree Name

This is a unique identifier for the tree. It cannot match the name of any other custom trees, or the built-in tree if the built-in tree is enabled.

##### 5.2.1.1.10.2. Sync Classes

Sync classes defined what should be synced to the CMDB. For a tree to be valid, the chosen sync classes must include all identifiers required by the CMDB. Test connection will inform you if there are any issues with any of the sync classes.

```
{
  "classIdentifier": "Sample Class ID",
  "vROps": {...},
  "serviceNow": {...},
  "identifierMapping": {...},
  "dataColumns": [],
  "relationships": [],
  "propertyFilter": {}
}
```

A custom tree example is available. See [Custom Tree Example](#) for more information.

**Class Identifier:** This is a unique identifier for the class. It cannot match the identifier of any other custom class, or the built-in class if the built-in tree is enabled.

**Aria Ops Definition:** This is the same object required for ResourceType Aria Ops Definitions

**ServiceNow Definition:** This is the same object required for ResourceType ServiceNow Definitions

**Identifier Definition:** The identifier mapping is our way of linking our resources to ServiceNow's CIs. This is a combination of identifiers which uniquely identify the resource. For example, Aria Ops resources should use the vCenter UUID in combination with the MOID (which is what is shown in the example in the link below).

These do not have to be identifiers on either the CMDB side or the Aria Ops side, though they often will be.

This is the same object required for ResourceType Identifier Mapping

**Data Columns:** Data columns represent the columns that we sync to the CMDB that are not used as part of the identification mapping. See [Aria Ops Types and Fields](#) for available options for the vropsType and vropsField parameters.

```
{
  "cmdbColumn": "capacity",
  "vropsType": "PROPERTY",
  "vropsField": "storage|capacity",
  "conversion": 1024
}
```

Conversion is optional and defaults to null if not specified. If a conversion is set, it will multiply the value returned by Aria Ops by the conversion factor. For example, if you want to convert Bytes to KB, set `conversion` to `.0001`. If you want to convert KB to Bytes, set `conversion` to `1000`.

**Relationships** Defines all relationships to children. The `childClassIdentifier` is the unique Class Identifier assigned to the child class. The child class must be a part of this tree. The `relationshipType` is not required if the CI Class Definition defines only one possible relationship type between the two CI Classes, but is required if there are multiple options.

```
"relationships": [
  {
    "childClassIdentifier": "CustomDatastore",
    "relationshipType": "Contains::Contained By"
  }
]
```

Property Filter: This is used to filter resources based on properties. An example use-case would be if you wanted to map VMWARE VirtualMachine resources to `cmdb_ci_vmware_instance` CIs only when the property `summary|config|isTemplate` is set to `false`. You could make a second Custom Class which maps to `cmdb_ci_vmware_template` when `summary|config|isTemplate` is set to `true` from the same Aria Ops ResourceType.

If this is not needed for your class, omit the block.

This sample filters to only resources that are not templates

```
"propertyFilter": {
  "conditions": [
    {
      "vropsField": "summary|config|isTemplate",
      "value": "false",
      "filterOperation": "EQUALS",
      "default": false
    }
  ],
  "joiner": "AND or OR"
}
```

Property Filter Value: Set this to the value to search for in the string returned from the `vropsField`. If your field is a boolean or an integer, set it to the string representation of that boolean or integer (such as `"15"` or `"true"`)

Property Filter Operation: This specifies how we determine whether the filter condition is met.

Available options:

Option	Behavior
EQUALS	The condition is true if the property value returned by Aria Ops exactly matches the provided value
DOES_NOT_EQUAL	The condition is true if the property value returned by Aria Ops does not match the provided value
CONTAINS	The condition is true if the value returned from Aria Ops contains the provided value
DOES_NOT_CONTAIN	The condition is true if the value returned from Aria Ops does not contain the provided value

Property Filter Default: This specifies how we should evaluate the condition if we cannot retrieve the property in Aria Ops. Since the most common reason this could occur is for Aria Ops to discover a new resource but not complete a collection and thus assign properties yet, we recommend defaulting to `false`. Using `false` as default will cause the filter to exclude all resources that do not have this property assigned from the sync.

Property Filter Joiner: This specifies whether we should include the resource if all of the conditions are met (AND) or if at least one of the conditions are met (OR)

Available options:

Option	Behavior
AND	All property filters must resolve to true in order to include the resource
OR	At least one of the property filters must resolve to true in order to include the resource

### 5.2.1.2. Custom Tree Example

Since custom trees are likely different for every user, we don't have samples that are useable out-of-the box. We do have an example custom tree with explanations of all fields in `'conf/custom_trees/example.json'`. To use this, copy it to `'work/custom_trees/treeFileName.json'` (any name can be used) and replace the `instructions` values with the values you require for your tree. Since you can make as many trees as you want, it is recommended for users to copy the tree and then modify it rather than modify it in `conf` and then copy it. This way, it can continue to serve as an example for other trees you wish to define.

## 5.3. Appendix C: CMDDB Sync Config File Example

Below is a sample config for our environment. This is default from the template.

```
{
  "cmdbSync": {
```

```
"syncMode": "POPULATE_ONLY",
"objectIdentifierSource": "MOID"
},
"resourceTypes": [
{
"vROps": {
"type": {
"resourceKind": "VirtualMachine",
"adapterKind": "VMWARE"
}
},
"serviceNow": {
"type": {
"cmdbCiClass": "cmdb_ci_vmware_instance"
}
},
"identifierMapping": [
{
"serviceNowId": {
"column": "object_id"
},
"vropsId": {
"type": "IDENTIFIER",
"field": "VMEntityObjectID"
}
},
{
"serviceNowId": {
"column": "vcenter_uuid"
},
"vropsId": {
"type": "IDENTIFIER",
"field": "VMEntityVCID"
}
}
]
},
{
"vROps": {
"type": {
```



```
"resourceKind": "ClusterComputeResource",
"adapterKind": "VMWARE"
}
},
"serviceNow": {
"type": {
"cmdbCiClass": "cmdb_ci_vcenter_cluster"
}
},
"identifierMapping": [
{
"serviceNowId": {
"column": "object_id"
},
"vropsId": {
"type": "IDENTIFIER",
"field": "VMEntityObjectID"
}
},
{
"serviceNowId": {
"column": "vcenter_uuid"
},
"vropsId": {
"type": "IDENTIFIER",
"field": "VMEntityVCID"
}
}
]
},
{
"vROps": {
"type": {
"resourceKind": "Datacenter",
"adapterKind": "VMWARE"
}
},
"serviceNow": {
"type": {
"cmdbCiClass": "cmdb_ci_vcenter_datacenter"
```

```
}
},
"identifierMapping": [
  {
    "serviceNowId": {
      "column": "object_id"
    },
    "vropsId": {
      "type": "IDENTIFIER",
      "field": "VMEntityObjectID"
    }
  },
  {
    "serviceNowId": {
      "column": "vcenter_uuid"
    },
    "vropsId": {
      "type": "IDENTIFIER",
      "field": "VMEntityVCID"
    }
  }
],
{
  "vROps": {
    "type": {
      "resourceKind": "Datastore",
      "adapterKind": "VMWARE"
    }
  },
  "serviceNow": {
    "type": {
      "cmdbCiClass": "cmdb_ci_vcenter_datastore"
    }
  },
  "identifierMapping": [
    {
      "serviceNowId": {
        "column": "object_id"
      },
```

```
"vropsId": {
  "type": "IDENTIFIER",
  "field": "VMEntityObjectID"
},
{
  "serviceNowId": {
    "column": "vcenter_uuid"
  },
  "vropsId": {
    "type": "IDENTIFIER",
    "field": "VMEntityVCID"
  }
},
],
{
  "vROps": {
    "type": {
      "resourceKind": "HostSystem",
      "adapterKind": "VMWARE"
    }
  },
  "serviceNow": {
    "type": {
      "cmdbCiClass": "cmdb_ci_esx_server"
    }
  },
  "identifierMapping": [
    {
      "serviceNowId": {
        "column": "object_id"
      },
      "vropsId": {
        "type": "IDENTIFIER",
        "field": "VMEntityObjectID"
      }
    }
  ],
  {
    "serviceNowId": {
```

```
"column": "vcenter_uuid"  
},  
"vropsId": {  
  "type": "IDENTIFIER",  
  "field": "VMEntityVCID"  
}  
}  
]  
}  
]  
}
```