Cisco Prime Service Catalog

Feature Review for v10.0 – Configuring CSS and Calling REST API

November 25, 2013 *(Updated April 30, 2014)*

Cisco Prime Service Catalog 10.0 Feature Review

- Configuring CSS to Implement Customer Branding
  - Setup a custom style for an OU or tenant
  - Configure Service Catalog CSS for a tenant
  - What does a customer-branded Service Catalog page look like?
  - Configuring the Service Portal Module Header
- Using Service Link to Call REST APIs
- Preview of 10.1 Enhancements

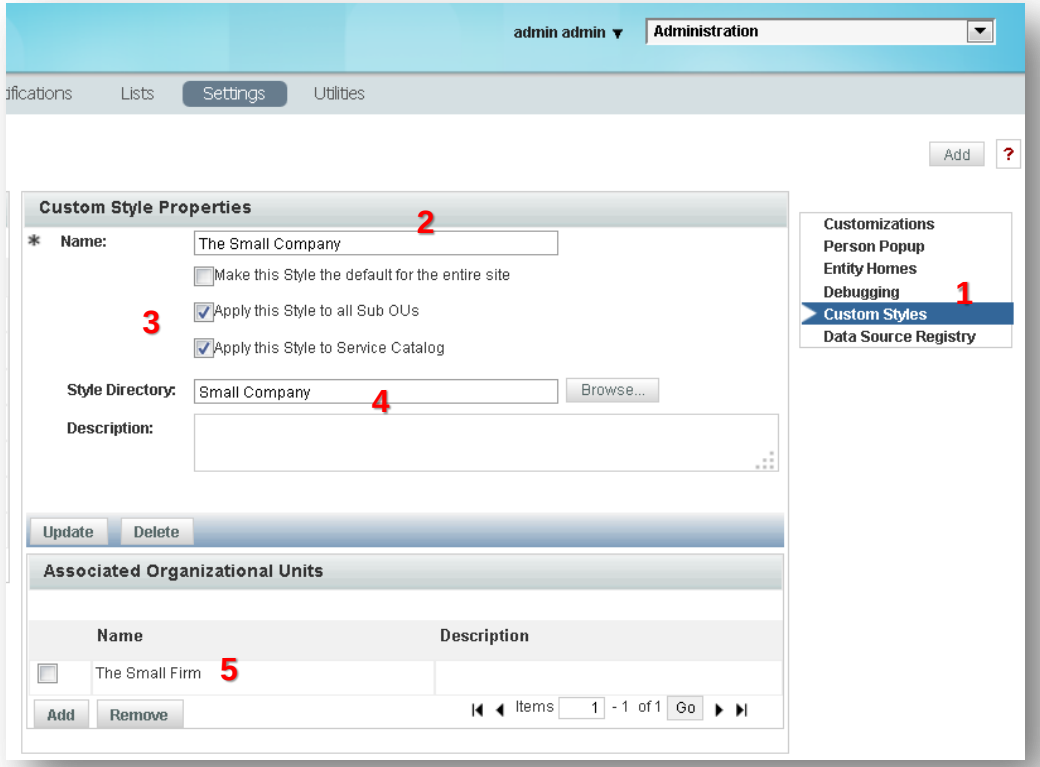
Configuring CSS to Implement Customer Branding

Setup a custom style for an OU or tenant

- The Service Catalog module is powered by HTML5, which means the techniques for configuring and customizing an HTML5 web application applies.
- This short tutorial describes how to setup an organizational unit that has custom style. In Prime Service Catalog 10.0 or earlier, use IE8 for this operation.
- To start with, locate the *custom* directory under RequestCenter.war directory. On Windows, it may be found at

  C:\CiscoPrimeServiceCatalog\jboss-as-7.1.1.Final\RequestCenterServer\deployments\RequestCenter.war\custom
- The examples CSS files are under the “ServiceCatalogExamples” directory.
- To start your CSS work, copy ServiceCatalogExample to a new directory

  e.g. C:\CiscoPrimeServiceCatalog\jboss-as-7.1.1.Final\RequestCenterServer\deployments\RequestCenter.war\custom\Small Company
- Create a user and an organizational unit (OU) called *The Small Firm*.
- Now go to Administration module, and set up a custom style for Small Company.



Configure Service Catalog CSS for a tenant #1

- There are a few files under C:\CiscoPrimeServiceCatalog\jboss-as-7.1.1.Final\RequestCenterServer\deployments\RequestCenter.war\custom*Small Company**application* that you can modify. The three main files to start with

- Copy example_before.html to before.html, and do the same for example_head.html as well as example_after.html

  before.html – The content of this file is prepended to the <body> of the HTML document. You can use this file to override product strings before they are displayed. For example, the product name in the header uses string ID 15162, so you can make the product show "Small Company" by inserting this block into before.html:

  ```
  <script>
  serviceCatalogMessage.putString("15162", 'Small Company');
  </script>
  ```

  Use **Firebug** or other web developer tools to examine the HTML source code of the Service Catalog module. You will see other strings that you can override in the format of "serviceCatalogMessage…"

  head.html – The content of this file is injected into the end of the <head> section of the HTML document. You can override the colors and styles specified by the various HTML classes and elements on the web page. For example, I use the following color settings to create an ugly version Small Company "homeslider".

```
<style>
.homeslider .blocks.block1 {
    background-color: #9E6F9A; //#2B6F9A;
    background-image: linear-gradient(to bottom, #985DC2, #B6115E); // #258DC2, #04415E);
    background-repeat: repeat-x;
}
.homeslider .blocks.block2 {
    background-color: #9E6F9A; //#2B6F9A;
    background-image: linear-gradient(to bottom, #D486C0, #B43557); // #6486C0, #213557);
    background-repeat: repeat-x;
}
.homeslider .blocks.block3 {
    background-color: #9E6F9A; //#2B6F9A;
    background-image: linear-gradient(to bottom, #EE86C0, #BE3157); // #7B86C0, #2B3157);
    background-repeat: repeat-x;
}
</style>
```

Configuring Service Catalog CSS for a tenant #2

- The third file that you can modify in the custom\style directory is

  after.html – The content in this file is appended at the end of the html <body>.

- These three files are shipped as example_before.html, example_head.html and example_after.html. Copy them and remove the “example_” prefix.

- Other files in the directory are available if you want to do more extensive customization of the UI but it is beyond the scope of this high level tutorial.

- You can change the default Cisco Logo to some other company logo in the head.html file, as it is a style override. The following code overrides the logo and page header background to a fictional company, ACME, Inc.

- Notice that the logo image is stored under custom/company name/images directory.

Example head.html CSS for ACME, Inc.

```
<style>
.navbar-inner {
                background-color: transparent;
                background-image: none;
                filter: progid:DXImageTransform.Microsoft.gradient(enabled=false); // disable the gray header bar in IE as ACME’s header background is white
}
.navbar .nav {
                color: #333;
                text-shadow: none;
}
.navbar .nav > li > a {
                color: #333;
                text-shadow: none;
}
.psc-goto-cart {
   color: #333 !important;
}
.psc-top-icon {
                border:2px solid #333;
                background-color: #333;
}

 /* Change company logo */
.psc-header .navbar .brand {
                background-image: url("/RequestCenter/custom/ACME/images/acme.png");
                background-repeat: no-repeat;
                background-position: 0 .514em;
                background-size:75px auto;
                padding-left: 3em!important;
                padding-top: 0.25em!important;
                padding-right: 5em!important;

                font-size: 1.571em;
                line-height: 2.28em;
                color: #333;   /* ACME’s website H1 color */
                text-shadow: none;
}  /* Change company logo */

</style>
```

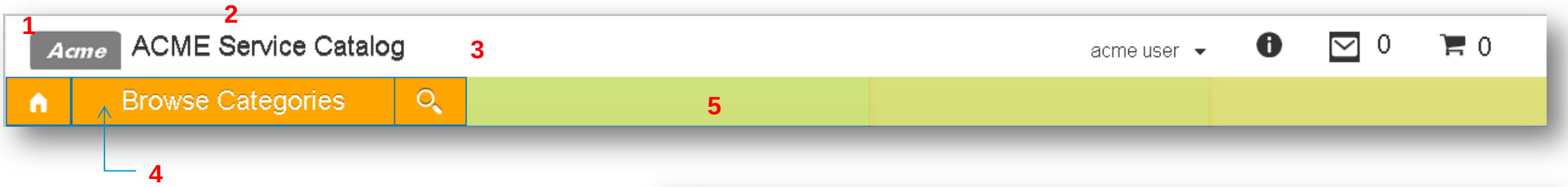
Configuring Service Catalog CSS for a tenant #3

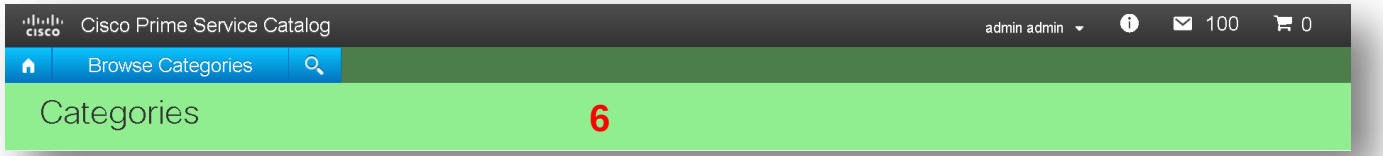
Example head.html CSS for ACME, Inc. (continued)

```
<style>

// The following block changes the colors "Home", "Browse Categories" and magnifying glass
// (search) box to orange(!), including the search box hover state

.categories > .navbar li.menu.menu > a, .categories > .navbar li.menuHome.menu > a, .categories >
.navbar li.search.menu > a {
    background: none repeat scroll 0 0 orange;
}
.categories > .navbar li.menu > a, .categories > .navbar li.menuHome > a, .categories > .navbar
li.search > a {
    background: none repeat scroll 0 0 orange;
}
.categories > .navbar li.menu, .categories > .navbar li.menuHome, .categories > .navbar li.search {
            background: orange;
}
.categories > .navbar li.menu:hover, .categories > .navbar li.menuHome:hover, .categories > .navbar
li.search:hover {
    background-color: orange;
    background-image: linear-gradient(to bottom, orange, orange);
    background-repeat: repeat-x;
}

// This changes the navbar background color to the right of the Home, Browse Categories and
magnifying glass box.
.categories > .navbar {
    background-color: rgba(255, 255, 102, 0.75);  // this sets the color and transparency of the navbar
}

.page-heading {  // This overrides the default :  background-image:
url("../../ngc/img/page_heading_bg.png"); in the “Categories” view, so the color behind the
“Categories” text is light green rather than the blue from the png.
                    background-color: lightgreen;
                    background-image: none;
}
</style>
```

What does the ACME-branded Service Catalog page look like?

- What were customized and the CSS tag used:
  1. Logo - *.psc-header .navbar .brand*
  2. Product name – *string ID 15162*
  3. Background color of the banner holding logo and product name – *navbar**
  4. Background color(s) of the Home, “Browse Categories” and “Magnifying Glass” controls – *.categories > .navbar**
  5. Background color of the navbar to the right of #4 - *.categories > .navbar**
  6. Background color of the “Categories” bar in the service category view – *page-heading*





Configuring the Service Portal Module Header #1

- The header for the Service Portal module can be configured by using the portal-custom-header.css file under the custom/style directory/.
- This file will be loaded at the end of the <head> section for all Portal Module pages.
- The following example works well if

  Logo is rectangular and close to H:92x V:33px

  Portal name is “Service Catalog”, which is the out-of-box *HeaderAppSubTitle*.
- Change the portal module header with the following CSS settings:

```
.reboot2 .xwtBackgroundSimplified {
    background: transparent !important;
}
/*style to modify logo image*/
.reboot2 .applicationHeader17 .applicationLogoImage {
        background: url("images/acme.png") no-repeat transparent !important;
        /*background-repeat: no-repeat;
        background-position: 0 .514em;
        background-size:75px auto;
        padding-left: 3em!important;
        padding-right: 5em!important;*/
        height: 33px !important;
        width:92px !important;
    /* Note: un-comment display property to hide the Product Logo if needed */
    /*display:none !important;*/
}
/* This style is used to display or hide the Product Title in Portal modules
   See Also: .applicationHeaderAppSubTitle, .applicationLogoImage, .applicationHeaderLogoText style - which will be the new style used to display branding logo
*/
.reboot2 .applicationHeader17 .applicationHeaderAppName {
        visibility: hidden; /* set to hidden because the AppName is set to “ACME”, which duplicates the text in the logo */
}
```

Configuring the Service Portal Module Header #2

- Now if your logo is significantly larger than 90x30 pixels or you need a different app name than “Service Catalog”, you need to do more.
- The following example handles these branding requirements

  Logo is larger, 135x70 pixels

  Portal name is “Cloud Manager”, which different from the out-of-box “Service Catalog”. (Note required changes on “Configuring the Service Portal Module Header #3”)

- Change the portal module header with the following CSS settings:

```
.reboot2 .xwtBackgroundSimplified {
    background: transparent !important;
}
.reboot2 .applicationHeader17 {
   padding-left: 5px !important;
}
/*style to modify logo image*/
.reboot2 .applicationHeader17 .applicationLogoImage {
        background: url("/RequestCenter/custom/style_dir/images/customer_logo.png") no-repeat scroll -2px -8px transparent !important;
        background-size: auto auto !important;
        height: 73px !important;
        line-height: 70px !important;
        top: 1px !important;
        width: 130px !important;
}
// This style will reduce the left right and top padding for logo container
.reboot2 .applicationHeader17 .applicationHeaderLogo {
        margin-left: 2px !important;
        margin-right: 2px !important;
        padding-top: 5px !important;
}
.reboot2 .applicationHeader17 .applicationHeaderAppSubTitle {
        color: #202020;
        cursor: default;
        font-size: 17px !important;
        font-weight: bold !important;
        font-family: CiscoSans,Arial,"Helvetica Neue",Helvetica,sans-serif !important;
        text-shadow: 0 0.071em 0 #585858 !important;
  }
```

Configuring the Service Portal Module Header #3

- If you are changing the application name from “Service Catalog”, you also need to customize the PortalFullpagePrimeUi.js file. In this example, it is changed to “Cloud Manager”
- On Windows: This file may be located in C:\CiscoPrimeServiceCatalog\jboss-as-7.1.1.Final\RequestCenterServer\deployments\RequestCenter.war\ns360\js\PortalFullpagePrimeUi.js
- Search for the function name “getNavItems”

```
if(defaultSelectedId != ""){              // existing code
                    // existing code
                   navItems.items.defaultSelected = defaultSelectedId;
}                                         // existing code

navItems.items.toolbar.push(getToolbar()); // existing code

// Overwrite the app header text with customer’s application name “Cloud Manager”
//  If text need to add for AppName div then from custom css file its "visibility“
// attribute should be "block !important“
var appNameDiv =  dojo.query('.applicationHeaderAppName')[0];
appNameDiv.style.display = "block !important";
appNameDiv.innerHTML = "Cloud Manager";
var appSubTitleDiv = dojo.query('.applicationHeaderAppSubTitle')[0];
appSubTitleDiv.style.display = "block !important";
appSubTitleDiv.innerHTML = "Cloud Manager";

return navItems;  // existing code
```

Using Service Link to Call REST APIs

Standard Integration Architecture with Prime Service Catalog

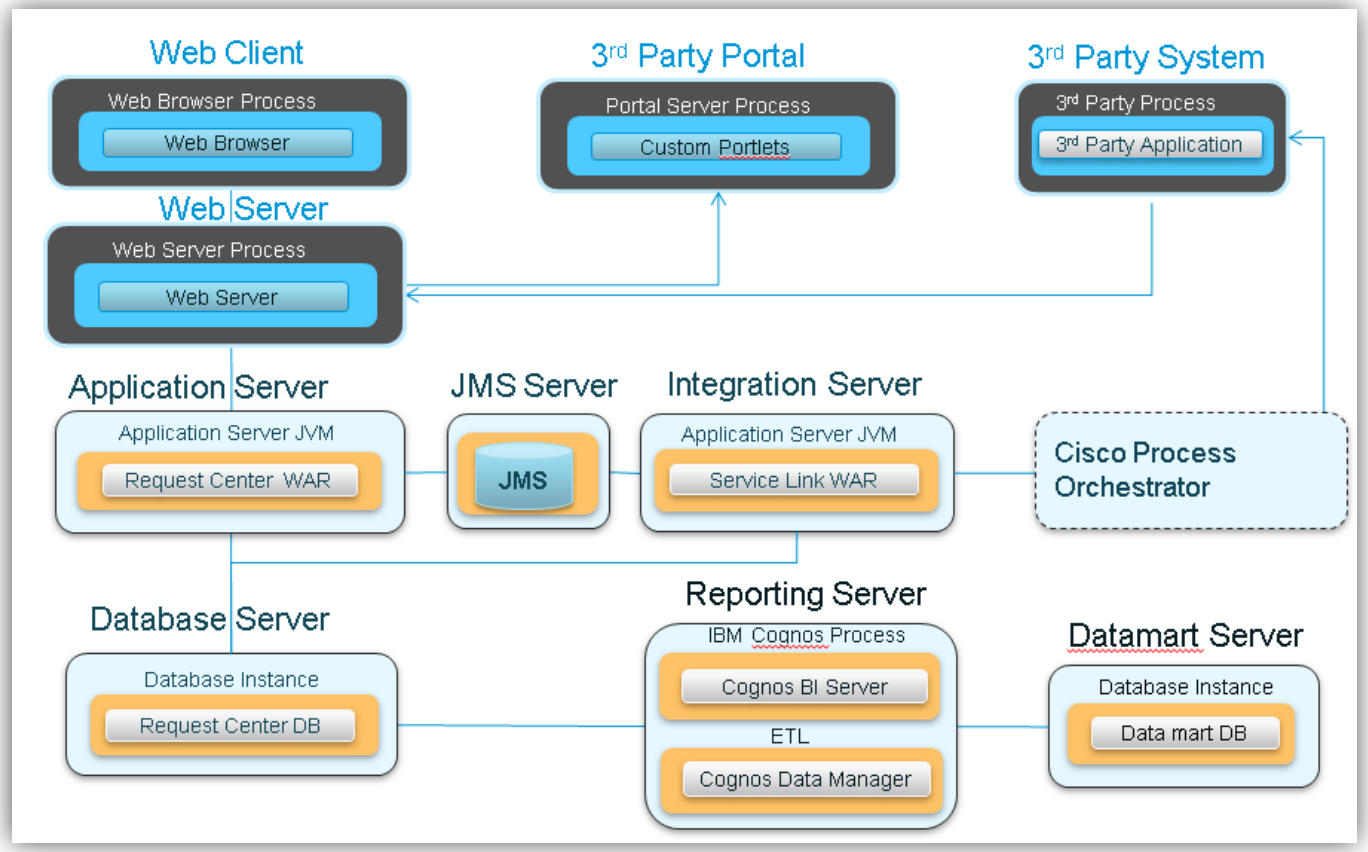
- Prime Service Catalog offers customers the right to use some Cisco Process Orchestrator functionality

  PO Core Functions Adapter, PO Windows Adapter (Single Instance), PO Windows Automation Pack (Single Instances), PO AD Adapter (Single Instances), PO AD Automation Pack (Single Instances), PO Email Adapter (Single Instance), PO Core Automation Pack (Single Instance), PO Common Activities Automation Pack (Single Instance), PO SNMP Adapter (Single Instance), PO Terminal Adapter (Single Instance), Oracle DB Adapter (2 Instances), Microsoft SQL Database Adapter (2 Instances), PO DB2 DB Adapter (2 Instances) & PO Generic (OLEDB) DB Adapter (2 Instances), Cisco Service Portal Adapter (Single Instance), PO Web Service Adapter (5 instances), PO VMware vSphere Adapter (5 Instances), PO VMware vCloud Director Adapter (5 Instances).

- Generally, we recommend using Cisco Process Orchestrator as the automation engine

  Service Link Remedy Adapter has been removed

  Service Link VMware Adapter is no longer supported



What is new and why?

- In Prime Service Catalog 10.0, we enhance Service Link's ability to call RESTful APIs
  - Through the HTTP/WS adapter, it now supports the HTTP PUT, GET, POST, DELETE methods
  - The adapter also now supports parameterized URL and HTTP header
  - This is in addition to supporting sending parameters in the HTTP message body
  - The same adapter is used to support SOAP calls
  - HTTPS is supported (In the Prime Service Catalog Configuration Guide under "Configuring SSL for Service Link Outbound Documents")
- Why extend this functionality in Service Link?
  - Customer requirement. Despite offering CPO for minimal additional cost, some customers insist on using the orchestration technology they have in-house, such as HPOO, Microsoft System Center 2012
  - Streamlined architecture for certain use cases. Prime Service Catalog for UCS-Director will use Service Link to call REST APIs on UCS-Director.
- The following examples illustrate calling REST nsAPIs but it can integrate against public REST APIs. More examples to follow on https://supportforums.cisco.com/community/netpro/data-center/intelligent_automation/service-catalog.

Configuring Service Link to call REST APIs #1

- Configure Service Link HTTP/WS adapter to make RESTful calls
  - Go to Manage Integrations
  - Create Agent
  - Create Transformation
- Create Agent
  1. Service Link → Manage Integrations → Agents
  2. Under General, select HTTP/WS adapter
  3. Under Outbound Properties, populate HttpOutboundAdapter.RoutingURL with the endpoint URL.
  4. Populate ContentType, e.g. application/json.
  5. If expecting data back, set ProcessResponse to true
  6. If there is any header parameters, set them in RequestHeaders.
     - Param1=#rateID#&Param2=#rateName#
     - rateID and rateName are agent parameters

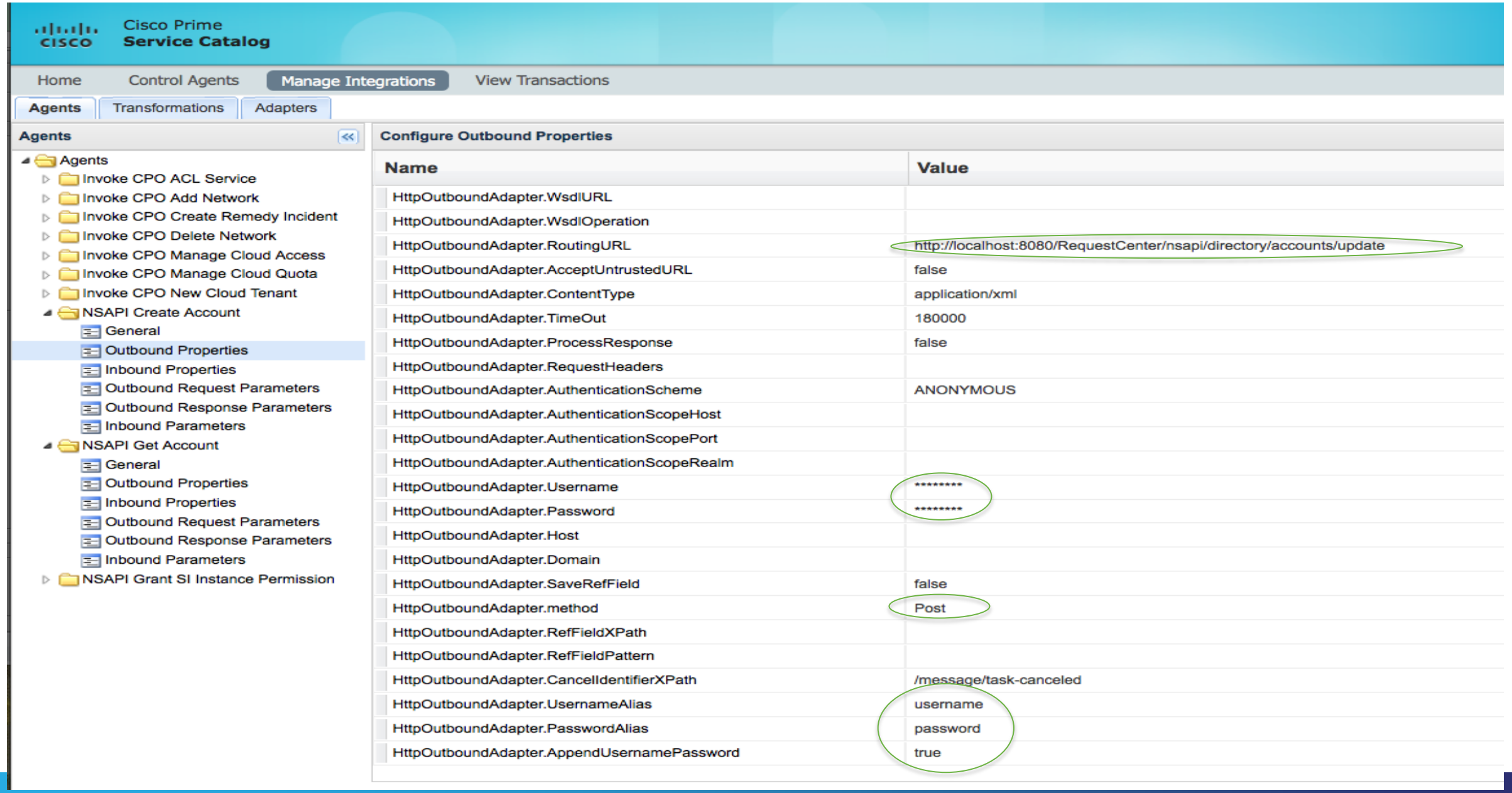
Configuring Service Link to call REST APIs #2

- You can use the header to send username and password from dictionary via agent parameters into namespaces
- If sending username and password in the header (encrypted with HTTPS), then set the AuthenticationScheme to ANONYMOUS.
- If using SSO via HTTP header, set AuthenticationScheme to BASIC and send the credentials via HTTP header.
- If using SSO via remote user, set AuthenticationScheme to NTLM or NTLM v2.
- Specify username, password, host, domain in HttpOutboundAdapter.
- Specify HttpOutboundAdapter.method = POST, GET, DELETE, PUT
- In the Agent, specify the namespaces in Outbound Request Parameters. If the data is being sent from dictionaries, choose Dictionaries.

Configuring Service Link to call REST APIs #3

- Pick a dictionary, a field from the dictionary, and then put into the Service Data Mapping field and create the corresponding Parameter. Apply and then save.
- After this you may have a Service Data (dictionary) field #AuthDictionary.billingID# that has been mapped to an Agent parameter “billingID”.
- If expecting anything back in the same call, set the Outbound Response Parameters similarly to receive the response in dictionary fields.
- If the outbound call does not have the hang around polling for a result, in the Agent General tab, set Inbound Adapter to “None (Auto complete).
- If the RESTful API you are calling expects a parameter as part of the URL (e.g. billing ID), you can set HttpOutboundAdapter.RoutingURL to https://api.rest-server.com/1.0/auth/#bilingID# (billing ID was setup as an Agent parameter in the Outbound Request Parameters in bullet point #11 above).

Example: Agent properties for an agent that invokes the create/update account nsAPI



Example: Outbound Transformation (for forming the REST request payload)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" />
   <xsl:template match="/">
<account>
  <name>
    <xsl:value-of select="/message/task-started/agent-parameter[name='accountName']/value" />
  </name>
  <accountType>
    <xsl:value-of select="/message/task-started/agent-parameter[name='accountType']/value" />
  </accountType>
  <description>
    <xsl:value-of select="/message/task-started/agent-parameter[name='description']/value" />
  </description>
  <billingRateGroup>
    <xsl:value-of select="/message/task-started/agent-parameter[name='BillingRateGroup']/value" />
  </billingRateGroup>

  <customAttribute name="ProviderTarget">
    <xsl:value-of select="/message/task-started/agent-parameter[name='ProviderTarget']/value" />
  </customAttribute>
   :
  <customAttribute name="BillToOUID">
    <xsl:value-of select="/message/task-started/agent-parameter[name='BillToOrganizationID']/value" />
  </customAttribute>

  <organizationalUnits>
    <organizationalUnit>
      <organizationalUnitName>
        <xsl:value-of select="/message/task-started/agent-parameter[name='accountName']/value" />
      </organizationalUnitName>
    <organizationalUnit>
  </organizationalUnits>
</account>
</xsl:template>
</xsl:stylesheet>
```

Example: Agent properties for an agent that invokes the read account nsAPI

Cisco Prime
Service Catalog

Home | Control Agents | **Manage Integrations** | View Transactions

Agents | Transformations | Adapters

Agents

- Agents
  - Invoke CPO ACL Service
  - Invoke CPO Add Network
  - Invoke CPO Create Remedy Incident
  - Invoke CPO Delete Network
  - Invoke CPO Manage Cloud Access
  - Invoke CPO Manage Cloud Quota
  - Invoke CPO New Cloud Tenant
  - NSAPI Create Account
    - General
    - Outbound Properties
    - Inbound Properties
    - Outbound Request Parameters
    - Outbound Response Parameters
    - Inbound Parameters
  - NSAPI Get Account
    - General
    - Outbound Properties
    - Inbound Properties
    - Outbound Request Parameters
    - Outbound Response Parameters
    - Inbound Parameters
  - NSAPI Grant SI Instance Permission

Configure Outbound Properties

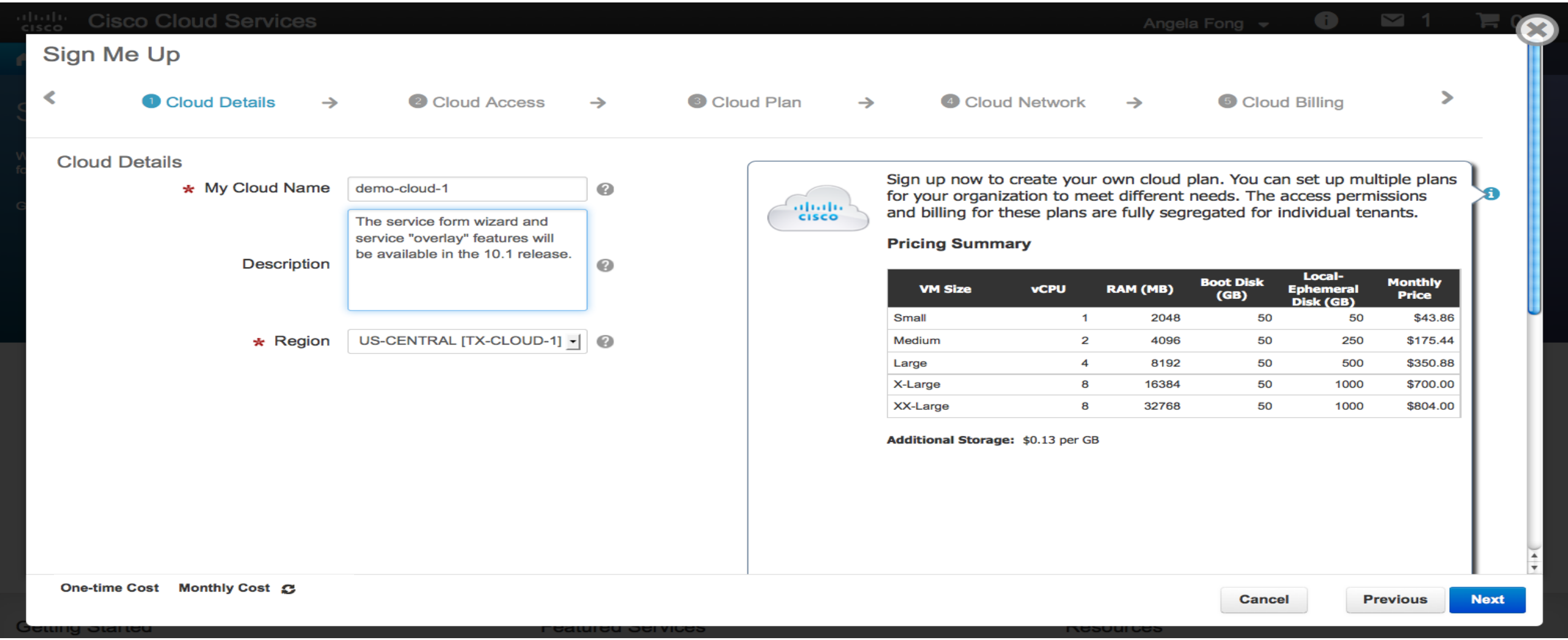
Name	Value
HttpOutboundAdapter.WsdlURL	
HttpOutboundAdapter.WsdlOperation	
HttpOutboundAdapter.RoutingURL	http://localhost:8080//RequestCenter/nsapi/directory/accounts/name/$AccountName$
HttpOutboundAdapter.AcceptUntrustedURL	false
HttpOutboundAdapter.ContentType	application/xml
HttpOutboundAdapter.TimeOut	180000
HttpOutboundAdapter.ProcessResponse	true
HttpOutboundAdapter.RequestHeaders	
HttpOutboundAdapter.AuthenticationScheme	BASIC
HttpOutboundAdapter.AuthenticationScopeHost	
HttpOutboundAdapter.AuthenticationScopePort	
HttpOutboundAdapter.AuthenticationScopeRealm	
HttpOutboundAdapter.Username	********
HttpOutboundAdapter.Password	********
HttpOutboundAdapter.Host	
HttpOutboundAdapter.Domain	
HttpOutboundAdapter.SaveRefField	false
HttpOutboundAdapter.method	Get
HttpOutboundAdapter.RefFieldXPath	
HttpOutboundAdapter.RefFieldPattern	
HttpOutboundAdapter.CancelIdentifierXPath	/message/task-canceled
HttpOutboundAdapter.UsernameAlias	username
HttpOutboundAdapter.PasswordAlias	password
HttpOutboundAdapter.AppendUsernamePassword	true

Example: Inbound Transformation (for capturing some data element from the REST response and updating it back to the service form)

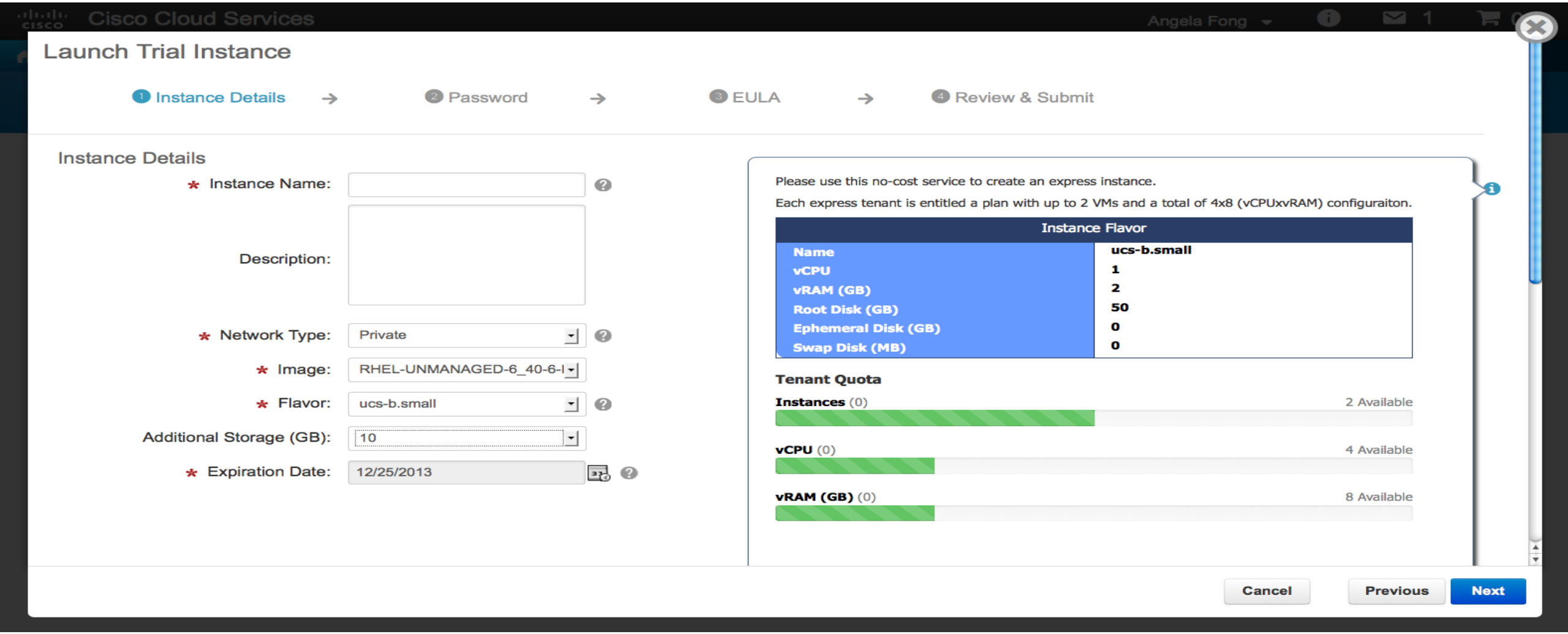
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" />
   <xsl:template match="/">
<message>
   <xsl:attribute name="channel-id">
   <xsl:value-of select="//@*[local-name()='channel-id']" />
   </xsl:attribute>
   <send-parameters>
      <agent-parameter>
         <name>AccountID</name>
         <value><xsl:value-of select="/message/account/accountId"/></value>
      </agent-parameter>
</send-parameters>
</message>
</xsl:template>
</xsl:stylesheet>
```

Preview of 10.1 Enhancements – UI and Billing Rates

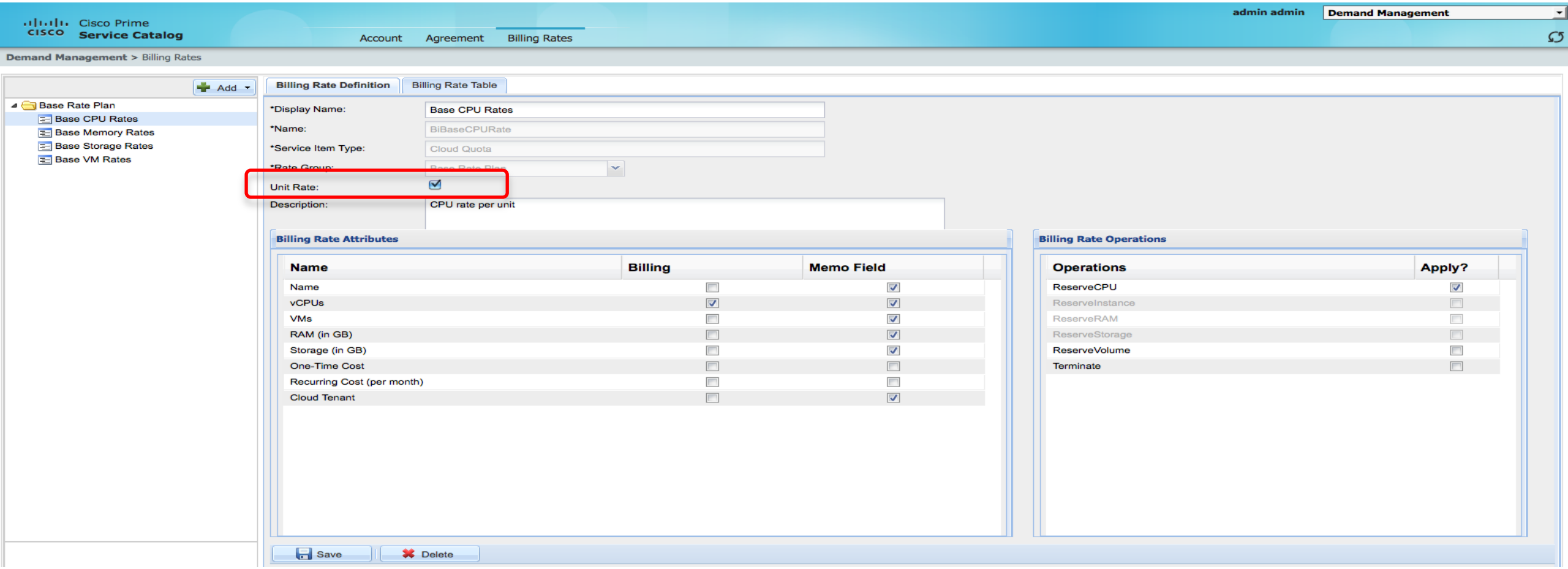
Coming up next: New service form look and feel



Coming up next: New service form look and feel (more)



Coming up next: Support for “Unit Rate” calculation



Thank you.