CentraSite

Built-In Design/Change-Time Actions Reference

Version 9.6

April 2014

This document applies to CentraSite Version 9.6.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: IINM-DG-ACTIONSB-96-20140318

Table of Contents

Built-In Design/Change-Time Actions Reference

This document describes the sets of design/change-time policy actions that are installed with CentraSite.

The content is organized under the following sections:

Summary of Actions in the ARIS Category	Lists the built-in actions that belong to the ARIS category.
Summary of Actions in the Change-Time Category	Lists the built-in actions that belong to the Change-Time category.
Summary of Actions in the Collector Category	Lists the built-in actions in the Collector category.
Summary of Actions in the Design-Time Category	Lists the built-in actions that belong to the Design-Time category.
Summary of Actions in the Global Category	Lists the built-in actions that belong to the Global category.
Summary of Actions in the Handler Category	Lists the built-in actions in the Handler category.
Summary of Actions in the WS-I Category	Generally describes the type of actions that belong to the WS-I category.
Built-In Actions for Design/Change-Time Policies	Describes the individual built-in actions for design/change-time policies. (Actions are listed alphabetically.)

1 Summary of Actions in the ARIS Category

The following action templates are available in the ARIS category:

Action Template	Description
Notify ARIS Service	Notifies the ARIS APG service endpoint when: ■ A Process object in CentraSite is updated or deleted. ■ A Service object (native or virtual) in CentraSite is updated or deleted, or when a user changes the state of the service to a "completed" lifecycle state (e.g, the Productive state).

2 Summary of Actions in the Change-Time Category

The following action templates are available in the Change-Time category:

Action Template	Description
Change Activation State	Activates or deactivates a lifecycle model or a policy.
Change Deployment Status	Enables or disables the deployment of a virtual service.
Classify	Classifies an object by one or more taxonomy categories.
Delete RuntimeEvents and RuntimeMetrics	Deletes the logged events and metrics associated with a service.
Initiate Approval	Initiates an approval workflow.
Initiate Group-Dependent Approval	Initiates an approval workflow based on the group to which the requestor belongs.
Mark Pending On RuntimePolicy Change	Marks a service as pending for redeployment on activation or deactivation of the applicable run-time policy.
Processing Steps Status	Enables or disables the **Processing Steps** profile for a virtual service.
Promote Asset	Promotes an asset to a new lifecycle stage (moving from one CentraSite instance to another CentraSite instance).
Register Consumer	Registers users and/or consumer applications as consumers of the requested asset.
Set Attribute Value	Assigns a value to a specified attribute in an organization, user or asset object.
Set Consumer Permission	Gives consumers instance-level permissions on the asset for which they have been registered.
Set State	Changes the lifecycle state of a lifecycle model, policy or asset.
UnClassify	Removes specified taxonomy categories from an object.
Validate Attribute Value	Validates the value of a specified attribute in an organization, user or asset against a list of allowed values.
Validate Classification	Checks whether an object is classified by a given taxonomy or taxonomy category.

Action Template	Description
Validate Lifecycle Model Activation	Checks whether a lifecycle model is ready to be activated.
Validate Policy Activation	Checks whether a policy is ready to be activated.
Validate Policy Deactivation	Verifies that a policy is not currently in-progress (i.e., undergoing execution) so that it can be successfully deactivated.
Validate State	Validates the current state of a lifecycle model, policy or asset against a given list of states.

3 Summary of Actions in the Collector Category

The following action templates are available in the Collector category:

Important: The actions in this category are used by the predefined collector policies that are installed with CentraSite. They are not intended to be used in user-defined policies. For more information about the predefined collector policies, see the section *Working with Pre-defined Policies* in the document *Working with Design/Change-Time Policies*.

Action Template	Description
BPEL Collector	Performs the collection process on BPEL Process objects
Default Collector	Performs the collection process for types that do not have a specified collector.
Lifecycle Model Collector	Performs the collection process on Lifecycle Models.
Policy Collector	Performs the collection process on Policy objects (both design/change-time policies and run-time policies)
REST Service Collector	Performs the collection process on REST Service objects.
Schema Collector	Performs the collection process on XML Schema objects
Virtual REST Service Collector	Performs the collection process on Virtual REST Service objects.
Virtual Service Collector	Performs the collection process on Virtual Service objects.
Virtual XML Service Collector	Performs the collection process on Virtual XML Service objects.
Webservice Collector	Performs the collection process on Service objects.
WS-Policy Collector	Performs the collection process on WS-Policy objects.
XML Service Collector	Performs the collection process on XML Service objects.
IS Service Interface Collector	Performs the collection process on IS Service Interface objects.

4 Summary of Actions in the Design-Time Category

The following action templates are available in the Design-Time category:

Action Template	Description
Validate Description	Validates the description of an object against a given pattern string.
Validate Name	Validates the name of an object against a given pattern string.
Validate Namespace	Checks that the target namespace attribute in a Service or XML Schema matches one of the valid namespaces in a given list.
Validate Service Binding	Checks that a Service supports the specified bindings.
Validate WSDL Size	Checks the size of the WSDL document associated with a Service to ensure that it falls within a specified range.
webMethods REST Publish	Creates a REST service in CentraSite from the published IS service interface object.

5 Summary of Actions in the Global Category

The following action templates are available in the Global category:

Action Template	Description
Call Web Service	Submits a given SOAP message to a specified Web service.
Enforce Unique Name	Ensures that the names of the Application Server type objects that are created in CentraSite are unique.
On Consumer Registration Request Send Email to Owner	Sends an email message to an object's owner when there is a consumer registration request for the object.
Send Email Notification	Sends an email message to a specified group of users.
Set Instance and Profile Permissions	Assigns instance-level permissions to an asset and to the asset's profiles.
Set Permissions	Sets instance-level permissions on an policy.
Set Profile Permissions	Assigns instance-level permissions to an asset's profiles.
Set View Permission For Service And Service Related Object To Everyone Group	Grants View permission to all users (including guests) on a given service.
Send Email Notification to Watchers	Sends an email notification to the watchers for an asset who are specific users asked to be notified for any modifications on that particular asset.

6 Summary of Actions in the Handler Category

The following action templates are available in the Handler category:

Important: The actions in this category are used by the predefined handler policies that are installed with CentraSite. They are not intended to be used in user-defined policies. For more information about the predefined handler policies, see the section *Working with Pre-defined Policies* in the document *Working with Design/Change-Time Policies*.

Action Template	Description
Asset Type Export Handler Action	Handler that CentraSite uses to export Type objects.
Default Delete Handler	Handler that CentraSite uses to delete instances of types that do not have a their own delete handlers.
Default Export Handler Action	Handler that CentraSite uses to export instances of types that do not have their own export handlers.
Default Move Handler	Handler that CentraSite uses to move instances of types that do not have their own move handlers (move to another user and/or to another organization).
Default User Move Handler	Handler that CentraSite uses to move users to other organizations.
Organization Export Handler Action	Handler that CentraSite uses to export organizations.
Reject Handler Action	Handler that prevents instances of a type from being deleted, exported, or moved, except as part of a composite object.
Taxonomy and Category Export Handler Action	Process that CentraSite uses to export taxonomies and their categories.
Virtual Service Export Handler Action	Process that CentraSite uses to export Virtual Service objects.

7 Summary of Actions in the WS-I Category

The WS-I category contains numerous actions from Basic Profile 1.1 and SSBP 1.0 that you can use to test a Web service (of type Service or Virtual Service) for compliance with Web Service Interoperability (WS-I) standards.

For more information about the various WS-I tests, see ***http://www.ws-i.org/***.

Important: A policy that contains WS-I actions must not contain any other type of action. If you need to execute other types of actions for the same event, you must place those actions in a separate policy.

8 Built-In Actions for Design/Change-Time Policies

This chapter covers the following topics:

Call Web Service

Submits a given SOAP message to a specified Web service. You can use this action to notify external systems, via a SOAP message, of changes that occur in the registry.

If the Web service returns a response, the response message is recorded to the policy log.

If the Web service produces a SOAP fault or the service cannot be successfully performed for other reasons (e.g., a network failure occurs), the policy action fails, and thus the policy itself fails. If the policy had been executed on a "pre" operation event (e.g., PreCreate, PreDelete), the requested operation is not executed.

Event Scope

PreCreate
PostCreate
PreUpdate
PostUpdate
PreDelete
PostDelete
PreStateChange
PostStateChange
OnTrigger

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Service Endpoint`	*String* The URL of the Web service that you want to call. Supported protocols are HTTP and HTTPS. **Example** `http://myServer:53307/wsstack/myService` **Note:** If the Web service that you want to invoke is registered in CentraSite, you can use the **Browse** button to select its URL.
`HTTP Basic Auth Enabled`	*Boolean* Specifies whether the service is secured by Basic HTTP authentication. If you enable this option, you can optionally specify the user ID and password that CentraSite is to submit when it invokes the service in the following parameters. If you

	leave these parameters empty, CentraSite will submit the credentials belonging to the user who triggered this policy action.	
	`HTTP Basic Auth Username`	The user ID that you want CentraSite to submit for HTTP basic authentication (if you do not want CentraSite to submit the user ID of the user who triggered the policy).
	`HTTP Basic Auth Password`	The password associated with the user ID specified in `HTTP Basic Auth Username`.

`SOAP Request Message`	*String* The SOAP message that CentraSite is to submit to the Web service. This message can include substitution tokens, if you want to insert run-time data into it. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.
	
	`<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">`
	`<env:Body>`
	`<m:keylogger xmlns:m=" http://mycompany.example.org/key ">`
	`<serviceName>${entity.name}</serviceName>`
	`<assetType>${entity.type}</assetType>`
	`<key>${entity.attribute.Key}</key>`
	`</m:keylogger>`
	`</env:Body>`
	`</env:Envelope>`	

`SOAP Action`	*String* The SOAP action that CentraSite will set in the message. If you do not set this parameter, CentraSite will set the SOAP action to the empty string.	
`Connection Timeout (in milliseconds)`	*Number* The length of time in milliseconds that CentraSite will wait for a response from the remote machine. If the timeout limit is exceeded, the policy action fails.	
`Content Type`	*String* The value that CentraSite is to assign to the Content-Type header in the SOAP request that it submits to the service. **Example:** `application/soap+xml; charset=utf-8` If you do not specify `Content Type`, the value, `application/soap+xml`, is assigned to the SOAP request.	

Change Activation State

Activates or deactivates a lifecycle model or a policy.

Event Scope

PostStateChange
OnTrigger

Object Scope

Lifecycle Model
Policy

Input Parameters

`Change Activation State To`	*String* The activation state to which you want to set the lifecycle model or policy as follows:	
	Active	Activates the policy or lifecycle model. This action will fail if it attempts to activate: ■ A policy whose parameter are not set. ■ A lifecycle model that does not have an associated object type. ■ A lifecycle model whose associated object type is already assigned to another lifecycle model. To prevent these types of failures from occurring, you should always execute the appropriate validation action before changing the activation state of a policy or lifecycle model. See the following: **Validate Policy Activation** **Validate Lifecycle Model Activation**
	Inactive	Deactivates the policy or lifecycle model.
	The following options are used to create policies that support the automatic deactivation of an older version of a policy or lifecycle model when a newer version is activated. In a lifecycle model for policies or lifecycle models, any state during which a policy or lifecycle is active must include a transition that places the policy or lifecycle model in one of the following activation states. For example, the default lifecycle model for policies includes the Productive state. This is the only state in the model during which the policy is active. The Productive state includes a transition to the Retired state, which triggers a policy that switches the policy's activation state to "Superseded and Retired". Because the Productive state includes this transition, CentraSite is able to automatically deactivate an old version of a policy when a new version is activated. It simply locates and executes the transition that places the policy in one of the following states. In the case of policies, this transition is the one to the Retired state, which puts the policy in the "Superseded and Retired" state of activation.	
	Superseded	Deactivates the policy and switches the policy's activation state to "Superseded" to indicate that the policy has been replaced by a newer version.

	Retired	Deactivates the policy and switches the policy's activation state to "Retired" to indicate that the policy is no longer available for use.
	Superseded and Retired	Deactivates the policy and switches the policy's activation state to "Superseded and Retired" to indicate that the policy has been replaced by a new version and is no longer available for use.
	This action will fail if it attempts to deactivate a policy that is in-progress. To prevent this type of failure from occurring, you should always execute the **Validate Policy Deactivation** action before using the Change Activation State action to deactivate a policy or lifecycle model.	

Change Deployment Status

Enables or disables the deployment status of a virtual service. You use this action to specify whether the given virtual service is eligible or ineligible for deployment.

- When you enable the deployment status for a virtual service, you enable the controls on the **Deployment** profile. These controls enable authorized users to deploy, undeploy or redeploy the virtual service.

  Additionally, enabling the deployment status of a virtual service makes the virtual service eligible for automatic re-deployment when changes occur to its run-time policies.
- When you disable the deployment status for a virtual service, you disable the controls on the virtual service's **Deployment** profile (thus, preventing users from deploying, undeploying or redeploy the virtual service).

  When the deployment status for a virtual service is in the disabled state, the virtual service is not eligible for automatic re-deployment when changes occur to its run-time policies.

  Note: Disabling the deployment status of a virtual service *does not* undeploy the virtual service if it is already deployed. If the virtual service is currently deployed on a Mediator, it remains deployed there. However, administrators will not be able to undeploy or re-deploy the virtual service from CentraSite Control until its deployment status is enabled.

To enable the deployment status of a virtual service, the following conditions must be satisfied:

- There must be at least one target defined in the registry.
- The Entry Protocol and Routing steps must be configured.

Typically, you use this action in combination with the **Processing Steps Status** action, which enables and disables the **Processing Steps** profile for a virtual service. For example, when you enable the **Deployment** profile, you generally disable the **Processing Steps** profile and vice versa.

Event Scope

PostStateChange

Object Scope

Service
Virtual Service
Virtual REST Service
Virtual XML Service

Input Parameters

`Enable Deployment`	*Boolean* Specifies whether the virtual service is eligible for deployment (parameter set to "Yes") or ineligible for deployment (parameter set to "No").

Classify

Classifies the target object (i.e., the object on which the policy was triggered) by one or more taxonomy categories. You can assign the taxonomy categories to a classification attribute of the target object, or you can assign the taxonomy categories as normal classifications of the target object.

The classifications you assign using this action will appear on the asset's **Classification** tab. The classifications you assign will also appear for the selected classification attribute.

You can choose whether the classifications you specify with this action will be added to the object's existing classifications or whether they will replace the object's existing classifications. This choice is only available for multi-value classification attributes, i.e. classification attributes that can reference more than one taxonomy category. If a classification attribute is a single-value classification attribute, its existing value will be replaced by the new one.

Event Scope

PostCreate
PostStateChange
OnTrigger
OnConsumerRegistration

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Classify With Attribute`	*Object Array* This holds the parameters `Classification Attribute` and `Categories`.
`Classification Attribute`	*String (optional)* This specifies the name of the object's attribute to which the following classification categories apply. If you leave this parameter empty, the classification categories will be used as normal classifications of the target object.
`Categories`	*Taxonomy Node Array* The taxonomy nodes by which you want to classify the object.
`Overwrite`	*Boolean* If true, this specifies that you want to overwrite all existing classifications with the newly specified classifications. If false, the newly specified classifications are added to the existing classifications. **Note:** This option applies only to multi-value classification attributes. If a classification attribute is a single-value classification attribute, its existing value will be replaced by the new one, regardless of the setting of the Overwrite parameter.

Consumer WSDL Generator

Enables the **Consumer WSDL** option on the Specification profile of SOAP-based virtual services. For information about the **Consumer WSDL** option, see the topic *The Specification Profile* in the section *Viewing or Editing the Profiles of Virtualized Services* in the document *Working with Virtualized Services*.

Event Scope

PreCreate
PreUpdate
OnTrigger

Object Scope

Virtual Service

Input Parameters

None.

Default Move Handler

Performs standard actions when an object's owner or organization changes.

This action is included in the *Default Move Handler* policy that is installed with CentraSite. This is the default policy that executes when an object is moved to a new owner or organization. See the section *Changing the Ownership of an Asset* in the document *Using the Asset Catalog* for related information.

Event Scope

OnMove

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Send Notification`	*Boolean* Specifies whether a notification should be sent to the object's new owner and previous owner. If this is set to "true", a notification will be sent to the new owner and the previous owner. Also, a subscription to the object will be created automatically for the new owner and the previous owner. If the ownership changes again at a later time, the subscriptions of the old owners (i.e. users who owned the object before the new owner and the immediate previous owner) will not be automatically deleted, so the old owners will continue to receive notifications of ownership changes until they delete the subscription explicitly. If this is set to "false", no notification will be sent to the new owner or the previous owner. However, a notification will be sent to any other user who has a subscription to be notified of an ownership change for the object. The default is "true".

Delete RuntimeEvents and RuntimeMetrics

Deletes the events and metrics that have been logged for a service.

This action is included in the *Delete RuntimeEvents and RuntimeMetrics of Service* policy that is installed with CentraSite. This policy executes when a service is deleted. The policy ensures that the metrics and events associated with a service are removed from the run-time logs when a service is deleted.

Event Scope

PreDelete

Object Scope

Service
Virtual Service
REST Service
Virtual REST Service
XML Service
Virtual XML Service
CEP Event Type

Input Parameters

`Delete Runtime Events`	*Boolean* Specifies whether the events that have been logged for a service are to be deleted.
`Delete Runtime Metrics`	*Boolean* Specifies whether the runtime metrics that have been logged for a service are to be deleted.

Enforce Unique Name

Ensures that the names of objects that are created in CentraSite are unique.

This action is included in the *Enforce Unique Name* policy that is installed with CentraSite. For information about this policy, see the section *Using CentraSite with ARIS* in the document *Suite Usage Aspects*.

Event Scope

PreCreate
PreUpdate
PreStateChange
OnTrigger

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Enforce Across Organizations`	*Boolean* If this parameter is set to True, then the unique name requirement for objects is enforced in all organizations defined in CentraSite.
`Allow Different Versions`	*Boolean* If this parameter is set to True, then different versions of an object can exist in CentraSite with the same name.

Initiate Approval

Initiates an approval workflow.

When this action is executed, CentraSite initiates the approval process. CentraSite will not process any subsequent actions in the policy or execute the requested operation until the approvals specified by the Initiate Approval action are received.

For more information about creating approval policies, see the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.

Caution: When you use this action on the PreStateChange event, only certain kinds of actions can be executed *after* this action in an approval policy. Some actions, if they occur after this action, will cause the policy to fail. For information about what kind of actions can follow this approval action, see the topic *Adding an Approval Policy to CentraSite* in the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.

Note: To use the email options provided by this action, CentraSite must have a connection to an SMTP email server. For instructions on how to configure CentraSite's connection to an email server, see the section *Configuring the Email Server* in the document *Basic Operations*.

If You Migrate this Action from a Pre-8.2 Release

If you have a policy that contains this action and the policy was created prior to version 8.2, that policy will continue to exhibit the old email-notification behavior (i.e., it will continue to send the earlier version's standard email message to approvers). If you want to use the email-notification enhancements that were introduced in version 8.2, simply edit the policy and enable the email parameters in the Initiate Approval action.

Event Scope

PreStateChange
OnConsumerRegistration

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`User`	*String* The user name that will be used together with the Password parameter as authentication credentials for performing a lifecycle model state change on a service asset. The credentials are stored in the approval request and passed to the web service for completing the approval. The user specified must have the permissions required to perform the state change. This parameter is only visible to users with the CentraSite Administrator role.
`Password`	*String* The password that will be used together with the User parameter as authentication credentials. This parameter is only visible to users with the CentraSite Administrator role.
`Approval Flow Name`	*String* The name to be given to the approval workflow that this action initiates. This name serves to identify the workflow in the Approval History log and in the approver's inbox. An approval flow name can contain any combination of characters, including a space. You can also include substitution tokens in the name to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.
`Approver Group`	*String Array* The user group (or groups) that identifies the set of users who are authorized to approve the requested operation. **Note:** If the user groups specified in `Approver Group` are empty at enforcement time, the user's request is auto-approved.
`Approval is Needed From`	*String* The manner in which the approval is to be processed:

Value	Description
`AnyOne`	*Default* The request can be approved or rejected by any single user in `Approver Group`. In this mode, only one user from the set of authorized approvers is required to approve or reject the request.
`EveryOne`	The request must be approved by all users specified in `Approver Group`. (It does not matter in which order the approvals are issued.) A single rejection will cause the request to be rejected.

`Reject State`	The lifecycle state that is to be assigned to the object if the approval request is rejected. If this parameter is not specified, the object's lifecycle state does not change when a rejection occurs. The lifecycle model must define a valid transition from the state that the target object is in at the time it is submitted for approval to the state specified in `Reject State`. Otherwise, the target object's state will not be switched when a rejection occurs. For more information about using this parameter, see the topic *Switching the State of an Object when an Approval Request is Rejected* in the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.
`Send Pending`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is initially submitted for approval. If you enable this option, you

`Approval Email`	must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** If the request is auto-approved, this message is not sent. **Note:** CentraSite automatically sends the email message to the approvers in addition to the users and/or groups that you specify below.	
	`Users`	*Array of Users* Users who are to receive the email. **Note:** You can specify the recipients of the email using the `Users` parameter, the `Groups` parameter, or both.
	`Groups`	*Array of Groups* Groups whose users are to receive the email. **Note:** CentraSite will only send the email to those users in the group whose CentraSite user account includes an email address.
	`Subject`	*String* The text that you want to appear in the subject line of the email. This text can include substitution tokens to insert run-time data into the subject line. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.
	`Use Email Template`	*Email Template* Specifies the template that is to be used to generate the body of the email message. For more information about using email templates, see the topic *Using Email Templates with Policy Actions* in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*. **Note:** You can use the predefined template, PendingNotification.html, for pending-approval notifications if you do not want to create an email template of your own. **Note:** If you use an email template to generate the body of the message, you cannot specify the body of the message using the `Custom Message` parameter. (In other words, you specify the body of the message using either the `Use Email Template` *or* the `Custom Message` parameter.)
	`Custom Message`	*TextArea* The text of the email message. This text can include substitution tokens to insert run-time data into the message. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.

		Note: If you use the `Custom Message` parameter to specify the body of the email message, you cannot generate the body of the message using an email template. (In other words, you specify the body of the message using either the `Custom Message` *or* the `Use Email Template` parameter.)
	`Format`	*String* Specifies whether the message in the `Custom Message` parameter is formatted as HTML or plain text. For more information about using this option, see the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*.
	`Include owner in notification`	*Boolean* When the parameter is enabled, CentraSite sends the email to the owner of the object (on which the policy is acting) in addition to the other recipients.
`Send Approval Email`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is approved. If you enable this option, you must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** CentraSite automatically sends the email message to the user who submitted the approval request in addition to the users and/or groups that you specify below. **Note:** When the EveryOne option is specified in the `Approval is Needed From` parameter, CentraSite sends this email only after all approvers have approved the request.	
	`Users`	See description of `Users` parameter above.
	`Groups`	See description of `Groups` parameter above.
	`Subject`	See description of `Subject` parameter above.
	`Use Email Template`	See description of `Use Email Template` parameter above. **Note:** You can use the predefined template, ApprovalNotification.html, for approval notifications if you do not want to create an email template of your own.
	`Custom Message`	See description of `Custom Message` parameter above.
	`Format`	See description of `Format` parameter above.
	`Include owner in notification`	See description of `Include owner in notification` parameter above.
`Send Rejection Email`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is rejected. If you enable this option, you must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** CentraSite automatically sends the email message to the approvers (except for the approver who rejected the request) and to the user who submitted the approval request in addition to the users and/or groups that you specify below.	
	`Users`	See description of `Users` parameter above.
	`Groups`	See description of `Groups` parameter above.

	`Subject`	See description of `Subject` parameter above.
	`Use Email Template`	See description of `Use Email Template` parameter above. **Note:** You can use the predefined template, RejectApprovalNotification.html, for rejection notifications if you do not want to create an email template of your own.
	`Custom Message`	See description of `Custom Message` parameter above.
	`Format`	See description of `Format` parameter above.
	`Include owner in notification`	See description of `Include owner in notification` parameter above.

Initiate Group-Dependent Approval

Initiates an approval workflow based on the group to which the requestor belongs. If the requestor does not belong to any of the groups specified in the Triggering Groups array, approval is waived and the action is considered to be completed successfully. For more information about using group-based approvals, see the topic *Using the Initiate Group-Dependent Approval Action* in the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.

Caution: When you use this action on the PreStateChange event, only certain kinds of actions can be executed *after* this action in an approval policy. Some actions, if they occur after this action, will cause the policy to fail. For information about what kind of actions can follow this approval action, see the topic *Including Multiple Actions in an Approval Policy* in the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.

Note: To use the email options provided by this action, CentraSite must have a connection to an SMTP email server. For instructions on how to configure CentraSite's connection to an email server, see the section *Configuring the Email Server* in the document *Basic Operations*.

If You Migrate this Action from a Pre-8.2 Release

If you have a policy that contains this action and the policy was created prior to version 8.2, that policy will continue to exhibit the old email-notification behavior (i.e., it will continue to send the earlier version's standard email message to approvers). If you want to use the email-notification enhancements that were introduced in version 8.2, simply edit the policy and enable the email parameters in the Initiate Group-Dependent Approval action.

Event Scope

PreStateChange
OnConsumerRegistration

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

<table>
<tr><td>User</td><td>String The user name that will be used together with the Password parameter as authentication credentials for performing a lifecycle model state change on a service asset. The credentials are stored in the approval request and passed to the web service for completing the approval. The user specified must have the permissions required to perform the state change.

This parameter is only visible to users with the CentraSite Administrator role.</td></tr>
<tr><td>Password</td><td>String The password that will be used together with the User parameter as authentication credentials.

This parameter is only visible to users with the CentraSite Administrator role.</td></tr>
<tr><td>Approval</td><td>Object Array The list of groups whose membership will determine whether the request requires approval, and if so, to which group of approvers the request is to be routed. Each object in the Approval array must contain the following information:
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Triggering Groups</td><td>String Array The user group (or groups) that identifies the users whose requests must be approved.</td></tr>
<tr><td>Approval Flow Name</td><td>String The name to be given to the approval workflow that this action initiates. This name serves to identify the workflow when activity relating to it appears in the Approval History log or an approver's inbox.

An Approval Flow Name can contain any combination of characters, including a space.

You can also include substitution tokens in the name to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the Send Email Notification action.</td></tr>
<tr><td>Approver Group</td><td>String Array The user group (or groups) that identifies the set of users who are authorized to approve the requested operation.

Note: If the user groups specified in Approver Group are empty at enforcement time, the user's request is auto-approved.</td></tr>
<tr><td>Approval is needed from</td><td>String The manner in which the approval is to be processed as follows:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>AnyOne</td><td>Default The request can be approved or rejected by any single user in Approver Group. In this mode, only one user from the set of authorized approvers is required to approve or reject the request.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

		EveryOne	The request must be approved by all users specified in `Approver Group`. (It does not matter in which order the approvals are issued.) A single rejection will cause the request to be rejected.
`Reject State`	The lifecycle state that is to be assigned to the object if the approval request is rejected. If this parameter is not specified, the object's lifecycle state does not change when a rejection occurs. The lifecycle model must define a valid transition from the state that the target object is in at the time it is submitted for approval to the state specified in `Reject State`. Otherwise, the target object's state will not be switched when a rejection occurs. For more information about using this parameter, see the topic *Switching the State of an Object when an Approval Request is Rejected* in the section *Using Approval Policies* in the document *Working with Design/Change-Time Policies*.		

`Send Pending Approval Email`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is initially submitted for approval. If you enable this option, you must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** If the request is auto-approved, this message is not sent. **Note:** CentraSite automatically sends the email message to the approvers in addition to the users and/or groups that you specify below.	
	`Users`	*Array of Users* Users who are to receive the email. **Note:** You can specify the recipients of the email using the `Users` parameter, the `Groups` parameter, or both.
	`Groups`	*Array of Groups* Groups whose users are to receive the email. **Note:** CentraSite will only send the email to those users in the group whose CentraSite user account includes an email address.
	`Subject`	*String* The text that you want to appear in the subject line of the email. This text can include substitution tokens to insert run-time data into the subject line. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.
	`Use Email Template`	*Email Template* Specifies the template that is to be used to generate the body of the email message. For more information about using email templates, see the topic *Using Email Templates with Policy Actions* in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*.

		Note: You can use the predefined template, PendingNotification.html, for pending-approval notifications if you do not want to create an email template of your own. **Note:** If you use an email template to generate the body of the message, you cannot specify the body of the message using the `Custom Message` parameter. (In other words, you specify the body of the message using either the `Use Email Template` *or* the `Custom Message` parameter.)
	`Custom Message`	*TextArea* The text of the email message. This text can include substitution tokens to insert run-time data into the message. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**. **Note:** If you use the `Custom Message` parameter to specify the body of the email message, you cannot generate the body of the message using an email template. (In other words, you specify the body of the message using either the `Custom Message` *or* the `Use Email Template` parameter.)
	`Format`	*String* Specifies whether the message in the `Custom Message` parameter is formatted as HTML or plain text. For more information about using this option, see the topic *Using a Custom Message in an Email Notification Action* in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*.
	`Include owner in notification`	*Boolean* When the parameter is enabled, CentraSite sends the email to the owner of the object (on which the policy is acting) in addition to the other recipients.
`Send Approval Email`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is approved. If you enable this option, you must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** CentraSite automatically sends the email message to the user who submitted the approval request in addition to the users and/or groups that you specify below. **Note:** When the EveryOne option is used in the `Approval is Needed From` parameter, this message is sent only after all approvers have approved the request.	
	`Users`	See description of `Users` parameter above.
	`Groups`	See description of `Groups` parameter above.
	`Subject`	See description of `Subject` parameter above.
	`Use Email Template`	See description of `Use Email Template` parameter above. **Note:** You can use the predefined template, ApprovalNotification.html, for approval notifications if you do not want to create an email template of your own.
	`Custom Message`	See description of `Custom Message` parameter above.

	`Format`	See description of `Format` parameter above.
	`Include owner in notification`	See description of `Include owner in notification` parameter above.
`Send Rejection Email`	*Boolean* Specifies whether CentraSite is to send an email message to specified users and/or groups when the request is rejected. If you enable this option, you must set the following parameters to specify the text of the message and to whom it is to be sent. **Note:** CentraSite automatically sends the email message to the group of approvers (except for the approver who rejected the request) and to the user who submitted the approval request in addition to the users and/or groups that you specify below.	
	`Users`	See description of `Users` parameter above.
	`Groups`	See description of `Groups` parameter above.
	`Subject`	See description of `Subject` parameter above.
	`Use Email Template`	See description of `Use Email Template` parameter above. **Note:** You can use the predefined template, RejectApprovalNotification.html, for rejection notifications if you do not want to create an email template of your own.
	`Custom Message`	See description of `Custom Message` parameter above.
	`Format`	See description of `Format` parameter above.
	`Include owner in notification`	See description of `Include owner in notification` parameter above.

Mark Pending On RuntimePolicy Change

Marks the deployed virtual services or consumer applications that are within the scope of run-time policy as pending for redeployment on activation or deactivation of the policy. After the policy is activated, the virtual services and consumer applications are automatically redeployed.

This action is included in the *Mark Pending-For-Redeployment On RuntimePolicy Change* policy that is installed with CentraSite. This policy executes when a run-time policy switches to the Productive state (which activates the policy) or Suspended state (which deactivates the policy).

If you customize the lifecycle model that CentraSite provides for policies and you add additional states to the model, you must execute this action during any transition that changes the activation state of a policy.

Event Scope

PreStateChange

Object Scope

Policy

Input Parameters

None.

Notify ARIS Service

Notifies the ARIS APG Service endpoint with the SOAP request message provided in this action. The APG Service endpoint is picked up from the associated ARIS Application Server.

You can use this action in the following policies:

- Notify ARIS on Process Changes
- Notify ARIS on Service Changes
- Notify ARIS on Service Completion
- Notify ARIS on Service Deletion

For information about these policies, see the section *Using CentraSite with ARIS* in the document *Suite Usage Aspects*.

Event Scope

PostUpdate
PostDelete
PostStateChange
OnTrigger

Object Scope

Process
Service

Input Parameters

`HTTP Basic Auth Enabled`	*Boolean* Specifies whether the service is secured by Basic HTTP authentication. If you enable this option, you can optionally specify the user ID and password that CentraSite is to submit when it invokes the service in the following parameters. If you leave these parameters empty, CentraSite will submit the credentials belonging to the user who triggered this policy action.

<table>
<tr><td></td><td>HTTP Basic Auth Username</td><td>The user ID that you want CentraSite to submit for HTTP basic authentication (if you do not want CentraSite to submit the user ID of the user who triggered the policy).</td></tr>
<tr><td></td><td>HTTP Basic Auth Password</td><td>The password associated with the user ID specified in HTTP Basic Auth Username.</td></tr>
<tr><td>SOAP Request Message</td><td colspan="2">String The SOAP message that CentraSite is to submit to the ARIS service. This message can include substitution tokens, if you want to insert run-time data into it. For available tokens, see the list of Substitution Tokens shown in the Send Email Notification action.

<pre>
<soapenv:Envelope ↩
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" ↩
xmlns:web="http://www.idsscheer.com/age/webMethods/">
   <soapenv:Header/>
   <soapenv:Body>
      <web:UpdateServiceRequest>
         <dbname>${context.ARIS_DB_CONTEXT}</dbname>
         <language>${user.locale}</language>
         <serviceDetail>
            <guid>${entity.key}</guid>
            <name>${entity.name}</name>
            <url>${entity.URL}</url>
            <lifeCycleState>${entity.state}</lifeCycleState>
            <owner>${entity.owner}</owner>
            <description>${entity.description}</description>
            <organization>${entity.organization}</organization>
            <version>${entity.version}</version>
            ${entity.attribute.Operations}
         </serviceDetail>
      </web:UpdateServiceRequest>
   </soapenv:Body>
</soapenv:Envelope>
</pre></td></tr>
<tr><td>SOAP Action</td><td colspan="2">String The SOAP action that CentraSite will set in the message. If you do not set this parameter, CentraSite will set the SOAP action to an empty string.</td></tr>
<tr><td>Connection Timeout (in milliseconds)</td><td colspan="2">Number The length of time (in milliseconds) that CentraSite will wait for a response from the remote machine. If the timeout limit is exceeded, the policy action fails.</td></tr>
<tr><td>Content Type</td><td colspan="2">String The value that CentraSite is to assign to the Content-Type header in the SOAP request that it submits to the service.

Example:

application/soap+xml; charset=utf-8

If you do not specify Content Type, the value application/soap+xml is assigned to the SOAP request.</td></tr>
</table>

On Consumer Registration Request Send Email to Owner

This action template allows an email to be sent to the owner of an object if there is a consumer registration request for the object.

Event Scope

PreCreate

Object Scope

Consumer Registration Request

Input Parameters

`Custom Message`	*TextArea* The text of the email message. This text can include substitution tokens to insert run-time data into the message. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification** action. **Note:** If you use the `Custom Message` parameter to specify the body of the email message, you cannot generate the body of the message using an email template. (In other words, you specify the body of the message using either the `Custom Message` *or* the `Use Email Template` parameter.)
`Subject`	*String* The text that you want to appear in the subject line of the email. This text can include substitution tokens to insert run-time data into the subject line. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification** action.
`Format`	*String* Specifies whether the message in the `Custom Message` parameter is formatted as HTML or plain text. For more information about using this option, see the topic *Using a Custom Message in an Email Notification Action* in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*.
`Use Email Template`	*Email Template* Specifies the template that is to be used to generate the body of the email message. This text can include substitution tokens to insert run-time data into the subject line. For available tokens, see the list of Substitution Tokens shown in the **Send Email Notification** action. For more information about using email templates, see the topic *Using Email Templates with Policy Actions* in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*. **Note:** You can use the predefined template, PendingNotification.html, for pending-approval notifications if you do not want to create an email template of your own. **Note:** If you use an email template to generate the body of the message, you cannot specify the body of the message using the `Custom Message` parameter. (In other words, you specify the body of the message using either the `Use Email Template` *or* the `Custom Message` parameter.)

Processing Steps Status

Enables or disables the **Processing Steps** profile for a virtual service.

- When you enable the processing steps status for a virtual service, you enable the controls on the **Processing Steps** profile for that virtual service. These controls enable authorized users to modify the processing steps for the virtual service.
- When you disable the processing steps status for a virtual service, you disable the controls on the **Processing Steps** profile. While this profile is disabled, users cannot make changes to the virtual service's processing steps.

Typically, you use this action in combination with the **Change Deployment Status** action, which enables and disables the **Deployment** profile for a virtual service. For example, when you enable the **Processing Steps** profile for a virtual service, you generally disable the **Deployment** profile and vice versa.

Event Scope

PostStateChange

Object Scope

Service
Virtual Service
Virtual REST Service
Virtual XML Service

Input Parameters

`Enable Processing Steps`	*Boolean* Specifies whether the Processing Steps profile for a virtual service is enabled (parameter set to "Yes") or disabled (parameter set to "No").

Promote Asset

This policy action allows you to promote an asset instance to a different CentraSite stage. The action can be executed on a lifecycle pre-state change, post-state change or on an OnTrigger event. The configuration options cover the following options:

- **Specify a stage to promote to**
  This can be either the name of a lifecycle stage or the URL of the target registry.
- **Specify optional user credentials for the target stage**
  The credentials specify a user name and password of a user defined on the target registry. This user should have the required permissions to create the asset on the target registry.
- **Include referenced objects in the promotion set**
  Assets that are referenced by the asset being promoted can be included in the promotion process.
- **Keep the asset owner unchanged**
  You can specify that the owner of the asset in the source registry will also be the owner in the target registry. If this user does not exist in the target registry, the owner will be the user specified in the optional user credentials described above.

  This user should be able to create assets in the target organization, which can be any of the following, depending on the input parameters you specify:

  - The organization mentioned in the `Target Organization` parameter.
  - The organization to which the user in the target registry belongs.
  - The organization to which the triggering user or the user in the `Username` parameter belongs.
- **Replace existing registry objects in the target stage**
  If an asset already exists on the target stage, it may be replaced by the asset being promoted.
- **Specify a target organization name**
  When the asset is promoted, it will belong to the organization specified.
- **Keep the lifecycle state**
  You can specify a lifecycle state for the promoted asset on the target registry. If you do not specify a state, the promoted asset will be placed in the initial state of the lifecycle model on the target registry.

Event Scope

PreStateChange
PostStateChange
OnTrigger

Object Scope

Asset

Input Parameters

The following table lists the input parameters for the policy action.

`Target Stage`	*String* The name of the target stage to which the asset will be promoted. This assumes that you have already defined the stage, as described in the section *Creating Lifecycle Stages* in the document *Customizing Lifecycle Management*. If a value is specified for the parameter `Target Stage URL`, the value of `Target Stage` is used instead of the value of the parameter `Target Stage URL`. At least one of the parameters `Target Stage` or `Target Stage URL` must be specified, i.e. they cannot both be empty.
`Target Stage URL`	*String* The URL of the target CentraSite registry. If a value is specified for the parameter `Target Stage URL`, the value of `Target Stage` is used instead of the value of the parameter `Target Stage URL`. At least one of the parameters `Target Stage` or `Target Stage URL` must be specified, i.e. they cannot both be empty.
`Username`	*String (optional)* The user name and password are used as authentication credentials for the target stage. The assets will be created in the target by this user. If the user name and password are not supplied, the user name and password of the triggering user on the source stage will be used. If this user is not defined on the target stage, the promotion will fail.
`Password`	*String (optional)* The user name and password are used as authentication credentials for the target stage. The assets will be created in the target by this user. If the user name and password are not supplied, the user name and password of the triggering user on the source stage will be used. If this user is not defined on the target stage, the promotion will fail.
`Include Referenced Assets`	*Boolean (optional)* Specifies whether the referenced assets (referenced via associations) of the applied asset will be included for the promotion. A value of "yes" means that the references assets will also be promoted. A value of "no" means that only the specified asset will be promoted. The default value is "yes".
`Keep Owner`	*Boolean (optional)* Specifies if the current owner will also be the owner in the target registry. This can only happen if the owner also exists as a user on the target registry and has the permissions required to create assets.

	A value of "yes" means that the asset owner on the target stage will be the same owner as on the source stage. A value of "no" means that the owner will be the specified user from the User Name parameter. The default value is "no".
`Replace Existing Assets`	*Boolean (optional)* Specifies if an asset that already exists on the target stage may be replaced by the asset being promoted. A value of "yes" means that an asset on the target stage can be replaced. A value of "no" means that an existing asset on the target stage cannot be replaced. The default value is "no".
`Keep Lifecycle State`	*Boolean (optional)* Specifies if the promoted asset should keep the lifecycle state that it has on the source stage. This can only happen if the lifecycle model used on the source stage is also defined and active on the target stage. A value of "yes" means that an asset on the target stage will have the same state as on the source stage. A value of "no" means that the promoted asset will be set to a lifecycle state according to the combinations as shown in the table below. The default value is "no".
`Target Organization`	*String (optional)* Specifies the owning organization of the asset on the target stage. This can only happen if the specified organization exists on the target.

As noted in the table, some of the promotion operations are only possible if the target stage contains users, organizations and lifecycle models that are compatible with those defined on the source stage. The possible combinations are listed in the following tables.

Note: During the promotion process, CentraSite copies the metadata of an asset from the source instance to the target instance. However, if the action is to be executed during a prestatechange event, the changes to the metadata in the source instance are not reflected in the target instance. You will need to explicitly update the asset if you want that change reflected in the target instance, too.

Important: Before you activate a policy that includes the Promote Asset action, ensure that the target's specified target stage URL or target stage is active and the user credentials of target registry are valid. To check this, click the **Check Connection** button. If the connection is not active and valid, activate the target specified in `Target Stage` or `Target Stage URL` and modify the user credentials as required.

Target Organization and Target Owner

When the asset is promoted to the target registry, it will belong to a specific organization and will be owned by a specific user. The organization and owner on the target registry are not necessarily the same organization and owner as on the source registry.

The owner on the target registry can be one of the following:

- the same owner as on the source registry (called "User A" in the following description)
- the user specified in the `Username` parameter (called "User B" in the following description)
- the triggering user, i.e. the user who activates the asset promotion (called "User C" in the following description)

The organization on the target registry can be one of the following:

- the organization specified in the `Target Organization` parameter (called "Organization P" in the following description)
- the organization of the user supplied in the `Username` parameter (called "Organization Q" in the following description)
- the organization of the triggering user (called "Organization R" in the following description)

Target Owner:

This user will be the owner ...	... under these circumstances
User A	If `Keep Owner` is specified, and User A has permission to create assets in Organization P or Q or R.
User B	If User A does not meet the requirements described in the previous row, and User B is defined.
User C	If User B not meet the requirements described in the previous row.

Target Organization

This organization will be the owning organization ...	... under these circumstances
Organization P	If `Target Organization` is specified and the target owner defined in the above table has permission to create assets in this organization.
Organization Q	If Organization P does not meet the requirements described in the previous row, and User B is defined.
Organization R	If Organization Q does not meet the requirements described in the previous row.

CentraSite attempts to create the asset on the target registry using the resulting combination of target owner and target organization. If the given user does not have permission to create assets in the given organization, the promotion will fail.

Keep Lifecycle State

Keep LCM State	Availability of the same LCM in the target stage	Does target have its own LCM	Result state of the promoted asset
yes	yes	n/a	Same state as in the source.
yes	no	yes	Initial state of the LCM in the target.
yes	no	no	No state assigned.
no	n/a	yes	Initial state of the LCM in the target.
no	n/a	no	No state assigned.

Register Consumer

Registers users, groups and/or consumer applications (as specified by the requestor) as consumers of an asset. This action creates a "consumed-by" relationship between the asset and the specified consumers. Once established, this relationship is visible in the asset's **Consumers** profile and also on the asset's Impact Analysis page.

The following actions are typically used in conjunction with the Register Consumer action.

- The approval actions (**Initiate Approval** or **Initiate Group-Dependent Approval**) are generally used to obtain necessary approvals prior to executing the Register Consumer action.
- The **Set Consumer Permission** action is typically executed after the Register Consumer action to give the specified consumers access to the requested asset.

Event Scope

OnConsumerRegistration

Object Scope

Asset (any type)

Input Parameters

None.

Send Email Notification

Sends an email message to specified users and/or groups.

Note: To use this action, CentraSite must have a connection to an SMTP email server. For instructions on how to configure CentraSite's connection to an email server, see the section *Configuring the Email Server* in the document *Basic Operations*.

Note: During an iteration of the policy, if the connection to a SMTP email server fails, this policy action returns a failure code. CentraSite writes the failure message to the policy log; however performs the next action in the policy (if one exists).

Event Scope

PreCreate
PostCreate
PreUpdate
PostUpdate
PreDelete
PostDelete
PreStateChange
PostStateChange
OnConsumerRegistration
OnTrigger

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Users`	*Array of Users* Users who are to receive the email. **Note:** You can specify the recipients of the email using the `Users` parameter, the `Groups` parameter, or both.
`Groups`	*Array of Groups* Groups whose users are to receive the email. **Note:** CentraSite will only send the email to those users in the group whose CentraSite user account includes an email address.
`Subject`	*String* The text that you want to appear in the email's subject line. This text can include substitution tokens to insert run-time data into the subject line. For information about using substitution tokens, see Substitution Tokens, below.
`Use Email Template`	*Email Template* Specifies the template that is to be used to generate the body of the email message. For more information about using email templates, see the topic *Using Email*

	Templates with Policy Actions in the section *Working with EMail Notifications* in the document *Working with Design/Change-Time Policies*. **Note:** You can use the predefined template, ChangeNotification.html, as your email template if you do not want to create an email template of your own. **Note:** If you use an email template to generate the body of the message, you cannot specify the body of the message using the `Custom Message` parameter. (In other words, you specify the body of the message using either the `Use Email Template` *or* the `Custom Message` parameter.)
`Custom Message`	*TextArea* The text of the email message. This text can include substitution tokens to insert run-time data into the message. For information about using substitution tokens, see Substitution Tokens, below. **Note:** If you use the `Custom Message` parameter to specify the body of the email message, you cannot generate the body of the message using an email template. (In other words, you specify the body of the message using either the `Custom Message` *or* the `Use Email Template` parameter.)
`Format`	*String* Specifies whether the custom mail message is formatted as HTML or plain text.
`Include owner in notification`	*Boolean* When enabled, this parameter sends the email notification to the owner of the object on which the policy is acting in addition to the users specified by the `Users` and `Groups` parameters.

Substitution Tokens

The following list describes substitution tokens that you can use to incorporate data from the run-time instance of a policy into the email. For example, you can use tokens to return information about the object on which the policy is acting, identify the user who triggered the policy, and/or indicate what type of event caused the policy to fire.

Be aware that some tokens are only meaningful for certain types of objects. User objects, for example, do not have a Description attribute, so the ${entity.description} token has no meaning for a User object. If you use a substitution token that is not supported by the policy's target object, CentraSite simply replaces the substitution token with a space at enforcement time.

If the target object includes the requested attribute, but the attribute itself has no value, CentraSite also replaces the substitution token with a space in the email message. If the requested attribute contains an array of values, CentraSite inserts the values into the email as a comma-separated list.

This token...	Inserts the following information into the parameter value at execution time...
`${entity.approver}`	The name of the user who approved or rejected the approval request. **Note:** This token is only meaningful in email messages that are issued by the **Initiate Approval** or **Initiate Group-dependent Approval** actions. If it is used in a context where there is no

This token...	Inserts the following information into the parameter value at execution time...
	approver or approval request, the token is simply replaced with a space.
`${entity.approvercomments}`	The comment provided by the approver when he or she approved or rejected the approval request. **Note:** This token is only meaningful in email messages that are issued by the **Initiate Approval** or **Initiate Group-dependent Approval** actions. If it is used in a context where there is no approval request, the token is simply replaced with a space.
`${entity.attribute.attributeName}`	The value of the attribute specified in *`attributeName`*. You can use this token with all attribute types (including computed types) except Classification, File, and Relationship types. **Important:** You must specify the attribute's schema name in *`attributeName`*, not its display name. For information about an attribute's schema name, see the topic *Attribute Names* in the section *What is a Type?* in the document *Object Type Management*.
`${entity.description}`	The object's description. **Note:** Users do not have a Description attribute.
`${entity.key}`	The object's key (i.e., the UUID that uniquely identifies the object within the registry).
`${entity.name}`	The object's name (in the user's locale).
`${entity.owner}`	The name of the user who owns the object against which the policy is acting.
`${entity.type}`	The type of object against which the policy acting.
`${entity.state}`	The state of the object against which the policy is acting. If the object is an Asset, Policy or Lifecycle Model, this action inserts the object's current lifecycle state. For all other object types, this token is ignored.
`${entity.URL}`	The URL for the object on which the policy is acting. (This is the URL that opens the object in CentraSite Control.)
`${entity.version}`	The object's user-assigned version identifier.
`${event.type}`	The type of event that triggered the policy.
`${from.state}`	The state from which the object is being switched (if the policy is executing on a PreStateChange or PostStateChange event.)
`${target.state}`	The state to which the object is being switched (if the policy is executing on a PreStateChange or PostStateChange event).
`${user.locale}`	The locale of the user who triggered the policy.
`${user.name}`	The name of the user who triggered the policy.

This token...	Inserts the following information into the parameter value at execution time...
`${user.organization}`	The name on the organization to which the user who triggered the policy belongs.

Example

```
User ${entity.owner} has added the following asset to the catalog: Name:
${entity.name} Description: ${entity.description}
```

Set Attribute Value

Assigns a value to a specified attribute in an organization, user or asset.

Event Scope

Post-State Change
OnTrigger
OnConsumerRegistration
Pre-Create
Post-Create

Object Scope

Organization
User
Asset (any type)

Input Parameters

`Attribute Name`	*String/Non-String* The name of the attribute that you want to set. **Note:** `Attribute Name` must be a non-arrayed String/Non-String attribute.

Set Consumer Permission

Assigns permission settings to the users and/or groups who are identified by a consumer-registration request.

The behavior of this action with respect to specific asset profiles depends on the policy's object scope.

- If you use this action in a policy that applies to multiple asset types, you can set only the asset's top-level View/Modify/Full permissions. Consumers do not receive View or Modify permission on the individual profiles associated with the asset. You will have to assign permissions to the asset's individual profiles manually.
- If you use this action in a policy that applies to one (and only one) type of asset, you can set the asset's top-level View/Modify/Full permissions and also the View/Modify permissions on its individual profiles.

The permission settings you specify in this action will either replace or be merged with the asset's existing settings, depending on how you set the `Remove Existing Permission` parameter.

If you set `Remove Existing Permission` to true, the permission settings specified in the action *completely replace* the asset's current settings. That is, the asset's previous instance-level settings are completely cleared and the permissions specified by the action are set.

For example if an asset's initial permission settings are as follows:

```
USER A    Full
USER B    Full
```

And you specify the following permissions (with `Remove Existing Permission` set to true):

```
USER A    Full
GROUP X   Modify
```

The resulting permissions on the asset will be:

```
USER A    Full
GROUP X   Modify
```

If you set `Remove Existing Permission` to false, the permission settings specified by this action are added to the asset's current settings. So, for example, if an asset has the following permission settings:

```
USER A    Full
USER B    View
```

And you specify the following permissions (with `Remove Existing Permission` set to false):

```
USER A    Modify
USER B    Full
GROUP X   Modify
```

The resulting permissions on the asset will be:

```
USER A    Full
USER B    Full
GROUP X   Modify
```

Note: The instance-level permissions that this action assigns to a user does not affect any role-based permissions that the user might already have. For example, if user ABC has "Manage Assets" permission for an organization, and that user also happens to be a member of a group to which this action assigns instance-level permissions, user ABC's "Manage Assets" permission will override the permission settings that this action assigns to him or her.

Event Scope

OnConsumerRegistration

Object Scope

Asset (any type)

Input Parameters

`Consumer Asset Profile Permission`	*Object* The instance-level permissions that are to be assigned to the users and/or groups (specified in the consumer registration request) for the requested asset.
`Remove existing permission`	*Boolean* Specifies whether the permission settings in the `Consumer Asset Profile Permission` parameter replace the existing permission settings or whether they are combined with the existing settings. See examples given above.

Set Instance and Profile Permissions

Sets instance-level permissions on an asset. You can use this action to set top-level View/Modify/Full permissions on an entire asset and to set View/Modify permissions on individual profiles within an asset.

Note: You use this action to set permissions on assets only. To set permissions on policies, you must use the **Set Permissions** action. If you want to assign asset permissions to consumers during the consumer registration process, use the **Set Consumer Permission** action.

Be aware that the behavior of this action varies depending on the policy's object scope.

- If you use this action in a policy that applies to multiple asset types, you can only use it to set the asset's top-level View/Modify/Full permissions. Users do not receive View or Modify permission on the individual profiles associated with the asset. You have to assign permissions to the asset's individual profiles manually.

- If you use this action in a policy that applies to one (and only one) type of asset, you can use it to set the asset's top-level View/Modify/Full permissions and also the View/Modify permissions on its individual profiles.

The permission settings you specify in this action will either replace or be merged with the asset's existing settings, depending on how you set the `Remove Existing Permission` parameter.

If you set `Remove Existing Permission` to true, the permission settings specified in the action *completely replace* the asset's current settings. That is, the asset's previous instance-level settings are completely cleared and the permissions specified by the action are set.

For example if an asset's initial permission settings are as follows:

```
USER A    Full
USER B    Full
```

And you specify the following permissions (with `Remove Existing Permission` set to true):

```
USER A    Full
GROUP X   Modify
```

The resulting permissions on the asset will be:

```
USER A    Full
GROUP X   Modify
```

If you set `Remove Existing Permission` to false, the permission settings specified by this action are added to the asset's current settings. So, for example, if an asset has the following permission settings:

```
USER A    Full
USER B    View
```

And you specify the following permissions (with `Remove Existing Permission` set to false):

```
USER A    Modify
USER B    Full
GROUP X   Modify
```

The resulting permissions on the asset will be:

```
USER A    Full
USER B    Full
GROUP X   Modify
```

Note: The instance-level permissions that this action assigns to a user does not affect any role-based permissions that the user might already have. For example, if user ABC has "Manage Assets" permission for an organization, and that user also happens to be a member of a group to which this action assigns instance-level permissions, user ABC's "Manage Assets" permission will override the permission settings that this action assigns to him or her.

Event Scope

PostCreate
PreStateChange
PostStateChange
OnTrigger

Object Scope

Asset (any type)

Input Parameters

`User/Group Asset Permission`	*Object Array* An array of permission settings. Each setting in the array identifies one individual user or one group and specifies the permissions for that user or group. If you specify multiple groups in this array and a user is a member of more than one group, the user will receive the permissions of all those groups combined. For example, if you assign Modify permission to Group A and Full permissions to Group B, users that are members of both groups will get Full permission on the object.
`Remove existing permission`	*Boolean* Specifies whether the permission settings in the parameters `User/Group Asset Permission`, `Propagate permissions to dependent objects` and `Propagate profile permissions` replace the existing permission settings or whether they are combined with the existing settings. See examples given above.
`Propagate permissions to dependent objects`	*Boolean* Specifies whether the access permissions defined for the asset instance will be automatically propagated to all dependent objects. For example, a Service asset can refer to a WSDL which in turn can refer to one or more XML Schema assets, and when you set this parameter to "yes", changes in the access permissions in the Service asset will be propagated to all of these dependent assets.
`Propagate profile permissions`	*Boolean* Specifies whether the profile permissions defined for the asset instance will be automatically propagated to all dependent assets of the same type. The restriction concerning the asset type arises because different asset types can have different sets of profiles. The use of this parameter is restricted to the following asset types: ■ Service

	■ XML Schema

Set Permissions

Grants View, Modify or Full permissions to specified users (or to groups of users) for a policy.

Note: You use this action to set permissions on policy objects. To set permissions on catalog assets, you must use **Set Instance and Profile Permissions**.

Be aware that the permission settings you specify in the action will either replace or be merged with the object's existing settings, depending on how you set the `Remove Existing Permission` parameter.

If you set `Remove Existing Permission` to true, the permission settings specified in the action will completely replace the object's current settings. That is, the action will clear the object's existing permission settings and replace them with the permissions you specify.

For example if a policy's initial permission settings were as follows:

```
USER A      Full
USER B      Full
GROUP ABC   Full
```

And you were to specify the following permissions with `Remove Existing Permission` set to true:

```
USER A      Full
GROUP X     Modify
```

The resulting permissions on the asset would be:

```
USER A      Full
GROUP X     Modify
```

If you set `Remove Existing Permission` to false, the permission settings specified in the action are *added to* the object's current settings. That is, the action will merge the new permission settings with the object's existing settings. For example, if an asset had the following permission settings:

```
USER A     Full
USER B     View
GROUP ABC  View
```

And you were to specify the following permissions with `Remove Existing Permission` set to false:

```
USER A     Modify
USER B     Full
GROUP X    Modify
```

The resulting permissions on the asset will be:

```
USER A     Full
USER B     Full
GROUP X    Modify
GROUP ABC  View
```

Note: The instance-level permissions that this action assigns to a user will not affect any role-based permissions that the user might already have. For example, if user ABC has "Manage Policies" permission for an organization and that user also happens to be a member of a group to which this action assigns instance-level permissions, user ABC's "Manage Policies" permission will override the permission settings that this action assigns to him or her.

Event Scope

PostCreate
PreStateChange
PostStateChange
OnTrigger

Object Scope

This action can be enforced on the following object types.

Policy

Input Parameters

`User/Group Permission`	*Object Array* An array of permission settings. Each setting in the array identifies one individual user or one group and specifies the permissions for that user or group. If you specify multiple groups in this array and a user is a member of more than one group, the user will receive the permissions of all those groups combined. For example, if you assign Modify permission to Group A and Full permissions to Group B, users that are members of both groups will get Full permissions on the object.

`Remove existing permission`	*Boolean* Specifies whether the permission settings in the `Users and Groups` parameter replace the existing permission settings or whether they are combined with the existing settings. See examples given above.
`Propagate permissions to dependent objects`	*Boolean* Specifies whether the access permissions defined for the asset instance will be automatically propagated to all dependent objects. For example, a Service asset can refer to a WSDL which in turn can refer to one or more XML Schema assets, and when you set this parameter to "yes", changes in the access permissions in the Service asset will be propagated to all of these dependent assets.

Set Profile Permissions

This action sets an asset's profile permissions for the users/groups specified without setting the asset's instance level permissions.

The users/groups specified in the parameter should have view or modify instance level permission on the asset.

Event Scope

PostCreate
PreStateChange
PostStateChange
OnTrigger

Object Scope

Asset (any type)

Input Parameters

`User/Group Permission`	*Object Array* An array of permission settings. Each setting in the array identifies one individual user or one group, the specified profile and the view/modify permissions for that user or group for the profile.
`Remove existing permission`	*Boolean* Specifies whether the permission settings in the `User/Group Permission` parameter replace the existing permission settings or whether they are combined with the existing settings.

Set State

Initiates a lifecycle state change for a lifecycle model, policy, asset or Process object.

When you use this action, be aware that:

- The state change performed by this action will trigger PreStateChange or PostStateChange policies if such policies exist for the specified state change.
- When CentraSite executes this action at enforcement time, it attempts to change the target object to the state you have specified. If this state is not a valid transition from the object's current state, the action will fail.
- If the target object is already in the specified state at enforcement time, this action does nothing. It does not initiate a state change. It simply exits and returns a successful completion code (i.e., this condition is not considered an error).

Event Scope

PostStateChange
OnTrigger
OnConsumerRegistration

Object Scope

Lifecycle Model
Policy
Asset (any type)
Process object

Input Parameters

`Change State To`	*String* The value to which you want to set the object's state.

Set View Permission For Service And Service Related Object To Everyone Group

Grants the View permission on a given service to the Everyone group. When permission is given to Everyone, all users, including guests, are able to view the service and its related interface, operation and binding objects. This policy action enables UDDIv2 clients to access the service without providing an authtoken.

This action is included in the *UDDIv2 Inquiry Policy* policy that is installed with CentraSite. This policy executes when a service or virtual service is created. This policy is disabled by default.

Event Scope

PostCreate
OnTrigger

Object Scope

Assets

Input Parameters

None.

Send Email Notification to Watchers

Sends an email notification to the watchers for an asset who are specific users asked to be notified for any modifications on that particular asset.

Note: This action is applicable to the CentraSite Business UI.

Event Scope

PostUpdate
PostDelete
OnTrigger

Object Scope

Asset (any type)

Input Parameters

`Email Template`	*Email Template* Specifies the template that is to be used to generate the body of the email message. **Note:** You can use the predefined template, NotifyUsersOnUpdate.html, as your email template if you do not want to create an email template of your own.
`Format`	*String* Specifies whether the mail message is formatted as HTML or plain text.

UnClassify

Removes specified taxonomy categories from an object.

You can use this action to unclassify an object generally or specifically. If you want to unclassify an object by removing from it all categories for an entire taxonomy, use the Taxonomies parameter to specify the taxonomy name. If you want to unclassify an object by removing just one particular category from its classification attributes, you use the Categories parameter to specify a specific category name. Both parameters can be used in the same action.

This action is executed against all classification attributes in the target object.

If the target object is not classified by any of the taxonomies or classifiers specified in the Taxonomies or Categories parameters, the action simply exits and returns a successful completion code. This condition is not considered to be an error.

Event Scope

PostStateChange
OnTrigger
OnConsumerRegistration

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Taxonomies`	*String Array* The names of the taxonomies whose categories are to be removed from the target object.
`Categories`	*String Array* The names of specific categories that are to be removed from the target object.

Validate Attribute Value

Validates the value of a specified attribute in an organization, user or asset against a list of allowed values.

Event Scope

PreStateChange
PreDelete
OnTrigger
OnConsumerRegistration

Object Scope

Organization
User
Asset (any type)

Input Parameters

`Attribute Name`	The name of the attribute that you want this action to test. The attribute's data type can be Boolean, Date and Time, Duration, Email, IP Address, Multiline String, Number, and URL/URI. **Note:** `Attribute Name` must be a non-arrayed attribute.
`Possible Attribute Value`	*String Array* An array of regular expression String values. If the value of the attribute specified in `Attribute Name` matches any entry in `Possible Attribute Values`, the action succeeds. The regular expressions you specify in `Possible Attribute Values` must support the regular expression specification for Java. The data types of possible attribute values can be Boolean, Date and Time, Duration, Email, IP Address, Multiline String, Number, and URL/URI. You can include substitution tokens in this parameter to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.

Validate Classification

Checks whether an object is classified by a given taxonomy or taxonomy category. This action examines all classification attributes in the target object.

If you just want to check that the target object has been classified by a given taxonomy, simply specify the taxonomy in the Taxonomies parameter. Leave the Categories parameter empty. The action will succeed if the object is classified by *any* category in the taxonomy (i.e., the action succeeds if the object includes at least one Classification attribute whose value represents a category that belongs the specified taxonomy).

If you want to check that the target object has been classified by a specific category in a taxonomy, specify the exact category in the Categories parameter. Leave the Taxonomies parameter empty. The action will succeed only if the object has been classified by the exact category you specify (i.e., the object includes at least one Classification attribute whose value is set to that specific category).

If you specify multiple taxonomies and categories in the Taxonomies and Categories parameters, be aware that action will succeeds if the target object is classified according to *any* taxonomy specified in the Taxonomies parameter or *any* category specified in the Categories parameter. If you

need to verify that an object has been classified by several different taxonomies or categories, you must test for each required taxonomy or category using a separate Validate Classification action.

Event Scope

PreStateChange
PreDelete
OnTrigger
OnConsumerRegistration

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Taxonomies`	*String Array* The names of the taxonomies by which the object must be classified.
`Categories`	*String Array* The names of specific taxonomy nodes by which the target object must be classified.

Validate Description

Validates the description of an object against a given pattern string.

Event Scope

PreCreate
PreStateChange
OnTrigger

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Allowed Description Pattern`	*String* Specifies a regular expression that the description must satisfy. The regular expressions you specify in `Allowed Description Pattern` must support the regular expression specification for Java. The regular expression can include substitution tokens to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.

Validate Lifecycle Model Activation

Verifies that a lifecycle model is ready to be activated by checking that the following conditions exist for the lifecycle model:

- That the lifecycle model is associated with at least one object type.
- That the object types associated with the lifecycle model are not already assigned to an active lifecycle model in your organization. (This check ensures that, within your organization, each object type is associated with no more than one lifecycle model.)

The action will not succeed unless both conditions are satisfied.

You should include this action in any policy that is triggered by a lifecycle state change that subsequently activates the lifecycle model. Executing this action before the state change occurs ensures that the state change (and subsequent activation) will not occur unless the lifecycle model is capable of being activated.

This action is executed by the default *Validate Lifecycle Activation* policy that is installed with CentraSite. The Validate Lifecycle Activation policy executes on the PreStateChange event that occurs when a lifecycle model switches to the Productive lifecycle state. The Validate Lifecycle Activation action in this policy ensures that a lifecycle model is not switched to the Productive state (and consequently, activated) unless the model has been properly associated with one or more object types.

Event Scope

PreStateChange

Object Scope

Lifecycle Model

Input Parameters

None.

Validate Name

Validates the name of an object against a given pattern string.

Event Scope

PreCreate
PreStateChange
OnTrigger

Object Scope

This action can be enforced on any object type that the policy engine supports.

Input Parameters

`Allowed Name Pattern`	*String* Specifies a regular expression that the object name must satisfy. The regular expressions you specify in `Allowed Name Pattern` must support the regular expression specification for Java. The regular expression can include substitution tokens to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.

Validate Namespace

Checks that the targetnamespace attribute in a Web Service or XML Schema matches one of the valid namespaces in a given list.

Event Scope

PreCreate
PreStateChange
OnTrigger

Object Scope

XML Schema
CEP Event Type
Service
Virtual Service
REST Service
Virtual REST Service
XML Service

Virtual XML Service

Input Parameters

`Allowed Namespaces`	*String Array* An array of regular expressions representing the valid namespaces. For this action to succeed, the value of the targetnamespace attribute in the service WSDL or XML schema must satisfy one of the regular expressions in the array. The regular expressions you specify in `Allowed Namespaces` must support the regular expression specification for Java. The regular expression can include substitution tokens to incorporate data from the target object on which the policy is acting. For a list of the allowed tokens, see the list of Substitution Tokens shown in the **Send Email Notification action**.

Validate Policy Activation

Verifies that a policy is ready to be activated by checking that the following conditions exist for the policy:

- That all of the required parameters in the policy's action list have been set.
- That all of the actions in the action list are supported by the policy's specified scope. That is, that the policy does not contain any action whose scope includes an object type or event type that is outside the scope of the policy itself. For additional information about requirements relating to an action's scope within a policy, see the topic *Policy Scope and Action Scope* in the section *Functional Scope* in the document *Working with Design/Change-Time Policies*.
- That a policy that contains one or more WS-I actions contains *only* WS-I actions.
- That a policy that executes on a PreStateChange or PostStateChange specifies the lifecycle states that will trigger the policy.
- Whether a previous version of the policy is already active, and if so, it verifies that the policy can be switched to a state in which it is retired or superseded.

The action will not succeed unless all conditions are satisfied.

You should include this action in any policy that is triggered by a lifecycle state change that subsequently activates the policy. Executing this action before the state change occurs ensures that state change (and subsequent activation) will not occur unless the policy is capable of being activated.

This action is executed by the default *Validate Policy Activation* policy that is installed with CentraSite. The Validate Policy Activation policy executes on the PreStateChange event that occurs when a policy switches to the Productive lifecycle state. The Validate Policy Activation action in

this policy ensures that a policy is not switched to the Productive state (and consequently activated) unless the policy's action parameters have been set.

Event Scope

PreStateChange

Object Scope

Policy

Input Parameters

None.

Validate Policy Deactivation

Verifies that a policy is not currently “in-progress” (i.e., undergoing execution) and can therefore be successfully deactivated. If the policy is in-progress when this action is executed, this action will fail.

You should include this action in any policy that is triggered by a lifecycle state change that subsequently deactivates the policy. Executing this action before the state change occurs helps ensure that the stage change (and subsequent policy deactivation) will not take place if the target policy is in-progress.

Note: A policy that initiates an approval workflow is considered to be “in-progress” until the required approvals are obtained for the workflow. Therefore, if the Validate Policy Deactivation action is triggered for a policy that is associated with one or more pending approval workflows, the action will fail.

This action is executed by the default *Validate Policy Deactivation* policy that is installed with CentraSite. The Validate Policy Deactivation policy executes on the PreStateChange event that occurs when a policy switches to the Revising or Retired state. The Validate Policy Deactivation action in this policy ensures that a policy is not switched to the Revising or Retired state (and consequently, deactivated) while it is undergoing execution.

Event Scope

PreStateChange

Object Scope

Policy

Input Parameters

None.

Validate Service Binding

Checks that a Web Service supports the specified bindings.

Event Scope

PreCreate
PreStateChange
OnTrigger

Object Scope

Service
Virtual Service

Input Parameters

`Binding Types`	*String Array* An array containing the list of binding types that the Web Service must support. The action will succeed only if the Web service supports all of the bindings specified in `Binding Types`.

Validate State

Validates the current state of a lifecycle model, policy or asset against a given list of states.

Event Scope

PreDelete
OnTrigger
OnConsumerRegistration

Object Scope

Lifecycle Model
Policy
Asset (any type)

Input Parameters

`Allowed States`	*String Array* An array that specifies the states for which you want the target object checked. If the state of the object matches any entry specified in `Allowed States`, the action succeeds.

Validate WSDL Size

Checks the size of the WSDL document associated with a Web Service to ensure it falls within a specified range.

Event Scope

PreCreate
PreStateChange
OnTrigger

Object Scope

Service
Virtual Service

Input Parameters

`WSDL Size`	*Number* The size limit (expressed in the units specified by the `Size Unit` parameter, below.)
`Comparator`	*String* A relational operator that specifies how the size of the WSDL document is to be compared to the value in `WSDL Size`.
`Size Unit`	*String* The units in which `WSDL Size` is expressed. Valid values are 'KB' (for Kilobytes) or 'MB' (for Megabytes).

webMethods REST Publish

Creates a REST service from the published IS service interface object.

The action is included in the *webMethods REST Publish* policy that is installed with CentraSite. This policy automatically executes when the webMethods Designer publishes an IS Service Interface object.

Important: This IS Service Interface object should be classified under the concept called "WMAssetType -> Integration Server Asset -> TypeOfIntegrationServiceInterface -> REST Service".

Event Scope

Post-Create
Pre-Update

Object Scope

IS Service Interface

Input Parameters

None.