Cisco Prime Service Catalog 12.1 Patch Product Enhancement Guide
First Published: February 08, 2019
Last Updated: February 26, 2022

Contents

Overview

The purpose of this document is to highlight all major 12.1 release enhancements that have been shipped through recent 12.1 patches.

The 12.1 patch zip file contains a readme document that highlights all information pertaining to installation instructions, configuration changes, defects reported by the customer and minor feature enhancements.

This is therefore a supplementary document that adds to the information already available in the 12.1 readme document

Introduced in 12.1_Patch_v18

Dynamic Pagination in Grid dictionary

Prime Service Catalog 12.1 patch v18 now introduces pagination through data retrieval which is used to populate the grid dictionary. User can set the number of records per page to be displayed at dictionary level in the "Total number of rows per page" option available.
The maximum number of records setting in the *newscale.properties* is now increased to 300.
Note: Pagination will be applicable when the data exceeds the number of rows per page.

Restrict end-user password reuse

Prime Service Catalog 12.1 patch v18 has enhanced to restrict the Password Reuse which can be achieved by enabling the flag "*Enforce Password History*" and setting "*Password History Length*" in Administration Settings.
Note:
1.Password History Length should be in between 3 and 5.
2.The Password History Length determines the number of previous passwords that must have defined before you can use the same once again.
3.Password reuse restriction will not be applicable for the User when "Enforce Password History" flag is disabled.

AMQP authorization

Prime Service Catalog 12.1 patch v18 has enhanced agent's outbound configuration for publish or subscribe case, the queue declared will have to bind to the exchange declared to receive the outbound message published to the exchange.

Jar Upgrades

Handlebars Upgrade to 4.7.7

modules/common/js/vendor/handlebarsjs/4.7.6/
website/CloudIntegrationsWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/ComplianceWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/ServiceCatalogMobileWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/ServiceCatalogWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/ServiceManagerWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/TenantManagementWebsite/common/js/vendor/handlebarsjs/4.7.6/
website/UserManagementWebsite/common/js/vendor/handlebarsjs/4.7.6/
swagger/lib/handlebars-4.7.6.js

Current User API

The Current User API has been enhanced to return the Hierarchical OU Tree for the given user .

- Method: GET
- URL:http://localhost:8080/RequestCenter/nsapi/directory/people/currentuser?responseType=json
- Response message :
  The response would include a field HierarchicalOUList as below,

```
"hierarchicalOUList": [
        {
           "id": 7,
```

```
            "name": "custom_0 Service Icons"
        },
        {
            "id": 22,
            "name": "team1",
            "childOus": [
                {
                    "id": 30,
                    "name": "team3"
                },
                {
                    "id": 32,
                    "name": "team"
                },
                {
                    "id": 42,
                    "name": "team666"
                }
            ]
        },
        {
            "id": 23,
            "name": "team1_admin"
        },
        {
            "id": 28,
            "name": "team2"
        }
    ]
```


Create, Read, Update and Delete of Service Definition-REST API's

Create Service :

- Method: POST
- URL: http://localhost:8080/RequestCenter/nsapi/definition/v2/service
- Payload:


```
{
    "service": {
        "name": "s1",
        "description": "create service s1",
        "serviceGroupID": 10
    }
}
```


Get Service:

- Method: GET
- URL: http://localhost:8080/RequestCenter/nsapi/definition/v2/service/{serviceName}
- Response:


```
{
    "service": {
        "id": 227,
        "name": "s2",
        "description": "create service s2",
        "serviceGroupID": 10,
        "isEntitlement": false,
        "isInactive": false,
        "orderingMode": 3,
        "computePrice": false,
        "paginationViewMode": 0,
        "showOrderSummary": false,
```

```
            "hideInServiceCatalog": false,
            "hideInMyOrders": false,
            "isTemplate": false,
            "maxQuantity": 0,
            "templateType": "",
            "tenantRelevant": true,
            "isOrderable": true,
            "isReportable": false
        }
    }
```


Update Service :

- Method: PUT
- URL: http://localhost:8080/RequestCenter/nsapi/definition/v2/service/{serviceId}
- Payload:


```
    {
        "service": {
            "id": 220,
            "name": "s2",
            "description": "create service s2",
            "descriptionURL": "",
            "serviceGroupID": 10,
            "isEntitlement": false,
            "isInactive": false,
            "forecastingMethod" :2,
            "orderingMode": 3,
            "computePrice": false,
            "paginationViewMode": 0,
            "showOrderSummary": false,
            "hideInServiceCatalog": false,
            "hideInMyOrders": false,
            "isTemplate": false,
            "templateType":"",
            "maxQuantity": 0,
            "tenantRelevant": true,
            "isOrderable": true,
            "isReportable": false,
            "serviceLevelDescription":"",
            "standardDuration": 0.0,
            "standardDurationUnits": "hours"
        }
    }
```


Delete Service :

- Method: DELETE
- URL: http://localhost:8080/RequestCenter/nsapi/definition/v2/service/{serviceId}

Introduced in 12.1_Patch_v17

Support for WildFly 21.0.2 Application Server

One of the major release enhancements shipped in 12.1_Patch_v17, is support for WildFly 21.0.2 application server. Patch v17 onwards, Prime Service Catalog release 12.1 is supported on three versions of WildFly:

- 21.0.2
- 18.0.1
- 10.1.0

The procedure to apply Prime Service Catalog patches remains the same across all supported WildFly versions.

Migration Procedure to WildFly 21.0.2

NOTE: Applicable only for clustered WildFly environments (both 2-VM and 4-VM topologies) running on a Linux distribution. After migrating to WildFly 21.0.2, the following server lifecycle scripts (located in <PSC_Install_Dir>/bin) will throw some errors, when executed. Prime Service Catalog operation and functionality will not be affected due to these errors.

- shutdownAllOnHC<n>, where n is any integer from 1-6
- deployServiceCatalogCluster
- undeployServiceCatalogCluster

Regardless of the WildFly server version (10.1.0 or 18.0.1) your Prime Service Catalog instance is currently running on, the migration procedure to WildFly 21.0.2 is the same.

Standalone WildFly Setup

NOTE: In the remainder of this section, the <Old_WildFly_Dir> placeholder, mentioned in directory paths, refers to:

- The directory, wildfly-10.1.0.Final, if your current WildFly version is 10.1.0
- The directory, wildfly-18.0.1.Final, if your current WildFly version is 18.0.1

1. Apply 12.1_Patch_v17 on your existing standalone WildFly environment. Instructions can be found in the following section of this Readme document.

   Section: *Patch Application Instructions* **>>** *Instructions for Standalone WildFly Setup*

2. Download the WildFly 21.0.2 distribution using the following URL. Copy it to a convenient location on your Prime Service Catalog host:

   https://download.jboss.org/wildfly/21.0.2.Final/wildfly-21.0.2.Final.zip

   The WildFly Downloads home page is located at: https://wildfly.org/downloads/

3. Download the JBoss Server Migration Tool, version 1.10.0.Final, using the URL below. Copy it to the same location as the downloaded WildFly 21 distribution:

   https://github.com/wildfly/wildfly-server-migration/releases/download/1.10.0.Final/jboss-server-migration-1.10.0.Final.zip

4. Stop the Prime Service Catalog servers, and remove the server temp files from the following locations:
   - <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceCatalogServer/tmp/
   - <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceLinkServer/tmp/

5. Take a backup of the <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.

6. Extract *wildfly-21.0.2.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
   - The <Old_WildFly_Dir> directory from your CPSC installation
   - The newly extracted directory: wildfly-21.0.2.Final

7. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.

8. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the ServiceCatalogServer managed server:

   i. Uncomment and update the following server path configurations:

   ```
   server.source.standalone.serverDir=ServiceCatalogServer
   server.source.standalone.configDir=configuration
   server.source.standalone.configFiles=standalone.xml,standalone-full.xml
   server.target.standalone.serverDir=ServiceCatalogServer
   server.target.standalone.configDir=configuration
   ```

   ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

9. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

   jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final

10. Answer the series of questions presented by the script, as follows:

    i. *Migrate the source's standalone server?* **Yes**

    ii. *Migrate all configurations?* **Yes**

    iii. *This tool is not able to assert if the non-persistent deployments found are compatible with the target server, skip scanner's deployments migration?* **No**

    iv. *Migrate the source's managed domain?* **No**

11. Edit the *environment.properties* file again, to set up the ServiceLinkServer managed server migration:

    i. Update the following server path configurations:

       server.source.standalone.serverDir=ServiceLinkServer
       server.source.standalone.configDir=configuration
       server.source.standalone.configFiles=standalone.xml,standalone-full.xml
       server.target.standalone.serverDir=ServiceLinkServer
       server.target.standalone.configDir=configuration

    ii. Update the modules.includes configuration as follows:

       modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. Repeat steps 9-10.

13. Copy all the *.cli* files, from <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceCatalogServer/configuration, to <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceCatalogServer/configuration. The files in question are:

    - *commonCLI_commands_RC.cli*
    - *oracleCLI_commands.cli*
    - *sqlCLI_commands.cli*

14. Copy all the *.cli* files from <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceLinkServer/configuration to <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceLinkServer/configuration. The relevant files are:

    - *commonCLI_commands_SL.cli*
    - *oracleCLI_commands.cli*
    - *sqlCLI_commands.cli*

15. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.

16. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceCatalogServer/configuration. Open the *standalone-full.xml* file for editing, and search for the AJP, HTTP, and HTTPS listener definitions as shown below:

    - <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
    - <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
    - <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

    Add the following attributes to ***all three*** listeners:

    - allow-unescaped-characters-in-url="true"

- max-parameters="5000"

17. Repeat step 16, for the *standalone-full.xml* file located in <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceLinkServer/configuration.
18. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.
19. If you are using Windows Services for starting and stopping the Prime Service Catalog servers:
    i. Navigate to the <PSC_Install_Dir>/conf directory.
    ii. Open the *servicecatalogwrapper.conf* and *servicelinkwrapper.conf* files, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.
    iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.
    iv. Re-install the "Cisco Prime Service Catalog" Windows Service. To do this, execute the following scripts in the specified order:
        a. uninstallServiceCatalogService
        b. installServiceCatalogService
    v. Re-install the "Cisco Prime Service Link" Windows Service To do this, execute the following scripts in the specified order:
        a. uninstallServiceLinkService
        b. installServiceLinkService
20. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

    Section: *Migration Procedure to WildFly 21.0.2* **>>** *Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 21.0.2*
21. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.
22. Start the Service Link server, followed by Service Catalog.
23. Use the following URLs to access the WildFly administration consoles, for the ServiceCatalogServer and ServiceLinkServer managed servers, and verify the WildFly version is correct.
    - Service Catalog console: <protocol>://<PSC_IP_or_Hostname>:9990/console
    - Service Link console: <protocol>://<PSC_IP_or_Hostname>:7990/console
24. Use the following URL to verify Prime Service Catalog is accessible. Log in, and navigate to the Service Link module, to confirm the connection status is active (green bubble).

    <protocol>://<PSC_IP_or_Hostname>:8080/RequestCenter
25. If you're using IIS as your web server, and the web server is installed on the same host where Prime Service Catalog is installed:
    i. Open IIS Manager, and select the website for Prime Service Catalog. Review the file system paths for the RequestCenter and IntegrationServer virtual directories.
    ii. If the physical paths begin with <PSC_Install_Dir>\<Old_WildFly_Dir>\..., update them to point to the respective folders inside the <PSC_Install_Dir>\wildfly-21.0.2.Final directory.
    iii. Restart the IIS website.
26. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Clustered WildFly Setup; 2-VM Topology

NOTE: In the remainder of this section, the <Old_WildFly_Dir> placeholder, mentioned in directory paths, refers to:

- The directory, wildfly-10.1.0.Final, if your current WildFly version is 10.1.0
- The directory, wildfly-18.0.1.Final, if your current WildFly version is 18.0.1

1. Apply 12.1_Patch v17 on your existing clustered WildFly environment

   Section: *Patch Application Instructions* **>>** *Instructions for Virtual Appliance and Clustered Wildfly Setup with 2-VM Topology*

2. Once the patch has been successfully applied, undeploy both *RequestCenter.war* and *ISEE.war.* To do this, log in to VM #1 (the Domain Controller node), navigate to <PSC_Install_Dir>/bin, and execute the following script:

undeployServiceCatalogCluster ALL

3. Stop the Prime Service Catalog servers completely, across ***all*** nodes.
4. Log in to VM #1 (the Domain Controller node).
5. Using the following URL, download the WildFly 21.0.2 distribution to a convenient location on this host:

https://download.jboss.org/wildfly/21.0.2.Final/wildfly-21.0.2.Final.zip

The WildFly downloads home page is located at: https://wildfly.org/downloads/

6. Download the JBoss Server Migration Tool, version 1.10.0.Final, to the same location as the WildFly 21 distribution. The tool can be obtained from

https://github.com/wildfly/wildfly-server-migration/releases/download/1.10.0.Final/jboss-server-migration-1.10.0.Final.zip

7. Remove the server temp files from the following locations:

- <PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host1-RC/tmp
- <PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host1-SL/tmp

8. Take a backup of the existing <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.
9. Extract *wildfly-21.0.2.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

- The <Old_WildFly_Dir> directory from your CPSC installation
- The newly extracted directory: wildfly-21.0.2.Final

10. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.
11. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

i. Uncomment and update the following server path configurations:

server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain.xml
server.source.domain.hostConfigFiles=host.xml,hostva_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final

13. Answer the series of questions presented by the script, as follows:

iii. *Migrate the source's standalone server?* ***No***

iv. *Migrate the source's managed domain?* ***Yes***

v. *Migrate all configurations?* ***Yes***

vi. *This tool is not able to assert if persistent deployments found are compatible with the target server, skip persistent deployments migration?* ***No***

vii. *Migrate all persistent deployments found?* ***Yes***

viii. *Migrate all configurations?* ***Yes***

14. Copy all the *.cli* files from <PSC_Install_Dir>/<Old_WildFly_Dir>/domain/configuration to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration. The files in question are:

- *cluster_commonCLI_commands.cli*
- *cluster_ha_profileCLI_commands.cli*
- *cluster_nonha_profileCLI_commands.cli*
- *cluster_oracleCLI_commands_ha.cli*
- *cluster_oracleCLI_commands_nonha.cli*
- *cluster_sqlCLI_commands_ha.cli*
- *cluster_sqlCLI_commands_nonha.cli*

15. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.

16. This step is required only if your Prime Service Catalog hosts are running on Microsoft Windows.

Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/cluster-bin/windows. Copy the contents of this directory over to <PSC_Install_Dir>/bin, overwriting the existing scripts.

NOTE: This step needs to be performed only on VM #1 (the Domain Controller node).

17. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration. Open the *domain.xml* file for editing, and locate ***all instances*** of the following listener definitions:

ix. <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>

x. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

xi. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

18. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

19. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallProcessControllerClusterService

b. installProcessControllerClusterService

20. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

Section: *Migration Procedure to WildFly 21.0.2* **>>** *Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 21.0.2*

21. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.

22. Start the Prime Service Catalog cluster on VM #1.

23. Deploy both *RequestCenter.war* and *ISEE.war*. To do this, navigate to <PSC_Install_Dir>/bin, and execute the following script:

deployServiceCatalogCluster ALL

24. Using the following URL, access the administration console for the WildFly domain, and verify the WildFly version is correct:

<protocol>://<VM1_IP_or_Hostname>:9990/console

25. Use the following URL to verify Prime Service Catalog is accessible. Log in, and navigate to the Service Link module, to confirm the connection status is active (green bubble).

<protocol>://<VM1_IP_or_Hostname>:8080/RequestCenter

26. Log in to VM #2 (Host Controller #2).

27. Download the WildFly 21.0.2 distribution, and the JBoss Server Migration Tool version 1.10.0.Final, to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.

28. Remove the server temp files from the following location:

<PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host2-RC/tmp

29. Take a backup of the existing <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.

30. Extract the *wildfly-21.0.2.Final.zip* file inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

- The <Old_WildFly_Dir> directory from your CPSC installation
- The newly extracted directory: wildfly-21.0.2.Final

31. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.

32. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

xii. Uncomment and update the following server path configurations:

```
server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain_backup.xml
server.source.domain.hostConfigFiles=host.xml,host2_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration
```

xiii. Update the modules.includes configuration as follows:

```
modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl
```

33. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

```
jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final
```

34. Answer the series of questions presented by the script, as follows:

xiv. *Migrate the source's standalone server?* ***No***

xv. *Migrate the source's managed domain?* ***Yes***

xvi. *Migrate all configurations?* ***Yes***

xvii. *Migrate all configurations?* ***Yes***

35. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.

36. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration.

37. Delete the *domain.xml* file.

38. Open the *domain_backup.xml* file for editing. Locate ***all instances*** of the following listener definitions:

xviii. <ajp-listener name="ajp" socket-binding="ajp"/>

xix. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

xx. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

39. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

40. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallProcessControllerClusterService

b. installProcessControllerClusterService

41. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.

42. Start the Prime Service Catalog cluster on VM #2.

43. Use the following URL to verify that cluster node #2 for Prime Service Catalog is available.

<protocol>://<VM2_IP_or_Hostname>:8080/RequestCenter

44. At this point, the migration process is complete, for a stock 2-VM topology cluster. If you have additional Host Controller nodes configured, repeat steps 26-43 on each of the additional nodes, while noting the following differences:

xxi. For Host Controller <N>, the server temp files are located at:

<PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host<N>-RC/tmp

xxii. In *environment.properties*, the server.source.domain.hostConfigFiles property value should be

server.source.domain.hostConfigFiles=host.xml,host<N>_backup.xml

xxiii. When performing step 43, use the IP address of the VM corresponding to the Host Controller you just configured.

45. If you’re using IIS as your web server, and the web server is installed on any of the cluster nodes:

i. Log in to the cluster node where IIS is installed.

ii. Open IIS Manager, and select the website for Prime Service Catalog. Review the physical paths for the RequestCenter and IntegrationServer virtual directories.

iii. If the physical paths begin with <PSC_Install_Dir>\<Old_WildFly_Dir>\content\..., update them to point to the respective folders inside the <PSC_Install_Dir>\wildfly-21.0.2.Final\content directory.

iv. Restart the IIS website.

46. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Clustered WildFly Setup; 4-VM Topology

NOTE: In the remainder of this section, the <Old_WildFly_Dir> placeholder, mentioned in directory paths, refers to:

- The directory, wildfly-10.1.0.Final, if your current WildFly version is 10.1.0
- The directory, wildfly-18.0.1.Final, if your current WildFly version is 18.0.1

1. Apply 12.1_Patch_v17 on your ***existing*** clustered WildFly environment. Instructions can be found in the following section of this Readme document.

   Section: *Patch Application Instructions* **>>** *Instructions for Clustered Wildfly Setup with 4-VM Topology*

2. Once the patch has been successfully applied, undeploy *RequestCenter.war.* To do this, log in to VM #1 (the Domain Controller node), navigate to <PSC_Install_Dir>/bin, and execute the following script:

   undeployServiceCatalogCluster RC

3. Stop the Prime Service Catalog servers completely, across ***all*** nodes.

4. Log in to VM #2 (the Service Link node).

5. Using the following URL, download the WildFly 21.0.2 distribution to a convenient location on this host:

   https://download.jboss.org/wildfly/21.0.2.Final/wildfly-21.0.2.Final.zip

   The WildFly downloads home page is located at: https://wildfly.org/downloads/

6. Download the JBoss Server Migration Tool, version 1.10.0.Final, to the same location as the WildFly 21 distribution. The tool can be obtained from

   https://github.com/wildfly/wildfly-server-migration/releases/download/1.10.0.Final/jboss-server-migration-1.10.0.Final.zip

7. Remove the server temp files from the following location:

   <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceLinkServer/tmp/

8. Take a backup of the existing <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.

9. Extract *wildfly-21.0.2.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

   - The <Old_WildFly_Dir> directory from your CPSC installation
   - The newly extracted directory: wildfly-21.0.2.Final

10. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.

11. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the ServiceLinkServer managed server:

    i. Update the following server path configurations:

       server.source.standalone.serverDir=ServiceLinkServer
       server.source.standalone.configDir=configuration
       server.source.standalone.configFiles=standalone.xml,standalone-full.xml
       server.target.standalone.serverDir=ServiceLinkServer
       server.target.standalone.configDir=configuration

    ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final

13. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* **Yes**

ii. *Migrate all configurations?* **Yes**

iii. *This tool is not able to assert if the non-persistent deployments found are compatible with the target server, skip scanner's deployments migration?* **No**

iv. *Migrate the source's managed domain?* **No**

14. Copy all the *.cli* files from <PSC_Install_Dir>/<Old_WildFly_Dir>/ServiceLinkServer/configuration to <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceLinkServer/configuration. The relevant files are:

- *commonCLI_commands_SL.cli*
- *oracleCLI_commands.cli*
- *sqlCLI_commands.cli*

15. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.

16. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceLinkServer/configuration. Open the *standalone-full.xml* file for editing, and search for the AJP, HTTP, and HTTPS listener definitions as shown below:

- <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
- <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
- <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***all three*** listeners:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

17. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

18. If you are using Windows Services for starting and stopping the Prime Service Catalog servers:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *servicelinkwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Link” Windows Service To do this, execute the following scripts in the specified order:

a. uninstallServiceLinkService

b. installServiceLinkService

19. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.

20. Start the Service Link server.

21. Log in to VM #1 (the Domain Controller node).

22. Download the WildFly 21.0.2 distribution, and the JBoss Server Migration Tool version 1.10.0, to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.
23. Take a backup of the existing <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.
24. Extract *wildfly-21.0.2.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
    - The <Old_WildFly_Dir> directory from your CPSC installation
    - The newly extracted directory: wildfly-21.0.2.Final
25. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.
26. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:
    i. Uncomment and update the following server path configurations:

       server.source.domain.domainDir=domain
       server.source.domain.configDir=configuration
       server.source.domain.domainConfigFiles=domain.xml
       server.source.domain.hostConfigFiles=host.xml,host_default.xml
       server.target.domain.domainDir=domain
       server.target.domain.configDir=configuration

    ii. Update the modules.includes configuration as follows:

       modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl
27. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

    jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final
28. Answer the series of questions presented by the script, as follows:
    i. *Migrate the source's standalone server?* ***No***
    ii. *Migrate the source's managed domain?* ***Yes***
    iii. *Migrate all configurations?* ***Yes***
    iv. *This tool is not able to assert if persistent deployments found are compatible with the target server, skip persistent deployments migration?* ***No***
    v. *Migrate all configurations?* ***Yes***
29. Copy all the *.cli* files from <PSC_Install_Dir>/<Old_WildFly_Dir>/domain/configuration to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration. The files in question are:
    - *cluster_commonCLI_commands.cli*
    - *cluster_ha_profileCLI_commands.cli*
    - *cluster_nonha_profileCLI_commands.cli*
    - *cluster_oracleCLI_commands_ha.cli*
    - *cluster_oracleCLI_commands_nonha.cli*
    - *cluster_sqlCLI_commands_ha.cli*
    - *cluster_sqlCLI_commands_nonha.cli*
30. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.
31. This step is required only if your Prime Service Catalog hosts are running on Microsoft Windows.

Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/cluster-bin/windows. Copy the contents of this directory over to <PSC_Install_Dir>/bin, overwriting the existing scripts.

NOTE: This step needs to be performed only on VM #1 (the Domain Controller node).

32. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration. Open the *domain.xml* file for editing, and locate ***all instances*** of the following listener definitions:

- <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
- <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
- <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

33. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

34. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallProcessControllerClusterService

b. installProcessControllerClusterService

35. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

Section: *Migration Procedure to WildFly 21.0.2* **>>** *Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 21.0.2*

36. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.

37. Start the Prime Service Catalog cluster on VM #1.

38. Re-deploy *RequestCenter.war*. To do this, navigate to <PSC_Install_Dir>/bin, and execute the following script:

deployServiceCatalogCluster RC

39. Log in to VM #3 (Host Controller #1).

40. Download the WildFly 21.0.2 distribution, and the JBoss Server Migration Tool version 1.10.0, to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.

41. Remove the server temp files from the following location:

<PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host1-RC/tmp

42. Take a backup of the existing <Old_WildFly_Dir> directory located in <PSC_Install_Dir>.

43. Extract the *wildfly-21.0.2.Final.zip* file inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

- The <Old_WildFly_Dir> directory from your CPSC installation
- The newly extracted directory: wildfly-21.0.2.Final

44. Extract the file, *jboss-server-migration-1.10.0.Final.zip*. This will create a directory called jboss-server-migration.

45. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

i. Uncomment and update the following server path configurations:

server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain_backup.xml
server.source.domain.hostConfigFiles=host.xml,host1_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

46. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/<Old_WildFly_Dir> --target <PSC_Install_Dir>/wildfly-21.0.2.Final

47. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* ***No***

ii. *Migrate the source's managed domain?* ***Yes***

iii. *Migrate all configurations?* ***Yes***

iv. *Migrate all configurations?* ***Yes***

48. Navigate to the directory where 12.1_Patch_v17 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-21.0.2.Final/modules.

49. Navigate to <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration.

50. Delete the *domain.xml* file.

51. Open the *domain_backup.xml* file for editing. Locate ***all instances*** of the following listener definitions:

i. <ajp-listener name="ajp" socket-binding="ajp"/>

ii. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

iii. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

52. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

53. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 21.0.2 directory: <PSC_Install_Dir>/wildfly-21.0.2.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallProcessControllerClusterService

b. installProcessControllerClusterService

54. Delete the <Old_WildFly_Dir> directory inside <PSC_Install_Dir>.

55. Start the Prime Service Catalog cluster on this node.

56. Repeat steps 39-55 on VM #4 (Host Controller #2), noting the following differences:

i. Remove the server temp files from the following location:

<PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host2-RC/tmp

ii. Configure the server path configurations in *environment.properties*, as follows:

server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain_backup.xml
server.source.domain.hostConfigFiles=host.xml,host2_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

57. Using the following URL, access the administration console for the WildFly domain, and verify the WildFly version is correct:

<protocol>://<VM1_IP_or_Hostname>:9990/console

58. Verify that Prime Service Catalog is accessible using ***both*** the following URLs. Log in, and navigate to the Service Link module, to confirm the connection status is active (green bubble).

- <protocol>://<VM3_IP_or_Hostname>:8080/RequestCenter
- <protocol>://<VM4_IP_or_Hostname>:8080/RequestCenter

59. At this point, the migration process is complete, for a stock 4-VM topology cluster. If you have additional Host Controller nodes configured, repeat steps 39-55, and step 58, on each of the additional nodes, while noting the following differences:

i. For Host Controller <N>, the server temp files are located at:

<PSC_Install_Dir>/<Old_WildFly_Dir>/domain/servers/server-host<N>-RC/tmp

ii. In *environment.properties*, the server.source.domain.hostConfigFiles property value should be

server.source.domain.hostConfigFiles=host.xml,host<N>_backup.xml

iii. When performing step 58, use the IP address of the VM corresponding to the Host Controller you just configured.

60. If you're using IIS as your web server, and the web server is installed on either VM #3, VM #4, or any of the additional Host Controller cluster nodes:

i. Log in to the cluster node where IIS is installed.

ii. Open IIS Manager, and select the website for Prime Service Catalog. Review the file system path for the RequestCenter virtual directory.

iii. If the physical path is set to <PSC_Install_Dir>\<Old_WildFly_Dir>\content\RequestCenter.war, update it to point to: <PSC_Install_Dir>\wildfly-21.0.2.Final\content\RequestCenter.war.

iv. Restart the IIS website.

61. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 21.0.2

If your Prime Service Catalog instance is hosted without the Service Link component, the following additional configuration changes are needed as part of the migration procedure to WildFly 21.0.2.

NOTE: General feature documentation, on how to configure Prime Service Catalog to operate without the Service Link component, can be found in the ***Cisco Prime Service Catalog 12.1 Patch Product Enhancement Guide*** published on Cisco.com.

Standalone WildFly Setup

1. On your Prime Service Catalog host, navigate to the following directory:

   <PSC_Install_Dir>/wildfly-21.0.2.Final/ServiceCatalogServer/deployments/RequestCenter.war/WEB-INF/classes/config

2. Open the beeeCamelContext.xml file, for editing. Locate the following bean definitions:

   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_1">
     <property name="from" value="activemq:queue:jms.queue.BEEEAuthorizationsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_2">
     <property name="from" value="activemq:queue:jms.queue.BEEERequisitionsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_3">
     <property name="from" value="activemq:queue:jms.queue.BEEEInboundQueue{Queue_Suffix}"/>

   NOTE: {Queue_Suffix} placeholder is the value passed for the optional queue name suffix parameter, during the execution of the jmsrunner utility. The placeholder may be empty.

   For ***all*** three definitions, remove the “jms.queue” substring from the value of the value attribute. Save the changes.

Clustered WildFly Setup; 2-VM and 4-VM Topologies

NOTE: In case of a clustered setup, it is recommended that you apply the following configuration changes once the migration process to WildFly 21.0.2 is complete.

1. On the Domain Controller node (VM #1 in both topologies) of your Prime Service Catalog installation, navigate to the following location inside the extracted *RequestCenter.war*:

   <PSC_Install_Dir>/tmp/RequestCenter.war/WEB-INF/classes/config

2. Open the beeeCamelContext.xml file, for editing. Locate the following bean definitions:

   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_1">
     <property name="from" value="activemq:queue:jms.queue.BEEEAuthorizationsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_2">
     <property name="from" value="activemq:queue:jms.queue.BEEERequisitionsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_3">
     <property name="from" value="activemq:queue:jms.queue.BEEEInboundQueue{Queue_Suffix}"/>

   NOTE: {Queue_Suffix} placeholder is the value passed for the optional queue name suffix parameter, during the execution of the jmsrunner utility. The placeholder may be empty.

   For ***all*** three definitions, remove the “jms.queue” substring from the value of the value attribute. Save the changes.

   In order for the above changes to take effect, the post-customizationOnDC script needs to be executed—a new *RequestCenter.war* file will be created and deployed into the Prime Service Catalog cluster.

3. Navigate to the <PSC_Install_Dir>/wildfly-21.0.2.Final/domain/configuration directory.

4. Edit *domain.xml*. Locate ***all*** instances of the snippet

   ```
   <subsystem xmlns="urn:jboss:domain:messaging-activemq:8.0">
     <server name="default">
   ```

   and add the following child element

   ```
   <remote-connector name="remote-artemis" socket-binding="remote-artemis"/>
   ```

   for the <server> tag. Save the changes.

Git-Based Collaborative Version Control Support for Prime Service Catalog Content Deployment

Prime Service Catalog can now be integrated with a Git-based source code repository hosting service, such as GitHub, Bitbucket, etc. The goal of this feature is to provide collaborative version control capabilities for content exported and imported through the Catalog Deployer module.
The following operations are supported in 12.1_Patch_v17:

- Push an assembled deployment package to a remote Git repository. This operation is equivalent to exporting the package, except the generated XML file is saved to the configured remote Git repository instead of the user's local file system.
- Pull a deployment package from a remote Git repository to your CPSC environment. This operation is equivalent to importing a package. Once pulled, the package can be deployed.

Defining the Integration

NOTE: As part of creating the integration, Prime Service Catalog clones the remote Git repository to a local directory on the application server host.
To define an integration with a Git repository hosting service:

1. On the application server host, designate a directory to be used by Prime Service Catalog for the integration.

   If Prime Service Catalog is running in a clustered setup, the directory needs to be created on all the cluster nodes. The fully qualified path for the directory must be identical across all nodes.

   NOTE: On Linux systems, the directory must have read and write permissions for all users.
2. Login to Prime Service Catalog as a user with *site administrator* privileges.
3. Navigate to the Integrations module.
4. From the home page, Click **New Integration**.
5. Select the entry: Git Repository Hosting Service.
6. On the Configure Integration page, fill out the connection details as follows:
   a. Identifier: Specify a 3-character, short name for the connection.
   b. Name: Specify a name for the connection.
   c. Protocol: At present, Prime Service Catalog supports cloning/connecting to a remote Git repository over the HTTPS protocol only.
   d. Remote Repository URL: The URL for the remote Git repository to connect to. This information can be obtained from the repository hosting service.
   e. Local Repository Directory: Specify the complete path for the directory created in step 1. Note that this is the parent directory, under which the remote repository will be cloned to its own sub-directory. The sub-directory is automatically created, and has the same name as the repository.
   f. Branch: Specify the name of the remote repository branch that corresponds to this Prime Service Catalog environment. E.g.: development, test, production.
   g. Repository Owner Access Token: This can be generated from the repository hosting service. The access token must belong to a user account that is either the repository owner or has administrative privileges for the repo.
7. Create Integration.

   Note: Once the connection is saved successfully, the user is redirected to the Manage Integration page.
8. Verify that the connection status shows "Active" (highlighted in green).

Updating the Integration

To modify a previously defined integration to a Git repository hosting service:

1. Navigate to the Integrations module as a site administrator user.
2. Click the gear icon next to the existing integration and select **Manage Integration**.
3. On the resulting page, Click the **EDIT** button.

4. Click **SAVE**.

Pushing Prime Service Catalog Content to Remote Git Repository

To save your service or entity definitions to a remote Git repository:

1. Navigate to the Catalog Deployer module.
2. Create a new deployment package.
3. Add the required service or entity definitions to the package and assemble it.
4. Once the package is assembled, the **Push to Git Repo** button appears above the package name.
5. Click and a popup opens. Fill out the fields as follows:
    a. Username: Specify the username of your account on the repository hosting service. Changes will be committed to the remote Git repository against this user account.
    b. Email: The email address corresponding to the user account above.
    c. Access Token: A personal access token for the user account above. This can be generated from the repository hosting service.
    d. Push Changes to: If you want to push your changes to an existing branch within the repository, select *Existing Branch* and choose a branch name from the Available Branches drop-down.

       Note: If you want your changes to be pushed to a new branch instead, select *New Branch* and specify a name in the New Branch Name field.
    e. Commit Message: Specify a meaningful description for the changes being committed.
6. Click **Ok**.
7. Once the changes have been successfully pushed, log in to your Git repository hosting service and review the commit.

RBAC for Pushing Content to Remote Git Repository

There is no separate RBAC in place for pushing content to remote Git repositories. If the user has the capability to manage deployment packages of a particular type, they will be allowed to push assembled packages to a remote Git repository. The existing system capabilities (pertaining to role definitions) that allow management of deployment packages are:

- Module: Catalog Deployer
- System Capabilities:
    - Manage Basic Service Deployments
    - Manage Advanced Service Deployments
    - Manage Custom Deployments

Pulling Prime Service Catalog from Remote Git Repository

To fetch service and/or entity definitions from a remote Git repository and deploy them to your Prime Service Catalog environment:

1. Navigate to the Catalog Deployer module.
2. Click on the **Action** drop-down and select **Pull from Git Repo**.
3. On the Commit History popup:
    a. Choose the branch from where you want to pull a deployment package.
    b. Type in a search string in the Search Commits field. A few points on the search behavior:
        - The Search Commits field implements incremental search. This means results will be displayed to the user as they keep typing.
        - Searches can be performed by file name, author, commit id, or commit message text.
        - To view the complete list of files available under the selected branch, use the asterisk (*) wildcard character as the search string.
    c. Select the package you wish to import, from the search results table.
4. Deploy the package once it is pulled successfully.

RBAC for Pulling Content from Remote Git Repository

Access to the **Pull from Git Repo** button is governed by an existing system capability (under role definition):

- Module: Catalog Deployer
- System Capability: Import Deployments

Package Upgrade for Virtual Appliances

Package name - sudo package

Version - 1.9.5.p2

To upgrade the package there are two methods:

- Downloading and installing the binary (.rpm) packages
- Downloading and building the source distribution

Upgrade Procedure Using Binary (.rpm) Packages

1. Login as root user to the primary app server node.
2. Run the following command to verify the CentOS version (expected value should be 7.X).

   cat /etc/centos-release.

3. Note down the **sudo 1.9.5p2** RPM package download URLs, for your OS version, from https://www.sudo.ws/download.html#binary.
4. Download the packages to the app server node, using the following command:
   wget {package_url}
5. Once the download is complete, install the packages using the command:
   yum localinstall {package_name}
6. Post installation, verify the sudo version using the command:
   sudo -V
7. Repeat above steps on all nodes of the Virtual Appliance.

Upgrade Procedure Using Source Distribution

1. Login as root user to the primary app server node.
2. Download the sudo **1.9.5p2** source distribution from the webpage, https://www.sudo.ws/download.html#source
3. Follow installation steps from the link - https://www.sudo.ws/install.html
4. Post installation, verify the sudo version using the command: **sudo -V**
5. Repeat above steps on all nodes of the Virtual Appliance.

Introduced in 12.1_Patch_v16

Utilities Purging

Prime Service Catalog patch v16 has added a new purging utility - Service Item Subscription purge.

The Service Item subscription purge utility deletes subscription and history data from the database related to service Item. This data is no longer used in the product and can be removed to reduce the database size. Executing this purge utility periodically could also provide overall performance improvement.

Steps to perform a purge

Step 1	Click the radio button next to Service Item Subscription Engine to choose the type of purge.
Step 2	Enter date ranges to filter the data to be purged. Date Range takes the submitted date of the service item.
Step 3	User can choose to enter a batch size and the total number of service item process. Batch Size: User has to enter the number of subscriptions that needs to be purged at a time in an iteration.
Step 4	

	Include Subscriptions of Open Requisition :User can select to purge subscriptions of the Requisitions that are open. By default only closed requisition will be purged
Step 5	Click Purge to start the purge.
Step 6	Click Yes on the Confirmation box to continue.
Step 7	The purge starts. Click OK.
Step 8	Click Refresh on Purge History pane after some time. When the purge or analysis completes, a new date/time entry is added in the Purge History pane at the top of the list.
Step 9	In the Purge History pane, click the purge completion date/time entry to see purge or analysis information in the Log Content pane on the right.

AMQP Authorization

Prime Service Catalog now supports AMQP authorization task which is similar to authorization using external task through the Service Link.

For this AMQP Agent need to be created with RabbitMQ adapter specified as both outbound and inbound. This agent should be specified as the workflow type in the authorization page.

The outbound connection to external Rabbitmq server details can be specified in the agent outbound properties as usual, however the inbound connection details for Rabbitmq server should be configured in the integration module. This is because for any AMQP connection details provided in integration module, PSC automatically creates the special queue named **'psc_inbound_queue'** for the connection and creates/manages a message listener for the queue to process the inbound messages. Possible inboundactions are approve, reject, ok, return to review to complete the authorization task which can be done by publishing the relevant action message in the **'psc_inbound_queue'**.

UCSD 6.7 Certification

Prime Service Catalog patch v16 is verified and certified with APIC UCSD 6.7.4.3 and fenced UCSD 6.7.3.0.
Note :User can update the properties mentioned below in the newscale.properties file for APIC container or VDC creation
# Flag to add/remove tenantID as prefix to tenant name when creating the APIC VDC
# By default flag is false and prefix will not be added to tenant name during VDC creation
ucsd.apic.tenant.addprefix=false

Introduced in 12.1_Patch_v15

Bulk Task Action

Prime Service Catalog patch v15 now supports Bulk task action where user can review various options. (Approve/Reject/Return to review).This is an asynchronous API call. Bulk task actions can be performed from the Service Catalog module.

Enabling bulk task action flag

Below mentioned are the properties to add in newscale.properties file to enable bulk task action flag.
Note: By default the property will be disabled.
Note: To log bulk task action separately user can enter valid log file location.

```
#####################################################
#Added to enable bulk approval feature.
#To enable:     servicecatalog.bulkapproval.enable=true
#To disable:    servicecatalog.bulkapproval.enable=false
#####################################################
servicecatalog.bulkapproval.enable=false
#####################################################
#Enable Bulk action log path here
servicecatalog.bulkapproval.log.path=C:\\log-folder
```

Authorization tab

- Approval, Reject and Return to review actions can be performed in the authorization tab and notifications section of Service Catalog module.
- After the flag is enabled, dropdown and checkboxes for selection will be enabled for the requisitions in Authorization tab. User will be able to select requisitions and perform the required bulk action.
- User can use “Being Approved” filter to search for authorisation tasks.

Notification tab

- When bulk approval flag is set to true, checkbox for each entry and ('Bulk Approve', 'Bulk Deny', 'Bulk Return to Review') buttons will be displayed (disabled but default).
- When at least one of the checkboxes is checked, then the buttons will be enabled and user can perform bulk task actions.

nsAPI to perform Bulk task action

- API: http://localhost:8080/RequestCenter/nsapi/transaction/tasks/action
- Method: POST
- Headers: Content-Type = application/json
  Payload:

```
{
   "taskActionList": {
      "actionId": 1,
      "comment": "comment here",
      "taskActions": [
        {
            "taskId": 1097
        },
         {
            "taskId": 1100
         },
                  ---------
          ----
            ---

      ]
   }
}
```

Note: Add “actionId” as
1 Approve, 2 Reject, 3 Return To review

Support for CloudCenter 4.10.0.9

By default Prime Service Catalog supports CloudCenter 4.9.0.1 integration. In patch v15 we are now supporting 4.10.0.9 also.

1. To Enable CloudCenter 4.10.0.9 integration add the following property to **newscale.properties** file and **support.properties** file as mentioned below.

```
##########################################################################################
# Flag to enable CloudCenter 4.10.0.9 version support
# By default below flag is false and it supports CloudCenter 4.9.0.1 version
#####################################
cloudcenter.4.10.0.9.enable=true
##########################################################################################
```

   - ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config**newscale.properties**
   - ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config**support.properties**

2. To create connection for CloudCenter Integration, the base templates to be selected are as below
    a. For CC 4.9.0.1 : *Cloud Center Base Template Service*
    b. For CC 4.10.0.9 : *Cloud Center 4.10.0.9 Base Template Service*

Note: Suspension policy and Scaling policy are not supported for Prime Service Catalog patch v15 with CloudCenter 4.10.0.9.

Certification with Apache Solr 8.6.2

Prime Service Catalog supports Solr integration only with SQL Server database.

Introduced in 12.1_Patch_v14

Duo Web Security Two Factor Authentication

Two-factor authentication adds a second layer of security to online accounts. Verifying identity using a second factor (like your phone or other mobile device) prevents from logging in, even in case of password compromised.

Duo Configurations

PSC supports Duo Security Two Factor authentication with a single factor log in process. Once the user logins Prime Service Catalog application a second layer of authentication is performed through the Duo Security Two Factor.

1. Duo server for has to be setup with the Web SDK application. Find the details in the link below. http://duo.com/docs/duoweb
2. Copy the Integration key, Secret key and API hostname while setting up the Web SDK application.
3. Enable the remembered devices for easy login.
4. Policy> edit global policy>enable remembered devices
5. Enable Duo Mobile (https://duo.com/docs/trusted-endpoints-duo-mobiles)
6. Enabling Duo self-service portal(https://duo.com/docs/self-service-portal

PSC Configuration for Duo Setup

Administration

1. Login as Site Administrator.
2. Go to Administration>Settings enable the flag “Enable DUO Login”. By default this feature ‘Enable DUO Login’ flag is off.

Integration module

1. Login as Site Administrator.
2. Add new DUO Integration connection from Integration module.
3. Enter the Integration key, Secret key and API hostname from Duo Web SDK application.
   Note :

   Newscale property to enable or disable Duo Integration test connection.
   ##
   # Duo Two-Factory Authentication
   # enables Duo integration test connection functionality with true/false values

```
# default value will be false
duo.integration.enable.test.connection=false
#################################################################
```

4. Note: To connect with Duo Two-Factor Authentication use application key along with Secret key. To change application key use Regenerate App key button from Duo Integration manage connection.
5. Logout and Login again.
6. After successful PSC login, it will prompt for Duo Two-Factor Authentication.
7. Follow the instructions that are shown in Duo Two-Factor Authentication and complete the login process.
8. PSC will be redirected with default login module.

Notes:

- PSC installed machine has setup with correct date and time format based on the machine time zone.
- PSC Duo Two-Factor Authentication supports local DB, LDAP, SAML and Windows based authentication.

Duo Two-Factor Authentication

Steps to Enroll DUO Web Security Two Factor Authentication

1. Login with your credentials.

2. Use your device to verify the authenticity. Your administrator can set up the system to do this via SMS, voice call, one-time passcode, the Duo Mobile smartphone app, and so on.

3. No mobile phone? You can use a landline or tablet, or ask your administrator for a hardware token. Duo lets you link multiple devices to your account, so you can use your mobile phone and a landline, a landline and a hardware token, two different mobile devices, etc.

Why Do We Need This?

1. Passwords are increasingly easy to compromise. They can often be stolen, guessed, or hacked — you might not even know someone is accessing your account.

2. Two-factor authentication adds a second layer of security, keeping your account secure even if your password is compromised. With Duo Push, you'll be alerted right away (on your phone) if someone is trying to log in as you.

3. This second factor of authentication is separate and independent from your username and password — Duo never sees your password.

Certification with Apache Solr 8.4.1

Prime Service Catalog supports Solr integration only with SQL Server database

Introduced in 12.1_Patch_v13

Republishing Service Link Messages

Prime Service Catalog now introduces a feature of automating the republishing of the service link messages using the poller. The poller wakes up weekly to get those requisitions whose service link messages were not published and republish the messages for the same.

The cut-off date parameter for the poller is by default seven days. This means the poller will look for requisitions created within the cut-off date(7 days).

For more details on manual configuration for the republish of service link messages you can refer https://www.cisco.com/c/en/us/td/docs/net_mgmt/datacenter_mgmt/intel_auto/service_portal/v_12_1/integration/Guide/CiscoPrimeServiceCatalog-12-1-IntegrationGuide.pdf

The poller can be configured as follows:

```
#Message Republish Data Poller
##########################################
#Cron Expression wakes up poller every day at 2 AM
```

messagerepublish.poller.cron=0 0 2 * * ?
#Cron Expression wakes up health check every 7day 2:30 AM
messagerepublish.poller.health.check.cron=0 30 2 * * ?
#High Availability Health checks threshold, this should be greater than Poller cron time specified in minutes, Poller will be killed if its running more than 2 hours 7 minutes
messagerepublish.healthCheck.threshold=127
messagerepublish.cutOffDate.days=7

support.properties

Message Republish Data Poller Settings
# In non-cluster mode: this should be enabled for the Message Republish script to be run from the Poller
# In clustered mode: this can be either enabled on all nodes in the cluster OR on a specific node in the cluster
# - only 1 node in the cluster will run at any given time, even if this is enabled on multiple nodes in a cluster (which ever node starts it first)
messagerepublish.poller.enable=true
For more details on manual configuations you can refer
https://www.cisco.com/c/en/us/td/docs/net_mgmt/datacenter_mgmt/intel_auto/service_portal/v_12_1/integration/Guide/CiscoPrimeServiceCatalog-12-1-IntegrationGuide.pdf

Support for WildFly 18.0.1 Application Server

One of the major release enhancements shipped in 12.1_Patch_v13, is support for WildFly 18.0.1 application server. Patch v13 onwards, Prime Service Catalog release 12.1 is supported on ***two*** versions of WildFly:

- 18.0.1
- 10.1.0

The procedure to apply Prime Service Catalog patches remains the same across all supported WildFly versions.

Migration Procedure from WildFly 10.1.0 to 18.0.1

ATTENTION:
12.1_Patch_v14 onwards, the clustered server lifecycle script errors, mentioned below, have been addressed for instances ***running on Microsoft Windows***. To apply the fix:

i. Navigate to the directory where 12.1_Patch_v14 (or a higher patch version) has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/cluster-bin/windows.

ii. Copy the contents of this directory over to the <PSC_Install_Dir>/bin directory **on VM #1** (the Domain Controller node), overwriting the existing scripts.

iii. The fix needs to be applied only on VM #1 (the Domain Controller node).

NOTE: Applicable for ***clustered*** WildFly environments (both 2-VM and 4-VM topologies). After migrating to WildFly 18.0.1, the following server lifecycle scripts (located in <PSC_Install_Dir>/bin) will throw some errors, when executed. ***Prime Service Catalog operation and functionality will not be affected due to these errors.***

- shutdownAllOnHC<n>, where n is any integer from 1-6
- deployServiceCatalogCluster
- undeployServiceCatalogCluster

Standalone WildFly Setup

1. Apply 12.1_Patch_v13 on your existing standalone WildFly 10.1.0 environment. Instructions can be found in the following section of this Readme document.

Section: *Patch Application Instructions* **>>** *Instructions for Standalone WildFly Setup*

2. Download the WildFly 18.0.1 distribution using the following URL. Copy it to a convenient location on your Prime Service Catalog host:

https://download.jboss.org/wildfly/18.0.1.Final/wildfly-18.0.1.Final.zip

The WildFly downloads home page is located at: https://wildfly.org/downloads/

3. Download the JBoss Server Migration Tool using the URL below. Copy it to the same location as the downloaded WildFly 18 distribution:

   https://github.com/wildfly/wildfly-server-migration/releases/download/1.7.0.Final/jboss-server-migration-1.7.0.Final.zip

4. Stop the Prime Service Catalog servers, and remove the server temp files from the following locations:
   - <PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceCatalogServer/tmp/
   - <PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceLinkServer/tmp/
5. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.
6. Extract *wildfly-18.0.1.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
   - wildfly-10.1.0.Final
   - wildfly-18.0.1.Final
7. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.
8. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the ServiceCatalogServer managed server:
   i. Uncomment and update the following server path configurations:

      server.source.standalone.serverDir=ServiceCatalogServer
      server.source.standalone.configDir=configuration
      server.source.standalone.configFiles=standalone.xml,standalone-full.xml
      server.target.standalone.serverDir=ServiceCatalogServer
      server.target.standalone.configDir=configuration

   ii. Update the modules.includes configuration as follows:

      modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

9. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

   jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final

10. Answer the series of questions presented by the script, as follows:
    i. *Migrate the source's standalone server?* **Yes**
    ii. *Migrate all configurations?* **Yes**
    iii. *This tool is not able to assert if the non-persistent deployments found are compatible with the target server, skip scanner's deployments migration?* **No**
    iv. *Migrate the source's managed domain?* **No**
11. Edit the *environment.properties* file again, to set up the ServiceLinkServer managed server migration:
    i. Update the following server path configurations:

       server.source.standalone.serverDir=ServiceLinkServer
       server.source.standalone.configDir=configuration
       server.source.standalone.configFiles=standalone.xml,standalone-full.xml
       server.target.standalone.serverDir=ServiceLinkServer
       server.target.standalone.configDir=configuration

    ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. Repeat steps 9-10.

13. Copy all the *.cli* files, from <PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceCatalogServer/configuration, to <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceCatalogServer/configuration. The files in question are:

- *commonCLI_commands_RC.cli*
- *oracleCLI_commands.cli*
- *sqlCLI_commands.cli*

14. Copy all the *.cli* files from <PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceLinkServer/configuration to <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceLinkServer/configuration. The relevant files are:

- *commonCLI_commands_SL.cli*
- *oracleCLI_commands.cli*
- *sqlCLI_commands.cli*

15. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.

16. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceCatalogServer/configuration. Open the *standalone-full.xml* file for editing, and search for the AJP, HTTP, and HTTPS listener definitions as shown below:

- <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
- <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
- <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***all three*** listeners:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

17. Repeat step 16, for the *standalone-full.xml* file located in <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceLinkServer/configuration.

18. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

19. If you are using Windows Services for starting and stopping the Prime Service Catalog servers:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *servicecatalogwrapper.conf* and *servicelinkwrapper.conf* files, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallServiceCatalogService

b. installServiceCatalogService

v. Re-install the “Cisco Prime Service Link” Windows Service To do this, execute the following scripts in the specified order:

a. uninstallServiceLinkService

b. installServiceLinkService

20. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

Section: *Migration Procedure From WildFly 10.1.0 to 18.0.1* **>>** Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 18.0.1

21. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.
22. Start the Service Link server, followed by Service Catalog.
23. Using the following URLs, access the WildFly administration consoles, for the ServiceCatalogServer and ServiceLinkServer managed servers, and verify the WildFly version is correct.
    - Service Catalog console: <protocol>://<PSC_IP_or_Hostname>:9990/console
    - Service Link console: <protocol>://<PSC_IP_or_Hostname>:7990/console
24. Use the following URL to verify Prime Service Catalog is accessible. Log in, and navigate to the Service Link module, to confirm the connection status is active (green bubble).

    <protocol>://<PSC_IP_or_Hostname>:8080/RequestCenter
25. If you're using IIS as your web server, and the web server is installed on the same host where Prime Service Catalog is installed:
    i. Open IIS Manager, and select the website for Prime Service Catalog. Review the file system paths for the RequestCenter and IntegrationServer virtual directories.
    ii. If the physical paths begin with <PSC_Install_Dir>\wildfly-10.1.0.Final\..., update them to point to the respective folders inside the <PSC_Install_Dir>\wildfly-18.0.1.Final directory.
    iii. Restart the IIS website.
26. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Clustered WildFly Setup; 2-VM Topology

1. Apply 12.1_Patch_v13 on your existing, clustered WildFly 10.1.0 environment. Instructions can be found in the following section of this Readme document.

   Section: *Patch Application Instructions* **>>** *Instructions for Virtual Appliance and Clustered Wildfly Setup with 2-VM Topology*
2. Once the patch has been successfully applied, undeploy both *RequestCenter.war* and *ISEE.war.* To do this, log in to VM #1 (the Domain Controller node), navigate to <PSC_Install_Dir>/bin, and execute the following script:

   undeployServiceCatalogCluster ALL
3. Stop the Prime Service Catalog servers completely, across ***all*** nodes.
4. Log in to VM #1 (the Domain Controller node).
5. Using the following URL, download the WildFly 18.0.1 distribution to a convenient location on this host:

   https://download.jboss.org/wildfly/18.0.1.Final/wildfly-18.0.1.Final.zip

   The WildFly downloads home page is located at: https://wildfly.org/downloads/
6. Download the JBoss Server Migration Tool to the same location as the WildFly 18 distribution. The tool can be obtained from

   https://github.com/wildfly/wildfly-server-migration/releases/download/1.7.0.Final/jboss-server-migration-1.7.0.Final.zip
7. Remove the server temp files from the following locations:
   - <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host1-RC/tmp
   - <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host1-SL/tmp
8. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.
9. Extract *wildfly-18.0.1.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
   - wildfly-10.1.0.Final
   - wildfly-18.0.1.Final
10. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.

11. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

i. Uncomment and update the following server path configurations:

server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain.xml
server.source.domain.hostConfigFiles=host.xml,hostva_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final

13. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* ***No***

ii. *Migrate the source's managed domain?* ***Yes***

iii. *Migrate all configurations?* ***Yes***

iv. *This tool is not able to assert if persistent deployments found are compatible with the target server, skip persistent deployments migration?* ***No***

v. *Migrate all persistent deployments found?* ***Yes***

vi. *Migrate all configurations?* ***Yes***

14. Copy all the *.cli* files from <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/configuration to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration. The files in question are:

- *cluster_commonCLI_commands.cli*
- *cluster_ha_profileCLI_commands.cli*
- *cluster_nonha_profileCLI_commands.cli*
- *cluster_oracleCLI_commands_ha.cli*
- *cluster_oracleCLI_commands_nonha.cli*
- *cluster_sqlCLI_commands_ha.cli*
- *cluster_sqlCLI_commands_nonha.cli*

15. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.

16. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration. Open the *domain.xml* file for editing, and locate ***all instances*** of the following listener definitions:

i. <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>

ii. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

iii. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

17. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.
18. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:
    i. Navigate to the <PSC_Install_Dir>/conf directory.
    ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.
    iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.
    iv. Re-install the "Cisco Prime Service Catalog Cluster" Windows Service. To do this, execute the following scripts in the specified order:
        a. uninstallProcessControllerClusterService
        b. installProcessControllerClusterService
19. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

    Section: *Migration Procedure From WildFly 10.1.0 to 18.0.1* **>>** *Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 18.0.1*
20. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.
21. Start the Prime Service Catalog cluster on VM #1.
22. Deploy both *RequestCenter.war* and *ISEE.war*. To do this, navigate to <PSC_Install_Dir>/bin, and execute the following script:

    deployServiceCatalogCluster ALL
23. Using the following URL, access the administration console for the WildFly domain, and verify the WildFly version is correct:

    <protocol>://<VM1_IP_or_Hostname>:9990/console
24. Use the following URL to verify Prime Service Catalog is accessible. Log in, and navigate to the Service Link module, to confirm the connection status shows green.

    <protocol>://<VM1_IP_or_Hostname>:8080/RequestCenter
25. Log in to VM #2 (Host Controller #2).
26. Download the WildFly 18.0.1 distribution and the JBoss Server Migration Tool to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.
27. Remove the server temp files from the following location:

    <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host2-RC/tmp
28. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.
29. Extract the *wildfly-18.0.1.Final.zip* file inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
    - wildfly-10.1.0.Final
    - wildfly-18.0.1.Final
30. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.
31. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:
    i. Uncomment and update the following server path configurations:

    server.source.domain.domainDir=domain
    server.source.domain.configDir=configuration

server.source.domain.domainConfigFiles=domain_backup.xml
server.source.domain.hostConfigFiles=host.xml,host2_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

32. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final

33. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* ***No***

ii. *Migrate the source's managed domain?* ***Yes***

iii. *Migrate all configurations?* ***Yes***

iv. *Migrate all configurations?* ***Yes***

34. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.

35. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration.

36. Delete the *domain.xml* file.

37. Open the *domain_backup.xml* file for editing. Locate ***all instances*** of the following listener definitions:

i. <ajp-listener name="ajp" socket-binding="ajp"/>

ii. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

iii. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

38. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

39. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

a. uninstallProcessControllerClusterService

b. installProcessControllerClusterService

40. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.

41. Start the Prime Service Catalog cluster on VM #2.

42. Use the following URL to verify that cluster node #2 for Prime Service Catalog is available.

<protocol>://<VM2_IP_or_Hostname>:8080/RequestCenter

43. At this point, the migration process is complete, for a stock 2-VM topology cluster. If you have additional Host Controller nodes configured, repeat steps 25-42 on each of the additional nodes, while noting the following differences:

i. For Host Controller <N>, the server temp files are located at:

<PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host<N>-RC/tmp

ii. In *environment.properties*, the server.source.domain.hostConfigFiles property value should be

server.source.domain.hostConfigFiles=host.xml,host<N>_backup.xml

iii. When performing step 42, use the IP address of the VM corresponding to the Host Controller you just configured

44. If you're using IIS as your web server, and the web server is installed on any of the cluster nodes:

i. Log in to the cluster node where IIS is installed.

ii. Open IIS Manager, and select the website for Prime Service Catalog. Review the physical paths for the RequestCenter and IntegrationServer virtual directories.

iii. If the physical paths begin with <PSC_Install_Dir>\wildfly-10.1.0.Final\content\..., update them to point to the respective folders inside the <PSC_Install_Dir>\wildfly-18.0.1.Final\content directory.

iv. Restart the IIS website.

45. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Clustered WildFly Setup; 4-VM Topology

1. Apply 12.1_Patch_v13 on your existing, clustered WildFly 10.1.0 environment. Instructions can be found in the following section of this Readme document.

Section: *Patch Application Instructions* **>>** *Instructions for Clustered Wildfly Setup with 4-VM Topology*

2. Once the patch has been successfully applied, undeploy *RequestCenter.war.* To do this, log in to VM #1 (the Domain Controller node), navigate to <PSC_Install_Dir>/bin, and execute the following script:

undeployServiceCatalogCluster RC

3. Stop the Prime Service Catalog servers completely, across ***all*** nodes.

4. Log in to VM #2 (the Service Link node).

5. Using the following URL, download the WildFly 18.0.1 distribution to a convenient location on this host:

https://download.jboss.org/wildfly/18.0.1.Final/wildfly-18.0.1.Final.zip

The WildFly downloads home page is located at: https://wildfly.org/downloads/

6. Download the JBoss Server Migration Tool to the same location as the WildFly 18 distribution. The tool can be obtained from

https://github.com/wildfly/wildfly-server-migration/releases/download/1.7.0.Final/jboss-server-migration-1.7.0.Final.zip

7. Remove the server temp files from the following location:

<PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceLinkServer/tmp/

8. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.

9. Extract *wildfly-18.0.1.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

- wildfly-10.1.0.Final
- wildfly-18.0.1.Final

10. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.

11. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the ServiceLinkServer managed server:

i. Update the following server path configurations:

server.source.standalone.serverDir=ServiceLinkServer
server.source.standalone.configDir=configuration
server.source.standalone.configFiles=standalone.xml,standalone-full.xml
server.target.standalone.serverDir=ServiceLinkServer
server.target.standalone.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

12. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final

13. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* **Yes**

ii. *Migrate all configurations?* **Yes**

iii. *This tool is not able to assert if the non-persistent deployments found are compatible with the target server, skip scanner's deployments migration?* **No**

iv. *Migrate the source's managed domain?* **No**

14. Copy all the *.cli* files from <PSC_Install_Dir>/wildfly-10.1.0.Final/ServiceLinkServer/configuration to <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceLinkServer/configuration. The relevant files are:

- *commonCLI_commands_SL.cli*
- *oracleCLI_commands.cli*
- *sqlCLI_commands.cli*

15. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.

16. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceLinkServer/configuration. Open the *standalone-full.xml* file for editing, and search for the AJP, HTTP, and HTTPS listener definitions as shown below:

- <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
- <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
- <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***all three*** listeners:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

17. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

18. If you are using Windows Services for starting and stopping the Prime Service Catalog servers:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *servicelinkwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Link” Windows Service To do this, execute the following scripts in the specified order:

a. uninstallServiceLinkService

b. installServiceLinkService

19. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.

20. Start the Service Link server.

21. Log in to VM #1 (the Domain Controller node).

22. Download the WildFly 18.0.1 distribution and the JBoss Server Migration Tool to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.

23. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.

24. Extract *wildfly-18.0.1.Final.zip* inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:

- wildfly-10.1.0.Final
- wildfly-18.0.1.Final

25. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.

26. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

i. Uncomment and update the following server path configurations:

```
server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain.xml
server.source.domain.hostConfigFiles=host.xml,host_default.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration
```

ii. Update the modules.includes configuration as follows:

```
modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl
```

27. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

```
jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final
```

28. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* ***No***

ii. *Migrate the source's managed domain?* ***Yes***

iii. *Migrate all configurations?* ***Yes***

iv. *This tool is not able to assert if persistent deployments found are compatible with the target server, skip persistent deployments migration?* ***No***

v. *Migrate all configurations?* ***Yes***

29. Copy all the *.cli* files from <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/configuration to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration. The files in question are:

- *cluster_commonCLI_commands.cli*
- *cluster_ha_profileCLI_commands.cli*
- *cluster_nonha_profileCLI_commands.cli*
- *cluster_oracleCLI_commands_ha.cli*
- *cluster_oracleCLI_commands_nonha.cli*

- *cluster_sqlCLI_commands_ha.cli*
- *cluster_sqlCLI_commands_nonha.cli*

30. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.
31. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration. Open the *domain.xml* file for editing, and locate ***all instances*** of the following listener definitions:
    - <ajp-listener name="ajp" socket-binding="ajp" scheme="http" max-ajp-packet-size="65536"/>
    - <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
    - <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

    Add the following attributes to ***each instance***:
    - allow-unescaped-characters-in-url="true"
    - max-parameters="5000"
32. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.
33. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:
    i. Navigate to the <PSC_Install_Dir>/conf directory.
    ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.
    iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.
    iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:
        a. uninstallProcessControllerClusterService
        b. installProcessControllerClusterService
34. If you are using Prime Service Catalog without the Service Link component, additional configuration changes are required. Refer to the following section of this Readme document.

    Section: *Migration Procedure From WildFly 10.1.0 to 18.0.1* **>>** *Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 18.0.1*
35. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.
36. Start the Prime Service Catalog cluster on VM #1.
37. Re-deploy *RequestCenter.war*. To do this, navigate to <PSC_Install_Dir>/bin, and execute the following script:

    deployServiceCatalogCluster RC
38. Log in to VM #3 (Host Controller #1).
39. Download the WildFly 18.0.1 distribution and the JBoss Server Migration Tool to a convenient location on this host. Refer to steps 5 and 6 for the download URLs.
40. Remove the server temp files from the following location:

    <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host1-RC/tmp
41. Take a backup of the existing wildfly-10.1.0.Final directory located in <PSC_Install_Dir>.
42. Extract the *wildfly-18.0.1.Final.zip* file inside <PSC_Install_Dir>. At this point, there will be two WildFly directories inside <PSC_Install_Dir>:
    - wildfly-10.1.0.Final

- wildfly-18.0.1.Final

43. Extract the file, *jboss-server-migration-1.7.0.Final.zip*. This will create a directory called jboss-server-migration.

44. Navigate to the jboss-server-migration/configuration directory. Make the following changes to the *environment.properties* file, in preparation for migrating the managed WildFly domain:

i. Uncomment and update the following server path configurations:

server.source.domain.domainDir=domain
server.source.domain.configDir=configuration
server.source.domain.domainConfigFiles=domain_backup.xml
server.source.domain.hostConfigFiles=host.xml,host1_backup.xml
server.target.domain.domainDir=domain
server.target.domain.configDir=configuration

ii. Update the modules.includes configuration as follows:

modules.includes=com.newscale.jdbc,com.microsoft.sqlserver,javax.ccpp,javax.portlet,net.sf.ehcache,oracle.jdbc,org.apache.pluto.container.driver,org.apache.pluto.container.om,org.apache.pluto.tags,org.osgi.core,org.jboss.sasl

45. From a terminal window, navigate to the jboss-server-migration directory, and execute the jboss-server-migration script using the following syntax:

jboss-server-migration --source <PSC_Install_Dir>/wildfly-10.1.0.Final --target <PSC_Install_Dir>/wildfly-18.0.1.Final

46. Answer the series of questions presented by the script, as follows:

i. *Migrate the source's standalone server?* ***No***

ii. *Migrate the source's managed domain?* ***Yes***

iii. *Migrate all configurations?* ***Yes***

iv. *Migrate all configurations?* ***Yes***

47. Navigate to the directory where 12.1_Patch_v13 has been extracted. Once inside, navigate to /database/Wildfly18 Upgrade/modules. Copy the contents of this directory over to <PSC_Install_Dir>/wildfly-18.0.1.Final/modules.

48. Navigate to <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration.

49. Delete the *domain.xml* file.

50. Open the *domain_backup.xml* file for editing. Locate ***all instances*** of the following listener definitions:

i. <ajp-listener name="ajp" socket-binding="ajp"/>

ii. <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

iii. <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>

Add the following attributes to ***each instance***:

- allow-unescaped-characters-in-url="true"
- max-parameters="5000"

51. Navigate to the <PSC_Install_Dir>/bin directory. Edit the setEnv script, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

52. If you are using Windows Services for starting and stopping the Prime Service Catalog cluster:

i. Navigate to the <PSC_Install_Dir>/conf directory.

ii. Open the *processcontrollerClusterwrapper.conf* file, for editing, and update the JBOSS_HOME environment variable to point to the WildFly 18.0.1 directory: <PSC_Install_Dir>/wildfly-18.0.1.Final.

iii. From a terminal window, navigate to <PSC_Install_Dir>/bin.

iv. Re-install the “Cisco Prime Service Catalog Cluster” Windows Service. To do this, execute the following scripts in the specified order:

   a. uninstallProcessControllerClusterService
   b. installProcessControllerClusterService

53. Delete the wildfly-10.1.0.Final directory inside <PSC_Install_Dir>.
54. Start the Prime Service Catalog cluster on this node.
55. Repeat steps 38-54 on VM #4 (Host Controller #2), noting the following differences:

   i. Remove the server temp files from the following location:

      <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host2-RC/tmp

   ii. Configure the server path configurations in *environment.properties*, as follows:

      server.source.domain.domainDir=domain
      server.source.domain.configDir=configuration
      server.source.domain.domainConfigFiles=domain_backup.xml
      server.source.domain.hostConfigFiles=host.xml,host2_backup.xml
      server.target.domain.domainDir=domain
      server.target.domain.configDir=configuration

56. Using the following URL, access the administration console for the WildFly domain, and verify the WildFly version is correct:

   <protocol>://<VM1_IP_or_Hostname>:9990/console

57. Verify that Prime Service Catalog is accessible using ***both*** the following URLs. Log in, and navigate to the Service Link module, to confirm the connection status is active (green bubble).

   - <protocol>://<VM3_IP_or_Hostname>:8080/RequestCenter
   - <protocol>://<VM4_IP_or_Hostname>:8080/RequestCenter

58. At this point, the migration process is complete, for a stock 4-VM topology cluster. If you have additional Host Controller nodes configured, repeat steps 38-54, and step 57, on each of the additional nodes, while noting the following differences:

   i. For Host Controller <N>, the server temp files are located at:

      <PSC_Install_Dir>/wildfly-10.1.0.Final/domain/servers/server-host<N>-RC/tmp

   ii. In *environment.properties*, the server.source.domain.hostConfigFiles property value should be

      server.source.domain.hostConfigFiles=host.xml,host<N>_backup.xml

   iii. When performing step 57, use the IP address of the VM corresponding to the Host Controller you just configured.

59. If you’re using IIS as your web server, and the web server is installed on either VM #3, VM #4, or any of the additional Host Controller cluster nodes:

   i. Log in to the cluster node where IIS is installed.

   ii. Open IIS Manager, and select the website for Prime Service Catalog. Review the file system path for the RequestCenter virtual directory.

   iii. If the physical path is set to <PSC_Install_Dir>\wildfly-10.1.0.Final\content\RequestCenter.war, update it to point to: <PSC_Install_Dir>\wildfly-18.0.1.Final\content\RequestCenter.war.

   iv. Restart the IIS website.

60. Verify that Prime Service Catalog can be accessed properly through the web server URL.

Configuration Changes for Hosting Prime Service Catalog Without Service Link on WildFly 18.0.1

If your Prime Service Catalog instance is hosted without the Service Link component, the following additional configuration changes are needed as part of the migration procedure to WildFly 18.0.1.
NOTE: General feature documentation, on how to configure Prime Service Catalog to operate without the Service Link component, can be found in the ***Cisco Prime Service Catalog 12.1 Patch Product Enhancement Guide*** published on Cisco.com.

Standalone WildFly Setup

1. On your Prime Service Catalog host, navigate to the following directory:

   <PSC_Install_Dir>/wildfly-18.0.1.Final/ServiceCatalogServer/deployments/RequestCenter.war/WEB-INF/classes/config

2. Open the beeeCamelContext.xml file, for editing. Locate the following bean definitions:

   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_1">
     <property name="from" value="activemq:queue:jms.queue.BEEEAuthorizationsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_2">
     <property name="from" value="activemq:queue:jms.queue.BEEERequisitionsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_3">
     <property name="from" value="activemq:queue:jms.queue.BEEEInboundQueue{Queue_Suffix}"/>

   NOTE: {Queue_Suffix} placeholder is the value passed for the optional queue name suffix parameter, during the execution of the jmsrunner utility. The placeholder may be empty.

   For ***all*** three definitions, remove the “jms.queue” substring from the value of the value attribute. Save the changes.

Clustered WildFly Setup; 2-VM and 4-VM Topologies

NOTE: In case of a clustered setup, it is recommended that you apply the following configuration changes once the migration process to WildFly 18.0.1 is complete.

1. On the Domain Controller node (VM 1 in both topologies) of your Prime Service Catalog installation, navigate to the following location inside the extracted *RequestCenter.war*:

   <PSC_Install_Dir>/tmp/RequestCenter.war/WEB-INF/classes/config

2. Open the beeeCamelContext.xml file, for editing. Locate the following bean definitions:

   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_1">
     <property name="from" value="activemq:queue:jms.queue.BEEEAuthorizationsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_2">
     <property name="from" value="activemq:queue:jms.queue.BEEERequisitionsQueue{Queue_Suffix}"/>
   - <bean class="com.newscale.bfw.util.jms.CamelRoute" id="route_3">
     <property name="from" value="activemq:queue:jms.queue.BEEEInboundQueue{Queue_Suffix}"/>

   NOTE: {Queue_Suffix} placeholder is the value passed for the optional queue name suffix parameter, during the execution of the jmsrunner utility. The placeholder may be empty.

   For ***all*** three definitions, remove the “jms.queue” substring from the value of the value attribute. Save the changes.

   In order for the above changes to take effect, the post-customizationOnDC script needs to be executed—a new *RequestCenter.war* file will be created and deployed into the Prime Service Catalog cluster.

3. Navigate to the <PSC_Install_Dir>/wildfly-18.0.1.Final/domain/configuration directory.

4. Edit *domain.xml*. Locate ***all*** instances of the snippet

   <subsystem xmlns="urn:jboss:domain:messaging-activemq:8.0">
   <server name="default">

   and add the following child element

   <remote-connector name="remote-artemis" socket-binding="remote-artemis"/>

   for the <server> tag. Save the changes.

Introduced in 12.1_Patch_v12

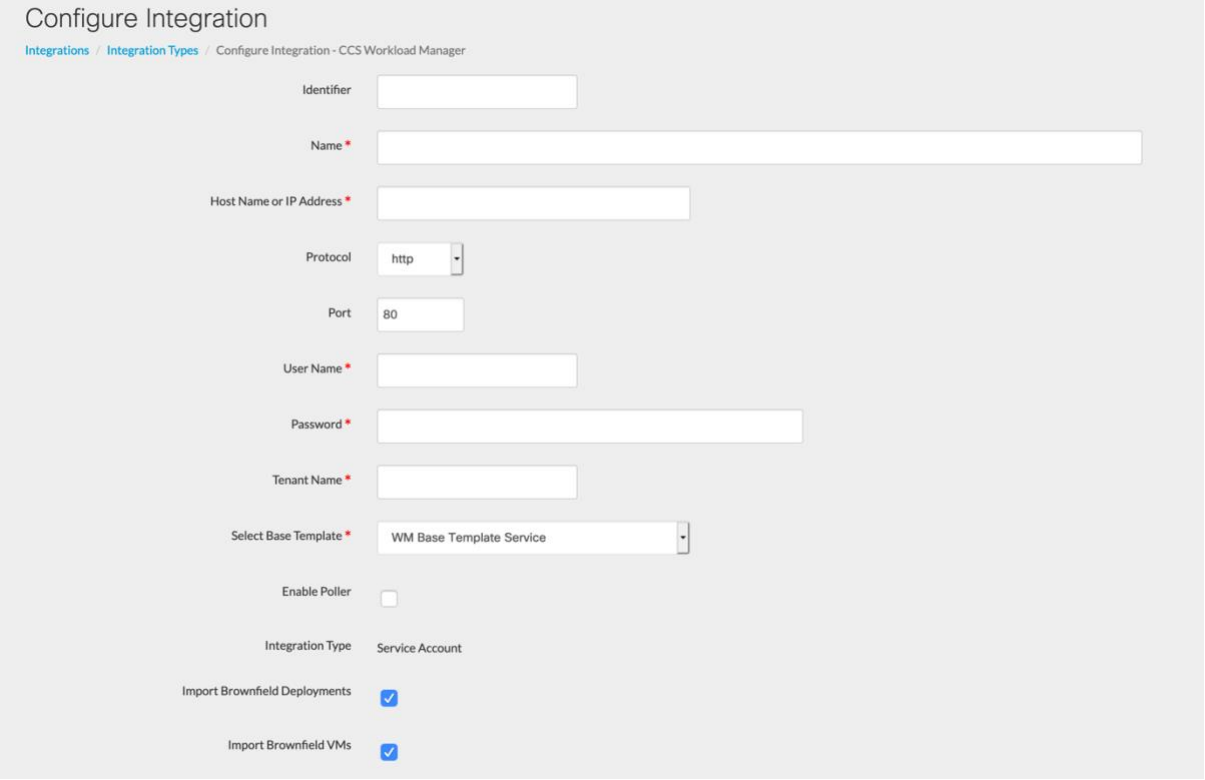
CCS 5.1x Workload Manager Integration

Prime Service Catalog now supports the following features of Workload Manager:

Adding Workload Manager Connection into PSC

Steps to add Workload Manager Connection in the PSC

1. Click Integration under Integration Module
2. Select Integration type as “CCS Workload Manager”



Note: Users - Only Integration Administrator User or Admin user will be able to create a connection.

- Identifier – Unique Identifier
- Name – Name of the Integration
- Host Name or IP Address – IP where the CC501 Suite is hosted
- Protocol – http/https
- Port – Port of CC501 Suite
- User Name – CC501 User should contains the tenant admin and suite admin groups
- Tenant Name: Name of the tenant
- Password – Password of the system
- Import Brownfield Deployments: If enabled, the workload manager will import to PSC.
- Import Brownfield VM’s: If enabled, the workload manager VM’s will import to PSC
- Enable Poller- The poller is to sync based on poller setting in Newscale and support properties (if user wants to enable below properties they should be part of Newscale and support)

Properties to be added for newscale.properties

To sync with API token, Cloud centre suite gets the latest token as per the poller configuration(CCS Token Poller). For example, if poller is configured to run every10th minute, then poller gets the latest token from CCS and updates PSC.

Note: The below mentioned settings are **mandatory** for PSC server to be up and running. The entries pertain to the Workload Manager and need to be added even if you are not using this feature. In case if the CCS token does not get enabled then the webservices for DDR will not work.

#WM Poller settings
##
#Cron Expression wakes up poller every 10 minutes of an hour
wm.poller.cron=0 0/10 * * * ?
#Cron Expression wakes up health check for 10th min and 15 mins thereafter of the hour ex: 10,25,40,55 minutes
wm.poller.health.check.cron=0 10/15 * * * ?
#High Availability Health checks threshold, this should be greater than Poller cron time specified in minutes
wm.healthCheck.threshold=91
##

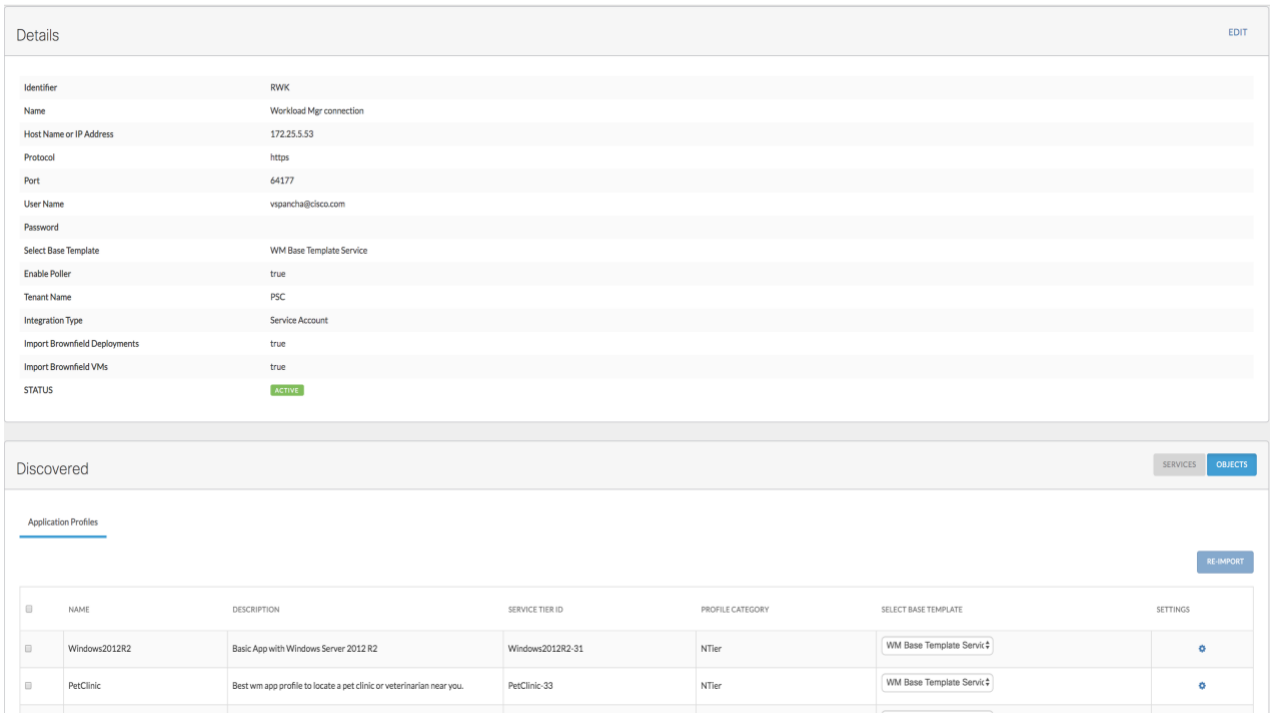
#CCS Token Poller settings
##
#Cron Expression wakes up poller every 10 minutes of an hour
ccstoken.poller.cron=0 0/10 * * * ?
#Cron Expression wakes up health check for 10th min and 15 mins thereafter of the hour ex: 10,25,40,55 minutes
ccstoken.poller.health.check.cron=0 10/15 * * * ?
#High Availability Health checks threshold, this should be greater than Poller cron time specified in minutes
ccstoken.healthCheck.threshold=91
##

Properties to be added for support.properties

WM Poller Settings
#Flag to enable/disable the poller
wm.poller.enable=true

CCS Token Poller Settings
#Flag to enable/disable the poller
ccstoken.poller.enable=true

3. Click Import for all objects (it will take some time to import the application profiles, activation profiles ,deployments ,VM's).
4. Application profiles will be converted as services in PSC and same will be available in SD under the WM Services.(Identifiers name) The same dictionary group and AFC will be created for workload manager services too.
5. The Manage integration – application profile and service will list as mentioned in the below screen shot.



Service Account Based integration

Workload Manager connection is associated with a common service account which is used for all requests such as application deployments, operations, and actions irrespective of the user who creates it. Users with permissions to deploy applications will be the owners of these applications. Users with read-write access on these deployments or VMs is permitted to perform actions.

Import Brownfield Deployments/VMs

When an integration is created Prime Service Catalog supports importing of deployments and VMs created in Cloud Centre Suite which are discovered by PSC and are then displaying on the Service Items page with Workload Manager.

Import Brownfield Deployments and Import Brownfield VMs are two options which are provided in the Create Workload Manager integration. You can choose to import the deployments and/or VMs for that connection during creation.

Note: The above mentioned setting can be edited at any point of time after the creation of the connection.

Import Selected Application Profiles

Option to Re-Import is introduced to refresh and reimport only the selected application profiles for Workload Manager connection. If there are any changes made to the application profiles in Workload Manager, on clicking Re-Import for the selected application profiles, the changes made are reflected in Prime Service Catalog and also services are regenerated.

This option is available under **Integrations > Workload Manager Integration > Manage Integrations > Application profiles.**

Order Workload Manager Service for a Group

Users with permissions to order Workload Manager deployments can now order a Workload Manager service for a group that the user belongs. Read/Write permission on the ordered Service Item will be provided to the selected Group and all users of the groups can perform actions on the deployment.

Option to Display Group List in Service Form is introduced. You can find it under Administration > Settings.

When enabled, the group field is displayed in the service form for any of the Workload Manager Service being ordered. The list will contain all the groups that the user of the order belongs to. Only the site administrator can enable this option under the administration settings.

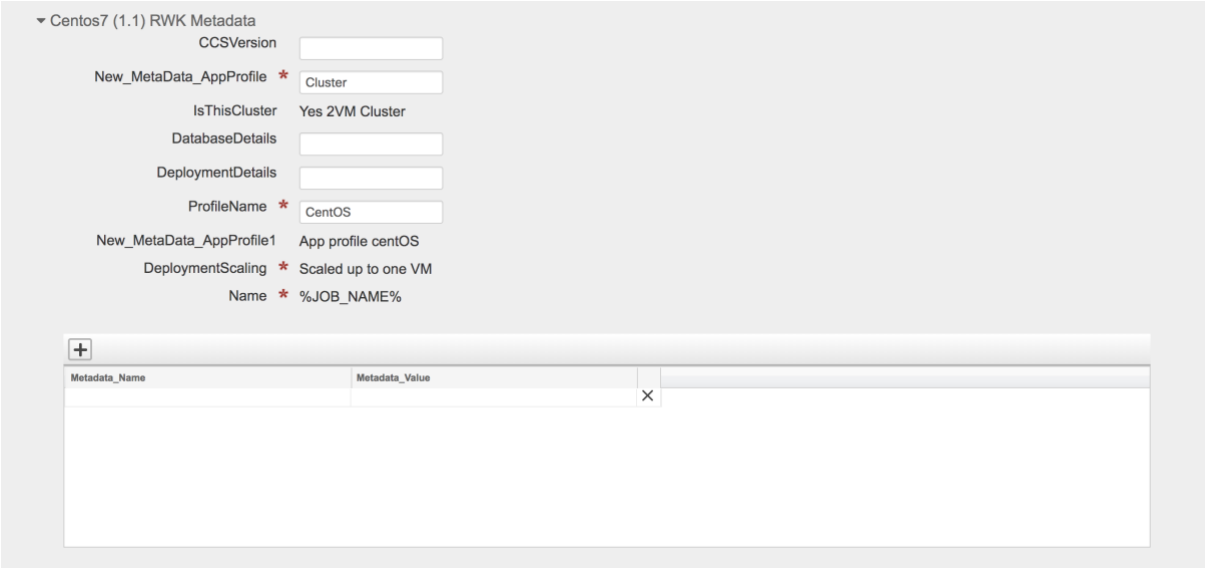
Import Application Profile Metadata Tags

If the application profiles have any metadata set for custom properties, the metadata are also imported to Prime Service Catalog during importing a Workload Manager connection. The metadata information is then used to build a new dictionary and new service form, and this service form is tied to the Workload Manager application service.

Note: The dictionary and service forms that are created by the system should not be tampered with. If it is tampered, Workload Manager deployment will fail.

Note: You can add the special characters in the metadata form while ordering Workload Manager services. But for “” special characters, you should add the below escape sequence.

\“System is having 32 GB\”



Deployment Actions

Prime Service Catalog performs a full-sync of both application and VM. The status is derived depending on the status of application or VM on Workload Manager and the details of the VM such as, CPU, RAM, HD, Public and Private IP Address, status on the VM

are displayed on the Summary Details Page.

Name	422d7217-9c20-dfdf-d979-16aef2b5a42e
Display Name	cqjw-24b889053
External ID	422d7217-9c20-dfdf-d979-16aef2b5a42e
Status	Active
Host Name	cqjw-24b889053
Public IP	172.25.5.10
Private IP	172.25.5.10
Start Time	2019-11-21 19:02:31.331
Cloud Connection ID	76
Cloud VM Type	DEPLOYMENT_VM
Cloud ID	2
Cloud Account ID	2
Cloud Region ID	2
External VM ID	380
External User ID	1
Termination Protection	0
Job ID	720
Virtual CPUs	1
Memory (MB)	2048
Instance Type	1CPU_2048MBMEM
Tags	New_MetaData_AppProfile1:App profile centOS,IsThisCluster:Yes 2VM Cluster,DeploymentScaling:Scaled up to one VM,ProfileName:CentOS,CliQrUserId:1,Name:shNov21_CentOS,associatedTags:PrivateCloud,PrivateCloud:USER_DEFINED_TAG
OS Family	Linux
Parent Job ID	719

Sync

The Sync Deployment action allows you to sync each deployment with Workload Manager and get the current status of particular deployment. For example, if changes are made in the deployments from Workload Manager such as, deployment is suspended or new node is added then, such changes can be synched immediately to the Prime Service Catalog using this action instead of waiting for the specified polling interval or manual import.

Migrate

The operation Migrate Deployment action triggers to migrate deployment operation in Cloud Centre.

By default, a copy of the deployment is created on the target cloud and the deployment is deleted on the original cloud. But you could choose to keep the deployment on the original cloud by choosing “Yes” from the option keepExistingDeployment on the deployment service form.

Delete

The delete action will delete the deployment and associated Vm’s.

Suspend

The Suspend action will power off the deployment and associated Vm’s.

VMs Actions

These operations are available only to the users with read/write permissions on the deployments or VMs. (Service Account based)

Power On/Power Off/Reboot

Depending on the status of the VM, you can:

- Power On: Power on and start the VM
- Power Off: Stop and power off the VM
- Reboot: Attempts a graceful shut down and restart of the VM

Attach Volume

The attach volume option allows you to attach multiple volumes to all tier types in entire applications. For each volume, you must specify the size, storage type for each root disk. The attach volume operations invoke a service which can be ordered for others or for oneself.

Detach volume

The detach volume option is used to remove the disk volumes from the VM for any user.

Create Snapshot

Creates the image snapshot for the specified VM.

Sync

This VM option gets the latest information of VM (Unmanaged VM, Deployed VM, and imported VM) from the Workload Manager to the Prime Service Catalog, instead of waiting for Poller trigger or manual import. Also syncs the deployment details if it is part of the deployment.

Import

There are two kinds of brownfield VMs that can be imported into Prime Service Catalog.

- Managed VMs: VMs that are managed by Workload Manager. For example: VMs that were created as a result of deployments created in Workload Manager. Or VMs that were imported into Workload Manager from a cloud.
- Unmanaged VMs: VMs that exist in the cloud account connected to Workload Manager. These are not imported into Workload Manager. Users have limited permissions on the Unmanaged VMs, to manage these VMs from Prime Service Catalog VMs, you must first import the VM to Prime Service Catalog. On the Service Items page, permitted actions on Unmanaged VMs are terminated
- Import—Imports the VM to Prime Service Catalog and Workload Manager and converts the VM status to Managed VM.

Terminate

This action will permanently delete the VM.

Console Access of VMs

You can now access the Workload Manager VMs from a web-based console from the Service Items page of the VM. User with the Read/Write on a managed VM can access the VM console.

- For a Windows VM:

Click on RDP button on the right-hand side to launch the console on a new webpage button. The console is launched on a new webpage.

- For a Linux VM:

Click on the SSH button on the right-hand side to launch the console on a new webpage.

Service Items Search

A search criterion display name is available to filter the Service Items by display name of the service item.

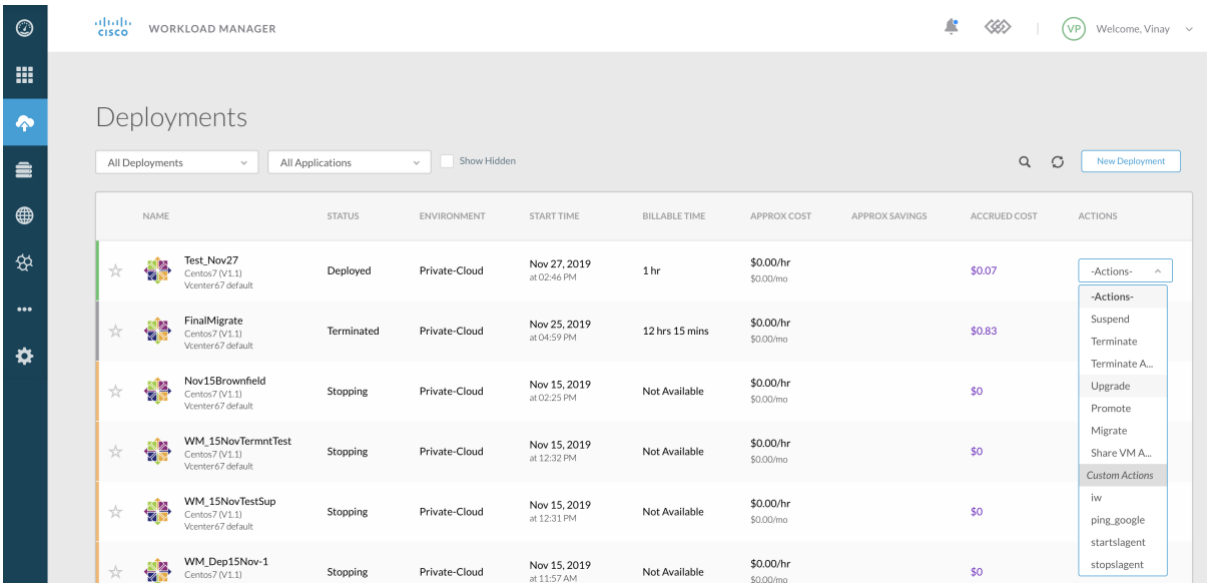
Support for Custom Actions on Workload Manager VMs and Deployments

To support custom actions on Workload Manger VMs and deployments follow the below procedure

Pre-requisite :

- Create a Custom Actions in WM- Action Library
- Go to Action Library – Click on New Action
- Select an Action type command script for VM's and Invoke webservice type for deployment
- Custom actions should be created and command script action type will be displayed for managed VM's under the action menu.
- Action type having invoke webservice should be displayed for workload manager deployment action menu as shown in the below screen shot.

Custom Actions for Deployments



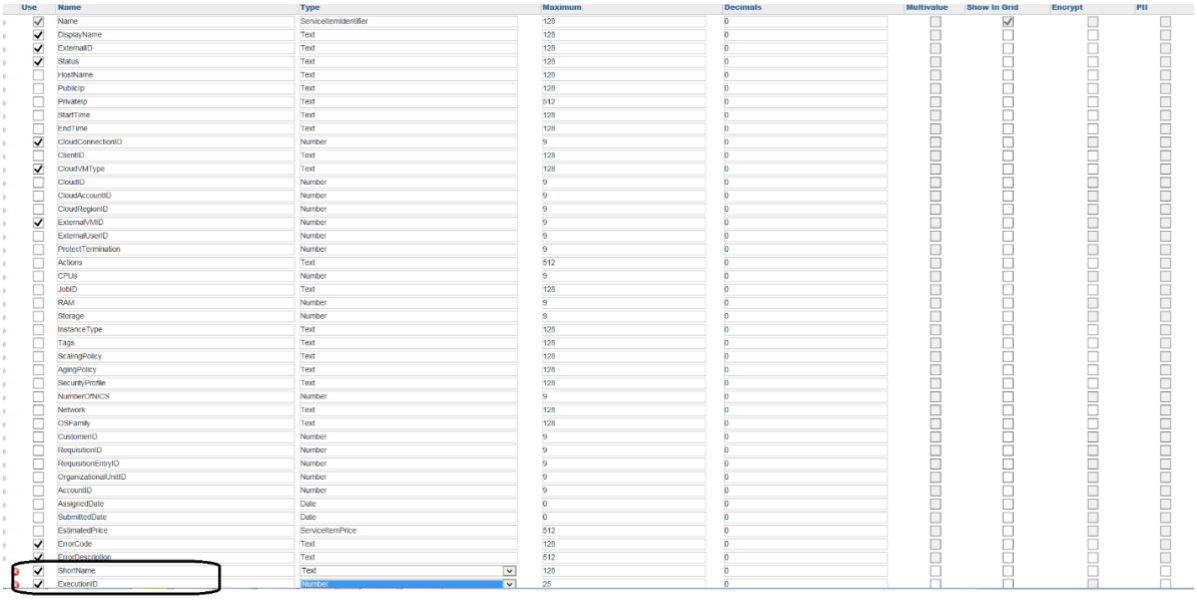
Custom Actions for Managed Virtual Machines

Designing a Dictionary, Service Forms and Service to consume the WM Custom Actions for Managed Vm's

Creating SI based Dictionary

1. Create a service-item based dictionary by selecting WM Application Virtual Machine

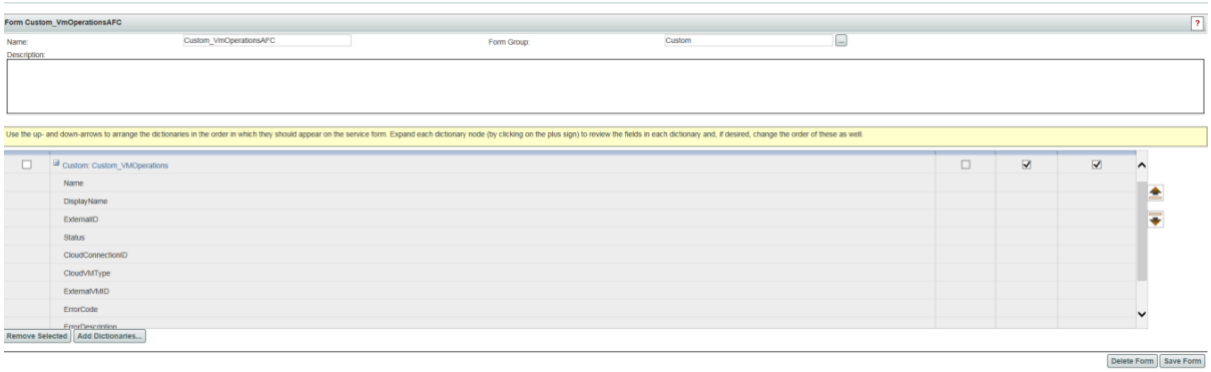
1. Choose **Service Designer > Dictionaries.**
2. Choose **New > New Dictionary** to display the New Dictionary page.
3. In the Add New Internal Dictionary section under the Service Item field, enter WM Application Virtual Machine and select the service item.
4. Enter details such as Dictionary Name and Group Name.
5. From the Dictionary Attributes section, click Add Field to add a user-defined field **ShortName and ExecutionID**.
6. Select all the fields in the dictionary that are required for the custom action (any fields that are used in custom FTL)Note: Make sure that the ExecutionID field is added to the dictionary.
7. **Save Dictionary**.



Creating Service Form

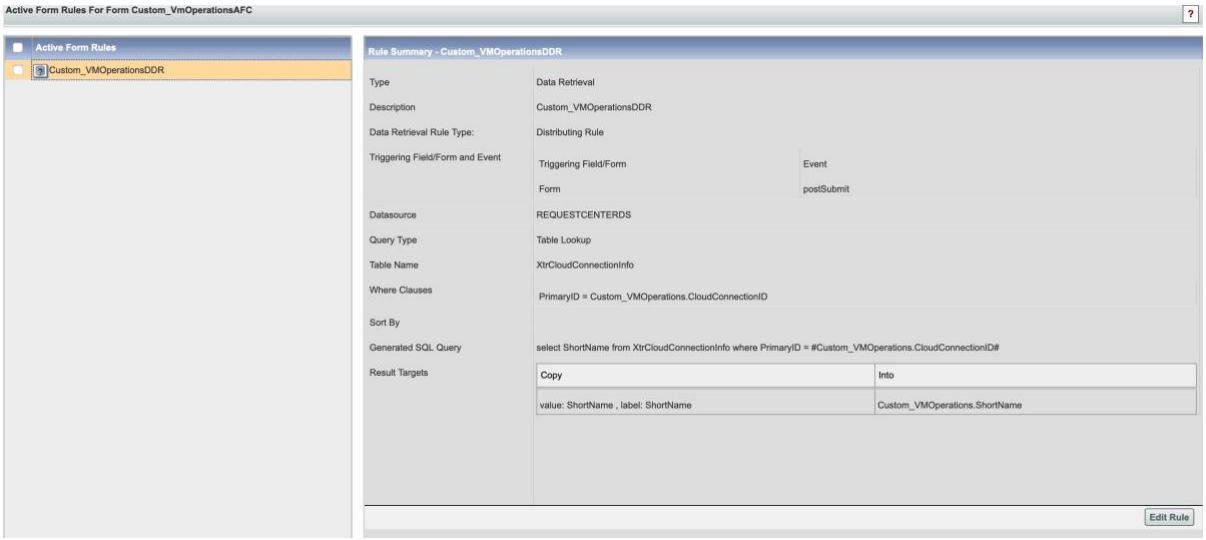
2. Create an active form component and add the above created dictionaries to the form.
    1. Choose **Service Designer > Active Form Component**.
    2. Choose **New > Active Form Component**.
    3. Enter a name and brief description for the new form.
    4. Click on the form group field and select one of the groups to associate with the form.
    5. Click **Save Form**.
    6. Choose Form Content tab and click **Add Dictionaries**.

       In the Add Dictionaries dialog box, search for the dictionary created in above section (Creating SI based Dictionary) and select it. and click **Add**.
       Note: Make sure you select the VMs that belongs to workload manager and all the mandatory dictionary fields is populated before form saving.
    7. Click Save Form.



Creating DDR rule to populate the short name Field

3. Creating a data retrieval rule. Which can populate Short Name field
    1. Go to **Active Form Rules** tab of the above created Active Form Component (Creating Service Form)
    2. Choose **New Rule** > **New Data Retrieval Rule**
    3. In the first page of the Data Retrieval Rule wizard, enter a unique name and a description for the rule, and specify the Rule Type as *Distributing Rule.*
    4. Specify other details in the wizard as in the below figure and **Save Rule** to populate the shortname**.**



Creating a service for the Custom Action on the Managed VM

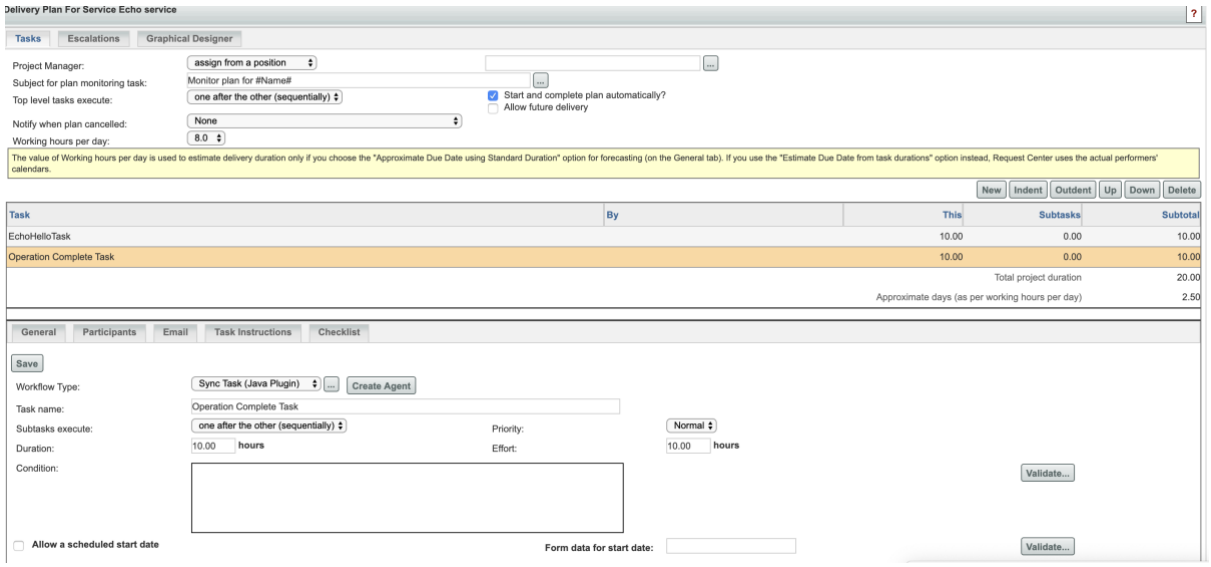
4. Create a Custom Service (for example, CustomOperationService) and add the form to the service.

1. Choose **Service Designer** > **New** > **New Service**.
2. Enter the details in the fields provided.
3. Click **Add This Service**.
4. After adding the service, you can begin to configure it by entering information on the **General** tab.
5. Click **Save**.
6. Click **Form** tab of the service created.
7. Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.
8. In the Search field, enter the form name created in step (Creating Service Form)
9. Check the form and then click **Add**.

Creating task for the service to execute the custom action in WM

5. Create the two tasks specified below in the **Plan** tab of the service.

Perform VM Operation Task and Operation Complete Task are mandatory. The Update Status task is optional.

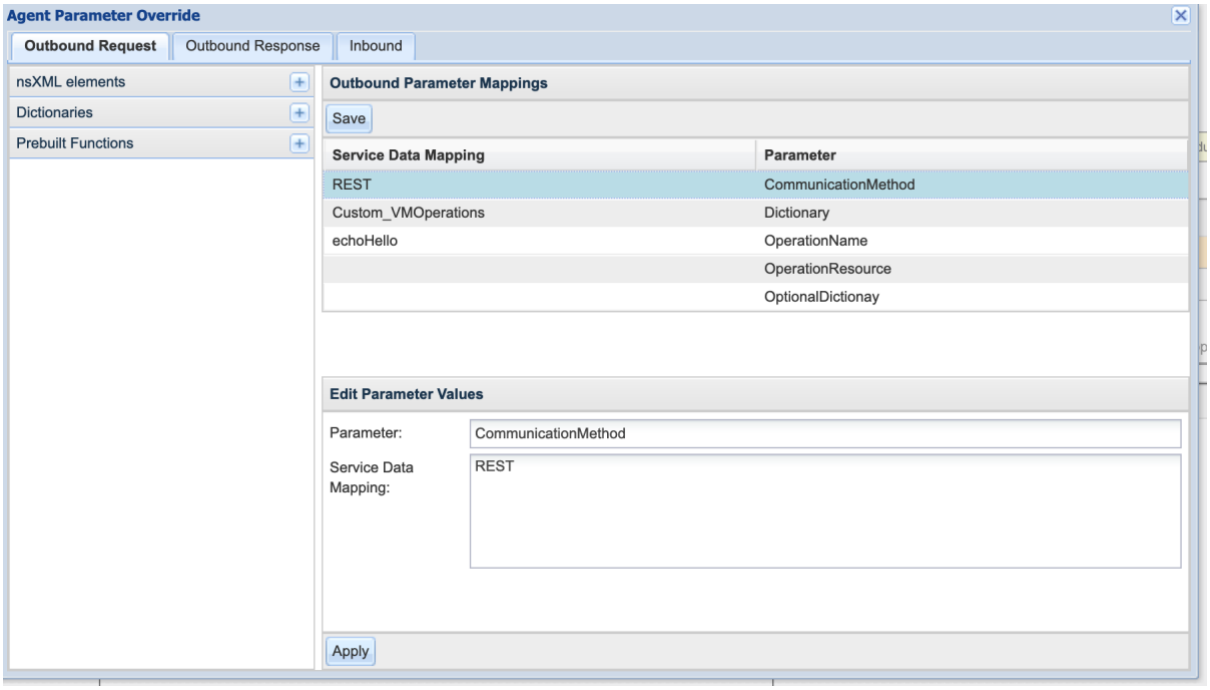


Creating task for a service

- Perform VM Operation Task – This task workflow type is WMAgent (select from the dropdown). Configure the agent paraments as below

Parameter	Service Data Mapping
Communication-Method	REST
Dictionary	Enter the name of the dictionary that was created in Step (Creating SI based Dictionary)
OperationName	Enter the custom operation name. It is recommended to use the <Cus- tomOperationName> which should match the action name in the Workload Manager For example, if the action name specified is **echohello** in Workload Manager then the name of the custom operation name should be echohello.

Below is the screen shot for the Agent Paraments -



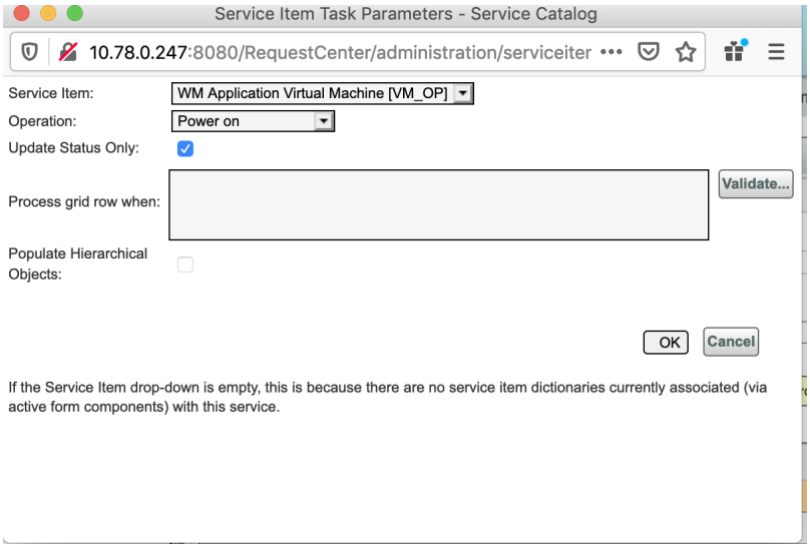
- Operation Complete Task

  This task workflow type will be Synch Task (Java Plugin) (select from the dropdown) – and configure the plugin task paraments as below

  Sync Task Plugin Class- com.celosis.event.SyncTaskWMPlugin
  Param1: <Dictionary Name (Created in the Create SI based Dictionary).JobID
  Param2: The custom Action name. It is recommended to use the which should match the action name in the Workload Manager
  Param3: <Dictionary Name (Created in the Create SI based Dictionary).Status
  Param4: <Dictionary Name (Created in the Create SI based Dictionary).ShortName
  Param5: WM

  Save the service

  Note : When User is trying to create a custom action for lifecycle operations (poweron/poweroff/reboot etc) and if the user need to update an intermediate status for a VM then update status task could be added as the first task as shown below



Mapping above service to the Service item (to make this service as action for a vm)

1. Add the Custom Services in **Service Item Manager > Design Service Items > CCS Worklaod Manager > WM Application Virtual Machines > Associated Services**.
2. Click on Add Service (and search for a service which is created in the Create a Service Section and select the service and save. This would show up services in the gear icon of the VM.

Creating FTL and Modify the Properties Files

Below is the example for custom action FTL file (Example for Echo Hello Command Custom action)

Note – For FTL file we need to update the dictionary (Creating SI based Dictionary Section) and also update the action id for a custom action created in Workload-Manager (which you can get using custom action WM API)

Once the FTL file is ready it should be placed in the below locations :

- -wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
- -wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud

```
<#--
*  FTL will support Headers,Post parameters, url (get) Parameters and payload.
*
*  ================================================================
*  Header *******   Header_propertyname=Propertyvalue
*  Post   *******   Post_propertyname=Propertyvalue
*  BaseUrl *******   give you the url and fallowd  by query parameters
*  Payload   *******   Payload=payload content
*  Header_AuthToken  **** auth token/cookie used for authentication
*  ================================================================
*
* Header_Accept_Encoding=UTF-8
* Header_Content-Type=application/xml
*  These parameters will applies to the payload type only.Propertyvalues can be changed based on the type.
*
*InputMap should provide BaseUrl and authKey values.
*
group, catalog, container, comments
-->


Protocol=https
```

RequestMethod=POST
CommunicationMethod=REST
AuthenticationMethod=Header
Header_Content-Type=application/json
Header_Accept=application/json
Header_Accept-Encoding=UTF-8
Header_Authorization=${authKey}
BaseUrl=https://${authority}/cloudcenter-ccm-backend/api/v1/actions/56/executions
Payload={
"resourceType": "VIRTUAL_MACHINE",
"executionResources": [
{
"id": "<#list doc['message']['task-started']['requisition']['requisition-entry']['data-values']['data-value'] as datavalue><#if (datavalue['name'])=='**Custom_VMOperations**.ExternalVMID'>${datavalue['value']}</#if></#list>"
}
]
}
AssertResponseStatus=$.id::notNullValue
ExtractResponseStatus=$.id
ExtractResponseStatusError=$.errors[0].message

Modify the Intercloud.properties files

Open the Intercloud.properties file from below location

wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config
Add the entry of FTL file in below format at the end of the file

WM_<name of the CustomAction created in the WM>=json,config/cloud/<FTL File Name>

Ex- WM_echoHello=json,config/cloud/WM_echoHello.ftl

Modify the below properties files

- Open the file SyncCustomWMVMExecutionOperations.properties in the below paths:

  wildfly-10.1.0.Final \ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
  wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud
  Add the below entry into the file
  <Operation_Name>=Operation_Name
  Example - echoHello=echoHello
  Note – If the User wants create a custom actions for Lifecycle operations like (PowerON/PowerOFF/Reboot) then below files should be modified – open the below file from below location -

  wildfly-10.1.0.Final \ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
  wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud

  UpdateWMVMHistoryOperations.properties
  <Operation_Name>=Operation_Name
  Example - powerOnVMCustom=powerOnVMCustom

  UpdateWMVMStatusOperations.properties
  <Operation_Name>=Operation_Name
  Example - powerOnVM=powerOnVM

3. Clear the server cache and restart the server.
   Order the custom services and all the task should be completed
   Verify the Comments and History in the Ordered Requisition (proper comments should be updated in the comments and history section)

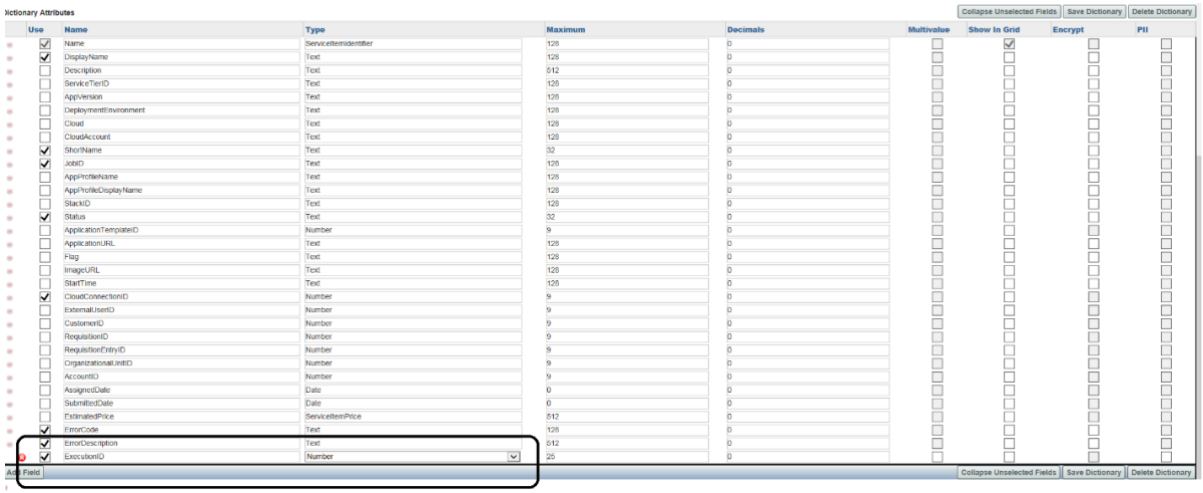
   Below is the example of screen shot –

Designing a Dictionary ,Service Forms and Service to consume the Custom Actions for Deployments

To support custom actions on Workload Manager deployments follow the below procedure:

Creating SI based Dictionary

1. Create a service-item based dictionary by selecting WM *Application Stack* as Service item type.
    1. Choose **Service Designer > Dictionaries.**
    2. Choose **New > New Dictionary** to display the New Dictionary page.
    3. In the Add New Internal Dictionary section, in the Service Item field, enter WM Application Stack and select the service item.
    4. Enter details such as Dictionary Name and Group Name.
    5. From the Dictionary Attributes section, click Add Field to add a user-defined field **ShortName, JobID, and ExecutionID**.
    6. Select all the fields in the dictionary that is required for the custom action (Any fields that are used in custom FTL). Make sure that the JobID field is selected and ExecutionID field is added to the dictionary.
    7. Click **Save Dictionary**

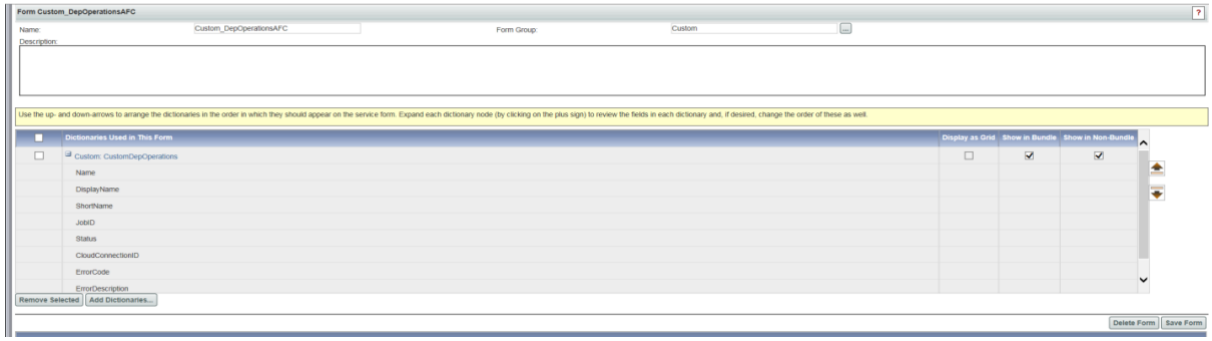


Creating Service Forms

2. Create active form component and add the above created dictionaries to the form.
   1. Choose **Service Designer > Active Form Component**.
   2. Choose **New > Active Form Component**.
   3. Enter a name and brief description for the new form.
   4. Click on the form group field and select one of the groups to associate with the form.
   5. Click **Save Form**.
   6. Choose Form Content tab and click **Add Dictionaries**.
   7. In the Add Dictionaries dialog box, search for the dictionary created in Create SI Based Dictionary Section and select it. And click **Add**.

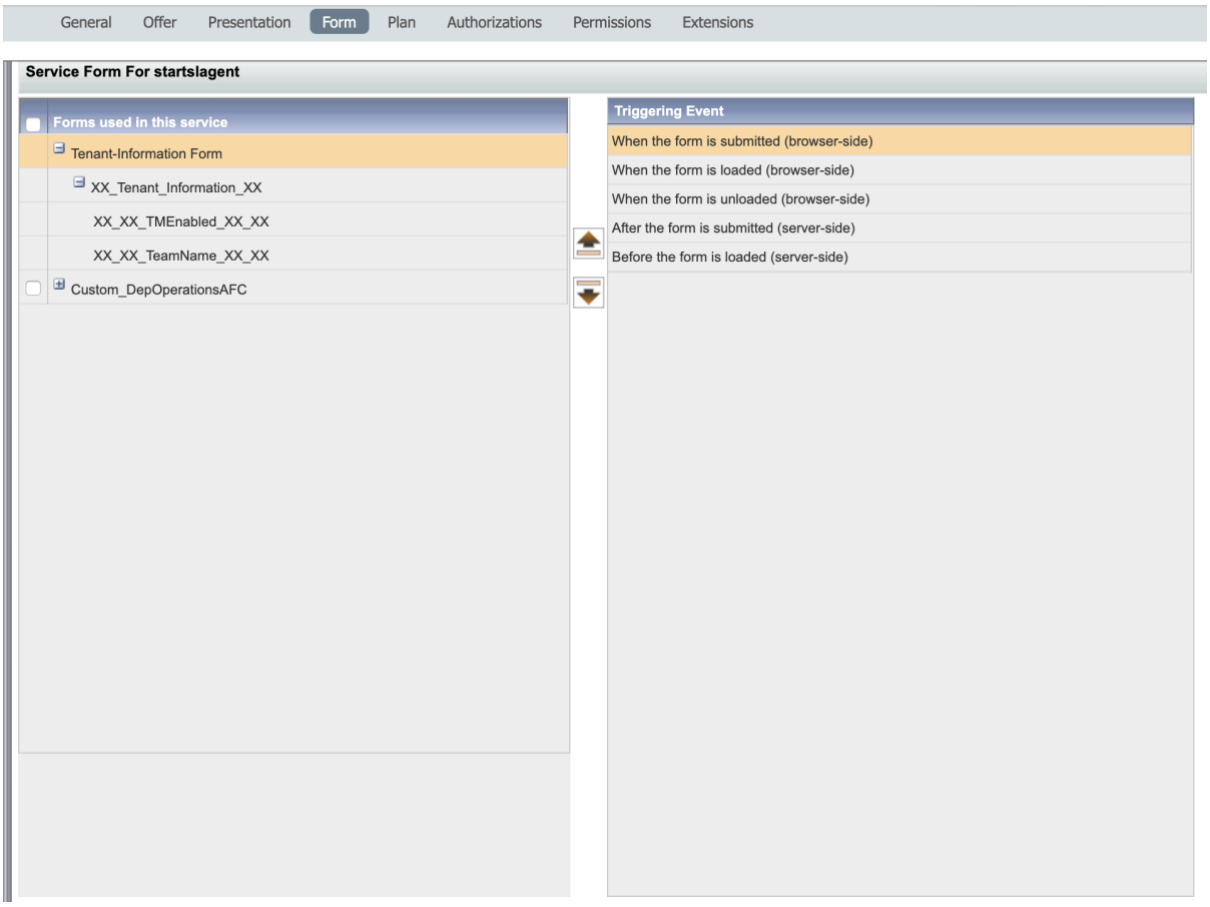
Note: Make sure you select the Deployments that belongs to Workload Manager and all the mandatory dictionary fields is populated before form Saving.

   8. Click Save Form.



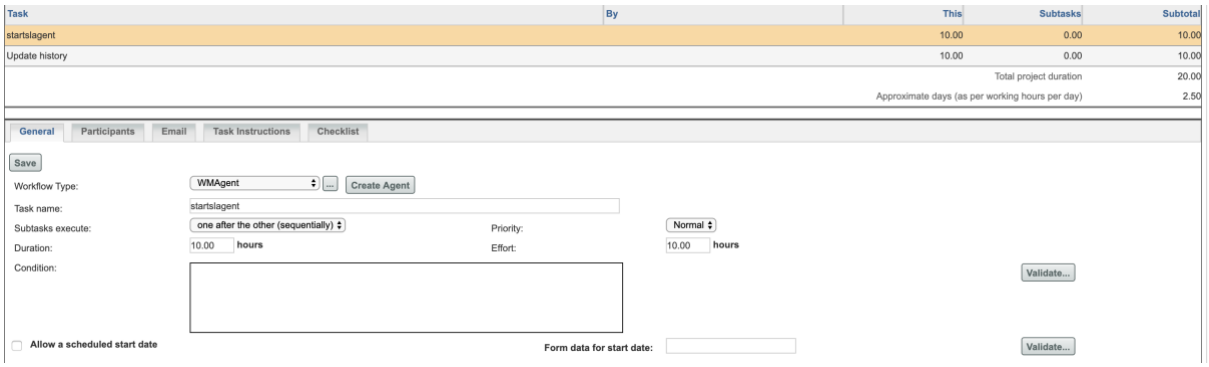
Creating a Service to Consume the Custom Actions

3. Create a Custom Service (for example, CustomOperationService) and add the form to the service.
   1. Choose **Service Designer** > **New** > **New Service**.
   2. Enter the details in the fields provided.
   3. Click **Add This Service**.
   4. After adding the service, you can begin to configure it by entering information on the **General** tab.
   5. Click **Save**.
   6. Click **Form** tab of the service created.
   7. Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.
   8. In the Search field, enter the form name created in Create Service Form Section
   9. Check the form and then click **Add**.



Create a Task for a Service

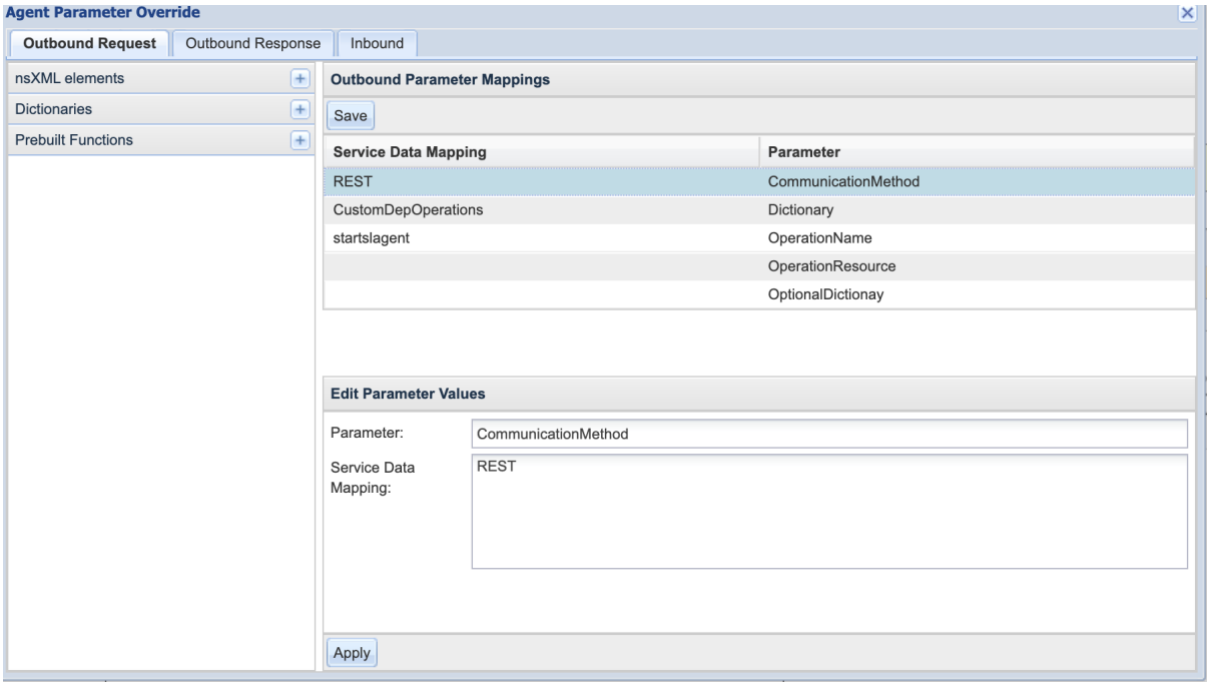
4. Create the two tasks specified below in the **Plan** tab of the service.
   Perform Deployment Operation Task and Operation Complete Task are mandatory. The Update Status task is optional.



To Perform the custom actions on deployment below two tasks should be created.

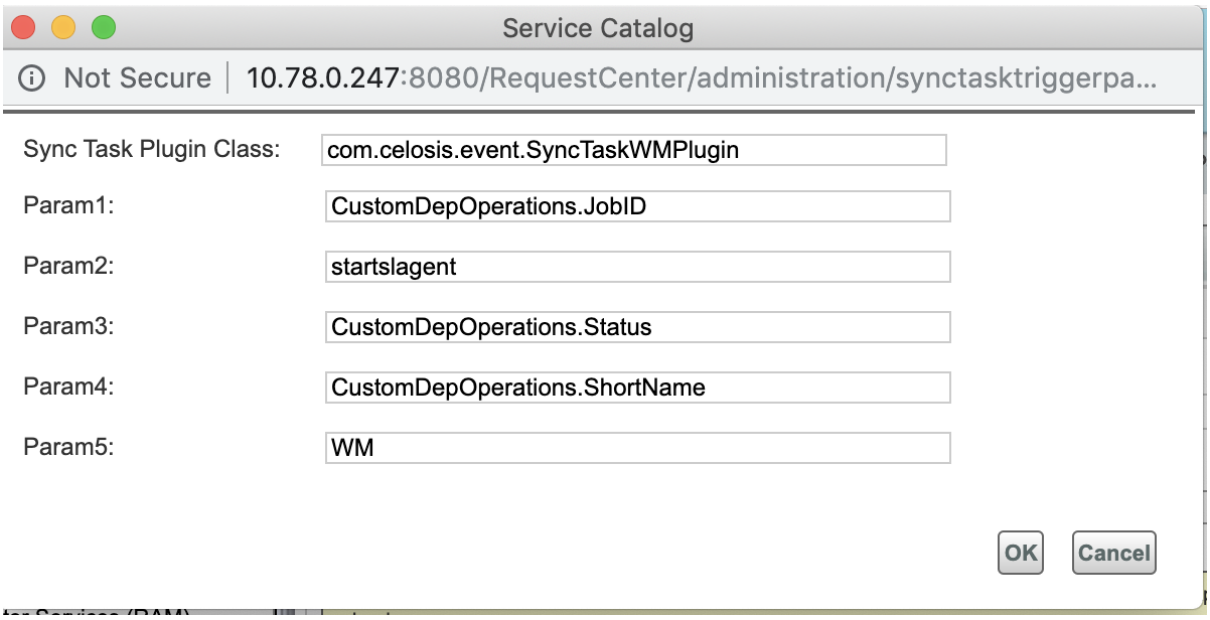
- **Perform Action on Deployment** – This Task of Workflow type Mapped to WMAgent (Select from the drop down menu of workflow type)and Below is Agent Parameter details need to be updated.

Parameter	Service Data Mapping
Communication-Method	REST
Dictionary	Enter the name of the dictionary that was created in the Create a SI Based Dictionary Section
OperationName	Enter the custom action name. It is recommended to use the <CustomOperationName> which should match the action name in the Worklaod Manager For example, if the action name specified is **startslagent** in Worklaod Manager then the name of the custom operation name should be **startslagent** .



- **Operation Complete Task –** Create a task with Synch Task (Java Plugin) select from the drop down menu and update the details in the Popup

Sync Task Plugin Class- com.celosis.event.SyncTaskWMPlugin
Param1: <Dictionary Name (Created in the Create SI based Dictionary).JobID
Param2: The custom action name. It is recommended to use the which should match the action name in the Workload Manager
Param3: <Dictionary Name (Created in the Create SI based Dictionary).Status
Param4: <Dictionary Name (Created in the Create SI based Dictionary).ShortName
Param5: WM



5. Save the service.

Mapping above service to the Service item (to make this service as action for a Deployment)

6. Add the Custom Service in **Service Item Manager > Design Service Items > CCS Worklaod Manager> WM Application Stack - Associated Services**.
7. Click on Add Service (and search for a service which created in the Create a Service Section) select the service and save. This would show up services in the gear icon of the deployment.

Creating FTL and Modify the Properties Files

Below is the example of the Custom action FTL file for deployment (Example for start SL agent Invoke webservice Custom action

Note – FTL file we need to update the dictionary (Create SI Based Dictionary Section) and Action id for a custom action created in Workload-Manager (which can get using custom action WM API)

Once the FTL file is ready it should be place in the below locations

- wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
- wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud

```
<#--
*  FTL will support Headers,Post parameters, url (get) Parameters and payload.
*
*  ================================================================
*  Header *******   Header_propertyname=Propertyvalue
*  Post   *******   Post_propertyname=Propertyvalue
*  BaseUrl *******   give you the url and fallowd  by query parameters
*  Payload   *******   Payload=payload content
*  Header_AuthToken  **** auth token/cookie used for authentication
*  ================================================================
*
* Header_Accept_Encoding=UTF-8
* Header_Content-Type=application/xml
*  These parameters will applies to the payload type only.Propertyvalues can be changed based on the type.
*
*InputMap should provide BaseUrl and authKey values.
```

```
*
group, catalog, container, comments
-->

Protocol=https
RequestMethod=POST
CommunicationMethod=REST
AuthenticationMethod=Header
Header_Content-Type=application/json
Header_Accept=application/json
Header_Accept-Encoding=UTF-8
Header_Authorization=${authKey}
BaseUrl=https://${authority}/cloudcenter-ccm-backend/api/v1/actions/57/executions
Payload={
  "resourceType":"DEPLOYMENT",
  "executionResources":[
    {
      "id":"<#list doc['message']['task-started']['requisition']['requisition-entry']['data-values']['data-value'] as datavalue><#if
(datavalue['name'])=='CustomDepOperations.JobID'>${datavalue['value']}</#if></#list>"
    }
  ]
}
AssertResponseStatus=$.id::notNullValue
ExtractResponseStatus=$.id
ExtractResponseStatusError=$.errors[0].message
```

Modify the Intercloud.properties file

Open the Intercloud.properties file from below location

wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config
Add the entry of FTL file in below format at the end of the file

WM_<name of the CustomAction created in the WM>=json,config/cloud/<FTL File Name>

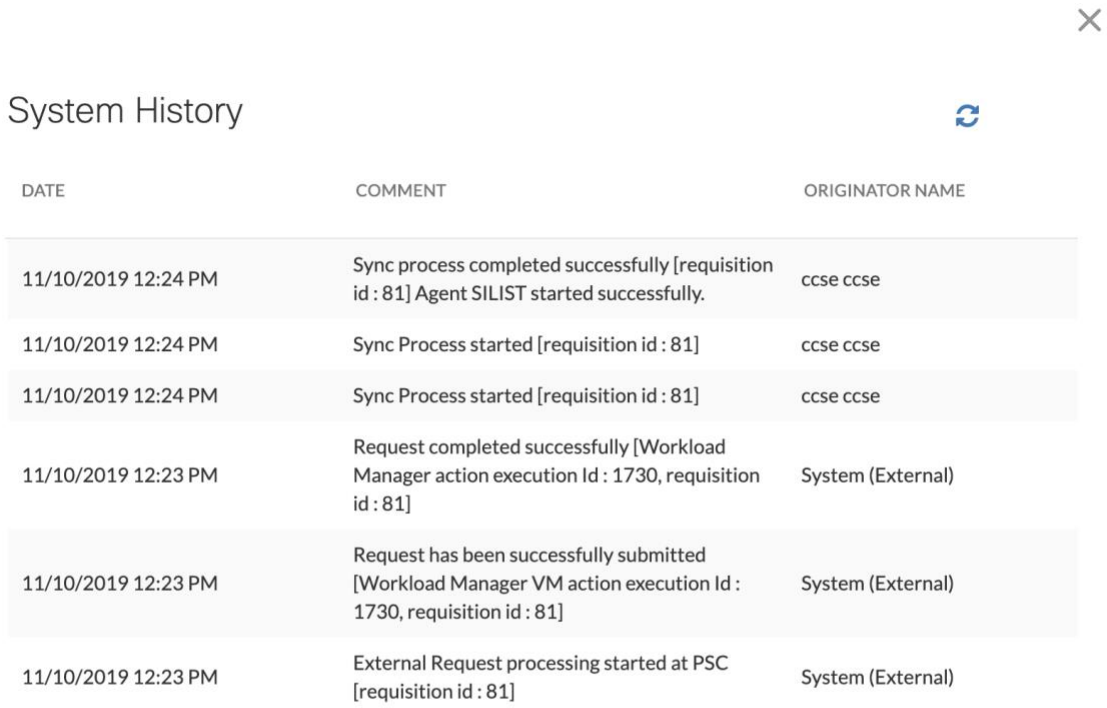
Ex- WM_echoHello=json,config/cloud/startslagent.ftl

Modify the below properties files

- Open the file SyncCustomWMDeploymentExecutionOperations.properties in the below paths:
  wildfly-10.1.0.Final \ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
  wildfly-10.1.0.Final\ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud
  Add the below entry into the file
  <Operation_Name>=Operation_Name
  Example - startslagnet= startslagnet

8. Clear the server cache and restart the server.
   Order the custom services , all the task should be completed
   Verify the Comments and History in the Ordered Requisition (proper comments should be updated in the comments and history section)

   Below is the example of screen shot –



Ability to Save Filled-Out Service Order Form as Template

- User can now save service form as template which will create new requisition with Draft status and as template.
- User can copy and create new requisition with draft status or even submit as new requisition for both logged in user or for others.
- Template requisitions are not allowed to submit.

- Administration setting to enable or disable Template feature is mentioned below:
  - Administration -> Settings -> Enable Save As Template
- To enable and use template feature we should enable save as draft feature as well.
  - Administration -> Settings -> Enable Save As Draft
- Once the template requisitions is created we can manage from Service Catalog Orders page by filtering with ‘Template’.
- Template requisitions can perform below actions

Create Order	To copy template requisition as new draft requisition with customer and initiator as logged in user
Create Order for Others	To copy template requisition as new draft requisition with selected customer and logged in user as initiator.
Submit Order	To copy template requisition and submit with customer and initiator as logged in user

Submit Order for Others	Copy template requisition and submit with selected customer and logged in user as initiator
Add Attachment	To add an attachment
Remove Attachment	To remove an attachment
Cancel Order	To cancel a requisition

Jar Upgrades

- dom4j jar is upgraded to 2.1.1

  RequestCenter.war\WEB-INF\lib\dom4j-2.1.1.jar
  ServiceLink.war\WEB-INF\lib\lib\dom4j-2.1.1.jar

- Commons beanutils jar is upgraded to 1.9.4 version

  ServiceLink.war\WEB-INF\lib\commons-beanutils-1.9.4.jar
  RequestCenter.war\WEB-INF\lib\commons-beanutils-1.9.4.jar

- Jackson-databind and its dependent jar's are upgraded to 2.10.0 version

  RequestCenter.war\WEB-INF\lib\jackson-databind-2.10.0.jar
  RequestCenter.war\WEB-INF\lib\jackson-annotations-2.10.0.jar
  RequestCenter.war\WEB-INF\lib\jackson-core-2.10.0.jar

nsAPI's Introduced in Patch_v12

Localization API

- METHOD : GET
- URL: http://10.78.0.241:8080/RequestCenter/nsapi/localization/v1/productStrings?resourceId=109&startIndex=0&recordsPerPage=20

  QueryParams:
  name = { product String }
  resourceId = { ResourceId}
  startIndex = 1
  recordsPerPage = 20

- Response:


```
{
   "Result": {
    "totalRecords": 1,
   "recordsReturned": 1,
  "records": {
   "109": [
  {
    "id": 0,
    "outOfSync": 0,
    "languageName": "US English",
    "propertyId": 109,
    "value": "Text",
    "localeId": 1
     },
     {
      "id": 0,
```

```
"outOfSync": 0,
"languageName": "Chinese (Simplified)",
"propertyId": 109,
"value": "文本",
"localeId": 6
}
]
}
}
}
```


Save as Template APIs

- Save as template from cart page
  - URL: http://localhost:8080/RequestCenter/nsapi/transaction/saverequisitionasdraft?responseType=json&customerId=1&saveAsTemplate=true
  - Method: POST
  - Headers: Content-Type : application/json
  - Payload:

```
[
  {
    "requisitionId": "2122",
    "requisitionEntryId": "2135",
    "quantity": "1",
    "pricePerUnit": "$ 0.00"
  }
]
```

  - Response: <nsapi-response>Requisition saved as a draft successfully.</nsapi-response>
- Create Order or Create Order For Others
  - URL: /RequestCenter/nsapi/catalog/v1/requisition/{id}/copyAsDraft/customer/{customer_id}/initiator/{initiator_id}
  - Method: POST
  - Headers: Accept : application/json
  - Response:

```
{
  "HashMap": {
    "RequisitionId": "2103",
    "Message": "The Template Requisition has been successfully copied."
  }
}
```

- Submit Order or Submit Order For Others
  - URL: /RequestCenter/nsapi/catalog/v1/requisition/{id}/orderFromTemplate/customer/{customer_id}/initiator/{initiator_id}
  - Method: POST
  - Headers: Accept : application/json
  - Response:

```
{
  "RequisitionSubmit": {
    "id": 2126,
    "customer": "admin admin",
    "initiator": "admin admin",
    "startedDateRaw": 1574136178633,
    "startedDate": "11/19/2019 1:32 AM",
    "status": "Ordered"
```

}
}

Modified CloudCenter APIs to Support Workload Manager

- Cloud Center Api's that are modified with cloudProvider query param
  - Refresh single/multiple application profiles to Prime Service Catalog
    - /RequestCenter/nsapi/cloudCenter/applicationProfiles?connectionId=2&cloudProvider=<WM or CLIQR>
  - Get VM Details
    - /RequestCenter/nsapi/cloudCenter/vm/id/<vm_id>? cloudProvider=<WM or CLIQR>
  - Get VM Console Access
    - /RequestCenter/nsapi/cloudCenter/vm/<vm_id>/console? cloudProvider=<WM or CLIQR>
  - Get Windows VM login details
    - /RequestCenter/nsapi/cloudCenter/vm/windows/<vm_id>/logindetails/?connectionId=2&cloudProvider=<WM or CLIQR>

Introduced in 12.1_Patch_v11

Ability to Export List of Tasks from New Service Manager UI

"Export" button has been introduced in Prime Service Catalog 12.1 alongside other actions on the bottom right corner of the Service Manager page .The data can be exported in CSV and EXCEL format.

Browser Upgrades

- IE-11
- Firefox-67
- Chrome-75
- Safari-12

Certifications

- **CCS 5.0.1 AO Integration - Certified against CCS 5.1 AO**

CCS 5.0 AO is not supported for this patch. If you have already integrated with CCS 5.0.1 AO Integration you need to re-import the PSC AO workflow from
https://github.com/cisco/ActionOrchestratorContent/tree/master/workflow-examples/PSCAOIntegrationWorkflow__definition_workflow_016QL80Y3D2J62KI5hebU1SiGXzunNcCkQj

- **UCSD 6.7 Certification-Certified against UCSD 6.7 (Fenced), UCSD 6.5(MSP), UCSPM 2.5**

Database Upgrade

- **Microsoft SQLServer 2016(SP2-CU7) (KB4495256) - 13.0.5337.0 (X64))**
- **Oracle 19c Officially Weblogic 12C does not support Oracle 19C**

  Note: There is no official support for Oracle 19C for Cognos 10.2.2

Jar Upgrades

Security fixes are made and updated in the readme document along with the patch zip file.

jQuery libraries
JQuery version used by Prime Service Catalog is upgraded to version 3.4.1.

API to Get Tasks

This is an existing API enhanced by adding queryParams

- Method: GET
- URL: RequestCenter/nsapi/transaction/v2/tasks
- Response message :
  "Priority","RequisitionID","DueOn","ServiceTask","ServiceName","Initiator","CustomerOU","CustomerName","Performer","Queue","Status","Effort","RequisitionEntryID","ScheduledStart","TaskType"

  QueryParams:

- Action: EXPORT
- ViewID: {viewId}

Introduced in 12.1_Patch_v10

Poller for Action Orchestration

Note: **The flags ao.poller.cron and ao.poller.health.check.cron are mandatory to be added even if poller is not enabled.**

Some of the properties are needed to be added to the newscale.properties and support.properties to configure AO Poller.

Following two fields are mandatory to be added in newscale.properties for the server to come up.
ao.poller.cron=0 0/5 * * * ?
ao.poller.health.check.cron=0 10/15 * * * ?

Properties to be added for newscale.properties

#AO Poller

##

#Cron Expression wakes up poller every 10 minutes of an hour

ao.poller.cron=0 0/10 * * * ?

#Cron Expression wakes up health check for 10th min and 15 mins thereafter of the hour ex: 10,25,40,55 minutes

ao.poller.health.check.cron=0 10/15 * * * ?

#High Availability Health checks threshold, this should be greater than Poller cron time specified in minutes

ao.healthCheck.threshold=91

##

Properties to be added for support.properties

AO Poller Settings

#Flag to enable/disable the poller

ao.poller.enable=true

Decoupled UI Support

In V9 Patch Decouple UI was supported for Standalone Wildfly topology, in this patch, addition to Standalone Wildfly topology we are supporting for cluster topology.
Hosting Decoupled Service Catalogue and Service Manager websites on a server separate from the one where Prime Service Catalogue has been installed.

- Support for PSC running on clustered WildFly application server.
- Support for PSC running on standalone and clustered WebLogic application server.
- Support for hosting the decoupled Service catalogue and Service manager websites in a clustered setup.

- App Server
    - Wildfly Standalone
    - Wildfly Cluster Topology
    - Weblogic standalone
    - Weblogic Cluster
- LB Servers
    - IIS and Apache httpd
- Database Servers
    - SQL and Oracle
- UI Servers

- Apache Tomcat 9.0 and Nginx 1.15.8

- Browser Support
  - IE11
  - Chrome 74
  - Firefox 66
  - safari 12.x

Setting up Tomcat Cluster for UI Decouple

Pre-requisite: Wildlfy 10.1 2 VM Node or Weblogic two Node with PSC and LB (IIS or Apache httpd) should be up and running. All the nodes should be on the same domain.
The steps are as follows to setup tomcat nodes on two different nodes:

1. Install tomcat 9.0.14 version.
2. Go to Apache directory and open server.xml. The location of the file - C:\Program Files\Apache Software Foundation\Tomcat 9.0\conf
3. Search defaultHost and update with FQDN of the system and on the same line add jvmRoute="node1", below is the example
   <Engine name="Catalina" defaultHost="<FQDN>" jvmRoute="node1">
4. Search Host name and update the FQDN of the system, below is the example

```
<Host name="<FQDN>"  appBase="webapps"
        unpackWARs="true" autoDeploy="true">
```

Same settings need to be performed for other nodes.
Hit the below URL, you will see tomcat home page.
http://<Host_Name>:<PORT>/

Setting up LB for the above nodes

1. Download and Install Apache 2.4
   Reference: https://httpd.apache.org/docs/2.4/platform/windows.html
2. Open the httpd.conf file and add below entry into the end of the file.

```
ServerName <FQDN>
KeepAlive On
DocumentRoot "c:/Apache24/htdocs"
#ServerName VMDEVSLw2k8-ovf
<Directory "c:/Apache24/htdocs">
EnableSendfile off
</Directory>
Listen FQDN:80
AcceptFilter http none
AcceptFilter https none
   <VirtualHost *:80>
  ServerName <FQDN>
  ProxyRequests Off
  ProxyPreserveHost On
  <Location /balancer-manager>
   SetHandler balancer-manager
   Order allow,deny
   Allow from all
  </Location>
  Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
env=BALANCER_ROUTE_CHANGED
  <Proxy balancer://mycluster>
     BalancerMember ajp:// <FQDN>:8009 route=1
     BalancerMember ajp://abc.cisco.com:8009 route=2
     #BalancerMember ajp://IP_Address:8009 route=3
           #BalancerMember ajp://IP_Address:8009 route=4
     ProxySet stickysession=ROUTEID
  </Proxy>
  ProxyPass / balancer://mycluster/
  ProxyPassReverse / balancer://mycluster/
 </VirtualHost>
```

3. Restart httpd Apache server.
4. Hit the below URL and you will see the home page. Just to make sure HA, you can stop one of the above node and refresh, the tomcat home page will be displayed.
   http://<Host_Name>:<PORT>/

Enabling CROS Settings in WF Cluster Nodes.

Note: The below mentioned settings is only for Wildlfy.

1. Open the domain.XML from “wildfly-10.1.0. Final\domain\configuration”
2. Add highlighted entry to the file
   Note: There will be two profiles in the domain.xml
3. Profile name is default and ha for both profile entry should be made

```
<subsystem xmlns="urn: jboss: domain:undertow:3.1" instance-id="${jboss.web.instanceId}">
        <buffer-cache name        ="default"/>
        <server name="default-server">
          <ajp-listener name="ajp" socket-binding="ajp" scheme="https" max-ajp-packet-size="65536"/>
          <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>
          <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content"/>
            <filter-ref name="server-header"/>
            <filter-ref name="x-powered-by-header"/>
            <filter-ref name="Access-Control-Allow-Origin"/>
            <filter-ref name="Access-Control-Allow-Headers"/>
            <filter-ref name="Access-Control-Allow-Credentials"/>
          </host>
        </server>
        <servlet-container name="default">
          <jsp-config/>
          <websockets/>
        </servlet-container>
        <handlers>
          <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
        </handlers>
        <filters>
          <response-header name="server-header" header-name="Server" header-value="WildFly/10"/>
          <response-header name="x-powered-by-header" header-name="X-Powered-By" header-
value="Undertow/1"/>
          <response-header name="Access-Control-Allow-Origin" header-name="Access-Control-
Allow-Origin" header-value="http://abc.cisco.com"/>
          <response-header name="Access-Control-Allow-Headers" header-name="Access-Control-
Allow-Headers" header-value="accept, authorization, content-type, x-requested-with, Content-Type"/>
          <response-header name="Access-Control-Allow-Credentials" header-name="Access-Control-
Allow-Credentials" header-value="true"/>
        </filters>
    </subsystem>
```

4. Modify the Newscale Properties –

```
WildFly Newscale:
################################################################
#decoupled UI domain name
decoupled.cookie.domain=abc.cisco.com
################################################################
#decoupled UI authentication mode
decoupled.authentication=true
################################################################
#decoupled UI landing page url for servicecatalog module
decoupled.servicecatalog.url=http://abc.cisco.com/RequestCenter/website/CustomSC/application/index.h
tml
################################################################
#decoupled UI landing page url for servicemanager module
decoupled.servicemanager.url=http://abc.cisco.com/RequestCenter/website/CustomSM/application/servic
emanager.html?route=servicemanager#homepage
################################################################
decoupled.saml.logout.url=http://abc.cisco.com/Logout/ssoLogout.html
```

decoupled.servicecatalog.mobile.url=http://orac-estr-w2k12.cisco.com/RequestCenter/website/ServiceCatalogMobileWebsite/application/index.html

Weblogic Newscale:
##
#decoupled UI domain name
decoupled.cookie.domain=.abc.com
##
#decoupled UI authentication mode
decoupled.authentication=true
decoupled.domain.url=http:// abc.cisco.com
##
#decoupled UI landing page url for servicecatalog module
decoupled.servicecatalog.url=http:// abc.cisco.com /RequestCenter/website/ServiceCatalogWebsite/application/index.html
##
#decoupled UI landing page url for servicemanager module
decoupled.servicemanager.url=http:// abc.cisco.com /RequestCenter/website/ServiceManagerWebsite/application/servicemanager.html?route=servicemanager#homepage
##
decoupled.saml.logout.url=http://abc.cisco.com/Logout/ssoLogout.html

5. Server restart is always applicable, regardless of app server type.

Enabling Cors for IIS

Note: if the application server is using IIS then below entry should be made

1. Go to the VM where IIS is located, install the CORS module from the below mentioned link. https://www.iis.net/downloads/microsoft/iis-cors-module
2. Download X64 installer and install the CORS module.
3. Open the Web.config of IIS and add the below tags within the system.webserver

```
<cors enabled="true">
    <add origin="http://vm236.cisco.com" allowCredentials="true" >
        <allowMethods>
            <add method="PUT" />
                <add method="GET" />
            <add method="POST" />
            <add method="DELETE" />
                <add method="OPTIONS" />
        </allowMethods>
    </add>
</cors>
<httpProtocol>
    <customHeaders>
        <remove name="X-Powered-By" />
        <add name="Access-Control-Allow-Headers" value="x-requested-with, Content-Type, origin, Authorization, Accept, locale, access_token, utid, Range" />
        <add name="X-Powered-By" value="ASP.NET,Undertow/1" />
    </customHeaders>
</httpProtocol>
```

4. Restart IIS.

Setting up PSC UI on different WebServer

1. Go to VM where the tomcat is installed
2. Go to the deployment folder of webserver in this case its C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps
3. Create folder structure like below
    i. Create RequestCenter, Login and Logout folder
    ii. Inside the RequestCenter create sub folder called website

a. Copy ServiceCatalog, Service Manager, ServiceCatalogMobileWebsite and FormServer folder from the PSC Base Instance “wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website and place under the webapps/RequestCenter/website/ of tomcat VM

4. Copy the content of dist folder from wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website\LoginWebsite and place into the webapps/Login of the tomcat VM
5. copy the LoginConfig.Json form wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website and place into the webapps/RequestCenter/website
6. Copy the help and jsbundle folder from the wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war and paste into the webapps/RequestCenter of the tomcat vm
7. Copy the content of ssologout folder from the wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website\LoginWebsite\ssoLogout to the webapps/Logout folder of tomcat VM
8. Now modify the json file with below –

Host – PSC Base Instance -Mandatory for decoupled website

Cookie Domain – Domain Which base system belongs too. Ex if the VM is belongs cisco.com and then cookie domain is.cisco.com

serviceCatalogUrl – Provide the SC URL which is the IP or host name of tomcat Server hosted

serviceManagerUrl -- Provide the SC URL which is the IP or host name of tomcat Server hosted

The properties to configure Login page information can be passed here.

copyRightText – Copy right message to display on login Page

displayString – Product Name (Cisco Service Prime Catalog is the default)

marketingVersion – Product Version

Sample LoginConfig.json

```
{
  "host":"http://<HostName>:<Port>",
  "cookieDomain":".cisco.com",
  "ssoUrl":"",
  "loginModule":{

"serviceCatalogUrl":"http://<Host_Name_of_Tomcat>/RequestCenter/website/ServiceCatalogWebsite/application/index.html",
    "serviceManagerUrl":"http://<Host_Name_of_Tomcat>/RequestCenter/website/ServiceManagerWebsite/application/servicemanager.html?route=servicemanager#homepage"
  "serviceCatalogMobileUrl":"http://orac-estr-w2k12.cisco.com/RequestCenter/website/ServiceCatalogMobilewebsite/application/index.html"
  },
  "loginPageInfo" : {
    "copyRightText" : "",
    "displayString" : "",
    "marketingVersion":""
  }
}
```

9. Restart the Tomcat Server nodes on both VM and restart apache LB
10. Hit the Apache LB URL server like below

http://<Apache_LB>:<Apache_Port>/Login

The URL should navigate to PSC login page and user should be able to login and should be able to see Service Catalog and Service Manager.

Troubleshoot

1. IE 11 - If you are not able to access the PSC using LB of UI

JAVA Script - should be enabled for Internet and Trusted site

SCRIPT7002: XMLHttpRequest: Network Error 0x80070005, Access is denied.

Security area (Internet Options> Security) should be ENABLE

The following setting: Miscellaneous> Access data sources across domains.

2. SAML Logout not working properly? first time logged in successfully but on logout, on relogin failed with 500 internal error. (on Re login it is getting redirected to RC.war)

Change session time out from 30 to 20.

This is original setting in web.xml (both tomcat cluster nodes)

<session-config>
<session-timeout>30</session-timeout>
</session-config>
change to 20
<session-config>
<session-timeout>20</session-timeout>
</session-config>
Restart the Tomcat Nodes, Issue will be not seen after this setting.

3. On Session Expired, logged out on relogin blank page on the Store and seen error in console, on refresh of browser it will logged out and user will be placed on the login page. To come out of this issue, user should clear browser cache every time.
Request URL:
https://vmqa20_node1.cisco.com/RequestCenter/nsapi/v1/common/resourcestrings?utid=C0EEB917AB123082EC83DBB34CFF87CD
Request Method: PUT
Status Code: 401 Unauthorized

Solution: Update httpd.conf file with below entry after the </Directory> tag
AcceptFilter http none
AcceptFilter https none

4. LB System will get Hang when user Perform Search on SC

Solution: Update httpd.conf file with below entry after the </Directory> tag
AcceptFilter http none
AcceptFilter https none

Grid and Pagination

PSC is now supporting grid and pagination for Material UI Service Form.

Colormapping for Grid

Grid is supported through colormap where the properties of the grid (font, color) can be defined in servicedesigner module under the scripts tab. This should be associated with the active form events
Below is the sample script for grid colormap.

```
Grid_DDRReqStatus(form){

    var colorMap=[{
      "dictionaryName":"GridRequisitionStatus","colorMap":[{
      "fieldName":"StatusID",
      "colorList":[
        {"value":"1","rowColor":"#FF0000","cellColor":"#00FF00","textColor":"#0000FF"},
        {"value":"2","rowColor":"green","cellColor":"yellow","textColor":"black"},
      {"value":"3","rowColor":"blue","cellColor":"red","textColor":"green"},
      {"value":"-5","rowColor":"red","cellColor":"red","textColor":"black"}
      ]
    }
    ]
    }];

        ISF.isf_gridCellColor(form,colorMap);
      }
```

Method to get all the data in grid in json format.

```
/**
    *method to get grid data in json format
   @param form: object - service form object which contains all form specific information
   @param gridDictionaryName: string-ID of grid dictionary

   */
 isf_getGridData(form, gridDictionaryName)
```

Method to get grid cell value by cell index

```
/**
    *method to get grid cell value by cell index
   @param form: object - service form object which contains all form specific information
   @param gridDictionaryName: string-ID of grid dictionary
   @param rowIndex: int-row index(0 based) of grid
   @param colIndex: int-column index(0 based) of grid
   */
  isf_getGridCellValue(form, gridDictionaryName, rowIndex, colIndex)
```

Method to get grid cell value by fieldname

```
/**
 @param form: object - service form object which contains all form specific information
 @param gridDictionaryName: string-ID of grid dictionary
 @param rowIndex: int-row index(0 based) of grid
 @param colKey: string-column key of grid(field name)

 */
 isf_getGridCellValueByKey(form, gridDictionaryName, rowIndex, colKey)
```

Method to set grid cell value by cell index

```
/**
 @param form: object - service form object which contains all form specific information
 @param gridDictionaryName: string-ID of grid dictionary
 @param rowIndex: int-row index(0 based) of grid
 @param colIndex: int-column index(0 based) of grid
@param value: string-new value to be updated
 */
 isf_setGridCellValue(form, gridDictionaryName, rowIndex, colIndex, value)
```

Method to set grid cell value by fieldname

```
/**
    *method to get grid cell value by cell index
   @param form: object - service form object which contains all form specific information
   @param gridDictionaryName: string-ID of grid dictionary
   @param rowIndex: int-row index(0 based) of grid
   @param colKey: int-column name of grid(fieldName)
  @param value: string-new value to be updated
   */
  isf_setGridCellValueByKey(form, gridDictionaryName, rowIndex, colKey, value)
```

Method to set grid rowColor or cellcolour or celltextcolor

To set grid rowColor/cellcolour/celltextcolor based on the value match of cell for particular dictionary and field. The data for setting the color is provided in form of json containing dictionary name,fieldname, cell value to be matched and color value to set.

```
/**
  *
  @param form: object - service form object which contains all form specific information
  @param jsonColorMap: json-standard json containing color map for cell value
  */
 isf_gridCellColor(form, jsonColorMap)
```

Sample json to set color

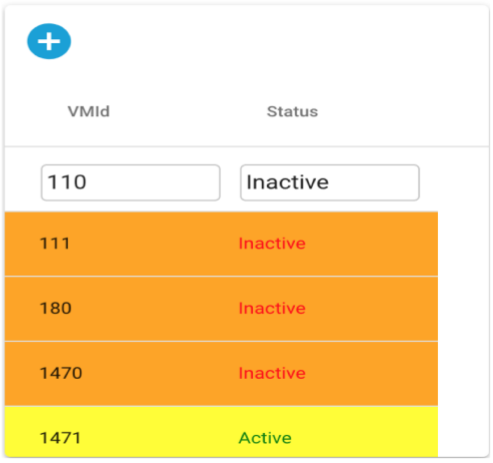
```
var colorMap = [{
```

```
"dictionaryName": "",
"clorMap": [{
  "fieldName": "Text",
  "clolorList": [
    { "value": "abc", "rowColor": "red", "cellColor": "green", "textColor": "blue" },
    { "value": "xyz", "rowColor": "green", "cellColor": "yellow", "textColor": "black" },
  ]
},
{
  "fieldName": "Phone",
  "colorList": [
    { "value": "1234", "rowColor": "red", "cellColor": "green", "textColor": "blue" },
    { "value": "5678", "rowColor": "green", "cellColor": "yellow", "textColor": "black" },
  ]
}
]
```

Steps to create a Grid Service Form with Color Mapping

1. Create a JavaScript function using Service Designer -> Scripts page

```
var colorMap=[{
        "dictionaryName":"Grid_UCSDVMStatus",
        "colorMap":[{
                "fieldName":"Status",
                "colorList":[
                  {"value":"Active","rowColor":"yellow","cellColor":"","textColor":"green"},
                  {"value":"Inactive","rowColor":"orange","cellColor":"","textColor":"red"}
                ]
    }]
}];
```



2. Add this JS function into AFC->Active Form Behavior->Triggering Event-> When the form is loaded
3. The function will load during the loading of Material Service Form.

Service Item Namespace Support in email Template

Service item namespace(freemarker based) is supported for email template. The namespace support is for the service item in the context of the service e.g virtual machine for a provision vm service. Service item namespace is supported in the following scenarios:-

Plan tab tasks

1. Service item task and the Service Link task following it
2. UCSD sync task
3. Cloud Center sync task

Example for a UCSD VM would be:-

[#if serviceitem.SiUCSDVM.Name??] Name: [=serviceitem.SiUCSDVM.Name][/#if]
[#if serviceitem.SiUCSDVM.RAM??] RAM: [=serviceitem.SiUCSDVM.RAM][/#if]
[#if serviceitem.SiUCSDVM.HostName??] HostName: [=serviceitem.SiUCSDVM.HostName][/#if]

This feature needs to be enabled by setting the below property in newscale.properties:-email.use.serviceitem.namespace=true
[#if serviceitem.SiUCSDVM.Name??] Name: [=serviceitem.SiUCSDVM.Name][/#if]
[#if serviceitem.SiUCSDVM.RAM??] RAM: [=serviceitem.SiUCSDVM.RAM][/#if]
[#if serviceitem.SiUCSDVM.HostName??] HostName: [=serviceitem.SiUCSDVM.HostName][/#if]

Subscription records of the service item can be referred to as
[#if serviceitem.<serviceItemTypeName>.header.<subscriptionAttributeName>??] Name:
[=serviceitem.<serviceItemTypeName>.header.<subscriptionAttributeName>][/#if].
An example for a UCSD VM would be:-
[#if serviceitem.SiUCSDVM.header.RequisitionID??]ServiceItem_ReqID: [=serviceitem.SiUCSDVM.header.Requisi- tionID][/#if]
[#if serviceitem.SiUCSDVM.header.CustomerID??]ServiceItem_CustomerID: [=serviceitem.SiUCSDVM.header.CustomerID][/#if]
[#if serviceitem.SiUCSDVM.header.AssignedDate??]ServiceItem_AssignedDate:
[=serviceitem.SiUCSDVM.header.AssignedDate?date][/#if]
[#if serviceitem.SiUCSDVM.header.SubmittedDate??]ServiceItem_SubmittedDate:
[=serviceitem.SiUCSDVM.header.SubmittedDate?date][/#if]
[#if serviceitem.SiUCSDVM.header.OrganizationUnitID??]ServiceItem_OUID:
[=serviceitem.SiUCSDVM.header.OrganizationUnitID][/#if]
[#if serviceitem.SiUCSDVM.header.AccountID??]ServiceItem_AccID: [=serviceitem.SiUCSDVM.header.AccountID][/#if] [#if serviceitem.SiUCSDVM.header.AgreementID??]ServiceItem_AgreementID: [=serviceitem.SiUCSDVM.header.AgreementID][/#if

Service item can have a parent service item e.g virtual machine being part of the container. The container then could be referred to as: -
[#if serviceitem.SiUCSDVM.parent.Name??]Application Name: [=serviceitem.SiUCSDVM.parent.Name][/#if]
[#if serviceitem.SiUCSDVM.parent.Status??] Status: [=serviceitem.SiUCSDVM.parent.Status][/#if]
[#if serviceitem.SiUCSDVM.parent.ContainerID??]Container ID: [=serviceitem.SiUCSDVM.parent.ContainerID][/#if]
[#if serviceitem.SiUCSDVM.parent.DisplayName??]DisplayName: [=serviceitem.SiUCSDVM.parent.DisplayName][/#if]

If the service item has child service items e.g container having multiple virtual machines. These can be referred to as: -
[#if serviceitem.SiUCSDContainer.VirtualMachines?has_content]
Details of VMs: -
[#assign indx=1]
[#list serviceitem.SiUCSDContainer.VirtualMachines as vm]
VM [=indx]: -
[#if vm.Name??] Name: [=vm.Name][/#if]
[#if vm.UCSDName??] UCSDName: [=vm.UCSDName][/#if]
[#if vm.HostName??] HostName: [=vm.HostName][/#if]
[#if vm.IPAddress??] IPAddress: [=vm.IPAddress][/#if]
[#if vm.MACAddress??] MACAddress: [=vm.MACAddress][/#if]
[#if vm.VMId??] VMId: [=vm.VMId][/#if]
[#if vm.Description??] Description: [=vm.Description][/#if]
[#assign indx=indx+1]
[/#list]
[/#if]

For more information refer to freemarker documentation.
Namespace email templates are supported for the following task events:

i. Notify when activity completes
ii. Notify when activity is cancelled
iii. Notify when external task fails

Host RequestCenter in PSC without ServiceLink

JMS standalone configuration with PSC

In patch v10, there is an option to host RequestCenter without ServiceLink.
Following steps would need to be done

Pre-requisite: User should have the Active MQ Artemis Server installed and able to access Management console from the client machines**.**

Steps to install and configure Artemis server

1. Download Artemis software from below location and unzip and place it in the C drive.https://activemq.apache.org/components/artemis/download/

Get more information on the Installation from the below link.
https://activemq.apache.org/components/artemis/documentation/1.0.0/running-server.html

2. After installation, create broker folder.
3. Run the below command

Example: Navigate to below location and execute the command.
C:\apache-artemis-2.7.0-bin\apache-artemis-2.7.0\bin

1. activemq create mybroker
Creates a folder mybroker unde the bin folder, fill in the username and password when prompted.
Note: User needs to change some configuration on set of files and take below file backup before modifying – Ex for location - apache-artemis-2.7.0\bin\mybroker\etc
   - broker.xml
   - bootstrap.xml
   - jolokia-aceess.xml
   - login.config

2. Find bootstrap.xml inside the mybroker/etc and open the bootstrap.xml and search for **<web bind="http://localhost:8161" path="web">** and replace the localhost to IP of VM where the ARTEMIS server is installed.

3. Open broker.xml in the same location and search for following and replace 0.0.0.0 with Host Name of Active MQ ARTEMIS Server<name>0.0.0.0</name>
<acceptor name="artemis">tcp://0.0.0.0:61616?tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=CORE,AMQP,STOMP,HORNETQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpLowCredits=300</acceptor>
<!-- AMQP Acceptor. Listens on default AMQP port for AMQP traffic.-->
<acceptor name="amqp">tcp://0.0.0.0:5672?tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=AMQP;useEpoll=true;amqpCredits=1000;amqpLowCredits=300</acceptor>
<!-- STOMP Acceptor. -->
<acceptor name="stomp">tcp://0.0.0.0:61613?tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=STOMP;useEpoll=true</acceptor>
<!-- HornetQ Compatibility Acceptor. Enables HornetQ Core and STOMP for legacy HornetQ clients. -->
<acceptor name="hornetq">tcp://0.0.0.0:5445?anycastPrefix=jms.queue.;multicastPrefix=jms.topic.;protocols=HORNETQ,STOMP;useEpoll=true</acceptor>
<!-- MQTT Acceptor -->
<acceptor name="mqtt">tcp://0.0.0.0:1883?tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=MQTT;useEpoll=true</acceptor>
4. Search for <auto-create-jms-topics>true</auto-create-jms-topics> in the broker.xml file
Add below line and save the file and close it.
<auto-delete-queues>false</auto-delete-queues>
5. Access jolokia-access.xml in the same location and search for <allow-origin>*://localhost*</allow-origin> and replace with IP of VM where ARTEMIS Server is installed.
6. Open login.config file in the same location and search for org.apache.activemq.artemis.spi.core.security.jaas.PropertiesLoginModule sufficient and replace word sufficient with required. Search org.apache.activemq.artemis.spi.core.security.jaas.GuestLoginModule sufficient on the same file and replace word sufficient with required.

Access the Active MQ Management console – http://<VM_IP>:8161/console

Login with the credentials, same password should be used which was used while creating broker.

Following is the syntax to run the jmsrunner script. The script is used for doing the configuration changes for standaloneJMS to work with Wildfly.

For Linux

Usage for Microsoft SQLServer:

jmsrunner.sh SqlServer {CPSCServer} {CPSCPort} {CPSCDatabaseName} {CPSCUser} {CPSCPassword} {ActiveMQServer} {ActiveMQPort} {ActiveMQUser} {ActiveMQPassword} {WildflyConfigPath} {RequestCenterWarPath} {QueueSuffix} {isWildfly18}

Usage for Oracle:

jmsrunner.sh Oracle {CPSCServer} {CPSCPort} {CPSCSID} {CPSCUser}
{CPSCPassword} {ActiveMQServer} {ActiveMQPort} {ActiveMQUser}
{ActiveMQPassword} {WildflyConfigPath} {RequestCenterWarPath} {QueueSuffix} {isWildfly18}

QueueSuffix is an optional parameter to differentiate the queues used for a particular PSC instance with respect to other PSC instances. This parameter adds the suffix to the queue's names.

./jmsrunner.sh Oracle localhost 1521 XE RCDB RC 192.168.174.1 61616 admin admin "/Users/user/_svn_repo/thirdparty/org/wildfly/wildfly-dist/10.1.0.Final/wildfly-10.1.0.Final_oracle/CPSC-12-1-0/configuration/standalone-full.xml" "/Users/user/_svn_repo/thirdparty/org/wildfly/wildfly-dist/10.1.0.Final/wildfly-10.1.0.Final_oracle/CPSC-12-1-0/deployments/RequestCenter.war"

For Windows

Usage for Microsoft SQLServer:

jmsrunner.cmd SqlServer <CPSCServer> <CPSCPort> <CPSCDatabaseName>
<CPSCUser> <CPSCPassword> <ActiveMQServer> <ActiveMQPort> <ActiveMQUser>
<ActiveMQPassword> <WildflyConfigPath> <RequestCenterWarPath> <QueueSuffix> <isWildfly18>

Usage for Oracle

jmsrunner.cmd Oracle <CPSCServer> <CPSCPort> <CPSCSID> <CPSCUser>
<CPSCPassword> <ActiveMQServer> <ActiveMQPort> <ActiveMQUser>
<ActiveMQPassword> <WildflyConfigPath> <RequestCenterWarPath> <QueueSuffix> <isWildfly18>

jmsrunner.cmd Oracle localhost 1521 XE RCDB RC 192.168.174.1 61616 admin admin Z:\user_svn_repo\thirdparty\org\wildfly\wildfly-dist\10.1.0.Final\wildfly-10.1.0.Final_oracle\CPSC-12-1-0\configuration\standalone-full.xml Z:\user_svn_repo\thirdparty\org\wildfly\wildfly-dist\10.1.0.Final\wildfly-10.1.0.Final_oracle\CPSC-12-1-0\deployments\RequestCenter.war

Notes

- Kek file need to be updated with appropriate values corresponding to the current database in the patch path (psc-12.1-patch-main-614/database/classes/config/kek_new.txt, psc-12.1-patch-main-614/database/classes/config/kek_old.txt)
- RequestCenter.war path is required for modifying the beeeCamelContext.xml and rcjms.properties.
- If the queues are already created with suffix and you want to remove the suffix and want to go with default, then you should pass the noSuffix argument at the end of the command.
- If you are configuring for Wildfly18 app server without servicelink then we have upgraded the same by adding add one more additional argument "isWildfly18" for which the value should be true. This parameter is optional for Wildfly10 configuration.

Setting up Active MQ Artemis server to Requestcenter

Prerequisite: Active – MQ Artemis Server should be up and running and PSC 12.1 latest patch should be available

Steps for Standalone:

1. Stop the PSC servers and run the db installer and unzip the latest war files on the RequestCenter.
2. Run the JMSruneer script to configure the Active MQ ARTMEIS Server to PSC. Find below example

Example- jmsrunner.cmd SQLSERVER 10.78.0.166 1433 VM184_RCDB_121GA CPSCUser RC 10.78.0.249 61616 admin admin C:\CiscoPrimeServiceCatalog_121GAPatchV7\wildfly-10.1.0.Final\ServiceCatalogServer\configuration\standalone-full.xml C:\CiscoPrimeServiceCatalog_121GAPatchV7\wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war Winqueue

3. From above script – standalone-full.xml, rcjms and beecamelcontext file will get modified (user should get the successful message)
4. Go to CiscoPrimeServiceCatalog 12.1/bin folder and run the startServiceCatalog.cmd.
5. Once server is up and running, access the PSC and go to Service Link, it should be red (that means service link is stopped)
6. Order any service, user should be able to place the order without any issues.

Steps for Cluster:

1. Run the Db Installer and unzip the latest war files on the RequestCenter.
2. Run the JMSruneer script to configure the Active MQ ARTMEIS Server to PSC. Find below example (Note: For Cluster user should extract folder to modify the files, on running pre-customization script RC.war will be unzipped and folder will be placed in the tmp directory, same location should be use as RequestCenter.war file

Example- jmsrunner.cmd SQLSERVER 10.78.0.166 1433 VM184_RCDB_121GA CPSCUser RC 10.78.0.249 61616 admin admin C:\CiscoPrimeServiceCatalog_121GAPatchV7\wildfly-10.1.0.Final\domain\configuration\domain.xml C:\CiscoPrimeServiceCatalog_121GAPatchV7\tmp\RequestCenter.war

3. From above script – Domain.xml, rcjms and beecamelcontext file will get modified (user should get the successful message)

4. Go to CiscoPrimeServiceCatalog 12.1/bin folder and run the post-customization script to apply the patch till it completes.
5. If user has more nodes, then on each node user should run the apply-customization script.
6. Once server is up and running, access the PSC and go to Service Link, it should be red (that means service link is stopped)
7. Order any service, user should be able to place the order without any issues.

Setting up RC without SL in the Weblogic

Follow the patch steps for Weblogic from the WebLogic apply patch section.

1. Deploy only RequestCenter.
2. Do not deploy the Service Link Application.
3. The Managed Server which is pointing to JMS Server should be up and running. (Mandatory)
4. RC deployed.
5. Login to the system.
6. Check the Service Link status it should be red.
7. Order the services.
8. User should place the order.

CCS 5.0.1 AO Integration

The AO_PSC_Integration_Workflow is used to integrate AO with PSC. This workflow is triggered based on an AMQP event from PSC. Below are the steps to configure this workflow:

Steps to configure the AO PSC Integration workflow

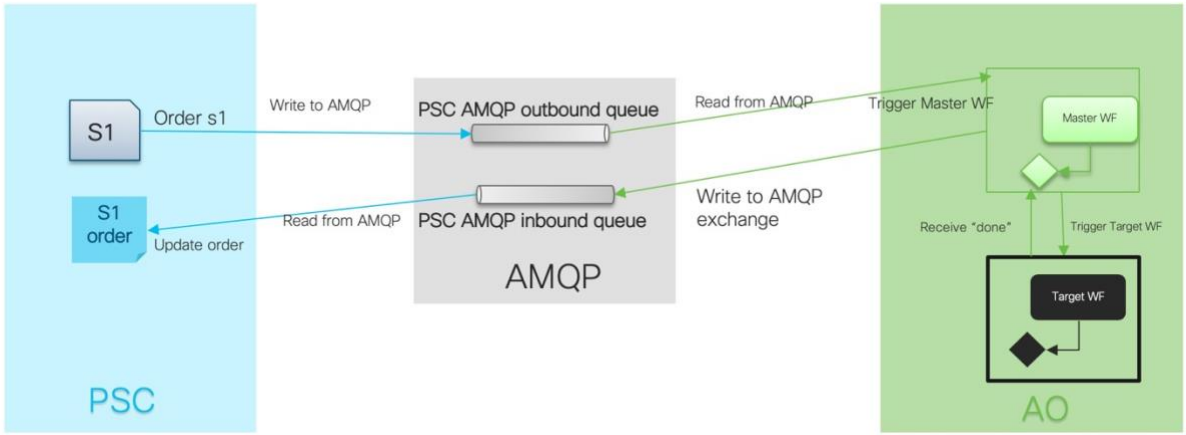
1. Import workflow from https://github.com/cisco/ActionOrchestratorContent/tree/master/workflow-examples/PSCAOIntegrationWorkflow__definition_workflow_016QL80Y3D2J62KI5hebU1SiGXzunNcCkQj
2. Create an AMQP target in AO with proper credentials
3. Create an AMQP Event in AO and use the name of the PSC outbound queue for queue name
4. Add a trigger in the AO_PSC_Integration_Workflow with the event created in step 3
5. Modify the “Get input variables” activity’s source query json
    a. Trigger>Name_of_trigger_created_in_step4>Output>MessageBody
6. Modify the value of the “AOToPSCExchange” variable if necessary
7. Modify the value of the “AmqpRoutingKey” variable if necessary, with the corresponding routing key for the exchange set in step 6
8. For all of the “Publish AMQP Message” activities (4 activities) select the target that was created in step 2
9. Create a CloudCenter Endpoint target to the env where you import the workflow
10. Add the target created in step 9 to all the “Generic CCS API Request” (3 activities) activities instead of the Suite Internal target

Note：Any changes to the output variables configured in the imported workflow will be detected and the updated value will be sent to PSC through AMQP nsxml message in the below format. The output variable should not contain any XML specific characters, else the nsxml will be invalid.

<message channel-id="D06AE3C8-CBEB-4F03-8D4E-6EACF074BA92"><add-comments><comment>Workflow completed successfully
output variables: #OUTPUT_VARIABLE_VALUE_GOES_HERE# </comment></add-comments><take-action action="done"></take-action></message>

Troubleshooting

1. When workflows are not triggered after ordering service from PSC:
    a. Check if the outbound queue has consumers in AMQP
        i. If there are no consumer, check if the trigger AO_PSC_Integration_Workflow is in a started-polling state, if not, update the workflow and wait for it to refresh

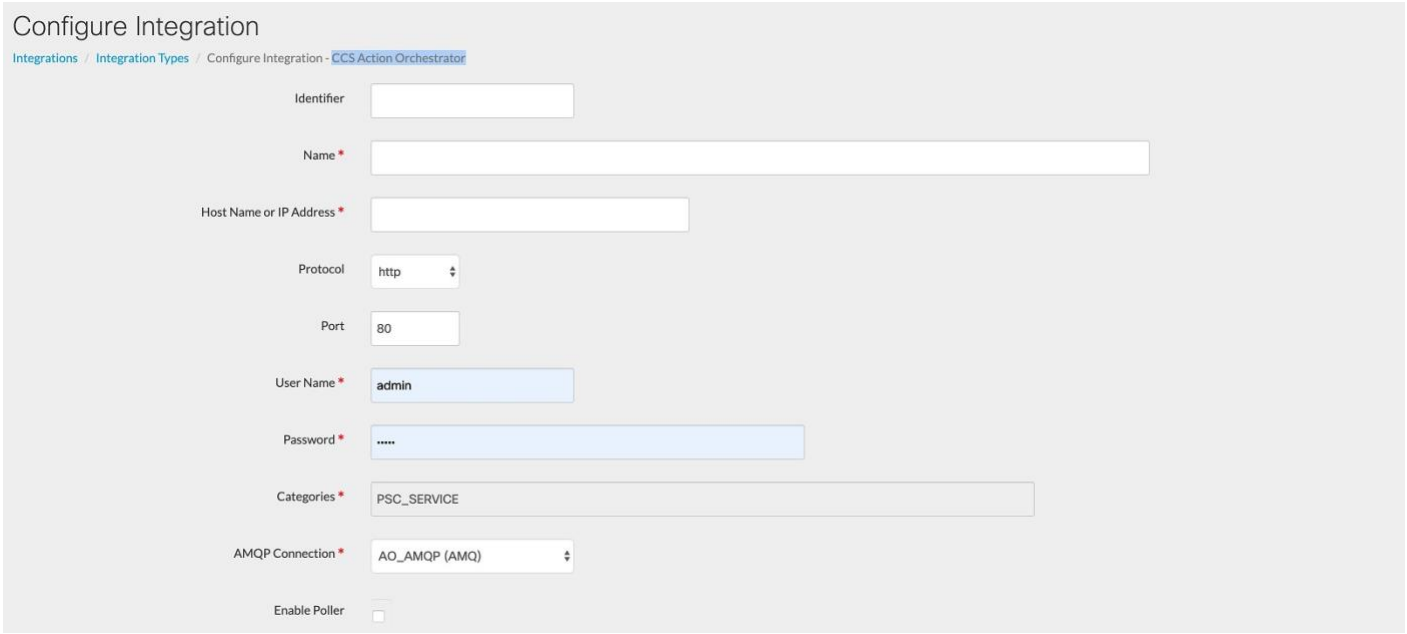


Adding AO Connection into PSC

Pre-requisite: AMQP Connection should be created and Inbound queue should be created in the AMQP system. CCS 5.0.1 (Cloud CenterSuite) should be up and running and AO should be part of the CCS 5.0.1 Setup. Master Workflow should be imported to the AO(Refer above section). PSC should be up and running with latest Patch v10.
In the AO PSC_SERVICE category should be created, and whichever workflow are relevant for PSC application should be part of this category.

Steps to add AO Connection in the PSC

1. Go to Integration Module in PSC and click New Integration and Select the “CCS Action Orchestrator” Integration type
   Users : Integration Administrator User or Admin user can be able to create a connection



- Identifier – Unique Identifier
- Name – Name of the Integration
- Host Name or IP Address – IP where the CC501 Suite is hosted
- Protocol – http/https
- Port – Port of CC501 Suite
- User Name – CC501 User should contains the tenant admin and suite admin groups
- Password – Password of the system
- Categories - This is PSC_SERVICE (this is read only)

- AMQP Connection – relevant connection should be selected
- Enable Poller- This is poller to sync based on poller setting in Newscale and support properties (if user wants to enable below properties should be part of Newscale and support.

Properties to be added for newscale.properties

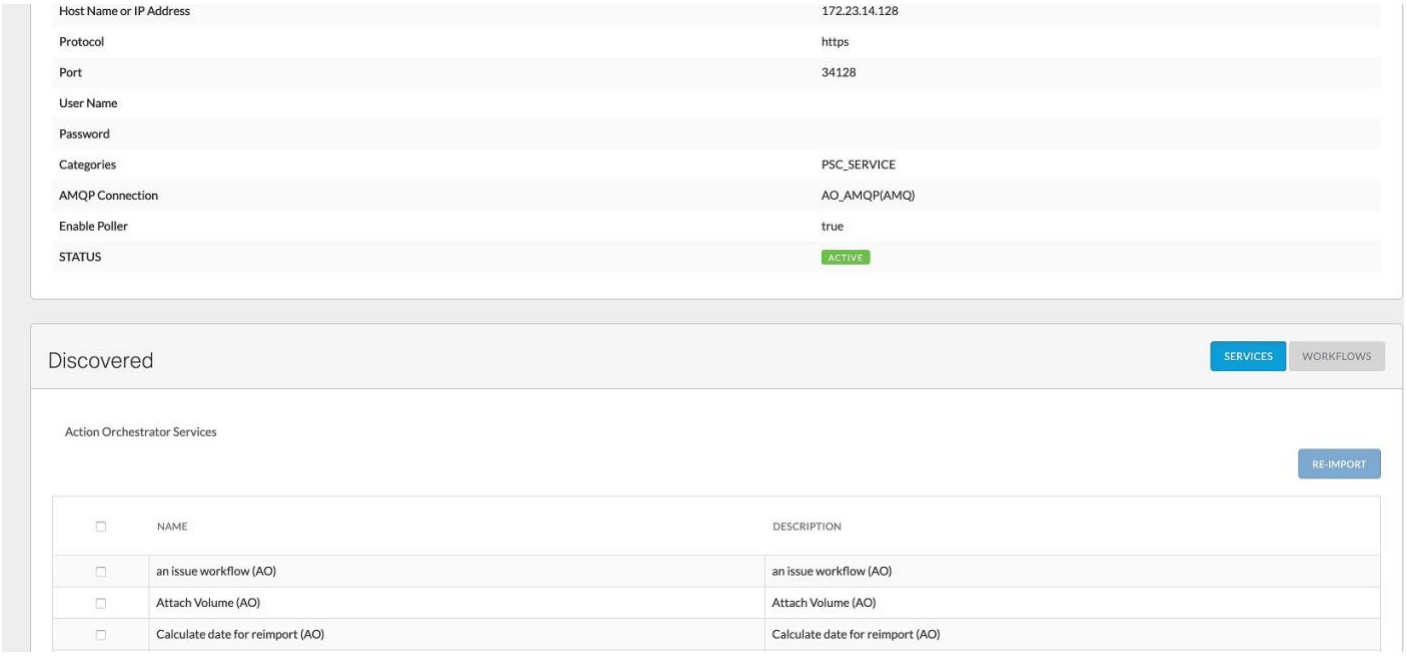
```
#AO Poller
##############################################
#Cron Expression wakes up poller every 10 minutes of an hour
ao.poller.cron=0 0/5 * * * ?
#Cron Expression wakes up health check for 10th min and 15 mins thereafter of the hour ex: 10,25,40,55 minutes
ao.poller.health.check.cron=0 10/15 * * * ?
#High Availability Health checks threshold, this should be greater than Poller cron time specified in minutes
ao.healthCheck.threshold=91
##############################################
```

Properties to be added for support.properties

```
###### AO Poller Settings ###################
#Flag to enable/disable the poller
ao.poller.enable=true
```

2. Create integration and connection will be saved successfully.
3. Click Import workflows (it will take some time to import the workflow , depending on no of workflows)
4. User will prompt with number of workflows imported to PSC
5. Workflows will be converted as services in PSC and same will be available in SD under the CCS AO Services (Identifiers name) and same dictionary group and AFC will be created for AO services
6. The Manage integration – workflow and service will list as below screen shot
7. Tenant Name given in AO should be configured in the AO_getToken.ftl file, example tenantName": "cisco"
   This file will be there in both RequestCenter and ISEE deployments in below location.

   WEB-INF/classes/config/cloud/AO_getToken.ftl

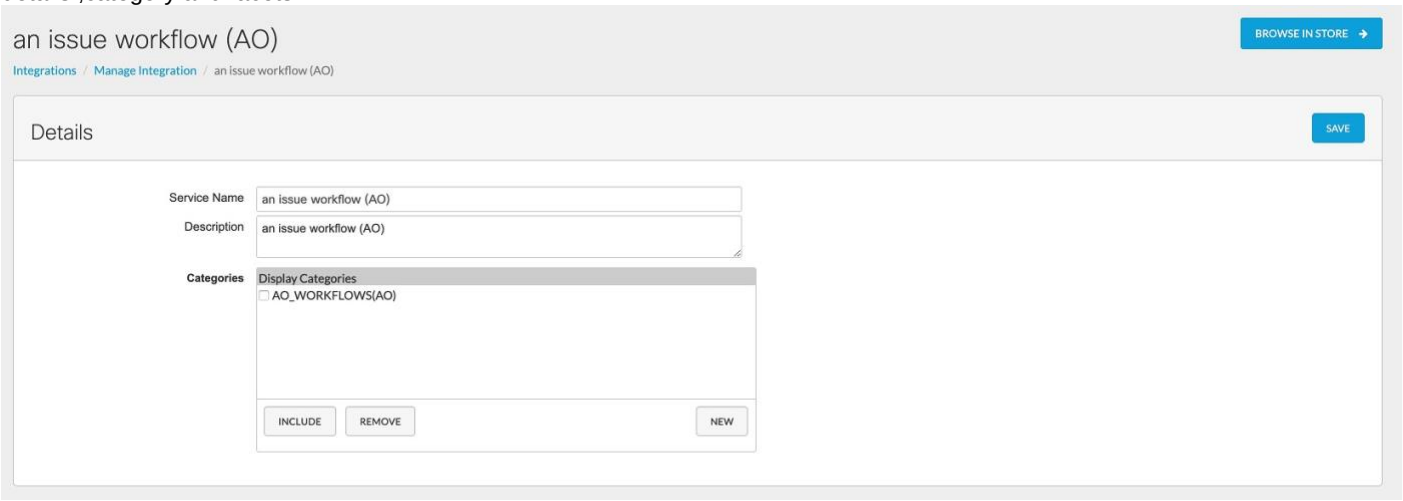


User should be able to order these services from the PSC and it will get processed through AMQP task and results will be updated back to PSC in the comments history.

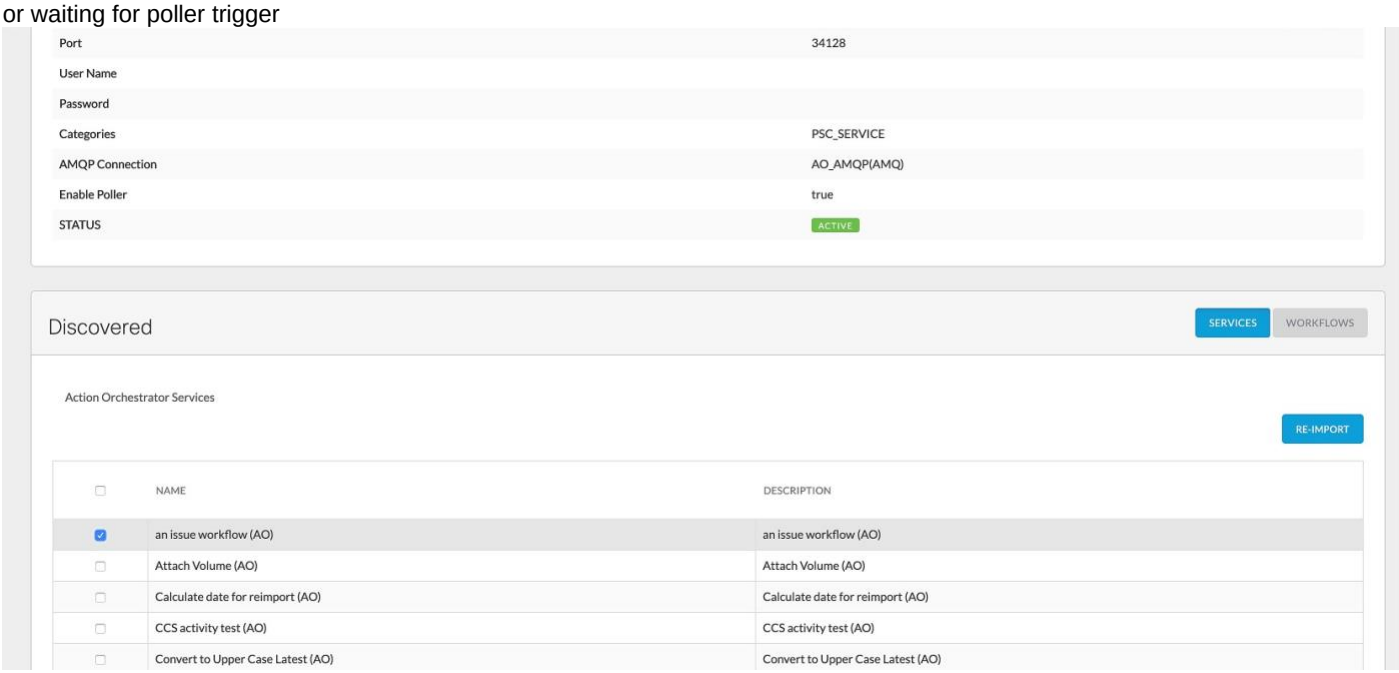
Once order placed in PSC, the message will get processed through the AMQP Queue and Workflow will get executed in the AO, and processed result will be sent back to AMQP queue, the PSC will read the response from the AMQP queue and same thing will get updated for respective requisition in the comments and History Section.

More Features from the AO Manage Integration –

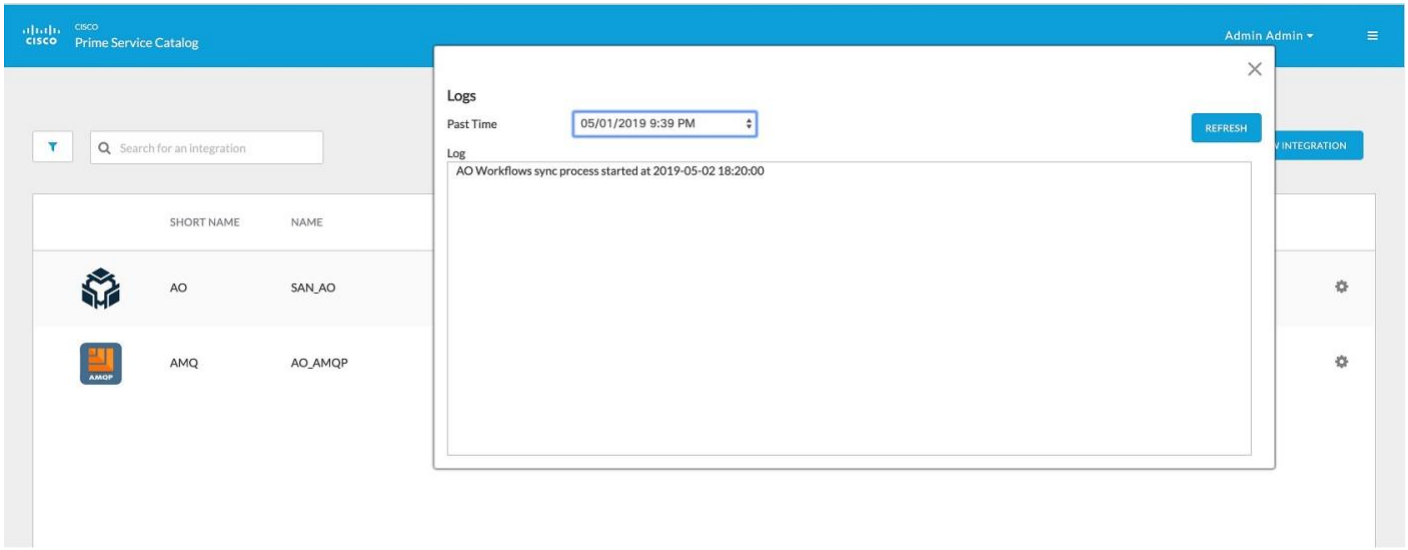
Mini Service Designer – This feature is useful to design the imported service in minimum SD such as adding image ,presentation details ,category and facets



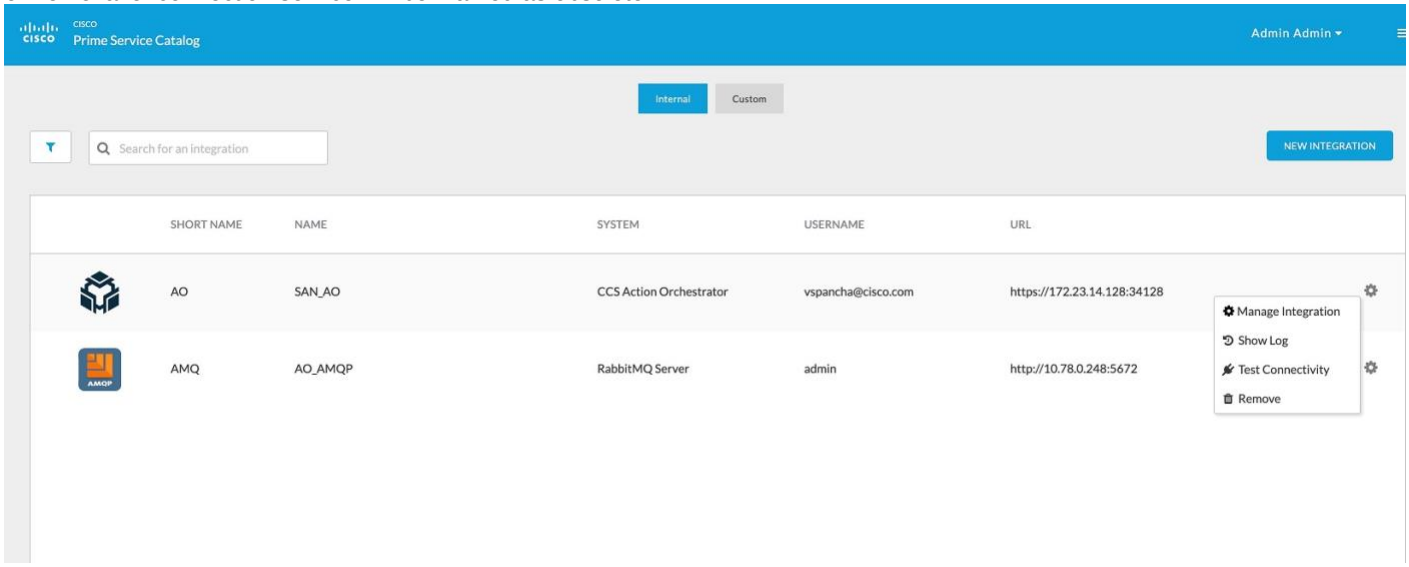
Reimport – if User modify any Workflow in the AO, that alone can be re imported using this feature , instead of importing manually or waiting for poller trigger



Show Log – It will be logged recent three logs of the Import action

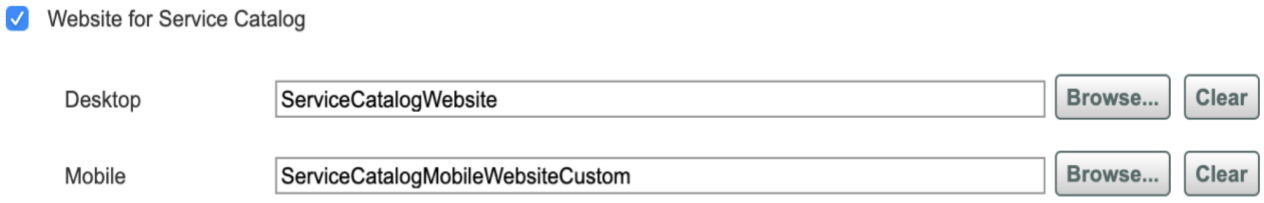


Test Connection – test the Connection of the Integration
Remove – Remove the connection from the PSC , it will remove the Service, AFC and Dictionary group , if any service is consumed on removal of connection service will be marked as obsolete



Responsive UI for Service Catalogue

PSC is now supporting responsive UI for mobile. After Login the user will be directed to ServiceCatalogMobileWebsite by default,
Custom Website Setting



From <http://rch-plstech-001.cisco.com/prrq-docrev/Review/ShowMarkup.aspx?documentid=286716&paragraphid=P019;Y000498;X000272.25;W0;H0&vid=_vidJBbqvdpuuV>

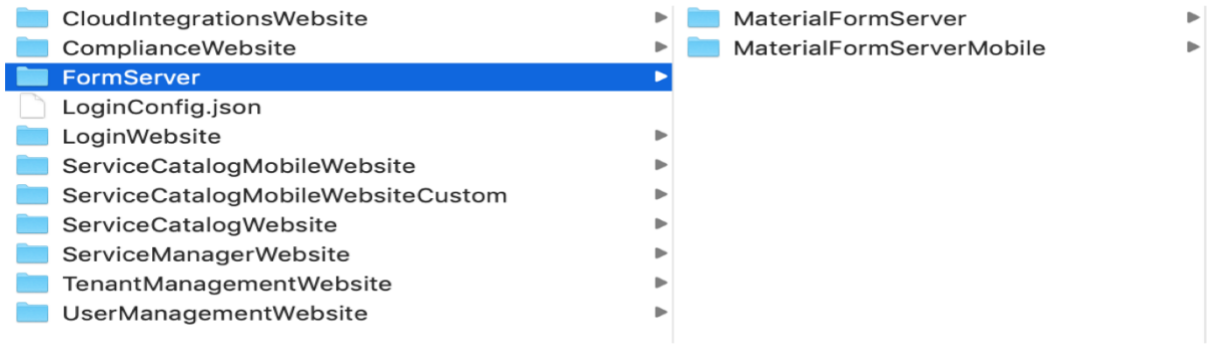
Responsive UI Customization

1) ServiceCatalogMobileWebsite is a new website which is specifically designed for mobile devices with a new page layout.
2) This mobile website can be fully customized from :-

   - Website Base URL: /RequestCenter/website/ServiceCatalogMobileWebsite/application
   - CSS File URL: /RequestCenter/website/ServiceCatalogMobileWebsite/common/css/ngc-bootstrap-mobile.css

3) This mobile website is having its own form server to order the services, such as:-

   - Form Server URL: /RequestCenter/website/FormServer/MaterialFormServerMobile



ResponsiveUI for mobile supports material service.
Find below the details of the resolution form for the different devices tested.

	Android	**IOS**
Viewport	360*640	375*667
Resolution	1080*1920	750*1334

The different profile shown are:

- Displaying information and preferences tabs.
- No service items page is shown in ResponsiveUI.
- Order, approvals and categories in the hamburger menu are displayed.

The authorization page in patch v10 has an enhancement where the approval tab provides more details about the customer, Orderby and Performer information when clicked.

Person Profile API

API to Get Current user information

- Method - GET
- Request URL: http://localhost:8080/RequestCenter/nsapi/directory/people/currentuser
- Response -

```
{
  "person":{
    "name":"admin admin",
    "firstName":"admin",
    "lastName":"admin",
```

```
"personId":1,
"recordStateId":1,
"email":"internal@newscale.com",
"timeZoneId":256,
"timeZoneName":"America/Tijuana",
"timeZoneDisplayName":"<s ID=\"13576\"/>",
"homeOrganizationalUnitId":1,
"homeOrganizationalUnitName":"Site Administration",
"login":"admin",
"isLdapEnabled":true,
"isLdapEnabledProperty":true,
"localeId":1,
"languageCode":"USEN",
"languageName":"US English",
"statusId":1,
"status":"Active",
"supervisorId":0,
"supervisorName":"",
"preferences":{
   "preferenceId":1,
   "shortDateSeperator":"/",
   "shortDateFormat":"MM/dd/yyyy",
   "longDateFormat":"MMMM dd, yyyy",
   "loginModuleId":21,
   "loginModule":"Service Catalog",
   "serviceManagerView":"My Work",
   "serviceManagerStatus":"All Ongoing",
   "timeFormat":"h:mm aa",
   "viewAuthorizationsPortlet":"true",
   "viewMyServiceItemsPortlet":"true",
   "nextGenarationCatalog":"true",
   "authorizationDelegateId":0
},
"teamCount":0,
"taskCount":0,
"contacts":[
   {
      "contactId":1,
      "contactTypeId":1,
      "contactType":"Email",
      "value":"internal@newscale.com"
   }
],
"addresses":[
   {
      "addressId":1,
      "addressTypeId":1,
      "addressType":"Company Address",
      "state":"",
      "zip":"",
      "city":"",
      "country":"",
      "street1":"",
      "street2":"",
      "building":"",
      "buildingLevel":"",
      "cubicle":"",
      "office":""
   },
    . . . .

],
```

```
    "includedGroups":[

    ],
    "includedOrganizationalUnits":[
      {
        "id":1,
        "name":"Site Administration"
      },


       .....


      {
        "id":387,
        "name":"[SQL Server] SG - Tenant-Information Reserved AFC"
      }
    ],
    "personURL":"<a
href='/RequestCenter/organizationdesigner/scnavigate.do?displayRecordId=1&id=1&query=search&resetBN=true&formAc
tion=display&displayRec=Y&forwardPage=people&forwardTo=showGeneralSuccess&mdicontentPortlet=portlet.sc.person.
general&mdicomponentsPortlet=portlet.sc.person.mdi&selectMDI=sc.person.general&isCreate=Y&sFlId=Y&layout=popup
_p' onclick=\"return GB_showFullScreen('Person', this.href)\">admin admin</a>",
    "personURLOnly":"/RequestCenter/organizationdesigner/scnavigate.do?displayRecordId=1&id=1&query=search&reset
BN=true&formAction=display&displayRec=Y&forwardPage=people&forwardTo=showGeneralSuccess&mdicontentPortlet=
portlet.sc.person.general&mdicomponentsPortlet=portlet.sc.person.mdi&selectMDI=sc.person.general&isCreate=Y&sFlId=
Y",
    "includedAccounts":{

    },
    "permissions":{
      "showTeams":true,
      "accessProfile":true,
      "accessMyStuff":true,
      "accessIntegration":true
    }
  }
}
```


API to get last login information

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/lastlogininfo?utid=0E500219AC7B988801F3D3CB813BCAAE&r
  esponseType=json&_=1586355716730
- Response -


```
{

  "List":[

    {

      "lastLoginStatus":"0",
      "lastLoginHost":"127.0.0.1",
      "lastLoginDate":"Thu Apr 16 15:37:24 IST 2020"
    }
  ]
}
```

API to list the available time zone

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/people/timezones?responseType=json&_=1586355716731
- Response –


```
{

  "List":[

    {

      "timeZoneID":201,
      "displayName":"(GMT) Greenwich Mean Time",
      "name":"Etc/Greenwich",
      "offset":"GMT+0:00"
    },
    {

      "timeZoneID":202,
      "displayName":"(GMT) Casablanca, Monrovia",
      "name":"Africa/Casablanca",
      "offset":"GMT+0:00"
    },

          . . .

  {

      "timeZoneID":285,
      "displayName":"(GMT) Western European Time: Dublin, Edinburgh, Lisbon, London",
      "name":"Europe/Lisbon",
      "offset":"GMT+0:00"
    }
  ]
}
```


API to list the locales

- Method - GET
- Request URL: http://localhost:8080/RequestCenter/nsapi/directory/people/locales?responseType=json&_=1586355716732
- Response -
-


```
{

  "List":[

    {

      "localeID":1,
      "localeCode":"USEN",
      "name":"US English",
      "active":0
    }
  ]
}
```

API for the time zone with calendar working hours

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/calendar/timeschedule?utid=0E500219AC7B988801F3D3CB813BCAAE&responseType=json&_=1586355716733
- Response -


```
{

  "personCalendar":{

    "timeZoneName":"(GMT-08:00) Pacific Time (US and Canada), Tijuana",
    "localTime":"04/17/2020",
    "timeScheduleEntry":[

      {

        "id":8,
        "objectId":2,
        "entityId":1,
        "dayId":1,
        "fromDate":"12:00 AM",
        "toDate":"12:00 PM",
        "tenantId":1
      },
      {

        "id":9,
        "objectId":2,
        "entityId":1,
        "dayId":2,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM",
        "tenantId":1
      },
      {

        "id":10,
        "objectId":2,
        "entityId":1,
        "dayId":3,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM",
        "tenantId":1
      },
      {

        "id":11,
        "objectId":2,
        "entityId":1,
        "dayId":4,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM",
        "tenantId":1
      },
      {

        "id":12,
        "objectId":2,
        "entityId":1,
```

```
        "dayId":5,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM",
        "tenantId":1
      },
      {

        "id":13,
        "objectId":2,
        "entityId":1,
        "dayId":6,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM",
        "tenantId":1
      },
      {

        "id":14,
        "objectId":2,
        "entityId":1,
        "dayId":7,
        "fromDate":"12:00 AM",
        "toDate":"12:00 PM",
        "tenantId":1
      }
    ]
  }
}
```


API for user to entry to specific calendar

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/calendar/entries?utid=0E500219AC7B988801F3D3CB813BCAAE&responseType=json&_=1586355716734
- Response-


```
{

  "personCalendar":{

    "calendarEntry":[

      {

        "id":1,
        "objectId":2,
        "entityId":1,
        "name":"Labour day",
        "dayTypeId":2,
        "stringCalendarDate":"05/01/2020"
      }
    ]
  }
}
```


Preferences

- Method - GET

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/people/currentuser?responseType=json&_=1586355716742 – mentioned above

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/information/modules?utid=0E500219AC7B988801F3D3CB813BCAAE&responseType=json&_=1586355716743
- Response -


```
{

  "List":[

    {

      "moduleId":7,
      "moduleName":"Administration",
      "logicName":"administration"
    },
    {

      "moduleId":15,
      "moduleName":"Advanced Reporting",
      "logicName":"advancedreporting"
    },

   . . .
    {

      "moduleId":27,
      "moduleName":"User Management",
      "logicName":"usermanagement"
    }
  ]
}
```


- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/1/smviews?utid=0E500219AC7B988801F3D3CB813BCAAE&responseType=json&_=1586355716744
- Response -


```
{

  "List":[

    {

      "display":true,
      "isActiveView":false,
      "isGlobal":true,
      "isGlobalSearch":false,
      "isPublic":false,
      "order":5,
      "viewId":"Administration",
      "viewName":"Administration",
      "datatableName":"ServiceMgrStaticViews",
      "filters":[


      ]
```

```
    },
    {

      "display":false,
      "isActiveView":false,
      "isGlobal":true,
      "isGlobalSearch":false,
      "isPublic":false,
      "order":100,
      "viewId":"DueTomorrow",
      "viewName":"Due Tomorrow",
      "datatableName":"ServiceMgrStaticViews",
      "filters":[


      ]
    },


  . . .

    {

      "display":true,
      "isActiveView":false,
      "isGlobal":true,
      "isGlobalSearch":false,
      "isPublic":false,
      "order":1,
      "viewId":"AvailableWork",
      "viewName":"Available Work",
      "datatableName":"ServiceMgrStaticViews",
      "filters":[


      ]
    },
    {

      "display":true,
      "isActiveView":false,
      "isGlobal":true,
      "isGlobalSearch":false,
      "isPublic":false,
      "order":6,
      "viewId":"MyRequisition",
      "viewName":"Recent Requisitions",
      "datatableName":"ServiceMgrRequisitionView",
      "filters":[


      ]
    }
  ]
}
```


API to get privilege page

- Method - GET

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/people/currentuser?responseType=json&_=1586355716755 - mentioned above

- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/people/1/roles/capabilities?utid=0E500219AC7B988801F3D3CB813BCAAE&roleNameLike=&responseType=json&_=1586355716756
- Response -


```
{
  "List":[
    {
      "0 Service Icons":[
        {
          "systemCapabilityId":80,
          "moduleId":18,
          "name":"Access Service Item Instance Data",
          "inherited":false,
          "moduleName":"Service Item Manager",
          "logicName":"access_service_item_data"
        },
        {
          "systemCapabilityId":79,
          "moduleId":18,
          "name":"Access Service Item Definition",
          "inherited":false,
          "moduleName":"Service Item Manager",
          "logicName":"access_service_item_definition"
        },
        . . .
        {
          "systemCapabilityId":66,
          "moduleId":17,
          "name":"NSAPI Access",
          "inherited":false,
          "moduleName":"Web Services",
          "logicName":"web_services_nsapi_access"
        }
      ],
      "1 Private Cloud IaaS Group":[
        {
          "systemCapabilityId":80,
          "moduleId":18,
          "name":"Access Service Item Instance Data",
          "inherited":false,
          "moduleName":"Service Item Manager",
          "logicName":"access_service_item_data"
        },
        {
          "systemCapabilityId":79,
          "moduleId":18,
          "name":"Access Service Item Definition",
          "inherited":false,
          "moduleName":"Service Item Manager",
          "logicName":"access_service_item_definition"
        },

        . . .

        {
          "systemCapabilityId":66,
```

```
      "moduleId":17,
      "name":"NSAPI Access",
      "inherited":false,
      "moduleName":"Web Services",
      "logicName":"web_services_nsapi_access"
    }
  ],
  "2 Unified Communication Group":[
    {
      "systemCapabilityId":80,
      "moduleId":18,
      "name":"Access Service Item Instance Data",
      "inherited":false,
      "moduleName":"Service Item Manager",
      "logicName":"access_service_item_data"
    },
    {
      "systemCapabilityId":79,
      "moduleId":18,
      "name":"Access Service Item Definition",
      "inherited":false,
      "moduleName":"Service Item Manager",
      "logicName":"access_service_item_definition"
    },

    . . .


    {
      "systemCapabilityId":66,
      "moduleId":17,
      "name":"NSAPI Access",
      "inherited":false,
      "moduleName":"Web Services",
      "logicName":"web_services_nsapi_access"
    }
  ],
  "3 Unified Workplace Demo":[
    {
      "systemCapabilityId":80,
      "moduleId":18,
      "name":"Access Service Item Instance Data",
      "inherited":false,
      "moduleName":"Service Item Manager",
      "logicName":"access_service_item_data"
    },
    {
      "systemCapabilityId":79,
      "moduleId":18,
      "name":"Access Service Item Definition",
      "inherited":false,
      "moduleName":"Service Item Manager",
      "logicName":"access_service_item_definition"
    },

    . . .

    {
      "systemCapabilityId":66,
      "moduleId":17,
      "name":"NSAPI Access",
      "inherited":false,
```

```
        "moduleName":"Web Services",
        "logicName":"web_services_nsapi_access"
      }
    ],
    "4 BYOD / Mobility Group":[
      {
        "systemCapabilityId":80,
        "moduleId":18,
        "name":"Access Service Item Instance Data",
        "inherited":false,
        "moduleName":"Service Item Manager",
        "logicName":"access_service_item_data"
      },
      {
        "systemCapabilityId":79,
        "moduleId":18,
        "name":"Access Service Item Definition",
        "inherited":false,
        "moduleName":"Service Item Manager",
        "logicName":"access_service_item_definition"
      },

      . . .

      {
        "systemCapabilityId":66,
        "moduleId":17,
        "name":"NSAPI Access",
        "inherited":false,
        "moduleName":"Web Services",
        "logicName":"web_services_nsapi_access"
      }
    ],



       . . .


    "[SQL Server] SG - Tenant-Information Reserved AFC":[
      {
        "systemCapabilityId":80,
        "moduleId":18,
        "name":"Access Service Item Instance Data",
        "inherited":false,
        "moduleName":"Service Item Manager",
        "logicName":"access_service_item_data"
      },

      . . .
      {
        "systemCapabilityId":66,
        "moduleId":17,
        "name":"NSAPI Access",
        "inherited":false,
        "moduleName":"Web Services",
        "logicName":"web_services_nsapi_access"
      }
    ]
  }
```

```
  ]
}
```


- Method - GET
- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/people/1/roles/capabilities?utid=F699F3DBF83E7D23E126004E46246C1F&roleNameLike={roleName}&responseType=json&_=1586750555449

- Example
  : http://localhost:8080/RequestCenter/nsapi/directory/people/1/roles/capabilities?utid=83BEDED575D568A017ECC1E5E4B0BFFF&roleNameLike=Cloud%20Center%20Reserved%20Services&responseType=json&_=1587123121426
- Response


```
{

  "List":[

    {

      "Cloud Center Reserved Services":[

        {

          "systemCapabilityId":80,
          "moduleId":18,
          "name":"Access Service Item Instance Data",
          "inherited":false,
          "moduleName":"Service Item Manager",
          "logicName":"access_service_item_data"
        },

                . . .

        {

          "systemCapabilityId":66,
          "moduleId":17,
          "name":"NSAPI Access",
          "inherited":false,
          "moduleName":"Web Services",
          "logicName":"web_services_nsapi_access"
        }
      ]
    }
  ]
}
```


Update user

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/information?utid=CA54557D8001B972743F195AA73568AD&responseType=json

- Method - PUT
- Payload:


```
{

  "person":{

    "firstName":"admin",
    "lastName":"admin",
    "timeZoneId":"256",
    "homeOrganizationalUnitId":1,
    "login":"admin",
    "localeId":"1",
    "statusId":1,
    "contacts":[

      {

        "contactId":1,
        "contactTypeId":1,
        "contactType":"Email",
        "value":"internal@newscale.com"
      }
    ],
    "addresses":[

      {

        "addressId":1,
        "addressTypeId":1,
        "addressType":"Company Address",
        "state":"",
        "zip":"",
        "city":"",
        "country":"",
        "street1":"",
        "street2":"",
        "building":"",
        "buildingLevel":"",
        "cubicle":"",
        "office":""
      },
      {

        "addressId":2,
        "addressTypeId":2,
        "addressType":"Personal Address",
        "state":"",
        "zip":"",
        "city":"",
        "country":"",
        "street1":"",
        "street2":""
      }
    ]
  }
}
```

Response -


```
{
  "status-message":{
    "code":"TA_116",
    "value":"Person Profile updated successfully"
  }
}
```


Update Calendar

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/calendar/timeschedule?utid=F699F3DBF83E7D23E126004E46246C1F&responseType=json
-
- Method - PUT
- Payload -


```
{
  "personCalendar":{
    "timeZoneName":"(GMT-08:00) Pacific Time (US and Canada), Tijuana",
    "localTime":"04/12/2020",
    "timeScheduleEntry":[
      {
        "id":8,
        "fromDate":"12:00 PM",
        "toDate":"12:00 PM"
      },
      {
        "id":9,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM"
      },
      {
        "id":10,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM"
      },
      {
        "id":11,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM"
      },
      {
        "id":12,
        "fromDate":"09:00 AM",
```

```
        "toDate":"5:00 PM"
      },
      {

        "id":13,
        "fromDate":"09:00 AM",
        "toDate":"5:00 PM"
      },
      {

        "id":14,
        "fromDate":"12:00 PM",
        "toDate":"12:00 PM"
      }
    ]
  }
}
```


Response -


```
{

  "status-message":{

    "code":"TA_131",
    "value":"Time schedule updated successfully."
  }
}
```


API to add additional date which is not available in regular calendar

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/calendar/entries?utid=F699F3DBF83E7D23E126004E46246C1F&responseType=json

- Method - PUT
- Payload -


```
{

  "personCalendar":{

    "calendarEntry":[

      {

        "name":"Labour day",
        "dayTypeId":2,
        "stringCalendarDate":"05/01/2020"
      }
    ]
  }
}
```


Response -


```
{
```

```
  "status-message":{

    "code":"TA_131",
    "value":"Additional dates added successfully."
  }
}
```


Update - Preferences

- Request
  URL: http://localhost:8080/RequestCenter/nsapi/directory/v2/person/1/preferences?oldShortDateFormat=MM/dd/yyyy&utid=F699F3DBF83E7D23E126004E46246C1F&responseType=json

- Method - PUT

- Payload : -


```
{

  "person":{

    "preferences":{

      "preferenceId":1,
      "shortDateSeperator":"/",
      "shortDateFormat":"MM/dd/yyyy",
      "longDateFormat":"MMMM dd, yyyy",
      "loginModuleId":21,
      "serviceManagerView":"datatable.do?datatableFormAction=activateSavedView&viewID=MyWork&datatableID=ServiceMgrStaticViews&returnToURL=%2FRequestCenter%2Fservicemanager%2Fhomepage.do%3FdatatableID%3DServiceMgrStaticViews",
      "serviceManagerStatus":"All Ongoing",
      "timeFormat":"h:mm aa",
      "viewAuthorizationsPortlet":true,
      "viewMyServiceItemsPortlet":true,
      "nextGenarationCatalog":true,
      "authorizationDelegateId":"0",
      "delegationStartDate":"",
      "delegationEndDate":""
    }
  }
}
```


Response. -


```
{

  "status-message":{

    "code":"TA_116",
    "value":"Person Profile updated successfully"
  }
}
```

API to Get Resource strings

Get Resource String API values for specified key and body JSON format: [11789,11790]. If we want to pass placeholders to resource strings, like param1, param2...., param5 we can pass like JSON format \"5349#param1#param2#param3#param4#param5\".

Example: [11789,11790,\"5349#1#100\"]

- Method : PUT
- URL : /RequestCenter/nsapi/v1/common/resourcestrings
- Headers : Content-Type = application/json
- Payload : ["5349#1#100",63, 123]
- Response :

```
{
"63": "Logout",
"123": "Key Customer Project",
"5349": "Please enter a number between 1 and 100."
}
```

API to Import AO Workflows

- Method : PUT
- URL : RequestCenter/nsapi/cloud/ao/{shortName}
- Response message : 15 AO Workflows imported successfully for 'PSC_SERVICE' category.

API to Re-import AO Workflow Service(s)

- Method : PUT
- URL : RequestCenter/nsapi/cloud/ao/reimport/{shortName}
- Request body : {"List":["SRI::AO::0166CV0K6YBZJ4GufPGfqyfTIjLaoFFG4Lv::1218"]}
- Response message : Re-import Completed Successfully

API to Show Log API's

Get list of log iterations

- Method : GET
- URL : RequestCenter/nsapi/cloud/ao/{shortName}/logs
- Response message :

```
{
   "Map": {
      "4": "4-30-19 9:51 AM",
     "5": "4-30-19 9:30 PM",
      "6": "5-3-19 4:17 AM",
      "Fri May 03 11:17:45 2019": {
         "6": [
                  "AO Workflows sync process started at 2019-04-30 17:39:01",
                   "AO Workflows Discovery Done in 0.324 seconds",
                   "AO Workflows sync process completed at 2019-04-30 17:39:10"          ]
      }
   }
}
```

Get one iteration logs

- Method : GET
- URL : RequestCenter/nsapi/cloud/ao/{shortName}/logs/{iterationId}
- Response message :

```
{
   "Map": {
      "1": [
         "AO Workflows sync process started at 2019-04-30 17:39:01",
         "AO Workflows Discovery Done in 0.324 seconds",
         "AO Workflows sync process completed at 2019-04-30 17:39:10"
      ]
   }
}
```


API to Get list of AO Workflows

- Method : GET
- URL : RequestCenter/nsapi/cloud/ao/reimport/{shortName}/workflows
- Response message :


```
{
   "WorkFlowServiceList": {
      "startRow": 0,
      "totalCount": 0,
      "recordSize": 3,
      "workflowServices": [
         {
            "workFlowName": "CCS VM Actions",
            "externalId": "0166CV0K6YBZJ4GufPGfqyfTljLaoFFG4Lv",
            "serviceName": "CCS VM Actions (SRI)",
            "serviceId": 1218,
            "createdOn": 1556606350467,
            "cloudConnectionId": 30
         },
         {
            "workFlowName": "Create CCS Deployment for CentOS",
            "externalId": "0166D3O1A9NTP21PMswoZSFYhIm7EAopYyC",
            "serviceName": "Create CCS Deployment for CentOS (SRI)",
            "serviceId": 1217,
            "createdOn": 1556606347687,
            "cloudConnectionId": 30
         },
         {
            "workFlowName": "To UPPER",
            "externalId": "016BD6EZDHLRG708Shs9arXyWYovZrS0HTe",
            "serviceName": "To UPPER (SRI)",
            "serviceId": 189,
            "createdOn": 1556606345083,
            "cloudConnectionId": 30
         }
      ]
   }
}
```

Introduced in 12.1_Patch_v9

PSC Service Catalog and Service Manager Website UI Decoupling for Standalone Wildfly

PSC supports for hosting decoupled Service Catalog and Service Manager websites on a server that is separate from the one where Prime Service Catalog has been installed. The separately hosted Service Catalog website can only utilize the Material Form Server renderer (Introduced in 12.1_Patch_v8) to generate the Service form.

Overview

PSC now supports hosting of the ServiceCatalog and ServiceManager websites on a web server different from the server where RequestCenter is deployed (Wildfly). These decoupled websites make REST API calls to the RequestCenter.war. ServiceCatalog and ServiceManager, if hosted on a different web server (Nginx, tomcat, etc.) can be accessed using the web server domain name and IP. End users who wish to use only ServiceCatalog and ServiceManager are provided with the URL to the decoupled websites. While the websites can be hosted on any web server, they have been tested and certified for Nginx and tomcat.
Note:

1. In the decoupled UI of ServiceCatalog, the ServiceForm can be rendered using only MaterialUI and hence has limited capabilities as of now.
2. The previous behavior of ServiceCatalog and ServiceManager hosted on the Server where RequestCenter is deployed from RequestCenter.war remains unchanged.

Setting up Tomcat web server

Install Tomcat 9 on Windows. You will be able to access tomcat and manage from outside the VM.
If you are not able to access tomcat with IP or Hostname do the following

1. Open server.xml from below location-
    a. <Install_Dir>\Apache Software Foundation\Tomcat 9.0\conf
2. Add address="0.0.0.0" to connector string
   Note: Instead of 0.0.0.0 you can also can map the IP address of the web server VM .

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"
           address="0.0.0.0" />
```

3. Change the defaultHost and Hostname values which should be updated with IP address or Host Name of the VM from localhost.
4. Access the http:<WebServer_IP or HostName>:<WebServer_PORT>/ –
5. Welcome page of webserver is loaded.

Note for Ngnix webserver: The user should modify the **nginx_conf** file and map the server name to Hostname of the VM. The deployment folder for Nginx is HTML folder.
Note: Both PSC Server where RequestCenter is deployed (Application Server) and UI Decoupled System (Webserver) should be in the same domain. The decoupled website supports only MaterialFormServer for ServiceForm to either order or update.

Setting up PSC UI on a Different Webserver

1. Choose VM where the tomcat is installed and go to the deployment folder of webserver,in this case, its C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps .
2. Create a folder structure like below
    a. Create RequestCenter, Login and Logout folder.
    b. Inside the RequestCenter create subfolder called website.
3. Copy the ServiceCatalog, Service Manager and FormServer folder from the PSC server where RequestCenter is deployed “wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website and place it under the webapps/RequestCenter/website/ of tomcat VM.
4. Copy the content of LoginWebsite dist folder from wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website\LoginWebsite and place into the webapps/ or Log in of the tomcat VM.
5. Copy the LoginConfig.Json form wildfly10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website and place into the webapps/RequestCenter/website.
6. Copy the help and jsbundle folder from the wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war and paste into the webapps/RequestCenter of the tomcat vm.
7. Copy the content of ssologout folder from the wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war\website\LoginWebsite\ssoLogout to the webapps/Logout folder of tomcat VM.
8. Modify the JSON file.
    - Host: PSC server where RequestCenter is deployed is mandatory for decoupled website.
    - Cookie Domain: PSC server where RequestCenter is deployed belongs to the domain. Example: If the VM belongs to cisco.com and then cookie domain will be cisco.com.

- serviceCatalogUrl : Provide the Service Catalog URL which is the IP or hostname of tomcat Server hosted.
- serviceManagerUrl : Provide the Service Catalog URL which is the IP or hostname of tomcat Server hosted.

If the user wants to provide the custom information for below strings on the Log in page, the information can be passed here, and the custom information will reflect on the Log in Page.

- copyRightText : Copyright message to display on Log in Page.
- displayString : Product Name (Cisco Service Prime Catalog is the default).
- marketingVersion: Product Version.

Sample LoginConfig.json

```
{

  "host":"http://<Host_Name>:<Port>",

  "cookieDomain":".cisco.com",

  "ssoUrl":"",

  "loginModule":{

"serviceCatalogUrl":"http://<Host_Name_of_Tomcat>/RequestCenter/website/ServiceCatalogWebsite/application/index.html",
  "serviceManagerUrl":"http://<Host_Name_of_Tomcat>/RequestCenter/website/ServiceManagerWebsite/application/servicemanager.html?route=servicemanager#homepage"
  },

  "loginPageInfo" : {
    "copyRightText" : "",
    "displayString" : "",
    "marketingVersion":""
  }

}
```

9. Restart the Tomcat Server
10. Enter the tomcat server details as below
    http://<IP_WebServer>:<WebServer_Port>/Login

The URL opens the PSC login page and upon logging in, the user can see ServiceCatalog and ServiceManager.

Authentication and Session management in Decoupled PSC UI

There are four areas of configurationout for decoupling of PSC UI

1. PSC DB Credentials
2. LDAP Authentication
3. LDAP with External Authentication
4. SAML SSO

PSC DB Credentials

Prerequisite: PSC Server where RequestCenter is deployed should be installed on the VM with Wildfly app server, and it should be up and running. You should be able to login with PSC DB credentials successfully. Make sure that the latest patch is installed on PSC instance.

To use the system with Decouple feature:

1. Add the properties mentioned into **Newscale.properties** for PSC server where RequestCenter is deployed.

```
##################################################################
#decoupled UI domain name
decoupled.cookie.domain=.cisco.com
##################################################################
#decoupled UI authentication mode
decoupled.authentication=true
##################################################################
#decoupled UI landing page url for servicecatalog module
```

decoupled.servicecatalog.url=http://host_name:port/RequestCenter/website/ServiceCatalogWebsite/application/index.html
##
#decoupled UI landing page url for servicemanager module
decoupled.servicemanager.url=http://host_name:port/RequestCenter/website/ServiceManagerWebsite/application/servicemanager.html?route=servicemanager#homepage
#Only if User is using SSO with SAML##########
decoupled.saml.logout.url=http://<Host_Name>:<Port>

2. Modify the Standalone_full.xml
3. Add the italicized entries
4. Once CORS headers is updated, change the header-value to the FQDN where UI is deployed.ui.cisco.com.
5. The CORS specific header to RequestCenter.war) to the Standalone_full.xml file for ajax request

```
<subsystem xmlns="urn: jboss:domain:undertow:3.1">
  <buffer-cache name="default"/>
  <server name="default-server">
    <http-listener name="default" socket-binding="http" max-post-size="104857600" redirect-socket="https" enable-http2="true"/>
    <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>
    <host name="default-host" alias="localhost">
      <location name="/" handler="welcome-content"/>
      <filter-ref name="server-header"/>
      <filter-ref name="x-powered-by-header"/>
      <filter-ref name="Access-Control-Allow-Origin"/>
      <filter-ref name="Access-Control-Allow-Headers"/>
      <filter-ref name="Access-Control-Allow-Credentials"/>
    </host>
  </server>
  <servlet-container name="default">
    <jsp-config/>
    <websockets/>
  </servlet-container>
  <handlers>
    <file name="welcome-content" path="${jboss.home.dir}/welcome-content"/>
  </handlers>
  <filters>
    <response-header name="server-header" header-name="Server" header-value="WildFly/10"/>
    <response-header name="x-powered-by-header" header-name="X-Powered-By" header-value="Undertow/1"/>
    <response-header name="Access-Control-Allow-Origin" header-name="Access-Control-Allow-Origin" header-value="ui.cisco.com"/>
    <response-header name="Access-Control-Allow-Headers" header-name="Access-Control-Allow-Headers" header-value="accept, authorization, content-type, x-requested-with, Content-Type"/>
    <response-header name="Access-Control-Allow-Credentials" header-name="Access-Control-Allow-Credentials" header-value="true"/>
  </filters>
</subsystem>
```

LDAP Authentication

Prerequisite: LDAP should be configured on system and user should be able to authenticate with Windows authentication. User should be able to perform the single sign on with PSC Server where RequestCenter is deployed.
Find below configuration to make SSO work using Decoupled system

1. See Using PSC DB Credentials for settings. Apart from the setting given you must ensure that additional setting on the IIS are also done.
2. In the web server
    1. Modify the LoginConfig.json
    2. Set the below flag to true
    3. Provide the SSOUrls
    4. "isWindowAuth":"true",
    5. "ssoUrl":"https://<Host Name of PSC where RequestCenter is deployed >/RequestCenter/sessioninfo.jsp?isDecoupled=true",
    6. Go to the VM where IIS is hosted.

7. Download the CORS module from https://www.iis.net/downloads/microsoft/iis-cors-module
8. Install the CORS module.
9. Open the Web.config of IIS and add the below tags within the system.webserver

```
<cors enabled="true">
<add origin="http://vm235.cisco.com" allowCredentials="true" >
<allowMethods>
<add method="PUT" />
<add method="GET" />
<add method="POST" />
<add method="DELETE" />
<add method="OPTIONS" />
</allowMethods>
</add>
</cors>
<httpProtocol>
<customHeaders>
<remove name="X-Powered-By" />
<add name="Access-Control-Allow-Headers" value="x-requested-with, Content-Type, origin, Authorization, Accept, locale, access_token, utid, Range" />
<add name="X-Powered-By" value="ASP.NET,Undertow/1" />
</customHeaders>
</httpProtocol>
```

Below is the Example of IIS web.config

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<system.web>
<!-- Default <,>,*,%,&,:,\,? or %u003c,%u003e,%u002a,%u0025,%u0026,%u003a,%u005c,%u003f -->
<httpRuntime maxUrlLength="1000" requestPathInvalidCharacters="%u003c,%u003e,%u0025,%u005c,%u003f,%u252F" />

</system.web>
<system.webServer>
<modules>
<remove name="WebDAVModule" />
</modules>
<handlers>
<remove name="WebDAV" />
</handlers>
<security>
<requestFiltering allowDoubleEscaping="true">
<requestLimits maxAllowedContentLength="104857600" maxQueryString="2048" />
<verbs>
<add verb="POST" allowed="true" />
<add verb="PUT" allowed="true" />
<add verb="DELETE" allowed="true" />
<add verb="GET" allowed="true" />
<add verb="OPTIONS" allowed="true" />
</verbs>
</requestFiltering>
</security>
<httpErrors>
<remove statusCode="403" subStatusCode="-1" />
<error statusCode="403" prefixLanguageFilePath="" path="https://10.78.0.251/RequestCenter" responseMode="Redirect" />
</httpErrors>
<cors enabled="true">
<add origin="http://vm235.cisco.com" allowCredentials="true" >
<allowMethods>
<add method="PUT" />
<add method="GET" />
<add method="POST" />
<add method="DELETE" />
```

```
                                        <add method="OPTIONS" />
                </allowMethods>
              </add>
            </cors>
            <httpProtocol>
              <customHeaders>
                <remove name="X-Powered-By" />
                <add name="Access-Control-Allow-Headers" value="x-requested-with, Content-Type, origin,
Authorization, Accept, locale, access_token, utid, Range" />
                <add name="X-Powered-By" value="ASP.NET,Undertow/1" />
              </customHeaders>
            </httpProtocol>
          </system.webServer>
</configuration>
```

Save the file and restart the IIS Web Server and hit the Decoupled URL to verify.

LDAP with External Authentication

Prerequisite: LDAP should be configured on system and user should be able to authenticate with external authentication. The user should be able to perform the external authentication on PSC server where RequestCenter is deployed.
Below are the configurations for external Auth Using Decoupled.

1. Refer Using PSC DB Credentials for settings, apart from the setting given you must ensure that additional setting on the LoginConfig.Json are also done.
2. From Tomcat Side
    a. Modify the LoginConfig.json
    b. Set the below flag to true
    "isWindowAuth":"true",

SAML SSO

Prerequisite: SAML SSO should be enabled and required settings should be done on the PSC server where RequestCenter is deployed. The user should be able to Login with SAML Auth.
Below are the configurations for SSO with SAML Auth Using Decoupled

1. Refer Using PSC DB Credentials for settings, apart from the setting given you must ensure that additional setting on the LoginConfig.Json are also done.
2. From Tomcat Side
    a. Modify the LoginConfig.json

    Set the below flag to true and provide the SSO URL
    "isWindowAuth":"false",
    "ssoUrl":"https://<Host Name of PSC server where RequestCenter is deployed>/RequestCenter/sessioninfo.jsp?isDecoupled=true",

    Note: To use CustomWebsite instead of the out of the box ServiceCatalog or ServiceManager websites:

    1. Create the respective folder in the web server of webapps, or RequestCenter or website.
    2. Provide redirect URLs in the newscale.properties and LoginConfig.json file.

Troubleshooting while Authentication and Session Management

- The service using the Image URL from the system relative path, then the same images should be placed in the tomcat server.
- Under the webapps/RequestCenter, create a folder called images and place all the images which are used for image URL in the service. (These images can be found in the PSC instance wildfly-10.1.0.Final\ServiceCatalogServer\deployments\RequestCenter.war/images.
- Restart the tomcat server and Login, and you can see the image load for the respective service.
- When PSC server where RequestCenter is deployed points to the IIS Web Server and UI decoupled is on another tomcat server or any other web server, the system does not display an error message even if the user enters wrong credentials. To work around this issue,

1. Edit feature setting page details radio button and save.
2. Restart the IIS Server.
   Please refer below link for reference.https://stackoverflow.com/questions/2640526/detailed-500-error-message-asp-iis-7-5

- In IIS make all error code to details, so that any error messages we can capture from PSC instance and show it on the IIS or end-user. Ex 412,405,404,406 etc.

API to Reset the Password

- API – GET Method –
- End point -
  http://localhost:8080/RequestCenter/nsapi/authentication/login?forgotPassword=true&NSA_LOGIN_NAME=user1

Header Parameters

- Content-Type - application/json
  Response of the API will be in JSON as is below-

```
{
  "nsapi-response": {
    "NSA_FORGOT_PASSWORD_OPERATION_SUCCESS": "Your Request for a new password is being processed. An email has been sent to your email address."
  }
}
```

API to Generate System Compliant Sample Password

- API – GET Method –
- End point - http://localhost:8080/RequestCenter/nsapi/directory/randompass
  Header Parameters
- Content-Type - application/json
  Response of the API will be in JSON as is below-

```
{
  "List": [
    "XJ(j@3x4"
  ]
}
```

API to Change Password

- API – GET Method –
- End point -
  http://localhost:8080/RequestCenter/nsapi/authentication/login?resetPassword=true&NSA_LOGIN_NAME=user1&CURRENT_PASSWORD=!@^_YWRtaW4=_^@!&NEW_PASSWORD=!@^_YWRtaW4=_^@!&CONFIRM_PASSWORD=!@^_YWRtaW4=_^@!
  Header Parameters
- Content-Type - application/json
- The passwords sent are base64 encoded.
  Response of the API will be in JSON as is below-

```
{
  "nsapi-response": {
    "NSA_PASSWORD_UPDATE_SUCCESS": "You have successfully changed your password. You may log in now with the new password."
  }
}
```

API to Log in and Get Session Id

- API – GET Method –
- End point - http://localhost:8080/RequestCenter/nsapi/authentication/token?persistent=true
  Header Parameters
- Content-Type - application/json
  Response of the API will be in JSON as is below-

```
{
  "sessiontoken": {
    "utid": "P_D7E79E1479600246A4932444172E058D",
    "sessionId": "yIspNsKqJUbiy0zdF8eLzHC5NOrYUxfyCmTn_QkU",
    "loginModule": "Service Catalog",
    "loginModuleLogicName": "servicecatalog"
```

```
  }
}
```


API to Logout of the System

- API – GET Method –
- End point - http://localhost:8080/RequestCenter/nsapi/authentication/logout?persistent=true&isDecoupled=true
  Header Parameters
- Content-Type - application/json
  Response of the API will be in JSON as is below-


```
{
  "nsapi-response": {
    "Success": "User is Logged out."
  }
}
```


API for Global Configuration

Configuration API is used in Log in page to get site level parameters, like compliance website enabled or is SAML authentication enabled etc.

- API – GET Method –
  End point - http://localhost:8080/RequestCenter/nsapi/global/configuration
  Header Parameters
- Content-Type - application/json
  Response of the API will be in JSON as is below -


```
{
  "Map": {
    "enableCustomLoginLogout": false,
    "isSAMLSSO": false,
    "gdprConsentUrl": "http://kjcjbvcbajs.com",
    "enableComplianceWebsite": true
  }
```


API to Execute the DDR Rule for Browser based Events

This API was previously introduced in version **Introduced in 12.1_Patch_v6** where ruleId was part of PathParam,
In this patch the API has been upgraded to take ruleId as queryparam also.

- API – PUT Method –
  End point - End point - http://ServerURL:Port/RequestCenter/nsapi/catalog/v1/service/Id/rule?ruleId=<encoded ruleId>
  Header Parameters
- Content-Type - application/json
- Accept - application/json
  Response of the API will be in JSON as is below-


```
{
  "listOfValues": [
    [
      {
        "key": "<dictionary_name>.<field_name>",
        "value": [
          {
            "label": "<label_configured_in_rule>",
            "value": "<value_mapped>"
          },
          {
            "label": "3-tier Production Application SVG",
            "value": "7"
          },
          {
            "label": "Activity SVG",
            "value": "8"
          },
```

```
            {
              "label": "Apple Macbook",
              "value": "9"
            }
          ]
        }
      ]
    ],
    "listOfTargets": [
      "<dictionary_name>.<field_name>"
    ]
  }
```

Introduced in 12.1_Patch_v8

Note: Particle Form Server support is dropped and the folder has to be removed from:

/Requestcenter.war/website/FormServer

Service Form Rendering using Material UI Component.

There is a provision to render the Service Form Using the Material UI Components and it is designed on **ReactJS** framework**,** where all the features are similar to classic service form and it is customizable. This new service form is designed using APIs and it is decoupled from the backend. The out of the box Material UI ServiceForm supports all the fields except Grid Control and Paginated ServiceForm in this release.

Please refer the **New Service Form** section in Introduced in 12.1_Patch_v7 to understand on how the New Service Form UI works.

Enable New Service Form

The default rendered service form is classic and we can enable the new Material UI service form from the following locations,

- Administration→ Settings
- Service Designer> General tab

Note: JavaScript functions has to be re-written which should be compatible to Material Form Server and the existing JavaScript functions do not work with the new service form.

Location of New Service Form

The New Service Form Source Code is available at:

/Requestcenter.war/website/FormServer/MaterialFormServer

Browser Compatibility for Material Form Server

- ✓ Supported browsers:
  - Microsoft Edge Version 42,
  - Mozilla Firefox Version 64,
  - Google Chrome Version 70,
  - Safari Version 12.0
- **Note:** Internet Explorer 11 is depreciated and should not be used with material form server

Technology Stack for Material Form Server

The new service form UI uses the following libraries,

a) **NodeJS**
   - Base for the following technology stack

b) **WebPack version 4.20.2**
   - Used to compile the entire project and prepare the production build

c) **React JS version 16.7.0**
   - Used for developing the new material form server

d) **Material UI library version 3.9.0**
   - Used as a core library to design material form server

e) **Syntactically Awesome StyleSheets (Sass)**
   - Used for CSS styling

Modifying Custom Service Form

This section provides information about modifying the Service Form for designing new components or enhancing the existing components.

- To introduce a new library, update the **package.json** file.
  If required, make necessary changes in the following configuration file:
  - config/webpack.config.js

- To modify the PSC provided New Service Form code to make any custom changes on any component or to design new components, Update the following files in the correct order:

  a) Index.js
  b) App.js
  c) AppInit.js
  d) ServiceForm.js *(Creates service form header, body, and footer components)*

- To modify the new service form (CSS) styling folder root path is,

  - **/src/styles**

  All CSS style files are available at the parent folder of the above root path.
  All these files have the extension as **.scss**. You need to import and merge the **.scss** files into a single file, named as **shared.scss**. While including a new file, start the filename with underscore (_). For example, name a new file as **_BasicForm.scss** and import the file into **shared.scss**. The custom CSS for material ui component is placed in folder **Shared/cssUtility.js**

➢ **To compile all the new libraries and modified files:**
Pre-requisite:
  - ✓ Install the latest **node.js** library in the corresponding system.
  - ✓ Install **ReactJS** and other related libraries used in the new service form using “npm install”.
    Note: You must be connected to Internet to use the **npm install** command.

➢ **To compile all the changes and to prepare the production build, run the following command:**

  - **npm run build**

Note: Production build is a set of minified files.

- Copy the production build folder (**/dist**) and replace it in the following location:
  Requestcenter.war/website/FormServer/MaterialFormServer/dist

- **Material UI Home URL:** https://material-ui.com/

Limitations of Material UI with respect to the Service Form

Slider component

Material Form Server does not support the **onFocus** and **onBlur** events.

Date and DateTime components

Material UI Date and DateTime Components works on based on Selected User Short Date Format.

- Material UI does not support the following Date Formats: DD/MM/YY,DD/MM/YYYY,YY/MM/DD
  - In Safari Browser DD/Month/YY and DD/Month/YYYY will be replaced with YYYY/MM/DD
  - Following date formats are not supported In Microrsoft Edge: DD/Month/YY and DD/Month/YYYY
- The date and dateTime components supports slash (/), hyphen (-) and period (.) as a profile date separators, but UI will display only slash (/).
- The date and dateTime components do not support the **onClick** and **onBlur** events.

Note:

- Only the date formats supported by the Material UI and the corresponding browser will work.
- If the person field is set under the conditional rule then value should be passed as **personID::personLoginName**

Person Profile UI

The Person Profile UI has been replaced with BackBone JavaScript based UI and has supporting APIs to provide the required data. It is now available for customization.

The new Person Profile UI is consumed in the website of the below mentioned modules:

- Cloud Integrations
- Compliance
- Service Catalog
- Service Manager
- Tenant Management
- User Management

Service Catalog Website

The Service Catalog website is now fully API based website, which means all the struts calls have been removed. The files related to dashboard, reports, orders, authorizations & service items have now been moved under the Service Catalog Website**.**

Modifying Renderer Settings for New Service Form

- **Site Level Renderer Settings**
  Path to renderer settings at the admin level,
  Administrations > Settings > Select renderer > Classic or MaterialFormServer

- **Service Level Renderer Settings**
  Path to renderer settings at the service level,
  Service designer > Select a Service in left pane > Override Renderer > Classic or MaterialFormServer > Save

Accessing Localized JavaScript Strings

While developing custom scripts for a service, to access the localized resource strings from JavaScript for a service from a service designer module, use the following function:

getLocalizedString("<js_resource_id");

- See the Translating JavaScript Strings in Cisco Prime Service Catalog 12.1 Designer Guide

JavaScript Renderer Support

Java Script function definition can be defined for the different renderers at **Service Designer→Scripts**

When the Service Form is rendered, based on the renderer of the Service, the corresponding Java Script will be fetched and executed.

Authentication Token API

The response of the Authentication API has been enhanced and new response attributes are added: **sessionID**, **loginModule**, and **loginModuleLogicName,**

- **sessionID** – jsessionId which is unique for the session and used for authentication.
- **Loginmodule** – the default module that will be displayed post login. It is based on the Person Profile Settings.
- **loginModuleLogicName –** Unique identifier for the respective module.

Rest API

RequestCenter/nsapi/authentication/token?persistent=true

API Method: GET

Sample Payload:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<sessiontoken utid="P_95A5F7D47752CF96324466027606864B">

<sessionId>EYsEWXmVgCqaWZKAK1fMU_HJJ7m_QOOP0JEhVMrW</sessionId>

<loginModule>Service Catalog</loginModule>

<loginModuleLogicName>servicecatalog</loginModuleLogicName>

</sessiontoken>

Introduced in 12.1_Patch_v7

New Service Form UI

The new service form UI is designed using new technology stacks such as **ReactJS** with **ParticleUX**. Service Form UI is a website where a custom built ServiceForm implementation can be placed and the same will be rendered. The New Service Form UI consumes APIs to render service form and has been completely decoupled from the back end.
The Particle UI ServiceForm is in beta form and does not completely support all of the ServiceForm functions. However, the implementation can be used as a reference to develop custom ServiceForm UI. Also, no customization support is planned for the ParticleUX components currently.
This new service form can be enabled either from **Administration->Settings or from Service Designer->General tab**.
Note: For Particle form UI/Custom UI, java scripts needs to be written using the corresponding JavaScript libraries. Pre-existing Javascripts will not work in the new service form.

One implementation of UI logic which is particle UI has been provided and have three major components,

Components

Types of Service Form

1. **Classic Service Form**
   - Old Service Form
2. **Particle form server**
   - New Service Form designed using ReactJS with ParticleUX

Location of New Service Form

Note: Location for New Service Form Source Code is

/Requestcenter.war/website/FormServer/ParticleFormServer

Browser Compatibility for Particle Form Server

- ✓ Supports all the browsers for Eg.
  - IE Edge (Version 42),
  - Mozilla Firefox (Version 64),
  - Google Chrome (Version 70)
  - Safari (12.0)
- **Note:** IE11 is depreciated and not suggested to use with particle form server

Technology Stack for Particle Form Server

The new service form UI is designed using following libraries:-

f) **NodeJS**
   - Base for the below technology stack

g) **WebPack (version 4.20.2)**
   - Required to compiles the entire project and prepare the production build

h) **React JS (version 15.6.2)**
   - Used for developing the new particle form server

i) **Cisco Particle UI library (version 2.9.0)**
   - Used as a core library to design particle form server

j) **Sass (Syntactically Awesome StyleSheets)**

Used for CSS styling

ISF Library for Form Rules

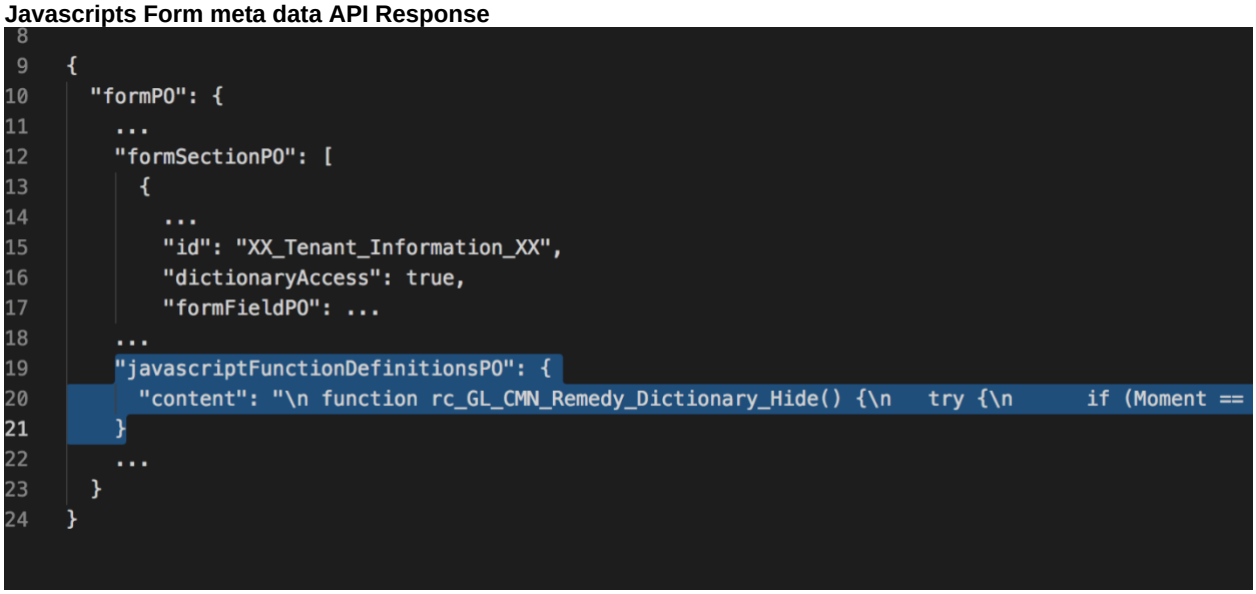
- Service Form Rule execution (CR and DDR) can be achieved with rule specific generated javascript functions. These functions in turn invokes functions in ISF library to achieve intended service form manipulation dynamically.
- A template file (formrules_javascript_tpl.ftl) generates rule specific javascript functions. This template also generate code to bind rules with their configured DOM event.
  - Example:
    - To execute a DDR rule, template generates a function which invokes ISF.isf_ddr_call()
    - To execute alert action, template generates a function which invokes ISF.isf_cr_alert()
- ✓ This template file can be found under path: **/RequestCenter.war/WEB-INF/classes/service-form-rules-js-templates/formrules_javascript_tpl.ftl**
- ✓ The path to this template file is configurable. Refer Newscale Properties Settings for FTL File Path section for more details.

- ✓ Patch is shipped with a template which uses FTL transformation to achieve this javascript code generation. This template can be rewritten to achieve different implementation of complete form rule executions. Refer to FTL Template to Generate Form Rules Specific javascripts section for more details.

- This generated javascript can be found in the response for form meta data API under attribute "javascriptFunctionDefinitionsPO".

- Execute this javascript to make these functions part of DOM. Executing this code will also initialize event to function map for form rules. This map is declared in ISF library.

Javascripts Form meta data API Response



On trigger of an event, service form will execute form rules javascript functions mapped to this event. ISF event to rule map will be used to retrieve these form rules javascript functions.

- ISF (Interactive Service Form) library is a javascript object defined "/ParticleFormServer/src/addins/isf-addin.js". We make it part of window object in browser.
  - Implementation details on ISF library functions can be found in Cisco Prime Service Catalog 12.1 Patch - Particle Form Server ISF Library Method Signatures on Cisco.com

FTL Template to Generate Form Rules Specific javascripts

The template takes in form rule configuration in xml and generates corresponding javascript.
To achieve complete execution of form rules the template should have

- A handler for each form rule to invoke corresponding ISF java script functions, these js functions will perform the intended rule behavior. ISF functions can also be changed under new renderer.
- Each handler will be bound to trigger on an event configured for the form rule.

External JavaScript

Guidelines to follow and write scripts for particle form server.

To Access serviceForm Data inside Custom Scripts

serviceForm object is passed as a last argument value to all script functions. It can be accessed in two ways,

1. **Additional** function **argument serviceForm**

```
function1(arg1, arg2,... , serviceForm) {
        //write script here
        .....
}
```

- "**arg1, arg2**,..." are the arguments that will be defined from "**Function Arguments**" section of service designer scripts.

- "**serviceForm**" is the argument that will have the serviceForm object and it should not be defined from "**Function Arguments**" section of service designer scripts.

2. **Using "arguments" javascript local variable inside** function **definition**
   The last argument is the **serviceForm** and can be accessed as below:

```
function2(arg1, arg2,...) {
        let serviceForm = arguments[arguments.length-1];
        //write script here
        .....
}
```

- Here, "**arg1, arg2**,..." are the arguments that will be defined from "**Function Arguments**" section of service designer scripts.

To get value of a Service Form Field

ISF.isf_getValue(serviceForm, <Dictionary_Name.Field_Name>);

```
function3() {
    let serviceForm = arguments[arguments.length-1];
    let fieldName = ISF.isf_getValue(serviceForm,"Dictinary1.Name");
}
```

To set value to a Service Form Field

ISF.setTargetFieldValue(serviceForm, <Dictionary_Name.Field_Name>, <Field_Value>);

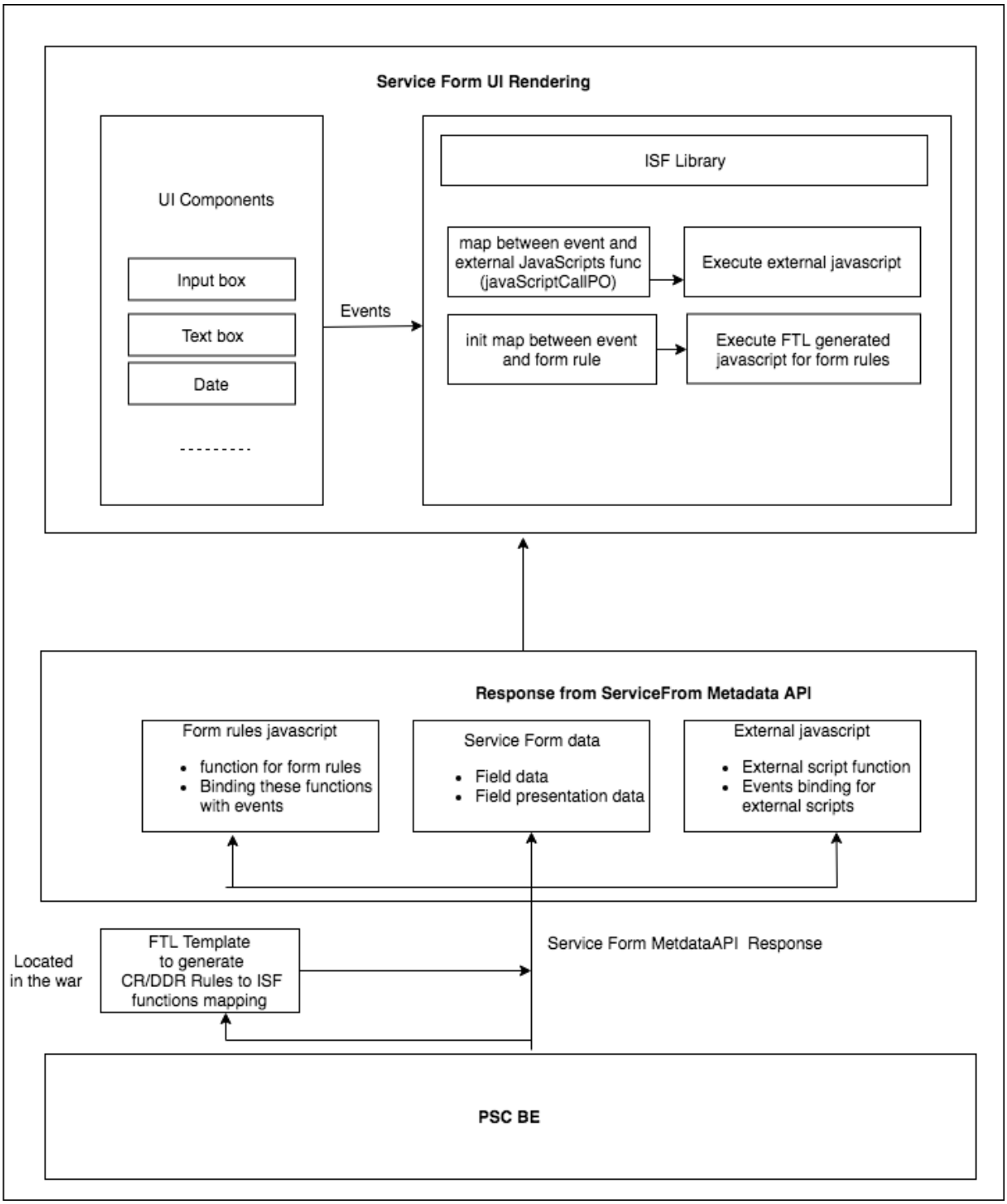
```
function4() {
let serviceForm = arguments[arguments.length-1];
let fieldName = ISF.isf_getValue(serviceForm,"Dictinary1.Name");

ISF.setTargetFieldValue(serviceForm, "Dictinary2.Name", fieldName);
}
```

Note: For remaining js functions that ISF library is supporting please refer below JS file.

RequestCenter.war ▸ website ▸ FormServer ▸ ParticleFormServer ▸ src ▸ addins ▸ **isf-addin.js**

Service Form UI Render Flow



Files To Be Changed For New Custom Service Form

Steps to modify the new service form for any type of enhancement in existing components or design of new components mentioned below,

- To introduce any new library the **package.json** has to update which is going to consume down the line project
  Note: If required then do the necessary changes in configuration files, such as:-
  - config/webpack.config.dev.js

- config/webpack.config.prod.js

- For modifying the PSC provided **New Service Form** code to do any custom changes on any component or to design new component below mentioned files needs to be touched and should be in a proper sequence

  e) Index.js
  f) App.js
  g) AppInit.js
  h) ServiceForm.js *(Service form header, body and footer components will be created here)*

- To change the new service form (CSS) styling folder root path is

  - **/src/styles**

  Note: All CSS style files are available on above mentioned root path parent folder. All these files are **(.scss)** extension based files and imported and merged into single file known as **(shared.scss).** If any new file is going to introduced then the file name must be start with underscore mark, for eg. **_BasicForm.scss** and this new file has to be imported in **(shared.scss)**

- Compilation of all the modified files and newly introduced libraries

  Pre-requisite:
  - ✓ Latest **node.js** library has to be installed in the corresponding system
  - ✓ Install **ReactJS** and other related libraries which are being consumed in the new service form using “npm install”.
    Note: **npm install** (For this command internet connection is must)

  Compilation:

  - **npm run build**

  Note: Above command will compile all the changes and prepare the production build which is a set of minified files.

- The production build folder **(/dist)** has to be copied and replaced in the following location,

  /Requestcenter.war/website/FormServer/ParticleFormServer/dist

Note: New service form doesn’t support UCSD and Cloud Center Services.

API Enhancements to New Service Form UI

Service Order Form Context API

Service Order Form Context API provides context parameters which are used for loading the service order form and now has an additional field **hasUnsubmittedRequisitions** which confirms whether the user has any unsubmitted requisition.

/RequestCenter/nsapi/catalog/v1/service/{id}/orderformcontext?customerId={customerID}

API Method: GET

Response Payload:


```
{
    "Map": {
        "siteAdminUser": true,
        "EnableOOBAutoComplete": false,
        "orderConfirmationDeliverToFormat": "Customer(Login)",
        "allowAddAttachment": true,
        "isNextGenUser": true,
        "clickableHelpIcon": false,
        "timeZone": "America/Tijuana",
        "hideNRDicts": false,
        "servicedescproperty": false,
        "hideMonitorFlag": false,
        "Serviceformelement.display.instructional.helptext": false,
```

```
                "isMessageInfo": "false",
                "DescriptionLimit": "0",
                "nsBasePath": "/RequestCenter/",
                "service": {
                        "serviceID": 150,
                        "name": "testService1",
                        "documentID": 0,
                        "estimatedCost": 0.0,
                        "expectedDuration": 0.0,
                        "hasPortal3": true,
                        "portalText3": "<p>TEST3</p>\n",
                        "isEntitlement": 0,
                        "isInactive": 0,
                        "priceDescription": "",
                        "shortPriceDescription": "",
                        "priceDisplaySchemaID": 0,
                        "pricingSchema": 0,
                        "revisionNumber": 34,
                        "allowFutureDelivery": true,
                        "isBundle": false,
                        "isTemplate": false,
                        "tenantRelevant": true,
                        "isOrderable": true,
                        "orderingMode": 3,
                        "computePrice": false,
                        "paginationViewMode": 0,
                        "isDefinedDuration": false,
                        "showOrderSummary": false
                },
                "allowUpdateQty": false,
                "sessionToken": "F85E174AB7425AD66555E5CBD517CDF4",
                "ldapLookupOnOnlyForPerson": true,
                "ldapLookupOn": true,
                "Serviceform.label.alignment": "right",
                "Serviceform.field.line.separator": false,
                "hasUnsubmittedRequisitions": false
        }
}
```


Requisition Entry API

Additional fields **momentLogicName** and **isBundle** is added to the requisitionentries API

- When we order a Service the Service goes through multiple Moments that are configured namely,
  - Ordering
  - Departmental Authorizations
  - Departmental Reviews
  - Service Group Authorizations
  - Service Group Reviews
  - Financial Authorizations
  - Service Delivery
  - Pricing
  - Service Completed

momentLogicName will have the appropriate value of the above moments and it is used in conditional rule, data retrieval rule for condition evaluation.

- When **isBundle=true** requisition entry is for a bundled service.

/RequestCenter/nsapi/transaction/requisitionentries/id/{id}?isNgcRequest=true&responseType=json

API Method: GET
Response Payload:


```
{
```

```
  "requisitionEntry": {
    "requisitionEntryId": 10,
    "requisitionId": 8,
    "serviceId": 142,
    "serviceDescription": "",
    "serviceName": "DDDR",
    "standardDuration": "Not Defined",
    "unitCost": "$ 0.00",
    "subTotal": "$ 0.00",
    "status": "Ongoing",
    "dueOnDate": "10/31/2018",
    "dueOnDateRaw": 1540981800000,
    "quantity": 1,
    "maxQuantity": 0,
    "serviceLevelDescription": "",
    "percentageCompleted": 0.0,
    "statusId": 1,
    "expectedDuration": 0.0,
    "isBundle": false,
    "reqEntryServiceURL": "<a href=\"#\" onclick=\"GB_showFullScreen('DDDR','/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&layout=popup_p&reqid=8&reqentryid=10&formAction=displayEntryStatus&performerID=&serviceid=142&requisitionId=8')\">DDDR</a>",
    "reqEntryServiceURLFor_Preparation": "<a href=\"#\" onclick=\"GB_showFullScreen('DDDR','/RequestCenter/myservices/navigate.do?query=vieworderform&layout=popup_p&serviceid=142&requisitionId=8&requisitionEntryId=10')\">DDDR</a>",
    "reqEntryServiceURLOnlyFor_Preparation": "/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&reqid=8&reqentryid=10&formAction=displayEntryStatus&performerID=&serviceid=142&requisitionId=8",
    "reqEntryServiceURLOnly": "/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&reqid=8&reqentryid=10&formAction=displayEntryStatus&performerID=&serviceid=142&requisitionId=8",
    "requisitionStatusId": 1,
    "requisitionStatus": "Preparation",
    "startedDate": "10/30/2018",
    "startedDateRaw": 1540949340000,
    "customerName": "admin admin",
    "customerId": 1,
    "customerLoginName": "admin",
    "customerOUId": 1,
    "customerEmail": "xyz@company.com",
    "reviewsCount": 0,
    "selfRating": 0,
    "isServiceOrderable": 1,
    "isServiceActive": 1,
    "isCancelable": 1,
    "momentLogicName": "servicedelivery",
    "ownerId": 1,
    "dueBy": "41 day(s) overdue",
    "renderer": "Classic",
    "orderOnBehalf": false
  }
}
```


Additional Query Parameter

An additional query parameter **pricePerUnit** has been added in the new service order form, which denotes the unit cost of the service.

Add to cart API

/RequestCenter/nsapi/catalog/v1/requisition?pricePerUnit={"__DEFAULT_PRICE_KEY":"40"}

API Method: PUT

Submit Requisition API

/RequestCenter/nsapi/catalog/v1/requisition?pricePerUnit={"__DEFAULT_PRICE_KEY":"40"}

API Method: POST

Update Requisition API

/RequestCenter/nsapi/catalog/v1/requisition?pricePerUnit={"__DEFAULT_PRICE_KEY":"40"}
API Method: PUT

Save as Draft API for Update Requisition

This API provides user to save service as a draft and while performing this action it skips all the service form field validation except Team name if the Team Management is enabled.

/RequestCenter/nsapi/catalog/v1/requisitionentry/{requisisitionentry}I/form?responseType=json&saveAsDraft=true

API Method: PUT
Sample Payload:

```
{
  "dictionaries": [
    {
      "name": ”<dictionary-name1>",
      "data": {
          ”<attribute1>” : “<value1>”,
          “<attribute2>”: “<value2>”
      }
    },
          {
      "name": ”<dictionary-name2>",
      "data": {
          ”<attribute1>” : “<value1>”,
          “<attribute2>”: “<value2>”
      }
    },
          ……
  ]
}
```

Service Item Type for SIBD Dictionary

Service Form/Requisition entry metadata API is now have an additional field called **dataTypeName** which is used for auto-data retrieval for Service Item on the Service Form. The value of this field is set to the service item type of SIBD dictionary as shown below

/RequestCenter/nsapi/catalog/v1/service/<Service_Id>/form

/RequestCenter/nsapi/catalog/v1/requisitionentry/ <requisitionentryid>/form
API Method: GET
Sample Response:

```
{
  "formPO": {
    "method": "post",
    "timeStamp": "",
     ……
    "serviceID": 180,
    "formSectionPO": [
     {
       "name": {
        "uiresource": {
          "key": "siService",
          "escapeJS": "false",
          "keyType": "preprocess"
        }
       },
       "id": "SIDict1",
       "maxRows": 0,
       "displayRows": 0,
       "gridEditable": false,
       "displayAsGrid": false,
       "autoDataRetrieval": false,
```

```
      "dataTypeName": "SitestSI01",
      "templateTypeId": "8",
      "mdrRecordID": 0,
      "formFieldPO": [
       {
        "value": "",
        "id": "SIDict.Name",
        "scale": 0,
        "maxLength": 128,
        "inputType": "text",
        "fieldDataType": "ServiceItemIdentifier",
        "hasError": false,
        "serverEditableOnly": false,
        "isMandatory": false,
        "isSecure": 0,
        "formFieldLabelPO": {
         "content": "Name"
        },
        "generateUnique": 0,
        "showInGrid": 1,
        "enableTypeahead": 0,
        "typeaheadMinStart": 0,
        "typeaheadMaxSearch": 0,
        "presentationDataType": {
         "textBox": {
          "length": "80",
          "name": "SIDict.Name",
          "value": "",
          "height": "5",
          "fieldDataType": "ServiceItemIdentifier",
          "isEditable": "true",
          "controlName": "nstextbox"
         }
        },
       }
      ……..
   ]
  }
  ]
 }
}
```


Service Item API

This is a pre-existing API which is used for auto data retrieval of service item in new service order form.

/RequestCenter/nsapi/serviceitem/{dataTypeName}/name=<serviceItemName>

API Method: GET

The {dataTypeName} is the service item type name which is got from service/requisitionentry metadata API for the related SIBD dictionary. New query param **fetchSubscription** has been added, whose value will determine if the subscription data is to be fetched.

By default subscription data will be fetched for the service item.

Expression API in Service Form

Expression API is used to calculate the value of expression.

Rest API:

/RequestCenter/nsapi/catalog/v1/expression

API Method: PUT

Request Payload


```
{
        "expression": "4.95624+5.0+29.216",
        "scale": "2",
        "fielddatatype":"Money"
}
```

Response


```
{
        "result": "39.17"
}
```


Notes:

- ‘scale’ and ‘fielddatatype’ are optional.
- Possible values for ‘fielddatatype’ are ‘Text’, ‘Money’.
- ‘scale’ value should be numeric and denotes the number of digits after the decimal point.
- ‘scale’ value is used when ‘fielddatatype’ value is ‘Money’.

Service Item Estimated Cost API

This is a pre-existing API which is used to get all the service item’s estimated price in the new service form.

/RequestCenter/nsapi/catalog/v1/service/{id}/price

DDR API

Below API needs to be called for executing DDR. For more details on this API refer to patch v6 readme content.

/RequestCenter/nsapi/catalog/v1/service/Id/rule/<encoded ruleId>

The response for this API has been enhanced to have multiple values for a key, grouped together as shown below,

API Method: GET

Sample Response


```
{
  "listOfValues": [
    [
      {
        "key": "DDRDcitionary.service",
        "value": [
          {
            "label": "service1",
            "value": "service1"
          },
          {
            "label": "service2",
            "value": "service2"
          },
          {
            "label": "service3",
            "value": "service3"
          }
        ]
      }
    ],
    [
      {
        "key": "DDRDcitionary.serviceGroup",
        "value": [
          {
            "label": "group1",
            "value": "group1"
          },
          {
            "label": "group2",
            "value": "group2"
          },
          {
            "label": "group3",
            "value": "group3"
          }
        ]
      }
    ]
  ],
  "listOfTargets": [
    "DDRDcitionary.service",
```

```
    "DDRDcitionary.serviceGroup"
  ]
}
```


API Enhancements to Service Form URL

Classic Service Form URL have been further enhanced to New Service Form which is changing the look and feel of Service Form and the way it is getting displayed.
By default classic renderer type has been set in the global parameter with in administration settings however admin user can decides what type of renderer to be set at site wide or at individual service level and further service level renderer will override the global parameter settings.
With the new technology stack when the renderer type is set to the particle form server, the new service form can be open throughout the product instead of classic service form.

Enhanced Pre-existing APIs in Service Form

Following APIs have been modified to include the field renderer for each service that was set either at Site Level or Service Level.

- To search the services by Service name

  /Requestcenter/nsapi/definition/servicedefs/?search={servicename}&searchFrom=ServiceCatalog_Home&responseType=json&recCount=30

- To get the related services for a given service id which is service item based

  /RequestCenter/nsapi/definition/servicedefs/id/{service id}/relatedservices

- To get the services which are service item based

  /RequestCenter/nsapi/definition/servicedefs/serviceitemtypes

- To get the service by service id

  /RequestCenter/nsapi/definition/servicedefs/id/{service id}

- To get the service by service name

  /RequestCenter/nsapi/definition/servicedefs/name/{service name}

- To get authorizations by view name

  /RequestCenter/nsapi/transaction/authorizations/AuthViewName={viewname}

- To get the service item subscription by service items IDs

  /RequestCenter/nsapi/serviceitems/serviceitemsubscription/PrimaryID={serviceitemids}?appendServiceData=true&appendRelatedServices=true&responseType=json&excludeServiceItemURLs=true&recordSize=20

- To get requisition entries by view name for the logged in user

  /RequestCenter/nsapi/transaction/requisitionentries/ViewName=InitiatorANDstatus=Preparation?responseType=json

Response for one of the API with field renderer and its value,

/RequestCenter/nsapi/definition/servicedefs/serviceitemtypes

API Method: GET
Sample Payload:


```
{
   "services": {
      "startRow": 1,
      "totalCount": 58,
      "recordSize": 50,
      "service": [
         {
            "serviceId": 187,
            "serviceName": "2TierContiane_rAug15(UCS)",
            "topDescription": "",
```

```
            "middleDescription": "",
            "bottomDescription": "",
            "revisionNumber": 310,
            "status": "Active",
            "statusId": 1,
            "expectedDuration": 0,
            "expectedDurationUnits": "hours",
            "canStartLater": false,
            "isBundle": false,
            "dateQualityId": 4,
            "isOrderable": true,
            "showOrderSummary": false,
            "hideInServiceCatalog": false,
            "hideInMyOrders": false,
            "maxQuantity": 0,
            "isReportable": false,
            "serviceURL": "<a href='/RequestCenter/myservices/navigate.do?query=serviceid&sid=187&layout=popup_p'
onclick=\"return GB_showFullScreen('2TierContiane_rAug15(UCS)', this.href)\">2TierContiane_rAug15(UCS)</a>",
            "serviceOrderURL": "<a href='/RequestCenter/myservices/navigate.do?query=orderform&sid=187&layout=popup_p'
onclick=\"return GB_showFullScreen('2TierContiane_rAug15(UCS)', this.href)\">Order</a>",
            "serviceURLOnly": "/RequestCenter/myservices/navigate.do?query=serviceid&sid=187",
            "serviceOrderURLOnly": "/RequestCenter/myservices/navigate.do?query=orderform&sid=187",
            "orderingMode": 2,
            "computePrice": false,
            "portalText1": "",
            "portalText2": "",
            "portalText3": "",
            "hasPortal1": false,
            "hasPortal2": false,
            "hasPortal3": false,
            "isTemplate": false,
            "isBaseService": true,
            "templateType": "null",
            "hoursPerDay": 0,
            "templateRelevant": false,
            "tenantRelevant": true,
            "renderer": "ParticleFormServer"
        }
]
}
}
```


Renderer Settings for New Service Form

Service form uses the selected renderer for loading UI component. Renderer settings can be changed by following the steps mentioned below,

- **Site Level Renderer Settings**
  Path to renderer settings at the admin level,
  Administrations > Settings > Select renderer > Classic or Particle Form Server

- **Service Level Renderer Settings**
  Path to renderer settings at the service level,
  Service designer > Select a Service in left pane > Override Renderer > Classic or Particle Form Server > Save

Newscale Properties Settings for FTL File Path

- Property name : formrule.js.template.path – Provides the relative path to the FTL template to be used for Form Rule Javascript generation
- Default Value is service-form-rules-js-templates/formrules_javascript_tpl.ftl
- Assign the corresponding path for custom FTL file templates.

Field Events in Particle Form UI

Some events are not supported for different data types in new Particle UI as shown below:-

- Date: OnFocus, OnBlur,OnClick, OnMouseover and OnMouseout
- Date Time: OnFocus, OnBlur,OnClick, OnMouseover and OnMouseout
- Person: OnFocus, OnClick, OnBlur, OnMouseover and OnMouseout
- Tenant: OnFocus, OnClick, OnBlur, OnMouseover and OnMouseout
- Radio: OnFocus, OnClick, OnBlur, OnMouseover and OnMouseout
- Checkbox: OnFocus, OnBlur, OnMouseover and OnMouseout
- Single Select: OnFocus, OnClick, OnBlur, OnMouseover and OnMouseout
- Multi Select: OnClick, OnMouseover and OnMouseout

Note:

- New Service form will not show the multiple values in a Date and Datetime fields, earlier if these fields have more than one values all used to show as separated by comma.
- DDR rules are not supported for Date and Datetime Fields

Introduced in 12.1_Patch_v6

The following table highlights the new APIs provided by this release. Detailed information on their usage as well as other feature enhancements introduced in this patch can be found in the following sections.

Table 1. List of APIs Introduced in Patch v6

nsAPI	Description	Rest URL
Fetch Service Order Form Metadata for New and Existing Orders	Provides all the required metadata to be able to render the Service Order Form	**New Order:** /RequestCenter/nsapi/catalog/v1/service/<Service_Id>/form **Existing Order:** /RequestCenter/nsapi/catalog/v1/requisitionentry/ <requisitionentryid>/form
Execute DDR Rules	API to execute the DDR rule for the browser based events	**API to execute the DDR rule:** /RequestCenter/nsapi/catalog/v1/service/Id/rule/<encoded ruleId> **Sample request and response for the DDR:** /RequestCenter/nsapi/catalog/v1/service/169/rule/RJZi0%2FXysRe6RdF%2BH1NWcQ%3D%3D
Service Order Form Action APIs	Submits a requisition, adds to cart, save service alone as a draft & service as draft including services in the Draft	**Submit Requisition:** /RequestCenter/nsapi/catalog/v1/requisition **Add to cart:** /RequestCenter/nsapi/catalog/v1/requisition **Save Service Alone as a Draft:** /RequestCenter/nsapi/catalog/v1/requisition?saveAsDraft=true **Save Service as Draft including services in the Draft:** /RequestCenter/nsapi/catalog/v1/requisition?saveAsDraft=true
Export/Import Multiple Services	Takes a list of Service IDs and exports/imports the Service details in to an xml	**Export of multiple ServiceIds namely S1=241, S2=242 & S3=243:** /RequestCenter/nsapi/rex/v1/definition/service?id=243&id=242&id=241&includeContainedServices=true **Import the exported XML:** /RequestCenter/nsapi/rex/v1/definition/service

Get Service/User Context	provides context parameters which are used for loading the service order form	/RequestCenter/nsapi/catalog/v1/service/150/orderformcontext
Update Requisition Entry	Used for updating the Requisition Entry pertaining to a Requisition	/RequestCenter/nsapi/catalog/v1/requisitionentry/217/form
Calculate Estimated Cost for Service Based on Service Item Billing	Provides all Service Item's estimated price for a given service form data	/RequestCenter/nsapi/catalog/v1/service/152/price

API to get Service Order Form Metadata for New Order

This API provides all the required metadata to be able to render the Service Order Form. The metadata includes all the dictionaries with the form fields and their display properties, the generated Java script for the CR and DDR rules and the global, Service specific JavaScripts

Pre-Requisite – Before executing the API below parameters needs to be update in the newscale properties.

##

##Particle form rules js template path: service-form-rules-js-templates/formrules_javascript_tpl.ftl

formrule.js.template.path=service-form-rules-js-templates/formrules_javascript_tpl.ftl

##

All the form details content can GET from the below API and response will be in JSON

- API – GET Method –
  End point - http://<ServerURL>: Port/RequestCenter/nsapi/catalog/v1/service/<Service_Id>/form

Header Parameters

- Content-Type - application/json
- Accept - application/json

RBAC Access – User who have the order permission can execute this API.

Response of the API is as below –


```
{
        "formPO": {
                "method": "post",
                "timeStamp": "",
                "pages": "[{\"description\":\"Tenant Information Form for when Tenant Management is enabled\",\"name\":\"Tenant-Information Form\",\"sections\":[\"XX_Tenant_Information_XX\"]},{\"description\":\"\",\"name\":\"Attach Volume\",\"sections\":[\"Volume_Action\",\"VolumeInfo_RO_Action\",\"VolumeInfo_Action\"]}]",
                "orderingMode": 3,
                "serviceID": 136,
          "javascriptFunctionDefinationsPO": {},
          "formSectionsPO": []
           "showCurrencySymbol": "1",
                "langCode": "USEN",
                "monitorPO": [],
                "jsResourceExists": "0",
                "monitorPOCount": 0,
                "isFormReadOnly": false,
                "javascriptLibraryPO": [],
                "currencyLangCode": "USEN",
                "formSectionPOCount": 4,
                "hasServerError": false,
                "javascriptLibraryPOCount": 0,
          "errorMessagePO":{}
          "javascriptCallPOCount": 1,
          "javascriptCallPO": []
```

```
  }
}
```

Response Payload description

- **javascriptFunctionDefinitionsPO –** This body contain the Java scripts, DDR and CR consumed by the services.
- **formPO-** Is the main attribute in the payload. Under this payload all the form content details will be displayed as metadata format.
- **Method:** Post method to submit form.
- **Pages:** This contains all the service forms name which are associated to a service.
- **FormSectionPO-** Contains all the General and Display Properties contents.
- **Key –** Caption of the service form
- **formFieldPO –** All the form filed level content will be displayed under this body
- **Id –** Name of the dictionary
- **Value –** Default Value will be displayed
- **maxLength –** Length of the filed (defined when creating a dictionary)
- **fieldDataType –** Data type of the filed (defined when creating a dictionary)
- **inputType –** It is type of the field (such as select single, select multiple, radio , check box etc)
- **sliderType-** If the input type is slider , then slider type will be range
- **Scale –** Value will displayed when it is slider and range is set
- **validatorPO-** This value set when validate range set to true, minimum and maximum values will be defined in the same body.
- **GenerateUnique –** Set to zero if the Generate unique value is unchecked and if it is set to one when it is checked and when it is set to 1 Value filed will replace the default value with unique value generated by system
- **instructionalText-** It displays the help text of the field
- **ValidatiorPOCount-** This is count of validators. Example when validate range is checked, Mandatory is checked, Minimum/Maximum values are provided. Count set to 3, if one is unchecked then value set 2 and so on.
- **formFieldLabelPO – content -** This contains the label and advance formatting details of the files
- **associatedControlPO-** This contains the details of the add a button, Button Text, URL and Send data
- **showInGrid –** Set 0 if it is not grid, if it is grid value will be set as 1
- **isMandatory –** Set to true when it is mandatory if not it will be set to false
- **isHidden –** Set to true when it is hidden if not it will be set to false
- **errorMessagePO –** This contains the generic error message for DDR failure while loading the form. Ex : while loading service form , if the DDR execution message will be displayed for the user “Error executing webservice DDR DemoDDR. Please contact your system administrator”. same message will be stored in this body
- **javascriptCallPO –** This content the function name , which is consumed by the service and also contains events (onload, submit) and if the function is having some arguments. Same arguments it will be contained by this body.

Example for API Response


```
{
        "formPO": {
                "method": "post",
                "timeStamp": "",
                "pages": "[{\"description\":\"Tenant Information Form for when Tenant Management is
enabled\",\"name\":\"Tenant-Information
Form\",\"sections\":[\"XX_Tenant_Information_XX\"]},{\"description\":\"\",\"name\":\"demo_SF\",\"sections\":[\"demo_SF\",\"demo_Gri
dSF\"]}]",
                "serviceID": 240,
                "orderingMode": 3,
                "showCurrencySymbol": "1",
                "javascriptFunctionDefinitionsPO": {
                        "content": "\n function
checkOrderingResources(){\n\n//=============================================================================\n/
/    RETRIEVE GLOBAL SERVICE OFFERING
OPTIONS\n//=============================================================================\n\n\t\tfunction
```

```
retrieveGlobalServiceOptions() {\n\n\t\t\tvar getUrl = '/RequestCenter/nsapi/serviceitem/SiSetGlobalServiceOfferings';\n\n\t\t\tif
(window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari\n\t\t\t\txmlhttp = new XMLHttpRequest();\n\t\t\t} else {//
code for IE6, IE5\n\t\t\t\txmlhttp = new ActiveXObject(\"Microsoft.XMLHTTP\");\n\t\t\t}\n\t\t\txmlhttp.open(\"GET\", getUrl,
false);\n\t\t\txmlhttp.send();\n\t\t\txmlDoc = xmlhttp.responseXML;\n\n\t\t\tvar pid = new Array();\n\n\t\t\tif (xmlhttp.readyState == 4 &&
xmlhttp.status == 200) {\n\n\t\t\t\t// Optional Customer Message\n\t\t\t\tpid[0] =
xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[4].childNodes[0].nodeValue;\n\t\t\t\t//alert(pid[0]);\n\n\t\t\t\t// Physical
Server Ordering\n\t\t\t\tpid[1] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[2].childNodes[0].nodeValue;\n\n\t\t\t\t//
VDC Ordering\n\t\t\t\tpid[2] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[3].childNodes[0].nodeValue;\n\n\t\t\t\t//
Virtual Server from Template\n\t\t\t\tpid[3] =
xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[5].childNodes[0].nodeValue;\n\n\t\t\t\t// Virtual Server and Install
OS\n\t\t\t\tpid[4] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[6].childNodes[0].nodeValue;\n\n\t\t\t\t//
ESXiProvisioning\n\t\t\t\tpid[5] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[7].childNodes[0].nodeValue;\n\n\t\t\t\t//
SharedZone\n\t\t\t\tpid[6] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[8].childNodes[0].nodeValue;\n\n\t\t\t\t//
Snapshot Max\n\t\t\t\tpid[7] = xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[9].childNodes[0].nodeValue;\n\n\t\t\t\t//
VMTemp Snapshot\n\t\t\t\tpid[9] =
xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[10].childNodes[0].nodeValue;\n\n\t\t\t\t// VMOS Snapshot\n\t\t\t\tpid[9] =
xmlDoc.getElementsByTagName(\"serviceItemAttribute\")[11].childNodes[0].nodeValue;\n\n\n\t\t\t}\n\n\t\treturn
pid;\n\n\t}\n\n\n\n//============================================================================\n// RETRIEVE
SERVICE ITEM
TOTALS\n//============================================================================\n\nfunction
getSITotalCount(getUrl){\n\nif (window.XMLHttpRequest)\n  {// code for IE7+, Firefox, Chrome, Opera, Safari\n  xmlhttp=new
XMLHttpRequest();\n  }\nelse\n  {// code for IE6, IE5\n  xmlhttp=new ActiveXObject(\"Microsoft.XMLHTTP\");\n
}\nxmlhttp.open(\"GET\",getUrl,false);\nxmlhttp.send();\nxmlDoc=xmlhttp.responseXML;\n\nif(xmlhttp.readyState==4 &&
xmlhttp.status==200){\n\nvar totalCount = xmlDoc.getElementsByTagName(\"serviceitem\")[0].getAttribute(\"totalCount\");\n\n} else
{\n\nvar totalCount = 0;\n\n}\n\nreturn
totalCount;\n\n}\n\n\n//============================================================================\n//
RETRIEVE SERVICE ITEM TABLE SERVICE ITEMS
TOTAL\n//============================================================================\n\nfunction
getSITotal(siName,siType){\n\n//var
vmURL='/RequestCenter/nsapi/serviceitems/serviceitemsubscription/ServiceItemTypeName=Virtual%20Machine%20(VM)';\nvar
countURL='/RequestCenter/nsapi/serviceitem/'+siName;\n\nif(siType && siType != 'Registered') countURL=
countURL+'/Status='+siType;\n\nreturn getSITotalCount(countURL);\n\n}\n\nvar gOffering = retrieveGlobalServiceOptions();\nvar
vcenterTotal = getSITotal('SiVirtualCenterSever');\nvar ucsTotal = getSITotal('SiCiscoUCSManagerElement');\nvar cspTotal =
getSITotal('SiCiscoSPPlatformElement');\n\n//============================================================================
========\n// CHECK VM TEMPLATE
ORDERING\n//============================================================================\n\nif(document.get
ElementById('SIBDChangeSystemWideServiceOptions.VMTemplate.Message')){\n\n//==============================
==========================================\n//  NOW FOR THE MESSAGE
DISPLAY\n//============================================================================\nvar
vmTempMessage = document.getElementById('SIBDChangeSystemWideServiceOptions.VMTemplate.Message');\n\nvar
vmTempYesId = document.getElementById('SIBDChangeSystemWideServiceOptions.VMTemplate::1');\nvar vmTempNoId=
document.getElementById('SIBDChangeSystemWideServiceOptions.VMTemplate::2');\n\nif(document.getElementById('vmTempMa
xSnap')){\ndocument.getElementById('vmTempMaxSnap').innerHTML = '';
\ndocument.getElementById('vmTempMaxSnap').innerHTML = gOffering[7];\n}\n\n\nvar vmSnapVal =
document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Message_INFO').innerHTML;\n\nvar
vmSnapYes = document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot::1');\nvar vmSnapNo =
document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot::2');\n\nvar yesSelect = 'checked';\nvar noSelect
= '';\n\nif(vmSnapNo.checked){\n\n var yesSelect = '';\n var noSelect = 'checked';\n \n }\n\n var createChkBox =
'<br>   <input type=\"radio\" id=\"SIBDChangeSystemWideServiceOptions.VMSnapshot::1\"
name=\"SIBDChangeSystemWideServiceOptions.VMSnapshot\" value=\"Yes\" style=\"vertical-align: middle;\" fielddatatype=\"Text\"
isfhook=\"1\" onfocus=\"cc_f(this,event);\" onblur=\"cc_f(this,event);\" nalign=\"horizontal\" onchange=\"cc_f(this,event);\"
'+yesSelect+'> Yes   <input type=\"radio\" id=\"SIBDChangeSystemWideServiceOptions.VMSnapshot::2\"
name=\"SIBDChangeSystemWideServiceOptions.VMSnapshot\" value=\"No\" style=\"vertical-align: middle;\" fielddatatype=\"Text\"
isfhook=\"1\" onfocus=\"cc_f(this,event);\" onblur=\"cc_f(this,event);\" nalign=\"horizontal\" onchange=\"cc_f(this,event);\"
'+noSelect+'> No';\n \ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Input').innerHTML =
'';\n\ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Input').innerHTML = vmSnapVal
+createChkBox;\n\n//document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Message_INFO').innerHTM
L = '';\ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Message_INFO').style.color =
'#ffffff';\n\nif(vmTempNoId.checked){\n\nvar colorFont = '#888888';\nvar vcenterAddMsg ='VMware vCenter Element Not
Required';\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row'))
document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row').style.display = 'none';\n\n} else
{\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row'))
```

document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row').style.display = 'table-row';\n\nvar colorFont = '#66CC33';\nvar colorFont = '#669900';\n\nvar vcenterAddMsg ='VMware vCenter Element Available';\n\nif(vcenterTotal < 1){\n\nvar colorFont = '#B40000';\n\nvar vcenterAddMsg ='VMware vCenter Element Required';\n\n}\n\n}\n\n var vcenterMessage = '<span style=\"color: '+colorFont+';\">'+vcenterAddMsg+'';\n\nvmTempMessage.innerHTML = vcenterMessage;\n\n}\n\n\n\n\n\n//==========\n// CHECK VIRTUAL OS ORDERING\n//==========\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.VMOS.Message')){\n\n//==========\n// NOW FOR THE MESSAGE DISPLAY\n//==========\nvar vmOSMessage = document.getElementById('SIBDChangeSystemWideServiceOptions.VMOS.Message');\n\nvar vmOSYesId = document.getElementById('SIBDChangeSystemWideServiceOptions.VMOS::1');\nvar vmOSNoId= document.getElementById('SIBDChangeSystemWideServiceOptions.VMOS::2');\n\nif(document.getElementById('vmOSMaxSnap')) {\ndocument.getElementById('vmOSMaxSnap').innerHTML = ''; \ndocument.getElementById('vmOSMaxSnap').innerHTML = gOffering[7];\n}\n\n\n\nvar vmOSSnapVal = document.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Message_INFO').innerHTML;\n\nvar vmOSSnapYes = document.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot::1');\nvar vmOSSnapNo = document.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot::2');\n\nvar yesSelect = 'checked';\nvar noSelect = '';\n\nif(vmOSSnapNo.checked){\n\n var yesSelect = '';\n var noSelect = 'checked';\n \n }\n\n var createChkBox = '
   <input type=\"radio\" id=\"SIBDChangeSystemWideServiceOptions.VMOSSnapshot::1\" name=\"SIBDChangeSystemWideServiceOptions.VMOSSnapshot\" value=\"Yes\" style=\"vertical-align: middle;\" fielddatatype=\"Text\" isfhook=\"1\" onfocus=\"cc_f(this,event);\" onblur=\"cc_f(this,event);\" nalign=\"horizontal\" onchange=\"cc_f(this,event);\" '+yesSelect+'> Yes   <input type=\"radio\" id=\"SIBDChangeSystemWideServiceOptions.VMOSSnapshot::2\" name=\"SIBDChangeSystemWideServiceOptions.VMOSSnapshot\" value=\"No\" style=\"vertical-align: middle;\" fielddatatype=\"Text\" isfhook=\"1\" onfocus=\"cc_f(this,event);\" onblur=\"cc_f(this,event);\" nalign=\"horizontal\" onchange=\"cc_f(this,event);\" '+noSelect+'> No';\n\n\ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Input').innerHTML = '';\n\n\ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Input').innerHTML = vmOSSnapVal +createChkBox;\n\n\ndocument.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Message_INFO').style.color = '#ffffff';\n\n\nif(vmOSNoId.checked){\n\nvar colorFont = '#888888';\n\nvar cspAddMsg ='Cisco SP Element Not Required';\nvar ucsAddMsg ='UCS Manager Element Not Required';\n\n\nvar cspMessage = '<span style=\"color: '+colorFont+';\">'+cspAddMsg+'';\nvar ucsMessage = '<span style=\"color: '+colorFont+';\">'+ucsAddMsg+'';\n\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row')) document.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Row').style.display = 'none';\n\n\n} else {\n\nvar colorFont = '#66CC33';\nvar colorFont = '#669900';\nvar cspAddMsg ='Cisco SP Element Available';\n\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.VMSnapshot.Row')) document.getElementById('SIBDChangeSystemWideServiceOptions.VMOSSnapshot.Row').style.display = 'table-row';\n\n\nif(cspTotal < 1){\n\nvar colorFont = '#B40000';\n\nvar cspAddMsg ='Cisco SP Element Required';\n\n}\n\nvar cspMessage = '<span style=\"color: '+colorFont+';\">'+cspAddMsg+'';\n \nvar colorFont = '#66CC33';\nvar colorFont = '#669900';\nvar ucsAddMsg ='UCS Manager Element Available';\n\n\nif(ucsTotal < 1){\n\nvar colorFont = '#B40000';\nvar ucsAddMsg ='UCS Manager Element Required';\n}\n\nvar ucsMessage = '<span style=\"color: '+colorFont+';\">'+ucsAddMsg+'';\n\n\n}\n\nvmOSMessage.innerHTML = ucsMessage+cspMessage;\n\n\n}\n\n\n\n//==========\n// CHECK PHYSICAL SERVER ORDERING\n//==========\n\nif(document.getElementById('SIBDChangeSystemWideServiceOptions.PhysicalServerOrdering.Message')){\n\n//==========\n// NOW FOR THE MESSAGE DISPLAY\n//==========\nvar physMessage = document.getElementById('SIBDChangeSystemWideServiceOptions.PhysicalServerOrdering.Message');\n\nvar physYesId = document.getElementById('SIBDChangeSystemWideServiceOptions.PhysicalServerOrdering::1');\nvar physNoId= document.getElementById('SIBDChangeSystemWideServiceOptions.PhysicalServerOrdering::2');\n\nif(physNoId.checked){\n\nvar colorFont = '#888888';\n\nvar cspAddMsg ='Cisco SP Element Not Required';\nvar ucsAddMsg ='UCS Manager Element Not Required';\n\n\nvar cspMessage = '<span style=\"color: '+colorFont+';\">'+cspAddMsg+'';\nvar ucsMessage = '<span style=\"color: '+colorFont+';\">'+ucsAddMsg+'';\n\n\n} else {\n\nvar colorFont = '#66CC33';\nvar colorFont = '#669900';\nvar cspAddMsg ='Cisco SP Element Available';\n\n\nif(cspTotal < 1){\n\nvar colorFont = '#B40000';\n\nvar cspAddMsg ='Cisco SP Element Required';\n\n}\n\nvar cspMessage = '<span style=\"color: '+colorFont+';\">'+cspAddMsg+'';\n\nvar colorFont = '#66CC33';\nvar colorFont = '#669900';\nvar ucsAddMsg ='UCS Manager Element Available';\n\n\nif(ucsTotal < 1){\n\nvar colorFont = '#B40000';\nvar ucsAddMsg ='UCS Manager Element Required';\n}\n\nvar ucsMessage = '<span style=\"color: '+colorFont+';\">'+ucsAddMsg+'';\n\n\n }\n\nphysMessage.innerHTML = ucsMessage+cspMessage;\n\n\n}\n\n\n\n\n//==========\n// CHECK ESXi

PROVISIONING\n//==========\n\nif(document
.getElementById('SIBDChangeSystemWideServiceOptions.ESXiProvisioning.Message')){\n\n//==========
==========\n// NOW FOR THE MESSAGE
DISPLAY\n//==========\nvar esxiMessage =
document.getElementById('SIBDChangeSystemWideServiceOptions.ESXiProvisioning.Message');\n\nvar esxiYesId =
document.getElementById('SIBDChangeSystemWideServiceOptions.ESXiProvisioning::1');\nvar esxiNoId=
document.getElementById('SIBDChangeSystemWideServiceOptions.ESXiProvisioning::2');\n\nif(esxiNoId.checked){\n\nvar
colorFont = '#888888';\nvar vcenterAddMsg ='VMware vCenter Element Not Required';\n\n} else {\n\nvar colorFont =
'#66CC33';\nvar colorFont = '#669900';\n\nvar vcenterAddMsg ='VMware vCenter Element Available';\n\nif(vcenterTotal < 1){\n\nvar
colorFont = '#B40000';\n\nvar vcenterAddMsg ='VMware vCenter Element Required';\n\n}\n\n}\n\nvar vcenterMessage = '<span
style=\"color: '+colorFont+';\">'+vcenterAddMsg+'';\n\nesxiMessage.innerHTML =
vcenterMessage;\n\n}\n\n\n\n//==========\n//
CHECK SHARED ZONE
ORDERING\n//==========\n\nif(document.get
ElementById('SIBDChangeSystemWideServiceOptions.SharedZone.Message')){\n\n//==========
==========\n// NOW FOR THE MESSAGE
DISPLAY\n//==========\nvar
sharedZoneMessage = document.getElementById('SIBDChangeSystemWideServiceOptions.SharedZone.Message');\n\nvar
szYesId = document.getElementById('SIBDChangeSystemWideServiceOptions.SharedZone::1');\nvar szNoId=
document.getElementById('SIBDChangeSystemWideServiceOptions.SharedZone::2');\n\nif(szNoId.checked){\n\nvar colorFont =
'#888888';\nvar vcenterAddMsg ='VMware vCenter Element Not Required';\n\n} else {\n\nvar colorFont = '#66CC33';\nvar colorFont
= '#669900';\n\nvar vcenterAddMsg ='VMware vCenter Element Available';\n\nif(vcenterTotal < 1){\n\nvar colorFont =
'#B40000';\n\nvar vcenterAddMsg ='VMware vCenter Element Required';\n\n}\n\n}\n\nvar vcenterMessage = '<span style=\"color:
'+colorFont+';\">'+vcenterAddMsg+'';\n\nsharedZoneMessage.innerHTML =
vcenterMessage;\n\n}\n\n\n\n//==========\n//
CHECK VDC
ORDERING\n//==========\n\nif(document.get
ElementById('SIBDChangeSystemWideServiceOptions.VDCOrdering.Message')){\n\n//==========
==========\n// NOW FOR THE MESSAGE
DISPLAY\n//==========\nvar vdcMessage =
document.getElementById('SIBDChangeSystemWideServiceOptions.VDCOrdering.Message');\n\nvar vdcYesId =
document.getElementById('SIBDChangeSystemWideServiceOptions.VDCOrdering::1');\nvar vdcNoId=
document.getElementById('SIBDChangeSystemWideServiceOptions.VDCOrdering::2');\n\nif(vdcNoId.checked){\n\nvar colorFont =
'#888888';\nvar vcenterAddMsg ='VMware vCenter Element Not Required';\n\n} else {\n\nvar colorFont = '#66CC33';\nvar colorFont
= '#669900';\n\nvar vcenterAddMsg ='VMware vCenter Element Available';\n\nif(vcenterTotal < 1){\n\nvar colorFont =
'#B40000';\n\nvar vcenterAddMsg ='VMware vCenter Element Required';\n\n}\n\n}\n\nvar vcenterMessage = '<span style=\"color:
'+colorFont+';\">'+vcenterAddMsg+'';\n\nvdcMessage.innerHTML =
vcenterMessage;\n\n}\n\n\n\n//==========\n//
SHOW / HIDE SUBMIT AND RESET
BUTTONS\n//==========\n\nif(document.getEl
ementById('SIBDChangeSystemWideServiceOptions.VDCOrdering.Message') &&
document.getElementById('SIBDChangeSystemWideServiceOptions.PhysicalServerOrdering.Message')){\n\ndocument.getElement
ById('cheader').style.visibility=\"visible\";\ndocument.getElementById('cfooter').style.visibility=\"visible\";\n\n//alert('CSP: '+cspTotal+'
| UCS: '+ucsTotal+' | vCenter: '+vcenterTotal+' | p: '+p);\n\nif((vmOSYesId.checked && (cspTotal == 0 || ucsTotal == 0)) ||
(physYesId.checked && (cspTotal == 0 || ucsTotal == 0)) || ((vmTempYesId.checked || esxiYesId.checked || szYesId.checked ||
vdcYesId.checked) && vcenterTotal == 0)
){\n\ndocument.getElementById('cheader').style.visibility=\"hidden\";\ndocument.getElementById('cfooter').style.visibility=\"hidden\";\
n\n}\n\n}\n\n}\n\n function Allstate_Common_OnSubmit()\n{\n ProcessFormOnSubmit();\n}\n\n//This is for Form ID 2 \n//f11112 is
for rule: ShowNameField \nvar f11112 = function f11112(url, divName, source, eventCopy, index, arr) { var local_rule_id =
encodeURIComponent(\"WcHmG8SJ/H1yYWnT13lmbg==\"); \n   \n var result = true; \n // it is a conditional rule \n // loop thro the
conditions list \n // loop thro the actions list \n if (\n   ((eventCopy && eventCopy.srcElement && eventCopy.srcElement.name
== 'XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX' && getValueFromIsfCode(eventCopy.srcElement) ==
getProperValueForLiteral('1',true,serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX)) || (!(eventCopy &&
eventCopy.srcElement && eventCopy.srcElement.name == 'XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX') &&
serviceForm.XX_Tenant_Information_XX && serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX &&
getProperValueForField(getSafeValue(serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX), true,
serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX) ==
getProperValueForLiteral('1',true,serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX))) \n   ) \n   { \n   var
actions = []; \n   var actionValues = []; \n   actionValues[0] = resolveISFNamespace(\"\", \"#\", true, true); \n   actions[0] =
function() {if (serviceForm.XX_Tenant_Information_XX && serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX)
serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX.setVisible(true);}; \n   actionValues[1] =
resolveISFNamespace(\"\", \"#\", true, true); \n   actions[1] = function() {if (serviceForm.XX_Tenant_Information_XX &&

serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX) serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX.setMandatory(true);}; \n result = runActions(actions, source, eventCopy, index, arr); \n } \n return result; \n} \n//f11113 is for rule: HideNameField \nvar f11113 = function f11113(url, divName, source, eventCopy, index, arr) { var local_rule_id = encodeURIComponent(\"wgfRPAlcqeZh3jpbZm7wLQ==\"); \n \n var result = true; \n // it is a conditional rule \n // loop thro the conditions list \n // loop thro the actions list \n if (\n ((eventCopy && eventCopy.srcElement && eventCopy.srcElement.name == 'XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX' && getValueFromIsfCode(eventCopy.srcElement) == getProperValueForLiteral('0',true,serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX)) || (!(eventCopy && eventCopy.srcElement && eventCopy.srcElement.name == 'XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX') && serviceForm.XX_Tenant_Information_XX && serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX && getProperValueForField(getSafeValue(serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX), true, serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX) == getProperValueForLiteral('0',true,serviceForm.XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX))) \n) \n { \n var actions = []; \n var actionValues = []; \n actionValues[0] = resolveISFNamespace(\"\", \"#\", true, true); \n actions[0] = function() {if (serviceForm.XX_Tenant_Information_XX && serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX) serviceForm.XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX.setMandatory(false);}; \n actionValues[1] = resolveISFNamespace(\"\", \"#\", true, true); \n actions[1] = function() {if (serviceForm.XX_Tenant_Information_XX) serviceForm.XX_Tenant_Information_XX.setVisible(false);}; \n result = runActions(actions, source, eventCopy, index, arr); \n } \n return result; \n} \n \n//type_of_method_array will hold what type of rule it is. This is needed for DDR callbacks and to know that the \n// \"checkAndFireNext\" method needs to be called after a CR. \n// could have been done as part of the previous rule list loop, doing it here for easier understanding \nif (!type_of_method_array) \n type_of_method_array = new Object(); \n type_of_method_array[f11112] = \"CR\"; \n type_of_method_array[f11113] = \"CR\"; \n \n// We know from the event to rules association table, the following rules get fired \n// in a specific order for a specific event \nif (onload_on_serviceForm) \n onload_on_serviceForm.push(f11112); \nelse { \n var onload_on_serviceForm = new Array; \n onload_on_serviceForm.push(f11112); \n} \nif (onload_on_serviceForm) \n onload_on_serviceForm.push(f11113); \nelse { \n var onload_on_serviceForm = new Array; \n onload_on_serviceForm.push(f11113); \n} \n \n// the following is the event to the rules map \nif (! event_to_rules_map) \n{ \n alert(\"the event_to_rules_map is not defined\"); \n event_to_rules_map = new Object(); \n} \n \nevent_to_rules_map[\"onload_serviceForm\"] = onload_on_serviceForm; \nevent_to_rules_map[\"onload_serviceForm\"] = onload_on_serviceForm; \n \n// \n \nvar formEventBeenThere = {}; \n \n// \n \n//This is for Form ID 157 \n//f11116 is for rule: DDRUSERNAME \nvar f11116 = function f11116(url, divName, source, eventCopy, index, arr) { var local_rule_id = encodeURIComponent(\"ZBUI1h+RqWFdYG/IAYr8AQ==\"); \n \n var result = true; \n var filters = new Array(); \n var initiatorFilter = new Object(); \n initiatorFilter.key = \"Requisition.InitiatorID\"; \n initiatorFilter.value = ReqInitiatorID; \n filters.push(initiatorFilter); \n \n var customerFilter = new Object(); \n customerFilter.key = \"Requisition.CustomerID\"; \n customerFilter.value = ReqCustomerID; \n filters.push(customerFilter); \n \n var serviceIdFilter = new Object(); \n serviceIdFilter.key = \"ServiceForm.ServiceId\"; \n serviceIdFilter.value = ServiceID; \n filters.push(serviceIdFilter); \n \n var filter_json = \"\"; \n \n if(filters.length > 0) \n filter_json = JSON.stringify(filters); \n \n filter_json = encodeURIComponent(filter_json); \n \n var qStr = \"formAction=executeFormRule&ruleParameters=\" + filter_json + \"&ruleId=\" + local_rule_id;\t \n\tcommonDDRCall(localBasePath+\"myservices/executeFormRules.do\", null, source, eventCopy, index, arr, 11116, qStr); \n return result; \n} \n//f11117 is for rule: CRPSCUSERS \nvar f11117 = function f11117(url, divName, source, eventCopy, index, arr) { var local_rule_id = encodeURIComponent(\"xFceFrjivgz95Mwt/nb4zw==\"); \n \n var result = true; \n // it is a conditional rule \n // loop thro the conditions list \n // loop thro the actions list \n if (\n true \n) \n { \n var actions = []; \n var actionValues = []; \n actionValues[0] = resolveISFNamespace(\"PSCUSER\", \"#\", true, true); \n actions[0] = function() {if (serviceForm.demo_SF && serviceForm.demo_SF.pscusers) serviceForm.demo_SF.pscusers.setValueByCRCheckError(eventCopy, 'PSCUSER', 'PSCUSER', false);}; \n result = runActions(actions, source, eventCopy, index, arr); \n } \n return result; \n} \n \n//type_of_method_array will hold what type of rule it is. This is needed for DDR callbacks and to know that the \n// \"checkAndFireNext\" method needs to be called after a CR. \n// could have been done as part of the previous rule list loop, doing it here for easier understanding \nif (!type_of_method_array) \n type_of_method_array = new Object(); \n type_of_method_array[f11116] = \"DDR\"; \n type_of_method_array[f11117] = \"CR\"; \n \n// We know from the event to rules association table, the following rules get fired \n// in a specific order for a specific event \nif (onsubmit_on_serviceForm) \n onsubmit_on_serviceForm.push(f11117); \nelse { \n var onsubmit_on_serviceForm = new Array; \n onsubmit_on_serviceForm.push(f11117); \n} \nif (onload_on_serviceForm) \n onload_on_serviceForm.push(f11116); \nelse { \n var onload_on_serviceForm = new Array; \n onload_on_serviceForm.push(f11116); \n} \nif (onload_on_serviceForm) \n onload_on_serviceForm.push(f11117); \nelse { \n var onload_on_serviceForm = new Array; \n onload_on_serviceForm.push(f11117); \n} \nif (onunload_on_serviceForm) \n onunload_on_serviceForm.push(f11117); \nelse { \n var onunload_on_serviceForm = new Array; \n onunload_on_serviceForm.push(f11117); \n} \n \n// the following is the event to the rules map \nif (! event_to_rules_map) \n{ \n alert(\"the event_to_rules_map is not defined\"); \n event_to_rules_map = new Object(); \n} \n \nevent_to_rules_map[\"onsubmit_serviceForm\"] = onsubmit_on_serviceForm; \nevent_to_rules_map[\"onload_serviceForm\"] = onload_on_serviceForm; \nevent_to_rules_map[\"onload_serviceForm\"] = onload_on_serviceForm; \nevent_to_rules_map[\"onunload_serviceForm\"] = onunload_on_serviceForm; \n \n// \n \nvar formEventBeenThere = {}; \n \n// \n \n\n//# sourceURL=jsstream.js\n"

```
},
"formSectionPO": [
    {
        "name": {
            "uiresource": {
                "key": "Select team",
                "keyType": "preprocess",
                "escapeJS": "false"
            }
        },
        "id": "XX_Tenant_Information_XX",
        "autoDataRetrieval": false,
        "templateTypeId": "1",
        "maxRows": 0,
        "mdrRecordID": 0,
        "displayRows": 0,
        "formFieldPO": [
            {
                "value": "1",
                "id": "XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX",
                "scale": 0,
                "maxLength": 512,
                "inputType": "hidden",
                "fieldDataType": "Text",
                "sliderType": "basic",
                "formFieldLabelPO": {
                    "content": "XX_XX_TMEnabled_XX_XX"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                    "hidden": {
                        "name":
"XX_Tenant_Information_XX.XX_XX_TMEnabled_XX_XX",
                        "value": "1",
                        "fieldDataType": "Text",
                        "controlName": "nshidden",
                        "isEditable": "true"
```

```
                    }
                },
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 0,
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                    "uiresourceCount": 0,
                    "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": true
            },
            {
                "value": "TeamVspancha",
                "id": "XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX",
                "scale": 0,
                "maxLength": 100,
                "inputType": "text",
                "fieldDataType": "Tenant",
                "sliderType": "basic",
                "formFieldLabelPO": {
                    "content": "TeamName"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "instructionalText": "Please select the tenant to which you want to order this
service for.",
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                    "tenant": {
                        "length": "80",
                        "name":
"XX_Tenant_Information_XX.XX_XX_TeamName_XX_XX",
```

```
                                        "value": "TeamVspancha",
                                        "fieldDataType": "Tenant",
                                        "controlName": "nstenant",
                                        "idValue": "",
                                        "isEditable": "true"
                                }
                        },
                        "typeaheadMinStart": 0,
                        "isSecure": 0,
                        "isMandatory": false,
                        "showInGrid": 0,
                        "hasError": false,
                        "javascriptCallPOCount": 0,
                        "errorMessagePO": {
                                "uiresourceCount": 0,
                                "uiresource": []
                        },
                        "validatorPOCount": 0,
                        "javascriptCallPO": [],
                        "validatorPO": [],
                        "isEditable": true,
                        "isHidden": false
                }
        ],
        "gridEditable": false,
        "displayAsGrid": false,
        "formFieldPOCount": 2,
        "dictionaryAccess": true,
        "isVisible": true
},
{
        "name": {
                "uiresource": {
                        "key": "demo_SF",
                        "keyType": "preprocess",
                        "escapeJS": "false"
                }
        },
        "id": "demo_SF",
        "autoDataRetrieval": false,
```

```
"templateTypeId": "1",
"maxRows": 0,
"mdrRecordID": 0,
"displayRows": 0,
"formFieldPO": [
        {
                "value": "Please enter the user name",
                "id": "demo_SF.Name",
                "scale": 0,
                "maxLength": 50,
                "inputType": "text",
                "fieldDataType": "Text",
                "sliderType": "basic",
                "formFieldLabelPO": {
                        "content": "Name"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "instructionalText": "Help Text",
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "50",
                                "name": "demo_SF.Name",
                                "value": "Please enter the user name",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
                                "isEditable": "true"
                        }
                },
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": true,
                "showInGrid": 0,
                "hasError": false,
                "associatedControlPO": {
                        "url":
"http://10.78.0.247:8080/RequestCenter/organizationdesigner/selectperson.do?fromPage=ABC&search_pattern=&firstName=&last
```

Name=&p_data=customerId&p_disp=customerName&use_ou=true&o_data=orgUnitId&o_disp=costAssignedToBU&retfunc=setManager",
"label": "Select Person",
"sendData": true
},
"javascriptCallPOCount": 0,
"errorMessagePO": {
"uiresourceCount": 0,
"uiresource": []
},
"validatorPOCount": 3,
"javascriptCallPO": [],
"validatorPO": [
{
"type": "null",
"param": "true",
"validatorMessagePO": {
"uiresource": {
"key": "common.alertmsgs.plsEnterAppropriateInfo",
"keyType": "friendly",
"escapeJS": "false"
}
}
},
{
"type": "maximum",
"param": "10",
"validatorMessagePO": {
"uiresource": {
"key": "common.alertmsgs.plsEnterValueLessThan",
"keyType": "friendly",
"escapeJS": "false"
}
}
},
{
"type": "minimum",
"param": "1",
"validatorMessagePO": {

```
                                                    "uiresource": {
                                                        "key":
"common.alertmsgs.plsEnterValueGreaterThan",
                                                        "keyType": "friendly",
                                                        "escapeJS": "false"
                                                    }
                                                }
                                            }
                                        ],
                                        "isEditable": true,
                                        "isHidden": false
                                    },
                                    {
                                        "value": "may16user_6, may18user_c, may20user_L1, rkholyav_j1,
rkholyav_k1, soa7_x5, rkuser_B1, duserff94_y7, new_y3, soa_z3, dallaire_-9, fff_k3, NSAPIIUserNew_y5, shashi_qf, sanjajag_-3,
uhosursh_34, NSAPIIUser01_V3, mgujar_.3, dinkumar_X3, ipestel_Y3",
                                        "id": "demo_SF.username",
                                        "scale": 0,
                                        "maxLength": 5000,
                                        "inputType": "text",
                                        "fieldDataType": "Text",
                                        "formFieldLabelPO": {
                                            "content": "username"
                                        },
                                        "generateUnique": 0,
                                        "serverEditableOnly": false,
                                        "enableTypeahead": 0,
                                        "typeaheadMaxSearch": 0,
                                        "presentationDataType": {
                                            "textBox": {
                                                "length": "80",
                                                "name": "demo_SF.username",
                                                "value": "may16user_6, may18user_c, may20user_L1,
rkholyav_j1, rkholyav_k1, soa7_x5, rkuser_B1, duserff94_y7, new_y3, soa_z3, dallaire_-9, fff_k3, NSAPIIUserNew_y5, shashi_qf,
sanjajag_-3, uhosursh_34, NSAPIIUser01_V3, mgujar_.3, dinkumar_X3, ipestel_Y3",
                                                "height": "5",
                                                "fieldDataType": "Text",
                                                "controlName": "nstextbox",
                                                "isEditable": "true"
                                            }
                                        },
```

```
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 0,
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "PSCUSER",
                "id": "demo_SF.pscusers",
                "scale": 0,
                "maxLength": 500,
                "inputType": "text",
                "fieldDataType": "Text",
                "formFieldLabelPO": {
                        "content": "pscusers"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "80",
                                "name": "demo_SF.pscusers",
                                "value": "PSCUSER",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
                                "isEditable": "true"
                        }
```

```
                },
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 0,
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "",
                "id": "demo_SF.Country",
                "scale": 0,
                "maxLength": 50,
                "inputType": "select",
                "fieldDataType": "Text",
                "sliderType": "basic",
                "formFieldLabelPO": {
                        "content": "Country"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "multiSelectBox": {
                                "length": "50",
                                "name": "demo_SF.Country",
                                "value": "",
                                "selectOptionsCount": 9,
                                "disabled": "false",
```

```
"label": "",
"height": "5",
"fieldDataType": "Text",
"selectOptions": [
        {
                "name": "USA",
                "value": "USA",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
        },
        {
                "name": "UK",
                "value": "UK",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
        },
        {
                "name": "IN",
                "value": "IN",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
        },
        {
                "name": "JP",
                "value": "JP",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
        },
        {
                "name": "FR",
                "value": "FR",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
        },
        {
```

```
                "name": "CA",
                "value": "CA",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
            },
            {
                "name": "SA",
                "value": "SA",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
            },
            {
                "name": "ENG",
                "value": "ENG",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
            },
            {
                "name": "EUR",
                "value": "EUR",
                "isSelected": false,
                "controlName": "nsselectoptions",
                "isEditable": true
            }
        ],
        "controlName": "nsmulti",
        "isEditable": "true"
    }
},
"typeaheadMinStart": 0,
"isSecure": 0,
"isMandatory": false,
"showInGrid": 0,
"hasError": false,
"javascriptCallPOCount": 0,
"errorMessagePO": {
```

```
                "uiresourceCount": 0,
                "uiresource": []
            },
            "validatorPOCount": 0,
            "javascriptCallPO": [],
            "validatorPO": [],
            "isEditable": true,
            "isHidden": false
    },
    {
            "value": "",
            "id": "demo_SF.ProjectName",
            "scale": 0,
            "maxLength": 50,
            "inputType": "text",
            "fieldDataType": "Text",
            "formFieldLabelPO": {
                "content": "ProjectName"
            },
            "generateUnique": 0,
            "serverEditableOnly": false,
            "enableTypeahead": 0,
            "typeaheadMaxSearch": 0,
            "presentationDataType": {
                "textBox": {
                    "length": "50",
                    "name": "demo_SF.ProjectName",
                    "value": "",
                    "height": "5",
                    "fieldDataType": "Text",
                    "controlName": "nstextbox",
                    "isEditable": "true"
                }
            },
            "typeaheadMinStart": 0,
            "isSecure": 0,
            "isMandatory": false,
            "showInGrid": 0,
            "hasError": false,
            "javascriptCallPOCount": 0,
```

```
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "",
                "id": "demo_SF.LaptopBrand",
                "scale": 0,
                "maxLength": 50,
                "inputType": "text",
                "fieldDataType": "Text",
                "formFieldLabelPO": {
                        "content": "LaptopBrand"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "50",
                                "name": "demo_SF.LaptopBrand",
                                "value": "",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
                                "isEditable": "true"
                        }
                },
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 0,
```

```
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "",
                "id": "demo_SF.RAM",
                "scale": 2,
                "maxLength": 25,
                "inputType": "slider",
                "fieldDataType": "Money",
                "sliderType": "range",
                "interval": "10",
                "formFieldLabelPO": {
                        "content": "RAM"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "slider": {
                                "length": "50",
                                "name": "demo_SF.RAM",
                                "value": "",
                                "selectOptionsCount": 0,
                                "maxValue": "32",
                                "minValue": "4",
                                "label": "",
                                "height": "5",
                                "fieldDataType": "Money",
                                "sliderType": "range",
```

```
                "interval": "10",
                "booleanTruePrompt": "True",
                "booleanFalsePrompt": "False",
                "selectOptions": [],
                "controlName": "nsslider",
                "isEditable": "true"
            }
        },
        "typeaheadMinStart": 0,
        "booleanTruePrompt": "True",
        "booleanFalsePrompt": "False",
        "isSecure": 0,
        "isMandatory": true,
        "showInGrid": 0,
        "hasError": false,
        "javascriptCallPOCount": 0,
        "errorMessagePO": {
            "uiresourceCount": 0,
            "uiresource": []
        },
        "validatorPOCount": 3,
        "javascriptCallPO": [],
        "validatorPO": [
            {
                "type": "null",
                "param": "true",
                "validatorMessagePO": {
                    "uiresource": {
                        "key":
"common.alertmsgs.plsEnterAppropriateInfo",
                        "keyType": "friendly",
                        "escapeJS": "false"
                    }
                }
            },
            {
                "type": "maximum",
                "param": "32",
                "validatorMessagePO": {
                    "uiresource": {
```

```
                                                                "key":
"common.alertmsgs.plsEnterValueLessThan",
                                                                "keyType": "friendly",
                                                                "escapeJS": "false"
                                                        }
                                                }
                                        },
                                        {
                                                "type": "minimum",
                                                "param": "4",
                                                "validatorMessagePO": {
                                                        "uiresource": {
                                                                "key":
"common.alertmsgs.plsEnterValueGreaterThan",
                                                                "keyType": "friendly",
                                                                "escapeJS": "false"
                                                        }
                                                }
                                        }
                                ],
                                "isEditable": true,
                                "isHidden": false
                        }
                ],
                "gridEditable": false,
                "displayAsGrid": false,
                "formFieldPOCount": 7,
                "dictionaryAccess": true,
                "isVisible": true
        },
        {
                "name": {
                        "uiresource": {
                                "key": "Demo_GridSF",
                                "keyType": "preprocess",
                                "escapeJS": "false"
                        }
                },
                "id": "demo_GridSF",
                "autoDataRetrieval": false,
```

```
"templateTypeId": "1",
"maxRows": 5,
"recordSet": {
        "content": ""
},
"mdrRecordID": 0,
"displayRows": 5,
"formFieldPO": [
        {
                "value": "",
                "id": "demo_GridSF.OS",
                "defaultValue": "",
                "scale": 0,
                "maxLength": 50,
                "inputType": "text",
                "fieldDataType": "Text",
                "formFieldLabelPO": {
                        "content": "OS"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "instructionalText": "",
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "50",
                                "name": "demo_GridSF.OS",
                                "value": "",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
                                "isEditable": "true"
                        }
                },
                "typeaheadMinStart": 0,
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 1,
```

```
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "",
                "id": "demo_GridSF.Systembit",
                "defaultValue": "",
                "scale": 0,
                "maxLength": 50,
                "inputType": "text",
                "fieldDataType": "Text",
                "formFieldLabelPO": {
                        "content": "Systembit"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "instructionalText": "",
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "50",
                                "name": "demo_GridSF.Systembit",
                                "value": "",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
                                "isEditable": "true"
                        }
                },
                "typeaheadMinStart": 0,
```

```
                "isSecure": 0,
                "isMandatory": false,
                "showInGrid": 1,
                "hasError": false,
                "javascriptCallPOCount": 0,
                "errorMessagePO": {
                        "uiresourceCount": 0,
                        "uiresource": []
                },
                "validatorPOCount": 0,
                "javascriptCallPO": [],
                "validatorPO": [],
                "isEditable": true,
                "isHidden": false
        },
        {
                "value": "",
                "id": "demo_GridSF.Version",
                "defaultValue": "",
                "scale": 0,
                "maxLength": 50,
                "inputType": "text",
                "fieldDataType": "Text",
                "formFieldLabelPO": {
                        "content": "Version"
                },
                "generateUnique": 0,
                "serverEditableOnly": false,
                "enableTypeahead": 0,
                "instructionalText": "",
                "typeaheadMaxSearch": 0,
                "presentationDataType": {
                        "textBox": {
                                "length": "50",
                                "name": "demo_GridSF.Version",
                                "value": "",
                                "height": "5",
                                "fieldDataType": "Text",
                                "controlName": "nstextbox",
```

```
                                                "isEditable": "true"
                                        }
                                },
                                "typeaheadMinStart": 0,
                                "isSecure": 0,
                                "isMandatory": false,
                                "showInGrid": 1,
                                "hasError": false,
                                "javascriptCallPOCount": 0,
                                "errorMessagePO": {
                                        "uiresourceCount": 0,
                                        "uiresource": []
                                },
                                "validatorPOCount": 0,
                                "javascriptCallPO": [],
                                "validatorPO": [],
                                "isEditable": true,
                                "isHidden": false
                        }
                ],
                "gridEditable": true,
                "displayAsGrid": true,
                "formFieldPOCount": 3,
                "dictionaryAccess": true,
                "isVisible": true
        }
],
"javascriptCallPOCount": 2,
"errorMessagePO": {
        "uiresourceCount": 0,
        "uiresource": []
},
"javascriptCallPO": [
        {
                "content": "checkOrderingResources();",
                "event": "submit"
        },
        {
                "content": "Allstate_Common_OnSubmit(TestArg,TestArg1);",
                "event": "unload"
```

```
                        }
                ],
                "formSectionPOCount": 3,
                "hasServerError": false,
                "javascriptLibraryPOCount": 0,
                "javascriptLibraryPO": [],
                "jsResourceExists": "0",
                "currencyLangCode": "TWZH",
                "isFormReadOnly": false,
                "monitorPOCount": 0,
                "langCode": "TWZH",
                "monitorPO": []
        }
}
```


API to Get Service Order Form Metadata for Existing Order

Pre-Requisite – Before executing the API below parameters needs to be update in the newscale.properties.

```
################################################################
##Particle form rules js template path: service-form-rules-js-templates/formrules_javascript_tpl.ftl
formrule.js.template.path=service-form-rules-js-templates/formrules_javascript_tpl.ftl
################################################################
```

All the form details content can GET from the below API and response will be in Json

- API – GET Method - http://<Server_URL>:PORT/RequestCenter/nsapi/catalog/v1/requisitionentry/<requisitionentryid>/form

Below are the parameters needs to pass in the header

- Content-Type - application/json
- Accept - application/json

RBAC Access – User who have the order permission on the service or service group can execute this API.

Response of the API as below –


```
{
        "formPO": {
                "method": "post",
                "timeStamp": "",
                "pages": "[{\"description\":\"Tenant Information Form for when Tenant Management is enabled\",\"name\":\"Tenant-Information Form\",\"sections\":[\"XX_Tenant_Information_XX\"]},{\"description\":\"\",\"name\":\"Attach Volume\",\"sections\":[\"Volume_Action\",\"VolumeInfo_RO_Action\",\"VolumeInfo_Action\"]}]",
                "orderingMode": 3,
                "serviceID": 136,
          "javascriptFunctionDefinationsPO": {},
```

```
            “formSectionsPO”: []
             "showCurrencySymbol": "1",
                    "langCode": "USEN",
                    "monitorPO": [],
                    "jsResourceExists": "0",
                    "monitorPOCount": 0,
                    "isFormReadOnly": false,
                    "javascriptLibraryPO": [],
                    "currencyLangCode": "USEN",
                    "formSectionPOCount": 4,
                    "hasServerError": false,
                    "javascriptLibraryPOCount": 0,
            “errorMessagePO”:{}
            "javascriptCallPOCount": 1,
            “javascriptCallPO”: []
 }
}
```


Response Payload Description

- **javascriptFunctionDefinitionsPO –** This body contain the Java scripts, DDR and CR consumed by the services.
- **formPO-** Is the main attribute in the payload. Under this payload all the form content details will be displayed as metadata format.
- **Method:** Post method to submit form.
- **Pages:** This contains all the service forms name which are associated to a service.
- **FormSectionPO-** Contains all the General and Display Properties contents.
- **Key –** Caption of the service form
- **formFieldPO –** All the form filed level content will be displayed under this body
- **Id –** name of the dictionary
- **Value –** Value which passed while ordering service or default value
- **maxLength –** Length of the filed (defined when creating a dictionary)
- **fieldDataType –** Data type of the filed (defined when creating a dictionary)
- **inputType –** It is type of the field (such as select single, select multiple, radio , check box etc)
- **sliderType-** If the input type is slider , then slider type will be range
- **Scale –** Value will displayed when it is slider and range is set
- **validatorPO-** This value set when validate range set to true, minimum and maximum values will be defined in the same body.
- **GenerateUnique –** Set to zero if the Generate unique value is unchecked and if it is set to one when it is checked and when it is set to 1 Value filed will replace the default value with unique value generated by system
- **instructionalText-** It displays the help text of the field
- **ValidatiorPOCount-** This is count of validators. Example when validate range is checked, Mandatory is checked, Minimum/Maximum values are provided. Count set to 3, if one is unchecked then value set 2 and so on.
- **formFieldLabelPO – content -** This contains the label and advance formatting details of the files
- **associatedControlPO-** This contains the details of the add a button, Button Text, URL and Send data
- **showInGrid –** Set 0 if it is not grid, if it is grid value will be set as 1
- **isMandatory –** Set to true when it is mandatory if not it will be set to false
- **isHidden –** Set to true when it is hidden if not it will be set to false
- **errorMessagePO –** This contains the generic error message for DDR failure while loading the form. Ex : while loading service form , if the DDR execution message will be displayed for the user “Error executing webservice DDR DemoDDR. Please contact your system administrator”. same message will be stored in this body
- **javascriptCallPO –** This contains the function name , which is consumed by the service and also contains events (onload, submit) and if the function is having some arguments. Same arguments it will be contained by this body.

Accessing Service Details UI using Service Name

Previously the Service details UI URL was based on the ServiceId.

Eg for Service id 11.

Eg URL: http://localhost:8080/RequestCenter/website/ServiceCatalogWebsite/application/offer.html?route=offer&id=11

The Service Details page can now be accessed by passing the ServiceName as the query parameter.

Eg for Service Name AppleiPad using the name query parameter in the below URL

Eg URL:
http://localhost:8080/RequestCenter/website/ServiceCatalogWebsite/application/offer.html?route=offer&name=Apple%20iPad

API to execute DDR calls

The Service form Metadata API will return list of DDR rules that are configured for the various Service Forms in the Service. The Rule Ids are in encrypted format. The encrypted rule ids may have special characters in it and hence it is required to encode the rule id using base 64 encoding.

Note: The API execution for DDR calls only applicable for Form level (Browser-side) event

API to execute the DDR rule

Method: PUT
REST URL: /RequestCenter/nsapi/catalog/v1/service/Id/rule/<encoded ruleId>

Request Payload containing the values for the DDR execution.

Sample Request Payload:

[{"key":"Requisition.InitiatorID","value":"1"},{"key":"Requisition.CustomerID","value":"1"},{"key":"ServiceForm.ServiceId","value":"169"}]

Sample request and response for the DDR rule which is associated during on load event of the Form
Method: PUT

REST URL:
http://10.78.0.216:8080/RequestCenter/nsapi/catalog/v1/service/169/rule/RJZi0%2FXysRe6RdF%2BH1NWcQ%3D%3D

Sample Request Payload:

[{"key":"Requisition.InitiatorID","value":"1"},{"key":"Requisition.CustomerID","value":"1"},{"key":"ServiceForm.ServiceId","value":"169"}]

Sample Response Payload:

listOFValues :

key - Dictionary.Field

value – The value for the dictionary field.

listOFTargets:

list of the dictionary fields that are to be populated with the values fetched by executing the DDR rule.

Response:

```
{
    "listOfValues": [
        [
            {
                "key": "Personal_Computer_Dictionary.Make"
                "value": "Lenovo"
            },
            {
                "key": "Personal_Computer_Dictionary.Name",
                "value": "PC1"
```

```
                },
                {
                        "key": "Personal_Computer_Dictionary.NetName",
                        "value": "L1"
                },
                {
                        "key": "Personal_Computer_Dictionary.RAM",
                        "value": "64"
                }
        ]
]
        "listOfTargets": [
                "Personal_Computer_Dictionary.Make",
                "Personal_Computer_Dictionary.Name",
                "Personal_Computer_Dictionary.NetName",
                "Personal_Computer_Dictionary.RAM"
        ]
}
```


Service Order Form actions APIs

Submit Requisition API

API to submit a requisition. All the post submit DDRs are executed, field validations are performed and on successful validation of the entries the Requistion is submitted.

The Request Payload contains list of the Requistion Entries and the corresponding Dictionary Fields with the Values to be submitted.

Method: POST

REST URL: http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/requisition

Sample Request Payload :


```
{
  "requisition": {
    "customerLoginName": "admin",
    "services": [
      {
        "name": "Service_Grid",
        "quantity": "1",
        "version": "0",
        "dictionaries": [

                                {
          "name": "XX_Tenant_Information_XX",
          "data": {
```

```
            "XX_XX_TeamName_XX_XX" : "TeamA"

          }
        },

        {
          "name": "Dictioanary_Grid",
          "data": [{
            "Text":"Test1",
            "Boolean": "Yes",
            "SelectSingle": "GSS1"
          },
          {X
            "Text": "Test2",
           "Boolean": "Yes",
            "SelectSingle": "GSS1"
          },
                                                    {
            "Text": "Test3",
            "Boolean": "Yes",
            "SelectSingle": "GSS1"
          },
                                                    {
            "Text": "Test4",
            "Boolean": "Yes",
            "SelectSingle": "GSS1"
          },
                                                    {
            "Text": "Test5",
            "Boolean": "Yes",
            "SelectSingle": "GSS1"
          }]
        }
      ]
    }
  ]
 }
}
```

Add to cart API

Method: PUT

REST URL: http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/requisition

Sample Request Payload :

```
{
  "requisition": {
    "customerLoginName": "admin",
    "services": [
      {
        "name": "Service_Feature",
        "quantity": "1",
        "version": "0",
        "dictionaries": [
          {
            "name": "DictionaryFeature",
            "data": {
              "Field_Text1" : "hello",
              "Field_Number1": "3"
            }
          }
        ]
      }
    ]
  }
}
```

Save Service Alone as a Draft

Method: POST

REST URL: http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/requisition?saveAsDraft=true

Sample Request Payload :

```
{
  "requisition": {
    "customerLoginName": "admin",
    "services": [
      {
        "name": "Service_Feature",
        "quantity": "1",
        "version": "0",
        "dictionaries": [
```

```
        {
          "name": "DictionaryFeature",
          "data": {
            "Field_Text1" : "hello",
           "Field_Number1": "3"
          }
        }
       ]
     }
   ]
 }
}
```


Save Service as Draft including services in the Draft

Method: PUT

REST URL: http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/requisition?saveAsDraft=true

Sample Request Payload :


```
{
  "requisition": {
    "customerLoginName": "admin",
    "services": [
      {
        "name": "Service_Feature",
        "quantity": "1",
        "version": "0",
        "dictionaries": [
          {
            "name": "DictionaryFeature",
            "data": {
              "Field_Text1" : "hello",
             "Field_Number1": "3"
            }
          }
        ]
      }
    ]
  }
}
```

API to Export/Import Services

Export Service API

The Export service API takes a list of Service IDs and exports the Service details in to an xml.
The response is similar to the xml that is provided on the export of the services from Catalog Deployer.

- Export of multiple ServiceIds namely S1=241, S2=242 and S3=243 are passed as below,
  EndPoints:
  http://10.78.0.247:8080/RequestCenter/nsapi/rex/v1/definition/service?id=243&id=242&id=241&includeContainedServices=true
  Method: GET

- In the event we have a Service S1 which is a Bundled Service enclosing Service S2 and we also want export another Service S3 then the End Point would be as below,
  EndPoints:
  http://10.78.0.247:8080/RequestCenter/nsapi/rex/v1/definition/service?id=243&id=241&includeContainedServices=true
  Method: GET

- In the event we need to export Service S1 which is a Bundled Service enclosing Service S2 but we do not want S2 to be exported. Then the End Point would be as below,
  EndPoints:
  http://10.78.0.247:8080/RequestCenter/nsapi/rex/v1/definition/service?id=243&id=241&includeContainedServices=false
  Method: GET

Import Service API

- When we want to import the exported XML the End Point would be as below,
  EndPoints: http://10.78.0.251:8080/RequestCenter/nsapi/rex/v1/definition/service
  Method: PUT

Note:

- The Export file should not be tampered and imported as is
- The User Exporting/Importing should have SD Capability
- The User should have the WebServices- Rex Api Access Capability
- The User Exporting/Importing should have the Integrations Administrator Role out of the box
- The includeContainedServices flag is by default true.

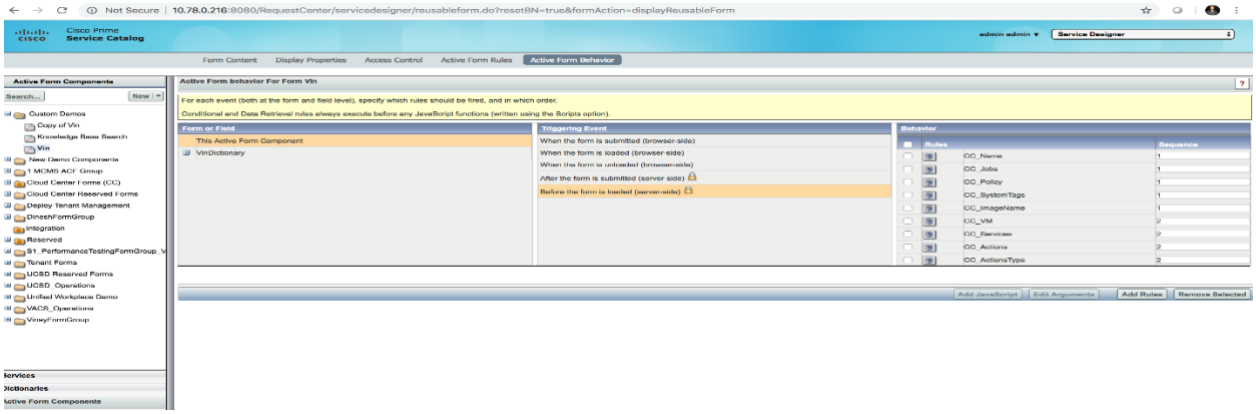
User Exporting/Importing have the following Persmissions

1) Read/Write Definition-Service Item Definition”All Objects”
2) Read/Write Definition-Service Item Group ”All Objects”

Execute Independent Webservice DDR in parallel threads

- This feature provides a provision to execute server-side rules in parallel to optimize the to load the Service Form.
- Based on the dependency the DDR rules have they need to be sequenced. For Example, if DDR2 depends on the response of DDR1. DDR2 needs to be sequenced after DDR1.
  if there is no dependency for DDR2, it can have the same sequence as DDR1 and this will result in both the rules being executed in parallel and hence optimizing the time to execute them. Previously the DDR rules were executed one after the other.
- To do this the sequence of execution can be defined from **Active Form Components> Active Form Behavior >Select Triggering Event (Before the Form is Loaded(server-side))>Add Rules and update the Sequence**.
- When two or more Rules have the same sequence number, i.e. they are grouped together, they will be executed in parallel. An example of such rules might be ones which are independent of the results from the rules in their group.
- The number of threads spawned per request can be configured from administration settings,
- **Administration> Settings>Customizations>Maximum Parallel rules to execute.**
- If its set to “0”—there will be no limit (this is not recommended)
- By default, the value is set to 25(which means 25 threads spawned per request)

Active Form Behavior



Administration Settings

Form Rule javascript FTL Changes

- Service Form Rule execution (DDR and CR) depends on implementation of ISF functions. It is a javascript code which is executed when events configured are triggered. This javascript is service form specific javascript which is generated from service form rules configuration done on service designer. This javascript could be retrieved from service form meta data api response <add hyperlink here>. javascriptFunctionDefinitionsPO property of the response contains it. Execution of this scripts will configure event handlers to undertake actions specific to the service.
- Form rule javascript is generated from freemarker template(ftl) file using rule configuration done on service designer. .
- A shipped version of the ftl file could be found at, /RequestCenter.war/WEB-INF/classes/service-form-rules-js-templates/formrules_javascript_tpl.ftl.

- The generated javascript uses an ISF library provided by the renderer form rule executions flow.
- New service form renderer implementation could be achieved by rewriting the ftl and corresponding ISF library.

```
// isf_methodArray holds type of the rule (DDR or CR) against rule id
ISF.isf_methodArray();

// isf_eventMap holds array of javascript functions to be invoked for a rule against event and dictionary field details as key
ISF.isf_eventMap();

/*
to make ddr execution api call
        form: contains service form meta data and values
        dictionaryParameters: dictionary params for request body
        encodedRuleId: encoded encrypted rule id
*/
isf_ddr_call = (form, dictionaryParameters, encodedRuleId)

/*
        form: contains service form meta data and values
        compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal
```

```
*/
isf_cr_equals = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
        compares left operand with right operand case ignored. leftOperandIsField/rightOperandIsField will if
leftOperand/rightOperand is dictionary field name or a literal
*/
isf_cr_equals_nocase = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
        compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is
dictionary field name or a literal
*/
isf_cr_not_equals = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
        compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is
dictionary field name or a literal
*/
isf_cr_gt = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
        compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is
dictionary field name or a literal
*/
isf_cr_lt = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
        compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is
dictionary field name or a literal
*/
isf_cr_gte = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*
        form: contains service form meta data and values
```

compares left operand with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_lte = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*

form: contains service form meta data and values

checks if left operand starts with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_starts_with = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*

form: contains service form meta data and values

checks if left operand ends with right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_ends_with = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)

/*

form: contains service form meta data and values

leftOperand field is not equal to undefined. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_exists = (form, leftOperand, leftOperandIsField)

/*

form: contains service form meta data and values

checks if isDisabled is true for left operand field. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_isDisabled = (form, leftOperand, leftOperandIsField)

/*

form: contains service form meta data and values

checks if isDisabled is false for left operand field. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_not_isDisabled = (form, leftOperand, leftOperandIsField)

/*

```
        form: contains service form meta data and values

        checks if isHidden is true for left operand field. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_isHidden = (form, leftOperand, leftOperandIsField)


/*

        form: contains service form meta data and values

        checks if leftOperand is not equal to null. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_isNull = (form, leftOperand, leftOperandIsField)


/*

        form: contains service form meta data and values

        checks if leftOperand contains right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_contains = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)


/*

        form: contains service form meta data and values

        checks if leftOperand does not contain right operand. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_not_contains = (form, leftOperand, leftOperandIsField, rightOperandIsField, rightOperand)


/*

        form: contains service form meta data and values

        checks if isHidden is false for left operand field. leftOperandIsField/rightOperandIsField will if leftOperand/rightOperand is dictionary field name or a literal

*/

isf_cr_not_isHidden = (form, leftOperand, leftOperandIsField)


/*

        form: contains service form meta data and values

        sets isHidden for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName

*/

isf_cr_setHidden = (form, elem, onoff = false)


/*

        form: contains service form meta data and values
```

```
        sets value for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName
*/
isf_cr_setPrice = (form, elem, value = "")

/*
        form: contains service form meta data and values
        sets value for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName
*/
isf_cr_setValue = (form, elem, value = "")

/*
        form: contains service form meta data and values
        sets isMandatory for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName
*/
isf_cr_setMandatory = (form, elem, onoff = false)

/*
        form: contains service form meta data and values
        sets isEditable for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName
*/
isf_cr_setDisabled = (form, elem, onoff = true)

/*
        form: contains service form meta data and values
        sets sets focus for dictionary field in form object. elem could be key for the field. e.g ele = DictionaryName.fieldName
*/
isf_cr_setFocus = (form, elem)

/*
        form: contains service form meta data and values
        show alert popup with the message
*/
isf_cr_alert = (form, msg)

/*
        form: contains service form meta data and values
        stop submission of the service form
*/
isf_cr_stopSubmission = (form)
```

```
/*
        checks if logged in user is member of role
*/
isf_isMemberOfRole(roleId)

/*
        checks if customer is member of role
*/
isf_isCustomerMemberOfRole(roleId)


/*
checks if field is grid field or not
        form: contains service form meta data and values
*/
isf_isGridField = (form, fieldKey)

/*
checks if dictionary is available on the form or not:
        form: contains service form meta data and values
        dictionaryName
*/
isf_checkDictionaryExistence = (form, fieldKey)
```

Note: This FTL and available ISF function invocations are not exhaustive under development and subject to change.

UCSD VM Provisioning

Previously whenever a new VM is provisioned or failed VM is refreshed a complete sync of VDC would happen.

- ✓ **New behaviour:** Whenever a new VM is provisioned or failed VM is refreshed only that VM data gets synced back to PSC, the complete VDC Synch does not happen.

API to get Service/User context

Order Form Context API (GET API) provides context parameters which are used for loading the service order form.

REST URL: http://localhost:8080/RequestCenter/nsapi/catalog/v1/service/150/orderformcontext

Method: GET

Query Param: customerId

In case of Order on behalf user scenario, the customer id of the user for whom it is being ordering needs to be passed

http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/service/148/orderformcontext?customerId=15

Response Payload:-

```
{
	"Map": {
		"siteAdminUser": true,
		"EnableOOBAutoComplete": false,
		"orderConfirmationDeliverToFormat": "Customer(Login)",
		"allowAddAttachment": true,
		"isNextGenUser": true,
		"clickableHelpIcon": false,
		"timeZone": "America/Tijuana",
		"hideNRDicts": false,
		"servicedescproperty": false,
		"hideMonitorFlag": false,
		"Serviceformelement.display.instructional.helptext": false,
		"isMessageInfo": "false",
		"DescriptionLimit": "0",
		"nsBasePath": "/RequestCenter/",
		"service": {
			"serviceID": 150,
			"name": "testService1",
			"documentID": 0,
			"estimatedCost": 0.0,
			"expectedDuration": 0.0,
			"hasPortal3": true,
			"portalText3": "<p>TEST3</p>\n",
			"isEntitlement": 0,
			"isInactive": 0,
			"priceDescription": "",
			"shortPriceDescription": "",
			"priceDisplaySchemaID": 0,
			"pricingSchema": 0,
			"revisionNumber": 34,
			"allowFutureDelivery": true,
			"isBundle": false,
			"isTemplate": false,
			"tenantRelevant": true,
			"isOrderable": true,
			"orderingMode": 3,
			"computePrice": false,
			"paginationViewMode": 0,
```

```
                    "isDefinedDuration": false,
                    "showOrderSummary": false
                },
                "allowUpdateQty": false,
                "sessionToken": "F85E174AB7425AD66555E5CBD517CDF4",
                "ldapLookupOnOnlyForPerson": true,
                "ldapLookupOn": true,
                "Serviceform.label.alignment": "right",
                "Serviceform.field.line.separator": false
        }
}
```


API to Update Requisition Entry

UpdateRequistionAPI is used for updating the RequisitionEntry pertaining to a Requistion. The Form fields in this RequisitionEntry would be updated.

- In this API we could have multiple Dictionaries with fields within. Validations are taken care at a Dictionary Level for the fields
- All the dictionary field validations and post submit DDRs are executed.
- Sample Payload for different Data types are illustrated in the example below

  EndPoints: http://10.78.0.247:8080/RequestCenter/nsapi/catalog/v1/requisitionentry/217/form

Method: PUT

UpdateRequsitionAPI Payload:


```
{


  "dictionaries": [
     {
      "name": "NagDictFF_NonGrid",
      "data": {
        "Text": "12",
        "Number": "12",
        "Account": "NewAccount",
        "Boolean": "No",
        "Person": "dusersd",
         "Tenant":"TeamA",
          "CheckBox": ["Checkbox3", "Checkbox4", "Checkbox5"],
          "Radio": "Radio3",
        "Date": "10/29/2018",
        "Money": "12",

        "Slider": "30",
        "SelectSingle": "SS3",
```

```
    "SelectMultiple": ["SM1", "SM4"],
    "DateAndTime": "10/29/2018 10:42 PM",
    "Phone": "23634408",
    "SSN": "1234",
    "URL": "http://google.co.in",
    "TextArea": "This is the textarea",
    "Password": "hello"
  }
},



{
  "name": "NagDictFF_Grid",
  "data": {
    "Text": "Din1",
    "Number": "123",
                                        "Account":"NewAccount",
                                        "Boolean":"No",
                                        "Tenant":"TeamZZ",
                                        "Person": "din01",
                                        "Date":"10/25/2018",
                                        "Money":"12",
                                        "DateAndTime":"10/24/2018 5:20 PM",
                                        "Phone":"23634408",
                                        "SSN":"1234",
                                        "URL":"http://google.co.in",
                                        "TextArea":"This is the textarea",
                                        "Password":"hello",
                                        "SelectSingle":"GSS1"


                                }
                        },
                        {
  "name": "NagDictFF_Grid",
  "data": {
    "Text": "Din1",
    "Number": "123",
                                        "Account":"NewAccount",
```

```
                                        "Boolean":"No",
                                        "Tenant":"TeamZZ",
                                        "Person": "din01",
                                        "Date":"10/25/2018",
                                        "Money":"12",
                                        "DateAndTime":"10/24/2018 5:20 PM",
                                        "Phone":"23634408",
                                        "SSN":"1234",
                                        "URL":"http://google.co.in",
                                        "TextArea":"This is the textarea",
                                        "Password":"hello",
                                        "SelectSingle":"GSS1"


                                }
                        },

                        {
    "name": "NagDictPrsnBsd",
    "data": {
      "Select_Person": "din01"



                        }
                                },

{
    "name": "NagDictSIBD",
    "data": {
      "Name": "din01 din01",
      "RequisitionID": "12",
                                        "Age":"12",
                                        "Address": "dineshk2@cisco.com",
                                        "Money":"132",
                                        "SubmittedDate": "10/16/2018",
                                        "Lkong":"12",
                                        "DateTime":"10/31/2018 12:00 AM",
                                        "DTime": "10/31/2018 12:00 AM"
```

```
                }
                        }


 ]


}
```

Service Item Billing Estimated Cost

This API is used to get all the service item's estimated price for a given service form data.

Rest URL: http://localhost:8080/RequestCenter/nsapi/catalog/v1/service/152/price

Method: PUT

Sample Request Payload:-

```
{
        "dictionaries": [{
                        "name": "AOC_SIBD_ComputePrice",
                        "data": {
                                "Name": "Jill",
                                "Field_Text": "DEF",
                                "Field_Number": "200",
                                "AccountID": "14",
                                "CustomerID": "1",
                                "OrganizationalUnitID": "1"

                        }
                },
                {
                        "name": "AOC_SIBD_ComputePrice2",
                        "data": {
                                "Name": "Jack",
                                "Field_Text2": "ABC",
                                "Field_Number2": "100",
                                "AccountID": "7",
```

```
                    "CustomerID": "1",
                    "OrganizationalUnitID": "1"

                }
            },
            {
                "name": "AOC_SIBD_ComputePriceUnit",
                "data": {
                    "Name": "Eve",
                    "Field_Number_UnitRate": "200",
                    "AccountID": "7",
                    "CustomerID": "1",
                    "OrganizationalUnitID": "1"

                }
            },
            {
                "name": "AOC_SIBD_ComputePriceUnit2",
                "data": {

                    "Name": "David",
                    "Field_Number_UnitRate2": "300",
                    "AccountID": "7",
                    "CustomerID": "1",
                    "OrganizationalUnitID": "1"
                }
            }
        ]

}
```


Response Payload:-


```
{
  "totalPrice": "26400.0 Sp_Storage GB per Billing Cycle; 12200.0 Sp_Transactions; 123.0 Sp_Years",
  "list": [
    {
      "dictionaryName": "AOC_SIBD_ComputePriceUnit2",
      "priceList": [
        "26400.0 Sp_Storage GB per Billing Cycle"
      ],
```

```
    "totalPrice": ""
   },
   {
    "dictionaryName": "AOC_SIBD_ComputePrice1",
    "priceList": [],
    "totalPrice": ""
   },
   {
    "dictionaryName": "AOC_SIBD_ComputePriceUnit1",
    "priceList": [
     "12200.0 Sp_Transactions"
    ],
    "totalPrice": ""
   },
   {
    "dictionaryName": "AOC_SIBD_ComputePrice2",
    "priceList": [
     "123.0 Sp_Years"
    ],
    "totalPrice": ""
   }
  ]
}
```


Introduced in 12.1_Patch_v4

Save an Order Form as a Draft

A new button **Save as Draft** in a Service Catalog order form allows users to save the incomplete forms and edit them at a later time. The status of these requisitions are indicated as Draft in the Orders page. These orders can be cancelled, moved to cart, or submitted after completing the order form. The Save as Draft option is also available at service form and cart.

Note:

1. Move to Cart operation can be performed only when the cart is empty.
2. Draft requisitions can be saved as many as per user.
3. If Team Management is enabled and service is Team Relevant then, Team name validation in Service form will not be skipped with save as draft (it should always have a value).

Return to Review

Any users having approval tasks assigned to them can now send back the requisition to the submitter using the *Return to Review* option in the Open authorizations tab of **My Stuff > Notifications** in Service Catalog module. The approval tasks could be any of the Service Group, Departmental, or Financial authorizations.

When an order is returned to review, the approver must add a comment for returning the order. This note would be useful for the submitter to make the necessary changes to the order and resubmit it. When the service is retuned to review, the subsequent service delivery process also moves to return to review state. Once the Requisition is resubmitted all of the authorizations will be re-triggered.

The *Return to Review* option is also available at the following locations in Prime Service Catalog:

- Service Catalog> MyStuff > Approvals > Authorizations
- Service Manager > Views for Approval Tasks
- Service Link > View Transactions > External Tasks

API for return to review is described in section *Gets Requisitions to* be archived or purged

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition?hasWorkFlowData=true
Sample response:


```
{
        "requisitions": {
                "startRow": 1,
                "totalCount": 1,
                "recordSize": 1,
                "requisition": [
                        {
                                "tenantId": 0,
                                "userId": 0,
                                "ownerId": 0,
                                "serviceId": 0,
                                "customerId": 10645,
                                "expectedDuration": 0.0,
                                "actualDuration": 0.0,
                                "expectedCost": 0.0,
                                "status": "Closed",
                                "requisitionId": 535192,
                                "lateFlag": false,
                                "statusId": 2,
                                "organizationalUnitId": 0,
                                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=535192&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">535192</a>",
                                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=535192",
                                "milestoneLink": "",
                                "percentageCompleted": 0.0,
                                "isCancelable": false,
                                "reviewsCount": 0,
                                "selfRating": 0,
                                "isServiceOrderable": 0,
                                "isServiceActive": 0,
                                "hasAttachment": false,
                                "teamId": 0,
                                "createdDate": "12/09/2010 7:39 下午"
```

```
                    }                    ]
        }
}
```


Archive or purge requisitions
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/requisitions/workflowdata?requisitioncount={recordSize}
Where, recordSize is number of requisitions that must be archived or purged based on the *Purge/Archive Workflow Data* setting in **Administrations > Settings**.

Sample response:


```
{
        "nsapi-error-response": {
                "status-messages": [
                        {
                                "code": "NSAPI_ERROR_004",
                                "value": " Requisition Purged successfully"
                        }
                ]
        }
}
```


Return for review.

Auto Suggest Search for Order on Behalf

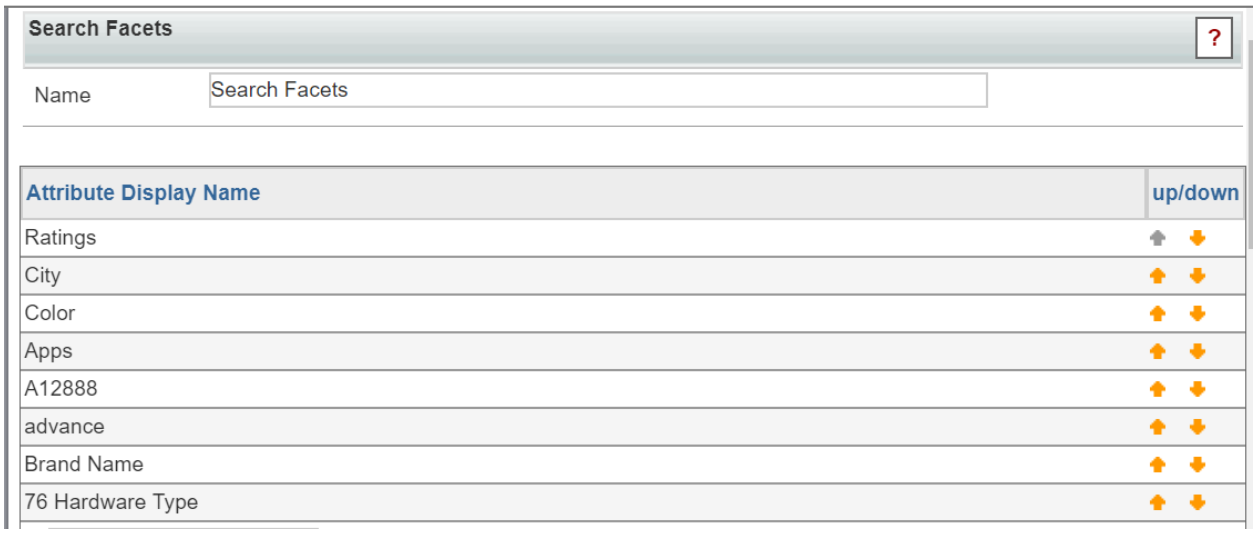
Name search for Order on Behalf person pop-up is now enhanced to single name search, where search operation can be performed by first name, last name, or login name.

In addition, you can enable auto suggest feature for the Order on Behalf person pop-up search. As you type into the search box, it tries to predict your query based on the characters you have entered. To enable search auto suggest feature, go to **Administration** module > **Settings** and enable the option *Enable OOB Auto Suggest.* This option is disabled by default.

Note: Logged in user must have Order on Behalf permissions to order services for any other user.

Configuring Search Facet Display Order

The search facets available on the left pane of Service Catalog module are used to filter or search services based on the criteria chosen. As a Site Administrator, you can now set the priority of the search facets by arranging the facet priority from **Service Designer > Extensions > Search Facets** list panel. Use the up/down arrow to position order of facets to be displayed on the Service Catalog page.



APIs for configuring facet display order is described in section *Get search facets with priority* and *Update priority of search facets*.

Sort Services within a Category in Service Catalog

As an administrator, you can set the sorting of services within a category or subcategory based on Most Ordered or Name (Ascending/Descending). By default, the services are displayed in Service Catalog as per sorting configured in Service Designer.

Use the option *Services Sorting Pattern* setting to change the sorting pattern provided in the **Administration** module > **Settings**.

System will display services that are ordered most based on order volume starting from a specified period. For example, if month is selected as January and year as 2012, then services which were ordered most since January 2012 till date will be displayed first for the selected category.

Authorization Delegate Assignment in Approvals

The Authorization Delegate feature which was available on Profile > Preferences is now available in the **Service Catalog > Approvals** from the hamburger menu. The new option *Authorization Delegate* allows you to delegate the authorizations tasks to a person for you during the time period you specify using the Delegation Start Date and Delegation End Date fields. For more information see section Assigning Permissions to a Person in *Cisco Prime Service Catalog 12.1 Administration and Operation Guide*.

Configuring Key Fields for Service Form Details API

A Service Designer can now configure the approval details in the get service data API such that only the key fields necessary for approval are displayed.

You can configure the fields at dictionary or active form component level for a service.

- Dictionary level:
  Go to **Service Designer > Dictionaries >** check/uncheck ***is Key*** field for each dictionary field. Only the dictionary fields marked as *Is Key* will appear on the service form API.
- Active Form Component level:
  Go to **Service Designer > Active form components > Display properties >** check/uncheck **overrideIsKey**. When this option is checked, it overrides the default *is Key* settings for each dictionary field.

Based on the configuration settings in Service Designer, the response in the below APIs will display only the necessary fields:

1. Get Key Fields for the specified requisition:
   http://<serverURL>/RequestCenter/nsapi/transaction/requisitions/id/{requisitionID}?requisitiondata=true&fetchKeyFieldsOnly=true

2. Get Key Fields for the specified requisition with requisitiondata:
   http://<serverURL>/RequestCenter/nsapi/transaction/requisitions/id/{requisitionID}/requisitiondata?fetchKeyFieldsOnly=true

Query params:

- requisitiondata = true/false – this specifies if the API should fetch the service form fields of requisition entries in the requisition
- fetchKeyFieldsOnly = true/false – this specifies whether to fetch only keyFields for this requisition or not and should be used when requisitiondata = true

See section *Get Key Fields for the specified requisition* and *Get Key Fields for the specified requisition with requisition data* for sample response.

Archiving and Purging Requisitions

As a Site administrator, you can now archive or purge historical completed requisitions using the *Purge/Archive Workflow Data* setting in **Administrations > Settings**. The associated APIs described below are used to set the count of requisitions to be purged or archived starting from the oldest requisition in the system.

Only the requisitions that are in Completed, Cancelled, Rejected and Delivery Cancelled state are archived or purged.

Use the below APIs to fetch the requisitions that are yet to be Archived/Purged and then set the number of entries to be archived or purged:

1. Get API to fetch requisitions that are yet to be Archived/Purged:
/RequestCenter/nsapi/transaction/v2/requisition?hasWorkFlowData=true

2. Get API to Archive/Purge requisitions based on the Administration Settings:
/RequestCenter/nsapi/transaction/v2/tasks/requisitions/workflowdata?requisitioncount={recordSize}

Where, the RecordSize in the second API is configurable, using the new flag in the *newscale.properties* file as below:

```
max.allowed.requisition.purge.count=100000
```

The maximum records that can be archived or purged is 100000.

If *Purge/Archive Workflow Data* radio button is set to Archive then the number of requisitions mentioned in the API will be archived. If the radio button is set to Purge then the number of requisitions mentioned in the API will be purged.

For more details on the APIs for for this feature, see section Gets Requisitions to be archived or purged and Archive or purge requisitionsGets Requisitions to be archived or purged

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition?hasWorkFlowData=true
Sample response:


```
{
    "requisitions": {
        "startRow": 1,
        "totalCount": 1,
        "recordSize": 1,
        "requisition": [
            {
                "tenantId": 0,
                "userId": 0,
                "ownerId": 0,
                "serviceId": 0,
                "customerId": 10645,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "expectedCost": 0.0,
                "status": "Closed",
                "requisitionId": 535192,
```

```
                                        "lateFlag": false,
                                        "statusId": 2,
                                        "organizationalUnitId": 0,
                                        "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=535192&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">535192</a>",
                                        "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=535192",
                                        "milestoneLink": "",
                                        "percentageCompleted": 0.0,
                                        "isCancelable": false,
                                        "reviewsCount": 0,
                                        "selfRating": 0,
                                        "isServiceOrderable": 0,
                                        "isServiceActive": 0,
                                        "hasAttachment": false,
                                        "teamId": 0,
                                        "createdDate": "12/09/2010 7:39 下午"
                                }                       ]
        }
}
```


Archive or purge requisitions

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/requisitions/workflowdata?requisitioncount={recordSize}
Where, recordSize is number of requisitions that must be archived or purged based on the Purge/Archive Workflow Data setting in Administrations > Settings.

Sample response:


```
{
        "nsapi-error-response": {
                "status-messages": [
                        {
                                "code": "NSAPI_ERROR_004",
                                "value": " Requisition Purged successfully"
                        }
                ]
        }
}
```


Return for review.

The internal DB workflow tables that are archived and purged are:

Purge tables	Archival Tables
TxJoin	ArJoin
TxJoinExt	ArJoinExt
TxSubscription	ArSubscription
TxTimer	ArTimer
TxCondition	ArCondition
TxRule	ArRule
TxService	ArService

During purge, the data from Purge Tables are deleted. However, during archival, the data from the Purge Tables moves to the respective Archival Tables.

Service Manager Website

A new UI for the Service Manager module has been introduced in this patch release which allows you to customise it using the website model. Once you upgrade to 12.1 patch V4 the new Service Manager UI is enabled by default. It must be noted that all the functionalities offered by the Service Manager Classic UI remains the same in the new Service Manager UI. If you prefer the Service Manager Classic UI, disable the option "Enable Service Manager Website" in **Administration > Settings**.

The appearance of Service Manager can be customised using website model similar to other modules such as Service Catalog, Tenant Management, Cloud Integrations, and User Management. Define a Custom Theme for Service Manager by enabling the check-box *Website for Service Manager* from **Administration** module**> Settings** tab > **Custom Themes > Custom Theme Properties.** For more details see section "*Customizing Customer Facing Modules"* in *Cisco Prime Service Catalog 12.1 Administration and Operation Guide*.

New APIs are introduced for Service Manager in this patch release. See section *New Service Manager APIs*.

Email Notifications for CloudCenter Deployments

You receive an email notification when you create or migrate CloudCenter application in Prime Service Catalog. When you deploy a new CloudCenter application successfully, an email notification is sent out, which includes service item details, subscription data, and specific details of the application such as name, display name, description, cloud name, and status of the application with its corresponding virtual machines.

This configuration of email notification is identical to the procedure explained in the section "*Configuring Email Notification on VDC Creation"* in *Cisco Prime Service Catalog 12.1 Administration and Operation Guide*.

Support for Custom Actions on Deployments and VMs Enhancement

The Support for Custom Lifecycle Operations on VMs feature deliver in 12.1 V3 Patch has been enhanced and the procedure in section *Support for Custom Lifecycle Operations on VMs* has been updated. In addition, Custom Actions is also supported on deployments. For detailed procedure to configure custom actions for CloudCenter deployments see section *Support for Custom Actions Operations on Deployments*.

Enhanced APIs

Authorization API to return only needed data

The existing Get Authorization API has been enhanced to display only the necessary information using the newly introduced query parameters.

Method: GET

REST URL: /RequestCenter/nsapi/transaction/authorizations/{filters}?requiredFields= dueOn,lateFlag,isOOB,
QueryParam: requiredFields = <dueOn,lateFlag,isOOB>

Sample Response:

```
{
        "authorizations": {
                "startRow": 1,
```

```
                "totalCount": 2,
                "recordSize": 2,
                "authorizationtask": [
                        {
                                "dueOn": "06/22/2018 4:00 PM",
                                "lateFlag": false,
                                "isOOB": false
                        },
                        {
                                "dueOn": "06/22/2018 4:00 PM",
                                "lateFlag": false,
                                "isOOB": false
                        }
                ]
        }
}
```


Requisition Header for Ongoing Authorizations

The existing authorization API for an ongoing transaction has been enhanced to add *includeHeader* query parameter. This API can be accessed only by a person who has permissions to approve the task.

Method: GET

REST URLs:
/RequestCenter/nsapi/transaction/authorizations/ /RequestCenter/nsapi/transaction/authorizations/{filters}?includeHeader=true

Added Query Param: includeHeader=True/false, returns following attributes in the response:

current task id, status , performer id, first name, last name, requisition due date, task due date, owner id, owner name, customer id, customer login name, whether it is order on behalf, Difference in due from today (how many day to go, how many days have passed)

Sample response:


```
{
        "authorizations": {
                "startRow": 1,
                "totalCount": 1,
                "recordSize": 1,
                "authorizationtask": [
                        {
                                "tenantId": 1,
                                "dueOn": "07/17/2018 8:00 PM",
                                "totalPrice": "$ 0.00",
                                "status": "Being approved",
                                "requisitionId": 2281,
                                "taskName": "AuthQ",
```

```
                                        "customer": "admin admin : Site Administration",
                                        "performer": "PSCQ Queue",
                                        "lateFlag": false,
                                        "priority": "Normal",
                                        "priorityIcon": "/RequestCenter/images/prioritynormalicon.gif",
                                        "dueOnRaw": 1531882800000,
                                        "activityId": 4143,
                                        "authorizationURL": "<a
href='/RequestCenter/servicemanager/navigate.do?query=task&taskID=4143&layout=popup_p&selectedTasks=&' onclick=\"return
GB_showFullScreen('Task', this.href)\">AuthQ</a>",
                                        "serviceName": "PersonalLaptop_service",
                                        "authorizationURLOnly":
"/RequestCenter/servicemanager/navigate.do?query=task&taskID=4143",
                                        "scheduledStartRaw": 1531782000000,
                                        "startedOnRaw": 1531769664493,
                                        "flagId": 0,
                                        "isPerformer": true,
                                        "statusId": "6",
                                        "priorityTypeId": "2",
                                        "createdOn": "07/16/2018 11:23 AM",
                                        "createdOnRaw": 1531765431757,
                                        "customerLoginName": "admin",
                                        "customerId": 1,
                                        "ownerName": "admin admin",
                                        "isOOB": false,
                                        "ownerId": 1,
                                        "performerId": 1060,
                                        "performerFirstName": "PSCQ",
                                        "performerLastName": "Queue",
                                        "requisitionDueOn": "07/17/2018 8:00 PM",
                                        "requisitionDueOnRaw": 1531882800000,
                                        "daysToDueDate": 0,
                                        "requisitionEntryId": 2416
                                    }
                                ]
                            }
                        }
```


Extra custom data on Closed Authorization API

Method: GET

REST URL: /RequestCenter/nsapi/transaction/authorizations/ViewName=<viewName>[|AND|Status=<status>]

Status for closed authorization can be: Approved, Rejected, Canceled, and Reviewed

Query params:

includeHeader=When set to *true,* the following new fields are displayed in the response:

createdOn,createdOnRaw, customerLoginName ,customerId ,ownerName, isOOB, ownerId ,performerId, performerFirstName, performerLastName, requisitionDueOn, requisitionDueOnRaw, daysToDueDate

Response:


```
{
    "authorizations": {
        "startRow": 1,
        "totalCount": 2,
        "recordSize": 2,
        "authorizationtask": [
            {
                "tenantId": 1,
                "dueOn": "06/22/2018 4:00 PM",
                "totalPrice": "$ 0.00",
                "status": "Being approved",
                "requisitionId": 671170,
                "taskName": "UserAuth1",
                "customer": "admin admin : Site Administration",
                "performer": "shashi1 shashi1",
                "lateFlag": false,
                "priority": "Normal",
                "priorityIcon": "/RequestCenter/images/…",
                "dueOnRaw": 1529663400000,
                "activityId": 2125752,
                "authorizationURL": "<a href='/RequestCenter/servicemanager/.../UserAuth1</a>",
                "serviceName": "UserAuthorization",
                "authorizationURLOnly": "/RequestCenter/servicemanager/…",
                "scheduledStartRaw": 1529663400000,
                "startedOnRaw": 1529628842143,
                "flagId": 0,
                "isPerformer": true,
                "statusId": "6",
                "priorityTypeId": "2",
                "createdOn": "06/22/2018 6:23 AM",
                "createdOnRaw": 1529628825920,
                "customerLoginName": "admin",
                "customerId": 1,
```

```
                "ownerName": "admin admin",
                "isOOB": false,
                "requisitionEntryId": 662941
            },
            {
                "tenantId": 1,
                "dueOn": "06/22/2018 4:00 PM",
                "totalPrice": "$ 0.00",
                "status": "Being approved",
                "requisitionId": 671170,
                "taskName": "UserAuth1",
                "customer": "admin admin : Site Administration",
                "performer": "shashi1 shashi1",
                "lateFlag": false,
                "priority": "Normal",
                "priorityIcon": "/RequestCenter/images/…",
                "dueOnRaw": 1529663400000,
                "activityId": 2125753,
                "authorizationURL": "<a href='/RequestCenter/servicemanager/…/UserrAuth1</a>",
                "serviceName": "UserAuthorization",
                "authorizationURLOnly": "/RequestCenter/servicemanager…",
                "scheduledStartRaw": 1529663400000,
                "startedOnRaw": 1529628842737,
                "flagId": 0,
                "isPerformer": true,
                "statusId": "6",
                "priorityTypeId": "2",
                "createdOn": "06/22/2018 6:23 AM",
                "createdOnRaw": 1529628827027,
                "customerLoginName": "admin",
                "customerId": 1,
                "ownerName": "admin admin",
                "isOOB": false,
                "requisitionEntryId": 662942
            }
        ]
    }
}
```

Get count of open authorizations

Method: GET

REST URL: /RequestCenter/nsapi/transaction/authorizations/count/AuthViewName=[My Authorizations]|AND|AuthStatus=[Ongoing]
Response:


```
{
        "authorizations": {
                "authorizationsCount": 2
        }
}
```


Gets consumer service category by ID

Returns a service catalog category with the Id specified. Nested entities (subcategories and included services) are fetched. A new API has been introduced to replace the existing category API to improve the performance of the response.

The existing GET API is as follows:

/RequestCenter/nsapi/definition/categories/id/{categoryID}?responseType=json&recursive=true&isNgcRequest=true

Method: GET

REST URL: RequestCenter/nsapi/definition/v2/categories/<Category_ID>? hierarchyLevels=<No of Category Level>&Services=<true/false>
Where,

- hierarchyLevels: number – Returns categories up to a level in the hierarchy using the categoryId passed as the root. When this param isn't passed, all categories under the category ID is returned.

- Services: true/false– when set to true the API returns the serviceIDs of services that belong to the categories returned. When set to false, it will not fetch the serviceIDs.

Java Example –

/RequestCenter/nsapi/definition/v2/categories/137?hierarchyLevels=5&services=true

Sample response:


```
{
        "bottomDescription": "",
        "bottomDescriptionEnabled": false,
        "bottomDescriptionURL": "",
        "catalogType": "Consumer Services Catalog",
        "catalogTypeId": 1,
        "categoryExtensions": [
                {
                        "facetPriorities": "",
                        "facetValues": [],
```

```
                "logicName": "CategoryID",
                "name": "CategoryID",
                "type": "",
                "value": "47"
        },
        {
                "facetPriorities": "",
                "facetValues": [],
                "logicName": "imageUrl",
                "name": "ImageURL",
                "type": "",
                "value": "something"
        }
],
"categoryId": 47,
"categoryName": "C1",
"description": "",
"imageURL": "",
"isRoot": false,
"maxQuantity": 0,
"middleDescription": "",
"middleDescriptionEnabled": false,
"middleDescriptionURL": "",
"serviceIds": [2,3,4],
"showCaseCategoryLocation": "",
"subCategories": [
        {
                "id": 48,
                "logicName": "c2",
                "name": "C2",
                "parentId": 47,
                "serviceCount": 1
        },
        {
                "id": 49,
                "logicName": "c3",
                "name": "C3",
                "parentId": 47,
                "serviceCount": 1
```

```
                }
            ],
            "topDescription": "",
            "topDescriptionEnabled": false,
            "topDescriptionURL": ""
}
```


Get additional information of an open order

Method: GET

REST URL:

/RequestCenter/nsapi/transaction/requisitionentries/<ViewName>?taskDetails=true&currentActivities=true& serviceExtensions=true

Query params:

- taskDetails = true/false – this specifies if the API should fetch the task details for the requisition entries or not
- currentActivities = true/false – when set to true returns only the current tasks for the requisition entries
- serviceExtensions = true/false – specifies whether or not to fetch service extensions for requisition entries

New fields added:

The current ongoing task for the order, Current task id, performer ID, performer first name, last name, Requisition due date, task due date, owner id, owner name, customer id, customer login name, if it was order on behalf, date submitted, how long from due date(how many day left, how many day overdue), service extension for that request

Note: The fields highlighted in grey are newly added to the response below.

Sample response:


```
{
    "requisitionEntries": {
        "startRow": 1,
        "totalCount": 2,
        "recordSize": 2,
        "requisitionEntry": [
            {
                "requisitionEntryId": 2430,
                "requisitionId": 2293,
                "serviceId": 1429,
                "serviceDescription": "",
                "serviceName": "X1 Carbon_Laptop",
                "standardDuration": "Not Defined",
                "unitCost": "$ 0.00",
                "subTotal": "$ 0.00",
                "status": "Ongoing",
                "dueOnDate": "07/18/2018",
                "dueOnDateRaw": 1531960200000,
```

"quantity": 1,
"maxQuantity": 0,
"serviceLevelDescription": "",
"percentageCompleted": 0.0,
"statusId": 1,
"expectedDuration": 0.0,
"reqEntryServiceURL": "X1 Carbon_Laptop",
"reqEntryServiceURLOnly": "/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&reqid=2293&reqentryid=2430&formAction=displayEntryStatus&performerID=&serviceid=1429&requisitionId=2293",
"requisitionStatusId": 1,
"requisitionStatus": "Ongoing",
"startedDate": "07/16/2018",
"startedDateRaw": 1531832982207,
"submitDate": "07/16/2018",
"submitDateRaw": 1531832985850,
"customerName": "Vinay Panchagavi",
"customerId": 1061,
"customerLoginName": "vspancha",
"customerOUId": 30,
"customerEmail": "vspancha@cisco.com",
"reviewsCount": 0,
"selfRating": 0,
"isServiceOrderable": 1,
"isServiceActive": 1,
"isCancelable": 1,
"ownerName": "Vinay Panchagavi",
"ownerId": 1061,
"ownerLoginName": "vspancha",
"dueBy": "2 day(s) overdue",
"taskDetails": [
{
"taskId": 4198,
"taskName": "SGRSiteLevel",
"performerId": 26,
"performerFirstName": "dinesh07",
"performerLastName": "dinesh07",
"taskState": "Under review",
"dueOn": "07/17/2018",

```
                "dueOnRaw": 1531875600000,
                "requisitionEntryId": 2430,
                "requisitionId": 2293,
                "dueBy": "3 day(s) overdue",
                "office": false
            },
            {
                "taskId": 4199,
                "taskName": "SGRSiteLevel_Queue",
                "performerId": 1064,
                "performerFirstName": "ITServices_Queue",
                "performerLastName": "Queue",
                "taskState": "Under review",
                "dueOn": "07/17/2018",
                "dueOnRaw": 1531868400000,
                "requisitionEntryId": 2430,
                "requisitionId": 2293,
                "dueBy": "3 day(s) overdue",
                "office": true
            }

        ],
        "orderOnBehalf": false
    }
  ]
}
```


Get additional Information of an Closed Order

Method: GET

REST URL:

/RequestCenter/nsapi/transaction/requisitionentries/<ViewName>|AND|Status=AllCompleted?
taskDetails=true&currentActivities=true& serviceExtensions=true

Query params:

- taskDetails = true/false – this specifies if the API should fetch the task details for the requisition entries or not
- currentActivities = true/false – when set to true returns only the current tasks for the requisition entries
- serviceExtensions = true/false – specifies whether or not to fetch service extensions for requisition entries

New fields added:

The current ongoing task for the order, Current task id, performer ID, performer first name, last name, Requisition due date, task due date, owner id, owner name, customer id, customer login name, if it was order on behalf, date submitted, how long from due date(how many day left, how many day overdue), service extension for that request

Note: The fields highlighted in grey are newly added to the response below.

Sample response:


```
{
    "requisitionEntries": {
        "startRow": 1,
        "totalCount": 1,
        "recordSize": 1,
        "requisitionEntry": [
            {
                "requisitionEntryId": 41,
                "requisitionId": 7,
                "serviceId": 147,
                "serviceDescription": "",
                "serviceName": "order for others",
                "standardDuration": "Not Defined",
                "unitCost": "$ 0.00",
                "subTotal": "$ 0.00",
                "status": "Ongoing",
                "dueOnDate": "07/02/2018",
                "dueOnDateRaw": 1530527400000,
                "quantity": 1,
                "maxQuantity": 0,
                "serviceLevelDescription": "",
                "percentageCompleted": 0.0,
                "statusId": 1,
                "expectedDuration": 0.0,
                "reqEntryServiceURL": "<a href=\"#\" onclick=\"GB_showFullScreen('order for
others','/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&layout=popup_p&reqid=7&reqentryid=41&formAction
=displayEntryStatus&performerID=&serviceid=147&requisitionId=7')\">order for others</a>",
                "reqEntryServiceURLOnly":
"/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&reqid=7&reqentryid=41&formAction=displayEntryStatus&pe
rformerID=&serviceid=147&requisitionId=7",
                "requisitionStatusId": 1,
                "requisitionStatus": "Ongoing",
                "startedDate": "07/01/2018",
                "startedDateRaw": 1530480343000,
                "submitDate": "07/01/2018",
                "submitDateRaw": 1530480542000,
                "customerName": "user2 user2",
                "customerId": 6,
                "customerLoginName": "user2",
```

```
                "customerOUId": 21,
                "customerEmail": "user2@cisco.com",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "isCancelable": 1,
                "ownerName": "user1 user1",
                "ownerId": 5,
                "ownerLoginName": "user1",
                "daysToDueOnDate": "1 day(s) overdue",
                "taskDetails": [
                    {
                        "taskId": 5,
                        "taskName": "Monitor plan for order for others",
                        "performerId": 2,
                        "performerFirstName": "Default Service Delivery",
                        "performerLastName": "Queue",
                        "taskState": "Ongoing",
                        "dueOn": "07/02/2018",
                        "dueOnRaw": 1530527400000,
                        "requisitionEntryId": 41,
                        "requisitionId": 7,
                        "daysToDueDate": "1 day(s) overdue",
                        "office": true
                    }
                ],
                "orderOnBehalf": true
            }
        ]
    }
}
```


New APIs

Gets Requisitions to be archived or purged
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition?hasWorkFlowData=true
Sample response:


```
{
```

```
"requisitions": {
        "startRow": 1,
        "totalCount": 1,
        "recordSize": 1,
        "requisition": [
                {
                        "tenantId": 0,
                        "userId": 0,
                        "ownerId": 0,
                        "serviceId": 0,
                        "customerId": 10645,
                        "expectedDuration": 0.0,
                        "actualDuration": 0.0,
                        "expectedCost": 0.0,
                        "status": "Closed",
                        "requisitionId": 535192,
                        "lateFlag": false,
                        "statusId": 2,
                        "organizationalUnitId": 0,
                        "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=535192&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">535192</a>",
                        "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=535192",
                        "milestoneLink": "",
                        "percentageCompleted": 0.0,
                        "isCancelable": false,
                        "reviewsCount": 0,
                        "selfRating": 0,
                        "isServiceOrderable": 0,
                        "isServiceActive": 0,
                        "hasAttachment": false,
                        "teamId": 0,
                        "createdDate": "12/09/2010 7:39 下午"
                }                       ]
        }
}
```


Archive or purge requisitions

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/requisitions/workflowdata?requisitioncount={recordSize}

Where, recordSize is number of requisitions that must be archived or purged based on the *Purge/Archive Workflow Data* setting in **Administrations > Settings**.

Sample response:


```
{
    "nsapi-error-response": {
        "status-messages": [
            {
                "code": "NSAPI_ERROR_004",
                "value": " Requisition Purged successfully"
            }
        ]
    }
}
```


Return for review

Method: POST

REST URL: /RequestCenter/nsapi/transaction/tasks/{taskID}/returnforreview

Sample response:


```
<nsapi-response>Task successfully Returned to Review.</nsapi-response>
```


Get Key Fields for the specified requisition

Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitions/id/{requisitionID}/requisitiondata?fetchKeyFieldsOnly=true

Query params

- requisitiondata = true/false – this specifies if the API should fetch the service form fields of requisition entries in the requisition
- fetchKeyFieldsOnly = true/false – this specifies whether to fetch only keyFields for this requisition or not and should be used when requisitiondata = true

Sample response:


```
{
    "RequisitionEntries": [
        {
            "RequisitionEntry": {
                "ServiceName": "WebService1",
                "FormFields": [
                    {
                        "DictionaryName": "WebServiceDictionary1",
                        "WebServiceDictioanry1.FirstName": "admin",
                        "mandatory": false
                    },
                    {
```

```
                                                    "WebServiceDictioanry1.LastName": "kewl",
                                                    "DictionaryName": "WebServiceDictioanry1",
                                                    "mandatory": false
                                            }
                                    ],
                                    "RequisitonEntryID": 663571,
                                    "Quantity": 1
                            }
                    }
            ]
}
```


Get Key Fields for the specified requisition with requisition data

Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitions/id/{requisitionID}/?requisitiondata=true&fetchKeyFieldsOnly=true

Sample response:


```
{
        "requisition": {
                "actualDuration": 0,
                "attachmentImage": "",
                "closedDate": "",
                "closedDateRaw": null,
                "createdDate": "",
                "customerEmail": "sanjaymalligere@cisco.com",
                "customerId": 228075,
                "customerName": "SanjayMalligere Shankar",
                "customerWorkPhone": "",
                "dueDate": "07/06/2018 1:00 PM",
                "dueDateRaw": {
                        "date": 6,
                        "day": 5,
                        "hours": 13,
                        "minutes": 0,
                        "month": 6,
                        "seconds": 0,
                        "time": 1530862200000,
                        "timezoneOffset": -330,
                        "year": 118
                },
```

```
"expectedCost": 0,
"expectedDuration": 0,
"flagImage": "",
"hasAttachment": false,
"isCancelable": true,
"isServiceActive": 1,
"isServiceOrderable": 0,
"lateFlag": false,
"milestoneLink": " ",
"organizationalUnitId": 15272,
"organizationalUnitName": "SanjayMalligere_OU",
"ownerId": 228075,
"ownerName": "SanjayMalligere Shankar",
"percentageCompleted": 0,
"rating": "0",
"requisitionData": [
        {
                "RequisitionEntry": {
                        "ServiceName": "SanjayWebService",
                        "FormFields": [
                                {
                                        "DictionaryName": "SanjayWebServiceDictioanry",
                                        "SanjayWebServiceDictioanry.FirstName": "admin",
                                        "mandatory": false
                                }
                        ],
                        "RequisitonEntryID": 663571,
                        "Quantity": 1
                }
        }
],
"requisitionId": 671774,
"requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=671774&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">671774<\/a>",
"requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=671774",
"reviewsCount": 0,
"selfRating": 0,
"serviceId": 2824,
"serviceImage": "",
```

```
                "serviceName": "SanjayWebService",
                "services": [],
                "startedDate": "07/06/2018 7:17 AM",
                "startedDateRaw": {
                        "date": 6,
                        "day": 5,
                        "hours": 7,
                        "minutes": 17,
                        "month": 6,
                        "seconds": 26,
                        "time": 1530841646260,
                        "timezoneOffset": -330,
                        "year": 118
                },
                "status": "Ongoing",
                "statusId": 1,
                "submitDate": "07/06/2018 7:18 AM",
                "submitDateRaw": {
                        "date": 6,
                        "day": 5,
                        "hours": 7,
                        "minutes": 18,
                        "month": 6,
                        "seconds": 34,
                        "time": 1530841714787,
                        "timezoneOffset": -330,
                        "year": 118
                },
                "teamDisplayName": "",
                "teamId": 0,
                "tenantId": 1,
                "totalCost": "0.0",
                "userId": 228075
        }
}
```


Get all the requisition entries Ordered For MySelf and Ordered-For-Other
Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitionentries/ViewName=Orders with my involvement
Path param:

ViewName=Orders with my involvement - this view returns all requisition entries that the user was involved in i.e., requisitions ordered by the user for himself/others and that were ordered by others for the user.

Sample Response:


```
{
    "requisitions": {
        "startRow": 1,
        "totalCount": 4,
        "recordSize": 4,
        "lastUpdatedDateRaw": 1529935200000,
        "lastUpdatedDate": "06/25/2018 7:00 AM",
        "requisition": [
            {
                "tenantId": 1,
                "userId": 31,
                "ownerId": 31,
                "serviceId": 183,
                "customerId": 31,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "startedDate": "06/21/2018 12:01 PM",
                "dueDate": "06/21/2018 4:00 PM",
                "expectedCost": 0.0,
                "status": "Ongoing",
                "requisitionId": 1279,
                "flagImage": "/RequestCenter/images/flaglate.gif",
                "lateFlag": true,
                "customerName": "OOBUser2 User2",
                "organizationalUnitName": "OOB_OU2",
                "submitDate": "06/21/2018 12:03 PM",
                "statusId": 1,
                "serviceName": "SanjayDraftService",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 45,
                "startedDateRaw": 1529607719487,
                "dueDateRaw": 1529622000000,
                "submitDateRaw": 1529607785807,
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=1279&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">1279</a>",
```

```
                                        "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=1279",
                                        "milestoneLink": " ",
                                        "percentageCompleted": 0.0,
                                        "customerEmail": "oobuser2@cisco.com",
                                        "isCancelable": true,
                                        "rating": "0",
                                        "reviewsCount": 0,
                                        "selfRating": 0,
                                        "isServiceOrderable": 0,
                                        "isServiceActive": 1,
                                        "hasAttachment": false,
                                        "totalCost": "0.0",
                                        "teamId": 0
                        ]
                }
}
```


Get search facets with priority

Method: GET

REST URL: RequestCenter/nsapi/definition/searchFacets

Sample response:


```
{
        "SearchFacetList": {
                "searchFacets": [
                        {
                                "qualifierName": "Search Facets",
                                "attributeType": 7,
                                "facetValues": [
                                        "OneAndAbove",
                                        "TwoAndAbove",
                                        "ThreeAndAbove",
                                        "FourAndAbove",
                                        "None"
                                ],
                                "name": "FctAvgRating",
                                "caption": "Ratings",
                                "dataTypeCompositionID": 1220,
                                "multiValue": true,
                                "priority": 6
                        },
                        {
                                "qualifierName": "Search Facets",
                                "attributeType": 6,
                                "facetValues": [
                                        "Bangalore",
                                        "Sanjose",
                                        "Ottawo"
                                ],
                                "name": "City",
                                "caption": "City",
                                "dataTypeCompositionID": 1329,
```

```
                                        "multiValue": false,
                                        "priority": 7
                                }
]
        }
}
```


Update priority of search facets

Method: PUT

REST URL: /RequestCenter/nsapi/definition/searchFacets

Request Body:


```
{
   "SearchFacetList": {
      "searchFacets": [
         {

            "name": "FctAvgRating",
            "priority": 2
         },
         {
            "name": "test_facet_1",
            "priority": 3
         }
      ]
   }
}
```


New Service Manager APIs

Get Tasks

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks

Sample Response:


```
{
        "TaskList": {
                "startRow": 1,
                "totalCount": 13,
                "recordSize": 13,
                "tasks": [
                        {
                                "taskId": 313,
                                "taskName": "DeptAuth",
                                "taskNameSid": "DeptAuth",
                                "dueOnRaw": 1532082600000,
                                "scheduledStartRaw": 1532082600000,
                                "scheduledCompleteRaw": 1532082600000,
```

"completedOnRaw": 1532056932000,
"startedOnRaw": 1532056520000,
"dueOn": "07/20/2018 4:00 PM",
"requisitionId": 51,
"priority": "prioritynormalicon.gif",
"performerName": "Sanjay Malligere",
"performerId": 24,
"scheduledStart": "07/20/2018 4:00 PM",
"startedOn": "07/20/2018 8:45 AM",
"scheduledComplete": "07/20/2018",
"completedOn": "07/20/2018 8:52 AM",
"computeEarliestDate": "07/20/2018 9:15 AM",
"serviceId": 0,
"initiator": "admin admin",
"initiatorId": 1,
"customer": "admin admin",
"customerId": 1,
"customerOuId": 1,
"status": "Approved",
"effort": 0.0,
"taskType": "Basic approval",
"ouName": "Site Administration",
"queueId": 0,
"taskURL": "DeptAuth",
"taskURLOnly": "/RequestCenter/servicemanager/navigate.do?query=task&taskID=313",
"expectedDuration": 0.0,
"lateFlag": false,
"displayOrder": 0,
"parentTask": 0,
"stateId": 7,
"expanded": true,
"activityTypeId": 2,
"projectActivityID": 0,
"creatorObjectID": 57,
"treeLevel": 0,
"hasAttachment": false,
"requisitionEntryId": 0,
"flagId": 0,

```
                    "officeId": 0,
                    "priorityTypeId": 2,
                    "followUp": "Red Flag",
                    "isPerformer": true,
                    "priorityName": "Normal"
                }



            ]
        }
}
```


Get Task count API for objectType

Method: GET

REST URL: /RequestCenter/nsapi/directory/organizationalunits?type=serviceTeam&viewSM=true

Sample Response:


```
    {
    "OrganizationalUnitList": {
        "startRow": 6,
        "totalCount": 77,
        "recordSize": 20,
        "organizationalUnits": [
            {
                "organizationalUnitName": "BAT_OU",
                "organizationalUnitId": 57,
                "parentId": 0,
                "itemGroupId": 0,
                "itemInstanceId": 0,
                "organizationalUnitTypeId": 1,
                "organizationalUnitType": "Service Team",
                "statusId": 1,
                "status": "Active",
                "managerId": 0,
                "isBillable": "false",
                "accountId": 0,
                "taskCount": 2,
                "orgUnitURL": "<a
href='/RequestCenter/organizationdesigner/scnavigate.do?displayRecordId=57&id=57&query=search&resetBN=true&formAction=display&displayRec=Y&forwardPage=organizationalunits&forwardTo=showSearchGeneralSuccess&mdicontentPortlet=portlet.sc.ou.g
```

eneral&mdicomponentsPortlet=portlet.sc.ou.mdi&selectMDI=sc.ou.general&isCreate=Y&sFlId=Y&layout=popup_p' onclick=\"return GB_showFullScreen('Organizational Unit', this.href)\">BAT_OU",

"orgUnitURLOnly": "/RequestCenter/organizationdesigner/scnavigate.do?displayRecordId=57&id=57&query=search&resetBN=true&formAction=display&displayRec=Y&forwardPage=organizationalunits&forwardTo=showSearchGeneralSuccess&mdicontentPortlet=portlet.sc.ou.general&mdicomponentsPortlet=portlet.sc.ou.mdi&selectMDI=sc.ou.general&isCreate=Y&sFlId=Y"

}

}

Get queue details

Method: GET

REST URL: /RequestCenter/nsapi/directory/v2/queue

Sample Response:

{

"QueueList": {

"startRow": 1,

"totalCount": 18,

"recordSize": 18,

"queue": [

{

"id": 2,

"name": "Default Service Delivery",

"homeOrganizationalUnitId": 1,

"notes": "Queue for unassigned work. This queue is also used for calculating the Due Dates for a service if the service designer has chosen the \"Approximate using Standard Duration\" option.",

"placeId": 0,

"recordStateId": 1,

"status": 0,

"tenantId": 1,

"timeZoneId": 256,

"timeZoneDisplayName": "(GMT-08:00) Pacific Time (US and Canada), Tijuana",

"taskCount": 58,

"shared": false

}

]

}

Gets Calendar Details

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/calendar

Sample Response:


```
{
    "CalendarList": {
        "startRow": 1,
        "totalCount": 13,
        "recordSize": 13,
        "calendar": [
            {
                "calendarEntryID": 122,
                "subject": "Calendar Task",
                "calendarDate": 1517443200000,
                "startTime": 1517484600000,
                "endTime": 1517513400000,
                "sequence": 0,
                "scheduledObjectID": 2,
                "scheduledObjectInstID": 1,
                "eventObjectID": 23,
                "eventObjectInstID": 483,
                "isRead": 0,
                "orgCalendarDate": "02/01/2018 5:30",
                "orgStartDate": "02/01/2018 17:00",
                "orgEndDate": "02/02/2018 1:00",
                "requisitionId": 133,
                "guid": "586AAC4C-59B4-422A-BDF7-AB93AFA01C68",
                "priority": 2
            },
            {
                "calendarEntryID": 117,
                "subject": "Calendar Task",
                "calendarDate": 1517443200000,
                "startTime": 1517522400000,
                "endTime": 1517549250000,
                "sequence": 0,
                "scheduledObjectID": 2,
                "scheduledObjectInstID": 1,
                "eventObjectID": 23,
                "eventObjectInstID": 475,
                "isRead": 0,
                "orgCalendarDate": "02/01/2018 5:30",
```

```
        "orgStartDate": "02/02/2018 3:30",
        "orgEndDate": "02/02/2018 10:57",
        "requisitionId": 129,
        "guid": "32DC97C9-B8D1-434F-BDDF-F25F0D03CF6D",
        "priority": 2
    },
    {
        "calendarEntryID": 116,
        "subject": "Calendar Task",
        "calendarDate": 1517443200000,
        "startTime": 1517455650000,
        "endTime": 1517464800000,
        "sequence": 0,
        "scheduledObjectID": 2,
        "scheduledObjectInstID": 1,
        "eventObjectID": 23,
        "eventObjectInstID": 475,
        "isRead": 0,
        "orgCalendarDate": "02/01/2018 5:30",
        "orgStartDate": "02/01/2018 8:57",
        "orgEndDate": "02/01/2018 11:30",
        "requisitionId": 129,
        "guid": "7DE22DA3-C426-498B-BA82-BB30096B036A",
        "priority": 2
    },
    {
        "calendarEntryID": 120,
        "subject": "Calendar Task",
        "calendarDate": 1517443200000,
        "startTime": 1517484600000,
        "endTime": 1517513400000,
        "sequence": 0,
        "scheduledObjectID": 2,
        "scheduledObjectInstID": 1,
        "eventObjectID": 23,
        "eventObjectInstID": 479,
        "isRead": 0,
        "orgCalendarDate": "02/01/2018 5:30",
        "orgStartDate": "02/01/2018 17:00",
```

```
                "orgEndDate": "02/02/2018 1:00",
                "requisitionId": 131,
                "guid": "7E81885A-0394-4E88-A7FA-A33BF3A156F6",
                "priority": 2
            },
            {
                "calendarEntryID": 123,
                "subject": "Calendar Task",
                "calendarDate": 1517443200000,
                "startTime": 1517460722000,
                "endTime": 1517484600000,
                "sequence": 0,
                "scheduledObjectID": 2,
                "scheduledObjectInstID": 1,
                "eventObjectID": 23,
                "eventObjectInstID": 485,
                "isRead": 0,
                "orgCalendarDate": "02/01/2018 5:30",
                "orgStartDate": "02/01/2018 10:22",
                "orgEndDate": "02/01/2018 17:00",
                "requisitionId": 134,
                "guid": "09365C3D-0647-40DA-AA3D-C1E8D8A8FB2C",
                "priority": 2
            }
        ]
    }
}
```


Export Calendar Details

Method: GET

REST URL: RequestCenter/nsapi/transaction/v2/tasks/calendar/export/{calendarEntryId}

Sample Payload:


```
BEGIN:VCALENDAR
PRODID:Microsoft CDO for Microsoft Exchange
VERSION:2.0
METHOD:PUBLISH
BEGIN:VEVENT
DTSTAMP:20180201T053000
```

DTSTART:20180202T033000

DTEND:20180202T105730

UID:{32DC97C9-B8D1-434F-BDDF-F25F0D03CF6D}

SUMMARY:Calendar Task

DESCRIPTION:Perform Work On Calendar Task

SEQUENCE:0

PRIORITY:2

CLASS:Personal

STATUS:CONFIRMED

TRANSP:OPAQUE

X-MICROSOFT-CDO-BUSYSTATUS:BUSY

X-MICROSOFT-CDO-INSTTYPE:0

END:VEVENT

END:VCALENDAR

Gets instructions for multiple tasks

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/instructions?taskIds= {taskIDs}

Java Example:

http://<ServerURL>/RequestCenter/nsapi/transaction/v2/tasks/instructions?taskIds=203,204

Response


```
{
   "Map": {
      "203": {},
      "204": {
         "instruction": "test instructions",
         "instructionURL": "http://cisco.com",
         "instructionURLDescription": "test desc"
      }
   }
}
```


Gets information of multiple tasks

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/info?taskIds={taskIDs}

Java Example:

http://localhost:8080/RequestCenter/nsapi/transaction/v2/tasks/info?taskIds=201,202

Sample response:-


```
{
   "Map": {
      "201": {
```

```
    "taskTypeId": 1,
    "taskAgentId": 0,
    "projectTaskId": 199,
    "parentTaskId": 0,
    "ticketObjectId": 37,
    "creatorObjectId": 46,
    "creatorObjectInstId": 289,
    "showEffort": true,
    "hasChecklist": true,
    "hasInstructions": true,
    "initTab": "Tab2",
    "availableActions": [
      {
        "actionID": 2,
        "actionName": "Done",
        "isDefault": true
      },
      {
        "actionID": 8,
        "actionName": "Cancel",
        "isDefault": false
      },
      {
        "actionID": -4,
        "actionName": "Assign",
        "isDefault": false
      }
    ],
    "readOnlyReq": false,
    "priceTask": false
  },
  "202": {
    "taskTypeId": 4,
    "taskAgentId": 0,
    "projectTaskId": 0,
    "parentTaskId": 0,
    "ticketObjectId": 37,
    "creatorObjectId": 21,
    "creatorObjectInstId": 161,
    "showEffort": true,
```

```
        "hasChecklist": false,
        "hasInstructions": false,
        "initTab": "Tab2",
        "availableActions": [
          {
            "actionID": 17,
            "actionName": "Done",
            "isDefault": true
          },
          {
            "actionID": 109,
            "actionName": "Cancel Plan",
            "isDefault": false
          },
          {
            "actionID": -1,
            "actionName": "Check out",
            "isDefault": false
          },
          {
            "actionID": -4,
            "actionName": "Assign",
            "isDefault": false
          }
        ],
        "readOnlyReq": false,
        "priceTask": false
      }
   }
}
```


Gets checklist information of multiple tasks

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/checklist?taskIds={taskIDs}

Java Example:

http://<ServerURL>/RequestCenter/nsapi/transaction/v2/tasks/deliveryprocess?taskIds =176,177

Sample Response:


```
{
```

```
  "Map": {
    "201": [
      {
        "selected": true,
        "stepID": "1",
        "stepDescription": "check1",
        "lastPerson": "admin admin",
        "lastDate": "01/21/2018",
        "ismandatory": true
      },
      {
        "selected": true,
        "stepID": "2",
        "stepDescription": "check2",
        "lastPerson": "soauser5 soauser5",
        "lastDate": "01/23/2018",
        "ismandatory": true
      }
    ],
    "202": []
  }
}
```


Delivery Process

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/deliveryprocess?taskIds={taskIDs}

Response:-


```
{
  "Map": {
    "176": [
      {
        "activityId": 177,
        "activity": {
          "map": {
            "ActivityTypeID": "6",
            "AckOn": null,
            "AgentID": "4",
            "ActivityFormID": "4"
          }
        },
```

```
        "customerName": "Default Service Delivery Queue",
        "performerName": "admin admin",
        "stateName": "Completed",
        "icon": "task-completed.gif",
        "subject": "task1",
        "dueOn": "01/17/2018 07:00 PM",
        "completedOn": "01/16/2018 02:18 PM",
        "parentActivityID": 0,
        "startDate": "01/16/2018 01:59 PM",
        "scheduledStart": "01/16/2018 05:00 PM",
        "depth": 0,
        "hasChildren": false,
        "isLast": "false",
        "group": 0,
        "waiting": false
    }
],
"177": [
    {
        "activityId": 177,
        "activity": {
            "map": {
                "ActivityTypeID": "6",
                "AckOn": null,
                "AgentID": "4",
                "ActivityFormID": "4"
            }
        },
        "customerName": "Default Service Delivery Queue",
        "performerName": "admin admin",
        "stateName": "Completed",
        "icon": "task-completed.gif",
        "subject": "task1",
        "dueOn": "01/17/2018 07:00 PM",
        "completedOn": "01/16/2018 02:18 PM",
        "parentActivityID": 0,
        "startDate": "01/16/2018 01:59 PM",
        "scheduledStart": "01/16/2018 05:00 PM",
        "depth": 0,
```

```
            "hasChildren": false,
            "isLast": "false",
            "group": 0,
            "waiting": false
        }
    ]
  }
}
```


SM - TASK Operations API

Get Task Available Actions
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}


```
{
        "TaskData": {
                "taskTypeId": 3,
                "taskAgentId": 0,
                "projectTaskId": 0,
                "parentTaskId": 0,
                "ticketObjectId": 37,
                "creatorObjectId": 57,
                "creatorObjectInstId": 58,
                "showEffort": true,
                "hasChecklist": false,
                "hasInstructions": false,
                "initTab": "Tab2",
                "availableActions": [
                        {
                                "actionID": 119,
                                "actionName": "OK",
                                "isDefault": true
```

```
                },
                {
                        "actionID": -1,
                        "actionName": "Check out",
                        "isDefault": false
                },
                {
                        "actionID": -4,
                        "actionName": "Assign",
                        "isDefault": false
                }
        ],
        "readOnlyReq": false,
        "priceTask": false
    }
}
```


Get Staffing Details for a specified task:

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/staffing


```
{
    "List": [
        {
            "roleID": 54520,
            "roleName": "CSCC",
            "participant": 1,
            "taskList": [
                {
                    "performer": "Customer Service Call Center Queue",
                    "performerID": 12,
```

```
                                        "taskID": 55601,
                                        "taskSubject": "SAP Power Tools Purchasing - Change",
                                        "customerID": 0
                                    }
                                ]
                            }
                    ]
}
```


Gets staffing information of multiple tasks

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/staffing?taskIds={taskIDs}

Java Example: http://<ServerURL>/RequestCenter/nsapi/transaction/v2/tasks/staffing?taskIds=161

Sample Response:


```
{
   "Map": {
      "50": [],
      "161": [
         {
            "roleID": 169,
            "roleName": "perf1",
            "participant": 1,
            "taskList": [
               {
                  "performer": "Default Service Delivery Queue",
                  "performerID": 2,
                  "taskID": 162,
                  "taskSubject": "siTask1",
                  "customerID": 0
               }
            ]
         },
         {
            "roleID": 170,
            "roleName": "perf2",
            "participant": 0,
            "taskList": [
               {
                  "performerID": 0,
```

```
                "taskID": 162,
                "taskSubject": "siTask1",
                "customerID": 11,
                "customerName": "soa user4"
            }
        ]
    },
    {
        "roleID": 171,
        "roleName": "perf3",
        "participant": 1,
        "taskList": [
            {
                "performer": "Vivek Verma",
                "performerID": 8,
                "taskID": 163,
                "taskSubject": "task2",
                "customerID": 0
            }
        ]
    },
    {
        "roleID": 172,
        "roleName": "perf4",
        "participant": 0,
        "taskList": [
            {
                "performerID": 0,
                "taskID": 163,
                "taskSubject": "task2",
                "customerID": 2,
                "customerName": "Default Service Delivery Queue"
            }
        ]
    }
  ]
 }
}
```

Update Performer

Method: PUT

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/staffing

Sample response:


```
{
        "staffing":{
      "participant": 1,
      "taskList": [
        {
          "performerID": 2,
          "taskID": 162
        }
      ]
    }
}
```


Get performer List

Method: GET

REST URL: /RequestCenter/nsapi/transaction/performerlist?name=*&startRow=1&recordSize=10&responseType=json


```
{
  "performerList": {
    "startRow": 1,
    "totalCount": 5,
    "recordSize": 10,
    "performers": [
      {
        "id": 4,
        "firstName": "ProviderBusinessTenant_admin",
        "lastName": "Queue",
        "homeOrganizationalUnitName": "ProviderBusinessTenant_admin"
      },
      {
        "id": 32,
        "firstName": "Team2_admin",
        "lastName": "Queue",
        "homeOrganizationalUnitName": "Team2_admin"
      },
      {
        "id": 1,
        "firstName": "admin",
```

```
      "lastName": "admin",
      "homeOrganizationalUnitName": "Site Administration"
    },
    {
      "id": 22,
      "firstName": "nagadmin",
      "lastName": "nagadmin",
      "homeOrganizationalUnitName": "NagTestOU"
    },
    {
      "id": 77,
      "firstName": "shashi3",
      "lastName": "admin2",
      "homeOrganizationalUnitName": "Site Administration"
    }
   ]
 }
}
```


Update Supervisor

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/staffing


```
{
        "staffing":{
      "participant": 0,
      "taskList": [
        {
          "customerD": 2,
          "taskID": 162
        }
      ]
    }
}
```


Get Requisition Data

Method: GET

REST URL:
/RequestCenter/nsapi/transaction/v2/requisitions/requisitiondata?requisitionIds=2&fetchDictionaryCaption=true&utid=18ffaec697217d06ff7e519f838805c6&responseType=json
Sample response:


```
{
  "Requisitions": [
    {
      "RequisitionId": 2,
      "RequisitionEntries": [
        {
          "RequisitionEntry": {
            "Dictionaries": [],
            "ServiceName": "testSTService1",
            "FormFields": [],
            "RequisitonEntryID": 2,
            "Quantity": 1
          }
        }
      ]
    }
  ]
}
```


Task Checklist

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/task/{taskID}/checklist

Sample response


```
{
   "List":[
      {
         "selected":false,
         "stepID":"1",
         "stepDescription":"new item",
         "ismandatory":true
      },
      {
         "selected":false,
         "stepID":"2",
         "stepDescription":"new item",
         "ismandatory":true
      }
   ]
}
```

Delivery process API

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/deliveryprocess

Sample response:


```
{
    "deliveryProcessList": {
        "startRow": 1,
        "totalCount": 2,
        "recordSize": 20,
        "deliveryProcesses": [
            {
                "activityId": 5830,
                "activity": {
                    "map": {
                        "ActivityTypeID": "6",
                        "AckOn": null,
                        "AgentID": "-1",
                        "ActivityFormID": "4"
                    }
                },
                "customerName": "Default Service Delivery Queue",
                "performerName": "Default Service Delivery Queue",
                "stateId": "3",
                "stateName": "Completed",
                "icon": "task-completed.gif",
                "subject": "Create Si",
```

```
                "dueOn": "07-09-2018 07:57 PM",
                "completedOn": "07-06-2018 05:57 PM",
                "parentActivityID": 0,
                "startDate": "07-06-2018 05:57 PM",
                "scheduledStart": "07-06-2018 05:57 PM",
                "depth": 0,
                "hasChildren": false,
                "isLast": "false",
                "group": 0,
                "waiting": false
        },
        {
                "activityId": 5831,
                "activity": {
                        "map": {
                                "ActivityTypeID": "1",
                                "AckOn": null,
                                "AgentID": "0",
                                "ActivityFormID": "4"
                        }
                },
                "customerName": "Default Service Delivery Queue",
                "performerName": "admin admin",
                "performerRoleName": "DeliveryBoy",
                "stateId": "2",
                "stateName": "Ongoing",
                "icon": "task-ongoing.gif",
```

```
                                        "subject": "DeliveryPlanTask",
                                        "dueOn": "07-11-2018 05:30 AM",
                                        "completedOn": "",
                                        "parentActivityID": 0,
                                        "startDate": "07-06-2018 05:57 PM",
                                        "scheduledStart": "07-10-2018 03:30 AM",
                                        "depth": 0,
                                        "hasChildren": false,
                                        "isLast": "false",
                                        "group": 0,
                                        "waiting": false
                                }
                        ]
                }
}
```


Show skipped task

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/deliveryprocess?showSkippedTasks=true


```
{
        "deliveryProcessList": {
                "startRow": 1,
                "totalCount": 1,
                "recordSize": 20,
                "deliveryProcesses": [
                        {
                                "activityId": 6153,
                                "activity": {
                                        "map": {
```

```
                                        "ActivityTypeID": "1",
                                        "AckOn": null,
                                        "AgentID": "0",
                                        "ActivityFormID": "4"
                                    }
                                },
                                "customerName": "Default Service Delivery Queue",
                                "performerName": "Default Service Delivery Queue",
                                "stateId": "4",
                                "stateName": "Skipped",
                                "icon": "task-skipped.gif",
                                "subject": "Test1",
                                "dueOn": "07-09-2018 04:00 PM",
                                "completedOn": "07-09-2018 06:57 AM",
                                "parentActivityID": 0,
                                "startDate": "07-09-2018 06:57 AM",
                                "scheduledStart": "07-09-2018 04:00 PM",
                                "depth": 0,
                                "hasChildren": false,
                                "isLast": "false",
                                "group": 0,
                                "waiting": false
                            }
                        ]
                    }
}
```

Get pricing details for a Specified task

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/pricing

Sample response:


```
{
        "priceTask": {
                "estimatedPrice": 13000.0,
                "actualPrice": 40700.0,
                "requisitionNumber": 1322,
                "reqEntryNumber": 1400
        }
}
```


Update pricing details for a specified task

Method: PUT

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/pricing

Sample Payload:


```
{ "priceTask": { "estimatedPrice": 40000, "actualPrice": 40700, "requisitionNumber": 1322, "reqEntryNumber": 1400 } }
```


Sample Response:


```
{
        "status-message": {
                "code": "Success",
                "value": "Price is successfully updated."
        }
}
```


Get Requisition Entries by Requisition ID

Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitions/requisitionentries/id/{RequisitionID}?responseType=json


```
{
        "requisitionEntries": {
                "startRow": 1,
                "totalCount": 1,
                "recordSize": 1,
                "requisitionEntry": [
                        {
                                "requisitionEntryId": 663219,
                                "requisitionId": 671445,
                                "serviceId": 2793,
                                "serviceDescription": "CD Performace Test Service",
                                "serviceName": "S1_CD Performace Test Service_V1",
                                "standardDuration": "Not Defined",
                                "unitCost": "$ 0.00",
                                "subTotal": "$ 0.00",
                                "status": "Rejected",
```

```
                                        "dueOnDate": "06/29/2018",
                                        "dueOnDateRaw": 1530264600000,
                                        "quantity": 1,
                                        "maxQuantity": 0,
                                        "serviceLevelDescription": "",
                                        "percentageCompleted": 0.0,
                                        "statusId": 4,
                                        "expectedDuration": 0.0,
                                        "reqEntryServiceURL": "<a href=\"#\" onclick=\"GB_showFullScreen('S1_CD Performace Test
Service_V1','/RequestCenter/myservices/navigate.do?query=vieworderform&layout=popup_p&serviceid=2793&requisitionId=671445&requisitionEntryId=663219')\">S1_CD Performace Test Service_V1</a>",
                                        "reqEntryServiceURLOnly":
"/RequestCenter/myservices/navigate.do?query=requisitionentrystatus&reqid=671445&reqentryid=663219&formAction=displayEntryStatus&performerID=&serviceid=2793&requisitionId=671445",
                                        "requisitionStatusId": 0,
                                        "requisitionStatus": "Preparation",
                                        "customerName": " ",
                                        "customerId": 0,
                                        "customerOUId": 0,
                                        "reviewsCount": 0,
                                        "selfRating": 0,
                                        "isServiceOrderable": 1,
                                        "isServiceActive": 1,
                                        "isCancelable": 0,
                                        "ownerId": 0,
                                        "daysToDueOnDate": "Invalid short date format",
                                        "orderOnBehalf": false
                                    }
                                ]
                        }
}
```


SM - Effort APIs

Gets List of Unit Types
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/efforts/unittypes?responseType=json
Sample response:


```
{
   "entities": {
      "totalCount": 38,
      "hasMore": false,
      "units": [
         {
```

```
    "id": 1,
    "name": "Users",
    "logicName": "unitusers"
},
{
    "id": 2,
    "name": "Keyboards",
    "logicName": "unitkeyboards"
},
{
    "id": 3,
    "name": "Hours",
    "logicName": "unithours"
},
{
    "id": 4,
    "name": "Megabytes",
    "logicName": "unitmb"
},
{
    "id": 5,
    "name": "Servers",
    "logicName": "unitserver"
},
{
    "id": 6,
    "name": "CPUs",
    "logicName": "unitcpu"
},
{
    "id": 7,
    "name": "Departments",
    "logicName": "unitdepartment"
},
{
    "id": 8,
    "name": "Divisions",
    "logicName": "unitdivision"
},
{
    "id": 9,
    "name": "People",
    "logicName": "unitperson"
},
{
    "id": 10,
    "name": "Accounts",
    "logicName": "unitcustomer"
},
{
    "id": 11,
    "name": "Vendors",
    "logicName": "unitvendor"
},
{
    "id": 12,
    "name": "Desktops",
    "logicName": "unitdesktop"
},
{
    "id": 13,
    "name": "Devices",
```

```
    "logicName": "unitdevice"
},
{
    "id": 14,
    "name": "Minutes",
    "logicName": "unitminute"
},
{
    "id": 15,
    "name": "Days",
    "logicName": "unitday"
},
{
    "id": 16,
    "name": "Weeks",
    "logicName": "unitweek"
},
{
    "id": 17,
    "name": "Months",
    "logicName": "unitmonth"
},
{
    "id": 18,
    "name": "Years",
    "logicName": "unityear"
},
{
    "id": 19,
    "name": "Applications",
    "logicName": "unitapplication"
},
{
    "id": 20,
    "name": "Databases",
    "logicName": "unitdatabase"
},
{
    "id": 21,
    "name": "Webservers",
    "logicName": "unitwebserver"
},
{
    "id": 22,
    "name": "Conference calls",
    "logicName": "unitconfcall"
},
{
    "id": 23,
    "name": "Message hours",
    "logicName": "unitmessagehour"
},
{
    "id": 24,
    "name": "Telco ports",
    "logicName": "unittelcoport"
},
{
    "id": 25,
    "name": "Telco lines",
    "logicName": "unittelcoline"
},
```

```
{
   "id": 26,
   "name": "Calls",
   "logicName": "unitcall"
},
{
   "id": 27,
   "name": "Access lines",
   "logicName": "unitaccessline"
},
{
   "id": 28,
   "name": "Kilobytes",
   "logicName": "unitkilobyte"
},
{
   "id": 29,
   "name": "Gigabytes",
   "logicName": "unitgigabyte"
},
{
   "id": 30,
   "name": "Networks",
   "logicName": "unitnetwork"
},
{
   "id": 31,
   "name": "Domains",
   "logicName": "unitdomain"
},
{
   "id": 32,
   "name": "Cubicles",
   "logicName": "unitcubicle"
},
{
   "id": 33,
   "name": "Buildings",
   "logicName": "unitbuilding"
},
{
   "id": 34,
   "name": "SqFeet",
   "logicName": "unitsqfeet"
},
{
   "id": 35,
   "name": "Locations",
   "logicName": "unitlocation"
},
{
   "id": 36,
   "name": "Offices",
   "logicName": "unitoffice"
},
{
   "id": 37,
   "name": "BTUs",
   "logicName": "unitbtu"
},
{
   "id": 38,
```

```
            "name": "Other",
            "logicName": "unitother"
         }
      ]
   }
}
```


Gets List of Category

Method: GET

REST URL: RequestCenter/nsapi/transaction/v2/tasks/efforts/categories?responseType=json

Sample response:


```
{
   "entities": {
      "totalCount": 3,
      "hasMore": false,
      "categories": [
         {
            "billable": false,
            "id": 1,
            "name": "Labor"
         },
         {
            "billable": false,
            "id": 2,
            "name": "Materials"
         },
         {
            "billable": false,
            "id": 3,
            "name": "Other"
         }
      ]
   }
}
```


Get Effort Entries by Task ID

REST URL: RequestCenter/nsapi/transaction/v2/tasks/{taskID}/efforts?responseType=json

Sample response:


```
{
   "entities": {
      "totalCount": 4,
      "strReportDate": "01/31/2018",
      "billingID": 0,
      "totalExpense": "53.00",
      "items": [
         {
            "billingID": 29,
            "refObjectID": 0,
            "refObjectInstID": 0,
            "unitID": 12,
            "quantity": "1.0",
            "pricePerUnit": "50.0",
            "contextObjectID": 0,
            "contextObjectInstID": 0,
            "categoryID": 1,
            "serviceID": 0,
            "personID": 0,
            "areaID": 0,
```

```
    "reportDate": 1517404759000,
    "description": "from sm user1",
    "userID": 0,
    "statusID": 0,
    "rateID": 0,
    "rate": 0,
    "timeSpentMin": 0,
    "strReportDate": "01/31/2018",
    "expense": "50.00",
    "uirow": 0
},
{
    "billingID": 35,
    "refObjectID": 0,
    "refObjectInstID": 0,
    "unitID": 3,
    "quantity": "1.0",
    "pricePerUnit": "1.0",
    "contextObjectID": 0,
    "contextObjectInstID": 0,
    "categoryID": 1,
    "serviceID": 0,
    "personID": 0,
    "areaID": 0,
    "reportDate": 1517417940000,
    "description": "ee",
    "userID": 0,
    "statusID": 0,
    "rateID": 0,
    "rate": 0,
    "timeSpentMin": 0,
    "strReportDate": "01/31/2018",
    "expense": "1.00",
    "uirow": 0
},
{
    "billingID": 36,
    "refObjectID": 0,
    "refObjectInstID": 0,
    "unitID": 3,
    "quantity": "1.0",
    "pricePerUnit": "1.0",
    "contextObjectID": 0,
    "contextObjectInstID": 0,
    "categoryID": 1,
    "serviceID": 0,
    "personID": 0,
    "areaID": 0,
    "reportDate": 1517418000000,
    "description": "dfffdfddffd",
    "userID": 0,
    "statusID": 0,
    "rateID": 0,
    "rate": 0,
    "timeSpentMin": 0,
    "strReportDate": "01/31/2018",
    "expense": "1.00",
    "uirow": 0
},
{
    "billingID": 37,
    "refObjectID": 0,
```

```
            "refObjectInstID": 0,
            "unitID": 3,
            "quantity": "1.0",
            "pricePerUnit": "1.0",
            "contextObjectID": 0,
            "contextObjectInstID": 0,
            "categoryID": 1,
            "serviceID": 0,
            "personID": 0,
            "areaID": 0,
            "reportDate": 1517418060000,
            "description": "ssss",
            "userID": 0,
            "statusID": 0,
            "rateID": 0,
            "rate": 0,
            "timeSpentMin": 0,
            "strReportDate": "01/31/2018",
            "expense": "1.00",
            "uirow": 0
         }
      ]
   }
}
```


Create Effort Entry

Method: POST

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/efforts?responseType=json

Sample payload:


```
{
  "billing": {

    "unitID": 12,
    "categoryID": 1,
    "description": "from sm user1",
    "pricePerUnit": "50.0",
      "quantity": "1.0"
  }
}
```


Sample response:


```
{
   "status-message": {
      "code": "Success",
      "value": "The new Task Effort Entry is successfully added."
   }
}
```


Update Effort Entry

Method: PUT

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/efforts/{billingID}

Sample payload:


```
{
  "billing": {

    "unitID": 17,
    "categoryID": 3,
    "description": "update description",
```

```
    "pricePerUnit": "50.0",
     "quantity": "3"
  }
}
```


Sample response:


```
{
   "status-message": {
      "code": "Success",
      "value": "Task Effort Entry is successfully updated."
   }
}
```


Delete Effort Entry

Method: Delete

REST URL: /RequestCenter/nsapi/transaction/v2/tasks/{taskID}/efforts/{billingID}

Sample response:


```
{
   "status-message": {
      "code": "Success",
      "value": "Task Effort Entry is successfully deleted."
   }
}
```


Quota and Policy for SI Bulk Update

Method: GET

REST URL: /RequestCenter/nsapi/serviceitem/v2/process?isBatch=true&commitType=2&batchSize=50)

If isBatch is not provided or sent as false then following is the behaviour of the API (non bulk operation):-

1. If the policies and quota are enabled for the service item type then these will be run against the service items for create, update operation.
2. Commit type and batchSize settings are not used. The operation is always partial that is all the proper service items will be operated upon and others will be reported as failed.
3. However for update operation and create operation if any policy is violated for a particular service item e.g si9 then operation will not be done for si9 and service items beyond si9.

When commitType is 2 for create operation if any policy is violated for one of the service item s1, then only all the service items before s1 will be created and service items s1 onwards will not be created.

Note: For delete no policy check is executed, however the corresponding attribute count if the policy is sum quota policy or service item count if policy is count quota policy will be updated. This is normal expected behaviour.

Sample response:


```
<serviceitem>
<name>testSIType01</name>
<serviceItemData>
```

```
  <serviceItemAttribute name="Name">D_1</serviceItemAttribute>
  <serviceItemAttribute name="Age">1</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">2451</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max11</serviceItemAttribute>
  <subscription>
    <loginID>testuser01</loginID>
    <ouname>testOU1</ouname>
    <accountName>testTenant1</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
<serviceItemData>
  <serviceItemAttribute name="Name">D_2</serviceItemAttribute>
  <serviceItemAttribute name="Age">2</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max12</serviceItemAttribute>
  <subscription>
    <loginID>testuser01</loginID>
    <ouname>testOU1</ouname>
    <accountName>testTenant1</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
<serviceItemData>
  <serviceItemAttribute name="Name">D_3</serviceItemAttribute>
  <serviceItemAttribute name="Age">1</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max13</serviceItemAttribute>
  <subscription>
    <loginID>testuser01</loginID>
    <ouname>testOU1</ouname>
```

```
    <accountName>testTenant1</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
<serviceItemData>
  <serviceItemAttribute name="Name">D_4</serviceItemAttribute>
  <serviceItemAttribute name="Age">1</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
 <serviceItemAttribute name="Dstringmax">I am string max14</serviceItemAttribute>
  <subscription>
    <loginID>testuser01</loginID>
    <ouname>testOU1</ouname>
    <accountName>testTenant1</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
<serviceItemData>
  <serviceItemAttribute name="Name">D_5</serviceItemAttribute>
  <serviceItemAttribute name="Age">0</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max15</serviceItemAttribute>
  <subscription>
    <loginID>test01</loginID>
    <ouname>testOU1</ouname>
    <accountName>testTenant1</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
</serviceitem>
```

Introduced in 12.1_Patch_v3

UCSD 6.6 Certification

Prime Service Catalog 12.1 Patch V3 is verified and certified with Cisco UCSD 6.6.

CloudCenter Enhancements

From this patch release, Prime Service Catalog supports the following features of CloudCenter:

Service Account Based CloudCenter Integration

CloudCenter integrations are now differentiated as:

- User Account Based Integration:
  CloudCenter connections are associated with the user who created it. If the user deploys an application, then only this user is the owner of the application and can perform actions on the applications. In User based integration, every user is pushed to CloudCenter to create a user record if they do not exist in CloudCenter.

  Note: In case of upgrade to this patch release, any existing cloud center connections prior to upgrade, will be considered as User Account based integration by default.

- Service Account Based integration:
  CloudCenter connection is associated with a common service account which is used for all requests sent to CloudCenter, including application deployments, operations, and actions irrespective of the user who creates it. In this case, users are not pushed to CloudCenter.

  Users with permissions to deploy applications will be the owners of these applications and any user with read-write access on these deployments or VMs is permitted to perform actions on them.

This is a one-time setting per connection when a Site Administrator or Integration Administrator creates a new CloudCenter integration. Go to **Integrations module > New Integration > Cisco CloudCenter**. For the Integration type, select either User Account or Service Account based integration with CloudCenter.

Important: This setting cannot be changed after creation of the integration.

User Account Based Integration

The below features are specific to user based CloudCenter integration.

Import Brownfield Deployments/VMs

Prime Service Catalog now supports importing of deployments and VMs created outside of Prime Service Catalog. These deployments and VMs are discovered by the Prime Service Catalog that are created in CloudCenter and are then displayed in the Service Items page when an integration is created with CloudCenter.

Two new options *Import Brownfield Deployments* and *Import Brownfield VMs* are provided in the Create CloudCenter integration screen. You can choose to import the deployments and/or VMs for that connection during creation. This setting can be edited at any point of time after the creation of the connection.

Note: It is recommended that import brownfield VMs and deployments are disabled for service account based integrations. For a Service Account based integration the behavior of Brownfield VMs import to Prime Service Catalog might be inconsistent. If there are deployments and VMs in CloudCenter, irrespective of whether it is owned by the service account or not, when imported to Prime Service Catalog will be assigned to admin. This would lead to inconsistent behavior as ownership is vital for permissions to perform actions on the VMs in CloudCenter.

Nevertheless, if it is necessary to import brownfield VMs and deployments into Prime Service Catalog then ensure that they are all owned by the service account in CloudCenter.

Associate User for Brownfield Deployments and VMs

The newly added **Map Users** action for the Cloud Center Connection drop down menu, maps the users to the objects imported. The users must be mapped to the integration right after Connect and import. If the user is not mapped to the integration, the Sync Managed VMs operation will fail.

Since the brownfield deployments and VMs are imported to Prime Service Catalog from CloudCenter, the users/owners of these deployments and VMs must also be associated in Prime Service Catalog. The users are handled differently in the following scenarios:

- If user already exists in Prime Service Catalog:

The user can access all the imported deployments owned by the user in the Service Items page. Will be able to perform all operations on the deployments and VMs.

- If the user does not exist in in Prime Service Catalog:

On Import of the deployment, the customer for Service Item is null. Later when the user is created in Prime Service Catalog manually either through LDAP/IDP(SAML)/ Prime Service Catalog database. The imported deployments and VMs get associated to the user.

IMPORTANT: The user creation should be done in a way that the user details maps to the external user ID. It is the onus of the admin to map the appropriate fields in the LDAP/ SAML.

Synch Managed VMs

A new option *Synch Managed VMs* is available for all users in **Service Catalog > Service Items** page. This option reloads and display all the VMs owned by the logged in user. Only the managed VMs that are owned by the user in CloudCenter are synched.

Import Selected Application Profiles

A new option *Re-Import* is introduced to refresh and reimport only the selected application profiles for any CloudCenter connection. If there are any changes made to the application profiles in CloudCenter, on clicking Re-Import for the selected application profiles, the changes made are reflected in Prime Service Catalog and also services are regenerated.

This option is available under **Integrations > CloudCenter Integration > Manage Integrations > Application profiles.**

Enhancements on Deployments

Perform Full-sync on a Deployed Application

Prime Service Catalog performs a full-sync of both Application and VM. The status is derived depending on the status of application or VM on CloudCenter and the details of the VM such as, CPU, RAM, HD, Public and Private IP Address, status on the VM are displayed on the Summary Details page.

Details

Name	42356edf-9dd1-c9a2-e1c0-71bf46e7861f
Display Name	CentOS_2
External ID	42356edf-9dd1-c9a2-e1c0-71bf46e7861f
Status	Active
Host Name	cqjw-51dd6591a
Public IP	172.23.14.76
Private IP	172.23.14.76
Start Time	2018-05-22 06:02:37.403
Cloud Connection ID	1
Cloud VM Type	DEPLOYMENT_VM
Cloud ID	1
Cloud Account ID	1
Cloud Region ID	1
External VM ID	940
External User ID	2
Termination Protection	0
Job ID	1822
Virtual CPUs	1
Memory (MB)	1024
Storage (GB)	10

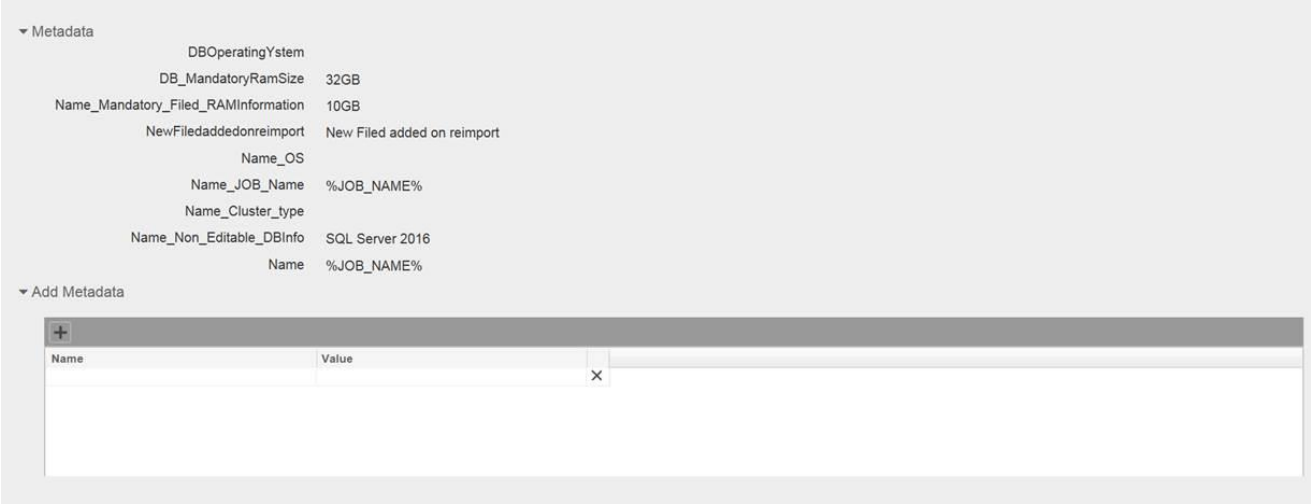
Order CloudCenter Service for a Group

Users with permissions to order CloudCenter deployments can now order a CloudCenter service for a group that the user belongs. Read/Write permission on the ordered Service Item will be provided to the selected Group and all users of the groups can perform actions on the deployment.

A new option “Display Group List in Service Form” in the **Administration > Settings** is introduced, when enabled, the *Group* field is displayed in the Service Form for any of the CloudCenter Service being ordered. The list will contain all the Groups that the user of the Order belongs to. Only the site administrator can enable this option in administration settings.

Import Application Profile Metadata Tags

If the application profiles have any metadata set for custom properties, the metadata are also imported to Prime Service Catalog during importing a CloudCenter connection. The metadata information is then used to build a new dictionary and new service form, and this service form is tied to the CloudCenter application service. While ordering such a service, the user can view the metadata field details and also add additional metadata on the fly. For more details on application profile metadata in CloudCenter, see section *Metadata for Custom Properties* in CloudCenter Documentation.



Note: The dictionary and service forms that are created by the system should not be tampered with. If it is tampered, CloudCenter deployment will fail.

Note: You can add the special characters in the metadata form while ordering CloudCenter services. But for “” special characters, you should add the below escape sequence.

\“System is having 32 GB\”

Override Network while Deploying a CloudCenter Application

You would now be able to override the network on which the CloudCenter application is deployed.

The Network dictionary field is hidden by default. The service designer or any user who has read/write permissions of the service will be able to make this option available for the particular service by following the below procedure:

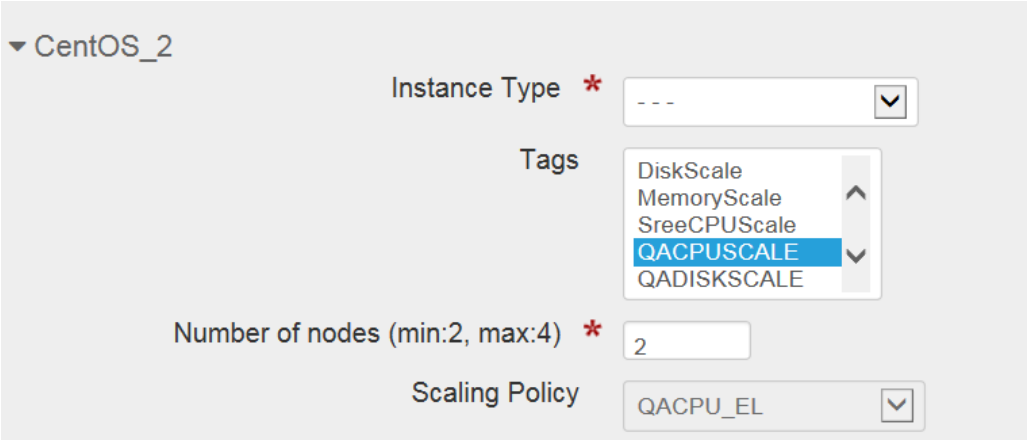
1. Choose **Service Designer > Active Form Component**.
2. Select the form you wish to configure and assuming there are dictionaries already added to the form, click the **Display Properties** tab.
3. From the **Dictionaries Used in This Form** section, expand the dictionary whose fields you want to edit.
4. Click on **Network** dictionary name to display its properties on the right.
5. Change the **Input Type** from hidden to text.
6. **Save**.

Note: If the network option is enabled and this field is left blank, the default network as per the cloud configured is considered.

Support for Scaling Policy

Prime Service Catalog now supports scaling policies defined in CloudCenter. You can associate an application deployment with the scaling policy in CloudCenter by selecting the appropriate system tags while ordering the CloudCenter service that is available on the service form. In the below screen shot, on selecting scaling tag the associated policy is auto populated.

Note: If multiple tags are selected, the scaling policy with higher priority is applied automatically.



New Actions on Deployment

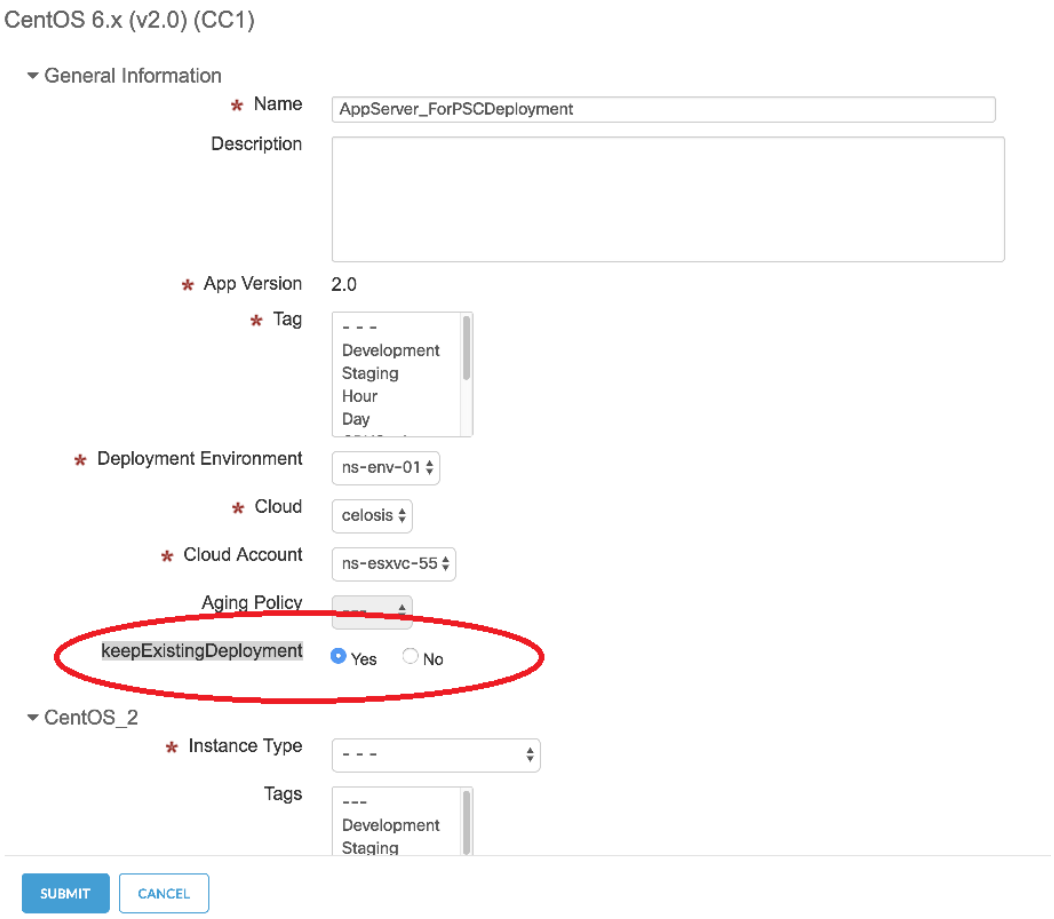
- Sync Deployment

The Sync Deployment action allows you to sync each deployment with CloudCenter and get the current status of particular deployment. For example, if changes are made in the deployments from CloudCenter such as, deployment is suspended or new node is added then, such changes can be synched immediately to the Prime Service Catalog using this action instead of waiting for the specified polling interval or manual import.

- Migrate Deployment

The operation *Migrate Deployment* action triggers migrate deployment operation in Cloud Center.

By default a copy of the deployment is created on the target cloud and the deployment is deleted on the original cloud. But you could choose to keep the deployment on the original cloud by choosing “Yes” for the option keepExistingDeployment on the deployment service form.



Enhancements on VMs

New Actions on Managed VMs

This patch release supports the following new operations on managed VMs:

Note: These operations are available only to the users with read/write permissions on the deployments or VMs for Service Account based integration and the user must be the customer of the VM in a user based integration.

- Start/Stop/Reboot App VM:
  Depending on the status of the VM, you can:
  - Start: Power on and start the VM
  - Stop: Stop and power off the VM
  - Reboot: Attempts a graceful shut down and restart of the VM

- Attach Volume:
  The attach volume option allows you to attach multiple volumes to all tier types in N-tier applications. For each volume, you must specify the size, storage type for each root disk. The attach volume operations invokes a service which can be ordered for others or for oneself. For detailed information see section *Attach Multiple Volumes to Tiers* in CloudCenter Documentation.

- Detach volume:
  The detach volume option is used to remove the disk volumes from the VM for any user.

- Create Snapshot:
  Creates the image snapshot for the specified VM.

- Sync App VM:

This VM option gets the latest information of VM (Unmanaged VM, Deployed VM, and Imported VM) from the CloudCenter to the Prime Service Catalog, instead of waiting for Poller trigger or manual import. Also syncs the deployment details if it is part of the deployment.

Import Unmanaged VMs

There are two kinds of brownfield VMs that can be imported into Prime Service Catalog.

- Managed VMs: VMs that are managed by CloudCenter. For example: VMs that were created as a result of deployments created in CloudCenter. Or VMs that were imported into CloudCenter from a cloud.
- Unmanaged VMs: VMs that exist in the cloud account connected to CloudCenter. These are not imported into CloudCenter.

Users have limited permissions on the Unmanaged VMs, to manage these VMs from Prime Service Catalog VMs, you must first import the VM to Prime Service Catalog.

On the Service Items page, permitted actions on Unmanaged VMs are:

- Terminate
- Import UnManagedVM—Imports the VM to Prime Service Catalog and CloudCenter and converts the VM status to Managed VM.

Console Access of VMs

You can now access the CloudCenter VMs from a web based console from the Service Items page of the VM. User with the Read/Write on a managed VM can access the VM console.

- For a Windows VM:

Click on the **Login Details** button to view the login credentials and then click the **RDP** button. The console is launched on a new webpage.

- For a Linux VM:

Click on the **SSH** button to launch the console on a new webpage.

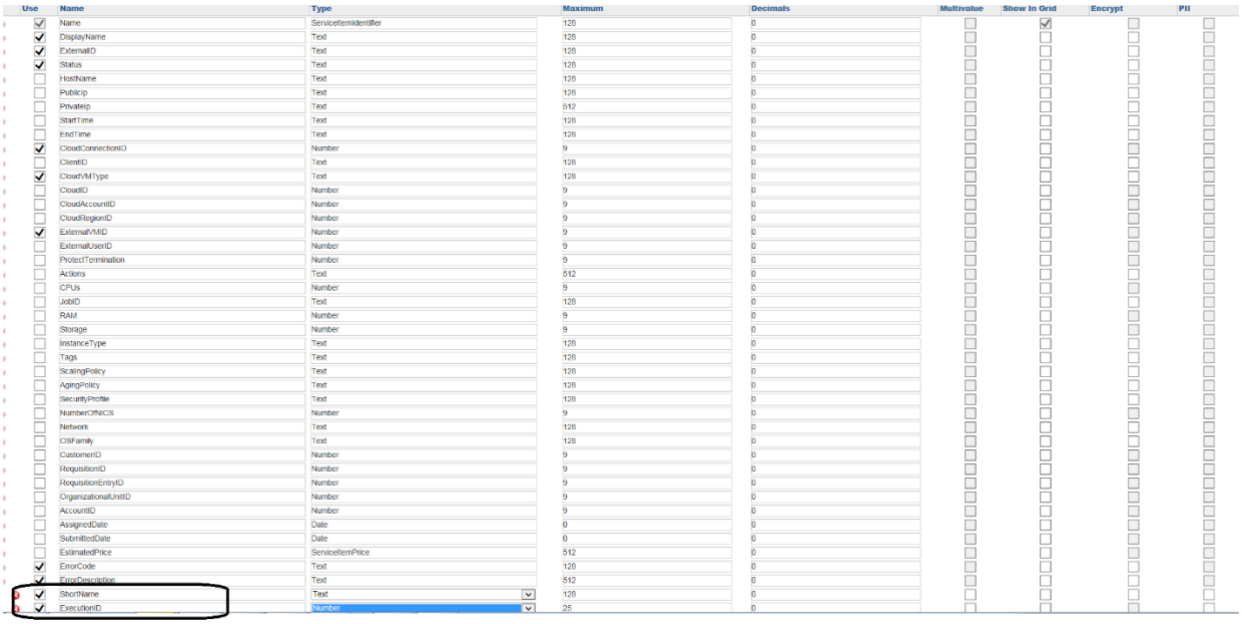
Enhanced Service Items Search

A new search criteria *Display Name* is available to filter the Service Items by display name of the service item.

Support for Custom Lifecycle Operations on VMs

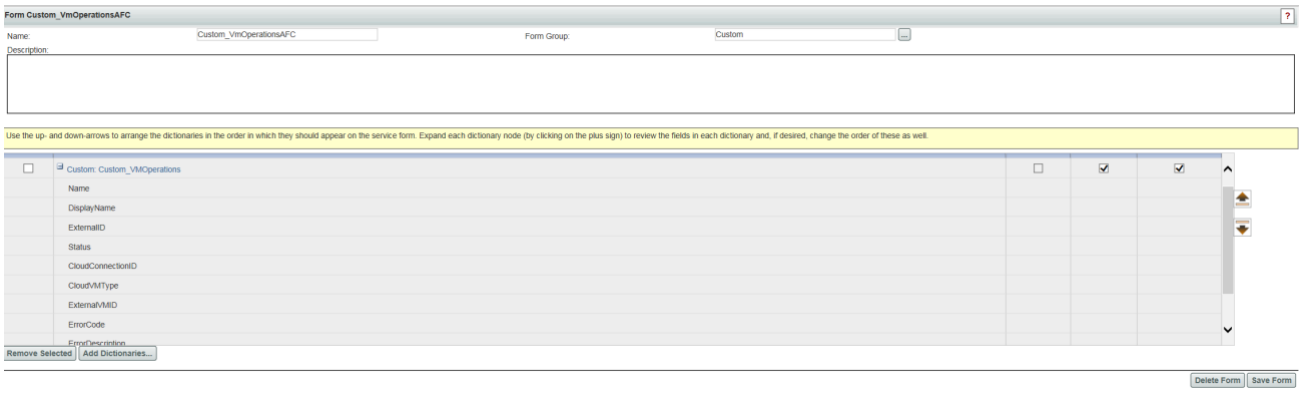
To support custom actions on CloudCenter VMs follow the below procedure:

1. Create a service-item based dictionary by selecting *Application Virtual Machine* as Service item type.
    a. Choose **Service Designer > Dictionaries.**
    b. Choose **New > New Dictionary** to display the New Dictionary page.
    c. In the Add New Internal Dictionary section, in the Service Item field, enter *Application Virtual Machine* and select the service item.
    d. Enter details such as Dictionary Name and Group Name.
    e. From the Dictionary Attributes section, click Add Field to add a user-defined field **ShortName and ExecutionID**.
    f. Select all the fields in the dictionary that is required for the custom action (Any fields that are used in custom FTL). Make sure that the ExecutionID field is added to the dictionary.
    g. Click **Save Dictionary**.

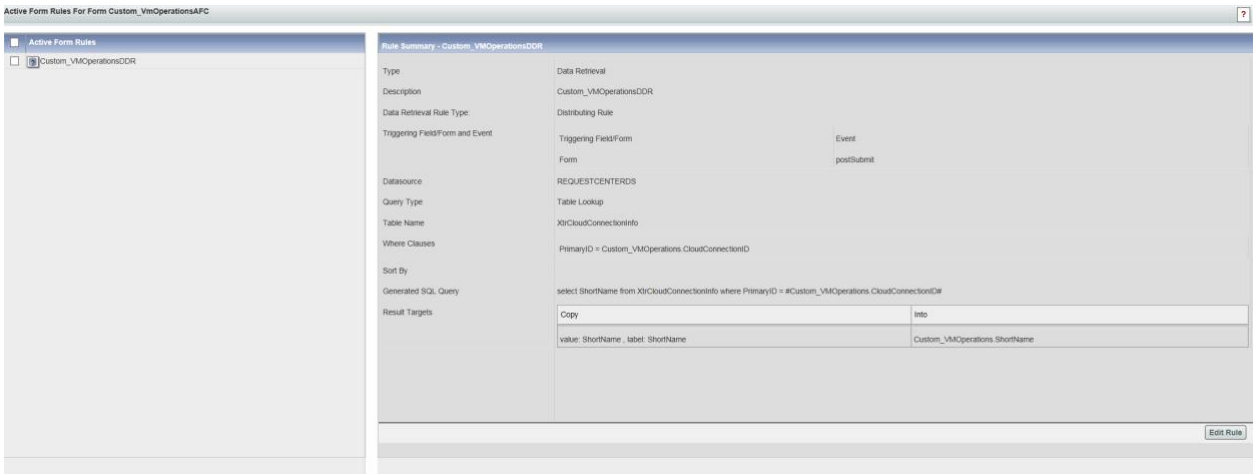


2. Create an active form component and add the above created dictionaries to the form.
    a. Choose **Service Designer > Active Form Component**.
    b. Choose **New > Active Form Component**.
    c. Enter a name and brief description for the new form.
    d. Click on the form group field and select one of the groups to associate with the form.
    e. Click **Save Form**.
    f. Choose Form Content tab and click **Add Dictionaries**.
    g. In the Add Dictionaries dialog box, search for the dictionary created in Step 1 and select it. And click **Add**.
    Note: Make sure you select the VMs that belongs to cloud center and all the mandatory dictionary fields is populated before form submission.

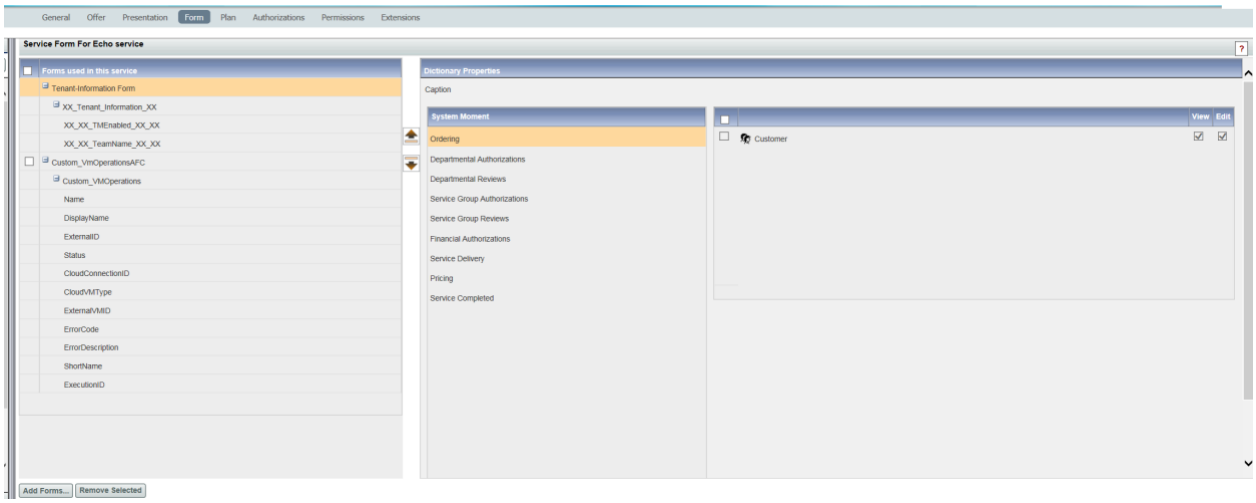
    h. Click **Save Form**.



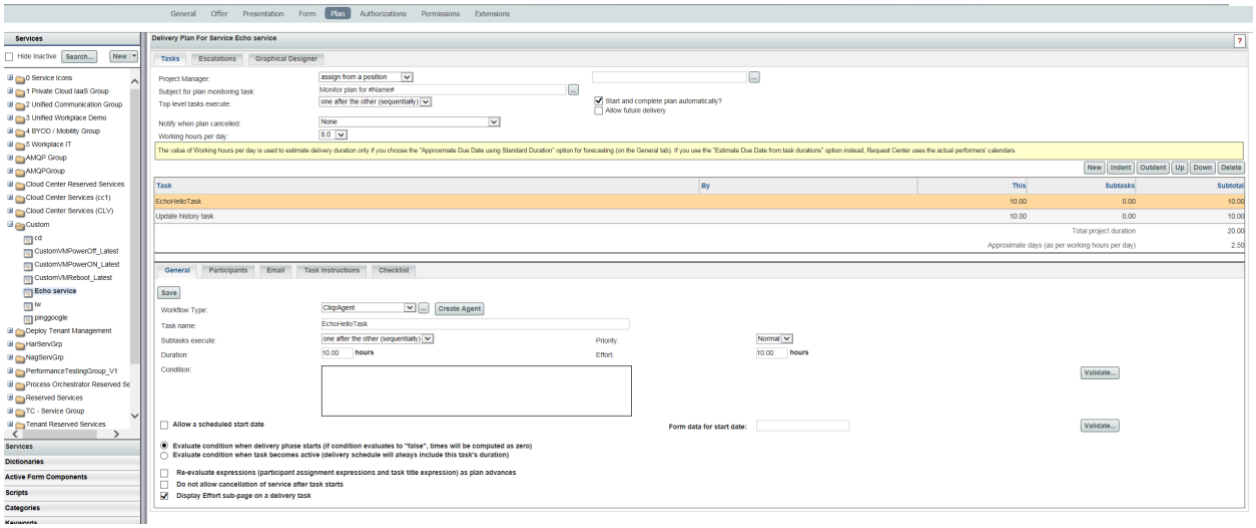
3. Make sure to populate Short Name field. This can be done by creating a data retrieval rule.
    a. Go to **Active Form Rules** tab of the created Active Form Component.
    b. Choose **New Rule** > **New Data Retrieval Rule**
    c. In the first page of the Data Retrieval Rule wizard, enter a unique name and a description for the rule, and specify the Rule Type as *Distributing Rule.*
    d. Specify other details in the wizard as in the below figure and **Save Rule** to populate the shortname**.**



4. Create a Custom Service (for example, CustomOperationService) and add the form to the service.
    a. Choose **Service Designer** > **New** > **New Service**.
    b. Enter the details in the fields provided.
    c. Click **Add This Service**.
    d. After adding the service, you can begin to configure it by entering information on the **General** tab.
    e. Click **Save**.
    f. Click **Form** tab of the service created.
    g. Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.
    h. In the Search field, enter the form name created in step 2.
    i. Check the form and then click **Add**.



5. Create the two tasks specified below in the **Plan** tab of the service.



Perform VM Operation Task and Operation Complete Task are mandatory. The Update History task is optional. For details on configuring each of these tasks, refer to the respective sections:

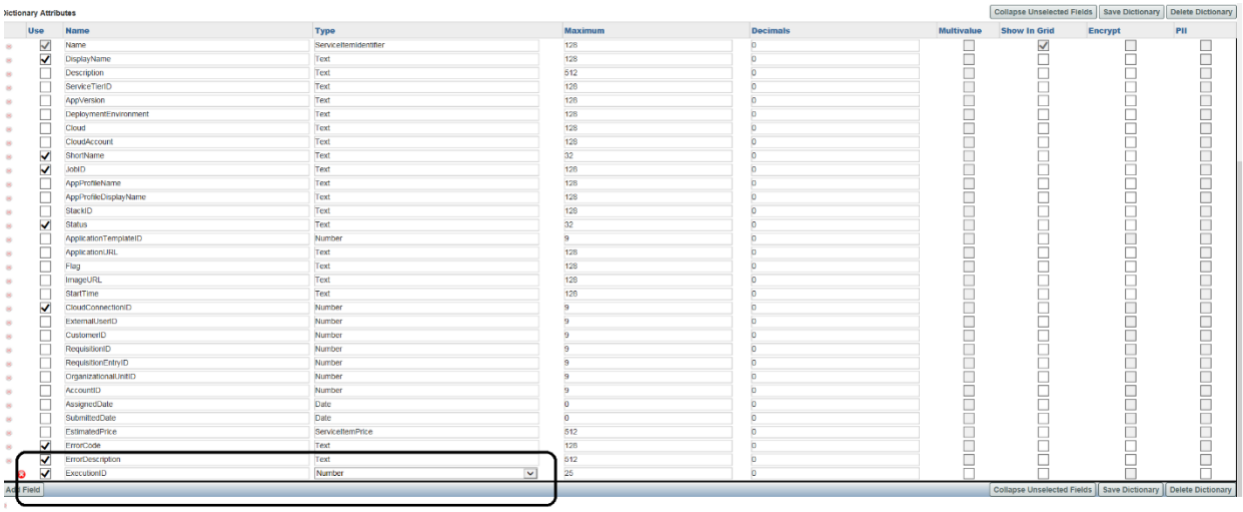
- *Perform VM Operation Task*
- ***Error! Reference source not found.***

6. Save the service.
7. Add the Custom Services in **Service Item Manager > Design Service Items > Application Virtual Machines > Associated Services**.
8. Go to **Service Item Manager > Design Service Items > Application Virtual Machines > Operations** add an operation and associate the service with the operation. This would show up services in the gear icon of the VM.
9. Clear the server cache and restart the server.
   Order the custom services to verify that the custom operations are created successfully.

Support for Custom Actions Operations on Deployments

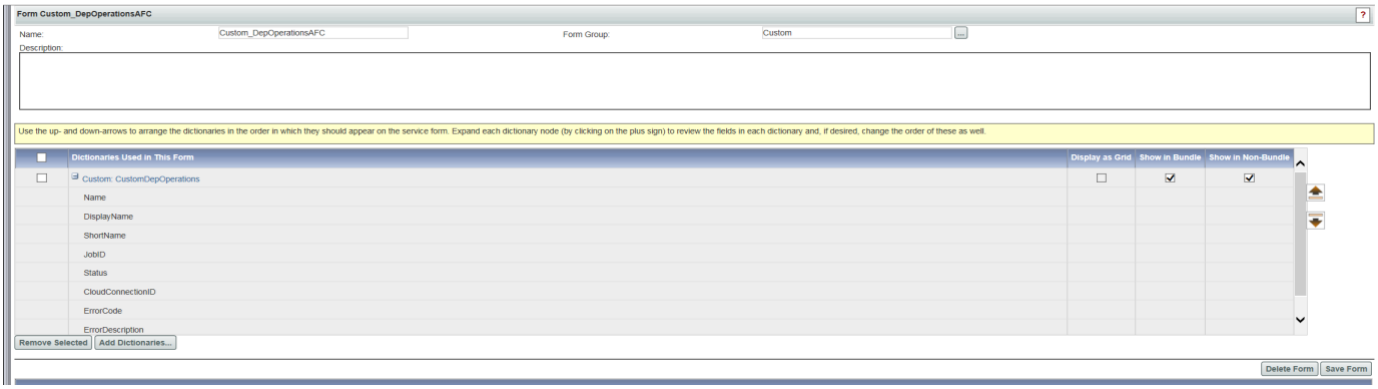
To support custom actions on CloudCenter deployments follow the below procedure:

1. Create a service-item based dictionary by selecting *Application Stack* as Service item type.
   a. Choose **Service Designer > Dictionaries.**
   b. Choose **New > New Dictionary** to display the New Dictionary page.
   c. In the Add New Internal Dictionary section, in the Service Item field, enter *Application Stack* and select the service item.
   d. Enter details such as Dictionary Name and Group Name.
   a. From the Dictionary Attributes section, click Add Field to add a user-defined field **ShortName, JobID, and ExecutionID**.
   e. Select all the fields in the dictionary that is required for the custom action (Any fields that are used in custom FTL). Make sure that the JobID field is selected and ExecutionID field is added to the dictionary.
   f. Click **Save Dictionary**

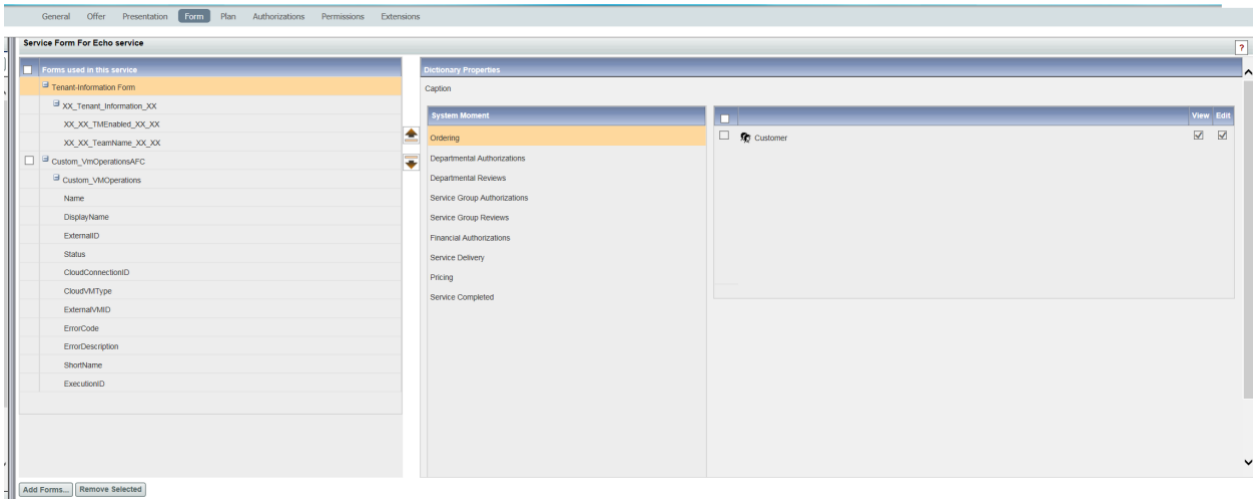


2. Create two active form component and add the above created dictionaries to the form.
    a. Choose **Service Designer > Active Form Component**.
    b. Choose **New > Active Form Component**.
    c. Enter a name and brief description for the new form.
    d. Click on the form group field and select one of the groups to associate with the form.
    e. Click **Save Form**.
    f. Choose Form Content tab and click **Add Dictionaries**.
    g. In the Add Dictionaries dialog box, search for the dictionary created in Step 1 and select it. And click **Add**.
    Note: Make sure you select the VMs that belongs to cloud center and all the mandatory dictionary fields is populated before form submission.

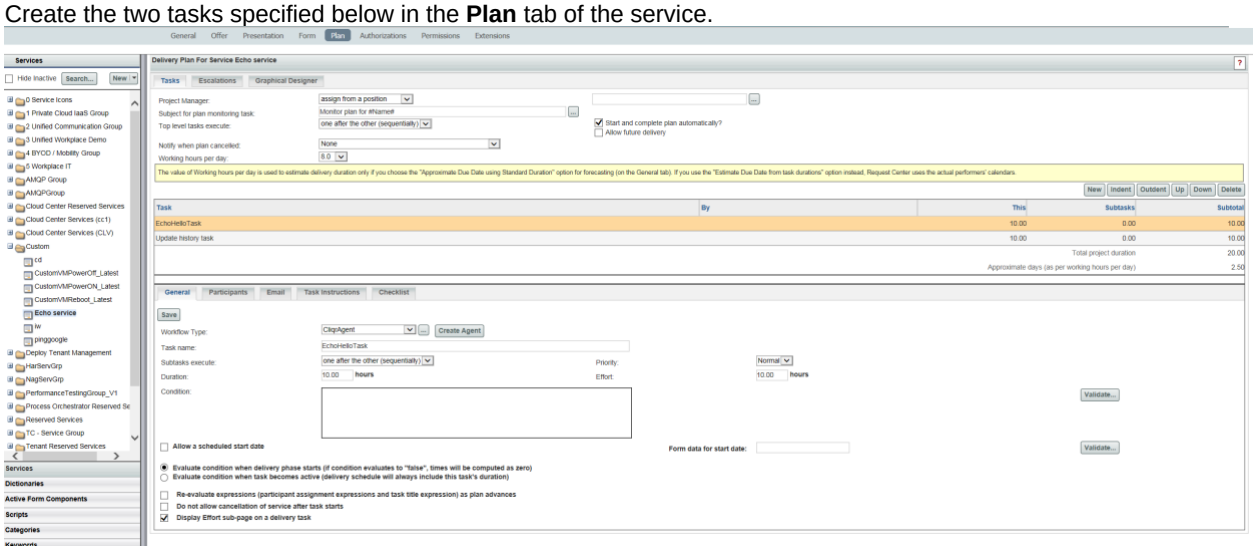
    h. Click **Save Form**.



3. Create a Custom Service (for example, CustomOperationService) and add the form to the service.
    a. Choose **Service Designer** > **New** > **New Service**.
    b. Enter the details in the fields provided.
    c. Click **Add This Service**.
    d. After adding the service, you can begin to configure it by entering information on the **General** tab.
    e. Click **Save**.
    f. Click **Form** tab of the service created.
    g. Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.
    h. In the Search field, enter the form name created in step 2.
    i. Check the form and then click **Add**.



4. Create the two tasks specified below in the **Plan** tab of the service.



Perform VM Operation Task and Operation Complete Task are mandatory. The Update History task is optional. For details on configuring each of these tasks, refer to the respective sections:

- *Perform VM Operation Task*
- ***Error! Reference source not found.***

5. Save the service.
6. Add the Custom Service in **Service Item Manager > Design Service Items > Application Stack > Associated Services**.
7. Go to **Service Item Manager > Design Service Items > Application Stack > Operations** add an operation and associate the service with the operation. This would show up services in the gear icon of the deployment.
8. Clear the server cache and restart the application server.
   Order the custom services to verify that the custom operations are created successfully.

Perform VM Operation Task

To add a Perform VM Operation Task follow the below procedure:

Pre-requisites:

- Create a FTL file containing the base URL for the Custom actions in CloudCenter and the transformation to generate the payload. Below is an example:

```
<#--
*  FTL will support Headers,Post parameters, url (get) Parameters and payload.
```

```
*
* ================================================================
* Header *******   Header_propertyname=Propertyvalue
* Post   *******   Post_propertyname=Propertyvalue
* BaseUrl *******   give you the url and fallowd  by query parameters
* Payload   *******   Payload=payload content
* Header_AuthToken  **** auth token/cookie used for authentication
* ================================================================
-->

Protocol=https
RequestMethod=POST
CommunicationMethod=REST
AuthenticationMethod=Basic
Header_Content-Type=application/json
Header_Accept=application/json
Header_Accept-Encoding=UTF-8
BaseUrl=https://${authority}/v1/actions/34/executions
Payload={
  "resourceType":"VIRTUAL_MACHINE",
  "executionSpecs":[],
  "executionResources":[
    {
      "id":"<#list doc['message']['task-started']['requisition']['requisition-entry']['data-values']['data-value'] as
datavalue><#if (datavalue['name'])=='Custom_VMOperations.ExternalVMID'>${datavalue['value']}</#if></#list>"
    }
  ]
}
AssertResponseToken=$.id::notNullValue
ExtractResponseToken=$.id
ExtractResponseTokenError=$.errors[0].message
```

Note: Highlighted "**Custom_VMOperations" is the dictionary created in Prime Service Catalog should be mapped in the FTL.**

- Add the FTL file at the following location in thePSC server where RequestCenter is deployed:
  ISEE.war/WEB-INF/classes/config/cloud
- Add a mapping in the InterCloud.properties file located at:
  ISEE.war/WEB-INF/classes/config/intercloud.properties

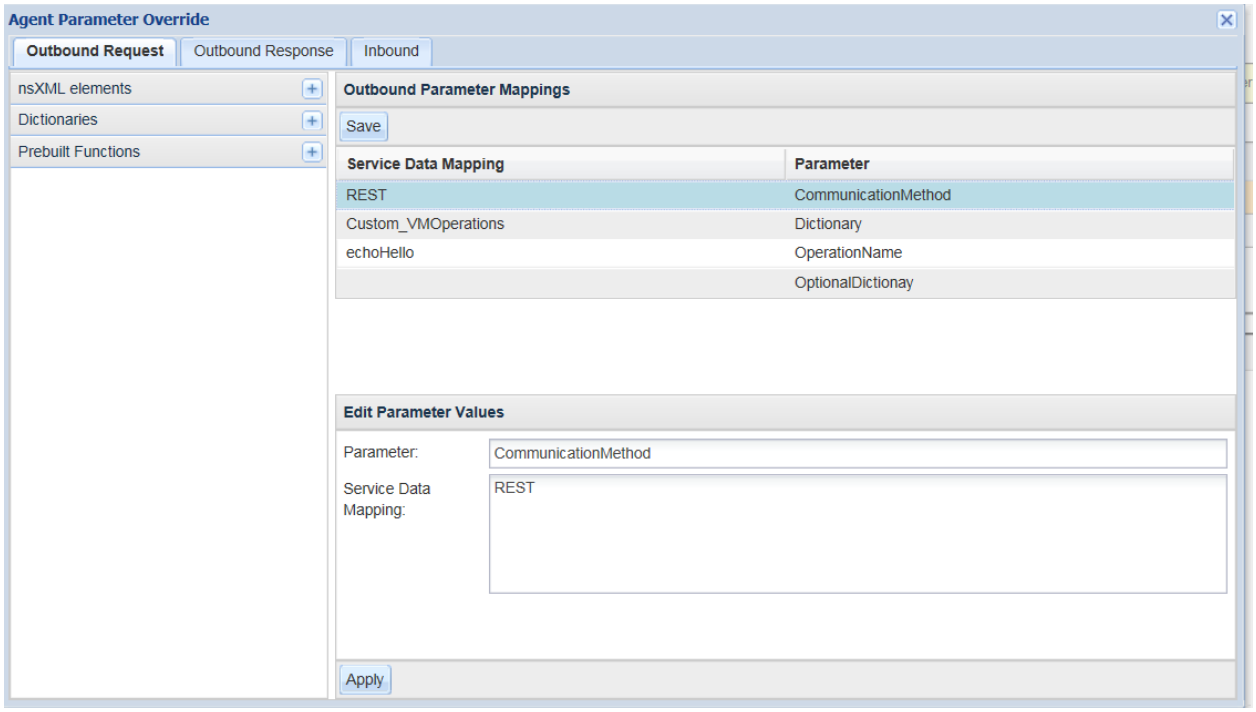
  Entry should be in the below form:
  KEY=ENCODING_SCHEME,PATH_TO_FTL_FILE_FROM_CONFIG_FOLDER

  Key should be CLIQR_{NAME_OF_FTL_FILE without ftl extension}
  E.g. CLIQR_EchoHello=**json**,**config/cloud/Echohello.ftl**

1. Go to **Service Designer > *Service* > Plan** tab.

Where, *Service* is the CustomOperationService created specifically for the custom operation.

2. Add the *Perform VM Operation Task* using a CliqrAgent Task and configure the parameters as within ellipses as shown below:



Override the outbound agent by specifying the below:

Parameter	**Service Data Mapping**
CommunicationMethod	REST
Dictionary	Enter the name of the dictionary that was created in Step 1
OperationName	Enter the custom operation name. It is recommended to use the <CustomOperationName> which should match the action name in the CloudCenter For example, if the action name specified is **echohello** in CloudCenter then the name of the custom operation name should be echohello.

Update History Task

To add an Update History Task for VM follow the below procedure:

Pre-requisites:

- Add an entry in the SyncCustomVMExecutionOperations.properties in the path: ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
- The entry in the UpdateVMHistoryOperations.properties and SyncCustomVMExecutionOperations.properties file must be in the format:
  <Operation_Name>=Operation_Name

  Example:

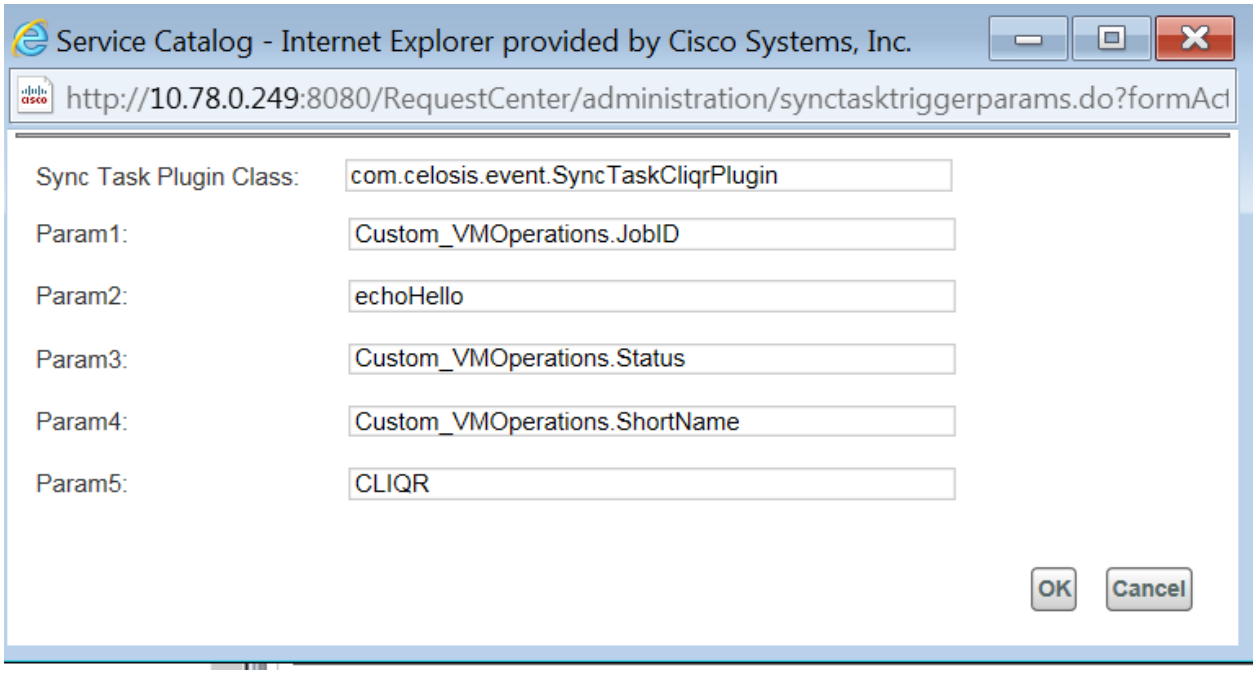
  Echohello=echohello

  Createdirectory=createdirectory

  Note: Same file also exists on the ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud – same entry should be updated for this file as well.

1. Go to **Service Designer > *Service* > Plan** tab.
   Where, *Service* is the CustomOperationService created specifically for the custom operation.

2. Create an Update History with workflow type as SyncPlugin which is configured as below:



To add an Update History Task for deployment follow the below procedure:

Pre-requisites:

- Add an entry in the UpdateDeploymentHistoryOperations.properties in the path: ServiceCatalogServer\deployments\RequestCenter.war\WEB-INF\classes\config\cloud
- The entry in the and UpdateDeploymentHistoryOperations.properties file must be in the format: <Operation_Name>=Operation_Name

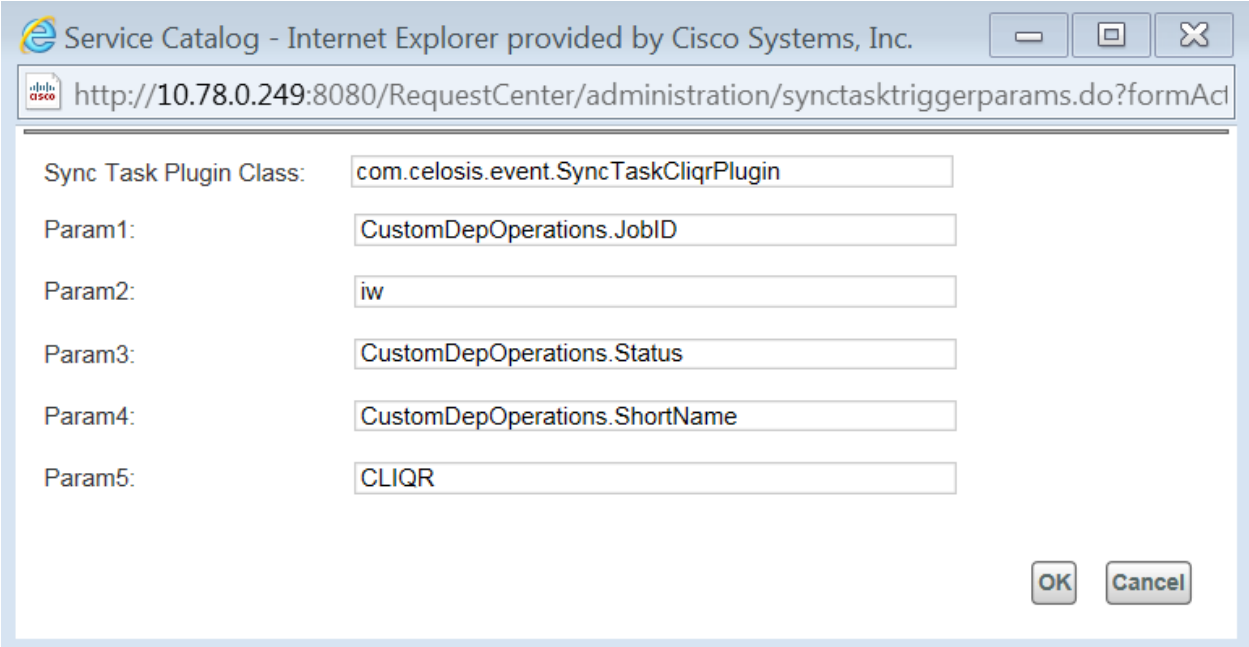
  Example:

  pinggoogle=pinggoogle

  invokewebservice =invokewebservice

  Note: SyncCustomdeploymentExecutionOperations.properties file also exists on the ServiceLinkServer\deployments\ServiceLink.war\WEB-INF\classes\config\cloud – same entry should be updated for this file as well.

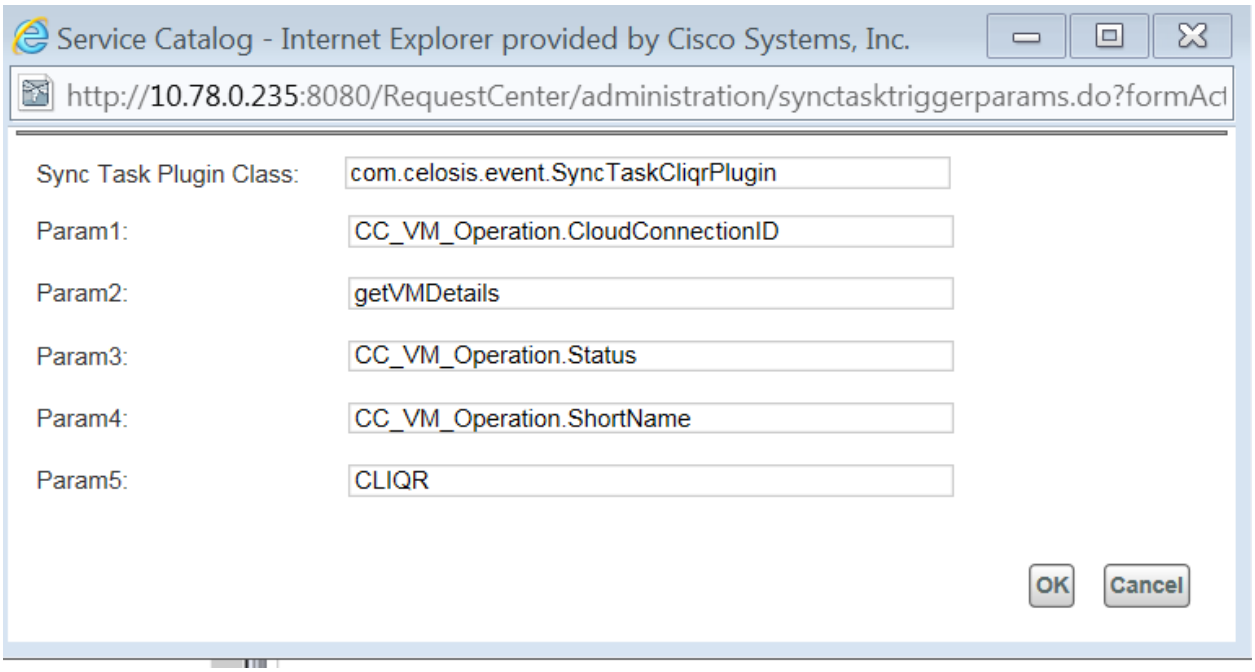
1. Go to **Service Designer > *Service* > Plan** tab.
   Where, *Service* is the CustomOperationService created specifically for the custom operation.
2. Create an Update History with workflow type as SyncPlugin which is configured as below:



Synch App VM Plugin Task

1. Go to **Service Designer > *Service* > Plan** tab.
   Where, *Service* is the one created in Step 4 of *Support for Custom Lifecycle Operations on VMs* section.
2. Create a custom SyncPlugin which is configured as below:

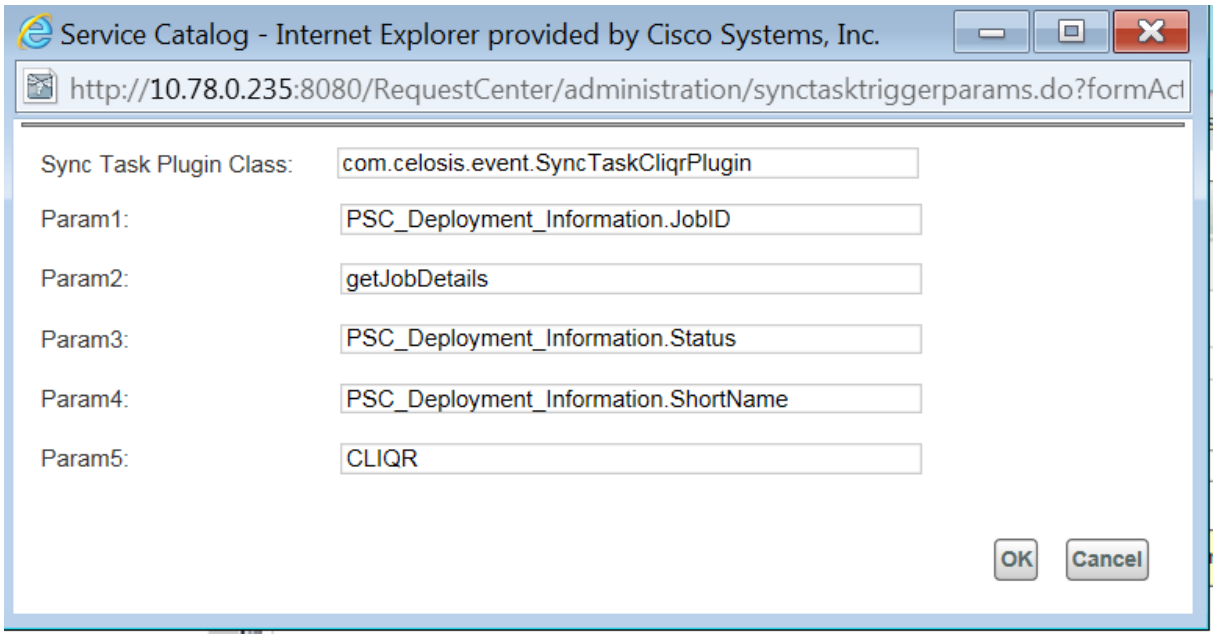
Note: Param1 and Param2 are mandatory.



Synch Deployment Plugin Task

1. Go to **Service Designer > *Service* > Plan** tab.
   Where, *Service* is the one created in Step 4 of *Support for Custom Lifecycle Operations on VMs* section.
2. Create a custom SyncPlugin which is configured as below:

Note: Param1 and Param2 are mandatory.



GDPR Support

To comply with the European Union General Data Protection Regulation (GDPR) laws and handle user data, Prime service catalog has introduced the Compliance module. Personally identifiable information (PII) is any data that can be used to identify a specific individual. This module allows the administrator to manage the GDPR support settings to enhance user data protection in the Prime Service Catalog system.

The compliance module can be enabled only by the Site Administrator using the setting *Enable PII Support* in **Administration > Settings**. By default, this option is disabled.

Compliance Module Capabilities

The Site admin can share the privileges to any user to take over the responsibility of data protection. New capabilities for compliance module are provided such as: access, read only, and write. These capabilities can be used to create custom role and assigned to any user to maintain the compliance module.

Identifying User data as PII

User data in Prime Service Catalog is primarily used in person profile, service items, and requisitions. These data can be identified as PII from following modules:

- **Service Designer > Dictionary**
  Users who have read/write permissions on service are provided options to mark the dictionary field attribute. Select the check-box “PII” for the required Dictionary Attributes.
- **Service Item Manager > Design Service Items**
  Users who have read/write permissions on service items are provided options to mark the service item attribute. Select the check-box “Is PII” for the required Item Attributes.
- **Compliance > Person PII**
  Users who have read/write permissions to the compliance module are provided options to select the person profile data as PII. For more details see section *Global Person Profile PII Settings*.

Global Person Profile PII Settings

The administrator of the compliance can set specific Person Profile Data for all users in the system as PII. For example, the administrator may want to set the mobile phone number, birth date, and personal address of all users as PII. Go to **Compliance > Person PII** select the check-boxes against the data to be set as PII and Save.

Viewing PII Data of a User

The PII Support tab is accessible for administrator or end user to view a user’s PII data all in one place. As an administrator, use the person picker to search for the user and click **Get PII Data**, all the data in the context of the PII linked to that user is displayed below. This page loads the following data in the respective areas:

- Person Profile – PII data of the requested user or logged in user

- Requisitions – Only those requisitions of the user, which have dictionaries with fields marked as PII
- Service Items – Only those Service Items of the user, which have fields marked as PII

The end user has a read-only access to the PII support tab to view the PII data of the logged in user. However, the end user can use the **Export Data** option to generate a .csv file of requisitions and service items with PII data.

Exporting PII Requisitions and Service Items Data

On the requisitions and service items area of PII Support tab, the logged in user or Site administrator is provided with an option to export PII data for any external use. When clicked, a .csv file containing details of each service item or requisition with PII data for that user is listed.

Anonymizing User Data

It is up to the Site administrator to decide whether certain user data must be set as anonymous to avoid any misuse of such data. Using the person picker, search for the user and click **Get PII Data**, all the person profile data, requisitions, and service items of that user are displayed below.

In the respective areas within this page, you can select any number of entries and mark them as anonymized by clicking **Anonymize PII Data**. On each of the areas you also have an option *PII Only* to view only those entries that are set as PII. The *View Archived Data Only* check-box displays only the archived data.

Viewing Anonymized Data

The Anonymized Data tab displays list of users for whom the data has been anonymized till date. The Site Administrator or administrator of compliance module can access this page to view the details of the anonymized PII data of the users.

Setting a Data Protection Policy URL on the Login Screen

The Properties tab of the Compliance module is accessible to the administrator with read/write access. On this page the Administrator can set a URL to an internal webpage that contains Data Protection Policy. This URL appears on the login screen of Prime Service Catalog.

APIs Introduced in 12.1_Patch_v3

CloudCenter APIs

Refresh application profiles to Prime Service Catalog

Method: PUT

REST URL: /RequestCenter/nsapi/cloudCenter/applicationProfiles?connectionId=2

Requestbody:

```
{
   "ApplicationProfileList":
      {
         "applicationProfiles": [
            {
               "name": "Magento",
               "version": "1.1.9.1"
            },
            {
               "name": "Magento1",
               "version": "1.1.9.2"
            }
         ]
      }
}
```

Get the CloudCenter connection ID.

Method: PUT

REST URL: /RequestCenter/nsapi/cloudCenter/connectionIds

Sample response:

```
          {
                    "ArrayList": [
```

```
                    1
                ]
        }
```

Get VM Details
Method: GET

REST URL: RequestCenter/nsapi/cloudCenter/vm/id/{nodeID}

Map Users to deployments
Method: PUT

REST URL: /RequestCenter/nsapi/v2/serviceitem/cloudCenter/{connectionId}/applications?action=mapUsers
Sample response:

200 OK

Sync Managed VMs
Method: PUT

REST URL: /RequestCenter/nsapi/v2/serviceitem/cloudCenter/{connectionId}/virtualMachines?action=sync
Sample response:

200 OK

View the Metadata of each deployment
Method: GET

REST URL: /RequestCenter/nsapi/serviceitem/cloudconnection/{shortName}/application/{applicationName}/metadata

Example

http://localhost:8080/RequestCenter/nsapi/serviceitem/cloudconnection/CLV/application/migrateEvening_from243FInalDeployment/metadata

Sample response:


```
{
        "Map": {
                "Name_Non_Editable_DBInfo": "SQL Server 2016",
                "DB_MandatoryRamSize": "32GB",
                "Install cliqr agent": "agent installled",
                "Name_Mandatory_Filed_RAMInformation": "10GB",
                "Cluster": "RAC system",
                "Windows": "2012 64 bit",
                "bit": "32 Bit",
                "NewFiledaddedonreimport": "New Filed added on reimport",
                "Suppoert": "Support from the oracle",
                "Name": "migrateEvening_from243FInalDeployment_run_1",
                "Imported VM": "Yes depoloted form Imported",
                "Oracle": "DB",
                "Name_JOB_Name": "migrateEvening_from243FInalDeployment_run_1",
```

```
                "CLient": "Intalled",
                "RAM": "8gb"
        }
}
```


Get cost and hours of each deployment
Method: GET

REST URL: /RequestCenter/nsapi/serviceitem/cloudconnection/{shortName}/application/{applicationName}

Example

http://localhost:8080/RequestCenter/nsapi/serviceitem/cloudconnection/CLV/application/migrateEvening_from243FInalDeployment

Sample response:


```
{
        "Map": {
                "vmHours": "18 Hours",
                "cost": "19.0$"
        }
}
```


PII APIs

Service item clear PII data
Only the Site admin can access this API.

Method: POST

Rest URL: /RequestCenter/nsapi/serviceitems/clearPII?customerId=5


```
{
  "serviceItems":[
    {
      "serviceItemTypeId":"164",
      "serviceItemIds":"1,2"
    },
    {
      "serviceItemTypeId":"162",
      "serviceItemIds":"5,6"
    }
  ]
}
```

Note: If customerId is specified then it will take precedence over the payload data. In the payload both serviceitemTypeId and serviceItemIds are mandatory.

Get all PII fields data of Requisitions for specified user IDs

Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitions/PII

Query Params:

gdprState= ALL,ANONYMIZED,NONANONYMIZED

customerId = 1 (mandatory)

startRow = 1 (default: 1)

recordSize = 1 (default: 20)

PIIOnly = true/false (default: false)

sortOrder = ASC/DESC (default: DESC)

sortBy = RequisitionID, CreatedOn (default: RequisitionID)

isArchived = true/false (default - false)

Response:


```
{
    "startRow": 1,
    "recordSize": 1,
    "totalRecords": 1,
    "requisitions": [
        {
            "requisitionId": 33,
            "createdOn": "Wed May 02 06:47:40 IST 2018",
            "customerId": 5,
            "archived": false,
            "anonymizedOn": "Wed May 02 06:49:50 IST 2018",
            "requisitionEntries": [
                {
                    "requisitionEntryId": 42,
                    "serviceId": 141,
                    "serviceName": "CustomerInformation_Delete",
                    "formFields": [
                        {
                            "fieldName": "Customer_Title",
                            "dictionaryName": "CustomerDelteInformation",
                            "dictionaryCaption": "CustomerDeleteInformationCaption",
                            "isPII": true,
                            "fieldValue": "aa 02-05-2018 08:50:33.927"
                        },
```

```
                                        {
                                                "fieldName": "Customer_Birthdate",
                                                "dictionaryName": "CustomerDelteInformation",
                                                "dictionaryCaption": "CustomerDeleteInformationCaption",
                                                "isPII": true,
                                                "fieldValue": "aa 02-05-2018 08:50:33.928"
                                        }

                                ]
                        }
                }
        }
```


Get all PII fields data of Service Items for specified user IDs
Method: GET

REST URL: RequestCenter/nsapi/serviceitems/PII
Query Params: PIIOnly=true/false

customerID = <specific_customer_id>

Response:


```
{
    "AllServiceItems": {
            "startRow": 1,
            "totalCount": 1,
            "recordSize": 1,
            "serviceitemsubscriptions": [
                    {
                            "id": 416,
                            "serviceItemClassificationName": "Hardware",
                            "organizationalUnitID": 22,
                            "assignedDate": "05/23/2018",
                            "requisitionID": 0,
                            "submittedDate": "05/23/2018",
                            "submittedDateRaw": 1527034608000,
                            "anonymizedOn": "05/23/2018",
                            "anonymizedOnRaw": 1527056601000,
                            "customerID": 6,
                            "requisitionEntryID": 0,
                            "serviceItemClassificationID": 23,
                            "organizationalUnitName": "ShashiOU",
```

```
"customerName": "shashi215 user",
"accountID": 0,
"accountName": " ",
"agreementID": 0,
"agreementName": " ",
"serviceitem": {
    "id": 162,
    "name": "UserServiceitem",
    "logicName": "SiUserServiceitem",
    "shareable": true,
    "serviceItemData": [
        {
            "rowId": 4,
            "serviceItemURL": null,
            "myservicesServiceItemURL": null,
            "serviceItemURLOnly": null,
            "protectedAttributes": [
                "Str32",
                "Str512"
            ],
            "items": [
                {
                    "Name": "Test4",
                    "Str32": "ss 2018-05-23",
                    "Str512": "ss 2018-05-23 11:53:21"
                }
            ],
            "subscription": {}
        }
    ]
},
"displayName": "Test4"
}
}
```


Export PII requisitions data of a customer
Method: GET

REST URL: /RequestCenter/nsapi/transaction/requisitions/exportPII
Query Params:

customerId = 1 (mandatory)

PIIOnly = true/false

Response:

Generates an excel file in csv format with all the customer data.

Export PII Service Items data of a customer

Method: GET

REST URL: /RequestCenter/nsapi/serviceitems/exportPII?CustomerID=<specific_customer_id>

Response:

Generates an excel file in csv format with all the customer data.

Anonymize Requisitions for specified users

Method: POST

REST URL: /RequestCenter/nsapi/transaction/requisitions/clearUserData

Query Params:

customerId=1

replaceWithString=gdpr_safe

Sample request:

request data: optional

```
{
 "requisitionIds":"1,2,3,4"
}
```

Note: Customer ID takes preference over requisitionsIds. So if customer ID is passed in query param, requisitionsIds values from request body will be ignored.

Set Person Fields as PII

Method: PUT

REST URL: /RequestCenter/nsapi/directory/people/gdprConfiguration?responseType=json

Sample payload:

```
{
    "persongdprconfigurations": {
        "totalCount": 0,
        "persongdprconfiguration": [
            {
                "fieldName": "FIRSTNAME",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 1,
                "displayName": "First Name"
            },
            {
                "fieldName": "LASTNAME",
                "entityTypeID": 5,
```

```
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 2,
                "displayName": "Last Name"
            }]
}
}
```


Get Person's PII fields:
Method: GET

REST URL: /RequestCenter/nsapi/directory/people/gdprConfiguration?responseType=json

Sample response:


```
{
    "persongdprconfigurations": {
        "totalCount": 0,
        "persongdprconfiguration": [
            {
                "fieldName": "FIRSTNAME",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 1,
                "displayName": "First Name"
            },
            {
                "fieldName": "LASTNAME",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 2,
                "displayName": "Last Name"
            },
            {
                "fieldName": "BIRTHDATE",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
```

```
                                    "protectedField": false,
                                    "configurationID": 3,
                                    "displayName": "Birth Date"
                                }]

}
}
```


Configure Compliance Module

Method: GET

REST URL: /RequestCenter/nsapi/v1/common/moduleconfiguration/compliance

Sample Response:


```
{
    "profileURL":
"../../../myservices/navigate.do?query=personprofile&id=10&moduleId=49&personProfileModuleId=5&force=true&layout=popup&utid
=70BA0BAA5AE8B7683ED76EF94BDB2D80",
    "portalNSApiPageSizeForDirectory": 20,
    "portal_refactor_menu": null,
    "isNextGenUser": null,
    "accessAuthorization": true,
    "accessProfile": true,
    "personInfo": {
        "firstName": "test1",
        "lastName": "test1",
        "timeZoneId": 256,
        "timeZoneDisplayName": "(GMT-08:00) Pacific Time (US and Canada), Tijuana",
        "languageId": 0,
        "placeId": 1,
        "homeOrganizationalUnitId": 24,
        "goals": "",
        "login": "test1",
        "localeId": 1,
        "status": 0,
        "managerId": 0,
        "languageName": "<s ID=\"153\"/>",
        "id": 10,
        "recordStateId": 1,
        "tenantId": 1,
        "dailyLoad": 0,
        "supervisorId": 0,
        "isActive": true,
        "count": 0,
        "queue": false,
        "shouldCreatePerson": true,
        "cloudPassword": "admin",
        "guid": "26F6B280-1EB5-4120-854D-39F8F57A2042",
        "isInactive": false,
        "isLocked": 0,
        "dirNetworkInfo": {
            "loginname": "test1",
            "personID": 0,
            "networkInfoId": 0,
            "isLocked": 0,
            "login": "test1"
        },
```

```
    "passwordChange": false,
    "loginInfoDTO": {
        "loginname": "test1",
        "personID": 0,
        "networkInfoId": 0,
        "isLocked": 0,
        "login": "test1"
    },
    "personalAddressDTO": {
        "addressTypeId": 0,
        "objectId": 0,
        "objectInstanceId": 0,
        "id": 0,
        "tenantId": 0,
        "personId": 0
    },
    "businessAddressDTO": {
        "addressTypeId": 0,
        "objectId": 0,
        "objectInstanceId": 0,
        "id": 0,
        "tenantId": 0,
        "personId": 0
    },
    "homeOUID": 24,
    "personalIAddressDTO": {
        "addressTypeId": 0,
        "objectId": 0,
        "objectInstanceId": 0,
        "id": 0,
        "tenantId": 0,
        "personId": 0
    },
    "businessIAddressDTO": {
        "addressTypeId": 0,
        "objectId": 0,
        "objectInstanceId": 0,
        "id": 0,
        "tenantId": 0,
        "personId": 0
    },
    "birthDateAsString": "",
    "hireDateAsString": "",
    "idEncrypted": "YjkNQUU2GzrZyirVnzLQyw%3D%3D",
    "iloginInfoDTO": {
        "loginname": "test1",
        "personID": 0,
        "networkInfoId": 0,
        "isLocked": 0,
        "login": "test1"
    }
},
"accessMyStuff": true,
"accessIntegration": false,
"showTeams": false,
"hasComplianceAccess": false,
"refactor_menu": null,
"enableHeaderFooter": false,
"moduleId": "28",
"accessMyVDC": false,
"hasComplianceReadCapability": false,
"accessVDCReport": false,
```

```
    "viewRequisitions": true,
    "isComplianceWebsiteOn": false,
    "accessMyServers": false,
    "EUIOOBSearch": false,
    "shortDateFormat": "MM/dd/yyyy",
    "issiteadmin": false,
    "enableCloudCenter": true,
    "debugJ2EE": false,
    "enableCustomCSS": true,
    "sessionToken": "70BA0BAA5AE8B7683ED76EF94BDB2D80",
    "accessMyApplications": false,
    "hasComplianceWriteCapability": false
}
```


Get list of users for whom the data is anonymized:
Method: GET

REST URL: /RequestCenter/nsapi/directory/people/anonymized

Response:


```
{
"userList": {
        "startRow": 1,
        "totalCount": 3,
        "recordSize": 3,
        "user": [
                {
                        "personId": 5,
                        "login": "ramesh2"
                },
                {
                        "personId": 8,
                        "login": "user1"
                },
                {
                        "personId": 9,
                        "login": "user2"
                }

        ]
}
}
```


Anonymize a User
Method: PUT

REST URL: /RequestCenter/nsapi/directory/people/anonymized?responseType=json&PersonId=1
Sample Payload


```
{
    "status-message": {
        "code": "Success",
        "value": "Updated Successfully"
    }
}
```


Get list of anonymized data

Method: GET

REST URL: /RequestCenter/nsapi/directory/people/id/{personID}/anonymizedData?responseType=json&PIIOnly=false

Sample response:


```
{
    "persongdprconfigurations": {
        "totalCount": 30,
        "anonymizedOnStr": "",
        "persongdprconfiguration": [
            {
                "fieldName": "FIRSTNAME",
                "fieldValue": "cctest111",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 31,
                "displayName": "First Name"
            },
            {
                "fieldName": "LASTNAME",
                "fieldValue": "cctest111",
                "entityTypeID": 5,
                "entityTypeName": "person",
                "entityTypeLocalizedName": "Person",
                "protectedField": true,
                "configurationID": 32,
                "displayName": "Last Name"
            },
```

```
{
    "fieldName": "BIRTHDATE",
    "entityTypeID": 5,
    "entityTypeName": "person",
    "entityTypeLocalizedName": "Person",
    "protectedField": true,
    "configurationID": 33,
    "displayName": "Birth Date"
},
{
    "fieldName": "SSN",
    "entityTypeID": 5,
    "entityTypeName": "person",
    "entityTypeLocalizedName": "Person",
    "protectedField": true,
    "configurationID": 34,
    "displayName": "SSN"
},
{
    "fieldName": "STREET1",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 35,
    "displayName": "Street 1"
},
{
    "fieldName": "STREET2",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 36,
    "displayName": "Street 2"
},
{
    "fieldName": "CITY",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
```

```
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 37,
    "displayName": "City"
},
{
    "fieldName": "STATEPROVINCE",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 38,
    "displayName": "State or Province"
},
{
    "fieldName": "ZIP",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 39,
    "displayName": "ZIP"
},
{
    "fieldName": "COUNTRY",
    "entityTypeID": 2,
    "entityTypeName": "personaladdress",
    "entityTypeLocalizedName": "Personal Address",
    "protectedField": true,
    "configurationID": 40,
    "displayName": "Country"
},
{
    "fieldName": "EMAIL",
    "fieldValue": "cctest111@cisco.com",
    "entityTypeID": 6,
    "entityTypeName": "email",
    "entityTypeLocalizedName": "Email",
    "protectedField": true,
```

```
    "configurationID": 41,
    "displayName": "Email"
},
{
    "fieldName": "WORKPHONE",
    "entityTypeID": 7,
    "entityTypeName": "workphone",
    "entityTypeLocalizedName": "Work Phone",
    "protectedField": true,
    "configurationID": 42,
    "displayName": "Work Phone"
},
{
    "fieldName": "HOMEPHONE",
    "entityTypeID": 8,
    "entityTypeName": "homephone",
    "entityTypeLocalizedName": "Home phone",
    "protectedField": true,
    "configurationID": 43,
    "displayName": "Home phone"
},
{
    "fieldName": "FAX",
    "entityTypeID": 9,
    "entityTypeName": "fax",
    "entityTypeLocalizedName": "Fax",
    "protectedField": false,
    "configurationID": 44,
    "displayName": "Fax"
},
{
    "fieldName": "MOBILEPHONE",
    "entityTypeID": 10,
    "entityTypeName": "mobilephone",
    "entityTypeLocalizedName": "Mobile phone",
    "protectedField": true,
    "configurationID": 45,
    "displayName": "Mobile phone"
},
{
```

```
    "fieldName": "PAGER",
    "entityTypeID": 11,
    "entityTypeName": "pager",
    "entityTypeLocalizedName": "Pager",
    "protectedField": false,
    "configurationID": 46,
    "displayName": "Pager"
},
{
    "fieldName": "OTHER",
    "entityTypeID": 12,
    "entityTypeName": "other",
    "entityTypeLocalizedName": "Other",
    "protectedField": false,
    "configurationID": 47,
    "displayName": "Other"
},
{
    "fieldName": "MAINPHONE",
    "entityTypeID": 13,
    "entityTypeName": "mainphone",
    "entityTypeLocalizedName": "Main Phone",
    "protectedField": true,
    "configurationID": 48,
    "displayName": "Main Phone"
},
{
    "fieldName": "PRIMARYPHONE",
    "entityTypeID": 14,
    "entityTypeName": "primaryphone",
    "entityTypeLocalizedName": "Primary Phone",
    "protectedField": true,
    "configurationID": 49,
    "displayName": "Primary Phone"
},
{
    "fieldName": "PRIMARYFAX",
    "entityTypeID": 15,
    "entityTypeName": "primaryfax",
```

```
    "entityTypeLocalizedName": "Primary Fax",
    "protectedField": true,
    "configurationID": 50,
    "displayName": "Primary Fax"
},
{
    "fieldName": "SALESPHONE",
    "entityTypeID": 16,
    "entityTypeName": "salesphone",
    "entityTypeLocalizedName": "Sales Phone",
    "protectedField": false,
    "configurationID": 51,
    "displayName": "Sales Phone"
},
{
    "fieldName": "SUPPORTPHONE",
    "entityTypeID": 17,
    "entityTypeName": "supportphone",
    "entityTypeLocalizedName": "Support Phone",
    "protectedField": false,
    "configurationID": 52,
    "displayName": "Support Phone"
},
{
    "fieldName": "BILLINGPHONE",
    "entityTypeID": 18,
    "entityTypeName": "billingphone",
    "entityTypeLocalizedName": "Billing Phone",
    "protectedField": false,
    "configurationID": 53,
    "displayName": "Billing Phone"
},
{
    "fieldName": "OTHERCONTACT",
    "entityTypeID": 19,
    "entityTypeName": "othercontact",
    "entityTypeLocalizedName": "Other Contact",
    "protectedField": false,
    "configurationID": 54,
    "displayName": "Other Contact"
```

```
},
{
    "fieldName": "STREET1",
    "entityTypeID": 1,
    "entityTypeName": "companyaddress",
    "entityTypeLocalizedName": "Company Address",
    "protectedField": false,
    "configurationID": 55,
    "displayName": "Street 1"
},
{
    "fieldName": "STREET2",
    "entityTypeID": 1,
    "entityTypeName": "companyaddress",
    "entityTypeLocalizedName": "Company Address",
    "protectedField": false,
    "configurationID": 56,
    "displayName": "Street 2"
},
```


Miscellaneous APIs

Get attachments for requisitions.
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition/{requisitionID}/attachment
Sample Response:


```
{
    "ArrayList": [
        {
            "startRow": 0,
            "totalCount": 2,
            "recordSize": 2,
            "requisitionId": 31,
            "document": [
                {
                    "documentID": 4,
                    "objectID": 29,
                    "objectInstID": 31,
                    "fileDisplayName": "Mac Setup Assistant Summary.pdf",
                    "docSize": 44789,
```

```
                    "lastUploadDate": "05/18/2018 8:38 AM"
                },
                {
                    "documentID": 3,
                    "objectID": 29,
                    "objectInstID": 31,
                    "fileDisplayName": "Screen Shot 2018-05-04 at 2.57.58 PM.png",
                    "docSize": 58862,
                    "lastUploadDate": "05/18/2018 8:38 AM"
                }
            ]
        }
    ]
}
```


Delete Attachments

Method: DELETE

REST URL: /RequestCenter/nsapi/transaction/requisitions/id/{requisitionID}/attachments?documentIds=6

Response:

<nsapi-response>Attachments successfully deleted. </nsapi-response>

Download attachment with document ID

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition/attachment/{requisitionID}

Get status of the requisition and pending for approval

Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition/{requisitionIDs}/status

Sample response:


```
{
    "ArrayList": [
        {
            "status": {
                "requisitionId": 13,
                "statusId": 2,
                "statusName": "Closed"
            }
        },
        {
            "status": {
                "requisitionId": 12,
                "statusId": 4,
```

```
                    "statusName": "Rejected"
                }
            },
            {
                "status": {
                    "requisitionId": 10,
                    "statusId": 5,
                    "statusName": "Delivery Cancelled"
                }
            }
        ]
}
```


Pull comments of the requisition (both user comments and system comments).
Method: GET

REST URL: /RequestCenter/nsapi/transaction/v2/requisition/comments
Sample response:


```
{
    "entities": {
        "totalCount": 1,
        "hasMore": false,
        "items": [
            {
                "startRow": 0,
                "totalCount": 0,
                "recordSize": 0,
                "id": 14,
                "usercomments": [
                    {
                        "commentedOn": "05/16/2018 2:20 PM",
                        "commentText": "See my comments",
                        "originatorName": "Dhananjya k",
                        "commentsID": 36
                    }
                ],
                "systemcomments": [
                    {
                        "commentedOn": "05/16/2018 10:37 AM",
                        "commentText": "admin admin has completed task [Monitor plan for Apple Macbook] in requisition number: 14.",
                        "originatorName": "admin admin",
```

```
                                        "commentsID": 27
                                    },
                                    {
                                        "commentedOn": "05/16/2018 10:36 AM",
                                        "commentText": "In Requisition 14, activity Second Level Auth, for Service Apple Macbook,   admin admin  performed [Approve].",
                                        "originatorName": "admin admin",
                                        "commentsID": 25
                                    },
                                    {
                                        "commentedOn": "05/16/2018 10:36 AM",
                                        "commentText": "In Requisition 14, activity First Level Auth, for Service Apple Macbook,   admin admin  performed [Approve].",
                                        "originatorName": "admin admin",
                                        "commentsID": 24
                                    },
                                    {
                                        "commentedOn": "05/16/2018 10:36 AM",
                                        "commentText": "In Requisition 14, activity Review1, for Service Apple Macbook,   admin admin  performed [OK].",
                                        "originatorName": "admin admin",
                                        "commentsID": 23
                                    },
                                    {
                                        "commentedOn": "05/16/2018 10:35 AM",
                                        "commentText": " has updated service form details of requisition number: 14.",
                                        "originatorName": "admin admin",
                                        "commentsID": 21
                                    }
                                ]
                            }
                        ]
                    }
}
```


Service Item bulk operation (CREATE, UPDATE, DELETE)

Note: Service items of a particular type is supported for bulk operations.

Method: PUT

REST URL: /RequestCenter/nsapi/serviceitem/v2/process?isBatch=true&commitType=2&batchSize=50)

Query Params:

- isBatch=true/false, indicates that bulk operation is used.
- commitType=1/2

    - 1 implies commit type is ALL i.e., if any one service item has issue in getting updated none of the service items will be updated.
    - 2 implies commit type is partial i.e., even if one or more service items have issues, other service items will be updated.
- batchSize=<numeric value> implies the size of the batch in the bulk operation e.g., if 1000 service items are sent for bulk update and batchSize is set as 50 then 20 batches will be executed in database.

Note: If isBatch is set to false then it implies the operation will be done in a non-batch manner, and other query params listed above are also not applicable.

Sample payload:

```
<serviceitem>
<name>CustomSIType02</name>
<serviceItemData>
  <serviceItemAttribute name="Name">D_1</serviceItemAttribute>
  <serviceItemAttribute name="Age">12</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max11</serviceItemAttribute>
  <subscription>
    <loginID>dinesh01</loginID>
    <ouname>CustomOU1</ouname>
    <accountName>CustomTenant</accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
<serviceItemData>
  <serviceItemAttribute name="Name">D_2</serviceItemAttribute>
  <serviceItemAttribute name="Age">12</serviceItemAttribute>
  <serviceItemAttribute name="Rupees">245</serviceItemAttribute>
  <serviceItemAttribute name="DLongInte">2453232</serviceItemAttribute>
  <serviceItemAttribute name="DDoubleFloat">2453232.11</serviceItemAttribute>
  <serviceItemAttribute name="DDateTime">2018-01-01 11:30</serviceItemAttribute>
  <serviceItemAttribute name="Dstringmax">I am string max12</serviceItemAttribute>
  <subscription>
    <loginID>dinesh01</loginID>
    <ouname> CustomOU1</ouname>
    <accountName> CustomTenant </accountName>
    <requisitionEntryID>1</requisitionEntryID>
  </subscription>
</serviceItemData>
```

<serviceitem>

Get all the requisitions Ordered For MySelf and Ordered-For-Other

Method: Get

REST URL: /RequestCenter/nsapi/transaction/requisitions/ViewName=Orders with my involvement

Sample Response:


```
{
    "requisitions": {
        "startRow": 1,
        "totalCount": 15,
        "recordSize": 15,
        "requisition": [
            {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 128,
                "customerId": 30,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "startedDate": "05/17/2018 10:56 AM",
                "closedDate": "05/17/2018 10:56 AM",
                "dueDate": "05/18/2018 6:00 PM",
                "expectedCost": 0.0,
                "status": "Closed",
                "requisitionId": 60,
                "lateFlag": false,
                "customerName": "OOBUser2 User2",
                "organizationalUnitName": "OOB_OU2",
                "submitDate": "05/17/2018 10:56 AM",
                "statusId": 2,
                "serviceName": "Create Team",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 53,
                "startedDateRaw": 1526534789093,
                "closedDateRaw": 1526534791630,
                "dueDateRaw": 1526646600000,
                "submitDateRaw": 1526534789333,
```

```
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=60&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">60</a>",
                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=60",
                "milestoneLink": " ",
                "percentageCompleted": 0.0,
                "customerEmail": "oobuser2@cisco.com",
                "isCancelable": true,
                "rating": "0",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "teamId": 0
        },
        {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 174,
                "customerId": 30,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "startedDate": "05/17/2018 10:57 AM",
                "dueDate": "05/17/2018 4:00 PM",
                "expectedCost": 0.0,
                "status": "Ongoing",
                "requisitionId": 62,
                "flagImage": "/RequestCenter/images/flaglate.gif",
                "lateFlag": true,
                "customerName": "OOBUser2 User2",
                "organizationalUnitName": "TeamB",
                "submitDate": "05/17/2018 10:57 AM",
                "statusId": 1,
                "serviceName": "OOB_Service",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 60,
                "startedDateRaw": 1526534825223,
                "dueDateRaw": 1526553000000,
                "submitDateRaw": 1526534831743,
```

```
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=62&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">62</a>",
                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=62",
                "milestoneLink": " ",
                "percentageCompleted": 0.0,
                "customerEmail": "oobuser2@cisco.com",
                "isCancelable": true,
                "rating": "0",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "teamId": 0
        },
        {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 174,
                "customerId": 29,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "startedDate": "05/22/2018 6:30 AM",
                "dueDate": "05/22/2018 4:00 PM",
                "expectedCost": 0.0,
                "status": "Ongoing",
                "requisitionId": 95,
                "flagImage": "/RequestCenter/images/flaglate.gif",
                "lateFlag": true,
                "customerName": "OOBUser1 User1",
                "organizationalUnitName": "TeamA",
                "submitDate": "05/22/2018 6:30 AM",
                "statusId": 1,
                "serviceName": "OOB_Service",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 58,
                "startedDateRaw": 1526950820487,
                "dueDateRaw": 1526985000000,
                "submitDateRaw": 1526950828123,
```

```
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=95&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">95</a>",
                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=95",
                "milestoneLink": " ",
                "percentageCompleted": 0.0,
                "customerEmail": "oobuser1@cisco.com",
                "isCancelable": true,
                "rating": "0",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "teamId": 0
        },
        {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 145,
                "serviceImage":
"/RequestCenter/nsapi/definition/images?tenantId=1&type=def&documentId=371",
                "customerId": 30,
                "expectedDuration": 36.0,
                "actualDuration": 0.0,
                "startedDate": "05/22/2018 8:54 AM",
                "closedDate": "05/22/2018 9:03 AM",
                "dueDate": "05/25/2018 10:00 PM",
                "expectedCost": 0.0,
                "status": "Closed",
                "requisitionId": 99,
                "lateFlag": false,
                "customerName": "OOBUser2 User2",
                "organizationalUnitName": "OOB_OU2",
                "submitDate": "05/22/2018 8:54 AM",
                "statusId": 2,
                "serviceName": "CentOS 6.x (v2.0) (CC1)",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 53,
                "startedDateRaw": 1526959443110,
```

"closedDateRaw": 1526959988160,

"dueDateRaw": 1527265800000,

"submitDateRaw": 1526959443350,

"requisitionURL": "99",

"requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=99",

"milestoneLink": " ",

"percentageCompleted": 0.0,

"customerEmail": "oobuser2@cisco.com",

"isCancelable": true,

"rating": "0",

"reviewsCount": 0,

"selfRating": 0,

"isServiceOrderable": 1,

"isServiceActive": 1,

"teamId": 0

},

{

"tenantId": 1,

"userId": 30,

"ownerId": 30,

"serviceId": 9,

"serviceImage": "/RequestCenter/nsapi/definition/images?tenantId=1&type=def&documentId=11",

"customerId": 30,

"expectedDuration": 10.0,

"actualDuration": 0.0,

"startedDate": "05/25/2018 8:32 AM",

"closedDate": "05/25/2018 8:33 AM",

"dueDate": "05/28/2018 6:00 PM",

"expectedCost": 999.95,

"status": "Closed",

"requisitionId": 115,

"lateFlag": false,

"customerName": "OOBUser2 User2",

"organizationalUnitName": "OOB_OU2",

"submitDate": "05/25/2018 8:32 AM",

"statusId": 2,

"serviceName": "Apple Macbook",

"ownerName": "OOBUser2 User2",

```
                "organizationalUnitId": 53,
                "startedDateRaw": 1527217376520,
                "closedDateRaw": 1527217380063,
                "dueDateRaw": 1527510600000,
                "submitDateRaw": 1527217379393,
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=115&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">115</a>",
                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=115",
                "milestoneLink": " ",
                "percentageCompleted": 0.0,
                "customerEmail": "oobuser2@cisco.com",
                "isCancelable": true,
                "rating": "0",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "teamId": 0
        },
        {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 9,
                "serviceImage":
"/RequestCenter/nsapi/definition/images?tenantId=1&type=def&documentId=11",
                "customerId": 29,
                "expectedDuration": 10.0,
                "actualDuration": 0.0,
                "startedDate": "05/25/2018 8:33 AM",
                "closedDate": "05/25/2018 8:33 AM",
                "dueDate": "05/28/2018 6:00 PM",
                "expectedCost": 999.95,
                "status": "Closed",
                "requisitionId": 116,
                "lateFlag": false,
                "customerName": "OOBUser1 User1",
                "organizationalUnitName": "OOB_OU1",
                "submitDate": "05/25/2018 8:33 AM",
```

```
                "statusId": 2,
                "serviceName": "Apple Macbook",
                "ownerName": "OOBUser2 User2",
                "organizationalUnitId": 52,
                "startedDateRaw": 1527217395690,
                "closedDateRaw": 1527217407687,
                "dueDateRaw": 1527510600000,
                "submitDateRaw": 1527217407360,
                "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=116&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">116</a>",
                "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=116",
                "milestoneLink": " ",
                "percentageCompleted": 0.0,
                "customerEmail": "oobuser1@cisco.com",
                "isCancelable": true,
                "rating": "0",
                "reviewsCount": 0,
                "selfRating": 0,
                "isServiceOrderable": 1,
                "isServiceActive": 1,
                "teamId": 0
        },
        {
                "tenantId": 1,
                "userId": 30,
                "ownerId": 30,
                "serviceId": 175,
                "customerId": 30,
                "expectedDuration": 0.0,
                "actualDuration": 0.0,
                "startedDate": "05/25/2018 8:34 AM",
                "dueDate": "05/25/2018 4:00 PM",
                "expectedCost": 0.0,
                "status": "Ongoing",
                "requisitionId": 117,
                "flagImage": "/RequestCenter/images/flaglate.gif",
                "lateFlag": true,
                "customerName": "OOBUser2 User2",
                "organizationalUnitName": "TeamB",
```

```
                                        "submitDate": "05/25/2018 8:34 AM",
                                        "statusId": 1,
                                        "serviceName": "Copy of OOB_Service",
                                        "ownerName": "OOBUser2 User2",
                                        "organizationalUnitId": 60,
                                        "startedDateRaw": 1527217467360,
                                        "dueDateRaw": 1527244200000,
                                        "submitDateRaw": 1527217469977,
                                        "requisitionURL": "<a
href='/RequestCenter/myservices/navigate.do?reqid=117&layout=popup_p&query=requisitionstatus' onclick=\"return
GB_showFullScreen('Requisition', this.href)\">117</a>",
                                        "requisitionURLOnly": "/RequestCenter/myservices/navigate.do?reqid=117",
                                        "milestoneLink": " ",
                                        "percentageCompleted": 0.0,
                                        "customerEmail": "oobuser2@cisco.com",
                                        "isCancelable": true,
                                        "rating": "0",
                                        "reviewsCount": 0,
                                        "selfRating": 0,
                                        "isServiceOrderable": 1,
                                        "isServiceActive": 1,
                                        "teamId": 0
                                }
}
```


Get get service definitions

Method: GET

REST URL: /RequestCenter/nsapi/definition/servicedefs

Query Param: serviceIds

Response:


```
{
        "services": {
                "startRow": 1,
                "totalCount": 1,
                "recordSize": 1,
                "service": [
                        {
                                "serviceId": 149,
                                "serviceName": "Cassandra Cluster App (v2.1.1) (CLQ)",
                                "description": "Cassandra Cluster Admin is a GUI tool to help people administrate their
Apache Cassandra cluster.",
```

```
"topDescription": "",
"middleDescription": "",
"bottomDescription": "",
"revisionNumber": 293,
"status": "Active",
"statusId": 1,
"expectedDuration": 0.0,
"expectedDurationUnits": "hours",
"canStartLater": false,
"isBundle": false,
"dateQualityId": 4,
"serviceLevelDescription": "",
"isOrderable": true,
"showOrderSummary": false,
"hideInServiceCatalog": false,
"hideInMyOrders": false,
"maxQuantity": 0,
"isReportable": false,
"serviceImage":
"/RequestCenter/nsapi/definition/images?tenantId=1&type=def&documentId=108",
"serviceURL": "<a
href='/RequestCenter/myservices/navigate.do?query=serviceid&sid=149&layout=popup_p' onclick=\"return
GB_showFullScreen('Cassandra Cluster App (v2.1.1) (CLQ)', this.href)\">Cassandra Cluster App (v2.1.1) (CLQ)</a>",
"serviceOrderURL": "<a
href='/RequestCenter/myservices/navigate.do?query=orderform&sid=149&layout=popup_p' onclick=\"return
GB_showFullScreen('Cassandra Cluster App (v2.1.1) (CLQ)', this.href)\">Order</a>",
"serviceURLOnly": "/RequestCenter/myservices/navigate.do?query=serviceid&sid=149",
"serviceOrderURLOnly":
"/RequestCenter/myservices/navigate.do?query=orderform&sid=149",
"orderingMode": 2,
"computePrice": false,
"serviceExtensions": {
        "serviceExtension": [
                {
                        "name": "ServiceID",
                        "logicName": "ServiceID",
                        "value": "149",
                        "type": ""
                },
                {
                        "name": "searchFacets",
                        "logicName": "searchFacets",
```

```
                                                    "value": "{\"Ratings\":[\"None\"]}",
                                                    "type": "Search Facets"
                                                }
                                            ]
                                        },
                                        "serviceAccessories": [],
                                        "servicePrerequisites": [],
                                        "portalText1": "",
                                        "portalText2": "",
                                        "portalText3": "",
                                        "hasPortal1": false,
                                        "hasPortal2": false,
                                        "hasPortal3": false,
                                        "isTemplate": false,
                                        "isBaseService": true,
                                        "templateType": "Application",
                                        "hoursPerDay": 0.0,
                                        "templateRelevant": false,
                                        "tenantRelevant": true
                                    }
                                ]
                            }
}
```


Introduced in 12.1_Patch_v2

Support for IBM Cognos 10.2.2 installation instructions please refer 12.1 readme file.