Namespaces

This appendix contains the following topics:

Namespaces

The Business Engine provides facilities for service designers and system administrators to customize the contents of emails and to dynamically configure the progress of a requisition through the approval and delivery process by specifying conditional workflows.

In order to implement this capability, Service Catalog provides a unified way to access the values of data elements, such as a request's initiator or the service form data related to a request. This unified way is referred as the Business Engine Namespace.

Namespace functionality increases the flexibility of the service design, by allowing the delivery plan to incorporate approvals or tasks that are executed conditionally, depending on the current value of a namespace variable, or to dynamically adjust the routing of a particular task. It also allows the content, subject, and addresses of emails to be dynamically adjusted.

Note The use of grid dictionary fields for Business Engine namespace is not supported.

Namespace is a term used to describe a set of valid names that address the data objects used within Service Catalog, exposing these objects to service designers. This allows designers to use these elements in these contexts:

- Within an email, to dynamically resolve the recipient, subject, or references within the email body.
- To conditionally execute reviews, authorizations, or tasks in a delivery plan.
- In expressions which determine the person or queue to which an authorization or delivery task is assigned.
- In task names.

Namespaces are hierarchical. Namespace names reflect the structure of the data which defines a service, and the data entered on the service form when that service is requested. The key to manipulating namespace variables is knowledge of the hierarchy which defines Service Catalog data and the contexts in which particular Namespace variables can be used.

Each element in the hierarchy is case-insensitive. The elements are separated by periods ("."). The last element may also be referred to as a "property". All elements before the last one are "nodes" in the hierarchy.

This chapter describes Namespace in detail, its standards and implementation.

References

When Namespaces are used in a textual context, as in an email template or an assignment expression, delimiters are required to differentiate the Namespace from surrounding text. This delimiter is the character "#" surrounding the namespace references.

EXAMPLES:

Table 1: References

For emails:	Dear #Customer.Firstname#, We are pleased to inform you that the service you requested has been approved...
For expressions:	CN=#Service.Data.Initiator_Information.First_Name#
For conditions:	ActualCost > 2000.00

Nodes

Every node in the hierarchy has a node type. The type determines the subnodes and properties available for that element.

Node types are summarized in the table below.

Table 2: Nodes

Node Type	Description
Activity	Information about an activity (task), including its priority, status, and scheduled and actual start date, completion date, and duration
Service Form Data	Data in the fields in dictionaries used in a service
OrganizationalUnit	Information about an organizational unit
Person	Information about a person or queue, including name and, for person, company affiliation and contact information
Process	Information about a process (task), including its scheduled and actual dates and duration, and its current status and escalation level

Node Type	Description
Requisition	Information about a service request (shopping cart), including the start date and expected duration
Service	Information about the definition of the service requested, including form data
Message	Information about a failed Service Link task

The OrganizationalUnit (OU) and Person node types have multiple instances within the Namespace. For example, the Person node appears as the requisition's Customer and Initiator, as well as in many other contexts. In this case the “PersonType” is used to reference data for the appropriate person.

The Service node, and its children, relates to a particular service request. Elements in these nodes are available only in contexts when there is a current service request. In particular, emails and design configuration details pertaining to site- and organization-level authorizations, which apply to a complete shopping cart, rather than to an individual request, should not contain Service nodes. Similarly, the Order Confirmation email Template, which pertains to the requisition rather than to individual service requests, should not contain Service nodes.

Details on PersonTypes, as well as the hierarchical structure of the namespace and the properties and subnodes available for each node are summarized in the Person-Based Namespaces, on page 29.

All dates are maintained in Greenwich Mean Time (GMT). Date data includes both the date and the time. All dates are presented in the format:

YYYY-MM-DD mm:HH:ss SSS

That is, a 4-digit year, 2-digit month, 2-digit (zero-filled) day, followed by minutes, hours, seconds, and fractions of seconds using a 24-hour clock. For example, “2007-06-27 23:19:39.197” corresponds to June 27, 2007 at 11:19 PM.

Expressions

Authorizations, reviews, and delivery tasks may be assigned to a person or queue or by using an expression. The expression provides a means to identify the person or queue, stored within Organization Designer, to whom a task should be dynamically assigned.

Configuring an Expression

The person/queue may be identified by using one of four assignment types:

Table 3: Assignment Types

Assignment Type	Meaning
CN	“Common name” – The full name of a user, which is expressed as Firstname Lastname
ID	User ID; the unique numeric identifier of a user

Assignment Type	Meaning
LOGINNAME	The login name of a user
QUEUE	The name of a queue defined in Organization Designer

Expression-based task assignments use the format:

```
AssignmentType=#Namespace_Expression#
```

The assignment statement should follow the rules below:

- The assignment operator (=) must immediately follow the AssignmentType and immediately precede the Namespace_Expression; no spaces are allowed.
- Namespaces used in the Namespace_Expression must be enclosed in hash marks (#).
- Expressions can consist of a combination of namespaces and constant data to create a person's full name or the name of a queue. Separate expression elements by a single space.
- Do not include spaces in variable and functional position names, so that the expressions can evaluate properly.
- All lookups into the database using the result of an expression are case insensitive. For example, LOGINNAME assignment types in which the expression evaluated to "Ann" and "ann" would yield identical results; both would find a person whose login name was "ann"—or "Ann" or "ANN".

CN (Common Name) Assignments

The CN assignment type attempts to find a person by matching the specified full name against the data stored in Organization Designer.

EXAMPLES:

```
CN=#Service.Data.Customer_Information.First_Name#
#Service.Data.Customer_Information.Last_Name#
CN=#Customer.FirstName# #Customer.LastName#
```

The values in the Namespace Expression must exactly match the name of the desired user as stored in Organization Designer. For example, a CN expression whose value is "Carroll Hastings" will not match a user whose name in Organization Designer is "Carol Hastings". For this reason, you should not use dictionary fields into which users have been allowed to type data as free-format text. Instead, use dictionary fields that have been populated by using a Person data type, as this allows the user to choose people from a dialog box, eliminating typing errors. Alternatively, other Namespaces, such as the Customer, Initiator, or Performer, may be used if appropriate.

Use caution when using a CN assignment. It should not be used if two people could have identical first and last names. The assignment will always pick the first person (the one with the lowest PersonID) with the specified name.

When a field in a dictionary is defined as a Person data type, as is the "Select_Person" field in a person-based dictionary, its value is populated by choosing a person from the Select Person search box. The value is set to the unique identifier of the person in Organization Designer. This "ID" value may be referenced in a CN assignment.

EXAMPLE:

```
CN=#Service.Data.Approver.Select_Person#
```

ID (Identifier) Assignments

The PersonID for any of the Person nodes available via Namespace may be referenced in an ID assignment. Alternatively, a text field to which a PersonID has been copied or assigned may be used.

EXAMPLE:

```
ID=#Service.Data.TT_Contact.SupervisorID#
```

LOGINNAME Assignments

Login names uniquely identify a person in the application and can safely be used for assignments.

EXAMPLES:

```
LOGINNAME=#Service.Data.Customer_Information.Login_ID#
LOGINNAME=#Customer.HomeOU.Manager.LoginName#
```

QUEUE Assignments

A queue must sometimes be assigned based not only on a service team, but also on the location of the customer. Such location-based queues can be accommodated by using a QUEUE assignment.

EXAMPLE:

```
QUEUE=#Service.Data.RC_Queue.Location# Desktop Services
```

This assignment would direct the task or approval to a queue named "*Location* Desktop Services," where *Location* is the current value of the Location field in the RC_Queue dictionary on the service form. The resultant value for the QUEUE must *exactly* match the name assigned to a queue defined in Organization Designer. The match is case sensitive. If a match is not found, the task is directed to an Unassigned Work Queue.

Namespace Usage in an Email Template

Namespaces can be used in the following sections of an email template:

- Subject – the subject of the email
- To(s) – the addressees/recipients of the email
- Body – the text of the email

The Namespace data available depends on the context in which the email is sent. For example, task-level data (such as Process or Activity details) will not be available for use in an email that is triggered by a Requisition-level event, such as a Financial Authorization. See the Namespace Reference, on page 24 for details as to which Namespaces are allowable in which emails.

Defining an Email Template

The email Template definition page is accessed from the Administration menu:

Choose **Administration** > **Notifications** > **Email Templates**.

To preview email templates:

Step 1 Start the Service Designer module.

Step 2 From the Services component, choose a service from the left side of the screen.

Step 3 Click the **Plan** tab.

Step 4 Click the **Email** subtab.

Step 5 Assign the email template you wish to preview to one of the moments listed in the drop-down lists in the email subtab area.

Step 6 Click the corresponding **Preview** button to the right of the email template you just assigned.

Whitelist Images in Email Template

You can add external images to the email template for the notification emails generated from Service Catalog. If you find any missing image in the generated email template, then you must whitelist the resources so that authentication is not required to display an image. The ImageServlet generates a url with the documentId for every image that is uploaded in the request center. You can whitelist these images based on their documentId.

To display the images in email template:

Step 1 Open the file '**signon-resources.xml**' from the path ../Cisco Service Portal/deployments/RequestCenter.war/WEB-INF/classes/xml/.

Step 2 Search for 'ExcludedResources'.

Step 3 Add the following ResourceDefinitions to the xml file under ExcludedResources.

Example:

```
 <ResourceDefinition> /* For imageservlet */
      <Name>ImageServlet</Name>
<UrlPattern>/RequestCenter/imageServlet.img?tenantId=1&amp;type=def&amp;documentId=1</UrlPattern>
    </ResourceDefinition>
<ResourceDefinition> /* For other images */
      <Name>Common PNG</Name>
      <UrlPattern>/RequestCenter/images/<file name>.png</UrlPattern>
    </ResourceDefinition>
```

Step 4 Ensure to delete the CnfFile table entries from the database table. Run the query : Delete from CnfFile where LogicName = 'signon-resources.config'.

Step 5 Save and close the xml file.

Step 6 Restart the server for the changes to take effect.

Note Add only individual image files to exclude from authentication and not the entire image folder from ImageServlet, as this will result in a security compromise. To find the documentId of a particular image, click **Source** in Email Template editor, after the image has been uploaded.

Recipients

Namespaces can be used to send emails to people with an interest in the requisition.

Table 4: Recipients

Namespace Example	Resolves To
#Requisition.Customer.Email#	The customer’s email, maintained on the To(s): field on the Notifications tab of the Administration module.
#Requisition.Customer.Supervisor.Email#	The email of the customer’s supervisor.
#Alternate.Email#	The approver’s authorization delegate. The delegate can be assigned via the user's Profile.
#Performer.Email#	The task performer’s email, where the performer could be either a person or queue. The person’s and queue’s email can be found on the To(s): field on the Notifications tab of the Administration module.
#Position.EscalationManager.Email#	The escalation manager's email, where Escalation Manager is a user-defined functional position that has been associated, for example, with a Service Group, and to which a person on the Service Team has been assigned.
#Service.Data.**DictionaryName.FieldName#**	The current value of the specified field in the specified dictionary in the service form. The field should be validated to hold an email address.

Subject

In addition to hard coding data into the subject line of the email template, namespaces can be used to make the template more specific to the requisition (while still maintaining a consistent look throughout services).

Table 5: Subject Namespace

Namespace Example	Resolves To
#Requisition.RequisitionID#	The requisition number

Namespace Example	Resolves To
#Requisition.Customer.FirstName#	The customer’s first name
#Service.ServiceDefinition.Name#	The service name

Body

As with the subject field, the body of an email can also be configured to include requisition data, task data, service data, or service form data.

Table 6: Body Namespace

Namespace Example	Resolves To
#Requisition.RequisitionID#	The requisition number.
#Service.Data.**DictionaryName.FieldName#**	The current value of the specified field in the specified dictionary on the current service form.
#Site.URL#	The URL to access Service Catalog; this will resolve to the user's default view, which is set under **Profiles > Preferences**.
#Site.URL#myservices/myservice.do?	A link to the My Services module.
#Site.URL#servicemanager/homepage.do	A link to the Service Manager module.
#URL#	A link to the Task Details (service form) page of the task in Service Manager.

Site URL Namespaces

The namespace #Site.URL# designates the URL through which Service Catalog is accessed. The URL path may also include a specific module to run as well as parameters (following the question mark) to pass to that module. The examples above generate a hyperlink in the email body that will direct the user to the My Services module, the Service Manager module or to capabilities available within those modules.

It is possible to create a namespace reference that will link to just about any page in My Services or Service Manager. The key to determining the appropriate URL is to have a site administrator temporarily turn on site debugging (but please read the caveats in the Cisco Prime Service Catalog Administration and Operations Guide carefully) and note the URL that appears when you navigate to the desired page. When embedding that URL in an email, you must use an encoded representation of characters such as ampersand (&).

Additional examples of such namespace references are given below.

Namespace Example	Resolves To
#Site.URL#myservices/navigate.do?query=requisitionentry status&amp;reqid=#Service.Requisition.RequisitionID#&amp;reqentryid= #Service.RequisitionEntryID#&amp;formAction= displayEntryStatus&amp;serviceid=#Service.ServiceID#&amp;requisitionId= #Service.Requisition.RequisitionID#&amp;waiting=0&amp;**authTaskType=4**&amp;activityId= #ActivityID#&amp;buttonsPresent=true&amp;	A link to the service authorization page within My Services with the current approval/review task selected. The Authorization Task Type of 4 refers to a Service Group authorization.
#Site.URL#myservices/navigate.do?query=authorizationtask&amp;taskId=#ActivityID #&amp;return_to_url=authorizationslist&	A link to the Service Group Authorization identified by ActivityID, which will take the user directly to the requisition containing the service requiring approval.
#Site.URL#myservices/navigate.do?query=authorizationlist&amp;return_to_url=portal	A link to the authorizations list page within My Services where the user will see all the authorizations awaiting approval.
#Site.URL#myservices/navigate.do?query=requisition&amp;requisitionId= #Requisition.RequisitionID#	A link the requisition confirmation page in My Services.

Site URL Configuration

As part of the installation process, administrators specify the Service Catalog URL. This URL is stored in the configuration file newscale.properties. A sample entry is shown below.

```
!-----------------------------------------------------------------
!Define the URL for the application
!This is used in emails for constructing the various urls
```

```
!-----------------------------------------------------------------
ObjectCache.Application.URL=http://appsrv01.celosis.com/RequestCenter/
```

The entry for ObjectCache.Application.URL is the value of the namespace #Site.URL# and #URL#. The URL typically ends with "/RequestCenter/"; however, it is possible to configure the application server to automatically reroute requests to /RequestCenter/, so this portion of the path may be absent from the properties file.

In this case, the word **/RequestCenter/** must be added after #Site.URL#, such as #Site.URL#/RequestCenter/myservices/myservice.do? , and to #URL#, such as #URL#/RequestCenter/myservices/myservice.do? to ensure that the namespace link resolves to the correct URL.

In order to determine if the /RequestCenter reference must be inserted, you may consult the newscale.properties file (or have the application server administrator report on its contents). An alternate method is:

- Enter #Site.URL# or #URL# by itself in an email and send the email when, for example, a task starts.
- When receiving the email, note if the URL resolves to just http://www.mycompany.com or http://www.mycompany.com/RequestCenter/.
- If the resolved URL does not include the wording **/RequestCenter/**, then add **/RequestCenter/** after *#Site.URL#* or *#URL#* respectively.

Service Link Message Namespaces

The namespace node #Message# is available only within the body of email messages generated as a result of a Service Link failure to deliver an outbound message. The template to be used is specified as the "Failed email" in the Service Link agent definition. Elements in this node may be useful in helping a administrator diagnose the cause of the failure.

Service Manager Task Details URL

The #URL# namespace takes the user directly to the form data of a task in Service Manager. This is very nice for Ad-hoc tasks, and for teams that rarely perform tasks in Service Catalog. It can also be used for authorization tasks, but after the approver approves/denies the service, they find themselves in Service Manager, which can be confusing.

Some customers prefer this to the multistep process required to view the form data when approving via the #Site.URL#myservices/navigate.do?query=authorizationtask&amp;taskId=#ActivityID# namespace. In fact, here, an approver can approve without seeing the form data which could be a slight audit issue depending if the data is important to the approval—they have to know to click on the name of the service to see the form data.

Any change to the ObjectCache.Application.URL in the newscale.properties file will also affect the #URL# namespace in the same way as the #Site.URL#.

Namespace Processing and Alternate Values

When a Namespace element is referenced, Service Catalog retrieves its value from the current context and replaces the reference to the Namespace in the email with the resolved value. Some Namespace elements are always guaranteed to exist and have a valid value, such as those relating to the customer or initiator of a request. Other Namespace elements, however, may be undefined at certain times. For example, a functional position may be vacant; no authorization delegate has been designated; or an employee is temporarily without a supervisor.

Any reference to a Namespace that is undefined is blank. Except for the email namespace, it is possible to provide an alternate value, by using a slash (/) immediately following the Namespace element name, and then assigning the alternate value.

EXAMPLE:

```
TO: #Alternate.email# #Performer.email#
```

- Design emails to be sent to both the designated reviewer and a possible alternate (authorization delegate). If no delegate is currently designated, the #Alternate.email# is blank

```
#Supervisor.LastName/Your current supervisor#
```

- If the person referenced currently has a supervisor, the supervisor's last name will appear in the email; otherwise, the string "Your current supervisor" will appear.

Namespace Usage for Policy Configuration

Namespaces are used when you configure an action for a service item policy. The actions that use namespaces are policy alert, send email, and order service. These namespaces are resolved during execution.

For more information about policies, see Understanding Service Items Policies.

Table 7: Namespace Example

Namespace Example
#SI.Subscription.ID#
#SI.Subscription.OrganizationalUnit.ID#
#SI.Subscription.OrganizationalUnit.Name#
#SI.Subscription.Customer.Name#
#SI.Subscription.Customer.LoginName#
#SI.Subscription.Customer.Email#
#SI.Subscription.AssignedDate#
#SI.Subscription.SubmittedDate#
#SI.Subscription.RequisitionID#
#SI.Subscription.RequisitionEntryID#
#SI.Subscription.Operation#
Account Namespaces
#SI.Subscription.Account.ID#

Namespace Example
#SI.Subscription.Account.Name#
#SI.Subscription.Account.FunctionalPosition.<FPName>#.ID
#SI.Subscription.Account.FunctionalPosition.<FPName>#.Name
#SI.Subscription.Account.FunctionalPosition.<FPName>#.LoginName
#SI.Subscription.Account.FunctionalPosition.<FPName>#.Email
Agreement Namespaces
#SI.Subscription.Agreement.ID#
#SI.Subscription.Agreement.Name#
Service Item Attributes Namespaces
#SI.Type.ID#
#SI.Type.Name#
#SI.Type.DisplayName#
#SI.Type.Classification.ID#
#SI.Type.Classification.Name#
#SI.Attribute.<AttributeName>#
Policy Attributes
#Policy.ID#
#Policy.Name#
#Policy.Type#
#Policy.Parameter.Attribute#
#Policy.Parameter.Threshold#
#Policy.Parameter.Operator#
#Policy.Parameter.Hours#
#Policy.Parameter.Value#

Namespace Example
Account Namespaces
#Policy.Account.ID#
#Policy.Account.Name#
#Policy.Account.FunctionalPosition<FPName>.ID#
#Policy.Account.FunctionalPosition<FPName>.Name#
#Policy.Account.FunctionalPosition<FPName>.LoginName#
#Policy.Account.FunctionalPosition<FPName>.Email#

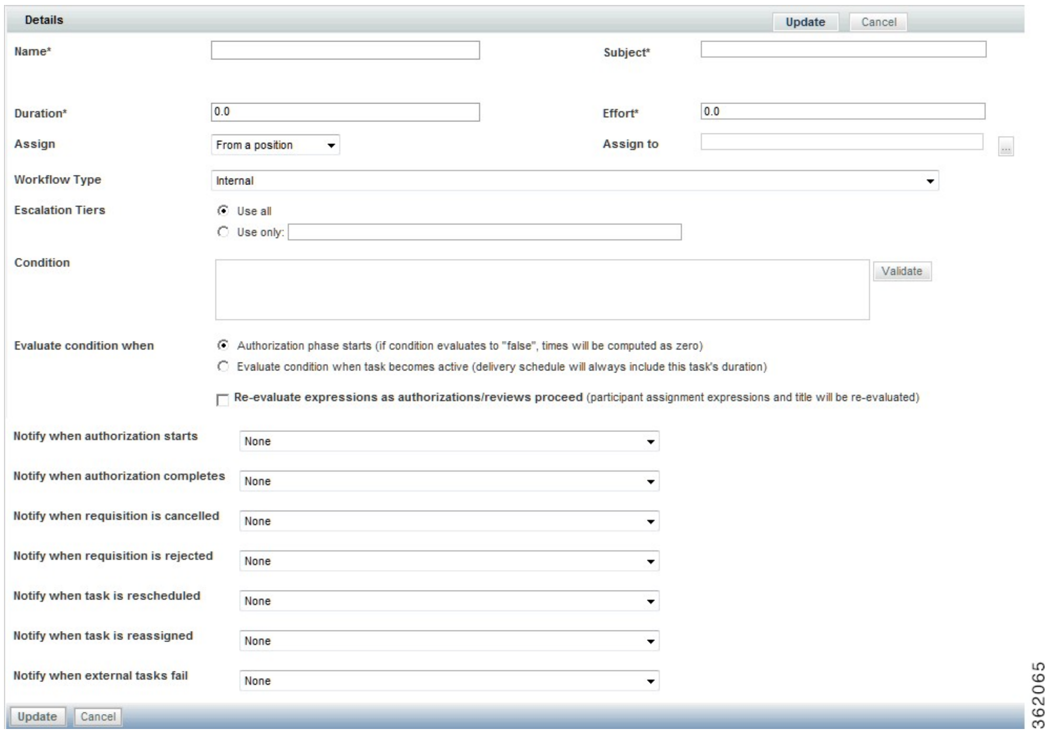
Namespace Usage for Authorizations and Reviews

Namespaces can be used in the following components of an authorization/review definition:

- Subject: Namespaces can be used to include form data in the task name
- Assign to: The authorization/review may be assigned from an expression.
- Condition: The authorization/review task may be conditionally performed

The subtab for specifying the details of an authorization or review looks like the figure below:

Figure 1: Service Definition Subtab for an Authorization Details



Details

Name*		Subject*	
Duration*	0.0	Effort*	0.0
Assign	From a position	Assign to	
Workflow Type	Internal		
Escalation Tiers	(•) Use all ( ) Use only:		
Condition			
Evaluate condition when	(•) Authorization phase starts (if condition evaluates to "false", times will be computed as zero) ( ) Evaluate condition when task becomes active (delivery schedule will always include this task's duration) ☐ **Re-evaluate expressions as authorizations/reviews proceed** (participant assignment expressions and title will be re-evaluated)		
Notify when authorization starts	None		
Notify when authorization completes	None		
Notify when requisition is cancelled	None		
Notify when requisition is rejected	None		
Notify when task is rescheduled	None		
Notify when task is reassigned	None		
Notify when external tasks fail	None		

Subject

The subject of an authorization or review task can be configured to reflect requisition data through the use of namespaces, thus making the subject specific to the customer’s order. Simply include the namespace in the Subject field.

The most frequently used namespace is the name of the service to which the current review/authorization applies. This is typically concatenated with other descriptive text; for example:

```
Group Review for #Name#
```

The namespace #Name# is available as a shortened form for the Service Name.

Only namespaces referring to a specific attribute of the service and **NOT** the content of a service form can be used for Organizational Unit Reviews or Authorizations or Financial Authorizations. For instance, when using a namespace to refer to the customer’s first name:

```
#Requisition.Customer.FirstName# -- CORRECT
#Service.Data.Customer_Information.First_Name# -- INCORRECT
```

Service Group Reviews and Authorizations may use #Service.Data# namespaces. Such reviews/authorizations apply to an individual service, so the service (form) data is available. Financial Authorizations and Organizational Unit Reviews/Authorizations apply to an entire requisition, which may include multiple services (requisition entries); therefore, the service data for an individual service is not available.

Assigning from an Expression

Authorization and review tasks can be assigned to:

- the person currently filling a *functional position* defined in Organization Designer.
- a static person or queue.
- the person or queue identified by the value of an *expression* .

In many cases, assigning tasks to a predefined position, individual, or queue is not a viable option. For example, the same service may be handled by different queues, depending on the location of the requestor. In cases like these, you may need to route the tasks based on requisition data entered or otherwise supplied in the ordering moment.

Step 1 Choose **From an expression** in the Assign drop-down list.

Step 2 Enter the desired expression in the “Assign to” field.

Example:

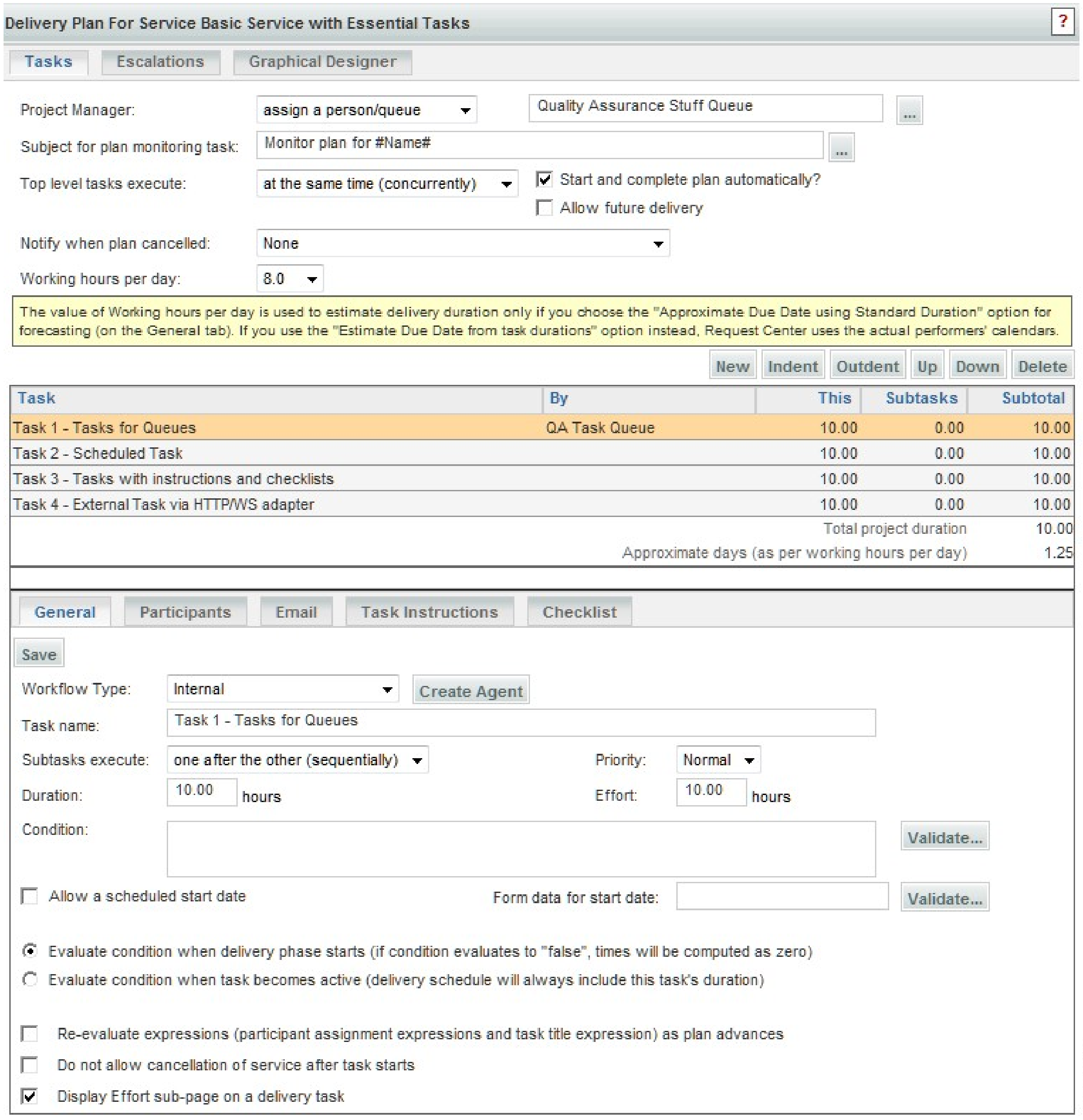
```
ID=#Service.Data.Customer_Information.SupervisorID#
```

Step 3 Click **Update** to save.

Delivery Plans and Tasks

The delivery plan is configured via the Plan tab of **Service Designer** > **Services**.

Figure 2: Monitor Task Definition



Delivery Plan For Service Basic Service with Essential Tasks

Project Manager:	assign a person/queue — Quality Assurance Stuff Queue
Subject for plan monitoring task:	Monitor plan for #Name#
Top level tasks execute:	at the same time (concurrently)
Notify when plan cancelled:	None
Working hours per day:	8.0

[✓] Start and complete plan automatically?
[ ] Allow future delivery

The value of Working hours per day is used to estimate delivery duration only if you choose the "Approximate Due Date using Standard Duration" option for forecasting (on the General tab). If you use the "Estimate Due Date from task durations" option instead, Request Center uses the actual performers' calendars.

Task	By	This	Subtasks	Subtotal
Task 1 - Tasks for Queues	QA Task Queue	10.00	0.00	10.00
Task 2 - Scheduled Task		10.00	0.00	10.00
Task 3 - Tasks with instructions and checklists		10.00	0.00	10.00
Task 4 - External Task via HTTP/WS adapter		10.00	0.00	10.00
			Total project duration	10.00
			Approximate days (as per working hours per day)	1.25

Workflow Type:	Internal		
Task name:	Task 1 - Tasks for Queues		
Subtasks execute:	one after the other (sequentially)	Priority:	Normal
Duration:	10.00 hours	Effort:	10.00 hours
Condition:			

[ ] Allow a scheduled start date — Form data for start date:

(●) Evaluate condition when delivery phase starts (if condition evaluates to "false", times will be computed as zero)
( ) Evaluate condition when task becomes active (delivery schedule will always include this task's duration)

[ ] Re-evaluate expressions (participant assignment expressions and task title expression) as plan advances
[ ] Do not allow cancellation of service after task starts
[✓] Display Effort sub-page on a delivery task

362067

Each task is configured by choosing the task from the list of tasks (or clicking New to create a new task) and filling out the task-related subtabs.

For more information about delivery plan, see Configuring AMQP Tasks for Publishing Service Request to an External System.

Namespace Usage for Delivery Tasks

Configuring a service's delivery plan and component tasks provides multiple opportunities to use expressions and namespaces.

Namespaces can be used in the following components of an authorization/review definition:

- Project Manager: May be assigned from a namespace expression.
- Subject for Monitor Plan: May include namespaces as well as literal text.
- Task name: May include namespaces as well as literal text.
- Participants: Performer and supervisor may be assigned from a namespace expression.
- Condition: The task may be performed only if the specified condition, which may use namespaces, is true.

Project Manager for the Delivery Plan

The Project Manager may be assigned from a person, queue, or expression. The expression may use any of the Assignment Types to identify a person in Organization Designer.

EXAMPLES:

```
LOGINNAME=#Service.Data.Project.ManagerLogin#
ID=#Customer.Supervisor.ID#
```

Monitoring Task Subject

The subject for the monitoring task typically includes the #Name# namespace, to refer to the name of the service being monitored.

```
Monitor Plan for #Name#
```

Namespaces referring to performers, either people or queues, cannot be used.

Namespaces referring to the content of a service form can be used, but this may have side effects, as documented for the Delivery Task Name below.

Delivery Task Name

The delivery task name typically includes the service name, through the use of the #Name# Namespace. Service data (#Service.Data....#) can also be used. Since all tasks are created when the form is submitted, the value of the specified field must have been supplied in the ordering moment.

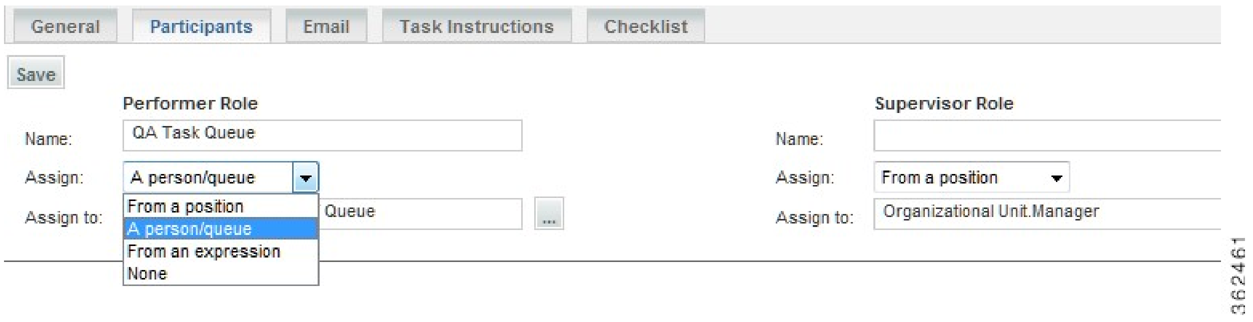
Note Using form data for a task name is not best practice, since the task would no longer be groupable by the task name in a reporting environment. Further, a Service Manager query would need to use the "contains" operator to retrieve the rest of the task name, which may adversely affect performance.

Assigning Roles (Performers and Supervisors) to the Task

As with authorizations, the service team or person assigned to tasks in the delivery plan may depend on data on each requisition, such as a customer's location. Expressions can be used to intelligently route tasks, rather than creating different forms or workflows for each possible scenario.

Plan tasks that can be dynamically routed include the Performer and Supervisor roles found on the Participants tab of each task:

Figure 3: Delivery Plan Task Participants



To assign "From an expression", choose that option from the Assign drop-down list and click on the ellipsis (...) next to the "Assign to" field. The Edit Expression window appears to allow you to enter and validate the expression.

Conditional Statements

A conditional statement allows tasks to be initiated or skipped based on the condition and can be configured to evaluate at various times within the authorization, review, and delivery plan.

Namespaces used in conditional statements are not enclosed in # signs:

```
Service.Data.DictionaryName.FieldName="Yes" -- CORRECT
#Service.Data.DictionaryName.FieldName#="Yes" -- INCORRECT
```

Conditional Authorization and Review Tasks

Authorizations and reviews may be needed only when certain conditions exist. For example, an authorization may be required only when a unit price exceeds a specified threshold. To define a conditional authorization or review task, follow the procedure below.

Step 1 Click on the authorization/review you are configuring. The Review/Authorization Details window appears, as shown in Figure 4: Review Details Window.

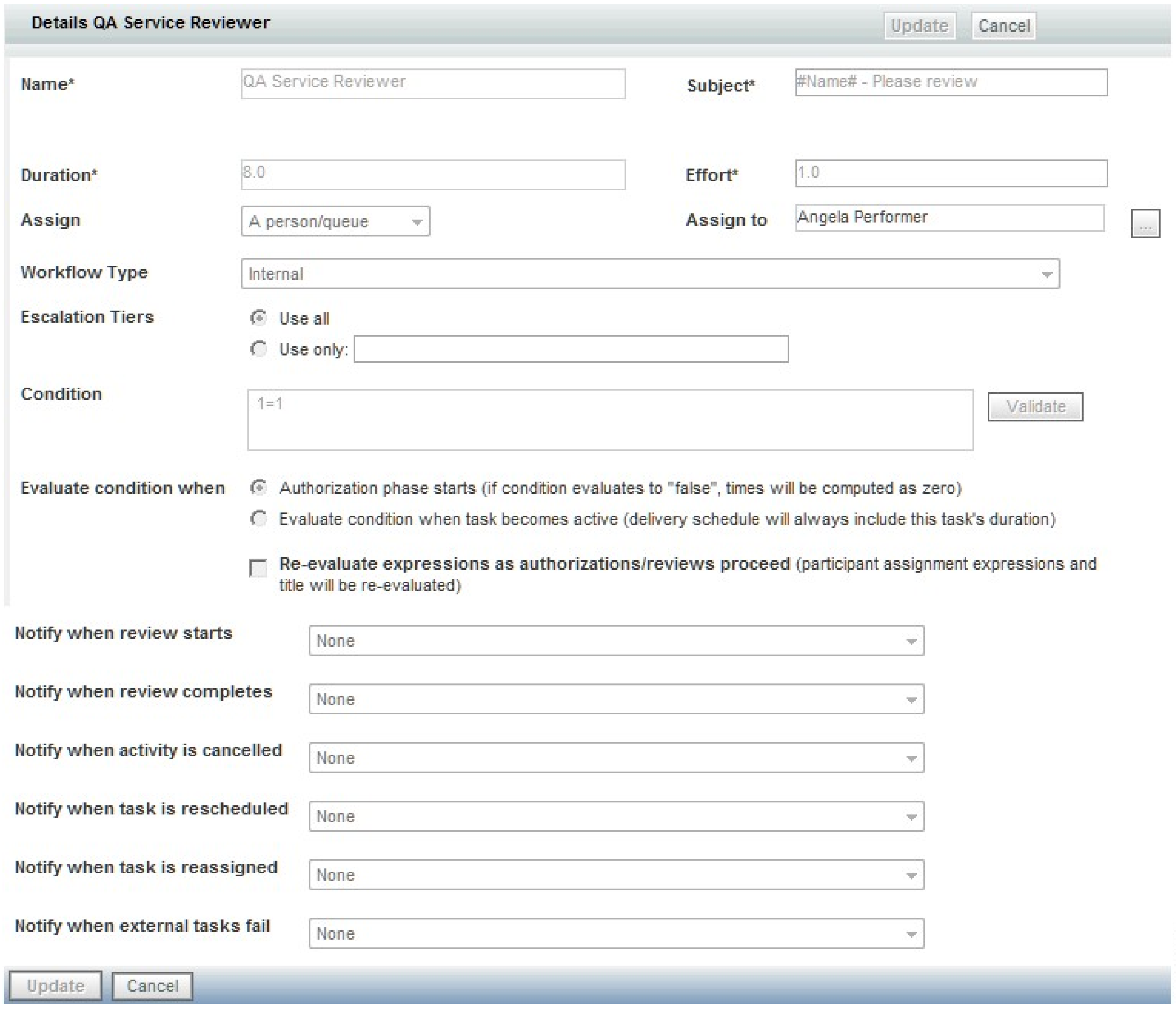
Step 2 Enter the Condition under which the particular authorization or review task should be executed.

Step 3 Validate the condition by clicking **Validate**.
A Validate window appears. The message "unexpected token" indicates that the namespace used is not valid in this context or, perhaps, that you have forgotten to enclose an alphanumeric literal in quotes.

Step 4 Click **OK** to dismiss the Validate window.

Step 5 Click **Update** to save the Review/Authorization Details.

Figure 4: Review Details Window



Name*	QA Service Reviewer	Subject*	#Name# - Please review
Duration*	8.0	Effort*	1.0
Assign	A person/queue	Assign to	Angela Performer
Workflow Type	Internal		
Escalation Tiers	[✓ Use all] Use only:		
Condition	1=1		
Evaluate condition when	[✓ Authorization phase starts (if condition evaluates to "false", times will be computed as zero)] Evaluate condition when task becomes active (delivery schedule will always include this task's duration)		
	☐ Re-evaluate expressions as authorizations/reviews proceed (participant assignment expressions and title will be re-evaluated)		
Notify when review starts	None		
Notify when review completes	None		
Notify when activity is cancelled	None		
Notify when task is rescheduled	None		
Notify when task is reassigned	None		
Notify when external tasks fail	None		

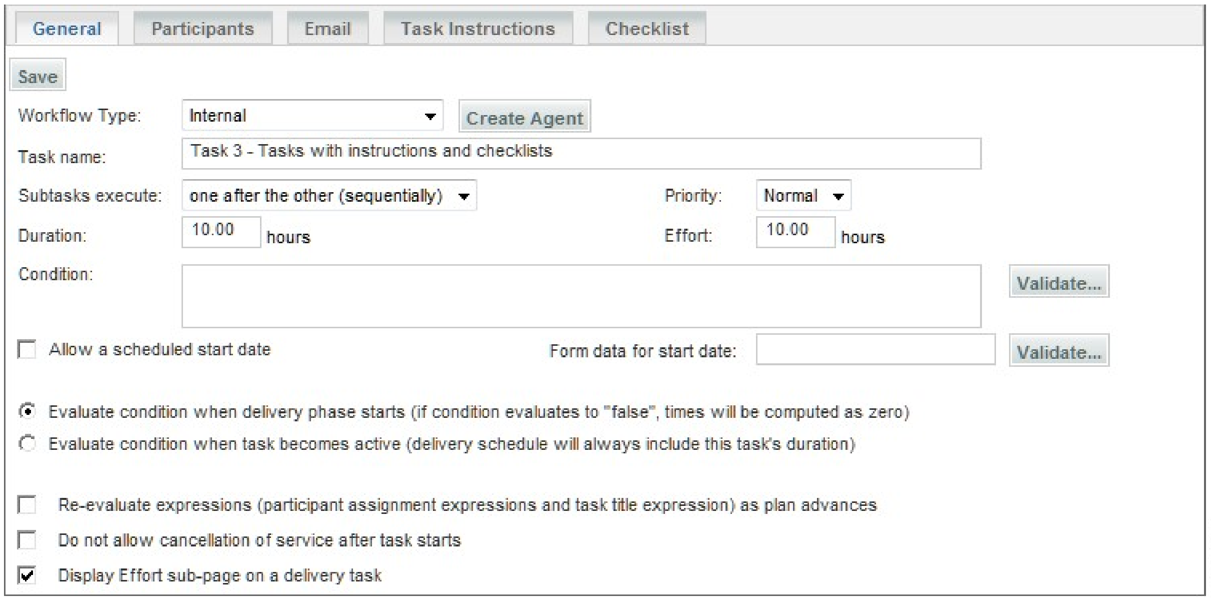
Validation checks that any namespace specified is valid for the current scope (the specific level of review/authorization or task), with the exception of dictionary fields (Data.**DictionaryName.FieldName**). This is perfectly legitimate, since the Review/Authorization may be defined at a site- or organization- level, independent of the service with which it is integrated. However, it may cause a runtime error: if the specified namespace, for example, Data.EUIT_ACCESS.Access_Type, does not exist.

Validation also checks that correct relational and arithmetic operators are used.

Conditional Delivery Plan Tasks

As with authorization and review tasks, a plan task in the service delivery moment may also be conditional.

Figure 5: Task Definition General Tab



Workflow Type:	Internal
Task name:	Task 3 - Tasks with instructions and checklists
Subtasks execute:	one after the other (sequentially)
Priority:	Normal
Duration:	10.00 hours
Effort:	10.00 hours
Condition:	
Form data for start date:	

- [ ] Allow a scheduled start date
- [✓] Evaluate condition when delivery phase starts (if condition evaluates to "false", times will be computed as zero)
- [ ] Evaluate condition when task becomes active (delivery schedule will always include this task's duration)
- [ ] Re-evaluate expressions (participant assignment expressions and task title expression) as plan advances
- [ ] Do not allow cancellation of service after task starts
- [✓] Display Effort sub-page on a delivery task

Evaluating the Condition

Once the condition has been entered, you must decide when that statement will be evaluated. Each task (approval, review, or delivery) with a condition can be evaluated either

- At the beginning of a phase (Authorization or Delivery moment), or
- When the activity becomes active.

Evaluate condition when authorization/delivery phase starts

The designer can choose to evaluate a conditional statement when a **phase** starts by choosing the "*Evaluate condition when delivery phase starts (if condition evaluates to "false", times will be computed as zero)* " option within a specific task, as shown above for delivery tasks. A " phase" corresponds to any of the system moments defined for processing a requisition. Each authorization or review has its own moment; all delivery tasks are performed within the Service Delivery moment.

Authorization tasks are always serial. You could put one conditional on one task saying if field= "somevalue" and a conditional on another saying field<> "somevalue". That way, you know one authorization task will always be executed, and if you choose "*when authorization phase starts* ", only one authorization task will appear in the process view. If you choose "when task becomes active", both tasks appear, but one would be skipped.

The "*if conditions evaluate to* "*false*"*, times will be computed as zero*" statement means that Service Catalog will evaluate the conditions at the beginning of the phase. If these conditions are false, then the corresponding tasks will not be executed, and the **Due Date** for the service will be calculated without including the duration of these tasks.

Evaluate condition when activity becomes active

Alternatively, the designer can choose to evaluate a conditional statement when the task starts by choosing the "*Evaluate condition when activity becomes active (times will not be affected, scheduling will be done by using these efforts)* " option.

Service Catalog will evaluate the condition at the beginning of each task. If the condition is false, the corresponding task will not be executed. Durations for any task configured with this option will be used to calculate the due date upon submission.

Re-evaluate Expressions as plan advances

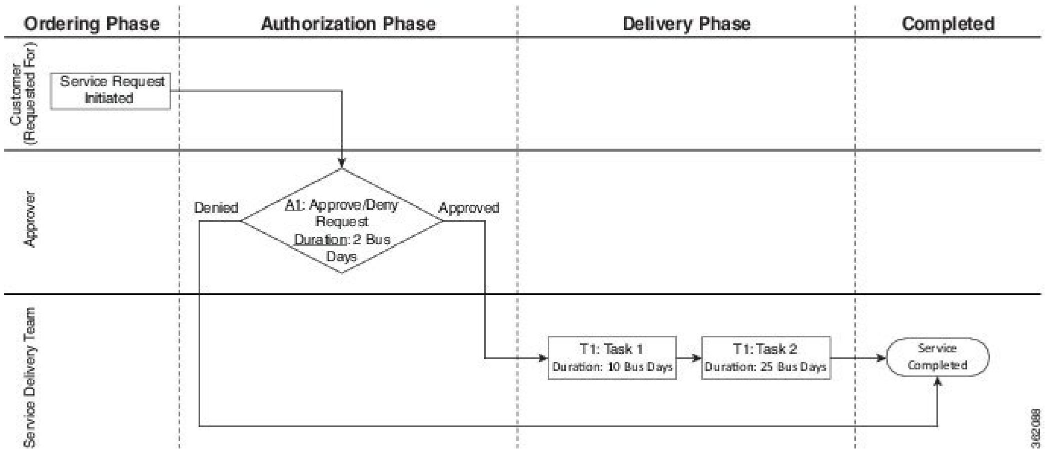
The Re-evaluate Expressions feature is useful for designs in which there are multiple authorizations. It enables the person performing an authorization task to enter information in the service form which is then used to re-evaluate the expression used to assign the performer for a *subsequent* authorization task. If this option is not checked, all information used in expressions in the authorization task must be present during the Ordering moment.

This feature allows dynamic assignment of an approval or review task to a user (person or queue) and dynamic adjustment of the task title during the authorization or review moment. Once the authorization/review becomes active, the expression will be evaluated and the task will be assigned appropriately. This capability is available only for authorization and review tasks, not for delivery tasks.

Start of a Phase and Start of a Task

The following figure pinpoints and differentiates the start of a phase and the start of a task.

Figure 6: Start of a Phase and Start of a Task



1	Start of Phase	2	Start of Task

Syntax for Conditional Statements

Conditional statements may include arithmetic operators as well as relational (logic) operators.

The type of logical operators you can use depend on the HTML representation of the field being tested. Most fields have (input elements) that allow only one value to be assigned; these include text, text area, radio button, single-select (drop-down) list, and radio buttons. The following logical operators can be used in conditional statements applied to such fields:

Table 8: Syntax for Conditional Statements

Operator	Usage
<	Less than.
>	Greater than.
<=	Less than or equal to.
>=	Greater than or equal to.
<>	Not equal to.
=	Equal to.
OR	Performs the logical OR of two or more statements. The result is true if all of the statements are true and false otherwise; that is, if any of the statements are false.
AND	Performs the logical AND of two or more statements. The result is true if any of the statements is true and false otherwise.

Operators for conditionals have the standard order of operation:

```
-  (negative)
*  (multiplication), /  (division)
+  (addition),   -  (subtraction)
<, >, >=, <=, <> (not equal), =
NOT
OR
AND
```

Parentheses can be used to change the operator precedence or clarify the condition.

EXAMPLES:

```
ActivityID >= 50
Customer.FirstName = "Ann"
Requisition.ActualCost >= 2000
Service.Quantity * Service.PricePerUnit <= 1000
Data.Approver.Custom_2 = "VP" OR Data.Approver.Custom_2 = “Director”
```

The INCLUDES operator can only be applied to fields that can hold multiple values. These include multi-select (drop-down boxes) and check boxes. The following logical operators can be used in conditional statements applied to such fields:

Operator	Usage
OR	Performs the logical OR of two or more statements. The result is true if all of the statements are true and false otherwise; that is, if any of the statements are false.

Operator	Usage
AND	Performs the logical AND of two or more statements. The result is true if any of the statements are true and false otherwise.
NOT	Negates the value of the condition.
INCLUDES	True if the specified value is a currently chosen option for the Namespace variable; applicable only to fields designated as "Multi-value" in the dictionary, typically fields with HTML representations of multi-select and check box.

EXAMPLE:

```
Data.EUIT_ACCESS.AccessType INCLUDES "DSL" AND NOT
Data.EUIT_ACCESS.AccessType INCLUDES "Dial-Up"
```

Note No "#" signs are used to enclose the Namespace.

- An alphanumeric value included in the condition must be enclosed within double quotation marks ("").
- Namespace names are not case sensitive. The recommended standard is to use Title case (capitalizing the first letter of all words).
- All alphanumeric comparisons are case sensitive. For example, the condition "Data.MoveIndividual.FirstName="Matt"" would be true only if the value of the FirstName field in the MoveIndividual dictionary were "Matt", with an initial capital letter and the rest lower case letters.
- Any Boolean Namespaces (those whose names start with "Is") have different values, depending on the underlying database. In SQLServer, the value of a Boolean is either true or false. In Oracle the corresponding values are 1 (for true) and 0 (for false).
- Less than and greater than operators for numeric value comparisons are not supported for dictionary fields. They are treated as text during comparisons. For example, the condition "Service.Data.VM.MemoryGB > 4" is evaluated to false if the dictionary field VM.MemoryGB has a value of 16.
- Multibyte characters are not supported for text string comparisons.
- An ampersand ("&") included in a literal must be encoded as "&amp;". For example, the value "two & three" would appear in an expression as "two &amp; three".
- The following operators are not supported for boolean type fields:
    - is greater than
    - is less than
    - is equal to ignore case
    - begins with

  - ends with
  - contains
  - does not contain

Tips and Techniques

Use a condition that always evaluates to false (for example, “1=2”) in a conditional statement to specify a task that will automatically be skipped.

The most common use cases for this are:

- When a service needs to “autocomplete” without having any tasks completed. The skipped task will mark the end of the delivery plan, and the requisition is marked as complete.
- When an email needs to be sent without having to complete a task or when multiple emails are needed during a given moment in the following ways:
  - Create a parent task. On the email tab, choose an email to be sent out at completion (you may also configure one to go out when the activity becomes active).
  - Create a child task with the condition 1=2. Set the condition to evaluate when the activity becomes active.

Namespace Reference

This reference section lists all Namespaces and the contexts in which each can be used.

Namespace Objects and their Relationships

The following diagram illustrates the nodes in the Namespace and the relationships between the nodes. The type of a node determines not only its own properties, but also the subcontexts (other nodes) to which it provides access.

The labeled boxes in this figure represent the types of Namespace nodes. Labeled arrows show the Namespace elements that allow the properties of one node to be accessed from another node. This figure can serve as a guide for service designers and administrators who need to navigate the Namespace to get access to Service Catalog data. For example, tracing the arrow labeled “Customer” from the Process node, we can see that the “Customer” element gives the Process node access to the properties of the Person node. So, for example, to access the Customer’s login name in a conditional statement for a task in a delivery plan, the condition would reference the namespace:

```
Requisition.Customer.LoginName
```

To access the same namespace from an email for a delivery plan, the namespace reference would be:

```
#Service.Requisition.Customer.LoginName#
```

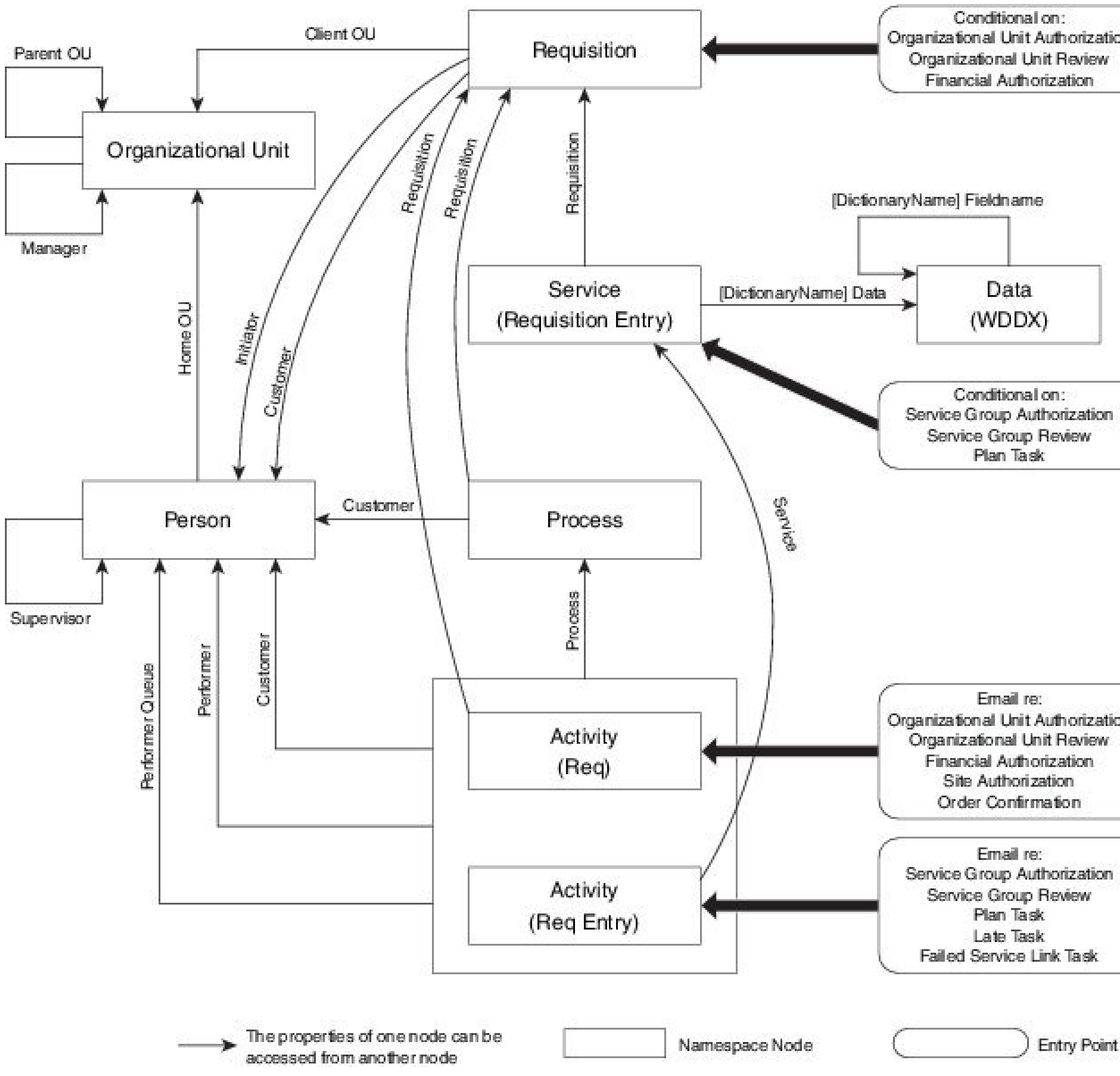
Node relationships are not bidirectional. The fact that the Process node has access to the properties of the Person node does not imply that the properties of the Process node are also available to the Person node.

From the perspective of the service designer or administrator who wishes to use Service Catalog data in formatted emails or conditional statements, there are multiple “entry points” to the paths between nodes:

- Activities associated with Requisition Entries (services), such as delivery plan tasks, Service Group Authorizations, Service group reviews, and emails generated by any of these activities.
- Activities associated with Requisitions, such as Departmental Authorizations, Departmental Reviews, and Financial Authorizations.

These entry points determine the Namespace context that applies when a service designer or administrator is configuring email to be sent from an activity, or defining a conditional execution statement. Think of them as the starting point for navigating the paths between nodes. Where you enter the Namespace determines what nodes, and ultimately what properties and data values, will be available to you from that particular activity.

Figure 7: Namespace Objects and their Relationships



Email Namespace Elements

The # character must be used to delimit Namespace elements in email notifications.

OU-based Authorizations and Reviews

The authorizations and reviews at an organizational level may trigger an email:

- Organizational Unit Authorization
- Organizational Unit Review
- Financial Authorization

These email entry points support the Namespace elements listed below:

```
#ActivityID#
#Priority# - priority assigned to the task: 1=high, 2=normal, 3=low
#DueOn#
#Subject# - the name of the task
#Waiting#
#ScheduledStart#
#StartedOn#
#CompletedOn#
#ExpectedDuration# - typical task duration, in hours
#ExpectedDurationUnits# - units in which task duration is displayed
#ActualDuration# - actual task duration, in hours
#CurrentDate#
#StateName# - the status of the task
#URL#
#Customer.*# - where * denotes any Customer element
#Performer.*# - where * denotes any Performer element
#PerformerQueue.*# where * denotes any PerformerQueue element
#Process.*# -- where * denotes any Process (task) element
#Requisition.*# - where * denotes any Requisition element
```

Tasks and Service Group Authorizations/Reviews

Tasks which are part of the delivery plan, as well as service group authorizations and reviews may trigger an email:

- Service Group Authorization
- Service Group Review
- Plan Task
- Late Task
- Ad-hoc Task
- Failed Service Link Task

These email entry points support the Namespace elements listed below:

```
#ActivityID#
#Priority#
#DueOn#
#Subject#
#Waiting#
#ScheduledStart#
```

```
#StartedOn#
#CompletedOn#
#ExpectedDuration#
#ActualDuration#
#Instructions# -- for Adhoc tasks only
#CurrentDate# -- the current date and time in GMT
#StateName# -- the current status (state) of the activity
#URL# -- the URL for directly accessing Task Details for this activity
#Customer.*# -- all Customer elements; for an adhoc task only, this   refers to the service
 performer who created the task
#Performer.*# -- all Performer elements
#PerformerQueue.*# -- all Queue elements
#Process.*# - all Process elements
#Service.ServiceID#
#Service.ProcessTrackingID#
#Service.Quantity#
#Service.PricePerUnit#
#Service.HasPrice# -- false (0) if the service has no price, 1 (true)otherwise
#Service.Bundled# -- true (1)if the service is a child service in a   bundle, false (0)
otherwise
#Service.isBundle# -- true (1) if the service is itself a bundle, false  (0) otherwise
#Service.BundledServices# -- the number of child services bundled with  the current service
#Service.ServiceDefinition.Name#
#Service.ServiceDefinition.IsEntitlement#
#Service.ServiceDefinition.DescriptionURL#
#Service.ServiceDefinition.ExpectedDuration#
#Service.ServiceDefinition.FunctionalPosition.PersonTypeElement#
#Service.ServiceDefinition.ServiceGroup.Name#
#Service.ServiceDefinition.ServiceGroup.FunctionalPosition
.PersonElement#
#Service.RequisitionEntryID#
#Service.Requisition.URL#
#Service.Requisition.ProcessTrackingID#
#Service.Requisition.ExpectedDuration#
#Service.Requisition.StartedDate#
#Service.Requisition.ActualCost#
#Service.Requisition.ExpectedCost#
#Service.Requisition.RequisitionID#
#Service.Requisition.Name#
#Service.Requisition.Customer.*# --all Customer elements
#Service.Requisition.Initiator.*# -- all Initiator element
#Service.Requisition.ClientOU.*# -- all ClientOU elements
#Service.Data.DictionaryName.FieldName#
```

The service data namespaces generally have the format

```
#Service.Data.DictionaryName.FieldName#
```

Conditional Namespace Elements

The # character is NOT used to access variables in conditionals. Some characters such as #, “, &, +, and – on the left side of the condition will cause database problems.

OU-based Authorizations and Reviews

The authorizations and reviews at an organizational level may be conditionally executed.

- Organizational Unit Authorization
- Organizational Unit Review
- Financial Authorization

These conditional entry points support the following Namespaces:

```
Site.URL
```

```
Customer.*
Initiator.*
ClientOU.*
```

Tasks and Service Group Authorizations/Reviews

Service group authorizations and reviews, as well as tasks which are part of the delivery plan allow conditions to determine whether the task/review is executed:

- Service Group Authorization
- Service Group Review
- Plan Task
- Late Task

These conditional entry points support the following Namespaces:

```
Requisition.*
Data.DictionaryName.FieldName
```

Organizational Unit-Based Namespaces

Information is available via Namespaces about organizational units. Organizational units may occur in many contexts, denoted by these namespace entity types:

Table 9: Organizational Unit-Based Namespaces

Entity Type	Description	Example
ClientOU	Organizational unit of the client for the requisition	#Requisition.HomeOU.**OrgElementType#**
HomeOU	Home organizational unit of the performer, performer queue, or customer or of the manager of any of these people	#Performer.HomeOU.**OrgElementType#**
ParentOU	The parent organizational unit of the current OU	#Customer.HomeOU.ParentOU.**OrgElementType#**

Organizational Unit Element Types (OrgElementType) are listed below, in the context of the ClientOU.

```
#ClientOU.Name#
#ClientOU.Description#
#ClientOU.OrganizationalUnitID#
#ClientOU.OrganizationalUnitTypeID#
#ClientOU.CostCenterCode#
#ClientOU.FunctionalPosition.PersonElementType#
#ClientOU.Manager.PersonTypeElement#
#ClientOU.ParentOU.Name#
#ClientOU.ParentOU.Description#
#ClientOU.ParentOU.OrganizationalUnitID#
#ClientOU.ParentOU.OrganizationalUnitTypeID#
```

```
#ClientOU.ParentOU.CostCenterCode#
#ClientOU.ParentOU.FunctionalPosition.PersonElementType#
```

Person-Based Namespaces

The Customer, Initiator, Alte rnate, and Performer Namespace elements expose information about a person stored in Organization Designer. Therefore, all entity types support the same variables; only the element denoting the entity type ("Customer", "Initiator", "Alternate", or "Person") will vary.

A list of person-based Namespace nodes is given below. Namespaces are available for all basic and extended person attributes. To form a valid Namespace variable, use these as the last part of the Namespace name, where the first part is the entity type. As always, the Namespace must be enclosed in hash marks (#) when used in an email.

PersonType is one of:	
Alternate	The designated delegate for the current authorization task performer. The following are the only namespaces supported for Alternate namespaces: #Alternate.FirstName# #Alternate.LastName# #Alternate.Title# #Alternate.TimeZoneID# #Alternate.Email#
Customer	Generally, the person for whom the current requisition was ordered; for task-based emails and escalations, the Supervisor of the task performer; for ad-hoc tasks, the task performer who created the ad-hoc task.
Customer.Supervisor	The supervisor of the customer for the current order.
Customer.HomeOU.Manager	The manager of the home organizational unit of the customer for the current order.
Initiator	The person who ordered the current requisition.
Initiator.Supervisor	The supervisor of the person who ordered the current requisition.
Initiator.HomeOU.Manager	The manager of the home organizational unit of the person who ordered the current requisition.
Performer	The person responsible for performing the current task.
Performer.Supervisor	The supervisor of the task performer.
Performer.HomeOU.Manager	The manager of the home organizational unit of the task performer.

PersonType is one of:	
ClientOU.Manager	The manager of the organizational unit for which an organizational unit review or authorization is being performed.
FunctionalPosition	A Service Catalog-defined functional position within an organization, service, or service group.
Position.**FunctionalPosition**	A user-defined functional position within an organization, service, or service group.

Elements of the Person-type namespace node are listed below. If no element is specified, the expression returns the person's unique identifier in the database.

```
#PersonType
.FirstName#
#PersonType
.LastName#
#PersonType
.Title#
#PersonType
.Birthdate#
#PersonType
.Hiredate#
#PersonType
.SSN#
#PersonType
.EmployeeCode#
#PersonType
.IsOffice#
#PersonType
.TimeZoneID#
#PersonType
.Email#
#PersonType
.CompanyAddress# -- a concatenation of all lines of the company address, including formatting
 into multiple lines
#PersonType
.PersonalAddress# -- a concatenation of all lines of the personal address, including
formatting into multiple lines
#PersonType
.SimpleCompanyAddress# -- a concatenation of all lines of the company (business) address
#PersonType
.SimplePersonalAddress# -- a concatenation of all lines of the personal address
#PersonType
.DetailedCompanyAddress.Street1#
#PersonType
.DetailedCompanyAddress.Street2#
#PersonType
.DetailedCompanyAddress.City#
#PersonType
.DetailedCompanyAddress.StateProvince#
#PersonType
.DetailedCompanyAddress.Zip#
#PersonType
.DetailedCompanyAddress.Country#
#PersonType
.DetailedPersonalAddress.Street1#
#PersonType
.DetailedPersonalAddress.Street2#
#PersonType
.DetailedPersonalAddress.City#
```

```
#PersonType
.DetailedPersonalAddress.StateProvince#
#PersonType
.DetailedPersonalAddress.Zip#
#PersonType
.DetailedPersonalAddress.Country#
#PersonType
.Location# - a concatenation of all aspects of the location, with formatting to place
elements on separate lines
#PersonType
.DetailedLocation.Building#
#PersonType
.DetailedLocation.BuildingLevel#
#PersonType
.DetailedLocation.Office#
#PersonType
.DetailedLocation.Cubicle#
#PersonType
.SimpleLocation# -- a concatenation of all aspects of the location, with spaces separating
 the elements
#PersonType
.WorkPhone#
#PersonType
.HomePhone#
#PersonType
.Fax#
#PersonType
.Mobile#
#PersonType
.Pager#
#PersonType
.LoginName#
#PersonType
.TimeZone#
#PersonType
.ExtManager#
#PersonType
.CompanyCode#
#PersonType
.Division#
#PersonType
.BusinessUnit#
#PersonType
.DepartmentNumber#
#PersonType
.CostCenter#
#PersonType
.ManagementLevel#
#PersonType
.Region#
#PersonType
.EmployeeType#
#PersonType
.LocationCode#
#PersonType
.Custom1#
#PersonType
.Custom2#
#PersonType
.Custom3#
#PersonType
.Custom4#
#PersonType
.Custom5#
#PersonType
.Custom6#
#PersonType
.Custom7#
#PersonType
.Custom8#
#PersonType
```

```
.Custom9#
#PersonType
.Custom10#
```

EXAMPLES:

```
#Person.FirstName# #Person.LastName#
#Customer.Supervisor.Fax#
```

Customer Namespaces

In delivery tasks, including escalations, the customer is the Task Supervisor. For ad-hoc tasks, the customer is the task performer who initiated the ad-hoc task. For all other tasks, and for a requisition, the customer is the person for whom the service has been requested. Customer namespaces allow access to customer information, including:

- Customer (person) information as defined in the person's profile and accessible via **Organization Designer > People**
- All person/profile information defined for the customer's supervisor
- Information about the home Organizational Unit of the customer
- All person/profile information defined for the manager of the customer's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces is listed in the Person-Based Namespaces, on page 29.

Customer Namespaces.

```
#Customer.PersonTypeElement#
#Customer.Supervisor.PersonTypeElement#
#Customer.HomeOU.Name#
#Customer.HomeOU.OrganizationalUnitID#
#Customer.HomeOU.OrganizationalUnitTypeID# -- the type of unit, where 1=service team, and
2=business unit
#Customer.HomeOU.CostCenterCode#
#Customer.HomeOU.Manager.PersonTypeElement#
#Customer.HomeOU.ParentOU.Name#
#Customer.HomeOU.ParentOU.OrganizationalUnitID#
#Customer.HomeOU.ParentOU.OrganizationalUnitTypeID# -- the type of unit, where 1=service
team, and 2=business unit
#Customer.HomeOU.ParentOU.CostCenterCode#
```

Performer Namespaces

The performer is the person responsible for performing a task in the service's delivery plan. Performer namespaces are not available in context in which no task is current—these include the email for Organization Unit reviews and authorizations; and financial authorizations.

Performer namespaces allow access to performer information, including:

- Performer (person) information as defined in the person's profile and accessible via **Organization Designer > People**
- All person/profile information defined for the performer's supervisor
- Information about the home Organizational Unit of the performer
- All person/profile information defined for the manager of the performer's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces is listed in the Person-Based Namespaces, on page 29.

```
#Performer.PersonTypeElement#
#Performer.Supervisor.PersonTypeElement#
#Performer.HomeOU.Name#
#Performer.HomeOU.OrganizationalUnitID#
#Performer.HomeOU.OrganizationalUnitTypeID#
#Performer.HomeOU.CostCenterCode#
#Performer.HomeOU.Manager.PersonTypeElement#
```

PeformerQueue Namespaces

Performer Queue namespaces provide information about the queue to which a review, authorization, or task was assigned. A subset of the Person-based namespace elements and properties are meaningful—those which are exposed in the user interface for maintaining queues in Organization Designer.

```
#PerformerQueue.FirstName#
#PerformerQueue.LastName#
#PerformerQueue.TimeZoneID#
#PerformerQueue.Email#
#PerformerQueue.WorkPhone#
#PerformerQueue.HomePhone#
#PerformerQueue.Fax#
#PerformerQueue.Mobile#
#PerformerQueue.Pager#
#PerformerQueue.TimeZone#
#PerformerQueue.HomeOU.Name#
#PerformerQueue.HomeOU.OrganizationalUnitID#
#PerformerQueue.HomeOU.OrganizationalUnitTypeID#
#PerformerQueue.HomeOU.CostCenterCode#
#PerformerQueue.HomeOU.ParentOU.Name#
#PerformerQueue.HomeOU.ParentOU.OrganizationalUnitID#
#PerformerQueue.HomeOU.ParentOU.OrganizationalUnitTypeID#
#PerformerQueue.HomeOU.ParentOU.CostCenterCode#
#PerformerQueue.HomeOU.Manager.PersonTypeElement#
```

Initiator Namespaces

The initiator is the person who orders a service.

Initiator namespaces allow access to initiator information, including:

- Initiator (person) information as defined in the person's profile and accessible via **Organization Designer > People**
- All person/profile information defined for the initiator's supervisor
- Information about the home Organizational Unit of the initiator
- All person/profile information defined for the manager of the initiator's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces is listed in the Person-Based Namespaces, on page 29.

```
#Initiator.PersonTypeElement#
#Initiator.Supervisor.PersonTypeElement#
#Initiator.HomeOU.Name#
#Initiator.HomeOU.OrganizationalUnitID#
#Initiator.HomeOU.OrganizationalUnitTypeID#
```

```
#Initiator.HomeOU.CostCenterCode#
#Initiator.HomeOU.Manager.PersonTypeElement#
#Initiator.HomeOU.ParentOU.Name#
#Initiator.HomeOU.ParentOU.OrganizationalUnitID#
#Initiator.HomeOU.ParentOU.OrganizationalUnitTypeID#
#Initiator.HomeOU.ParentOU.CostCenterCode#
```

Functional Positions

Functional positions allow you to access the Person information for individuals who have been assigned to these positions. You can access this information both for the default functional positions, and for those that have been configured for a particular implementation.

Functional positions are defined through the module selection **Organization Designer > Functional Positions**. Each functional position is associated with a particular entity—this may be a service, a service group, or an organizational unit.

Once the position has been defined, you may assign a person to the position through the Positions page for the organizational unit, or on the General tab for the service or service group. Some sample usages include:

Use **Requisition.Customer.HomeOU.BudgetManager.Emai**l to access the email address of the Budget Manager of the customer's home organizational unit.

Use **Performer.HomeOU.ParentOU.Manager.FirstName** to access the first name of the Manager of the parent organizational unit of the task performer's home organizational unit.

Default functional positions may have spaces in the position name, for example, "Budget Manager". The space is omitted with the functional position is used within a namespace reference.

User-specified functional positions may not include spaces. The namespace reference must prefix the functional position with the keyword "Position" as in the following example:

```
#Service.ServiceDefinition.Position.EscalationManager.Email#
```

For all of the functional positions, you have access to all the variables defined in the Person table above. If no person variable is included, the expression returns the person's database ID.

You can access information about Home OU, Parent OU, or Client OU functional positions from any person role that you can access in either the Requisition or the Service context. Information about Service and Service Group functional positions is only available in the Service context.

When using these expressions to include dynamic data in emails and task names, you must add the pound separator ('#') at the beginning and end of the expression, for example:

```
#Customer.HomeOU.Manager.Email#
```

Namespace Variables for Bundled Services

Namespace is a term used to describe a set of valid names that address the data objects used within Service Catalog, exposing these objects to service designers. This allows designers to use these elements in these contexts:

- Within an email, to dynamically resolve the recipient, subject, or references within the email body
- To conditionally execute reviews, authorizations, or tasks in a delivery plan
- In expressions which determine the person or queue to which an authorization or delivery task is assigned
- In task names

Namespaces are hierarchical. Namespace names reflect the structure of the data which defines a service, and the data entered on the service form when that service is requested. The key to manipulating namespace variables is knowledge of the hierarchy which defines Service Catalog data and the contexts in which particular Namespace variables can be used.

Each element in the hierarchy is case-insensitive. The elements are separated by periods ("."). The last element may also be referred to as a "property". All elements before the last one are "nodes" in the hierarchy.

The system provides a number of namespace variables that you can use when working with data in bundled services.

Namespace Variable	Data Type	Purpose/Usage
Service.Bundled	Boolean	True for a requisition entry if the service is a child on a bundle.
Service.IsBundle	Boolean	True for a requisition entry if the service is itself a bundle.
Service.BundledServices	Numeric	Indicates the number of child services belonging to a parent.
Requisition.Services	Numeric	Indicates the number of services in a requisition.
ParentService.Data.DictionaryName.FieldName		Syntax for referring to a data element within a bundle.

- If the value of the site configuration parameter **ShowBundleData** is **On**, ParentService.Data.*DictionaryName.FieldName* is equivalent to: Data.*DictionaryName.FieldName* .
- If the value of **ShowBundleData** is **Off**, ParentService.*Data.DictionaryName.FieldName* is the required syntax for referring to data elements in the parent service.

Lightweight Namespaces

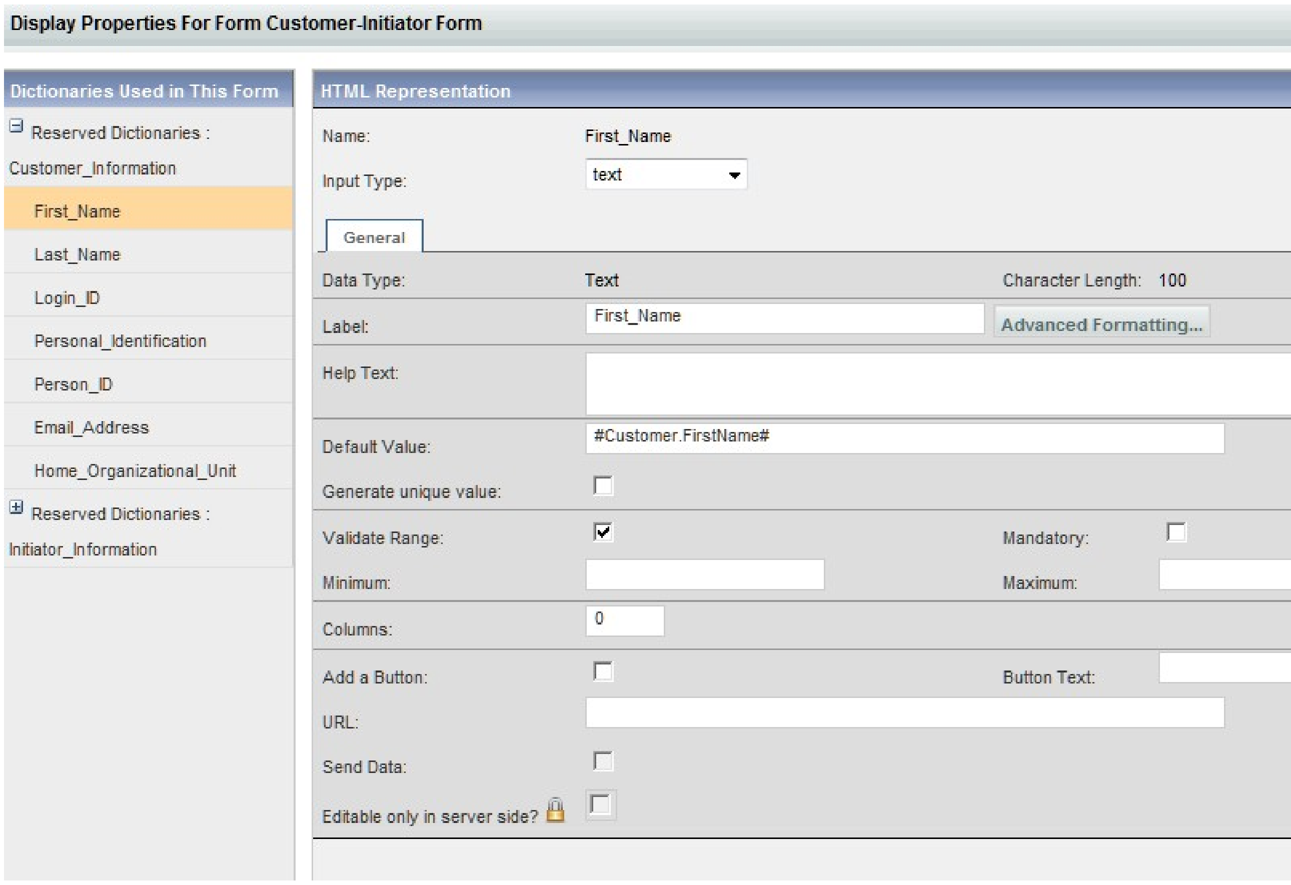
Active form rules need the equivalent of namespaces in order to dynamically access field values to be used or evaluated. For example, the service may need to display an additional dictionary or field if the user entered “Other” in a previous field; the current customer's organization may need to be used as the criteria for building a drop-down list to display valid locations for a service delivery; default values need to be provided for customer and initiator data.

Lightweight namespaces provide these capabilities. They are “lightweight” since only the information accessible to the form (not, for example, details about the service's delivery plan or task performers) can be used within the rules.

Any forms that include person-based dictionaries use lightweight namespaces to provide the values to the form fields, based on corresponding values in fields stored in the profile for the selected person. This includes both the Customer-Initiator form and any user-defined forms. Lightweight namespaces have the format #Customer.**FieldName**#, or #Initiator.**FieldName**#.

Note The use of grid dictionary fields for lightweight namespaces is not supported.

Figure 8: Grid Dictionary Display



Display Properties For Form Customer-Initiator Form

Dictionaries Used in This Form

⊟ Reserved Dictionaries : Customer_Information
- First_Name
- Last_Name
- Login_ID
- Personal_Identification
- Person_ID
- Email_Address
- Home_Organizational_Unit

⊞ Reserved Dictionaries : Initiator_Information

HTML Representation

Name: First_Name
Input Type: text

General

Data Type:	Text	Character Length:	100
Label:	First_Name	Advanced Formatting...	
Help Text:			
Default Value:	#Customer.FirstName#		
Generate unique value:	☐		
Validate Range:	✓	Mandatory:	☐
Minimum:		Maximum:	
Columns:	0		
Add a Button:	☐	Button Text:	
URL:			
Send Data:	☐		
Editable only in server side?	☐		

The namespace is automatically supplied as the Default Value for the field. If desired, the initial assignments can be replaced.

Lightweight namespaces referring to customer or initiator data can also be used as default values for fields in dictionaries that are not person-based. In this case the service designer will, of course, be responsible for mapping from the dictionary field to the appropriate person attribute. This capability allows you to define dictionaries that contain both person-based and other data.

Customer-Based Namespaces

Customer-based namespaces used for the Customer Information reserved dictionary are summarized below.

Table 10: Customer-Based Namespaces

Dictionary Field Name	Field Type	Lightweight Namespace
First_Name	Text	#Customer.FirstName#

Dictionary Field Name	Field Type	Lightweight Namespace
Last_Name	Text	#Customer.LastName#
Login_ID	Text	#Customer.LoginID#
Person_ID	Number	#Customer.PersonID#
Personal_Identification	Text	#Customer.PersonIdentification#
Email_Address	Text	#Customer.Email#
Home_Organizational_Unit	Text	#Customer.HomeOU.Name#
Title	Text	#Customer.Title#
Social_Security_Number	Text	#Customer.SSN#
Birthdate	Date	#Customer.Birthdate#
Hiredate	Date	#Customer.Hiredate#
Timezone	Text	#Customer.TimeZoneID#
Locale	Text	#Customer.LocaleID#
Supervisor	Text	#Customer.Supervisor.Name#
Employee_Code	Text	#Customer.EmployeeCode#
Supervisor_Email	Text	#Customer.Supervisor.Email#
Supervisor_ID	Number	#Customer.Supervisor.PersonID#
Supervisor_Phone	Text	#Customer.Supervisor.Phone#
Notes	Text	#Customer.Notes#
Company_Street_1	Text	#Customer.DetailedCompanyAddress.Street1#
Company_Street_2	Text	#Customer.DetailedCompanyAddress.Street2#
Company_City	Text	#Customer.DetailedCompanyAddress.City#
Company_State	Text	#Customer.DetailedCompanyAddress.StateProvince#
Company_Country	Text	#Customer.DetailedCompanyAddress.Country#
Company_Postal_Code	Text	#Customer.DetailedCompanyAddress.Zip#

Dictionary Field Name	Field Type	Lightweight Namespace
Building	Text	#Customer.DetailedLocation.Building#
Level	Text	#Customer.DetailedLocation.BuildingLevel#
Office	Text	#Customer.DetailedLocation.Office#
Cubicle	Text	#Customer.DetailedLocation.Cubicle#
Personal_Street_1	Text	#Customer.DetailedPersonalAddress.Street1#
Personal_Street2	Text	#Customer.DetailedPersonalAddress.Street2#
Personal_City	Text	#Customer.DetailedPersonalAddress.City#
Personal_State	Text	#Customer.DetailedPersonalAddress.StateProvince#
Personal_Country	Text	#Customer.DetailedPersonalAddress.Country#
Personal_Postal_Code	Text	#Customer.DetailedPersonalAddress.Zip#
Work_Phone	Text	#Customer.WorkPhone#
Home_Phone	Text	#Customer.HomePhone#
Fax	Text	#Customer.Fax#
Mobile_Phone	Text	#Customer.Mobile#
Pager	Text	#Customer.Pager#
Other	Text	#Customer.OtherPhone#
Main_Phone	Text	#Customer.MainPhone#
Primary_Phone	Text	#Customer.PrimaryPhone#
Primary_Fax	Text	#Customer.PrimaryFax#
Sales_Phone	Text	#Customer.SalesPhone#
Support_Phone	Text	#Customer.SupportPhone#
Billing_Phone	Text	#Customer.BillingPhone#
Other_Contact_Information	Text	#Customer.OtherContactInfo#
Company_Code	Text	#Customer.CompanyCode#

Dictionary Field Name	Field Type	Lightweight Namespace
Division	Text	#Customer.Division#
Business_Unit	Text	#Customer.BusinessUnit#
Department_Number	Text	#Customer.DepartmentNumber#
Cost_Center	Text	#Customer.CostCenter#
Management_Level	Text	#Customer.ManagementLevel#
Region	Text	#Customer.Region#
Employee_Type	Text	#Customer.EmployeeType#
Custom_1	Text	#Customer.Custom1#
Location_Code	Text	#Customer.LocationCode#
Custom_2	Text	#Customer.Custom2#
Custom_3	Text	#Customer.Custom3#
Custom_4	Text	#Customer.Custom4#
Custom_5	Text	#Customer.Custom5#
Custom_6	Text	#Customer.Custom6#
Custom_7	Text	#Customer.Custom7#
Custom_8	Text	#Customer.Custom8#
Custom_9	Text	#Customer.Custom9#
Custom_10	Text	#Customer.Custom10#

Initiator-Based Namespaces

Initiator-based namespaces used for the Initiator Information reserved dictionary are summarized below.

Table 11: Initiator-Based Namespaces

Dictionary Field Name	Field Type	Lightweight Namespace
First_Name	Text	#Initiator.FirstName#
Last_Name	Text	#Initiator.LastName#

Dictionary Field Name	Field Type	Lightweight Namespace
Login_ID	Text	#Initiator.LoginID#
Person_ID	Number	#Initiator.PersonID#
Email_Address	Text	#Initiator.Email#
Personal_Identification	Text	#Initiator.PersonIdentification#
Home_Organizational_Unit	Text	#Initiator.HomeOU.Name#
Title	Text	#Initiator.Title#
Social_Security_Number	Text	#Initiator.SSN#
Birthdate	Date	#Initiator.Birthdate#
Hiredate	Date	#Initiator.Hiredate#
Timezone	Text	#Initiator.TimeZoneID#
Locale	Text	#Initiator.LocaleID#
Employee_Code	Text	#Initiator.EmployeeCode#
Supervisor	Text	#Initiator.Supervisor.Name#
Supervisor_ID	Number	#Initiator.Supervisor.ID#
Supervisor_Phone	Text	#Initiator.Supervisor.Phone#
Supervisor_Email	Text	#Initiator.Supervisor.Email#
Notes	Text	#Initiator.Notes#
Company_Street_1	Text	#Initiator.DetailedCompanyAddress.Street1#
Company_Street_2	Text	#Initiator.DetailedCompanyAddress.Street2#
Company_City	Text	#Initiator.DetailedCompanyAddress.City#
Company_State	Text	#Initiator.DetailedCompanyAddress.StateProvince#
Company_Country	Text	#Initiator.DetailedCompanyAddress.Country#
Company_Postal_Code	Text	#Initiator.DetailedCompanyAddress.Zip#
Building	Text	#Initiator.DetailedLocation.Building#

Dictionary Field Name	Field Type	Lightweight Namespace
Level	Text	#Initiator.DetailedLocation.BuildingLevel#
Office	Text	#Initiator.DetailedLocation.Office#
Cubicle	Text	#Initiator.DetailedLocation.Cubicle#
Personal_Street_1	Text	#Initiator.DetailedPersonalAddress.Street1#
Personal_Street2	Text	#Initiator.DetailedPersonalAddress.Street2#
Personal_City	Text	#Initiator.DetailedPersonalAddress.City#
Personal_State	Text	#Initiator.DetailedPersonalAddress.StateProvince#
Personal_Country	Text	#Initiator.DetailedPersonalAddress.Country#
Personal_Postal_Code	Text	#Initiator.DetailedPersonalAddress.Zip#
Work_Phone	Text	#Initiator.WorkPhone#
Home_Phone	Text	#Initiator.HomePhone#
Fax	Text	#Initiator.Fax#
Mobile_Phone	Text	#Initiator.Mobile#
Pager	Text	#Initiator.Pager#
Other	Text	#Initiator.OtherPhone#
Main_Phone	Text	#Initiator.MainPhone#
Primary_Phone	Text	#Initiator.PrimaryPhone#
Primary_Fax	Text	#Initiator.PrimaryFax#
Sales_Phone	Text	#Initiator.SalesPhone#
Support_Phone	Text	#Initiator.SupportPhone#
Billing_Phone	Text	#Initiator.BillingPhone#
Other_Contact_Information	Text	#Initiator.OtherContactInfo#
Company_Code	Text	#Initiator.CompanyCode#
Division	Text	#Initiator.Division#

Dictionary Field Name	Field Type	Lightweight Namespace
Business_Unit	Text	#Initiator.BusinessUnit#
Department_Number	Text	#Initiator.DepartmentNumber#
Cost_Center	Text	#Initiator.CostCenter#
Management_Level	Text	#Initiator.ManagementLevel#
Region	Text	#Initiator.Region#
Employee_Type	Text	#Initiator.EmployeeType#
Location_Code	Text	#Initiator.LocationCode#
Custom_1	Text	#Initiator.Custom1#
Custom_2	Text	#Initiator.Custom2#
Custom_3	Text	#Initiator.Custom3#
Custom_4	Text	#Initiator.Custom4#
Custom_5	Text	#Initiator.Custom5#
Custom_6	Text	#Initiator.Custom6#
Custom_7	Text	#Initiator.Custom7#
Custom_8	Text	#Initiator.Custom8#
Custom_9	Text	#Initiator.Custom9#
Custom_10	Text	#Initiator.Custom10#

Process Namespaces

Process namespaces are available only for use in emails. They cannot be used in conditions. The Process object includes all namespaces regarding the Customer and Requisition. The Process refers to the current task.

```
#Process.Name#
#Process.Status#
#Process.StatusID#
#Process.StartedOn#
#Process.CompletedOn#
#Process.ExpectedDuration#
#Process.ActualDuration#
#Process.CostCenterID#
#Process.DueOn#
#Process.EscalationLevel#
#Process.TicketID#
```

```
#Process.TicketObjectID#
#Process.DueOnTZ#
#Process.StartedOnTZ#
#Process.DateNow# -- the current date and time in GMT
#Process.Customer.*# -- any Customer element
#Process.Requisition.*# -- any Requisition element
```

Requisition Namespaces

```
#Requisition.URL#
#Requisition.ProcessTrackingID#
#Requisition.ExpectedDuration#
#Requisition.StartedDate#
#Requisition.ActualCost#
#Requisition.ExpectedCost#
#Requisition.RequisitionID#
#Requisition.Name#
#Requisition.Services# -- The number of requisition entries in the requisition
#Requisition.Customer.*# -- Any Customer element
#Requisition.ClientOU.*# -- Any ClientOU element
#Requisition.Initiator.*# -- Any Initiator element
```

Message Namespaces

Message namespaces are available only for use in emails generated as a result of a failed Service Link task. They cannot be used in any other emails or in any conditions. The Message elements may be helpful in diagnosing the Service Link failure, and may eliminate having to consult the log files for diagnostics.

```
##Message.ExternalContent# -- Co
```

Table 12: Field Types for Active Forms

Type	Description and Active Form Implications
Text	Default data type, supports alphanumeric data; should be used for fields to be rendered as single- and multi-line text.
Number	Data type for all numbers, including integers and whole numbers; it is critical to specify the precision in the Decimals column, as the application will validate decimal precision as well as length.
Account	Documentation only; data treated as alphanumeric.
Date	Data compatible with the database's native datetime type, but display restricted to the date; calendar widget supplied for data entry.
Boolean	Data object whose possible values are “true” and “false”, presented as “Yes” and “No”.
Phone	Documentation only; data treated as alphanumeric.
SSN	Documentation only; data treated as alphanumeric.

Type	Description and Active Form Implications
Money	Data validated to contain only a valid number; monetary symbols or commas cannot be typed; accepts numerical characters and up to 3 decimal places.
Person	Data validated against a person ID in the personnel profiles; a Person Search dialog box is available via a "Search" button automatically rendered on the service form. The Person data type is provided primarily for backward compatibility; if this capability is required, a person-based dictionary should be created.
URL	Data stored as alphanumeric; a saved value is represented both as text and as an HTML representation of the value, providing a link to the specified URL.
Date and Time	Data stored as a date and time; calendar widget supplied for data entry contains a time-selection widget.