Rational. Change

IBM

Customization Reference Guide

IBM Rational Change

Customization Reference

Release 5.2

Before using this information, be sure to read the general information under Appendix, "Notices" on page 499.

Contents

1 Introduction

IBM® Rational® Change is a customizable Web-based change tracking tool that is integrated with IBM® Rational® Synergy.

The information in this document is essential for customizing Rational Change manually. You need this document only if you want to perform customizations that are beyond the scope of the customization capabilities provided in the Rational Change graphical user interface.

Prerequisites

Rational Change customizers are responsible for designing, implementing, and maintaining a customized installation and should have developer-level technical knowledge, including the following:

- Good knowledge of JavaScript™ and HTML
- Thorough knowledge of Rational Change
- An understanding of user needs

Locating information

In addition to the information presented in this document, the following documents and links can be helpful:

- *Admin* and *ReportBuilder* Help for Rational Change
- The Rational Change Support Web site.
- *HTML & XHTML: The Definitive Guide*, by Chuck Musciano and Bill Kennedy. Published by O'Reilly and Associates, Inc.
- *JavaScript: The Definitive Guide*, by David Flanagan. Published by O'Reilly and Associates, Inc.
- The World Wide Web Consortium Web site, at `http://www.w3.org`.

Conventions

This document uses the following conventions:

- Paths are shown in UNIX®-style syntax, such as:

  ```
  /ws2/wsconfig/pt.cfg
  ```

- A vertical ellipsis in a code listing indicates that one or more lines have been omitted.
- **When code is shown in the text, a) long lines are split over several lines, and b) blank lines and tabs are used, for visual clarity, even where doing so in a Rational Change file would result in an error.** You will see this particularly in excerpts from the `pt.cfg` file, where some lines are very long and are actually single lines but are shown, nonetheless, on more than one line.

  For example, the following `CCM_ATTRIBUTE` definition might appear in the text, even though it must be on one line in the file:

  ```
  [CCM_ATTRIBUTE]
  [NAME]product_name[/NAME]
  [TYPE]CCM_LISTBOX[/TYPE]
  [/CCM_ATTRIBUTE]
  ```

  The same is true in the following wslet call excerpt:

  ```
  window.document.pt_attachment.product_version.value =
  window.document.submit.product_version.options
  [modify_data.product_version].text;
  ```

- All true/false options can be specified in any combination of upper and lower case, even though only the `TRUE|FALSE` options are shown in examples.

The table below describes the typeface and symbol conventions used in this guide.

Typeface	Description
Italic	Used for book titles and terminology. Also designates names of roles (*developer*), states (*working*), groups (*ccm_root*), and users (*laura*).
Bold	Used for items that you can select and menu paths, also used for emphasis.
`Courier`	Used for commands, code samples, filenames, HTML, and directory paths. Represents command syntax to be entered verbatim. Signifies computer output that displays on-screen. Also used for the names of attributes (`modify_time`), functions (`remote_type`), and types (`csrc`).
`Courier Italic`	Represents values in a command string that you supply. For example, (*`drive`*`:\`*`username`*`\`*`commands`*).

Contacting IBM Rational Software Support

If the self-help resources have not provided a resolution to your problem, you can contact IBM® Rational® Software Support for assistance in resolving product issues.

> **Note**: If you are a heritage Telelogic customer, a single reference site for all support resources is located at http://www.ibm.com/software/rational/support/telelogic/

Prerequisites

To submit your problem to IBM Rational Software Support, you must have an active Passport Advantage® software maintenance agreement. Passport Advantage is the IBM comprehensive software licensing and software maintenance (product upgrades and technical support) offering. You can enroll online in Passport Advantage from http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html

- To learn more about Passport Advantage, visit the Passport Advantage FAQs at http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html.
- For further assistance, contact your IBM representative.

To submit your problem online (from the IBM Web site) to IBM Rational Software Support, you must additionally:

- Be a registered user on the IBM Rational Software Support Web site. For details about registering, go to http://www.ibm.com/software/support/.
- Be listed as an authorized caller in the service request tool.

Submitting problems

To submit your problem to IBM Rational Software Support:

1. Determine the business impact of your problem. When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem that you are reporting.

Use the following table to determine the severity level.

Severity	Description
1	The problem has a *critical* business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
2	This problem has a *significant* business impact: The program is usable, but it is severely limited.
3	The problem has *some* business impact: The program is usable, but less significant features (not critical to operations) are unavailable.
4	The problem has *minimal* business impact: The problem causes little impact on operations or a reasonable circumvention to the problem was implemented.

2. Describe your problem and gather background information, When describing a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Rational Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?

To determine the exact product name and version, use the option applicable to you:

- Start your product, and click **Help** > **About** to see the offering name and version number.
- What is your operating system and version number (including any service packs or patches)?
- Do you have logs, traces, and messages that are related to the problem symptoms?
- Can you recreate the problem? If so, what steps do you perform to recreate the problem?
- Did you make any changes to the system? For example, did you make changes to the hardware, operating system, networking software, or other system components?
- Are you currently using a workaround for the problem? If so, be prepared to describe the workaround when you report the problem.

3. Submit your problem to IBM Rational Software Support. You can submit your problem to IBM Rational Software Support in the following ways:

- **Online**: Go to the IBM Rational Software Support Web site at https://www.ibm.com/software/rational/support/ and in the Rational support task navigator, click **Open Service Request.** Select the electronic problem reporting tool, and open a Problem Management Record (PMR), describing the problem accurately in your own words.

  For more information about opening a service request, go to http://www.ibm.com/software/support/help.html

  You can also open an online service request using the IBM Support Assistant. For more information, go to http://www.ibm.com/software/support/isa/faq.html.
- **By phone**: For the phone number to call in your country or region, go to the IBM directory of worldwide contacts at http://www.ibm.com/planetwide/ and click the name of your country or geographic region.
- **Through your IBM Representative**: If you cannot access IBM Rational Software Support online or by phone, contact your IBM Representative. If necessary, your IBM Representative can open a service request for you. You can find complete contact information for each country at http://www.ibm.com/planetwide/.

2 Rational Change overview for customizers

Every company has its own approach to managing change and its own suite of products. Each product has its own characteristics such as version, hardware platform, and operating system. For this reason, Rational Change was designed as a highly customizable change tracking tool.

Although Rational Change ships with a useful set of characteristics, you might want to customize the tool to include your own products and releases, as well as other properties specific to your needs.

Before performing customizations, however, you should become familiar with how Rational Change creates dynamic HTML content in the user browser.

The following pages describe the Rational Change components and their functionality, and show the Rational Change directory structure:

- Components (page 8)
- Rational Change directories (page 12)
- Distributing customized lifecycles (page 21)

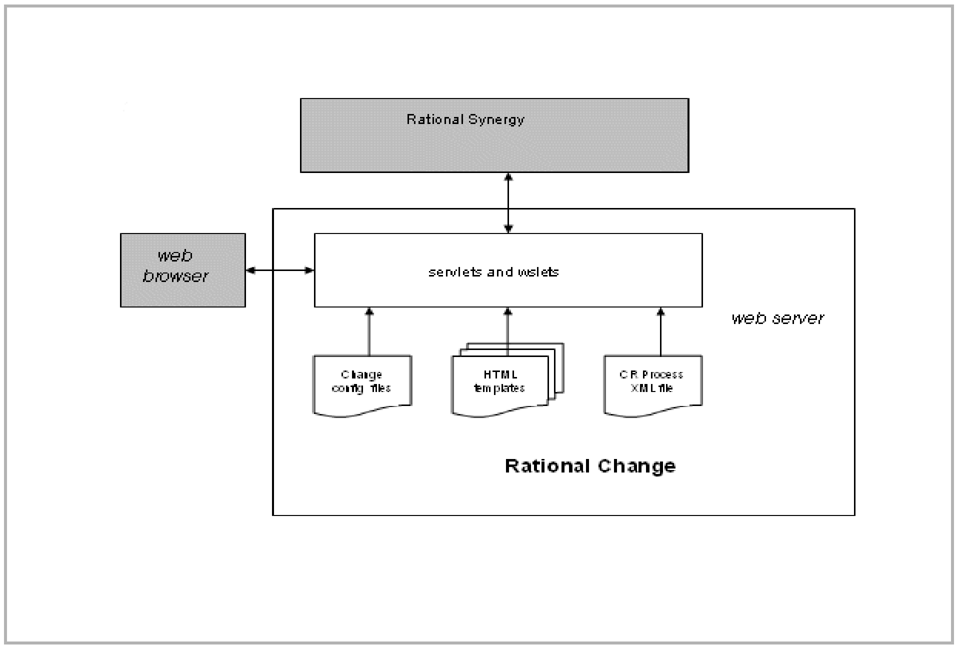
Components

Rational Change resides on a Web server and deploys its user interface using a Web browser. Data displayed in the interface is retrieved from Rational Change files and from the Rational Synergy database server. Rational Change also updates the Rational Synergy database with the values you enter and selections you make in the user interface.

You can better understand the details of these interactions by reading the following pages:

- Rational Synergy (page 9)
- PT CLI config file (page 9)
- Web server (page 9)
- HTML templates (page 10)
- Wslets (page 11)
- Servlets (page 11)
- Rational Change config files (page 11)
- CR Process file (XML) (page 11)

The diagram below shows the components of Rational Change.



Rational Synergy

Rational Synergy is the core product that stores data for your change requests and their associated objects.

Because Rational Synergy controls all change request data, all change request attributes—such as **product_name** or **request_type**—must have the same name in the Rational Synergy database as they do in the Rational Change interface.

PT CLI config file

The *PT CLI configuration (config) file* contains values that control the interactions between Rational Change and Rational Synergy.

Web server

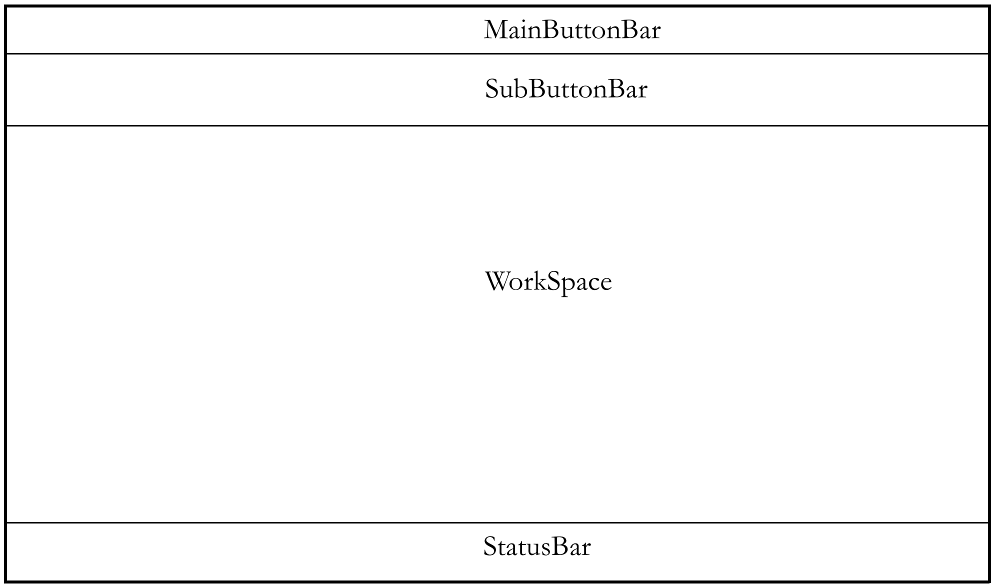
The Web server enables users to interact with the Rational Synergy database through the Web browser interface. The Web server generates the user interface and manages transactions with the Rational Synergy database.

Generating the interface means dynamically displaying the processed HTML pages (also called *forms*) needed to submit new change requests, modify existing change requests, query the database, and select reports. If the user requests information from Rational Synergy (such as by performing a query), the Web server contacts the database and retrieves the information. If the user changes information on a form (such as by submitting a change request), the Web server contacts the database and sends the new or updated information.

HTML templates

Each Rational Change form or report consists of one or more frames, and each frame is generated from an HTML template file.

For example, to generate the main Rational Change window, the templates shown in the following figure are required.



Each template file determines the appearance of the form (the colors, position of buttons, labels, and so forth), but not the dynamic content. For example, if you were to open the **Change Request Submission** dialog box template file (which fills the *WorkSpace* frame in the previous figure), the drop-down list boxes would be empty.

Servlets

Java™ servlets trigger server-side processing. The primary servlets are as follows:

- `PTweb`

  `PTweb` generates the user interface (Rational Change forms).
- `PTaction`

  `PTaction` processes commands to and from the Rational Synergy database.

Wslets

Configurable data (such as the list of valid selections for a change request attribute) and dynamic data (such as the results of a query) are retrieved from the Rational Synergy database using *wslets*: custom Web servlets that trigger Rational Synergy server-side processing.

Wslets are set up and called in HTML template files using hidden HTML. Hidden HTML is not visible in the browser and is used to set parameters for wslets and perform the wslet calls.

Rational Change config files

Rational Change configuration files control much of the functionality and appearance of Rational Change, and contain information specific to the Rational Change installation for your organization, such as product names and product versions. This information is loaded into memory when you start a Rational Change session.

CR Process file (XML)

The Change Request Process (CR Process) file is an Extensible Markup Language (XML) file that contains all the information to generate a CR Process package. The tools to create, edit and install this XML file are part of the Rational Change Administration graphical user interface (GUI).

You must have only one CR Process file installed at a time, for each Rational Change installation.

ACL files (XML)

The access-control list (ACL) files are Extensible Markup Language (XML) files that contain all the information to manage group security. The tools to create, edit, and install these XML files are part of the Rational Change Administration GUI.

You must have only one ACL file for each type (one for CRs, one for tasks, and one for objects) installed at a time for each Rational Change installation.

Rational Change directories

The following pages show the Rational Change directory structure and describes its contents:

- Directory hierarchies for Jetty installations (page 13)
- Directory hierarchies for WASCE installations (page 14)
- Directory hierarchies for WAS installations (page 15)
- Directory and file descriptions (page 16)

Note that throughout these pages, the following variables are used:

`CHANGE_HOME`

Specifies the parent Rational Change installation directory used during product installation (for example, `C:\Program Files\IBM\Rational\Change\5.2` or `/usr/local/rc52`).

`CHANGE_APP_HOME`

Specifies the Rational Change installation directory that contains the main Rational Change files (for example, using `jetty`, *`CHANGE_HOME`*`\jetty\webapps\`*`context`*`\` or `/usr/local/rc52/`*`context`*`/`).

where *`context`* is entered by the user during installation, such as

`central`

Directory hierarchies for Jetty installations

The *CHANGE_HOME* directory hierarchy is as follows:

```
CHANGE_HOME\
     jetty\
           bin\*
           cgi-bin\*
           etc\*
           ext\*
           lib\*
           logs\*
           webapps\*
               <context>\*
                   logs\
                   trapeze\
                   WEB-INF\
            win32\*
     license\
     uninstaller\
```

Directory hierarchies for WASCE installations

The *CHANGE_HOME* directory hierarchy for WASCE is as follows:

```
CHANGE_HOME\
         context.ear
                context.war
                    logs\*
                    trapeze\*
                    WEB-INF\*
        license\
        uninstaller
```

Directory hierarchies for WAS installations

The *CHANGE_HOME* directory hierarchy for WAS is as follows:

```
CHANGE_HOME\
       context
            logs\*
            trapeze\*
            WEB-INF\*
       license\
       uninstaller
```

Directory and file descriptions

The tables in this section describe the main Rational Change subdirectories and files. Some files specific to a given installation (such as Jetty) are not defined.

The following table describes the `CHANGE_HOME` subdirectories and files.

Directory	**Subdirectories and File(s)**	**Purpose\Contents**
CHANGE_APP_HOME\\`context`	`*`	Web applications running on Jetty, including Rational Change and Jetty installation subdirectores.
	`logs\*`	Rational Change application log files.
	`trapeze\*`	Rational Change subdirectories.
	`WEB-INF\*`	Web subdirectories.
CHANGE_HOME\\ `uninstaller`	`*`	Information used to uninstall Rational Change and Jetty.
CHANGE_HOME\\ `license`		License file.

The following table describes the *CHANGE_APP_HOME*\trapeze subdirectories and files.

Directory	**Subdirectories and File(s)**	**Purpose\Contents**
CHANGE_APP_HOME\trapeze	help\admin\*	HTML help files for the *Admin* role, and images and wwhgifs subdirectories for graphic files used in the help pages.
	help\reportbuilder\*	HTML help files for the *ReportBuilder* role, and images and wwhgifs subdirectories for graphic files used in the help pages.
	help\user\*	HTML help files for the *User* role, and images and wwhgifs subdirectories for graphic files used in the help pages.
	ptimages*	Images specific to Rational Change.
	wsfiles*	Generated pages the Web server can reference.

The following table describes the *CHANGE_APP_HOME*\logs files.

Directory	**Subdirectories and File(s)**	**Purpose\Contents**
CHANGE_APP_HOME\logs	audit_log.xml	This log shows changes to users, packages that have been installed/ uninstalled, group changes, and changes to ACLs.
	event.log	This log records actions executed by Rational Change. It can be accessed from **Home** page > **Logging**. in the Admin interface.
	search.log	This log is the search indexing log.

The following table describes the *CHANGE_APP_HOME*\Web subdirectories and files.

Directory	Subdirectories and File(s)	Purpose\Contents
CHANGE_APP_HOME\WEB-INF	cr_process*	CR Process files (for example, dev_proces.xml).
	cs_reports*	Rational Change report XML files.
	lib*	JAR files (XML parsing modules, and so forth).
	package_templates*	Package templates for merging with process packages.
	packages*	Available Rational Change packages.
	wsconfig*	Main Rational Change configuration directory.
CHANGE_APP_HOME\WEB-INF\ wsconfig	admin_framework.cfg	Configuration file for the *Admin* role.
	dev_process.cfg	Process-specific configuration file (named *process*.cfg.)
	init.cfg	Initialization file specifies the character encoding of configuration files.
	license_data.txt	License file
	local_admin.xml	Defines the local admin user ID and password.
	pt.cfg	Main Rational Change configuration definitions.
	ptcli.cfg	Controls interactions between Change and Synergy.
	pt_listbox.cfg	Saved Rational Change list box settings.
	rds_application_ registry.xml	Registry file.

Directory	**Subdirectories and File(s)**	**Purpose\Contents**
	`rds_config.xml`	Defines the RDS server to connect to.
	`synergy_user.xml`	Defines the Rational Synergy user used to start back-end sessions.
	`task_framework.cfg`	Configuration definitions for tasks.
	`template.cfg`	Sample Rational Change customization configuration file.
	`user_framework.cfg`	Configuration definitions for the *User* role and any other custom roles.
	`report_name``.cfg`	Configuration file for each report created using the Report Builder.
	`templates\*`	Template files.
`CHANGE_APP_HOME``\WEB-INF\wsconfig\`	`acl\`	Files defining Access Control Lists used in group security settings.
`CHANGE_APP_HOME``\WEB-INF\wsconfig\`	`scripts\*`	Scripts used by Rational Change group security customization features.
`CHANGE_APP_HOME``\WEB-INF\wsconfig\templates`	`pt\*`	Main Rational Change template directory.
`CHANGE_APP_HOME``\WEB-INF\wsconfig\templates\pt`	`etc\*`	Misc files.
	`forms\*`	Template files used to generate forms (Web pages). Includes templates common to all roles, and subdirectories for specific roles.
	`include\*`	Ready-to-use files to include using the PTInclude wslet. The root directory contains JavaScript files; subdirectories contain attribute control and custom control files, style sheets, and package-specific JavaScript files.
	`reports\*`	Template files used to generate reports.

Directory	Subdirectories and File(s)	Purpose\Contents
CHANGE_APP_HOME\WEB-INF\wsconfig\templates\pt\forms	admin_framework*	HTML templates unique to the *Admin* role.
	user_framework*	HTML templates unique to the *User* role (or custom roles).
	*package_process_name**	HTML templates for the installed CR Process (for example, dev_process). For the shipped processes, **Change Request Information**, **Change Request Submission**, and **Transition** templates are included.
CHANGE_APP_HOME\WEB-INF\wsconfig\templates\pt\include	admin_framework*	Include files unique to the *Admin* role.
	attr_controls*	Attribute "controls" (generic files containing the HTML and JavaScript to display each type of attribute).
	base_templates*	Base templates used to generate auto-generated templates.
	custom_controls*	Custom "controls" (files containing the HTML and JavaScript to display one or more types of attribute as defined by the customizer).
	styles*	HTML style sheets.
	user_framework*	Include files unique to the *User* role (or custom roles).
	*name*_process*	Include files for the installed CR Process (for example, dev_process).
	task_framework	Include files for task association.
CHANGE_APP_HOME\WEB-INF\wsconfig\templates\pt\reports	user_framework*	Basic report templates that can be used with all lifecycles.

Distributing customized lifecycles

Once you have set up and tested your customization, distribute it to the appropriate users. This can involve installing the lifecycle on another server or deploying it in another location. Use the following information to ensure all the necessary information is transferred.

If your customization includes changes to attributes, states, transitions, and/or security rules, you need to copy the CR Process XML file. This file is located at:

```
CHANGE_APP_HOME\WEB-INF\cr_process
or
/usr/local/rc52/context/WEB-INF/cr_process
```

If your customization includes custom lifecycle diagrams, listbox values, reports, and trigger scripts, copy your customized package template.

Users, roles, and privileges

Rational Change requires that a user have one or more valid login roles to start a session, and one or more Rational Synergy privileges to perform specific operations in the interface.

For example, a developer will typically be permitted to log on as *User* and create change requests (because he has the *developer* privilege), but will not be allowed to assign change requests to others (because he does not have the *assigner* privilege).

To define roles and privileges for users, use the lifecycle editor to map Rational Synergy privileges to Rational Change login roles. Then, in order for each user to have one or more Rational Change login roles and associated privileges, assign CM privileges to each user name using the **User** tab in the Admin interface.

When you save these mappings and assignments, Rational Change updates the role entries in the SYSTEM VARIABLES SECTION (page 27) section in the `pt.cfg` file.

Sending a request to a servlet

You can perform the following types of servlet requests:

- "Form load actions" on page 109
- "User actions" on page 131
- "Admin actions" on page 207
- "Miscellaneous actions" on page 227

To perform these actions, your template files must contain the following HTML content:

- JavaScript Function

  The JavaScript function sets the parameters used as inputs to the `pt_submission` form. The parameters you set depend on the type of form you are loading.
- Hidden Form

  The form request hidden form is the mechanism by which Rational Change loads the form. The form inputs depend on the type of form you are loading.

  Note You can also load forms by building a URL with a query string on the end, but that is not the recommended method because it requires you to perform your own encoding.

New/changed features in Rational Change

The following information has been added or revised in this document:

- Contacting IBM Rational Software Support (page 4)
- Rational Change directories (page 12)
- [ALL_USERS_ABS_LIMIT] (page 37)
- [BROWSE_RELATION_FILTER] (page 38)
- [BROWSE_VIEW_DISPLAY_STATUS_ATTRS] (page 38)
- CCM_LOGIN_ID (page 345)
- [CENTRAL_CR_DATABASE] (page 42)
- [CR_ACL_CLASSNAME] (page 44)
- FROM_CENTRAL (page 338)
- GetMirroredAttributes (page 248)
- [LDAP_GROUP_FILTER] (page 49)
- [LDAP_GROUP_OBJECT_CLASS] (page 49)
- Removed LDAP parameters:
  [LDAP_BUNDLED]
  [LDAP_UNIQUE_ID]
  [LDAP_JNDI_CONTEXT]
  [LDAP_URL]
  [LDAP_SECURITY]
  [LDAP_SEARCHBASE]
  [LDAP_GROUPSEARCHBASE]
  [LDAP_MODBASE]
  [LDAP_MANAGERDN]
  [LDAP_MANAGERPW]
- log file locations moved `CHANGE_APP_HOME`\logs (page 17)
- MESSAGE_OF_THE_DAY (page 357)
- [MIRRORED_ATTRIBUTE] (page 52)
- [OBJECT_ACL_CLASSNAME] (page 53)
- OFFLINE_DATABASES (page 363)
- Package hierarchy (page 228)
- REPORT_ITEM_LOCATION_DATABASE (page 407)

- RunBsfScript (page 463)
- [SEARCH_INDEX_DIRECTORY] (page 58)
- [SEARCH_MAX_RESULTS] (page 59)
- Setting fully qualified domain names (page 108)
- SERVER_TYPE (page 358)
- set_user_data (page 218)
- [SHOW_MOTD] (page 60)
- [TASK_ACL_CLASSNAME] (page 62)
- USER_ON_CENTRAL_DATABASE (page 252)
- WsletTimer BEGIN/END (page 464)
- WsletUpdate TDS_DOWN (page 466)

3 Configuration files

The configuration files are `pt.cfg`, the role-based configuration files (`admin_framework.cfg`, `user_framework.cfg`, and `task_framework.cfg`), and custom configuration files.

With the exception of most system variables, you can use all the tags described in this chapter in role-based and custom configuration files, as well as in the `pt.cfg` file.

Configuration files can consist of the sections shown in the following table. The section order is typical of the order in the `pt.cfg` file.

Section name	Description
SYSTEM VARIABLES SECTION	Defines variables that affect how the entire Rational Change system functions.
USER PROFILE SECTION	Defines data about individual users.
ATTRIBUTE NAME/DATATYPE SECTION	Defines all the change request attributes (product, version, hardware, and so forth) used by Rational Change.
OBJECT REPORT SECTION	Defines reports that show information about objects.
TASK REPORT SECTION	Defines reports that show information about tasks.
PROBLEM REPORT SECTION	Defines reports that show information about change requests.
QUERY DEFINITION SECTION	Defines named queries available in Rational Change.
RELATIONSHIP REPORT DEFINITION SECTION	Defines the named reports and report formats available in Rational Change.
VALUELISTBOX SECTION	Defines value list boxes used by Rational Change.

Section name	Description
LISTBOX SECTION	Defines list boxes used by Rational Change.
LISTS SECTION	Defines named lists for use in data list box controls.
DATALISTBOX SECTION	Defines the data list boxes used by Rational Change.
TEMPLATE DEFINITION SECTION	Defines information used to create Rational Change templates.

SYSTEM VARIABLES SECTION

Defines variables that affect how the entire Rational Change system functions. These variables include available Rational Synergy roles, the maximum number of query items to return, configuration for email submission of change requests, and the list of the configuration files to process at start-up.

All definitions in this section are enclosed in the following tag set:

```
[CCM_SYSTEM]...[/CCM_SYSTEM]
```

Tags summary

The following table shows the tags used in the `SYSTEM VARIABLES SECTION`.

Note You should use [CCM_SYSTEM] tags only in the `pt.cfg` file, with the exception of [ROLE], which you can include in role-based configuration files (for example, `admin_framework.cfg` and `user_framework.cfg`).

Tag	Description
[ACCENT_JAVA_DATE_FORMAT]	Specifies the format for date attributes (type CCM_DATE) sent back and forth between Rational Synergy sessions.
[ACCENT_ORIG_DATE_FORMAT]	Specifies the format for date attributes (type CCM_DATE) sent to and received from Rational Synergy ACcent **?????** functions.
[ACTIVE_USER_TIMEOUT]	Specifies how long users remain active after a period of inactivity.
[ADMIN_AUDIT_ENABLE]	Indicates if audit logging should occur.
[ADMIN_WINDOW_OPTIONS]	Controls the size and properties of pop-up windows.
[ALIAS]	Specifies the user-friendly name for a database.
[ALL_USERS_ABS_LIMIT]	Specifies the maximum user threshold.
[API_DATE_FORMAT]	Specifies the format for all date attributes (type `CCM_DATE`) sent to and received from Rational Change application programming interface (API) methods.

Tag	Description
[BEGIN_DAY_OF_WEEK]	Specifies the day of the week to be used for calculations based on weeks.
[BROWSE_RELATION_FILTER]	Limits the browse relationship results.
[BROWSE_VIEW_DISPLAY_STATUS_ATTRS]	Allows users to see status attributes.
[BROWSER_OS]	Specifies the default newline character sequence for the browser platform.
[CACHE_HEADERS]	Indicates whether the browser caches pages.
[CCM_DATABASE]	Specifies a database ID.
[CCM_DATABASE_UID]	Specifies a unique identifier for a database.
[CCM_HOST]	Specifies a host on which to run CM sessions.
[CCM_LDAP_MAPPING]	Maps lightweight directory access protocol (LDAP)/Synergy Directory Server (SDS) attributes to profile attributes.
[CCM_OPTIONAL_START_ARGS]	Specifies any optional arguments used when starting a CM session.
[CENTRAL_CR_DATABASE]	Specifies the path to the central server database.
[CFG_MERGE]	Lists the configuration files to process during start-up or when configuration files are reloaded.
[CHART_UNDEFINED_LABEL]	Specifies the value to use for a chart label if the charted attribute is missing (for example, "`undefined`").
[CM_STARTUP_MESSAGES]	Specifies a list of strings displayed if a session fails to start.
[CHART_COLORS]	Specifies the colors in hex format to be used in charts.
[CONTENT_TYPE]	Specifies the content type for a file downloaded from the Web.

Tag	Description
[CR_ACL_CLASSNAME]	Sets the path of a Java class to evaluate ACL rules for change requests.
[CRPROC_ID]	Specifies the ID of the installed CR Process.
[DEFAULT_DATABASE]	Specifies the database used for email change request submissions.
[DEFAULT_HOST_TYPE]	Specifies how the host machine is referenced.
[DELIMITER]	Specifies the character that delimits multiple values on the same line in the configuration file. Must be the **first** system variable tag.
[DESCRIPTION]	Describes a database, host, or role.
[ENABLED]	Indicates whether a database or host is available for running sessions.
[ENABLE_CRSTATUS_HYPERLINKS]	Specifies if the Status field has a hyperlink.
[END_DAY_OF_WEEK]	Specifies the day of the week to be used for calculations based on weeks.
[ENGINE_DAEMON]	Specifies whether an engine start-up daemon or standard socket session is used to start Rational Synergy sessions. (UNIX only.)
[FILE_EXTENSION]	Specifies the file extension for a file downloaded from the Web.
[FIND_CHAR]	Defines the character that is searched for in the body of text for an attribute.
[GENERATED_LINKS]	Specifies the format of the HTTP reference links.
[HOME_PAGE_ADMIN_ROLE]	Specifies the Rational Synergy privilege used to grant users the ability to administer the home page feature.
[HOST_TYPE]	Specifies the type of host.

Tag	**Description**
[JAVA_STRING_CHARSET]	Specifies the Java character set encoding to use when reading and writing text files.
[LDAP_GROUP_FILTER]	Limits group searches by using an additional query.
[LDAP_GROUP_OBJECT_CLASS]	Specifies which LDAP object class should be used when searching.
[MAIL_ADDR]	Specifies the `probtrac` email address for email change request submissions.
[MAIL_FILE]	Specifies the `probtrac` email change request submissions file.
[MAIL_SERVER]	Specifies the name of the mail server for email change request submissions.
[MAX_QUERY]	Specifies the system default for the maximum number of items that can be retrieved using a single query operation.
[MAX_SESSION_AGE]	Specifies how old a session must be before it is replaced.
[MAX_SESSIONS]	Specifies the maximum number of sessions to run against a specified database or host.
[MAX_STRING]	Specifies the system default for the maximum number of characters that can be retrieved for an attribute with the `text` data type in a query operation.
[MAX_TRANSITION_USERS]	Specifies the number of users that are supported by the User interface.
[MIME_TYPE]	Contains the file extension and content type definitions for a file downloaded from the Web.
[MIN_SESSIONS]	Specifies the minimum number of sessions to run against a database.
[MIRRORED_ATTRIBUTE]	

Tag	Description
[NAME]	Names a database, host, role, or the group for a report encoding definition, depending on the context.
[OBJECT_ACL_CLASSNAME]	Sets the path of a Java class to evaluate ACL rules for objects.
[QUERY_DATE_FORMAT]	Specifies the date format for queries.
[REPLACEMENT_TEXT]	Defines the string that will replace all instances of `FIND_CHAR` in the body of text.
[REPORT_ENCODING]	Allows character replacement in text output, for attributes, displayed in reports.
[REPORT_PROFILING_LEVEL]	Specifies the level of detail for information written to `event.log`.
[REPORT_SAVE_LIMIT]	Specifies for how many days an unsaved, user-run report is cached before being deleted automatically.
[REPORTS_DATE_FORMAT]	Specifies the date format for reports.
[REQUEST_RETRY_COUNT]	Specifies the number of attempts made to get a session before the request fails.
[ROLE]	Maps Rational Synergy roles to Rational Change roles.
[ROOT_PASSWORD]	Specifies the clear text user password for ROOT_USER.
[ROOT_USER]	Specifies the user name that overrides the root user name.
[SEARCH_INDEX_DIRECTORY]	Configures databases to share a search index.
[SEARCH_INDEX_TIMER]	Specifies the search indexing timer, in hours.
[SEND_MAIL]	Determines how email submissions are processed.

Tag	Description
[SESSION_MANAGER_TIMER]	Specifies how often the session pool is resized.
[SESSION_RATIO]	Specifies the number of users per session on the specified database.
[SHOW_DATE_FORMAT]	Specifies the format for all date attributes shown on forms.
[SHOW_MOTD]	Specifies if the message of the day is displayed.
[START_OFFLOAD]	Indicates whether the starting and stopping of sessions should be done in a separate thread.
[START_OFFLOAD_WAIT]	Specifies the number of seconds to wait for a thread that is starting or stopping a session.
[SUBMIT_DATE_FORMAT]	Specifies the format for all date attributes submitted from the browser.
[SUBREPORT_LEVELS]	Specifies the maximum number of subreport levels allowed in a report.
[SUPPORT_TASKS]	Specifies whether the task interface is enabled.
[TASK_ACL_CLASSNAME]	Sets the path of a Java class to evaluate ACL rules for tasks.
[THRESHOLD]	Specifies a host threshold.
[TRIGGER_OUTPUT_IS_UTF8]	Indicates if UTF-8 encoding is used by trigger I/O streams.
[TRANS_SERVER_CONFIG]	Groups the transaction server settings into a logical block.
[USER_SCOPE]	Groups the user list into manageable sizes.
[USER_SESSION_TIMEOUT]	Specifies the length of time before a user session times out.
[USER_WINDOW_OPTIONS]	Contains the list of available Rational Change roles.

[ACCENT_JAVA_DATE_FORMAT]

Specifies the format for all date attributes (type `CCM_DATE`) sent back and forth between Rational Synergy sessions. This format must be the same as the ACCENT_ORIG_DATE format, which uses different syntax for the corresponding format.

The ACCENT_JAVA_DATE_FORMAT applies to:

```
    [CCM_SYSTEM][SUBMIT_DATE_FORMAT][/SUBMIT_DATE_FORMAT][/
CCM_SYSTEM]
    [CCM_SYSTEM][SHOW_DATE_FORMAT][/SHOW_DATE_FORMAT][/
CCM_SYSTEM]
    [CCM_SYSTEM][REPORTS_DATE_FORMAT][/REPORTS_DATE_FORMAT][/
CCM_SYSTEM]
    [CCM_SYSTEM][API_DATE_FORMAT][/API_DATE_FORMAT][/CCM_SYSTEM]
    [CCM_SYSTEM][QUERY_DATE_FORMAT][/QUERY_DATE_FORMAT][/
CCM_SYSTEM]
    [CCM_SYSTEM][ACCENT_JAVA_DATE_FORMAT][/
ACCENT_JAVA_DATE_FORMAT][/CCM_SYSTEM]
```

Caution Your date format must match the Rational Synergy date format. Use the following Rational Synergy commands to show the date format:

UNIX:

```
$ ccm start
$ ccm query -n problem -t cvtype -f "%modify_time"
```

Windows:

```
> ccm start
> ccm query /n problem /t cvtype /f "%modify_time"
```

Specify the format as follows:

```
[CCM_SYSTEM]
   [ACCENT_JAVA_DATE_FORMAT]date_format[/
ACCENT_JAVA_DATE_FORMAT]
[/CCM_SYSTEM]
```

The default setting is as follows:

```
[CCM_SYSTEM]
   [ACCENT_JAVA_DATE_FORMAT]yyyy/MM/dd H:mm:s[/
ACCENT_JAVA_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`Tue May 10 12:15:22 2007`", specify the following format:

```
[CCM_SYSTEM]
   [ACCENT_JAVA_DATE_FORMAT]EEE MMM dd HH:mm:ss yyyy
   [/ACCENT_JAVA_DATE_FORMAT]
[/CCM_SYSTEM]
```

[ACCENT_ORIG_DATE_FORMAT]

Specifies the format for all date attributes (type `CCM_DATE`) sent back and forth internally between Rational Synergy sessions. This format must be the same as the ACCENT_JAVA_DATE format, which uses different syntax for the corresponding format.

Caution Your date format must match the Rational Synergy date format. Use the following Rational Synergy commands to show the date format:

UNIX:

```
$ ccm start
$ ccm query -n problem -t cvtype -f "%modify_time"
```

Windows:

```
> ccm start
> ccm query /n problem /t cvtype /f "%modify_time"
```

Specify the format as follows:

```
[CCM_SYSTEM]
   [ACCENT_ORIG_DATE_FORMAT]date_format[/
ACCENT_ORIG_DATE_FORMAT]
[/CCM_SYSTEM]
```

The default setting is as follows:

```
[[CCM_SYSTEM]
    [ACCENT_ORIG_DATE_FORMAT]%Y/%m/%d %H:%M:%S[/
ACCENT_ORIG_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`Tue May 10 12:15:22 2007`", specify the following format:

```
[CCM_SYSTEM]
    [ACCENT_ORIG_DATE_FORMAT]%a %B %d %H:%M:%S %Y
    [/ACCENT_ORIG_DATE_FORMAT]
[/CCM_SYSTEM]
```

[ACTIVE_USER_TIMEOUT]

Specifies how long users remain active after a period of inactivity.

For example, to time out users who have been idle for at least five minutes, use the following entry:

```
[CCM_SYSTEM]
    [ACTIVE_USER_TIMEOUT]5[/ACTIVE_USER_TIMEOUT]
[CCM_SYSTEM]
```

The number of active users is used to compute the session pool size, so timing out users frequently results in a "tighter" pool. However, timing out users too frequently can lead to unnecessary resizing and performance degradation.

[ADMIN_AUDIT_ENABLE]

Informs the server if admin audit logging is enabled. Valid entries are `true/false`. The default is `true`.

Disable logging as follows:

```
[CCM_SYSTEM][ADMIN_AUDIT_ENABLE]false[/ADMIN_AUDIT_ENABLE]
[/CCM_SYSTEM]
```

[ADMIN_WINDOW_OPTIONS]

Controls the size and properties of pop-up windows. The Administrator can change these settings to globally change pop-up window behavior for all users. The window options are accessed through the PTAdmin Wslet. The Wslet is replaced with window option values within the tag set.

The same options are available for the REPORT_WINDOW_OPTIONS, USER_WINDOW_OPTIONS, UTILITY_WINDOW OPTIONS, and UTILITY_SMALL_WINDOW_OPTIONS.

For example:

```
[ADMIN_WINDOW_OPTIONS]resizable,width=1014,height=710,screenX=0
,screenY=0,
   left=0,top=0[/ADMIN_WINDOW_OPTIONS]
```

The code:

```
  var newWindow =
window.open("load_template_call","new_window","<!-- WSLET
CODE=PTAdmin --><!-- PARAM NAME=ADMIN_WINDOW_OPTIONS --><!-- /
WSLET -->");
```

is returned as:

```
  var newWindow =
window.open("load_template_call","new_window","

resizable,width=1014,height=710,screenX=0,screenY=0,left=0,top=
0");
```

The following example shows the names (Admin, User, Report, Utility, and Utility Small) of the windows you can define:

```
[CCM_SYSTEM]
   [ADMIN_WINDOW_OPTIONS]your window options[/
ADMIN_WINDOW_OPTIONS]
[/CCM_SYSTEM]
[CCM_SYSTEM]
   [USER_WINDOW_OPTIONS]your window options[/
USER_WINDOW_OPTIONS]
[/CCM_SYSTEM]
[CCM_SYSTEM]
   [REPORT_WINDOW_OPTIONS]your window options[/
REPORT_WINDOW_OPTIONS]
[/CCM_SYSTEM]
[CCM_SYSTEM]
   [UTILITY_WINDOW_OPTIONS]your window options[/
UTILITY_WINDOW_OPTIONS]
[/CCM_SYSTEM]
[CCM_SYSTEM]
   [UTILITY_SMALL_WINDOW_OPTIONS]your window options[/
UTILITY_SMALL_WINDOW_OPTIONS]
[/CCM_SYSTEM]
```

[ALIAS]

Specifies the user-friendly name for a database. The value is any string.

This tag is only used inside the [CCM_DATABASE] tags.

[ALL_USERS_ABS_LIMIT]

Specifies the maximum limit of user threshold. This limit determines whether the All tab should be disabled or not, irrespective of User settings. If the "Disable the All tab" option is not checked, the threshold should be used to determine which of the tabs (All or Filter) is the default.

For example, to set the limit at 4000, use the following entry:

```
[CCM_SYSTEM]
   [ALL_USERS_ABS_LIMIT]4000[/ALL_USERS_ABS_LIMIT]
[CCM_SYSTEM]
```

[API_DATE_FORMAT]

Specifies the format for all date attributes (type `CCM_DATE`) sent to and received from Rational Change API methods.

Specify the format as follows:

```
[CCM_SYSTEM]
   [API_DATE_FORMAT]date_format[/API_DATE_FORMAT]
[/CCM_SYSTEM]
```

See [ACCENT_JAVA_DATE_FORMAT] for a list of date formats and their symbols.

[BEGIN_DAY_OF_WEEK]

Specifies the day of the week to be used for calculations based on weeks, such as with query operations. This tag is used in conjunction with [END_DAY_OF_WEEK] to define the beginning and ending days of the week.

The default settings are BEGIN_DAY_OF_WEEK: MONDAY and END_DAY_OF_WEEK: SUNDAY.

Define the week as follows:

```
[CCM_SYSTEM]

[BEGIN_DAY_OF_WEEK]SUNDAY|MONDAY|TUESDAY|WEDNESDAY|THURSDAY|FRI
DAY|SATURDAY
   [/BEGIN_DAY_OF_WEEK]
[/CCM_SYSTEM]

[CCM_SYSTEM]

[END_DAY_OF_WEEK]SUNDAY|MONDAY|TUESDAY|WEDNESDAY|THURSDAY|FRIDA
```

```
Y|SATURDAY
    [/END_DAY_OF_WEEK]
[/CCM_SYSTEM]
```

[BROWSE_RELATION_FILTER]

Limits the browse relationship results by supplying a query filter for each named relationship.

For example:

```
[CCM_SYSTEM]
    [BROWSE_RELATION_FILTER]
    [BROWSE_RELATION_NAME] duplicate
    [/BROWSE_RELATION_NAME]
    [QRY_STRING]crstatus='assigned'[/QRY_STRING}
    [/BROWSE_RELATION_FILTER]
[CCM_SYSTEM]
```

Note that a RELATION_NAME and QRY_STRING are required.

[BROWSE_VIEW_DISPLAY_STATUS_ATTRS]

Allows users to see the status attributes in the Browse view. For CRs, this is the crstatus, and for tasks and objects, the status. It doesn't apply for attachments. Legal values are `true` and `false`.

For example:

```
[CCM_SYSTEM]
    [BROWSE_VIEW_DISPLAY_STATUS_ATTRS]true
    [/BROWSE_VIEW_DISPLAY_STATUS_ATTRS]
[CCM_SYSTEM]
```

[BROWSER_OS]

Specifies the default newline character sequence for the browser platform. Possible values are `CRLF` (Windows) or `LF` (UNIX).

For example, make `CRLF` the Windows platforms newline character, and `LF` the Solaris™ platform newline character, as follows:

```
[CCM_SYSTEM]
    [BROWSER_OS]
    WinNT:CRLF|Windows NT:CRLF|Windows:CRLF|Solaris:LF
    [/BROWSER_OS]
[CCM_SYSTEM]
```

The default value for [BROWSER_OS] is `LF`.

[CACHE_HEADERS]

Indicates whether the browser caches pages. Setting header caching ON (`TRUE`) causes the browser to cache the page, so that resizing the browser window does not cause the page display request to be resubmitted.

In general, Microsoft ® Internet Explorer® users should set caching to `FALSE`, and Netscape™ users should set caching to `TRUE`.

The default setting is `FALSE`.

Note You must upgrade old (2.0) configuration files to use this tag.

Set caching as follows:

```
[CACHE_HEADERS]your_value[/CACHE_HEADERS]
```

For example, set header caching to `TRUE` as follows:

```
[CCM_SYSTEM]
   [CACHE_HEADERS]TRUE[/CACHE_HEADERS]
[/CCM_SYSTEM]
```

[CCM_DATABASE]

Specifies a database used by Rational Change. You can specify more than one database configuration entry.

Database entries have the following required subtags:

```
[CCM_SYSTEM]
   [CCM_DATABASE]
      [NAME]the database path[/NAME]
      [ALIAS]short, user-friendly name[/ALIAS]
      [SESSION_RATIO]users per session[/SESSION_RATIO]
      [MIN_SESSIONS]minimum sessions[/MIN_SESSIONS]
      [MAX_SESSIONS]maximum sessions[/MAX_SESSIONS]
      [DESCRIPTION]a description of the database[/DESCRIPTION]
      [CCM_DATABSE_UID]unique identifier for security file[/
CCM_DATABSE_UID]
      [ENABLED]true|false[/ENABLED]
   [/CCM_DATABASE]
[/CCM_SYSTEM]
```

[CCM_DATABASE_UID]

Specifies a unique identifier for a database.

Rational Change uses the identifier for naming the security file corresponding to each database. The security file is named `unique_id.cfg`, so the value can be a string suitable for a file name.

This tag is only used inside the [CCM_DATABASE] tags.

[CCM_HOST]

Specifies a host on which to run CM sessions. You can specify more than one host configuration entry.

Host entries have the following required subtags:

```
[CCM_SYSTEM]
    [CCM_HOST]
        [NAME]the name of the host[/NAME]
        [HOST_TYPE]NT|UNIX[/HOST_TYPE]
        [MAX_SESSIONS]maximum sessions[/MAX_SESSIONS]
        [PRIORITY]1 (highest) - 10 (lowest)[/PRIORITY]
        [THRESHOLD]the number of sessions at which degradation
occurs[/THRESHOLD]
        [DESCRIPTION]a description of the host[/DESCRIPTION]
        [ENABLED]true|false[/ENABLED]
    [/CCM_HOST]
[/CCM_SYSTEM]
```

[CCM_LDAP_MAPPING]

Maps LDAP/SDS attributes to Rational Change profile/preference attributes. This is useful in that basic information about a user needs to be entered only once in the LDAP/SDS server. If the information is changed in the LDAP/SDS server, Rational Change automatically picks up the changes.

The LDAP mapping attributes `NAME` and `ALIAS` properties generally begin with *`user_`*. By default, generated submit preference attributes are always *`user_cr_attribute_name`*.

The LDAP_ATTRIBUTE comes from a user's object on the LDAP/SDS server. For example, the Rational Change profile attribute `user_email` is mapped to the LDAP/SDS attribute mail, and the `user_address` is mapped to postalAddress.

You LDAP mapping entries should be similar to the following entry:

```
[CCM_SYSTEM]
   [CCM_LDAP_MAPPING]
      [NAME]user profile attribute name[/NAME]
      [LDAP_ATTRIBUTE]LDAP/SDS attribute name[/
LDAP_ATTRIBUTE]
      [ALIAS]optional submit preference attribute name[/
ALIAS]
   [/CCM_LDAP_MAPPING]
[/CCM_SYSTEM]
```

[CCM_OPTIONAL_START_ARGS]

Specifies any optional arguments used when starting a CM session.

The most common argument is `-h` *`hostname`*, which specifies that the engine processes should be run on the named host.

For example, to start all engine processes on the host `pulsar`, use the following entry:

```
[CCM_SYSTEM]
   [CCM_OPTIONAL_START_ARGS]-h pulsar[/
CCM_OPTIONAL_START_ARGS]
[/CCM_SYSTEM]
```

Maps Rational Synergy privileges to Rational Change roles. "*" is a wild card.

Uses parameters such as the following:

```
[CCM_SYSTEM]
   [CCM_ROLE]ccm_role1|ccm_role2[/CCM_ROLE]
   [WEB_ROLES]cs_login_role1|cs_login_role2[/WEB_ROLES]
[/CCM_SYSTEM]
```

Delimit the list using the defined delimiter ([DELIMITER]).

Do not include spaces between entries or more than one delimiter between values.

For example, give users with the *pt_admin* or *ccm_admin* privileges all Rational Change roles (for example, in the `admin_framework.cfg` file):

```
[CCM_SYSTEM]
   [CCM_ROLE]pt_admin|ccm_admin[/CCM_ROLE]
   [WEB_ROLES]*[/WEB_ROLES]
[/CCM_SYSTEM]
```

Give users with *any* Rational Synergy privilege the *User* Rational Change role (for example, in the `user_framework.cfg` file):

```
[CCM_SYSTEM]
    [CCM_ROLE]*[/CCM_ROLE]
    [WEB_ROLES]User[/WEB_ROLES]
[/CCM_SYSTEM]
```

Give users with the *developer*, *assigner*, or *tester* Rational Synergy privilege the *User* Rational Change role (for example, in the `user_framework.cfg` file):

```
[CCM_SYSTEM]
    [CCM_ROLE]developer|assigner|tester[/CCM_ROLE]
    [WEB_ROLES]User[/WEB_ROLES]
[/CCM_SYSTEM]
```

Also see "[USER_SCOPE]" on page 63 and "Users, roles, and privileges" on page 21.

[CENTRAL_CR_DATABASE]

Gives information about the central CR database as a fully qualified database path for a central server. This entry is not applicable for stand-alone and remote servers and is not present in these modes.

For example, to set the path to `\\abc123\ccmdb\test1`, use the following entry:

```
[CCM_SYSTEM]
    [CENTRAL_CR_DATABASE]\\abc123\ccmdb\test[/
CENTRAL_CR_DATABASE]
[CCM_SYSTEM]
```

[CFG_MERGE]

Lists the configuration files to merge during start-up or when configuration files are reloaded. This tag set is used only in the `pt.cfg` file.

When you install CSPackage, Rational Change includes the package configuration files in the list. For example, when you install the `dev_process` and `dev_reports` packages, the `dev_process.cfg` and `dev_reports.cfg` files are added to the merge list as follows:

```
[CCM_SYSTEM]
    [CFG_MERGE]dev_process.cfg|dev_reports.cfg
    [/CFG_MERGE]
[/CCM_SYSTEM]
```

Note The `admin_framework.cfg` and `user_framework.cfg` configuration files are not listed here, even though they are merged, as well.

You can define customized configuration entries in one or more configuration files by listing them in [CFG_MERGE]. Thereafter, when you configuration data, Rational Change does the following:

- Adds new template definitions.
- Adds role-based template definitions for new roles.
- Overrides old template definitions with new template definitions for existing roles.

For example, you can create a `myfile.cfg` configuration file that contains customized templates and HTML encoding values, then list the file in [CFG_MERGE], as follows:

```
[CCM_SYSTEM]
    [CFG_MERGE]dev_process.cfg|dev_reports.cfg|myfile.cfg
    [/CFG_MERGE]
[/CCM_SYSTEM]
```

When you restart a session or load configuration data, Rational Change merges all configuration files.

[CHART_UNDEFINED_LABEL]

Specifies the string to use for a chart label, for [CHART_TOOL], if the charted attribute is missing.

For example, the following chart uses `Undefined` for the label if the charted attribute is missing:

```
[CCM_SYSTEM]
    [CHART_UNDEFINED_LABEL]Undefined[/CHART_UNDEFINED_LABEL]
[/CCM_SYSTEM]
```

[CM_STARTUP_MESSAGES]

Specifies a list of strings displayed if a session fails to start. The messages prevent the host from becoming disabled automatically.

Set start-up messages as follows:

```
[CM_STARTUP_MESSAGES]your_value|your_value|your_value...[/
CM_STARTUP_MESSAGES]
```

For example:

```
[CCM_SYSTEM]
   [CM_STARTUP_MESSAGES]
   Starting a session is not allowed|is protected|Cannot locate 
object
   registrar|Lost connection to engine|Couldn't obtain security 
level from
   license manager|License manager is not running|is an invalid 
role|Illegal
   user or role, access denied
   [/CM_STARTUP_MESSAGES]
[/CCM_SYSTEM]
```

[CHART_COLORS]

Specifies a pipe-delimited list of hex colors to be used for charting.

For example:

```
[CCM_SYSTEM]
   [CHART_COLORS]#5555FF|#FF5555|#55FF55|#FFFF55[/CHART_COLORS]
[/CCM_SYSTEM]
```

[CONTENT_TYPE]

Specifies the content type for a file downloaded from the Web.

This tag is only used inside the [MIME_TYPE] tags.

[CR_ACL_CLASSNAME]

Sets the complete path of a Java class to run in order to evaluate ACL rules for change requests.

For example:

```
[CCM_SYSTEM]
   [CR_ACL_CLASSNAME]com.abcd.cs.acl.AclEvaluator
   [/CR_ACL_CLASSNAME]
[CCM_SYSTEM]
```

[CRPROC_ID]

Specifies the ID of the installed CR Process. This tag set is maintained by the package install/uninstall process; therefore, you should not edit it manually.

[DEFAULT_DATABASE]

Specifies the database used for email change request submissions (email submit forms).

Rational Change uses database names to load “from database” list boxes. When Rational Change parses a template and the database is not known, such as when email submit forms are being processed, the default database is used to fill the list box.

Set the default database as follows:

```
[CCM_SYSTEM]
   [DEFAULT_DATABASE]database_path[/DEFAULT_DATABASE]
[/CCM_SYSTEM]
```

For example, set the database path to `\\angler\ccmdb\sqe_test` as follows:

```
[CCM_SYSTEM]
   [DEFAULT_DATABASE]\\angler\ccmdb\sqe_test[/DEFAULT_DATABASE]
[/CCM_SYSTEM]
```

[DEFAULT_HOST_TYPE]

Specifies if the Rational Change host machine is specified as an IP address or a DNS name. The default setting is IP address.

Set the host type as follows:

```
[CCM_SYSTEM]
   [DEFAULT_HOST_TYPE]hostname|ipaddress[/DEFAULT_HOST_TYPE]
[/CCM_SYSTEM]
```

If a hostname parameter is specified in the `synergy.xml` file, the DEFAULT_HOST_TYPE setting is ignored.

By default the `synergy.xml` file does not contain a hostname argument. To disable the DEFAULT_HOST_TYPE setting, add the following lines to the `synergy.xml` file under the port argument:

```
<Call name="setInitParameter">
  <Arg>hostname</Arg>
  <Arg>value of fully qualified hostname</Arg>
</Call>
```

[DELIMITER]

Specifies the character that delimits multiple values on the same line in a configuration file.

This tag set is used only in the `pt.cfg` file.

The default delimiter is the pipe (|) character.

Note [DELIMITER] must be the first tag in the SYSTEM VARIABLES SECTION after the [CCM_SYSTEM] tag.

[DESCRIPTION]

Describes a database, host, or role, depending on the context of the enclosing tag.

This tag is used inside the [CCM_DATABASE], [CCM_HOST], and [ROLE] tags.

[ENABLE_CRSTATUS_HYPERLINKS]

Specifies if the crstatus (Status) field is displayed as a hyperlink in reports.

The default setting is false.

[ENABLED]

Indicates whether a database or host is available for running sessions. The value must be boolean.

This tag is used inside the [CCM_DATABASE] and [CCM_HOST] tags.

[END_DAY_OF_WEEK]

Specifies the day of week to be used for calculations based on weeks. This tag is used in conjunction with [BEGIN_DAY_OF_WEEK] to define the beginning and ending days of the week.

See [BEGIN_DAY_OF_WEEK].

[ENGINE_DAEMON]

Specifies whether an engine start-up daemon or standard socket session is used to start Rational Synergy sessions (UNIX only).

When set to TRUE, the engine start-up daemon is used. When set to FALSE, a standard socket connection is used.

Set the engine daemon as follows:

```
[ENGINE_DAEMON]your_value[/ENGINE_DAEMON]
```

For example, set engine daemon to `TRUE` as follows:

```
[CCM_SYSTEM]
    [ENGINE_DAEMON]TRUE[/ENGINE_DAEMON]
[/CCM_SYSTEM]
```

[FILE_EXTENSION]

Specifies the file extension for a file downloaded from the Web.

This tag is only used inside the [MIME_TYPE] tags.

[FIND_CHAR]

Defines the character that is searched for in the body of text for an attribute.

This tag is only used inside the [REPORT_ENCODING] tags.

Carriage returns and null characters are the only unprintable characters interpreted by Rational Change. Because these unprintable characters cannot be specified in [FIND_CHAR], CRLF and NULL are used in their place to denote whatever your system recognizes as carriage returns and null characters.

Therefore, if you want to replace carriage returns and null characters, specify CRLF and NULL for them, respectively, as is shown in the following default settings for HTML files:

```
[CCM_SYSTEM]
    [REPORT_ENCODING]
        [NAME]HTML[/NAME]
        [FIND_CHAR]CRLF[/FIND_CHAR]
        [REPLACEMENT_TEXT]<BR>[/REPLACEMENT_TEXT]
    [/REPORT_ENCODING]
[/CCM_SYSTEM]

[CCM_SYSTEM]
    [REPORT_ENCODING]
        [NAME]HTML[/NAME]
        [FIND_CHAR]NULL[/FIND_CHAR]
        [REPLACEMENT_TEXT]&nbsp;[/REPLACEMENT_TEXT]
    [/REPORT_ENCODING]
[/CCM_SYSTEM]
```

[GENERATED_LINKS]

Specifies the format of the http reference links, either absolute or relative. Absolute links contain the hostname and port address, relative links do not contain the hostname or port address. The default setting for [GENERATED_LINKS] is absolute.

For example, an absolute link is displayed as follows:

```
http://angler:8613/context/trapeze/ptimages/ws_gray_pixel.gif'
"hostname"
```

A relative link is displayed as follows:

```
/context/trapeze/ptimages/ws_gray_pixel.gif
```

[HTML_CONTENT]

Specifies the http content type for a response. Accepts any legal content-type declaration.

Note You should not change this value.

For example, the following definition is the http content specification for a standard HTML page:

```
[CCM_SYSTEM]
    [HTML_CONTENT]text/html;[/HTML_CONTENT]
[/CCM_SYSTEM]
```

[HOME_PAGE_ADMIN_ROLE]

Specifies the Rational Synergy privilege (*report_builder*, *developer*, and so on) a user must have to administer the home page feature.

In the example below, users having the privilege *report_builder* can administer the home pages.

```
[CCM_SYSTEM]
    [HOME_PAGE_ADMIN_ROLE]report_builder[/HOME_PAGE_ADMIN_ROLE]
[/CCM_SYSTEM]
```

[HOST_TYPE]

Specifies the host type. The value should be `NT` for Windows hosts and `UNIX` for UNIX hosts.

This tag is only used inside the [CCM_HOST] tags.

[JAVA_STRING_CHARSET]

Specifies the Java character set encoding to use when reading and writing text files.

For example, the following is an ISO character set for interpreting 8-bit characters:

```
[CCM_SYSTEM]
    [JAVA_STRING_CHARSET]ISO-8859-1[/JAVA_STRING_CHARSET]
[/CCM_SYSTEM]
```

Note You should not change this value.

[LDAP_GROUP_FILTER]

Specifies that group searches can be limited with an additional query filter. This can improve performance, as it revents too many results from being returned.

```
[CCM_SYSTEM]
    [LDAP_GROUP_FILTER]groupOfUniqueNames=t*[/LDAP_GROUP_FILTER]
[/CCM_SYSTEM]
```

[LDAP_GROUP_OBJECT_CLASS]

Specifies to the server which LDAP object class should be used when searching and dealing with LDAP groups.

```
[CCM_SYSTEM]
    [LDAP_GROUP_OBJECT_CLASS]groupOfUniqueNames[/
LDAP_GROUP_OBJECT_CLASS]
[/CCM_SYSTEM]
```

[MAIL_ADDR]

Specifies the `probtrac` email address for email change request submissions (for example, `probtrac@mycompany.com`).

This tag is only used inside the [SEND_MAIL] tags, when SEND_MAIL is set to `true`.

[MAIL_FILE]

Specifies the `probtrac` email change request submission file (for example, `/var/spool/mail/probtrac` or `g:\ccm\mail\probtrac`). This value must match the `probtrac` value in the Rational Synergy `dbpath.dft` file.

This tag is only used inside the [SEND_MAIL] tags, when SEND_MAIL is set to `false`.

[MAIL_PROTOCOL]

(Windows only). Specifies the mail server protocol used for email change request submissions.

This tag is used only inside the [SEND_MAIL] tags, when SEND_MAIL is set to `true`.

[MAIL_SERVER]

Specifies the name of the SMTP mail server that handles email change request submissions.

[MAX_QUERY]

Specifies the system default for the maximum number of items that can be retrieved using a single query operation. You can override this value in an individual report by setting [MAX_QUERY] in the Report definition.

The default value of [MAX_QUERY] is `1000`.

[MAX_SESSION_AGE]

Specifies how old a session must be before it is replaced.

CM sessions grow over time, so eventually they must be replaced to limit their memory footprint. The value is in hours and must be a positive integer.

For example, to terminate sessions that have existed for at least a day, use the following entry:

```
[CCM_SYSTEM]
```

```
    [MAX_SESSION_AGE]24[/MAX_SESSION_AGE]
[/CCM_SYSTEM]
```

[MAX_SESSIONS]

Specifies the maximum number of sessions to run against a specified database or host.

Regardless of the number of active users, the session pool will contain no more than this many sessions for the corresponding database or host. The value must be a positive integer.

This tag is only used inside the [CCM_DATABASE] and [CCM_HOST] tags.

[MAX_STRING]

Specifies the system default for the maximum number of characters that can be retrieved for an attribute with the `text` data type in a query operation. You can override this value in an individual report by setting [MAX_STRING] in the Report definition.

The default value of [MAX_STRING] is `32000`.

[MAX_TRANSITION_USERS]

Controls the number of users supported by the User interface. Once the limit is exceeded, users must specify a custom list of users from the Preferences page.

A value of 0 for this setting has a special meaning; that is, users must create their own list.

The setting governs the user list presented in any [CCM_USER] control.

For example, specify that 50 users is the maximum as follows:

```
[CCM_SYSTEM]
    [MAX_TRANSITION_USERS]50[/MAX_TRANSITION_USERS]
[/CCM_SYSTEM]
```

[MIME_TYPE]

Contains the file extension and content type definitions for a file downloaded from the Web, as well as specifying the browser type. Use this definition only for attachments and reports that are not HTML; all such attachments and reports require this definition.

Also see [FILE_EXTENSION] and [CONTENT_TYPE].

For example, the following are the default definitions for Word and PDF file MIME types:

```
[CCM_SYSTEM]
    [MIME_TYPE]
        [FILE_EXTENSION]doc[/FILE_EXTENSION]
        [CONTENT_TYPE]application/msword[/CONTENT_TYPE]
    [/MIME_TYPE]
[/CCM_SYSTEM]

[CCM_SYSTEM]
    [MIME_TYPE]
        [FILE_EXTENSION]pdf[/FILE_EXTENSION]
        [CONTENT_TYPE]application/pdf[/CONTENT_TYPE]
    [/MIME_TYPE]
[/CCM_SYSTEM]
```

Use one of the following to define the browser to be used. If true, the mime type is enabled for the browser; if false, the mime type is disabled for the browser.

```
[INTERNET_EXPLORER]true|false[/INTERNET_EXPLORER]
[NETSCAPE]true/false[/NETSCAPE]
```

[MIN_SESSIONS]

Specifies the minimum number of sessions to run against a specified database. Regardless of the number of active users, the session pool will contain at least this many sessions for the corresponding database. The value should be a non-negative integer.

This tag is only used inside the [CCM_DATABASE] tags.

[MIRRORED_ATTRIBUTE]

Mirrored attributes are attributes that can be used on a report to query for a single attribute on a related object and include that attribute as if it were found by the original query.

For example, if you were querying for CRs and wanted to include priorities for tasks, you could create a mirrored attribute that looks up the related tasks on the CR and returns their priorities. The DELIMITER attribute is used when multiple objects are found.

```
[MIRRORED_ATTRIBUTE]
    [NAME]Child Info[/NAME]
    [ATTRIBUTE]child_special[/ATTRIBUTE]
    [TYPE]CR[/TYPE]
    [QRY_STRING]is_associated_child_of('%instance/problem/
problem%problem_number/1')[/QRY_STRING]
    [DELIMITER],[/DELIMITER]
[/MIRRORED_ATTRIBUTE]
```

[NAME]

Names a database, host, role, or the group for a report encoding definition, depending on the context. Name is a subtag and serves to differentiate the enclosing tag (which establishes the context).

This tag is used inside [CCM_DATABASE], [CCM_HOST], [ROLE], and [REPORT_ENCODING] tags.

[OBJECT_ACL_CLASSNAME]

Sets the complete path of a Java class to run in order to evaluate ACL rules for objects.

For example:

```
[CCM_SYSTEM]
   [OBJECT_ACL_CLASSNAME]com.abcd.cs.acl.ObjectAclEvaluator
   [/OBJECT_ACL_CLASSNAME]
[CCM_SYSTEM]
```

[PRIORITY]

Specifies a host priority. The value should be an integer from 1 (highest) to 10 (lowest).

Rational Change uses the priority when determining from which host to add/ drop sessions when the session pool is resized. Sessions are started on high priority hosts first, and dropped from high priority hosts last. In general, a powerful host (that is a host with a fast CPU and a great deal of RAM) should be configured to have a higher priority than a less powerful host.

[QUERY_DATE_FORMAT]

Specifies the format for all date attributes used in queries.

Specify the format as follows:

```
[CCM_SYSTEM]
   [QUERY_DATE_FORMAT]date_format[/QUERY_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`05/10/2007 12:22 PM`" on queries, specify the following format:

```
[CCM_SYSTEM]
   [QUERY_DATE_FORMAT]MM/dd/yyyy h:mm a[/QUERY_DATE_FORMAT]
[/CCM_SYSTEM]
```

See "[ACCENT_JAVA_DATE_FORMAT]" on page 33 for a list of date formats and their symbols.

[REPLACEMENT_TEXT]

Defines the string that will replace all instances of [FIND_CHAR] in an attribute value.

This tag is only used inside the [REPORT_ENCODING] tags.

[REPORT_ENCODING]

Allows character replacement in text output for attributes displayed in reports.

Uses parameters such as the following:

```
[CCM_SYSTEM]
    [REPORT_ENCODING]
    [NAME]your_grouping_name[/NAME]
    [FIND_CHAR]character_to_replace[/FIND_CHAR]
    [REPLACEMENT_TEXT]replacement_string[/REPLACEMENT_TEXT]
    [/REPORT_ENCODING]
[/CCM_SYSTEM]
```

Also see [NAME], [FIND_CHAR], and [REPLACEMENT_TEXT].

This feature is used primarily for CCM_TEXT types, but you can use it for any type. However, you may use the feature in a REPORT_DETAILS, REPORT_AUTO_ATTR_VALUE, or REPORT_SPAN_ATTR_VALUE PTReport wslet call, and the wslet call VALUE should be the name of the REPORT_ENCODING group that you want used.

Note Using this feature excessively can degrade performance.

The following are the system defaults for report encoding:

```
[REPORT_ENCODING]
    [NAME]HTML[/NAME]
    [FIND_CHAR]<[/FIND_CHAR]
    [REPLACEMENT_TEXT]&lt;[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]

[REPORT_ENCODING]
    [NAME]HTML[/NAME]
    [FIND_CHAR]>[/FIND_CHAR]
    [REPLACEMENT_TEXT]&gt;[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]
```

```
[REPORT_ENCODING]
    [NAME]HTML[/NAME]
    [FIND_CHAR]&[/FIND_CHAR]
    [REPLACEMENT_TEXT]&amp;[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]

[REPORT_ENCODING]
    [NAME]HTML[/NAME]
    [FIND_CHAR]CRLF[/FIND_CHAR]
    [REPLACEMENT_TEXT]<BR>[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]

[REPORT_ENCODING]
    [NAME]HTML[/NAME]
    [FIND_CHAR]NULL[/FIND_CHAR]
    [REPLACEMENT_TEXT]&nbsp;[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]

[REPORT_ENCODING]
    [NAME]RTF[/NAME]
    [FIND_CHAR]CRLF[/FIND_CHAR]
    [REPLACEMENT_TEXT]\par[/REPLACEMENT_TEXT]
[/REPORT_ENCODING]
```

[REPORT_PROFILING_LEVEL]

Specifies the level of detail for information written to `event.log`.

Report profiling level has three levels: OFF, LOW, and HIGH. When activated (LOW or HIGH), the profiling generates additional `event.log` output for timing reporting routines (for example, querying in ACcent, creating a Java data structure, parsing the report templates, and streaming the report back to the browser). HIGH provides more verbose output.

Set profiling as follows:

```
[REPORT_PROFILING_LEVEL]your_value[/REPORT_PROFILING_LEVEL]
```

For example, set profiling to `OFF` as follows:

```
[CCM_SYSTEM]
    [REPORT_PROFILING_LEVEL]OFF[/REPORT_PROFILING_LEVEL]
[/CCM_SYSTEM]
```

[REPORT_SAVE_LIMIT]

Specifies for how many days an unsaved, user-run report is cached before being deleted automatically.

All reports a users runs are cached in the Web server file system for a number of days. The files are located in the *`CHANGE_APP_HOME`*`/trapeze/wsfiles/`*`user_name`* directory. When a user saves a report, the associated files are set to read only, which prevents the report manager from deleting the files. If the user does not save the report explicitly, the Rational Change report manager automatically deletes all reports that are older than REPORT_SAVE_LIMIT days.

A user can manually delete his own files using the user interface.

Set report saving limit as follows:

```
[REPORT_SAVE_LIMIT]your_value[/REPORT_SAVE_LIMIT]
```

For example, set the report-saving limit to `1 day` as follows:

```
[CCM_SYSTEM]
   [REPORT_SAVE_LIMIT]1[/REPORT_SAVE_LIMIT]
[/CCM_SYSTEM]
```

[REPORT_WINDOW_OPTIONS]

Controls the size and properties of pop-up windows.

See [ADMIN_WINDOW_OPTIONS] for more information.

[REPORTS_DATE_FORMAT]

Specifies the format for all date attributes shown on reports.

Specify the format as follows:

```
[CCM_SYSTEM]
   [REPORTS_DATE_FORMAT]date_format[/REPORTS_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`05/10/2007 12:22 PM`" on reports, specify the following format:

```
[CCM_SYSTEM]
   [REPORTS_DATE_FORMAT]MM/dd/yyyy h:mm a[/REPORTS_DATE_FORMAT]
[/CCM_SYSTEM]
```

See "[ACCENT_JAVA_DATE_FORMAT]" on page 33 for a list of date formats and their symbols.

Note You can override the default format using a wslet. See "CCM_USER_PROFILE" on page 352.

[REQUEST_RETRY_COUNT]

Specifies the number of attempts made to obtain a session before the request fails. If all sessions are busy, Rational Change tries to satisfy the request again after a brief wait. The non-negative integer value specified here indicates the number of times to retry.

To retry five times, use the following entry:

```
[CCM_SYSTEM]
    [REQUEST_RETRY_COUNT]5[/REQUEST_RETRY_COUNT]
[/CCM_SYSTEM]
```

[ROLE]

Defines Rational Synergy privileges to Rational Change roles.

For example, the following entry maps the Rational Change *Admin* role to the administrator role:

```
[CCM_SYSTEM]
    [ROLE]
        [NAME]Admin[/NAME]
        [DESCRIPTION]Rational Change Administrator[/DESCRIPTION]
    [/ROLE]
    ...
[/CCM_SYSTEM]
```

Note You should create and map all roles using the **Users** tab in the Admin interface.

Also see “Users, roles, and privileges” on page 21.

[ROOT_PASSWORD]

Specifies the clear text user password for [ROOT_USER].

Note Use this option only if the encrypted password file becomes corrupted or gets deleted.

Set the password as follows:

```
[ROOT_PASSWORD]clear text password[/ROOT_PASSWORD]
```

For example, set the ROOT_USER password to *my_password*:

```
[CCM_SYSTEM]
    [ROOT_PASSWORD]my_password[/ROOT_PASSWORD]
[/CCM_SYSTEM]
```

[ROOT_USER]

Specifies the user name that overrides the *root* user name.

The *root* user is the user with which Rational Synergy sessions are run.

Note Use this option only if the encrypted password file becomes corrupted or gets deleted.

Set the root user to `your_value` as follows:

```
[ROOT_USER]your_value[/ROOT_USER]
```

For example, set the *root* user to `ccm_root`:

```
[CCM_SYSTEM]
   [ROOT_USER]ccm_root[/ROOT_USER]
[/CCM_SYSTEM]
```

[SEARCH_INDEX_DIRECTORY]

Allows multiple Change servers configured against the same databases to share a search index.

The default value for the search index is the `wsconfig/system` directory. Do not point to another server's `wsconfig/system/index` directory as that will create another index; point to its `wsconfig/system` directory or point all servers to another directory outside of the Change directory hierarchy.

Set the search index directory as follows:

```
[CCM_SYSTEM]
   [SEARCH_INDEX_DIRECTORY]c:\shark\Change 52index[/
SEARCH_INDEX_DIRECTORY]
[/CCM_SYSTEM]
```

[SEARCH_INDEX_TIMER]

Specifies the search indexing timer, in minutes.

Set the search indexing timer as follows:

```
[SEARCH_INDEX_TIMER]your_value[/SEARCH_INDEX_TIMER]
```

For example, set the search indexing timer to update the search index every two hours, as follows:

```
[CCM_SYSTEM]
   [SEARCH_INDEX_TIMER]120[/SEARCH_INDEX_TIMER]
[/CCM_SYSTEM]
```

[SEARCH_MAX_RESULTS]

Specifies the maximum number of search results returned by the server.

Set the search results as follows:

```
[SEARCH_INDEX_TIMER]your_value[/SEARCH_INDEX_TIMER]
```

For example, set the maximum results to return at 100, as follows:

```
[CCM_SYSTEM]
   [SEARCH_MAX_RESULTS]100[/SEARCH_MAX_RESULTS]
[/CCM_SYSTEM]
```

[SEND_MAIL]

Determines how email submissions are processed.

When set to `FALSE`, the change request is appended to *`mail_file_path_and_name`* instead of being sent immediately to the change request tracking address, *`probtrac_email_address`*. The file at *`mail_file_path_and_name`* can then be processed using the Administration dialog **Process Email Submissions** button.

Uses parameters such as the following to send the change request using email:

```
[CCM_SYSTEM]
[SEND_MAIL]true[/SEND_MAIL]
   [MAIL_ADDR]probtrac_email_address[/MAIL_ADDR]
   [MAIL_PROTOCOL]mail_protocol[/MAIL_PROTOCOL]
   [MAIL_SERVER]your_server[/MAIL_SERVER]
[/CCM_SYSTEM]
```

Uses parameters such as the following to append the change request to a file that can be processed later:

```
[CCM_SYSTEM]
   [SEND_MAIL]false[/SEND_MAIL]
   [MAIL_FILE]mail_file_path_and_name[/MAIL_FILE]
[/CCM_SYSTEM]
```

[SESSION_MANAGER_TIMER]

Specifies how often the session pool is resized. The value is in minutes and should be a positive integer. A system with a stable load should use a high value (perhaps 10), while one with an erratic load should use a low value (such as 1).

For example, to resize the session pool every three minutes, use the following entry:

```
[CCM_SYSTEM]
   [SESSION_MANAGER_TIMER]3[/SESSION_MANAGER_TIMER]
[/CCM_SYSTEM]
```

[SESSION_RATIO]

Specifies the number of users per session on the specified database. The value should be a positive integer. With a low number, there is less contention for sessions but more system resources are required.

This tag is only used inside the [CCM_DATABASE] tags.

[SHOW_DATE_FORMAT]

Specifies the format for all date attributes shown on forms (except on reports and the transition log).

Specify the format as follows:

```
[CCM_SYSTEM]
   [SHOW_DATE_FORMAT]date_format[/SHOW_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`05/10/2006 12:22 PM`" on show forms, specify the following format:

```
[CCM_SYSTEM]
   [SHOW_DATE_FORMAT]MM/dd/yyyy h:mm a[/SHOW_DATE_FORMAT]
[/CCM_SYSTEM]
```

See [ACCENT_JAVA_DATE_FORMAT] for a list of date formats and their symbols.

[SHOW_MOTD]

Specifies if the message of the day should be diplayed to the user; by default it is not shown.

Also specifies the type of the message, such as. HTML or Text; the default is Text.

Specify the message as follows:

```
[CCM_SYSTEM]
   [SHOW_MOTD]false[/SHOW_MOTD][MOTD_TYPE]text[/MOTD_TYPE]
[/CCM_SYSTEM]
```

[START_OFFLOAD]

Indicates whether the starting and stopping of sessions should be done in a separate thread. The value should be a boolean. Under normal conditions, the value will be `true`.

Specify the indicator as follows:

```
[CCM_SYSTEM]
   [START_OFFLOAD]true|false[/START_OFFLOAD]
[/CCM_SYSTEM]
```

[START_OFFLOAD_WAIT]

Specifies the number of seconds to wait for a thread that is starting or stopping a session. The value should be a non-negative integer, and is only used when `[START_OFFLOAD]` is `true`.

For example, to wait 30 seconds, use the following entry:

```
[CCM_SYSTEM]
   [START_OFFLOAD]30[/START_OFFLOAD]
[/CCM_SYSTEM]
```

[SUBMIT_DATE_FORMAT]

Specifies the format for all date attributes submitted from the browser.

Note The CCM_DATE control (for example, `base.CCM_DATE`) must enforce/be consistent with SUBMIT_DATE_FORMAT.

You should do the following to ensure that dates used on the Submit forms are interpreted correctly by the Web server:

1. Configure your templates to use a uniform date format for change request submissions.
2. Use [SUBMIT_DATE_FORMAT] to notify the Web server of the format.

Specify the format as follows:

```
[CCM_SYSTEM]
   [SUBMIT_DATE_FORMAT]date_format[/SUBMIT_DATE_FORMAT]
[/CCM_SYSTEM]
```

For example, to use a date like "`05/10/2007`" on submit forms, specify the following format:

```
[CCM_SYSTEM]
   [SUBMIT_DATE_FORMAT]MM/dd/yyyy[/SUBMIT_DATE_FORMAT]
[/CCM_SYSTEM]
```

For the possible data formats, see [ACCENT_ORIG_DATE_FORMAT] and [ACCENT_JAVA_DATE_FORMAT].

[SUBREPORT_LEVELS]

Specifies the maximum number of subreport levels allowed in a reports.

To change this value, you must change all reports, including those provided. Contact Support for help with this (see "Contacting IBM Rational Software Support" on page 4).

The default value of [SUBREPORT_LEVELS] is three (3).

[SUPPORT_TASKS]

Specifies whether the task interface is enabled. By default, tasks are supported.

For example, to disable tasks, use the following entry:

```
[CCM_SYSTEM]
   [SUPPORT_TASKS]false[/SUPPORT_TASKS]
[/CCM_SYSTEM]
```

[TASK_ACL_CLASSNAME]

Sets the complete path of a Java class to run in order to evaluate ACL rules for tasks.

For example:

```
[CCM_SYSTEM]
   [TASK_ACL_CLASSNAME]com.abcd.cs.acl.AclEvaluator
   [/TASK_ACL_CLASSNAME]
[CCM_SYSTEM]
```

[THRESHOLD]

Specifies a host threshold. The value should be a non-negative integer with units in number of sessions.

After a host reaches the threshold, sessions can be started on lower priority hosts. No additional sessions are started on a host that has reached threshold until all other hosts have also reached the threshold. In this way, hosts with a high priority need not reach maximum capacity before low priority hosts are used. The threshold is important because a host performance might be degraded before the maximum number of sessions is running. The threshold marks the degradation point and ensures that other hosts are used first, if possible.

This tag is only used inside the [CCM_HOST] tags.

[TRIGGER_OUTPUT_IS_UTF8]

Informs the server if the encoding used by the stdout and stderr streams on triggers is UTF-8. If not, the system encoding is assumed. The value is chosen automatically but may be overridden.

For example, use the following when the encoding is UTF-8:

```
[CCM_SYSTEM][TRIGGER_OUTPUT_IS_UTF8]true
[/TRIGGER_OUTPUT_IS_UTF] [/CCM_SYSTEM]
```

[TRANS_SERVER_CONFIG]

Groups the transaction server configuration settings into a logical block.

Although `pt.cfg` entries are normally line-oriented, this tag set spans several lines. The opening tag should be on a line by itself, as should the closing tag.

Transaction server settings were designed to be changed through the GUI (using the **Server** tab in the Admin interface). Although you can edit these settings manually in `pt.cfg`, you should not because it requires Rational Change to be restarted for the new settings to take effect in entirety. When transaction server settings are saved through the GUI, however, the entire [TRANS_SERVER_CONFIG] section is overwritten.

The following tags should be enclosed grouped with the [TRANS_SERVER_CONFIG] tag set: [SESSION_MANAGER_TIMER], [ACTIVE_USER_TIMEOUT], [MAX_SESSION_AGE], [REQUEST_RETRY_COUNT], [CCM_OPTIONAL_START_ARGS], [DEFAULT_DATABASE], [CCM_DATABASE], and [CCM_HOST].

[USER_SCOPE]

Groups the user list into manageable sizes. The user list is presented from the Users dialog in the Admin interface. The setting is maintained from the Users dialog, so there is no need to manually adjust this setting. The [USER_SCOPE] is also presented in the User interface when setting a user's *`Custom User List`*.

Regardless of the prefix values used, the special reserved values "All" and "Other" will always be present (they cannot be deleted or modified). "All" presents the entire, un-segmented user list, while "Other" displays any users not shown in some other segment.

Display the user scope as follows:

```
[CCM_SYSTEM]
    [USER_SCOPE]
        [PREFIXES]All| "pipe delimited list of prefix values"
|Other[/PREFIXES]
        [DEFAULT]the default segmented list/prefix to
display when loading Users dialog[/DEFAULT]
    [/USER_SCOPE]
[/CCM_SYSTEM]
```

[USER_SESSION_TIMEOUT]

Specifies how many minutes users can be inactive before their sessions are timed out. They must then log on again. The value 0 means that user sessions never time out. This is also the default setting.

Admin sessions are not affected by this setting.

Set the user timeout as follows:

```
[CCM_SYSTEM]
    [USER_SESSION_TIMEOUT]number of minutes|0[/
USER_SESSION_TIMEOUT]
[/CCM_SYSTEM]
```

[USER_WINDOW_OPTIONS]

Controls the size and properties of pop-up windows.

See [ADMIN_WINDOW_OPTIONS] for more information.

[UTILITY_WINDOW_OPTIONS]

Controls the size and properties of pop-up windows.

See [ADMIN_WINDOW_OPTIONS] for more information.

[UTILITY_SMALL_WINDOW_OPTIONS]

Controls the size and properties of pop-up windows.

See [ADMIN_WINDOW_OPTIONS] for more information.

[WEB_ROLES]

Lists the available Rational Change roles. Any value in this list must be defined in one of the [ROLE] definitions line in the `pt.cfg` file.

Delimit the list using the defined delimiter ([DELIMITER]). The default delimiter is a pipe (|).

Do not include spaces between entries or more than one delimiter between values.

Refer to the examples in "[/CCM_SYSTEM]" on page 41. Also see "Users, roles, and privileges" on page 21 and "[ROLE]" on page 57.

USER PROFILE SECTION

Defines data about individual users. Rational Change uses this data to populate the Submit dialog Submitter Information fields with values for the current user.

All definitions in this section are enclosed in the following tag set:

```
[CCM_USER_PROFILE]...[/CCM_USER_PROFILE]
```

Tags summary

The following table shows the tags used in the `USER PROFILE SECTION`.

Tag	Description
[DATABASE_SPECIFIC]	When set to `TRUE`, requested user profile data is from the current database.
[NAME]	Specifies the name of this user profile tag.
[TYPE]	Specifies the Rational Change Web type used to display a tag.

[DATABASE_SPECIFIC]

When set to `TRUE`, the requested user profile data is specific to the current database. When set to `FALSE` (the default), the requested user profile data is not specific to a database.

For example, [DATABASE_SPECIFIC] would probably be `TRUE` for a product name and `FALSE` for a user ID.

Use the tag as follows:

```
[CCM_USER_PROFILE][NAME]user_profile_item[/NAME]
    [TYPE]item_type[/TYPE]
    [DATABASE_SPECIFIC]{TRUE|FALSE}[/DATABASE_SPECIFIC]
[/CCM_USER_PROFILE]
```

In this case, the database path is appended to the profile tag name in the user configuration file (*`user_name`*`.cfg`).

For example, make the user's default report database specific as follows:

```
[CCM_USER_PROFILE][NAME]user_default_report[/NAME]
    [TYPE]CCM_STRING[/TYPE]
    [DATABASE_SPECIFIC]true[/DATABASE_SPECIFIC]
[/CCM_USER_PROFILE]
```

[NAME]

Specifies the name of this user profile tag.

Note If you use an existing name, overwrite the existing name value.

[TYPE]

Specifies the Rational Change Web type used to display a tag. Available types are as shown in the following table.

Name of Data Type	**Description**
`CCM_DATE`	Specifies a date text area.
`CCM_LISTBOX`	Specifies a drop-down list box.
`CCM_READONLY`	Specifies a read-only text field.
`CCM_STRING`	Specifies a single-line text field.
`CCM_TEXT`	Specifies a multi-line text area.
`CCM_TOGGLE`	Specifies a toggle.
`CCM_VALUELISTBOX`	Specifies a drop-down list box in which the label (what you see in the list box) and the value (what is stored in the *`user_name`*`.cfg` file) are not the same.

ATTRIBUTE NAME/DATATYPE SECTION

Defines all attributes (product, version, hardware, so on) used by Rational Change.

All definitions in this section are enclosed in the following tag set:

```
[CCM_ATTRIBUTE]...[/CCM_ATTRIBUTE]
```

Tags summary

The following table shows tags used in the `ATTRIBUTE NAME/DATATYPE SECTION`.

Tag	Description
[ALIAS]	Allows role-based attribute aliases.
[NAME]	Specifies the change request attribute name.
[role_name]	Specifies the role for which an alias is set.
[TYPE]	Specifies the Rational Change Web data type.

[ALIAS]

Defines the default attribute label you can override with role-based attribute aliases.

The [ALIAS][/ALIAS] tag set defines the default value in a configuration file. If [ALIAS][/ALIAS] is omitted, the value within the [NAME][/NAME] tag set is returned.

[NAME]

Specifies the change request attribute name, and must be identical to the corresponding change request attribute in Rational Synergy.

Note If you use an existing name, overwrite the existing name value.

[role_name]

Specifies one or more roles that make an alias role-specific. You can list any valid Web role, as listed in the [ROLE] tags in the `pt.cfg` file (for example, *Admin* or *User*).

[TYPE]

Specifies the Rational Change Web data type. See “[TYPE]” on page 67.

Examples

- Replace the default Change Request Process image with a new image.

  Update the image entry.

  ```
  [CCM_ATTRIBUTE]
      [NAME]ChangeRequestProcessImage[/NAME]
      [ALIAS]my_cr_process.gif[/ALIAS]
      [TYPE]CCM_STRING[/TYPE]
  [/CCM_ATTRIBUTE]
  ```

  The `intro_lifecycle_dflt.html` (Welcome pages) file references the image as follows:

  ```
  <TABLE WIDTH='750' CELLSPACING='0' CELLPADDING='0'>
      <TR>
      <TD ALIGN=center>
      <IMG SRC='/trapeze/ptimages/<!-- WSLET CODE=PTString
      -->
      <!-- PARAM NAME=CCM_STRING
      VALUE=ChangeRequestProcessImage -->
      <!-- /WSLET -->' USEMAP='#wcs_intro' BORDER=0>
      </TD>
      </TR>
  </TABLE>
  ```

- Set a default alias and a role-based alias.

  Make Change Request Number an alias for `Problem_Identifier_ID`. If the login role is *CRUser*, use Change Request ID for the alias, instead.

  a. Create the configuration entry.

  ```
  [CCM_ATTRIBUTE]
  [NAME]Problem_Identifier_ID[/NAME]
  [ALIAS]Change Request Number[/ALIAS]
  [TYPE]CCM_STRING[/TYPE]
  [/CCM_ATTRIBUTE]
  ```

  b. Create the `cruser.cfg` entry.

  ```
  [CCM_ATTRIBUTE]
  [NAME]Problem_Identifier_ID[/NAME]
  [CRUser]Change Request ID[/CRUser]
  [/CCM_ATTRIBUTE]
  ```

c. Use the wslet call to set the alias.

```
        <!-- WSLET CODE=PTString -->
        <!-- PARAM NAME=CCM_STRING VALUE=Problem_Identifier_ID --
>
        <!-- /WSLET -->
```

Resulting HTML for *CRUser*:

```
        Change Request ID
```

Resulting HTML for non-*CRUser*:

```
        Change Request Number
```

OBJECT REPORT SECTION

Defines reports that show information about objects. Object reports can stand alone, but they are often included in other reports.

Names starting with "`data_report`" are reports used only with Rational Change API reporting methods. These reports do not need to have any templates defined for them, as a data-based API report does not need templates.

All definitions in this section are enclosed in the following tag set:

```
[CCM_OBJECT]...[/CCM_OBJECT]
```

Note All tags used in the `OBJECT REPORT SECTION` are also used in the `TASK REPORT SECTION` and `PROBLEM REPORT SECTION`.

Tags summary

The following table shows the tags used in the `OBJECT REPORT SECTION`.

Tag	Description
[ATTR_TEMPLATE]	Specifies the attribute template file.
[ATTRS]	Specifies the list of change request, task, or object attributes to include in the report.
[AUTO_ATTR_TEMPLATE]	For auto-generated reports, specifies the template used to present one attribute value/attribute label pair.
[AUTO_LABEL_TEMPLATE]	For auto-generated reports, specifies the attribute label template file.
[CUSTOM_WSLET]	Specifies a custom wslet to run for a subreport.
[EXPORT_FORMAT]	Specifies the file extension for report results.
[FTR_TEMPLATE]	Specifies the footer template file.
[GROUP_BY]	Groups change requests, tasks, or objects.
[GROUP_TEMPLATE]	Specifies the attribute grouping template file.
[HDR_TEMPLATE]	Specifies the header template file.
[IMG_TEMPLATE]	Specifies the image (chart) template file.
[LABEL_TEMPLATE]	Specifies the label template file.

Tag	Description
[MAIN_TEMPLATE]	Specifies the main template file.
[NAME]	Specifies the name of this object, task, or change request report tag.
[SORT_ORDER]	Sorts attributes.
[SPAN_ATTR_TEMPLATE]	Specifies the span attribute template file.
[XML_CONTENT]	Specifies XML data to pass to a custom wslet.

[ATTR_TEMPLATE]

Specifies an attribute template file. This template defines how the details (attributes) for a single change request, task, or object is displayed in the report.

This template is used many times: once for each change request, task, or object reported.

[ATTRS]

Specifies the list of change request, task, or object attributes to include in the report.

This tag requires a delimited list.

Do not include spaces between entries or more than one delimiter between values.

The general syntax for each listed attribute is as follows:

```
attribute_name:sort_position:span_option
```

The *sort_position* and *span_option* parameters are optional, and are used in custom reports. *sort_position* is the position of the attribute on the form (left to right, then top to bottom), and *span_option* is the binary indicator of whether the attribute spans columns.

For example, if the following attribute list is defined for a report, `problem_number` and `crstatus` are displayed in the first row, and `problem_synopsis` spans the second row:

```
[ATTRS]problem_number:0:false|crstatus:1:false|
problem_synopsis:2:true[/ATTRS]
```

Also see [AUTO_ATTR_TEMPLATE] and [SPAN_ATTR_TEMPLATE].

[AUTO_ATTR_TEMPLATE]

For auto-generated reports, specifies a template used to present one attribute value/attribute label pair.

Rational Change uses the value of this tag if an attribute `span_option` is set to `false`.

Also see [ATTRS] and [SPAN_ATTR_TEMPLATE].

[AUTO_LABEL_TEMPLATE]

For auto-generated reports, specifies an attribute label template file for custom, column-formatted reports. This template enables you to label a column that contains multiple values instead of formatting the report with label and value pairs.

Note You can use this template only for single-level, column-based reports.

[CUSTOM_WSLET]

Specifies a custom wslet to run for a subreport. The custom wslet has access to the subreport query results and [XML_CONTENT], if any.

The custom wslet is run indirectly by the CSChartDispatcher wslet. See the example in “CSChartDispatcher” on page 243.

[EXPORT_FORMAT]

Specifies the file extension for report results. The default is `html`.

Specify the HTML format as follows:

```
[EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
```

Specify the non-HTML format as follows:

```
[EXPORT_FORMAT]format_file_extension[/EXPORT_FORMAT]
```

The generated report is saved as a file named "*report_name.export_format*," then is downloaded to your browser using the MIME type declaration. You can either open the document directly or save the file to disk.

For example, specify an Excel file format as follows:

```
[EXPORT_FORMAT]xls[/EXPORT_FORMAT]
```

The file is saved as *report_name*.xls.

Note All files are transferred as binary files.

The following example shows how to define a report exported to a non-HTML format (Excel); that is, how to create a change request report that is loaded into Excel as a tab-delimited text file with an `.xls` file extension. From this definition set, the name of the file downloaded to the browser is "`All change_requests.xls`".

Change Request Report Definition:

```
[CCM_PROBLEM]
   [NAME]prob_excel[/NAME]
   [MAIN_TEMPLATE]excel_prob_rpt.xls[/MAIN_TEMPLATE]
   [HDR_TEMPLATE]excel_prob_hdr.xls[/HDR_TEMPLATE]
   [ATTR_TEMPLATE]excel_prob_attr.xls[/ATTR_TEMPLATE]
   [ATTRS]problem_number|status|release|resolver|severity
   [/ATTRS]
[/CCM_PROBLEM]
```

Report Definition:

```
[CCM_REPORT]
   [NAME]All change requests[/NAME]
   [QUERY]All change requests[/QUERY]
   [PROBLEM_DEF]prob_excel[/PROBLEM_DEF]
   [EXPORT_FORMAT]xls[/EXPORT_FORMAT]
   [DESCRIPTION]
   Show all change requests in the database as a excel
   spreadsheet.
   [/DESCRIPTION]
[/CCM_REPORT]
```

Report: `excel_prob_rpt.xls`

```
<!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_HEADER -->
<!-- /WSLET -->
<!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_ATTRS -->
<!-- /WSLET -->
```

Header: `excel_prob_hdr.xls`

```
Change Request
Number"tab_char"Status"tab_char"Release"tab_char"Resolver"ta
b_char"
Severity
```

Attributes: `excel_prob_attr.xls`

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=problem_number -->
<!-- /WSLET -->"tab_char"
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=status -->
<!-- /WSLET -->"tab_char"
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=release -->
<!-- /WSLET -->"tab_char"
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=resolver -->
<!-- /WSLET -->"tab_char"
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=severity -->
<!-- /WSLET -->
```

Note You can combine reports to show a combination of object, task, and change request information.See "Examples" on page 83.

[FTR_TEMPLATE]

Specifies a footer template file.

[GROUP_BY]

Groups change requests, tasks, or objects that have the same attribute name, as follows:

```
[GROUP_BY]attribute_name:sort_type:sort_direction|...[/
GROUP_BY]
```

Note Do not use the same *`attribute_name`* here that you are using in [SORT_ORDER].

In a grouping template, the REPORT_GROUP_BY wslet call returns dates in appropriate units when the [GROUP_BY] tag is in years, months, days, hours, or minutes:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_GROUP_BY -->
<!-- /WSLET -->
```

See [SORT_ORDER] for parameter details.

[GROUP_TEMPLATE]

Specifies an attribute grouping template file. Used with [GROUP_BY]. The grouping template also allows for charting based on the grouped change requests, tasks, or objects.

[HDR_TEMPLATE]

Specifies a header template file.

[IMG_TEMPLATE]

Specifies an image (chart) template file. Charts in this section chart attributes that span the entire set of change requests, tasks, or objects. Use the grouping charts if you want charting for groups of change requests, tasks, or objects.

[LABEL_TEMPLATE]

Specifies a label template file for column-formatted reports. This template enables you to label a column that contains multiple values instead of formatting the report with label and value pairs.

Note You can use this template only for single-level reports.

[MAIN_TEMPLATE]

Specifies a Main template file. The file can contain wslets to include the following templates:

- `REPORT_HEADER`, which defines the report header.
- `REPORT_ATTRS`, which defines how the change request, task, or object attributes are displayed.
- `REPORT_LABELS,` which defines a column label.
- `REPORT_IMAGE`, which defines any charts that are included.
- `REPORT_FOOTER,` which defines the report footer.

[NAME]

Specifies the name of this object, task, or change request report tag.

Note If you use an existing name, overwrite the existing name value.

[SORT_ORDER]

Sorts attributes as follows:

```
[SORT_ORDER]attribute_name:sort_type:sort_direction|...[/
SORT_ORDER]
```

The parameters for this tag are as follows:

- `attribute_name`
- `sort_type`
- `sort_direction`

For example:

```
[SORT_ORDER]severity:listbox:A[/SORT_ORDER]
```

`attribute_name`

Specifies the name of the attribute by which to group change requests, tasks, or objects.

`sort_type`

Specifies the sort type. The type is one of the types lists below. The default is `string`.

int

Specifies an integer type.

The `int*` data types (`intf`, `intfm`, `intb`, and `intbm`) are special cases of `int`. These types are strings containing an embedded integer. The embedded integer is extracted and used for the sort comparison.

— **`intf`**

Parse the string in the forward direction looking for a contiguous block of digits. The first character must be a digit. Parsing stops when a character is encountered or the end of the string is reached.

— **`intfm`**

Parse the string in the forward direction looking for a contiguous block of digits somewhere in the string. Parsing starts when a digit is found and stops when a character is encountered or the end of the string is reached.

— **`intb`**

Parse the string in the backward direction looking for a contiguous block of digits. The last character must be a digit. Parsing stops when a character is encountered or the beginning of the string is reached.

— **`intbm`**

Parse the string in the backward direction looking for a contiguous block of digits somewhere in the string. Parsing starts when a digit is found and stops when a character is encountered or the beginning of the string is reached.

float

Specifies a floating-point type (decimal).

date

Specifies a date type (seconds).

date_units

Specifies a date units type. The possible *`date_units`* values are as follows:

`date`: seconds

`days`: month, day, year (for example, May 22, 2001)

`hours`: month, day, year, and hours (for example, 12/22/2000 16)

`minutes`: month, day, year, hours, and minutes (for example, 12/22/2000 15:07)

`months`: month in string form and year (for example, April, 2001)

`quarters`: number of the quarter (1 - 4).

`seconds`: seconds

`weeks`: week of the year (1 - 52)

`years`: four-digit year (for example, 2001)

listbox

Specifies a list box type. To use this type, you must ensure that the *`attribute_name`* has a list box defined for it.

The list order is from left to right, in the order defined for the list box.

string

Specifies a string type.

sort_direction

Specifies the grouping sort direction: A (ascending) or D (descending). The default is "A."

Note If you specify a *sort_direction*, you must specify *sort_type*, as well, even if the sort type is `string` (the default).

For example, the following sort order definition sorts problem numbers in ascending order, starting at the end of the `problem_number` string:

```
[SORT_ORDER]problem_number:intb:A[/SORT_ORDER]
```

[SPAN_ATTR_TEMPLATE]

For reports, specifies a custom report template file that allows a label/attribute to span an entire row.

Rational Change uses the value of this tag if an attribute *span_option* is set to `TRUE`.

Report templates are row-oriented. By default, two labels/attributes occupy one row (one column for each label, and one column for each value, for a total of four columns). The span attribute template, however, allows the label/attribute to occupy an entire row (one column for the label, and three columns for the value).

For example, in the following excerpt (for a change request/problem subreport), the `custom_span_attr.html` span attribute template is specified, and is used for the `problem_synopsis` attribute:

```
[CCM_PROBLEM]
    [NAME]custom_prob[/NAME]
    [MAIN_TEMPLATE]user_framework/common_rpt.html[/
MAIN_TEMPLATE]
    [HDR_TEMPLATE]user_framework/common_hdr.html[/HDR_TEMPLATE]
    [ATTR_TEMPLATE]user_framework/custom_attr.html[/
ATTR_TEMPLATE]
    [SPAN_ATTR_TEMPLATE]user_framework/custom_span_attr.html
    [/SPAN_ATTR_TEMPLATE]
    [AUTO_ATTR_TEMPLATE]user_framework/custom_auto_attr.html
    [/AUTO_ATTR_TEMPLATE]
    [FTR_TEMPLATE]user_framework/common_ftr.html[/FTR_TEMPLATE]

[ATTRS]problem_number:0:false|crstatus:1:false|problem_synopsis
:2:true
    [/ATTRS]
    [SORT_ORDER]problem_number:intb:A[/SORT_ORDER]
[/CCM_PROBLEM]
```

The file specified by [ATTR_TEMPLATE] (`custom_attr.html`) is read for each attribute:

```
<BR>
<TABLE WIDTH=650 CELLSPACING=0 CELLPADDING=2 BORDER=1>
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_AUTO_GENERATE -->
    <!-- /WSLET -->
</TABLE>
```

PTReport performs a "report auto generate" using either AUTO_ATTR_TEMPLATE or SPAN_ATTR_TEMPLATE, depending on whether the attribute *`span_option`* is set to `true` or `false`. In this example, *`span_option`* is `false` for `problem_number` and `crstatus`, so `custom_auto_attr.html` is read for them. However, *`span_option`* is `true` for `problem_syopsis`, so `custom_span_attr.html` is read for it.

The `custom_span_attr.html` file might look like the following, where the label occupies the first column in the row, and the value occupies the remaining three columns:

```
<TR>
    <TD ALIGN=left VALIGN=center>
        <!-- WSLET CODE=PTReport -->
        <!-- PARAM NAME=REPORT_SPAN_ATTR_NAME -->
        <!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->
        <!-- /WSLET -->
    </TD>
    <TD COLSPAN=3 ALIGN=left VALIGN=center>
        <B>
        <!-- WSLET CODE=PTReport -->
        <!-- PARAM NAME=REPORT_SPAN_ATTR_VALUE -->
        <!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->
        <!-- PARAM NAME=AUTO_ATTR_REPORT_LOG VALUE=transition_log
-->
        <!-- PARAM NAME=TRANSITION_FONT VALUE="COLOR=purple" -->
        <!-- PARAM NAME=MODIFY_FONT VALUE="COLOR=red" -->
        <!-- PARAM NAME=NOTE_FONT VALUE="COLOR=blue" -->
        <!-- PARAM NAME=SHOW_TRANSITION_COMMENTS VALUE=true -->
        <!-- PARAM NAME=SHOW_MODIFY_EVENTS VALUE=true -->
        <!-- PARAM NAME=SHOW_NOTES VALUE=true -->
        <!-- /WSLET -->
        </B>
    </TD>
</TR>
```

Also see [ATTRS] and [AUTO_ATTR_TEMPLATE].

[XML_CONTENT]

Specifies XML data to pass to a custom wslet. The custom wslet must interpret the XML data.

TASK REPORT SECTION

Defines reports that show information about tasks. Task reports can stand alone, but they are often included in other reports.

Names starting with "`data_report`" are reports used only with Rational Change API reporting methods. These reports do not need to have any templates defined for them, as a data-based API report does not need templates.

All definitions in this section are enclosed in the following tag set:

```
[CCM_TASK]...[/CCM_TASK]
```

Tags summary

The tags you can use inside the [CCM_TASK] tags are identical to those used in the "OBJECT REPORT SECTION" on page 71.

Example

The following is a task report definition in a configuration file:

```
[CCM_TASK]
    [NAME]task_summary[/NAME]
    [MAIN_TEMPLATE][/MAIN_TEMPLATE]
    [HDR_TEMPLATE][/HDR_TEMPLATE]
    [ATTR_TEMPLATE]summary_task_attr.html[/ATTR_TEMPLATE]
    [IMG_TEMPLATE][/IMG_TEMPLATE]
    [GROUP_TEMPLATE][/GROUP_TEMPLATE]
    [ATTRS]task_number|status|release|resolver|est_duration[/
ATTRS]
    [GROUP_BY][/GROUP_BY]
    [AUTO_ATTR_TEMPLATE][/AUTO_ATTR_TEMPLATE]
    [FTR_TEMPLATE][/FTR_TEMPLATE]
    [SORT_ORDER][/SORT_ORDER]
[/CCM_TASK]
```

Note You can combine reports to show a combination of object, task, and change request information. See "Examples" on page 83.

PROBLEM REPORT SECTION

Defines reports that show information about change requests. Change request reports can stand alone, but they are often included in other reports.

Names starting with "`data_report`" are reports used only with Rational Change API reporting methods. These reports do not need to have any templates defined for them, as a data-based API report does not need templates.

All definitions in this section are enclosed in the following tag set:

```
[CCM_PROBLEM]...[/CCM_PROBLEM]
```

Tags summary

The tags you can use inside the `CCM_PROBLEM` tags are identical to those used in the “OBJECT REPORT SECTION” on page 71.

Examples

The following examples show groupings for the `PROBLEM REPORT SECTION`.

- Group by date units.

  Group `entry_date` by date units (`months`) using ascending sort order (`A`). Sort `release` by the default type and sort order, and sort `problem_number` by data type `intb`, using ascending sort order (`A`).

  ```
  [CCM_PROBLEM]
      [NAME]prob_summary[/NAME]
      [MAIN_TEMPLATE]summary_prob_rpt.html[/MAIN_TEMPLATE]
      [HDR_TEMPLATE]common_hdr.html[/HDR_TEMPLATE]
      [ATTR_TEMPLATE]summary_prob_attr.html[/ATTR_TEMPLATE]
      [FTR_TEMPLATE]common_ftr.html[/FTR_TEMPLATE]
      [GROUP_TEMPLATE]summary_prob_group_attr.html
      [/GROUP_TEMPLATE]
      [ATTRS]problem_number|problem_synopsis|status|release
      |resolver|severity|entry_date[/ATTRS]
      [GROUP_BY]entry_date:months:A[/GROUP_BY]
      [SORT_ORDER]release|problem_number:intb:A[/SORT_ORDER]
  [/CCM_PROBLEM]
  ```

- Group by date type.

  Group `resolver` by data type `string`, using descending sort order (`D`). Sort `problem_number` by data type `int`, using the default sort order (ascending); sort release using the default type and sort order; and sort `entry_date` by data type `date` (second), using descending sort order (`D`).

```
    [CCM_PROBLEM]
        [NAME]prob_summary[/NAME]
        [MAIN_TEMPLATE]summary_prob_rpt.html[/MAIN_TEMPLATE]
        [HDR_TEMPLATE]summary_prob_hdr.html[/HDR_TEMPLATE]
        [ATTR_TEMPLATE]summary_prob_attr.html[/ATTR_TEMPLATE]
        [FTR_TEMPLATE]summary_prob_ftr.html[/FTR_TEMPLATE]
        [GROUP_TEMPLATE]summary_prob_group_attr.html
        [/GROUP_TEMPLATE]

[ATTRS]problem_number|status|release|resolver|severity|entry_da
te[/ATTRS]
        [GROUP_BY]resolver:string:D[/GROUP_BY]
        [SORT_ORDER]problem_number:int|release|entry_date:date:D
        [/SORT_ORDER]
    [/CCM_PROBLEM]
```

You also can combine reports to show a combination of object, task, and change request information.The following example shows the default `Change Request - summary with task and object` report, which combines change request, task, and object information.

```
[CCM_REPORT]
    [NAME]Change Request - summary with task and object[/NAME]
    [QUERY][/QUERY]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [TASK_DEF]task_summary[/TASK_DEF]
    [OBJECT_DEF]obj_summary[/OBJECT_DEF]
    [DESCRIPTION]Shows important change request attributes, the
    associated tasks, and associated objects. Problems are
grouped
    by resolver, sorted by release and chart on status.
    [/DESCRIPTION]
[/CCM_REPORT]
```

QUERY DEFINITION SECTION

Defines named queries available in Rational Change.

All definitions in this section are enclosed in the following tag set:

```
[CCM_QUERY]...[/CCM_QUERY]
```

Tags summary

The following table shows the tags used in the `OBJECT REPORT SECTION`.

Tag	Description
[DATE_LAST_RUN]	Replaces the `%date_last_run` query string keyword.
[DESCRIPTION]	Describes the query.
[NAME]	Specifies the name of this user profile tag.
[QRY_STRING]	Specifies a Rational Synergy-compliant query string.
[QRY_TEMPLATE]	Specifies a custom query HTML template file for the specified query.

[DATE_LAST_RUN]

Replaces the `%date_last_run` query string keyword.

This tag has meaning only in user-defined queries.

[DESCRIPTION]

Describes the query.

[NAME]

Specifies the name of this named query, as it would appear to the user in a list box.

Note If you use an existing name, overwrite the existing name value.

[QRY_STRING]

Specifies a Rational Synergy-compliant query string.

[QRY_TEMPLATE]

Specifies a custom query HTML template file for the specified query.

When a user selects the specified query in a list box, Rational Change loads this template.

Example

Define the `All change requests concluded by me` query.

```
[CCM_QUERY]
    [NAME]All change requests concluded by me[/NAME]
    [QRY_STRING]concluder='%username' and status='concluded'
    [/QRY_STRING]
    [QRY_TEMPLATE][/QRY_TEMPLATE]
[/CCM_QUERY]
```

RELATIONSHIP REPORT DEFINITION SECTION

Defines the named reports and report formats available in Rational Change. A report format defines the layout in which query results are displayed. A report combines a *named report format* with a *named query.*

All definitions in this section are enclosed in the following tag set:

```
[CCM_REPORT]...[/CCM_REPORT]
```

Tags Summary

The following table shows the tags used in the `RELATIONSHIP REPORT DEFINITION SECTION`.

Tag	Description
[NAME]	Specifies the name of the report, as it would appear in a list box.
[RPT_TEMPLATE]	Defines a template that gets loaded when this report name is selected from a display list box.
The following tags must be in linear order.	
[QUERY] then [PROBLEM_DEF] or [TASK_DEF] or [OBJECT_DEF] ...or any number of any combination of these.	Specifies a named query for change requests, tasks, or objects.

Tag (Continued)	Description
[RELATION] then [PROBLEM_DEF] or [TASK_DEF] or [OBJECT_DEF]	Specifies which relationship is used for each preceding object.
The following tags need not be in linear order.	
[CUSTOM_DISPLAY_ORDER]	Specifies the order in which reports should be displayed.
[DESCRIPTION]	Describes the report.
[EXPORT_FORMAT]	Specifies the file extension for report results.
[IMAGE_PATH]	Specifies the path to the image that should be shown for the report.
[INCREMENT_SIZE]	Specifies the number of items per page in a paginated report.
[INCREMENTAL]	Specifies if a report is paginated.
[MAX_QUERY]	Specifies the maximum number of items that can be retrieved using the query operation.
[MAX_STRING]	Specifies the maximum number of characters that can be retrieved for an attribute with the TEXT data type in a query operation.
[STYLE]	Specifies the report charting style to be used.

[NAME]

Specifies the name of the report, as it would appear in a list box.

Note If you use an existing name, overwrite the existing name value.

[OBJECT_DEF]

Specifies a named object report. This value must be one of the object reports defined in the OBJECT REPORT SECTION.

Define this tag for a report that is used primarily for objects. You can include problem or task information by defining [PROBLEM_DEF] or [TASK_DEF].

[QUERY]

Specifies a named query. This value must be one of the queries defined in the "QUERY DEFINITION SECTION," page 85.

The query applies only to the specified set of objects; all other results are provided through the [RELATION] definitions.

You must include this tag even if specifying no value.

[PROBLEM_DEF]

Specifies a named change request report. This value must be one of the change request reports defined in the PROBLEM REPORT SECTION.

Define this field for a report that is used primarily for change requests. You can include task or object information by defining [TASK_DEF] or [OBJECT_DEF].

[TASK_DEF]

Specifies a named task report. This value must be one of the task reports defined in the TASK REPORT SECTION.

Define this tag for a report that is used primarily for tasks. You can include problem or object information by defining [PROBLEM_DEF] or [OBJECT_DEF].

[RELATION]

Specifies which relationship to use for a subreport, relative to each preceding object (PROBLEM_DEF, TASK_DEF, or OBJECT_DEF).

Resulting object details are displayed using the change request, task, or object definition. You can use this relation-to-change request|task|object syntax many times to produce complex relationship reports.

For example:

[CCM_REPORT]

```
      [NAME]DRP - Summary with Tasks and Task Folder[/NAME]
      [QUERY]All CRs[/QUERY]
      [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
      [RELATION]associated_task[/RELATION]
      [TASK_DEF]sub_summary_task[/TASK_DEF]
      [RELATION]has_task_in_folder[/RELATION]
      [OBJECT_DEF]drp_sub_summary_folder[/OBJECT_DEF]
      [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
      [MAX_STRING]50[/MAX_STRING]
      [DESCRIPTION]Shows important CR attributes and the
        associated tasks and the task folder. CRs are sorted by
        CR ID[/DESCRIPTION]
   [/CCM_REPORT]
```

Note The definition has a linear order, and you can use only one relationship per object.

Note For example, you cannot generate a report that lists all change requests with both their associated attachments and their associated tasks. Instead, you must use two reports: one that reports on all change requests and their associated attachments, and another that reports on all change requests and their associated tasks.

[RPT_TEMPLATE]

Specifies a template that is loaded when this report name is selected from a display list box, as is done with the query definition. The primary purpose of the report template is to run custom reports.

The Rational Change Administrator can create a variety of custom templates in which the user can select the attributes to display in reports. The Administrator can use the {PROBLEM_ATTRIBUTES|TASK_ATTRIBUTES|OBJECT_ATTRIBUTES} template UI control identifiers (keywords) to display lists of attributes for selection by the user. The Administrator can also force attributes by putting them in the report definition.

[CUSTOM_DISPLAY_ORDER]

Specifies the order in which reports are to be displayed, starting with 0.

[DESCRIPTION]

Describes the report. The user sees this description in the interface.

[EXPORT_FORMAT]

Specifies the file extension for report results. The default is HTML.

See “[EXPORT_FORMAT]” on page 73.

[IMAGE_PATH]

Specifies the path to the image that should be displayed in the report.

[INCREMENT_SIZE]

Specifies the number of items per page in a paginated report. The value is a positive integer.

[INCREMENTAL]

Specifies if the report is paginated or not. The value can be `true` or `false`.

[MAX_QUERY]

Specifies the maximum number of items that can be retrieved using the query operation. This setting overrides the setting by the same name in the `SYSTEM VARIABLES SECTION`.

You might want to set a lower limit for [MAX_QUERY] for detailed reports that retrieve a lot of information about each change request, because the report can take a long time to process.

Note This tag has been relocated from the individual change request/task/object definition sections to this section and is ignored if it is used in the change request/task/object definition section. The system defaults still apply to report results if the tags are omitted.

[MAX_STRING]

Specifies the maximum number of characters that can be retrieved for an attribute with the TEXT data type in a query operation. This setting overrides the setting by the same name in the `SYSTEM VARIABLES SECTION`.

You might want to set a lower limit for [MAX_STRING] in a report that does not need to display lengthy descriptions. For example, the Change Request

Description field can contain stack traces, but you would not want to display the entire trace in a manager's report.

Note This tag has been relocated from the individual change request/task/object definition sections to this section and is ignored if it is used in the change request/task/object definition section. The system defaults still apply to report results if the tags are omitted.

[STYLE]

Specifies the style type of the report, and is currently only being used for chart types (VBarChart, HBarChart, LineChart and PieChart) and matrix style reports.

Examples

The following are examples of tags in the `RELATIONSHIP REPORT DEFINITION SECTION`.

- Show a summary of all change requests in the database.

```
[CCM_REPORT]
    [NAME]Summary of all change requests[/NAME]
    [QUERY]All change requests[/QUERY]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [MAX_QUERY]100[/MAX_QUERY]
    [MAX_STRING]2000[/MAX_STRING]
    [DESCRIPTION]Show all change requests in the database
    [/DESCRIPTION]
[/CCM_REPORT]
```

- Show all change requests in the database with their associated tasks and objects.

```
[CCM_REPORT]
    [NAME]All change requests (with tasks and objects)[/NAME]
    [QUERY]All change requests[/QUERY]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [RELATION]associated_task[/RELATION]
    [TASK_DEF]task_summary[/TASK_DEF]
    [RELATION]associated_cv[/RELATION]
    [OBJECT_DEF]obj_summary[/OBJECT_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [DESCRIPTION]
    Show all change requests in the database with associated
    tasks and objects
    [/DESCRIPTION]
[/CCM_REPORT]
```

- Show all change requests in the database with their associated attachments.

```
[CCM_REPORT]
    [NAME]All change requests with attachments[/NAME]
    [QUERY]All change requests[/QUERY]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [RELATION]attachment[/RELATION]
    [OBJECT_DEF]attachment_details[/OBJECT_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [DESCRIPTION]
    Show all change requests in the database with associated
    attachments
    [/DESCRIPTION]
[/CCM_REPORT]
```

- Show all parent change requests in the database with their sub-change requests.

```
[CCM_REPORT]
    [NAME]All parent change requests with sub change requests
    [/NAME]
    [QUERY]All parent change requests[/QUERY]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [RELATION]sub_problem[/RELATION]
    [PROBLEM_DEF]prob_summary[/PROBLEM_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [DESCRIPTION]
    Show all parent change requests with their sub change
    requests[/DESCRIPTION]
[/CCM_REPORT]
```

- Show the summary report as a paginated report with 20 items per page.

```
[CCM_REPORT]
    [NAME]Basic Summary[/NAME]
    [QUERY]All CRs[/QUERY]
    [PROBLEM_DEF]prob_basicsummary[/PROBLEM_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [INCREMENTAL]true[/INCREMENTAL]
    [INCREMENT_SIZE]20[/INCREMENT_SIZE]
    [DESCRIPTION]List of problem numbers, status and
    synopsis[/DESCRIPTION]
[/CCM_REPORT]
```

- Show the report in a vertical bar chart format.

```
[CCM_REPORT]
    [NAME]Vertical Bar Chart[/NAME]
    [QUERY]All CRs[/QUERY]
    [PROBLEM_DEF]adhocChart[/PROBLEM_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
    [INCREMENTAL]false[/INCREMENTAL]
    [STYLE]VBarChart[/STYLE]
    [DESCRIPTION]Custom report format that allows you to
    select a single attribute to create a vertical bar chart.
    [/DESCRIPTION][
/CCM_REPORT]
```

VALUELISTBOX SECTION

Defines value list boxes put on a form as JavaScript using the PTValueListBox wslet call. A value list box is a list box that displays a set of labels that are mapped to other values. When the user selects a label, the corresponding value is submitted to the Web server.

All definitions in this section are enclosed in the following tag set:

```
[CCM_VALUELISTBOX]...[/CCM_VALUELISTBOX]
```

Tags summary

The following table shows the tags used in the `OBJECT REPORT SECTION`.

Tag	Description
[NAME]	Specifies a name for the list box, usually an attribute name.
[LABELS]	Specifies a delimited list of labels (strings) that will appear in the list box.
[VALUES]	Specifies the delimited list of values (strings) that are submitted to Rational Synergy.

[NAME]

Specifies a name for the list box, usually an attribute name.

Note If you use an existing name, overwrite the existing name value.

[LABELS]

Specifies a delimited list of labels (strings) that will appear in the list box.

Do not include spaces between entries or more than one delimiter between values.

[VALUES]

Specifies the delimited list of values (strings) that will appear in the list. You must define each listed value.

Do not include spaces between entries or more than one delimiter between values.

Example

The following CCM_VALUELISTBOX definition shows how to put `num_test` on the form.

Configuration file entry:

```
[CCM_VALUELISTBOX]
   [NAME]num_test[/NAME]
   [LABELS]a|b|c|d|e|f[/LABELS]
   [VALUES]101|233|444|678|221|555[/VALUES]
[/CCM_VALUELISTBOX]
```

Wslet call:

```
<!-- WSLET CODE=PTValueListBox -->
<!-- PARAM NAME=CCM_VALUELISTBOX VALUE="num_test" -->
<!-- /WSLET -->
```

When passed to the Web server by PTValueListBox and processed, the configuration file definition produces the following HTML:

```
<OPTION VALUE=101>a</OPTION>
<OPTION VALUE=233>b</OPTION>
<OPTION VALUE=444>c</OPTION>
<OPTION VALUE=678>d</OPTION>
<OPTION VALUE=221>e</OPTION>
<OPTION VALUE=555>f</OPTION>
```

LISTBOX SECTION

Defines list boxes put on a form as JavaScript using the PTListBox wslet call.

The `pt_listbox.cfg` file defines some list boxes and most list box values for the user interface. Template files access the data items defined in `pt_listbox.cfg` by means of wslets.

Note The [SUBLISTBOX] tag sets and [SUBLISTBOX] list box types are set in `pt_listbox.cfg` automatically using the **Listboxes** tab in the GUI, when the Administrator sets the [DEPENDENT_ATTR] tag. See Rational Change online help for the *Admin* role for how to set values for dependent list boxes.

All definitions in this section are enclosed in the following tag set:

```
[CCM_LISTBOX]...[/CCM_LISTBOX]
```

Tags summary

The following table shows the tags used in `LISTBOX SECTION`.

Tag	Description
[DEPENDENT_ATTR]	Specifies a dependent (child) attribute for a list box.
[FILE]	Specifies a file from which to get list box values.
[FROM_DATABASE]	Specifies a list box with values obtained from the Rational Synergy database.
[NAME]	Specifies the name for the list box, usually an attribute name.
[SUBLISTBOX]	Specifies the variables that contain values for each parent attribute choice.
[TYPE]	Specifies the type of list box.
[VALUES]	Contains the delimited list of values (strings) that will appear in the list box.

[DEPENDENT_ATTR]

Specifies a delimited list of child attributes that depend on the value of a parent attribute, as follows:

```
[DEPENDENT_ATTR]attribute_name|...[/DEPENDENT_ATTR]
```

A [DEPENDENT_ATTR] (child attribute) has a [SUBLISTBOX] entry for each value of its parent attribute. The [DEPENDENT_ATTR] list of values changes whenever a new value for the parent is selected.

A parent attribute can have many dependent attributes, but a child attribute can have only one parent attribute.

For example, perform the following steps to define product version values that depend on the product name, in the `pt_listbox.cfg` file:

1. Define product_name, with `product_version` (and `product_subsys`) as dependent attributes.

   ```
   [CCM_LISTBOX]
       [NAME]product_name[/NAME]
       [TYPE]ATTRIBUTE[/TYPE]
       [VALUES]Any|Rational Change|Continuus/CM[/VALUES]
       [DEPENDENT_ATTR]product_subsys|product_version
       [/DEPENDENT_ATTR]
   [/CCM_LISTBOX]
   ```

2. Define the `product_version` values.

   The [VALUES] tags must contain all *possible* values.

   The [SUBLISTBOX] tags contain sublists that define values appropriate for each value of the parent attribute (`product_name`); for example, `PRODUCT_VERSION_SYNERGY/Change` maps to `Rational Change` for the `product_name`.

   ```
       [CCM_LISTBOX]
           [NAME]product_version[/NAME]
           [TYPE]ATTRIBUTE[/TYPE]
           [VALUES]Any|2.0|3.0|3.5|4.0|4.5|5.0|6.0|6.0.1|6.1[/
   VALUES]
           [SUBLISTBOX]PRODUCT_VERSION_ANY|PRODUCT_VERSION_SYNERGY/
   Change|
          PRODUCT_VERSION_CONTINUUS/CM[/SUBLISTBOX]
       [/CCM_LISTBOX]
   ```

3. Define the sublist box values.

   For example, `PRODUCT_VERSION_SYNERGY/Change` must list the valid product versions for Rational Change, as follows:

   ```
   [CCM_LISTBOX]
       [NAME]PRODUCT_VERSION_SYNERGY/Change[/NAME]
       [TYPE]SUBLISTBOX[/TYPE]
   ```

```
        [VALUES]Any|2.0|3.0|3.5|4.0[/VALUES]
    [/CCM_LISTBOX]
```

Note that this list is type `SUBLISTBOX`, not `ATTRIBUTE`.

[FILE]

Specifies a file from which to get list box values, as follows:

```
[FILE]file_name[/FILE]
```

The file must be in the installation `etc` directory.

[FROM_DATABASE]

Specifies a list box with values obtained from the Rational Synergy database, as follows.

```
[FROM_DATABASE]list_name[/FROM_DATABASE]
```

You can load the user list and the releases list from the Rational Synergy database.

The value between the [FROM_DATABASE] tags references a [PARAMETER] definition of type `database` or `model` in the `ptcli.cfg` file. For example, the releases list box obtains its values from the `active_releases` database. Therefore, in the `ptcli.cfg` file, the following definition controls where to get the list of values (a `model` function with the parameter `active_releases`):

```
[PARAMETER]
    [TYPE]model[/TYPE]
    [NAME]active_releases[/NAME]
    [ARG1]active_releases[/ARG1]
[/PARAMETER]
```

[NAME]

Specifies the name for the list box, usually an attribute name.

Note If you use an existing name, overwrite the existing name value.

[SUBLISTBOX]

Specifies additional list boxes that contain values for each parent attribute value.

The values must be of the same number, and in the same order, as the parent attribute values. You must also have a [SUBLISTBOX] value for each parent value.

Note Use only the GUI to change list boxes.

[TYPE]

Specifies the type of list box, as follows:

```
[TYPE]listbox_type[/TYPE]
```

The possible type values are shown in the following table.

Type Value	Description
ATTRIBUTE	Indicates that the list box can have dependent attributes and is an attribute in the Rational Synergy database.
STANDARD	Indicates that the listbox can have dependent attributes but is not an attribute in the Rational Synergy database.
SUBLISTBOX	Indicates that the list box defines values for a child list box, for one of the parent values. Use this type with the NAME parameter to set one of the values within the parent attribute [SUBLISTBOX] tag set. Do not use this type with [SUBLIST] or [DEPENDENT_ATTR] tags.
SYSTEM	Indicates that the list box is maintained by Rational Change and should not be changed by users or administrators.

[VALUES]

Specifies the delimited list of values (strings) that will appear in the list. You must define each listed value.

Do not include spaces between entries or more than one delimiter between values.

LISTS SECTION

Defines named lists for use in data list box controls. A data list box is a list box in which each entry maps to a JavaScript data structure on the HTML form.

All definitions in this section are enclosed in the following tag set:

```
[CCM_LIST]...[/CCM_LIST]
```

Tags summary

The following table shows the tags used in the `LISTS SECTION`.

Tag	Description
[APPEND]	Determines whether values are appended to a list.
[NAME]	Specifies a name for the list.
[SECTION]	Specifies the section in the configuration file where the items listed in [VALUES] are defined.
[VALUES]	Specifies the delimited list of values (strings) that will appear in the list.

[APPEND]

Determines whether values are appended to a list.

When set to `TRUE`, allows values to be appended to a list box, as follows:

```
[CCM_LIST][NAME]listname[/NAME]
    [SECTION]{CCM_QUERY|CCM_REPORT|CCM_LIST|CCM_LISTBOX|
    CCM_VALUESLISTBOX}
    [/SECTION]
    [VALUES]value1|value2|value3...[/VALUES]
    [APPEND]{TRUE|FALSE}[/APPEND]
[CCM_LIST]
```

The default value is `FALSE` (overwrite the list).

[NAME]

Specifies a name for the list.

Note If you use an existing name, overwrite the existing name value.

[SECTION]

Specifies the section in the configuration file where the items listed in [VALUES] are defined. Valid section values are: CCM_QUERY, CCM_REPORT, CCM_LIST, CCM_LISTBOX, and CCM_VALUELISTBOX.

[VALUES]

Specifies the delimited list of values (strings) that will appear in the list. You must define each listed value.

Do not include spaces between entries or more than one delimiter between values.

Example

Specifies the list displayed in the **Select Query** list box in the Query dialog in the shipped version of Rational Change.

```
[CCM_LIST]
   [NAME]QueryGeneral[/NAME]
   [SECTION]CCM_QUERY[/SECTION]
   [VALUES]Entered change requests|In_review change
   requests|Assigned change requests|Resolved change
   requests|Concluded change requests|All change requests
   [/VALUES]
[/CCM_LIST]
```

DATALISTBOX SECTION

Defines one or more data list boxes put on a form as JavaScript using the PTDataListBox wslet call. This section is used only in the `*framework.cfg` and custom configuration files.

All definitions in this section are enclosed in the following tag set:

```
[CCM_DATALISTBOX]...[/CCM_DATALISTBOX]
```

The lists themselves must be defined in [CCM_LISTS]...[/CCM_LISTS].

Tags summary

The following table shows the tags used in the DATALISTBOX SECTION.

Tag	Description
[LIST]	Specifies the named list to display in the list box.
[NAME]	Specifies a name for data list box.
[role_name]	Specifies a role that makes a list role specific.

[LIST]

Specifies one or more lists to display in the list box. This value must be a [CCM_LIST] definition in the LISTS SECTION of a Rational Change configuration file.

[NAME]

Specifies a name for a data list box.

Note If you use an existing name, overwrite the existing name value.

[role_name]

Specifies one or more roles that make a list role-specific. You can list any valid Web role, as listed in the [ROLE] tags in the `pt.cfg` file (for example, *Admin* or *User*).

Examples

- The following [CCM_DATALISTBOX] definition is used to generate the JavaScript data structure that manages General Queries in the Query dialog.

```
[CCM_DATALISTBOX]
    [NAME]QUERYGENERAL[/NAME]
    [LIST]QueryGeneral[/LIST]
[/CCM_DATALISTBOX]
```

- The following [CCM_DATALISTBOX] definition is used to generate the JavaScript data structures that manage role-based General Queries and Admin Queries in the Query dialog.

```
[CCM_DATALISTBOX]
    [NAME]QUERYADMIN[/NAME]
    [Admin]
    [LIST]QueryAdmin[/LIST]
    [/Admin]
    [User]
    [LIST]QueryGeneral[/LIST]
    [/User]
[/CCM_DATALISTBOX]
```

TEMPLATE DEFINITION SECTION

Defines information used to create Rational Change templates, which provide the mechanism for role-based interface loading by allowing a call to a single template to load different template files under different circumstances.

You should understand that a [CCM_TEMPLATE] tag is *not* the same as a *template file*. In fact, [CCM_TEMPLATE] definitions can display different template files for different roles.

All definitions in this section are enclosed in the following tag set:

```
[CCM_TEMPLATE]...[/CCM_TEMPLATE]
```

Tags summary

The following table shows the tags used in the `TEMPLATE SECTION`.

Tag	Description
[ATTR]	Specifies the (optional) list of attributes to use on the form.
[NAME]	Specifies a name for a data list box.
[FILE]	Specifies the default template file.
[ONLOAD_ACTION]	Specifies the form type of the files that can be used by this template.
[role_name]	Specifies the template file that should be displayed for users with this role.

[ATTR]

Specifies the (optional) list of attributes to use on the form.

[NAME]

Specifies the name of this template. [CCM_TEMPLATE] definitions with the same name are merged according to the template merging rules.

[ONLOAD_ACTION]

Specifies the form type.

The form type (also called the *onload_action*) tells the server how to process the template file. Note that any template file that you reference in the [FILE] or [`role_name`] tags of a particular CCM_TEMPLATE tag *must* be of the form type specified in the template ONLOAD_ACTION tag.

The current valid onload actions are listed in “Servlet action summary” on page 109.

[role_name]

Specifies the template file that is displayed for users with this role (for example, *Admin* or *User*).

For example, you can define a role-based main Rational Change button bar to display an Admin button only when the user logs in with the *Admin* role, as follows:

```
[CCM_TEMPLATE]
    [NAME]ButtonBar[/NAME]
    [ONLOAD_ACTION]buttonbar_form[/ONLOAD_ACTION]
    [Admin]buttonbar_panel_admin.html[/Admin]
    [FILE]buttonbar_panel_dflt.html[/FILE]
[/CCM_TEMPLATE]
```

All users logged in as *Admin* see the button bar defined by [Admin]; that is, `buttonbar_panel_admin.html`.

Users logged in with any other role see the button bar defined by [FILE]; that is, `buttonbar_panel_dflt.html`.

[FILE]

Specifies the default template file. This is the template that is displayed for any role that does not have a [`role_name`] tag in the [CCM_TEMPLATE] definition.

Example

The default template used to create the blank Rational Change workspace is the file `workspace_blank_dflt.html`:

```
[CCM_TEMPLATE]
    [NAME]BlankWorkSpace[/NAME]
    [ONLOAD_ACTION]workspace_form[ONLOAD_ACTION]
    [FILE]workspace_blank_dflt.html[/FILE]
[/CCM_TEMPLATE]
```

4 Deployment descriptor settings

Rational Change uses a standard Web application deployment descriptor file. This XML file is used to describe servlets and other components of a Web application to define how a Web application should be deployed. The file is called `web.xml`, and it resides in a sub-directory called `WEB-INF`, directly under the Web application root directory.

Specifically, the `web.xml` file is used in Rational Change to do the following:

- Set up a filter that performs HTTP compression
- Configure initialization parameters
- Map logical servlet names to classes
- Provide URL mappings to servlets
- Set fully qualified domain names.

This chapter provides information about setting parameters in the `web.xml` file.

Configuring HTTP compression

HTTP compression, which is used to reduce download times by efficiently transferring response data back to the client, is enabled by default. However, compression will only be used if the following conditions exist:

- The requesting client can uncompress the response
- The response is at least 1024 bytes

Required parameter

None.

Example

To disable compression, comment out the `filter` section in `web.xml`:

```
<!--
<filter>
  <filter-name>CompressingFilter</filter-name>
  <filter-class>com.planetj.servlet.filter.compression.
    CompressingFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>CompressingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
-->
```

Setting fully qualified domain names

You can use a fully qualified domain name (FQDN) instead of a host name or IP address. In order to specify a FQDN, you must edit the `web.xml` file.

Locate the following information in your `web.xml` file:

```
<context-param>
<param-name>ccm_home</param-name>
<param-value>%CCM_HOME%</param-value>
</context-param>
<context-param>
<param-name>protocol</param-name>
<param-value>http</param-value>
</context-param>
<context-param>
<param-name>port</param-name>
<param-value>%PORT%</param-value>
</context-param>
```

After this info, create a new section containing the following info:

```
<context-param>
<param-name>hostname</param-name>
<param-value>your_FQDN</param-value>
</context-param>
```

where *`your_FQDN`* has a format similar to:

```
shark.abcd.abcco.com
```

The FQDN starts with a host name and continues all the way up to the top-level domain name.

5 Form load actions

PTweb actions lo ad forms. That is, PTweb generates the Rational Change user interface forms.

See “Sending a request to a servlet” on page 22 for general information about the JavaScript functions and hidden forms used to perform the actions.

Servlet action summary

The following table shows the PTweb actions for Rational Change.

Action	Description
admin_form	Loads an administrative form.
buttonbar_form	Loads a button bar form.
exit_form	Loads the exiting form.
frameset_form	Loads a frameset form, which defines the other frames that are loaded.
main_buttonbar_form	Loads the Rational Change button bar template.
object_attr_show_form	Loads an object details/modify form, based on an attribute value.
object_show_form	Loads an object details/modify form.
problem_attr_show_form	Loads a change request details/modify or transition form, based on an attribute value.
problem_show_form	Loads a change request details/modify or transition form.
problem_submit_form	Loads a form for submitting a change request.
query_form	Loads a form for performing a query and generating a report.
report_form	Loads a form for generating a predefined report.
status_form	Loads a status form.

Action	**Description**
task_attr_show_form	Loads a task details/modify or transition form, based on an attribute value.
task_show_form	Loads a task details/modify or transition form.
task_submit_form	Loads a form for submitting a task.
tokenless_form	Loads a form that does not require a role or token.
tokenless_frameset_form	Loads a frameset form that does not require a role or token (a sessionless form).
workspace_form	Loads a generic form that occupies the workspace.

Servlet section descriptions

The following sections describe individual PTweb actions in detail.

Note that for most form load actions, `window.document.pt_submission.action` is set to `"PTweb_servlet"`. `"PTweb_servlet"` is an abbreviation for the following string:

```
"<!-- WSLET CODE=PTAdmin --><!-- PARAM NAME=SERVLET_PATH
VALUE=PTWEB_SERVLET --><!-- /WSLET -->";
```

admin_form

Loads an administrative form.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"admin_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

buttonbar_form

Loads a button bar form.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"buttonbar_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

exit_form

Loads the exiting form.

You should use this action only for exiting Rational Change.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"exit_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

frameset_form

Loads a frameset form, which defines the other frames that are loaded.

Wslets are not available on frameset forms; therefore, Rational Change sets `CCM_*` variables in the specified frameset template file for any values set—and submitted—in the JavaScript function.

The parameters you can set in the JavaScript function, and the parameters set to these values in the `frameset_form` file, are shown in the following table.

In JavaScript Function	In Frameset Template	Substitution
`role`	`CCM_ROLE`	Replaced by the current user's role.
`token`	`CCM_TOKEN`	Replaced by the current user's token identifier.
`timestamp`	`CCM_TIMESTAMP`	Replaced by the timestamp you want loaded with this form.
`problem_number`	`CCM_PROBLEM_NUMBER`	Replaced by the focus change request `problem_number`.
`task_number`	`CCM_TASK_NUMBER`	Replaced by the focus task `task_number`.
`cvid`	`CCM_CVID`	Replaced by the focus object `cvid`.
`crstatus` or `status`	`CCM_STATUS`	Replaced by the focus change request (or task) `status` (or `tsk_status`).
`CR_PROCESS_XML`	`CCM_CR_PROCESS_XML`	Replaced by the CR Process XML file to be referenced.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"frameset_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
"template_name";

   <!-- Optional Data -->
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;
   window.document.pt_submission.problem_number.value =
problem_number_value;
```

```
   window.document.pt_submission.task_number.value =
task_number_value;
   window.document.pt_submission.cvid.value = cvid_value;
   window.document.pt_submission.crstatus.value =
crstatus_value;
   window.document.pt_submission.status.value = status_value;
   window.document.pt_submission.CR_PROCESS_XML.value =
CR_PROCESS_XML_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Optional Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="task_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="cvid" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="status" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="CR_PROCESS_XML" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

main_buttonbar_form

Loads the Rational Change main button bar template.

Typically, the button bar template will be role based.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"main_buttonbar_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

object_attr_show_form

Loads an object details/modify form, based on an attribute value.

The attribute value must be a template that is defined in a configuration file.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"object_attr_show_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
"attribute_name";
   window.document.pt_submission.cvid.value = cvid_value;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Attribute Data -->
   <INPUT NAME="cvid" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

object_show_form

Loads an object details/modify form. This requires a location database attribute be present. This attribute contains the database the task was looked up from. The database is the complete path as Change knows it.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"object_show_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.cvid.value = cvid_value;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
```

```
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Attribute Data -->
    <INPUT NAME="cvid" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

problem_attr_show_form

Loads a change request details/modify or transition form, based on an attributes value.

The attribute value must be a template that is defined in a configuration file.

JavaScript function

```
function request_form_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTweb_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"problem_attr_show_form";
    window.document.pt_submission.TEMPLATE_FLAG.value =
attribute_name;
    window.document.pt_submission.problem_number.value =
problem_number_value;
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <!-- PT Attribute Data -->
    <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

problem_show_form

Loads a change request details on the problem details, modify, or transition form.

JavaScript function

```
function request_form_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTweb_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"problem_show_form";
    window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
    window.document.pt_submission.problem_number.value =
problem_number_value;
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Attribute Data -->
    <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

problem_submit_form

Loads a form for submitting a change request.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"problem_submit_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

query_form

Loads a form for performing a query and generating a report.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"query_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

report_form

Loads a form for generating a predefined report.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"report_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

status_form

Loads a status form. The status form displays results of PTactions and PTAdminActions.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"status_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

For example, the following shows how to reload the configuration file into memory.

JavaScript function:

```
function Reload_Configuration_Data()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
      "/servlet/com.continuus.webpt.servlet.PTaction";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"loadconfig";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "custom response template of type status_form";

   // PT Security Data
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;
   window.document.pt_submission.submit();
}
```

Hidden form:

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

task_attr_show_form

Loads a task details/modify or transition form, based on an attribute value.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"task_attr_show_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
attribute_name;
   window.document.pt_submission.task_number.value =
task_number_value;
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
```

```
    <!-- PT Attribute Data -->
    <INPUT NAME="task_number" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

task_show_form

Loads a task details/modify or transition form. This requires a location database attribute be present. This attribute contains the database the task was looked up from. The database is the complete path as Change knows it, for example, `/vol/shark/ccmdb/Change 5.2`.

JavaScript function

```
function request_form_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTweb_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"task_show_form";
    window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
    window.document.pt_submission.task_number.value =
task_number_value;
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Attribute Data -->
    <INPUT NAME="task_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

task_submit_form

Loads a form for submitting a task.

JavaScript function

```
function request_form_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTweb_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"task_submit_form";
    window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

tokenless_form

Loads a form that does not require a role or token (a sessionless form), such as a temporary form that will be replaced with a new form when the data is available.

For example, this action is used to load the wait indicator in the following excerpt from a PTaction function call:

```
.
.
.
// load the wait indicator
parent.subbuttonbar.location="/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=
tokenless_form&TEMPLATE_FLAG=WaitButtonBar";
.
.
.
```

JavaScript function

```
function request_form_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTweb_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"tokenless_form";
    window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <!-- PT Action Information -->
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

tokenless_frameset_form

Loads a frameset form that does not require a role or token (a sessionless form), such as a temporary frameset that will be replaced with a new frameset.

This action is the same as `tokenless_form`, except that security information like `role` an `token` are not available.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"tokenless_frameset_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

workspace_form

Loads a generic form that occupies the workspace. Use this template type for instructional or transient data.

JavaScript function

```
function request_form_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTweb_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"workspace_form";
   window.document.pt_submission.TEMPLATE_FLAG.value =
template_name;
```

```
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

6 User actions

Rational Change uses the PTaction servlet to send user data to, and retrieve user data from, the Rational Synergy database.

See “Sending a request to a servlet” on page 22 for general information about the JavaScript functions and hidden forms used to perform the actions.

The following sections describe the actions you can perform using the PTaction servlet.

Servlet action summary

The following table shows the PTaction servlet actions.

Action	Description
add_user_listbox	Adds a new list box to the user’s configuration file.
add_user_query	Adds a query to the user’s configuration file.
add_user_report	Adds a report to the user’s configuration file.
auto_assign_problem	Submits a change request and assigns it to the current user.
auto_assign_problem_and_associate_with_problem	Submits a change request, assigns it to the current user, and associates it with another change request.
auto_assign_problem_and_associate_with_problem_and_attachment	Submits and auto-assigns the change request, associates it with another change request, and includes attachment(s).
auto_assign_problem_and_attachment	Submits and auto-assigns the change request and includes attachment(s).
auto_assign_task	Submits a task and assigns it to the current user.
delete_user_listbox	Deletes a list box from the user’s configuration file.
delete_user_query	Deletes a query from the user’s configuration file.

Action	Description
delete_user_report	Deletes a report from the user's configuration file.
delete_user_saved_report	Deletes selected user reports immediately.
immediate_report	Queries the database and builds a report of the results, without polling for the result.
modify_problem	Modifies the details/attributes of a change request.
modify_problem_new_attachment	Modifies the change request and adds new attachment(s).
modify_task	Modifies the attributes of a task.
object_object	Creates an object-to-object relationship.
object_problem	Creates an object-to-change request relationship.
object_task	Creates an object-to-task relationship.
objectreport	Reports on a single object.
problem_object	Creates a Change Request to Object relationship.
problem_problem	Creates a Change Request to Change Request relationship.
problem_task	Creates a Change Request to Task relationship.
problemreport	Reports on a single change request.
query	Queries the database and builds a report of the results.
report	Queries the database and builds a report of the results.
save_user_saved_report	Saves selected reports so that they will not be deleted and the owner can retrieve them later.
search	Searches for change requests or tasks containing the specified text.
submit_problem	Submits a change request.

Action	Description
submit_problem_and_associate_with_problem	Submits a change request and associates it with another change request.
submit_problem_and_associate_with_problem_and_attachment	Submits the change request, associates it with another change request, and includes attachment(s).
submit_problem_and_attachment	Submits the change request and includes attachment(s).
submit_task	Submits a task.
submit_task_and_associate_with_problem	Submits a task and associates it with a change request using the `associated_task` relationship.
switch_database	Logs the user into a different database.
task_problem	Creates a task-to-change request relationship.
task_task	Creates a task-to-task relationship.
taskreport	Reports on a single task.
tokenless_submit_problem	Submits a change request using email.
transition_problem	Transitions a change request to a new state.
transition_problem_new_attachment	Transitions the change request and adds new attachment(s).
transition_task	Transitions a task to a new state.
update_user_listbox	Updates/overwrites the elements of a list box in the user's configuration file.
update_user_profile	Updates or creates a user's profile file
update_user_profile_xml	Updates a query's query string and description in the user's configuration file.
update_user_listbox	Downloads a single attachment to the browser.
update_user_query	Updates a query's query string and description.
update_user_report	Updates a report query string and description in the user's configuration file.

Servlet action descriptions

The following sections describe individual PTaction actions in detail.

Note that for most user actions, `window.document.pt_submission.action` is set to `"`*`PTaction_servlet`*`"`. `"`*`PTaction_servlet`*`"` is an abbreviation for the following string:

```
"<!-- WSLET CODE=PTAdmin --><!-- PARAM NAME=SERVLET_PATH
VALUE=PTACTION_SERVLET --><!-- /WSLET -->";
```

add_user_listbox

Adds a new list box to the user's configuration file.

You can obtain the system delimiters using the following wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_SECURITY_DATA -->
<!-- PARAM NAME=CCM_DELIMITER -->
<!-- PARAM NAME=CCM_SUB_DELIMITER -->
<!-- /WSLET -->
```

Result:

```
var delimiter = "|";
var sub_delimiter = ":";
```

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"add_user_listbox";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CCM_DATALISTBOX.value =
      "user config data list box name";
   window.document.pt_submission.CCM_LISTBOX.value =
"listbox_name";
```

```
    window.document.pt_submission.VALUES.value =
        "delimited list of values";

    window.document.pt_submission.submit();
}
```

Hidden function form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CCM_LISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="VALUES" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

add_user_query

Adds a query to the user's configuration file.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"add_user_query";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    window.document.pt_submission.CCM_DATALISTBOX.value =
        "user config data list box name";
```

```
   window.document.pt_submission.CCM_QUERY.value =
"query_name";
   window.document.pt_submission.DESCRIPTION.value =
"query_description";
   window.document.pt_submission.CHOSEN_QUERY.value =
       "existing base query name";
   window.document.pt_submission.QUERY_STRING.value =
"query_string";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CCM_QUERY" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="DESCRIPTION" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

add_user_report

Adds a report to the user's configuration file.

The following optional parameters are accepted and, if specified, override the base definitions:

```
ATTRIBUTES_number
CUSTOM_WSLET_number
SORT_ORDER_number
XML_CONTENT_number
```

The parameters are configured using the [SUBREPORT_LEVELS] system setting. The *number* determines how many subreports you can define through one report creation. See “problemreport” on page 165 for an example.

The default "NO_ACTION" causes the option to be ignored when the option is unnecessary or invalid.

See [ATTRS] for the syntax of the "*delimited list of attributes...*" and "*delimited list of sort order attributes...*".

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"add_user_report";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CCM_DATALISTBOX.value =
      "user config data list box name";
   window.document.pt_submission.CCM_REPORT.value =
"report_name";
   window.document.pt_submission.DESCRIPTION.value =
"report_description";
   window.document.pt_submission.REPORT_TITLE.value =
"report_title";
   window.document.pt_submission.CHOSEN_REPORT.value =
      "base report from which to copy";
   window.document.pt_submission.CHOSEN_QUERY.value =
      "base query from which to copy";
   window.document.pt_submission.QUERY_STRING.value =
"custom_query_string";
   window.document.pt_submission.CHOSEN_DEFAULT_REPORT.value =
       "[TRUE|FALSE]: whether this is the default report";

   window.document.pt_submission.ATTRIBUTES_1.value =
      "delimited list of attributes for parent report";
   window.document.pt_submission.ATTRIBUTES_2.value =
      "delimited list of attributes for subreport 1";
   window.document.pt_submission.ATTRIBUTES_3.value =
      "delimited list of attributes for subreport 2";
   window.document.pt_submission.SORT_ORDER_1.value =
       "delimited list of sort order attributes for parent
report";
   window.document.pt_submission.SORT_ORDER_2.value =
       "delimited list of sort order attributes for subreport
1";
   window.document.pt_submission.SORT_ORDER_3.value =
       "delimited list of sort order attributes for subreport
2";

   window.document.pt_submission.CUSTOM_WSLET_1.value = "name
of a custom java
       class referenced in the parent report";
   window.document.pt_submission.CUSTOM_WSLET_2.value =
       "name of a custom java class referenced in subreport 1";
   window.document.pt_submission.CUSTOM_WSLET_3.value =
       "name of a custom java class referenced in subreport 2";

   window.document.pt_submission.XML_CONTENT_1.value = "XML
data referenced by
       the custom java class in the parent report";
   window.document.pt_submission.XML_CONTENT_2.value = "XML
data referenced by
       the custom java class in subreport 1";
   window.document.pt_submission.XML_CONTENT_3.value = "XML
data referenced by
       the custom java class in subreport 2";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CCM_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="DESCRIPTION" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_DEFAULT_REPORT" TYPE=HIDDEN
    VALUE="NO_ACTION"></INPUT>

    <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="ATTRIBUTES_2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="ATTRIBUTES_3" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_3" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="CUSTOM_WSLET_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_3" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_3" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

auto_assign_problem

Submits a change request and assigns it to the current user.

JavaScript function

```
function perform_action_OnClick()
```

```
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"auto_assign_problem";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.crstatus.value ="submit-to
state";
   window.document.pt_submission.submitter.value = "current
user's login name";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="submitter" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

auto_assign_problem_and_associate_with_problem

Submits a change request, assigns it to the current user, and associates the new change request with the specified change request number.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
      "auto_assign_problem_and_associate_with_problem";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.RELATION_NAME.value =
"relation name";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
      "CR to receive the association";
   window.document.pt_submission.crstatus.value ="submit-to
state";
   window.document.pt_submission.submitter.value = "current
user's login name";
```

```
   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="submitter" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

auto_assign_problem_and_associate_with_problem_and_attachment

Submits and auto-assigns the change request, associates it with another change request, and includes attachment(s).

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =

"auto_assign_problem_and_associate_with_problem_and_attachment"
;
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.RELATION_NAME.value =
"relation name";
   window.document.pt_submission.ATTACHMENT_RELATION.value =
      "attachment relation name";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
      "CR to receive the association";
   window.document.pt_submission.crstatus.value = "submit-to
state";
   window.document.pt_submission.submitter.value = "user's
login name";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="submitter" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Attachment Data -->
   <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
      VALUE="NO_ACTION"></INPUT>

   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR ALIGN=left>
      <TD WIDTH="750">
      <B>Attachment Information</B>
      </TD>
      </TR>
   </TABLE>

   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
      <TR>
      <TD WIDTH="730">
```

```
    <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
    <TR>
    <TD WIDTH="730">Attachment:&nbsp;
    <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
    </INPUT>&nbsp;
    <INPUT NAME="_ATTACHMENT_IS_BINARY1"
    TYPE=CHECKBOX>Binary File</INPUT>
    <BR>Description:&nbsp;
    <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
    MAXLENGTH=1024 SIZE=80></INPUT>
    </TD>
    </TR>
     .
     .
     .
    </TABLE>
    </TD>
    </TR>
</TABLE>
<TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
    <TR>
    <TD WIDTH="730">
    <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
    <TR>
    <TD WIDTH="730">Attachment:&nbsp;
    <INPUT NAME="_ATTACHMENT_NAME2" TYPE=FILE SIZE=50>
    </INPUT>&nbsp;
    <INPUT NAME="_ATTACHMENT_IS_BINARY2"
    TYPE=CHECKBOX>Binary File</INPUT>
    <BR>Description:&nbsp;
    <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
    MAXLENGTH=1024 SIZE=80></INPUT>
    </TD>
    </TR>
     .
     .
     .
    </TABLE>
    </TD>
    </TR>
</TABLE>
.
.
.
<TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
    <TR>
    <TD WIDTH="730">
    <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
    <TR>
```

```
            <TD WIDTH="730">Attachment:&nbsp;
            <INPUT NAME="_ATTACHMENT_NAMEN" TYPE=FILE SIZE=50>
            </INPUT>&nbsp;
            <INPUT NAME="_ATTACHMENT_IS_BINARYN"
            TYPE=CHECKBOX>Binary File</INPUT>
            <BR>Description:&nbsp;
            <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
            MAXLENGTH=1024 SIZE=80></INPUT>
            </TD>
            </TR>
             .
             .
             .
            </TABLE>
            </TD>
            </TR>
        </TABLE>
    </FORM>
```

auto_assign_problem_and_attachment

Submits and auto-assigns the change request and includes attachment(s).

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";
```

```
   window.document.pt_submission.ACTION_FLAG.value =
       "auto_assign_problem_and_attachment";
   window.document.pt_submission.TEMPLATE_FLAG.value =
       "optional custom response template";
   window.document.pt_submission.ATTACHMENT_RELATION.value =
       "attachment relation name";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.crstatus.value ="submit-to
state";
   window.document.pt_submission.submitter.value = "current
user’s login name";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="submitter" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
```

```
    <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- Attachment Data -->
    <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
        VALUE="NO_ACTION"></INPUT>

    <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
        <TR ALIGN=left>
        <TD WIDTH="750">
        <B>Attachment Information</B>
        </TD>
        </TR>
    </TABLE>

    <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
        <TR>
        <TD WIDTH="730">
        <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
        <TR>
        <TD WIDTH="730">Attachment:&nbsp;
        <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
        </INPUT>&nbsp;
        <INPUT NAME="_ATTACHMENT_IS_BINARY1"
        TYPE=CHECKBOX>Binary File</INPUT>
        <BR>Description:&nbsp;
        <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
        MAXLENGTH=1024 SIZE=80></INPUT>
        </TD>
        </TR>
         .
         .
         .
        </TABLE>
        </TD>
        </TR>
    </TABLE>
</FORM>
```

auto_assign_task

Submits a task and assigns it to the current user.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"auto_assign_task";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.resolver.value = "user's login
name";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
```

```
    <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

    <!-- Action Data -->
    <INPUT NAME="resolver" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

    <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

delete_user_listbox

Deletes a list box from the user's configuration file.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"delete_user_listbox";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    window.document.pt_submission.CCM_DATALISTBOX.value =
        "user config data list box name";
    window.document.pt_submission.CCM_LISTBOX.value =
"listbox_name";

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
```

```
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CCM_LISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

delete_user_query

Deletes a query from the user's configuration file.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"delete_user_query";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.CCM_CALLBACK =
"javascriptFunctionName";

   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CCM_DATALISTBOX.value =
      "user config data list box name";
   window.document.pt_submission.CHOSEN_QUERY.value =
"query_name";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
```

```
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

delete_user_report

Deletes a report from the user's configuration file.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"delete_user_report";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";
    window.document.pt_submission.CCM_CALLBACK =
“javascriptFunctionName”;

    window.document.pt_submission.token.value = token_value;

    window.document.pt_submission.CCM_DATALISTBOX.value =
        "user config data list box name";
    window.document.pt_submission.CHOSEN_REPORT.value =
"report_name";

    window.document.pt_submission.submit();
```

Hidden form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

delete_user_saved_report

Deletes selected user reports immediately.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"delete_user_saved_report";
   window.document.pt_submission.TEMPLATE_FLAG.value =
"ViewSavedReports";
   window.document.pt_submission.CCM_CALLBACK =
“javascriptFunctionName”;

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.XML_CONTENT.value = "delimited
list of file names";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG"   TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

immediate_report

Queries the database and builds a report of the results, without polling for the result. Therefore, the report should be small.

The following optional parameter is accepted:

`empty_results_template`

This specifies the name of the template to load if there are no results (instead of using the normal report template).

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"immediate_report";
    window.document.pt_submission.TEMPLATE_FLAG.value =
       "optional custom response template, if an error occurs";
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    // Action Data
    window.document.pt_submission.CHOSEN_QUERY.value =
optional_query;
    window.document.pt_submission.QUERY_STRING.value =
optional_qry_str;
    window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
    window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

    // Parent-Report data, Optional if no changes to defaults
    window.document.pt_submission.ATTRIBUTES_level.value =
attributes(level);
```

```
   window.document.pt_submission.SORT_ORDER_level.value =
sort_order(level);
   window.document.pt_submission.CUSTOM_WSLET_level.value =
cust_wslet(level);
   window.document.pt_submission.XML_CONTENT_level.value =
XML_content(level);
   .
   .
   .
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_0"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    .
    .
    .
    <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

modify_problem

Modifies the details/attributes of a change request.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"modify_problem";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
    window.document.pt_submission.problem_number.value =
problem_number;
    window.document.pt_submission.modify_time.value =
modify_time;

    window.document.pt_submission.attribute_name1.value =
attribute_value1;
    window.document.pt_submission.attribute_name2.value =
attribute_value2;
    .
```

```
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

modify_problem_new_attachment

Modifies the change request and adds new attachment(s).

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
      "modify_problem_new_attachment";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.ATTACHMENT_RELATION.value =
      "relationship name of type CCM_PROBLEM_OBJECT";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
problem_number;
   window.document.pt_submission.modify_time.value =
modify_time;

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
   <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Attachment Data -->
   <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
   VALUE="NO_ACTION"></INPUT>

   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR ALIGN=left>
      <TD WIDTH="750">
      <B>Attachment Information</B>
      </TD>
      </TR>
   </TABLE>
   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
      <TR>
      <TD WIDTH="730">
      <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR>
      <TD WIDTH="730">Attachment:&nbsp;
      <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
      </INPUT>&nbsp;
      <INPUT NAME="_ATTACHMENT_IS_BINARY1"
      TYPE=CHECKBOX>Binary File</INPUT>
      <BR>Description:&nbsp;
      <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
      MAXLENGTH=1024 SIZE=80></INPUT>
      </TD>
      </TR>
       .
       .
       .
      </TABLE>
      </TD>

      </TR>
      <TR>
      <TD WIDTH="730">
      <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
```

```
          <TR>
          <TD WIDTH="730">Attachment:&nbsp;
          <INPUT NAME="_ATTACHMENT_NAME2" TYPE=FILE SIZE=50>
          </INPUT>&nbsp;
          <INPUT NAME="_ATTACHMENT_IS_BINARY2"
          TYPE=CHECKBOX>Binary File</INPUT>
          <BR>Description:&nbsp;
          <INPUT NAME="_ATTACHMENT_COMMENT2" TYPE=TEXT
          MAXLENGTH=1024 SIZE=80></INPUT>
          </TD>
          </TR>
            .
            .
            .
          </TABLE>
          </TD>
          </TR>
          <TR>
          <TD WIDTH="730">
          <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
          <TR>
          <TD WIDTH="730">Attachment:&nbsp;
          <INPUT NAME="_ATTACHMENT_NAME3" TYPE=FILE SIZE=50>
          </INPUT>&nbsp;
          <INPUT NAME="_ATTACHMENT_IS_BINARY3"
          TYPE=CHECKBOX>Binary File</INPUT>
          <BR>Description:&nbsp;
          <INPUT NAME="_ATTACHMENT_COMMENT3" TYPE=TEXT
          MAXLENGTH=1024 SIZE=80></INPUT>
          </TD>
          </TR>
           .
           .
           .
          </TABLE>
          </TD>
          </TR>
      </TABLE>
  </FORM>
```

modify_task

Modifies the attributes of a task. This requires a location database attribute be present. This attribute contains the database the task was looked up from. The database is the complete path as Change knows it.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"modify_task";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.task_number.value =
task_number;
   window.document.pt_submission.modify_time.value =
modify_time;

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
```

```
    <!-- Action Data -->
    <INPUT NAME="task_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

object_object

Creates an object-to-object relationship.

See “Relationship actions” on page 204.

object_problem

Creates an object-to-change request relationship.

See “Relationship actions” on page 204.

object_task

Creates an object-to-task relationship.

See “Relationship actions” on page 204.

objectreport

Reports on a single object. This action must reference a query string that uses keyword "`%cvid`" substitution.

For example, the following definition is in [CCM_QUERY]:

```
[QRY_STRING]cvid='%cvid'[/QRY_STRING]
```

Because this action queries the database and builds a report without polling, the report should be small.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"objectreport";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template, if an error occurs";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.cvid.value = cvid_value;

   window.document.pt_submission.CHOSEN_QUERY.value =
optional_query;
   window.document.pt_submission.QUERY_STRING.value =
optional_qry_str;
   window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
   window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

   // Parent-Report data, Optional if no changes to defaults
   window.document.pt_submission.ATTRIBUTES_0.value =
attributes(0);
   window.document.pt_submission.SORT_ORDER_0.value =
sort_order(0);
   window.document.pt_submission.CUSTOM_WSLET_0.value =
cust_wslet(0);
   window.document.pt_submission.XML_CONTENT_0.value =
XML_content(0);

   // Subreport data. Optional if there are no sub-reports,
      or no changes to defaults.
   window.document.pt_submission.ATTRIBUTES_1.value =
attributes(1);
   window.document.pt_submission.SORT_ORDER_1.value =
sort_order(1);
   window.document.pt_submission.CUSTOM_WSLET_1.value =
cust_wslet(1);
   window.document.pt_submission.XML_CONTENT_1.value =
XML_content(1);
   .
```

```
    .
    .
    window.document.pt_submission.ATTRIBUTES_N.value =
attributes(N);
    window.document.pt_submission.SORT_ORDER_N.value =
sort_order(N);
    window.document.pt_submission.CUSTOM_WSLET_N.value =
cust_wslet(N);
    window.document.pt_submission.XML_CONTENT_N.value =
XML_content(N);

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="cvid" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="ATTRIBUTES_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_0"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

problem_object

Creates an change request-to-object relationship.

See “Relationship actions” on page 204.

problem_problem

Creates an change request-to-change request relationship.

See “Relationship actions” on page 204.

problem_task

Creates a change request-to-task relationship.

See “Relationship actions” on page 204.

problemreport

Reports on a single change request.

This action must reference a query string that uses keyword "`%problem_number`" substitution. For example:

```
[QRY_STRING]problem_number='%problem_number'[/QRY_STRING]
```

Because this action queries the database and builds a report without polling, the report should be small.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"problemreport";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template, if an error occurs";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
problem_number;
   window.document.pt_submission.CHOSEN_QUERY.value =
optional_query;
   window.document.pt_submission.QUERY_STRING.value =
optional_qry_str;
   window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
   window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

   // Parent-Report data, Optional if no changes to defaults
   window.document.pt_submission.ATTRIBUTES_0.value =
attributes(0);
   window.document.pt_submission.SORT_ORDER_0.value =
sort_order(0);
   window.document.pt_submission.CUSTOM_WSLET_0.value =
cust_wslet(0);
   window.document.pt_submission.XML_CONTENT_0.value =
XML_content(0);

   // Subreport data, Optional if there are no sub-reports,
      or no changes to defaults
   window.document.pt_submission.ATTRIBUTES_1.value =
attributes(1);
   window.document.pt_submission.SORT_ORDER_1.value =
sort_order(1);
   window.document.pt_submission.CUSTOM_WSLET_1.value =
cust_wslet(1);
   window.document.pt_submission.XML_CONTENT_1.value =
XML_content(1);
   .
   .
   .
   window.document.pt_submission.ATTRIBUTES_N.value =
attributes(N);
   window.document.pt_submission.SORT_ORDER_N.value =
sort_order(N);
   window.document.pt_submission.CUSTOM_WSLET_N.value =
cust_wslet(N);
   window.document.pt_submission.XML_CONTENT_N.value =
XML_content(N);

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
```

```
    .
    <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

query

Queries the database and builds a report of the results.

The polling template determines whether the report is completed. If you do not set TEMPLATE_FLAG, the default ReportCompletionCheck polling template is used. See report completion check.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value = "query";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "custom report polling template name"; //OPTIONAL!

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    // Action Data
    window.document.pt_submission.CHOSEN_QUERY.value = query;
    window.document.pt_submission.QUERY_STRING.value =
query_string;
    window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
    window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

    // Parent-Report data, Optional if no changes to defaults
    window.document.pt_submission.ATTRIBUTES_0.value =
attributes(0);
    window.document.pt_submission.SORT_ORDER_0.value =
sort_order(0);
```

```
    window.document.pt_submission.CUSTOM_WSLET_0.value =
cust_wslet(0);
    window.document.pt_submission.XML_CONTENT_0.value =
XML_content(0);

    // Subreport data. Optional if there are no sub-reports,
        or no changes to defaults.
    window.document.pt_submission.ATTRIBUTES_1.value =
attributes(1);
    window.document.pt_submission.SORT_ORDER_1.value =
sort_order(1);
    window.document.pt_submission.CUSTOM_WSLET_1.value =
cust_wslet(1);
    window.document.pt_submission.XML_CONTENT_1.value =
XML_content(1);
    .
    .
    .
    window.document.pt_submission.ATTRIBUTES_N.value =
attributes(N);
    window.document.pt_submission.SORT_ORDER_N.value =
sort_order(N);
    window.document.pt_submission.CUSTOM_WSLET_N.value =
cust_wslet(N);
    window.document.pt_submission.XML_CONTENT_N.value =
XML_content(N);

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="ATTRIBUTES_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_0"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

report

Queries the database and builds a predefined report of the results.

The polling template determines whether the report is completed. If you do not set `TEMPLATE_FLAG`, the default ReportCompletionCheck polling template is used. See report_completion_check.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";
```

```
   window.document.pt_submission.ACTION_FLAG.value = "report";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "custom report polling template name"; //OPTIONAL!

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.CHOSEN_QUERY.value =
optional_query;
   window.document.pt_submission.QUERY_STRING.value =
optional_query_string;
   window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
   window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

   // Parent-Report data, Optional if no changes to defaults
   window.document.pt_submission.ATTRIBUTES_0.value =
attributes(0);
   window.document.pt_submission.SORT_ORDER_0.value =
sort_order(0);
   window.document.pt_submission.CUSTOM_WSLET_0.value =
cust_wslet(0);
   window.document.pt_submission.XML_CONTENT_0.value =
XML_content(0);

   // Subreport data. Optional if there are no sub-reports, or
no changes to
   defaults
   window.document.pt_submission.ATTRIBUTES_1.value =
attributes(1);
   window.document.pt_submission.SORT_ORDER_1.value =
sort_order(1);
   window.document.pt_submission.CUSTOM_WSLET_1.value =
cust_wslet(1);
   window.document.pt_submission.XML_CONTENT_1.value =
XML_content(1);

   .
   .
   .
   window.document.pt_submission.ATTRIBUTES_N.value =
attributes(N);
   window.document.pt_submission.SORT_ORDER_N.value =
sort_order(N);
   window.document.pt_submission.CUSTOM_WSLET_N.value =
cust_wslet(N);
```

```
   window.document.pt_submission.XML_CONTENT_N.value =
XML_content(N);

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_0"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

save_user_saved_report

Saves selected reports so that they will not be deleted and the owner can retrieve them later.

The files are simply flagged to be read only. The report manager will not delete files automatically that are read only.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"save_user_saved_report";
   window.document.pt_submission.TEMPLATE_FLAG.value =
"ViewSavedReports";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.XML_CONTENT.value = "delimited
list of file names";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG"   TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

search

Searches for change requests or tasks containing the specified text.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value = "search";
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    // Action Data
    window.document.pt_submission.search_string.value ="string
to search for";
    window.document.pt_submission.search_object_type.value =
[problem|task|both];
    window.document.pt_submission.search_dbs.value =
[all|db1;db2;...];

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

    <!-- Action Data -->
    <INPUT NAME="search_string" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="search_object_type" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="search_dbs" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

submit_problem

Submits a change request.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value
="submit_problem";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
    window.document.pt_submission.crstatus.value ="submit-to
state";

    window.document.pt_submission.attribute_name1.value =
attribute_value1;
    window.document.pt_submission.attribute_name2.value =
attribute_value2;
    .
    .
    .
    window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

submit_problem_and_associate_with_problem

Submits a change request and associates the change request with an existing change request number.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
      "submit_problem_and_associate_with_problem";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.RELATION_NAME.value =
"relation name";
```

```
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
      "CR to receive the association";
   window.document.pt_submission.crstatus.value ="submit-to
state";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

submit_problem_and_associate_with_problem_and_attachment

Submits a change request, associates it with another change request, and includes attachment(s).

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =

"submit_problem_and_associate_with_problem_and_attachment";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.RELATION_NAME.value =
"relation name";
   window.document.pt_submission.ATTACHMENT_RELATION.value =
      "attachment relation name";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
      "CR to receive the association";
   window.document.pt_submission.crstatus.value ="submit-to
state";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Attachment Data -->
   <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
      VALUE="NO_ACTION"></INPUT>

      <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR ALIGN=left>
      <TD WIDTH="750">
      <B>Attachment Information</B>
      </TD>
      </TR>
   </TABLE>

   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
```

```
            <TR>
            <TD WIDTH="730">
            <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
            <TR>
            <TD WIDTH="730">Attachment:&nbsp;
            <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
            </INPUT>&nbsp;
            <INPUT NAME="_ATTACHMENT_IS_BINARY1"
            TYPE=CHECKBOX>Binary File</INPUT>
            <BR>Description:&nbsp;
            <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
            MAXLENGTH=1024 SIZE=80></INPUT>
            </TD>
            </TR>
              .
             .
             .
            </TABLE>
            </TD>
            </TR>
        </TABLE>
    </FORM>
```

submit_problem_and_attachment

Submits the change request and includes attachment(s).

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
        "submit_problem_and_attachment";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";
    window.document.pt_submission.ATTACHMENT_RELATION.value =
        "attachment relation name";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
```

```
   window.document.pt_submission.crstatus.value ="submit-to
state";

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Attachment Data -->
   <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
      VALUE="NO_ACTION"></INPUT>

   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR ALIGN=left>
```

```
            <TD WIDTH="750">
            <B>Attachment Information</B>
            </TD>
            </TR>
        </TABLE>

        <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
            <TR>
            <TD WIDTH="730">
            <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
            <TR>
            <TD WIDTH="730">Attachment:&nbsp;
            <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
            </INPUT>&nbsp;
            <INPUT NAME="_ATTACHMENT_IS_BINARY1"
            TYPE=CHECKBOX>Binary File</INPUT>
            <BR>Description:&nbsp;
            <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
            MAXLENGTH=1024 SIZE=80></INPUT>
            </TD>
            </TR>
             .
             .
             .
            </TABLE>
            </TD>
            </TR>
        </TABLE>
    </FORM>
```

submit_task

Creates a task.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value
="submit_task";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
```

```
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

submit_task_and_associate_with_problem

Submits a task and associates it with a change request using the `associated_task` relationship.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
      "submit_task_and_associate_with_problem";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.RELATION_NAME.value =
"relation name";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
   window.document.pt_submission.timestamp.value =
timestamp_value;

   // Action Data
   window.document.pt_submission.problem_number.value =
      "CR to receive the association";
   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
```

```
   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

switch_database

Logs the user into a different database. The template loaded by this action will have a new token for the user.

JavaScript function

```
function switchDatabase()
{
   document.dbForm.method = "POST";
   document.dbForm.action = ""PTAdminAction_servlet"
   document.dbForm.target = "pt_status";
   document.dbForm.ACTION_FLAG.value = "switch_database";
   document.dbForm.TEMPLATE_FLAG.value =
"SwitchDatabaseStatus";
   document.dbForm.token.value = token;
   document.dbForm.new_database.value = "/vol/ccm_web/ccm_web";
   document.dbForm.submit();
}
```

Hidden form

```
<FORM NAME="dbForm">
   <DIV>
   <INPUT NAME="ACTION_FLAG" TYPE="HIDDEN" VALUE="NO_ACTION">
   <INPUT NAME="TEMPLATE_FLAG" TYPE="HIDDEN" VALUE="NO_ACTION">
   <INPUT NAME="new_database" TYPE="HIDDEN" VALUE="NO_ACTION">
   <INPUT NAME="token" TYPE="HIDDEN" VALUE="NO_ACTION">
   </DIV>
</FORM>
```

task_object

Creates a task-to-object relationship.

See "Relationship actions" on page 204.

task_problem

Creates a task-to-change request relationship.

See "Relationship actions" on page 204.

task_task

Creates a task-to-task relationship.

See "Relationship actions" on page 204.

taskreport

Reports on a single task.

This action must reference a query string that uses keyword "`%task_number`" substitution. For example:

```
[QRY_STRING]task_number='%task_number'[/QRY_STRING]
```

Because this action queries the database and builds a report without polling, the report should be small.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"taskreport";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template, if an error occurs";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Action Data
   window.document.pt_submission.task_number.value =
task_number;
```

```
   window.document.pt_submission.CHOSEN_QUERY.value =
optional_query;
   window.document.pt_submission.QUERY_STRING.value =
optional_qry_str;
   window.document.pt_submission.CHOSEN_REPORT.value =
optional_report;
   window.document.pt_submission.REPORT_TITLE.value =
optional_report_title;

   // Parent-Report data, Optional if no changes to defaults
   window.document.pt_submission.ATTRIBUTES_0.value =
attributes(0);
   window.document.pt_submission.SORT_ORDER_0.value =
sort_order(0);
   window.document.pt_submission.CUSTOM_WSLET_0.value =
cust_wslet(0);
   window.document.pt_submission.XML_CONTENT_0.value =
XML_content(0);

   // Subreport data. Optional if there are no sub-reports, or
no changes to
   defaults.
   window.document.pt_submission.ATTRIBUTES_1.value =
attributes(1);
   window.document.pt_submission.SORT_ORDER_1.value =
sort_order(1);
   window.document.pt_submission.CUSTOM_WSLET_1.value =
cust_wslet(1);
   window.document.pt_submission.XML_CONTENT_1.value =
XML_content(1);

   .
   .
   .
   window.document.pt_submission.ATTRIBUTES_N.value =
attributes(N);
   window.document.pt_submission.SORT_ORDER_N.value =
sort_order(N);
   window.document.pt_submission.CUSTOM_WSLET_N.value =
cust_wslet(N);
   window.document.pt_submission.XML_CONTENT_N.value =
XML_content(N);

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="task_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="REPORT_TITLE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_0"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_0" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ATTRIBUTES_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CUSTOM_WSLET_1"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT_1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="ATTRIBUTES_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="SORT_ORDER_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="CUSTOM_WSLET_N"TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="XML_CONTENT_N" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

tokenless_submit_problem

Submits a change request using email, or stores a submitted change request in a file, without starting a Rational Synergy session.

To use email submission, the following settings must be in a configuration file:

```
[CCM_SYSTEM]
    [SEND_MAIL]false[/SEND_MAIL]
    [MAIL_FILE]mail_file_path_and_name[/MAIL_FILE]
[/CCM_SYSTEM]
```

or

```
[CCM_SYSTEM]
    [SEND_MAIL]true[/SEND_MAIL]
    [MAIL_ADDR]probtrac_email_address[/MAIL_ADDR]
    [MAIL_PROTOCOL]mail_protocol[/MAIL_PROTOCOL]
    [MAIL_SERVER]your_server[/MAIL_SERVER]
[/CCM_SYSTEM]
```

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action =
        "/servlet/com.continuus.webpt.servlet.PTExternalAction";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"tokenless_submit_problem";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    // Action Data
    window.document.pt_submission.crstatus.value ="submit-to
state";

    window.document.pt_submission.attribute_name1.value =
attribute_value1;
    window.document.pt_submission.attribute_name2.value =
attribute_value2;
    .
    .
```

```
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Action Data -->
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

transition_problem

Transitions a change request to a new state.

The `Transition_To_State` variable must be defined on the form; for example:

```
var Transition_To_State = "next_state";
```

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"transition_problem";
   window.document.pt_submission.TEMPLATE_FLAG.value =
       "optional custom response template";
```

```
    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
    window.document.pt_submission.problem_number.value =
problem_number;
    window.document.pt_submission.crstatus.value =
Transition_To_State;
    window.document.pt_submission.modify_time.value =
modify_time;

    window.document.pt_submission.attribute_name1.value =
attribute_value1;
    window.document.pt_submission.attribute_name2.value =
attribute_value2;
    .
    .
    .
    window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
    <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <!-- PT Security Data -->
    <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
    <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

    <!-- Action Data -->
    <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"><INPUT>
    <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

    <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
    <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
    .
    .
    .
    <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

transition_problem_new_attachment

Transitions the change request and adds new attachment(s).

You must submit attachments using visible forms. Also, you must define the `Transition_to_State` variable for use on the form; e.g,:

```
var Transition_To_State = "my_next_state";
```

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
        "transition_problem_new_attachment";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";
    window.document.pt_submission.ATTACHMENT_RELATION.value =
        "relationship name of type CCM_PROBLEM_OBJECT";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
    window.document.pt_submission.problem_number.value =
problem_number;
    window.document.pt_submission.crstatus.value =
Transition_To_State;
    window.document.pt_submission.modify_time.value =
modify_time;

    window.document.pt_submission.attribute_name1.value =
attribute_value1;
    window.document.pt_submission.attribute_name2.value =
attribute_value2;
```

```
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Visible form

```
<FORM NAME="pt_submission" ENCTYPE="multipart/form-data">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="crstatus" TYPE=HIDDEN VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- Attachment Data -->
   <INPUT NAME="ATTACHMENT_RELATION" TYPE=HIDDEN
      VALUE="NO_ACTION"></INPUT>
   <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=0 BORDER=0>
      <TR ALIGN=left>
      <TD WIDTH="750">
      <B>Attachment Information</B>
      </TD>
      </TR>
   </TABLE>
```

```
    <TABLE WIDTH="750" CELLSPACING=0 CELLPADDING=10 BORDER=1>
        <TR>
        <TD WIDTH="730">
        <TABLE WIDTH="730" CELLSPACING=0 CELLPADDING=0 BORDER=0>
        <TR>
        <TD WIDTH="730">Attachment:&nbsp;
        <INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
        </INPUT>&nbsp;
        <INPUT NAME="_ATTACHMENT_IS_BINARY1"
        TYPE=CHECKBOX>Binary File</INPUT>
        <BR>Description:&nbsp;
        <INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
        MAXLENGTH=1024 SIZE=80></INPUT>
        </TD>
        </TR>
         .
         .
         .
        </TABLE>
        </TD>
        </TR>
    </TABLE>
</FORM>
```

transition_task

Transitions a task to a new state.

JavaScript function

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
"transition_task";
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.timestamp.value =
timestamp_value;

    // Action Data
```

```
   window.document.pt_submission.task_number.value
=task_number;
   window.document.pt_submission.status.value = "next state";
   window.document.pt_submission.modify_time.value =
modify_time;

   window.document.pt_submission.attribute_name1.value =
attribute_value1;
   window.document.pt_submission.attribute_name2.value =
attribute_value2;
   .
   .
   .
   window.document.pt_submission.attribute_nameN.value =
attribute_valueN;
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- Action Data -->
   <INPUT NAME="task_number" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="status" TYPE=HIDDEN VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="modify_time" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="attribute_name1" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="attribute_name2" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
   <INPUT NAME="attribute_nameN" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

update_user_listbox

Updates/overwrites the elements of a list box in the user's configuration file. This method also adds the list box if it does not exist, so it is similar to `add_user_listbox`.

JavaScript function

The delimited list of values will look something like the following:

```
"value1|value2|value3|value4"
```

The pipe (|) character is the default system delimiter, obtained on a template with the following wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_SECURITY_DATA VALUE=AdminPrefers -->
<!-- PARAM NAME=CCM_DELIMITER -->
<!-- PARAM NAME=CCM_SUB_DELIMITER -->
<!-- /WSLET -->
```

The `CCM_DELIMITER` parameter evaluates to the system default delimiter, which is the pipe character unless you have changed it. The `CCM_SUB_DELIMITER` parameter is the colon (:) character, and cannot be changed.

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"update_user_listbox";
   window.document.pt_submission.TEMPLATE_FLAG.value =
       "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CCM_DATALISTBOX.value =
       "user config data list box name";
   window.document.pt_submission.CCM_LISTBOX.value =
"listbox_name";
   window.document.pt_submission.VALUES.value = "delimited list
of values";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_DATALISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="CCM_LISTBOX" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="VALUES" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

update_user_profile

Updates or creates a user's profile file. The number of profile items created is variable.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"update_user_profile";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   // Profile Attribute Data
   window.document.pt_submission.profile_data_item.value =
      "profile_data_item_value";
   .
   .
   .
   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Profile Attribute Data -->
   <INPUT NAME="profile_data_item" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   .
   .
   .
</FORM>
```

update_user_profile_xml

Updates or creates a user's profile file. The number of profile items created is variable.

The expected XML format is:

```
<USER_PREFERENCES>
   <ATTRIBUTE>
      <NAME>name of profile/preference item</NAME>
      <VALUE>profile/preference value</VALUE>
      <TYPE>listbox|name-value</TYPE>
      <LISTBOX>if (TYPE=listbox) this is the listbox name</
LISTBOX>
   </ATTRIBUTE>
   .
   .
   .
</USER_PREFERENCES>
```

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method  = "POST";
   window.document.pt_submission.action  = "PTaction_servlet";
   window.document.pt_submission.target  = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value   =
"update_user_profile_xml";
```

```
   window.document.pt_submission.TEMPLATE_FLAG.value =
   "custom response template name"; OPTIONAL

   window.document.pt_submission.preferencesXML.value =
   "XML content of the user(s) data";

   window.document.pt_submission.role.value  = role;
   window.document.pt_submission.token.value = token;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG"    TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="TEMPLATE_FLAG"  TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="preferencesXML" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

update_user_query

Updates a query string and description in the user's configuration file.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"update_user_query";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CHOSEN_QUERY.value =
"query_name";
   window.document.pt_submission.QUERY_STRING.value =
      "new query string for the report's query object";
   window.document.pt_submission.DESCRIPTION.value =
      "new description for this report";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CHOSEN_QUERY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="DESCRIPTION" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

update_user_report

Updates a report query string and description in the user’s configuration file.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"update_user_report";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CHOSEN_REPORT.value =
"report_name";
   window.document.pt_submission.QUERY_STRING.value =
      "new query string for the report's query object";
   window.document.pt_submission.DESCRIPTION.value =
      "new description for this report";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CHOSEN_REPORT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="QUERY_STRING" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="DESCRIPTION" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

view_attachment

Downloads a single attachment to the browser.

Note To create attachments on a form, you must define two visible forms: one for the main form, and one for the attachment form.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "PTaction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"view_attachment";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;
    window.document.pt_submission.cvid.value = "attachment
object cvid";

   window.document.pt_submission.submit();
}
```

Note The HTML is part of a visible form because the input type is `FILE`.

For example:

```
<TD WIDTH="730">
Attachment:&nbsp;
<INPUT NAME="_ATTACHMENT_NAME1" TYPE=FILE SIZE=50>
</INPUT>&nbsp;
<INPUT NAME="_ATTACHMENT_IS_BINARY1"
TYPE=CHECKBOX>Binary File</INPUT>
<BR>
Description:&nbsp;
<INPUT NAME="_ATTACHMENT_COMMENT1" TYPE=TEXT
MAXLENGTH=1024 SIZE=80>
</INPUT>
</TD>
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="cvid" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

Relationship actions

Relationship actions define relationships between change requests (problems), tasks, and objects.

JavaScript function

You must supply a relationship action and other parameters to the function. The required parameters are shown in the following table.

Parameter	Description

`ACTION_FLAG`	Must be one of the following PTaction actions:
	`problem_problem`
	Creates a problem-to-problem relationship.
	`problem_task`
	Creates a problem-to-task relationship.
	`problem_object`
	Creates a problem-to-object relationship.
	`task_problem`
	Creates a task-to-problem relationship.
	`task_task`
	Creates a task-to-task relationship.
	`task_object`
	Creates a task-to-object relationship.
	`object_problem`
	Creates a object-to-problem relationship.
	`object_task`
	Creates a object-to-task relationship.
	`object_object`
	Creates a object-to-object relationship.

`FROM_OBJECT`	Specifies the `problem_number`, `task_number`, or `cvid` object from which the relationship is created.
`TO_OBJECT`	Specifies the `problem_number`, `task_number`, or `cvid` object to which the relationship is created.
`BOTH_WAY_RELATIONSHIP`	Indicates whether the relationship can go in either direction.
`RELATION_NAME`	Specifies the user-defined relationship name.

You can reference a *relation_name* using either has_*relation_name* (a "forward" relationship) or is_*relation_name*_of (a "backward" relationship).

For example, suppose you define a *relation_name* called associated_task using the *relation_type* called problem_task. In a PTDBListBox wslet call (page 281), you would use CCM_PROBLEM_TASK for the *relation_type*, as follows:

```
<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_PROBLEM_TASK VALUE="relation_name" -->
<!-- /WSLET -->
```

You would then use associated_task for the *relation_name* to reference all tasks associated with the current change request, or has_associated_task to reference all change requests associated with the current task.

```
function perform_action_OnClick()
{
    window.document.pt_submission.method = "POST";
    window.document.pt_submission.action = "PTaction_servlet";
    window.document.pt_submission.target = "frame_name";

    window.document.pt_submission.ACTION_FLAG.value =
relation_action;
    window.document.pt_submission.TEMPLATE_FLAG.value =
        "optional custom response template";

    window.document.pt_submission.role.value = role_value;
    window.document.pt_submission.token.value = token_value;

    window.document.pt_submission.RELATION_NAME.value =
relation_name;
    window.document.pt_submission.FROM_OBJECT.value =
from_object;
    window.document.pt_submission.TO_OBJECT.value = to_object;
    window.document.pt_submission.BOTH_WAY_RELATIONSHIP.value =
        both_way_relationship;

    window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="FROM_OBJECT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TO_OBJECT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="BOTH_WAY_RELATIONSHIP" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
</FORM>
```

7 Admin actions

Rational Change uses the PTAdminAction servlet to perform administrative operations. For example, PTAdminAction sends Rational Synergy role assignment updates to, and receives role assignments from, the Rational Synergy database.

See "Sending a request to a servlet" on page 22 for general information about the JavaScript functions and hidden forms used to perform the actions.

The following sections describe the actions you can perform using the PTAdminAction servlet.

Servlet action summary

The following table shows the PTAdminAction servlet actions.

Action	Description
auto_gen_transition_templates	Generates a CR Process package in the run area's packages directory.
clearlog	Clears the log file (for example, `event.log`).
loadconfig	Reloads the configuration files.
process_email_submissions	Processes problems submitted using email.
report_completion_check	Downloads the contents of the report as a file.
saveas_cs_report	Saves a report definition file with a new name.
save_cr_process	Sets or creates a CR Process XML file.
save_cs_report	Saves a report definition file.
save_listbox_configuration	Sets or creates a `pt_listbox.cfg` configuration file
set_system_date	Sets the date formats that are stored in the `pt.cfg` file.
set_user_data	Sets the database users and their roles.
set_cr_process_directory	Sets the path to the CR Process files directory.

Action	Description
set_system_info	Sets the Rational Synergy `pt_app` and `pt_app_role` attributes for a database.
toggledebug	Toggles the debug flag ON and OFF.
viewlog	Displays the `event.log` file.
viewlogfile	Downloads the log file to save to disk.

Servlet action descriptions

The following sections describe individual PTAdminAction servlet actions in detail.

Note that for Admin actions, `window.document.pt_submission.action` is always set to "*PTAdminAction_servlet*". "*PTAdminAction_servlet*" is an abbreviation for the following string:

```
"<!-- WSLET CODE=PTAdmin --><!-- PARAM NAME=SERVLET_PATH
VALUE=PTADMINACTION_SERVLET --><!-- /WSLET -->";
```

auto_gen_transition_templates

Generates a CR Process package in the run area's packages directory.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
      "auto_gen_transition_templates";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CR_PROCESS_XML.value =
      "file name in which to create a CR Process package";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
```

```
   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CR_PROCESS_XML" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

clearlog

Clears the log file (for example, `event.log`).

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"clearlog";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"delete_user_listbox";
```

loadconfig

Reloads the configuration files.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"loadconfig";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

process_email_submissions

Processes all the email change request submissions when [SEND_DATA] is set to `FALSE` in a configuration file. Enables users without a network login to submit change requests.

You must configure the email submission section in a configuration file CCM_SYSTEM section before using this feature.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
      "process_email_submissions";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

report_completion_check

Determines whether the report is completed.

If the report is completed, the report is downloaded (using PTAlert RELOAD). Otherwise, a custom response template (specified by `TEMPLATE_FLAG`) is returned. The file type is based on the [EXPORT_FORMAT] of the report.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value =
"report_completion_check";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CCM_FILE.value =
      "name of report file to retrieve";
   window.document.pt_submission.REPORT_REPORTNAME.value =
      "report_name as defined in a [CCM_REPORT][/
CCM_REPORT]";
   window.document.pt_submission.REPORT_FORM_TYPE.value =
      "report type as defined by the [EXPORT_FORMAT]
setting for this report";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CCM_FILE" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="REPORT_REPORTNAME" TYPE=HIDDEN
      VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="REPORT_FORM_TYPE" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

saveas_cs_report

Saves the report definition file that was defined using the report builder in the `cs_reports` directory with a new name. See save_cs_report for information.

save_cr_process

Sets or creates a CR process XML file. The file is saved to the location specified by set_cr_process_directory.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"save_cr_process";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CR_PROCESS_XML.value =
      "file name in which to save the XML content";
   window.document.pt_submission.XML_CONTENT.value =
      "XML content of a CR process";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CR_PROCESS_XML" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

save_cs_report

Saves the report definition file that was defined using the report builder in the `cs_reports` directory.

JavaScript function

```
    function perform_action_OnClick()
    {
        window.document.pt_submission.method  = "POST";
        window.document.pt_submission.action  =
"PTAdminAction_servlet";
        window.document.pt_submission.target  = "frame_name";

        window.document.pt_submission.ACTION_FLAG.value   =
        "save_cs_report | saveas_cs_report";

        window.document.pt_submission.TEMPLATE_FLAG.value =
        "your custom response template name"; OPTIONAL!

        window.document.pt_submission.XML_CONTENT.value
=
        "The XML content of the report definition";
        window.document.pt_submission.CS_REPORT_XML.value
=
        "The file name to save as";
        window.document.pt_submission.CS_REPORT_INSTALL.value
= "false";
        window.document.pt_submission.CS_REPORT_UNINSTALL.value
= "false";

        // if save_cs_report

window.document.pt_submission.CS_REPORT_ALLOWOVERWRITE.value =
"true";

        // if saveas_cs_report

window.document.pt_submission.CS_REPORT_ALLOWOVERWRITE.value =
"false";

        window.document.pt_submission.role.value  = role;
        window.document.pt_submission.token.value = token;

        window.document.pt_submission.submit();
    }
```

Hidden form

```
   <FORM NAME="pt_submission">
      <!-- PT Action Information -->
      <INPUT NAME="ACTION_FLAG"   TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

      <!-- Action Data -->
      <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="CS_REPORT_XML" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="CS_REPORT_INSTALL" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="CS_REPORT_UNINSTALL"  TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="CS_REPORT_ALLOWOVERWRITE" TYPE=HIDDEN
VALUE="NO_ACTION">
      </INPUT>

      <!-- PT Security Data -->
      <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   </FORM>
```

save_listbox_configuration

Sets or creates a `pt_listbox.cfg` configuration file.

After list boxes are in this file, you cannot override them using [CFG_MERGE].

Note Deleting this file is comparable to restoring the defaults.

The system configuration files are loaded in the following order (lowest to highest precedence):

```
init.cfg
pt.cfg
admin_framework.cfg
user_framework.cfg
task_framework.cfg
```
(if tasks are supported)
```
pt_listbox.cfg
```

The standard override behavior exists for the system configuration files. List boxes of type SYSTEM are always overridden regardless of the configuration file in which they reside.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
      "save_listbox_configuration";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.XML_CONTENT.value =
      "list box configuration data to save (action-
specific string)";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

set_user_data

Applies the user changes (update read security and change database privileges) specified in the XML. The XML is generated by the User Administration dialog. This information is used to update or create a user's profile file. The number of profile items created is variable.

Note This action replaces *set_ccm_users*, which is no longer supported.

The expected XML format is:

```
<CS_USERS>
      <LDAP_USERS>
      <LDAP_USER>
      <UID>user login name</UID>
      <READ_SECURITY_VALUE>user's read security value|
      empty string means no read security for this user
      </READ_SECURITY_VALUE>
      </LDAP_USER>
      .
      .
      </LDAP_USERS>
      <DB_USERS>
      <DB_USER>
      <UID>user login name</UID>
      <DATABASE>Rational Synergy database path</DATABASE>
      <ROLES>database privilege list,</ROLES>
      </DB_USER>
      .
      .
      </DB_USERS>
   </CS_USERS>
```

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"set_user_data";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";
   window.document.pt_submission.XML_CONTENT.value =
      "all user data in XML format";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
<!-- PT Action Information -->
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="XML_CONTENT" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

set_cr_process_directory

Sets the path to the CR Process files directory.

The default is the run area's `cr_process` directory. The setting is valid only until the servlet runner is re-started.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"set_cr_process_directory";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.CR_PROCESS_PATH.value =
      "complete, platform-dependent system path to a
directory";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="CR_PROCESS_PATH" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

set_system_date

Sets the date formats that are stored in the `pt.cfg` file.

Note The values sent with this action must be a single string without newline characters and must be formatted as follows:

```
   [SUBMIT_DATE_FORMAT]submit date format[/
SUBMIT_DATE_FORMAT]
   [SHOW_DATE_FORMAT]show date format[/SHOW_DATE_FORMAT]
   [QUERY_DATE_FORMAT]query date format[/QUERY_DATE_FORMAT]
   [REPORTS_DATE_FORMAT]report date format[/
REPORTS_DATE_FORMAT]
   [API_DATE_FORMAT]api date format[/API_DATE_FORMAT]
```

JavaScript function

```
   function perform_action_OnClick()
   {
      window.document.pt_submission.method  = "POST";
      window.document.pt_submission.action =
"PTAdminAction_servlet";
      window.document.pt_submission.target = "frame_name";

      window.document.pt_submission.ACTION_FLAG.value   =
"set_system_date";
      window.document.pt_submission.TEMPLATE_FLAG.value =
      "custom response template name"; OPTIONAL

      window.document.pt_submission.pt_date_data.value =
"value (see note);

      window.document.pt_submission.role.value  = role;
      window.document.pt_submission.token.value = token;

      window.document.pt_submission.submit();
   }
```

Hidden form

```
   <FORM NAME="pt_submission">
      <!-- PT Action Information -->
      <INPUT NAME="ACTION_FLAG"    TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
      <INPUT NAME="TEMPLATE_FLAG"  TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

      <!-- Action Data -->
      <INPUT NAME="pt_date_data"   TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

      <!-- PT Security Data -->
      <INPUT NAME="role"           TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
```

```
       <INPUT NAME="token"           TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   </FORM>
```

set_system_info

Sets the Rational Synergy `pt_app` and/or `pt_app_role` attributes for a database.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"set_system_info";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.pt_app.value =
      "HTTP address to the web server";
```

and/or

```
   window.document.pt_submission.pt_app_role.value = "user's
default role";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <!-- Action Data -->
   <INPUT NAME="pt_app" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="pt_app_role" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

toggledebug

Toggles the debug flag ON and OFF.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"toggledebug";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

viewlog

Displays the `event.log` file in a browser window.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value = "viewlog";
   window.document.pt_submission.TEMPLATE_FLAG.value =
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

viewlogfile

Downloads the `event.log` file. The user may save the log file to disk.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action =
"PTAdminAction_servlet";
   window.document.pt_submission.target = "frame_name";
   window.document.pt_submission.ACTION_FLAG.value = 
"viewlogfile";
   window.document.pt_submission.TEMPLATE_FLAG.value = 
      "optional custom response template";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN 
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN 
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

8 Miscellaneous actions

Rational Change uses miscellaneous servlets to perform installation and administration actions.

For example, the WsUpdate servlet install_package action installs the selected package into the specified packages directory.

See “Sending a request to a servlet” on page 22 for general information about the JavaScript functions and hidden forms used to perform the actions.

The WSUpdate servlet manages Rational Change package installation. Package installation enables you to install Rational-supplied packages and patches, and customizations, using the GUI.

Package hierarchy

A package has the following directory structure:

```
package_name
|--README.txt
|--VERSION.txt
|--cr_process
|--trapeze
|   |--help files and directories
|   |   |--images
|   |   |--redirect
|   |   |--role_name(zero or more)
|   |      |--images
|   |      |--wwhgifs
|   |--images
|   |--ptimages
|   |--servlet
|      |<java_package_stucture> includes java class files
|--wsconfig
|--acl
|--messages
|--scripts
|--lib
|acl
|listbox
|--system
|index
|--templates
|--pt
|   |--etc
|   |   |--package_name
|   |--forms
|   |   |--package_name
|   |--include
|   |   |--package_name
|   |--reports
|   |   |--package_name
|   |--attr_controls
|   |--custom_controls
|   |--styles
|--tempdir
|--triggers
```

Servlet information

The following table shows the WsUpdate servlet actions.

Action	Description
install_package	Installs a Rational Change package.
package_form	Loads the package installation template.
set_package_directory	Sets the packages directory location.
set_template_directory	Sets the template packages directory location.
uninstall_package	Uninstalls a Rational Change package.
view_install_log	Displays a package installation log.

Note that for WsUpdate actions, `window.document.pt_submission.action` is set to `"`*`WsUpdate_servlet`*`"`. `"`*`WsUpdate_servlet`*`"` is an abbreviation for the following string:

```
    "<!-- WSLET CODE=PTAdmin --><!-- PARAM NAME=SERVLET_PATH
VALUE=PTUPDATE_SERVLET --><!-- /WSLET -->";
```

install_package

Installs a Rational Change package.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"install_package";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.PACKAGE_NAME.value =
      "name of a directory in the packages directory";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="PACKAGE_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

package_form

Loads the package installation template, `admin_framework/admin_package_install.html`.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"package_form";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
</FORM>
```

set_package_directory

Sets the package directory location.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"set_package_directory";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.PACKAGE_DIRECTORY.value =
      "packages directory on server";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="PACKAGE_DIRECTORY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

set_template_directory

Sets the template packages directory location.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"set_template_directory";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.PACKAGE_DIRECTORY.value =
      "template packages directory on server";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="PACKAGE_DIRECTORY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

uninstall_package

Uninstalls a Rational Change package.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"uninstall_package";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.PACKAGE_NAME.value =
"package_name";
   window.document.pt_submission.BACKUP_DIRECTORY.value =
"backup_dir_name";
   window.document.pt_submission.ROLE_LIST.value =
"role_list_name";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="PACKAGE_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="BACKUP_DIRECTORY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <INPUT NAME="ROLE_LIST"   TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

view_install_log

Displays a package installation log.

JavaScript function

```
function perform_action_OnClick()
{
   window.document.pt_submission.method = "POST";
   window.document.pt_submission.action = "WsUpdate_servlet";
   window.document.pt_submission.target = "frame_name";

   window.document.pt_submission.ACTION_FLAG.value =
"view_install_log";

   window.document.pt_submission.role.value = role_value;
   window.document.pt_submission.token.value = token_value;

   window.document.pt_submission.PACKAGE_NAME.value =
"package_name";
   window.document.pt_submission.BACKUP_DIRECTORY.value =
"backup_dir_name";

   window.document.pt_submission.submit();
}
```

Hidden form

```
<FORM NAME="pt_submission">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>

   <!-- Action Data -->
   <INPUT NAME="PACKAGE_NAME" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
   <INPUT NAME="BACKUP_DIRECTORY" TYPE=HIDDEN
VALUE="NO_ACTION"><INPUT>
</FORM>
```

9 Wslets

A *wslet* is a piece of Rational Change Java code that runs on the server and produces string output on a form.

The following sections describe the Rational Change wslets.

Note Some wslets have multiple uses. Use the "Wslets index" on page 467 to locate wslets based on their uses, as indicated by their required and optional parameters.

Wslet summary

The following table shows the Rational Change wslets.

Action	Description
CSAttribute	Accesses properties of CR/Task attributes
CSChartDispatcher	Invokes the wslet specified by [CUSTOM_WSLET] (page 73).
CSChartMultipleDateTrend	Creates the pre-defined report, "Trend Analysis".
CSChartOpenVsClosedTrend	Creates the pre-defined report, "Open vs Closed Trend Analysis".
CSChartSingleDateTrend	Creates the pre-defined report, "Trend Analysis with Value Breakdown".
GetSystemValue	Gets the value from any simple CCM_SYSTEM setting in the `pt.cfg` file.
PTAdmin	Used for administrative functions.
PTAlert	Generates HTML that displays Rational Change-generated status messages.
PTCRProcessInfo	Retrieves information from the installed CR Process file.
PTDataListBox	Generates a JavaScript data structure using a CCM_DATALISTBOX item defined in a configuration file.

Action	Description
PTDBListBox	Builds relationship select options in list boxes on templates.
PTDBReport	Displays database object relationships such as "duplicate" in a report format.
PTGenerate	Constructs transition templates.
PTInclude	Includes the content of other files and the output of JavaScript files.
PTListBox	Builds select options in list boxes on templates.
PTLoadData	Retrieves data from the database.
PTReport	Retrieves report information.
PTString	Retrieves attribute aliases, role-based text, and supplemental data sent in with a request.
PTText	Fetches an attribute value.
PTToggle	Sets HTML elements of type `CHECKBOX`.
PTValueListBox	Builds the select options for a list box that has both labels and values.

Wslet descriptions

The following sections describe individual wslets in detail.

CSAttribute

CSAttribute accesses properties of CR/Task attributes. The properties come from both the database lookup and/or PTInclude Wslet substitution parameters. CSAttribute Wslet calls are used exclusively on attribute controls.

Note The value that is returned from the Wslet is parsed by the template parser, and must follow all the rules of being parsable text.

Recursion of CSAttribute Wslet calls is supported.

The following attribute properties are accessible:

IS_REQUIRED:	Is a required attribute.
IS_READONLY:	Is a read-only attribute.
NOT_READONLY:	Is not a read-only attribute.
IS_OPTIONAL:	Is an optional attribute.
IS_MODIFIABLE:	Is a modifiable attribute.

Use

This wslet is used to look up attribute properties.

Required parameters

IS_REQUIRED
or

IS_READONLY
or

NOT_READONLY
or

IS_OPTIONAL
or

IS_MODIFIABLE
VALUE = *`SUB_ATTR`*: Can be the substitution identifier for the attribute name,

where it gets its value from the attribute control load Wslet call (PTInclude), or a hard-coded value.

IS_REQUIRED VALUE=`true|false`

or

IS_READONLY VALUE=`true|false`

or

NOT_READONLY
or

IS_OPTIONAL
or

IS_MODIFIABLE The VALUE parameter is `true` or `false` value which states that the attribute is or is not equal to the NAME action. This is useful when combined with the substitution functionality. VALUE =`true|false`

IF_TRUE This option is evaluated if the Wslet action is determined to be true. VALUE= "*the text to parse if Wslet action is true*"

IF_FALSE This option is evaluated if the Wslet action is determined to be false. VALUE = "*the text to parse if Wslet action is false*"

The VALUE parameter is a block of text that is parsed and substituted in place of the entire CSAttribute Wslet call. The value should be wrapped with quotations, either a "" pair, or a '' pair. Quotations within the body of text must be the opposite of the pair chosen for the VALUE.

Optional parameters

The following optional arguments are a replacement (over-ride) for the IF_TRUE case. The IF_FALSE condition cannot be substituted.

IF_SHOW This option is evaluated if the form type is a SHOW form. VALUE = "*text to parse if the form type is a SHOW form*"

IF_SUBMIT This option is evaluated if the form type is a SUBMIT form. VALUE = "*text to parse if the form type is a SUBMIT form*"

IF_TRANSITION This option is evaluated if the form type is a TRANSITION form. VALUE = "*`text to parse if the form type is a TRANSITION form`*"

IF_QUERY This option is evaluated if the form type is a QUERY form. VALUE = "*`text to parse if the form type is a QUERY form`*"

IF_PREF This option is evaluated if the form type is a PREF form. VALUE = "*`text to parse if the form type is a PREF form`*"

The evaluation precedence order for the above optional arguments is: SHOW->SUBMIT->TRANSITION->QUERY->PREF. (highest to lowest precedence).

These options are dependent on the global parameter: **SUB_TYPE_OF_FORM**, where **SUB_TYPE_OF_FORM** must equal: [**SUBMIT|SHOW|TRANSITION|QUERY|PREF**].

RELATION_ATTR The relationship attribute name, either hard-coded or passed using substitution.

If this Wslet option is used, the **RELATION_LIST** option is required also. VALUE = "*`SUB_ATTR`*"

RELATION_LIST A delimited list of copy dialog definitions. The VALUE can be either the **SUB_RELATION_FORM_NAME** substitution identifier or a hard-coded list of copy dialog definitions. VALUE = "*`SUB_RELATION_FORM_NAME`*"

The format for the VALUE is:
`"copy_template_name:copy_label|copy_template_name:copy_label|..."`

RELATION_LINKS This is a special substitution identifier. Auto-generated javascript is substituted in place of the identifier. The delimiter sequence between multiple links is `&"&nbsp;|&nbsp;"`.

COPY_LINKS This is a special substitution identifier. Auto-generated javascript is substituted in place of the identifier. The delimiter sequence between multiple links is `&"&nbsp;"`.

Example

- Control load Wslet call for “has_duplicate” relation attribute.

Uses the "attr_controls/has_duplicate.CCM_RELATION" attribute control.

```
<!-- WSLET CODE=PTInclude -->
<!-- PARAM NAME='attr_controls/has_duplicate.CCM_RELATION' -->
<!-- PARAM NAME=SUB_ATTR VALUE='has_duplicate' -->
<!-- PARAM NAME=SUB_FORM VALUE='modify' -->
<!-- PARAM NAME=SUB_RELATION_FORM_NAME VALUE='' -->
<!-- PARAM NAME=SUB_RELATION_NAME VALUE='NO_ACTION' -->
<!-- PARAM NAME=SUB_COLUMN_COUNT VALUE=3 -->
<!-- PARAM NAME=SUB_TYPE_OF_FORM VALUE='SHOW' -->
<!-- /WSLET -->
```

A check for the ″has_duplicate″ attribute being read-only. If the attribute is read-only, the user only has the ability to view items in the listbox. If the attribute is not read-only, the user has the ability to view, associate, and disassociate CRs with the “has_duplicate” relation. Newlines are added for readability.

```
<!-- WSLET CODE=CSAttribute -->
<!-- PARAM NAME=IS_READONLY VALUE='SUB_ATTR' -->
<!-- PARAM NAME=IF_TRUE VALUE=″<A
HREF='javascript:SUB_ATTR_show_OnClick()'>Show</A>″ -->
<!-- PARAM NAME=IF_FALSE VALUE=″<A
HREF='javascript:SUB_ATTR_show_OnClick()'>Show</A>
         <BR>
       <INPUT NAME='PROBLEM_PROBLEM_SUB_ATTR' TYPE=TEXT
MAXLENGTH=255 SIZE=6>
       </INPUT>&nbsp;
       <A HREF='javascript:SUB_ATTR_associate_OnClick()'>
       <IMG SRC='<!-- WSLET CODE=PTAdmin -->
                  <!-- PARAM NAME=SERVLET_PATH
VALUE=PTIMAGES_PATH -->
       <!-- /WSLET -->ws_associate.gif' ALIGN=top BORDER=0>
       </A>&nbsp;
       <A HREF='javascript:SUB_ATTR_disassociate_OnClick()'>
        <IMG SRC='<!-- WSLET CODE=PTAdmin -->
              <!-- PARAM NAME=SERVLET_PATH VALUE=PTIMAGES_PATH -
->
       <!-- /WSLET -->ws_disassociate.gif' ALIGN=top BORDER=0>
       </A>" -->
<!-- /WSLET -->
```

Notice that the surrounding quotation marks for the VALUE are "", but internal to the block of text, only '' single quotes are used. This is a requirement for imbedding internal quotations.

- Control load Wslet call for "has_child_cr" relation attribute.

Uses the "attr_controls/base.CCM_RELATION" attribute control.

Note Newlines are added for readability.

```
<!-- WSLET CODE=PTInclude -->
<!-- PARAM NAME='attr_controls/base.CCM_RELATION' -->
<!-- PARAM NAME=SUB_ATTR VALUE='has_child_cr' -->
<!-- PARAM NAME=SUB_FORM VALUE='modify' -->
<!-- PARAM NAME=SUB_RELATION_FORM_NAME
       VALUE='COPY_has_child_cr2assigned:Assign Parent A|
      COPY_has_child_cr2entered:Enter Parent A|
      COPY_has_child_cr2assigned_d_:Assign Parent B|
      COPY_has_child_cr2entered_a_:Enter Parent B|
      COPY_has_child_cr2assigned_d__p_:Assign Parent C|
      COPY_has_child_cr2entered_a__j_:Enter Parent C' -->
<!-- PARAM NAME=SUB_RELATION_NAME VALUE='NO_ACTION' -->
<!-- PARAM NAME=SUB_COLUMN_COUNT VALUE=3 -->
<!-- PARAM NAME=SUB_TYPE_OF_FORM VALUE='SHOW' -->
<!-- /WSLET -->
```

A check to see if the "`has_child_cr`" is not read-only. If the attribute is not read-only, the user has the ability to view, associate, and disassociate CRs with the "`has_child_cr`" relation, and create new CRs with all of the dialogs listed in the "SUB_RELATION_FORM_NAME" parameter. If the attribute is read-only, the user has the ability only to view items in the listbox.

```
<!-- WSLET CODE=CSAttribute -->
<!-- PARAM NAME=NOT_READONLY -->
<!-- PARAM NAME=RELATION_LIST VALUE="SUB_RELATION_FORM_NAME" --
>
<!-- PARAM NAME=RELATION_ATTR VALUE="SUB_ATTR" -->
<!-- PARAM NAME=IF_TRUE VALUE="<A
HREF='javascript:SUB_ATTR_show_OnClick()
       '>Show</A>
                             <RELATION_LINKS>
         <BR>
         <INPUT NAME='PROBLEM_PROBLEM_SUB_ATTR' TYPE=TEXT
       MAXLENGTH=255 SIZE=6>
         </INPUT>&nbsp;
         <A HREF='javascript:SUB_ATTR_associate_OnClick()'>
           <IMG SRC='<!-- WSLET CODE=PTAdmin -->
                 <!-- PARAM NAME=SERVLET_PATH
       VALUE=PTIMAGES_PATH -->
         <!-- /WSLET -->ws_associate.gif'
       ALIGN=top BORDER=0>
         </A>&nbsp;
```

```
                   <A HREF='javascript:SUB_ATTR_disassociate_OnClick
              ()'>
                       <IMG SRC='<!-- WSLET CODE=PTAdmin -->
                             <!-- PARAM NAME=SERVLET_PATH
               VALUE=PTIMAGES_PATH -->
                   <!-- /WSLET -->ws_disassociate.gif'
               ALIGN=top BORDER=0>
                   </A>" -->
<!-- PARAM NAME=IF_FALSE VALUE="<A
HREF='javascript:SUB_ATTR_show_OnClick()
               '>Show</A>" -->
<!-- /WSLET -->
```

CSChartDispatcher

CSChartDispatcher is used in subreport templates to invoke the wslet specified by [CUSTOM_WSLET] in the configuration file.

For example, the following is a trend analysis report definition with a custom wslet, `CSChartOpenVsClosedTrend`:

```
[CCM_PROBLEM]
    [NAME]open_vs_closed_trend[/NAME]
    [MAIN_TEMPLATE]user_framework/chart_rpt.html[/MAIN_TEMPLATE]
    [HDR_TEMPLATE]user_framework/common_minus_query_hdr.html
    [/HDR_TEMPLATE]
    [IMG_TEMPLATE]user_framework/cs_chart_common_img.html
    [/IMG_TEMPLATE]
    [FTR_TEMPLATE]user_framework/common_ftr.html
    [/FTR_TEMPLATE]
    [ATTRS][/ATTRS]
    [CUSTOM_WSLET]CSChartOpenVsClosedTrend[/CUSTOM_WSLET]
[/CCM_PROBLEM]
```

Note that CSChartDispatcher appears in the image template, `cs_chart_common_img.html`, which was part of the report definition:

```
<BR>
<TABLE WIDTH="650" BORDER=0>
    <TR ALIGN=center>
        <TD WIDTH="650" ALIGN=center VALIGN=center>
        <!-- WSLET CODE=CSChartDispatcher --><!-- /WSLET -->
        </TD>
    </TR>
</TABLE>
```

Because CSChartDispatcher is included in the template, Rational Change executes the CSChartOpenVsClosedTrend wslet.

Also, although the subreport entry in the configuration file does not specify [XML_CONTENT], XML content is sent (dynamically) from the Web browser, and CSChartOpenVsClosedTrend interprets the XML data and builds a report from these instructions and the query results.

Use

Reduces the number of necessary templates and is available only in the following report templates: header, footer, and image.

Required parameters

None.

Optional parameter

ANY.

Any parameters specified with CSChartDispatcher are passed on to the invoked wslet.

Example

The following is a trend analysis report definition with a custom wslet, `CSChartOpenVsClosedTrend`:

```
[CCM_PROBLEM]
    [NAME]trend[/NAME]
    [MAIN_TEMPLATE]user_framework/chart_rpt.html[/MAIN_TEMPLATE]
    [HDR_TEMPLATE]user_framework/common_minus_query_hdr.html
    [/HDR_TEMPLATE]
    [IMG_TEMPLATE]user_framework/cs_chart_common_img.html
    [/IMG_TEMPLATE]
    [FTR_TEMPLATE]user_framework/common_ftr.html[/FTR_TEMPLATE]
    [ATTRS][/ATTRS]
    [CUSTOM_WSLET]CSChartMultipleDateTrend[/CUSTOM_WSLET]
    [XML_CONTENT][/XML_CONTENT]
[/CCM_PROBLEM]
```

Note that CSChartDispatcher appears in the image template, `cs_chart_common_img.html`, which was part of the report definition:

```
<!-- WSLET CODE=CSChartDispatcher -->
<!-- PARAM NAME=HEIGHT VALUE=50 -->
<!-- /WSLET -->
```

Because CSChartDispatcher is included in the template, Rational Change executes the CSChartOpenVsClosedTrend wslet, with the following Output:

```
<!-- WSLET CODE=CSChartMultipleDateTrend -->
<!-- PARAM NAME=HEIGHT VALUE=50 -->
<!-- /WSLET -->
```

Note that multiple subreport definitions could therefore share the same image template while invoking different wslets (based on the subreport definition) on that template.

CSChartMultipleDateTrend

CSChartMultipleDateTrend creates the pre-defined report, "Trend Analysis" by performing computations on the query results and producing a chart.

CSChartMultipleDateTrend is invoked indirectly by CSChartDispatcher and is given a wslet-specific XML string that controls its behavior.

Use

This wslet is highly specialized and is intended only for use in the "Trend Analysis" report.

Required parameters

None.

Optional parameters

None.

CSChartOpenVsClosedTrend

CSChartOpenVsClosedTrend creates the pre-defined report, "Open vs Closed Trend Analysis," by performing computations on the query results and producing a chart.

CSChartOpenVsClosedTrend is invoked indirectly by CSChartDispatcher and is given a wslet-specific XML string that controls its behavior.

Use

This wslet is highly specialized and is intended only for use in the "Open vs Closed Trend Analysis" report.

Required parameters

None.

Optional parameters

None.

CSChartSingleDateTrend

CSChartSingleDateTrend creates the pre-defined report, "Trend Analysis with Value Breakdown". by performing computations on the query results and producing a chart.

CSChartSingleDateTrend is invoked indirectly by CSChartDispatcher and is given a wslet-specific XML string that controls its behavior.

Use

This wslet is highly specialized and is intended only for use in the "Trend Analysis with Value Breakdown" report.

Required parameters

None.

Optional parameters

None.

GetMirroredAttributes

GetMirroredAttributes gets information about all of the mirrored attributes defined in the system.

Use

This wslet gets the value from the `pt.cfg` file. The return value is a JSON data structure of all the mirrored attributes.

Wslet call:

```
<!-- WSLET CODE=GetMirroredAttributes -->
<!-- /WSLET -->
```

Output:

```
[{"delimiter":",","queryString":"is_associated_child_of('%insta
nce/problem/problem%problem_number/
1')","type":"CR","attribute":"child_special","name":"Child
Info"},{"delimiter":",","queryString":"is_associated_child_of('
%instance/problem/problem%problem_number/
1')","type":"CR","attribute":"severity","name":"Child
Severity"}]
```

Required parameters

None.

Optional parameters

None.

GetSystemValue

GetSystemValue gets the value from any simple (single node) *CCM_SYSTEM* setting in the `pt.cfg` file.

Use

This wslet gets the value from the `pt.cfg` file.

Wslet call:

```
<!-- WSLET CODE=GetSystemValue -->
<!-- PARAM NAME=NAME VALUE=ALL_USERS_THRESHOLD -->
<!-- PARAM NAME=DEFAULT VALUE=250 -->
<!-- /WSLET -->
```

Output (if value exists and is 500):

```
500
```

Required parameters

NAME Name of the system value to get.

Optional parameters

DEFAULT

Shows the default value if the NAME matching value is not found.

PTAdmin

The PTAdmin wislet is used for administrative operations, such as operations related to roles, security, and so on.

Use 1

CCM_USER_ROLE_LIST

The CCM_USER_ROLE_LIST parameter returns the current user list of Rational Synergy roles.

Wslet call:

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=CCM_USER_ROLE_LIST -->
<!-- /WSLET -->
```

Output:

```
var user_roles = unescape("developer|build_mgr");
```

This wslet is used in conjunction with customizing role privileges. The deletion of shared reports and operations on non-personal folders require either the process administrator CM role, or a customizable CM role that defaults to *build_mgr*.

To change the latter CM role, edit the file :

`<CS-HOME>\<context>\webapps\synergy\WEB-INF\wsconfig\templates\pt\include\queryAndReportPriviieges.js`

Find the line:

```
var sharedRole= "build_mgr";
```

Replace `"build_mgr"` with the desired CM role.

Use 1 required parameters

None.

Use 1 optional parameters

None.

Use 2

SERVLET_PATH

The SERVLET_PATH parameter returns the URL of a servlet or a server directory.

Wslet call:

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=SERVLET_PATH VALUE=PTWEB_SERVLET -->
<!-- /WSLET --> <
```

Output:

```
http://192.168.10.103:8613/PTweb
```

Use 2 required parameters

SERVLET_PATH

One of the following constants must be used:

PTWEB_SERVLET

PTACTION_SERVLET

PTADMINACTION_SERVLET

PTUPDATE_SERVLET

PTSTART_SERVLET

PTIMAGES_PATH

IMAGES_PATH

TRAPEZE_PATH

PTSCRIPTS_PATH

Use 2 optional parameters

None.

Use 3

USER_ON_CENTRAL_DATABASE

The USER_ON_CENTRAL_DATABASE parameter determines if the user is on the central database.

Wslet call:

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=USER_ON_CENTRAL_DATABASE -->
<!-- /WSLET -->
```

Output:

```
true
```

Use 3 required parameters

None.

PTAlert

PTAlert generates HTML that displays Rational Change-generated status messages on a response template.

Use this wslet on workspace and status forms for PTaction and PTAdminAction responses. The response template is specified using the TEMPLATE_FLAG parameter.

Use 1

ACTION_STATUS

The ACTION_STATUS parameter returns a boolean action status.

Wslet call:

```
<!-- WSLET CODE=PTAlert -->
<!-- PARAM NAME=ACTION_STATUS -->
<!-- /WSLET -->
```

Use 1 required parameters

None.

Use 1 optional parameters

None.

Use 1 example

Retrieve an action status.

```
<!-- WSLET CODE=PTAlert -->
<!-- PARAM NAME=ACTION_STATUS -->
<!-- /WSLET -->
```

Output:

```
var action_status = true;
```

or

```
var action_status = false;
```

Use 2

CCM_ALERT

Pops up a message box, in the form of the following JavaScript variable:

```
var strMessage = "result";
```

Note This parameter requires that you include `ptalert.js` on the form.

By default, the message appears in the alert box; however, you can suppress the message using the `NO_SHOW` option. (The JavaScript variable is preserved.)

Use 2 required parameters

None.

Use 2 optional parameters

ALERT_ICON Specifies the alert icon to use on the pop-up message.

When you use a PTAlert(strMessage,#), which is the default, the # must be1, 2, or 3. The possible VALUEs are: [0]=success, [1]=error, and [2]=info. The default is 2.

NO_FORMATTING Prevents the message from wrapping. (By default, <PRE></PRE> wraps the message).

NO_SHOW Suppresses the pop-up message but leaves the message in the HTML, enabling the user to customize the message before displaying it.

Use 2 examples

- Create an alert.

  Wslet call:

  ```
  <!-- WSLET CODE=PTAlert -->
  <!-- PARAM NAME=CCM_ALERT -->
  <!-- /WSLET -->
  ```

  Output:

  ```
  var strMessage = "message"
  PTAlert(strMessage,2);
  ```

- Suppress a pop-up alert.

  Wslet call:

  ```
  <!-- WSLET CODE=PTAlert -->
  <!-- PARAM NAME=CCM_ALERT -->
  <!-- PARAM NAME=NO_SHOW -->
  ```

```
<!-- /WSLET -->
```

Output:

```
var strMessage = "message"
```

Use 3

CCM_MESSAGE

Retrieves the status message for an action. The template builder may either assign it to a variable or display the result in an HTML page.

Use 3 required parameters

None.

Use 3 optional parameter

TRUNCATE Causes the resultant string to be truncated to the (integer) length specified by `VALUE`.

Use 3 examples

- Truncate a message to 80 characters.

  Wslet call:

```
<TABLE WIDTH="100%" CELLSPACING=0 CELLPADDING=0 BORDER=0>
   <TR ALIGN=center>
   <TD ALIGN=left VALIGN=center>
   <!-- WSLET CODE=PTAlert -->
   <!-- PARAM NAME=CCM_MESSAGE -->
   <!-- PARAM NAME=TRUNCATE VALUE=80 -->
   <!-- /WSLET -->
   </TD>
   </TR>
</TABLE>
```

  Output:

```
<TABLE WIDTH="100%" CELLSPACING=0 CELLPADDING=0 BORDER=0>
   <TR ALIGN=center>
   <TD ALIGN=left VALIGN=center>message</TD>
   </TR>
</TABLE>
```

Use 4

CCM_RELOAD

The CCM_RELOAD parameter causes a clock to start and is used to count down to an event. The template builder implements the methods.

Wslet call:

```
<!-- WSLET CODE=PTAlert -->
<!-- PARAM NAME=CCM_RELOAD -->
<!-- /WSLET -->
```

Use 4 required parameters

None.

Use 4 optional parameters

None.

Use 4 example

Start a clock.

Wslet call:

```
<!-- WSLET CODE=PTAlert -->
<!-- PARAM NAME=CCM_RELOAD -->
<!-- /WSLET -->
```

Output:

```
StartClock();
```

PTCRProcessInfo

PTCRProcessInfo retrieves information from the installed CR Process (*`crprocess_name`*`.xml`) file.

Use 1

CCM_CR_PROCESS_INFO=IS_ADMIN

Returns the type of Admin transition the current user can perform.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
<!-- PARAM NAME=CCM_CR_PROCESS_INFO VALUE=IS_ADMIN -->
<!-- /WSLET -->
```

Output:

```
var Is_Admin = 0;
```

or

```
var Is_Admin = 1;
```

or

```
var Is_Admin = 2;
```

"0" is not an admin.

"1" can transition a CR to a state in any lifecycle.

"2" can transition a CR to any state in the current lifecycle.

Use 1 required parameters

None.

Use 1 optional parameters

None.

Use 2

CCM_CR_PROCESS_INFO=CURRENT_LIFECYCLE

Returns the name of the lifecycle that contains the state of the current change request.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
```

```
<!-- PARAM NAME=CCM_CR_PROCESS_INFO VALUE=CURRENT_LIFECYCLE -->
<!-- /WSLET -->
```

Output:

```
var Current_Lifecycle = "lifecycle_name";
```

Use 2 required parameters

None.

Use 2 optional parameters

None.

Use 3

CCM_CR_PROCESS_INFO=LIFECYCLE_STATES

Returns an array of all lifecycles and their associated states.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
<!-- PARAM NAME=CCM_CR_PROCESS_INFO VALUE=LIFECYCLE_STATES -->
<!-- /WSLET -->
```

Output:

```
var Lifecycle_States = new Array();

for each lifecycle
{
   Lifecycle_States[i]    = new Array();
   Lifecycle_States[i][0] = "lifecycle name"
   Lifecycle_States[i][1] = new Array();

   for each state
   {
      Lifecycle_States[i][1][j] = "state name"
   }
}
```

Use 3 required parameters

None.

Use 3 optional parameters

None.

Use 4

CCM_CR_PROCESS_INFO=VALID_CREATES

Returns an array of change request initial states and labels.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
<!-- PARAM NAME=CCM_CR_PROCESS_INFO VALUE=VALID_CREATES -->
<!-- /WSLET -->
```

Output:

```
var Submit_States = new Array();

for each state
{
   Submit_States[i]    = new Array();
   Submit_States[i][0] = "state name"
   Submit_States[i][1] = "state label"
}
```

Use 4 required parameters

None.

Use 4 optional parameters

None.

Use 5

CCM_TRANSITION_LINK

Returns a set of links to transition templates for the current change request.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
<!-- PARAM NAME=CCM_TRANSITION_LINK -->
<!-- /WSLET -->
```

Use 5 required parameters

None.

Use 5 optional parameters

TARGET_FRAME Specifies the target frame for this HTTP request (for example, `workspace`). VALUE="*`frame_name`*".

HTML_FONT Any valid HTML FONT specifications. VALUE="*`HTML font specification`*".

Rational Change provides the following:

```
<FONT "font info here"></FONT>
```

The default has no effect:

```
<FONT></FONT>
```

ATTR_DELIMITER Specifies a string that will separate each transition label. VALUE="*`delimiter between states`*".

TAG_ADMIN_ROLE Specifies the label that represents the link for the Admin transition. VALUE="*`admin link label`*".

This link is displayed only if the user has the necessary *Admin* role.

USE_TRANSITION_FUNCTION Instead of a direct HREF call, the CCM_TRANSITION_LINK calls a function instead. This is done to confirm that changes have not been made to the CR being viewed. The user has a choice to discard changes (or not) before navigating away from the CR. VALUE="*`transition function to call`*"

TRANSITION_TEMPLATE The actual template name is substituted for the "TRANSITION_TEMPLATE" identifier during parsing.

TRANSITION_TARGET_FRAME The actual target frame name is substituted for the TRANSITION_TARGET_FRAME identifier during parsing.

ALL_TRANSITIONS Special identifier, typically used by Rational Change API templates, to return all transitions, including hidden transitions. Still takes into account security evaluation. Currently if an API request is made to get available transitions, only transitions that are valid for the user and VISIBLE (displayed as a transition link in the Show dialog) are returned.

For application lifecycle management (ALM), there is a need to have transitions that can be executed from triggers that cannot be executed from the GUI; thus, they are marked as invisible.

An API call option has been be added to return a list of valid transitions including invisible ones.

Create Admin transition links, targeting the `workspace` frame.

Wslet call:

```
<!-- WSLET CODE=PTCRProcessInfo -->
<!-- PARAM NAME=CCM_TRANSITION_LINK -->
<!-- PARAM NAME=TARGET_FRAME VALUE=workspace -->
<!-- PARAM NAME=ATTR_DELIMITER VALUE=" | " -->
<!-- PARAM NAME=ADMIN_ROLE VALUE="Admin Transition" -->
<!-- /WSLET -->
```

Output:

```
<A HREF=/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=problem_show_form
&TEMPLATE_FLAG=entered2in_review&problem_number=57&role=PTUser&
token=11109607368>
<FONT>Verify</FONT>
</A>|
<A HREF=/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=problem_show_form
&TEMPLATE_FLAG=entered2probreject&problem_number=57&role=PTUser
&token=11109607368>
<FONT>Reject</FONT>
</A>|
<A HREF=/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=problem_show_form
&TEMPLATE_FLAG=entered2forwarded&problem_number=57&role=PTUser&
token=11109607368>
<FONT>Forward</FONT>
</A>|
<A HREF=/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=problem_show_form
&TEMPLATE_FLAG=entered2duplicate&problem_number=57&role=PTUser&
token=11109607368>
<FONT>Mark as Duplicate</FONT>
</A>|
<A HREF=/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=problem_show_form
&TEMPLATE_FLAG=
AdminTransition&problem_number=57role=PTUser&token=11109607368>
<FONT>Admin Transition</FONT>
</A>
```

Use 5 optional parameters

Wslet call:

```
    <!-- WSLET CODE=PTCRProcessInfo -->
    <!-- PARAM NAME=CCM_TRANSITION_LINK -->
    <!-- PARAM NAME=TARGET_FRAME VALUE=workspace -->
    <!-- PARAM NAME=ATTR_DELIMITER VALUE="|" -->
    <!-- PARAM NAME=ADMIN_ROLE VALUE="Admin Transition" -->
    <!-- PARAM NAME=USE_TRANSITION_FUNCTION VALUE=

"canTransition(TRANSITION_TEMPLATE,TRANSITION_TARGET_FRAME)" --
>
    <!-- /WSLET -->
```

Output:

```
<A
HREF="javascript:canTransition('entered2assigned','workspace')"
>Assign</A> |
<A
HREF="javascript:canTransition('entered2conceded','workspace')"
>Concede</A> |
<A
HREF="javascript:canTransition('entered2deferred','workspace')"
>Defer</A> |
<A
HREF="javascript:canTransition('entered2duplicate','workspace')
">Mark as Duplicate</A> |
<A
HREF="javascript:canTransition('entered2in_review','workspace')
">Verify</A> |
<A
HREF="javascript:canTransition('entered2need_more_info','worksp
ace')">Need More Info</A> |
<A
HREF="javascript:canTransition('entered2probreject','workspace'
)">Reject</A> |
<A
HREF="javascript:canTransition('entered2resolved','workspace')"
>SQE Resolve</A> |
<A
HREF="javascript:canTransition('AdminTransition','workspace')">
Admin Transition</A>
```

PTDataListBox

Generates a JavaScript data structure using a CCM_DATALISTBOX item defined in a configuration file.

Use 1

CCM_DATALISTBOX

Used for customizable data list box controls on templates. Returns the data list box as listbox options.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_DATALISTBOX VALUE="data list box name, as
in a config file entry" -->
<!-- /WSLET -->
```

The output generates HTML options.

Use 1 required parameters

None.

Use 1 optional parameters

CCM_USER_PROFILE   Only look in the user's profile configuration data for this data list box value.

CCM_SHARED_PROFILE   Only look in the shared configuration data for this data list box value.

Use 2

CCM_LISTBOX

Obtain the specified list box object as a JavaScript data structure.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_LISTBOX VALUE="listbox_name as in a
config file entry" -->
<!-- /WSLET -->
```

Output:

```
var "listbox_name" = new Array(); //a LISTBOX

for each listbox value in the referenced LISTBOX object
{
   listbox_name[i]    = new Array(); //a LISTBOX entry
   listbox_name[i][0] = "integer position value"; //VALUE
   listbox_name[i][0] = "listbox text" //TEXT
}
```

Use 2 required parameters

None.

Use 2 optional parameters

JAVASCRIPT_IDENTIFIER When creating the JavaScript data structure, use this name as the root variable name. VALUE="*preferred variable name*".

OBJECT_BASED Return a JavaScript data structure that is object based instead of array based.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_LISTBOX VALUE="listbox_name, as in config
file entry" -->
<!-- PARAM NAME=OBJECT_BASED -->
<!-- /WSLET -->
```

Output:

```
var "listbox_name" = new Array(); //A LISTBOX

for each listbox value in referenced LISTBOX object
       {
       listbox_name[i] = new Object();
       listbox_name[i].value = "integer position value";
       listbox_name[i].text = "listbox text";
       }
```

Use 3

CCM_ONLOAD_DATA

Loads a JavaScript data structure of the referenced LIST VALUES based on section type.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_ONLOAD_DATA VALUE="datalistbox_name, as
in a config file entry" -->
<!-- /WSLET -->
```

Output, based on section type:

CCM_QUERY section

```
var datalistbox_name = new Array();

for each query in the referenced LIST object
{
   datalistbox_name[i]    = new Array();  //QUERY DATA
   datalistbox_name[i][0] = "query string" //QRY_STRING
   datalistbox_name[i][1] = "referenced query template" //
QRY_TEMPLATE
   datalistbox_name[i][2] = "query object name" //NAME
   datalistbox_name[i][3] = "query description" //
DESCRIPTION
   datalistbox_name[i][4] = "query template form type" //
ONLOAD_ACTION
   datalistbox_name[i][5] = "last time user ran this
query" //DATE_LAST_RUN
}
```

CCM_REPORT section

```
var datalistbox_name = new Array(); //LIST OF REPORTS

for each report in the referenced LIST object
{
   datalistbox_name[i] = new Array(); //REPORT DATA
   datalistbox_name[i][0] = "report name" //NAME
   datalistbox_name[i][1] = new Array(); //REPORT TEMPLATE
   datalistbox_name[i][1][0] = "report template name" //
NAME
   datalistbox_name[i][1][1] = "report template form type"
//ONLOAD_ACTION
   datalistbox_name[i][2] = new Array(); //REPORT QUERY
   datalistbox_name[i][2][0] = "query object name" //NAME
   datalistbox_name[i][2][1] = "query string" //QRY_STRING
   datalistbox_name[i][2][2] = "referenced query template"
//QRY_TEMPLATE
   datalistbox_name[i][2][3] = "query template form type" /
/ONLOAD_ACTION
   datalistbox_name[i][2][4] = "query description" //
DESCRIPTION
   datalistbox_name[i][2][5] = "last time user ran this
query" //DATE_LAST_RUN
   datalistbox_name[i][3] = "file extension of the report,
default HTML"
   //EXPORT_FORM
   datalistbox_name[i][4] = "maximum allowed number of
items queried"
   //MAX_QUERY
   datalistbox_name[i][5] = "maximum allowed string length
for queried
   attributes" //MAX_STRING
   datalistbox_name[i][6] = "report description" //
DESCRIPTION
   datalistbox_name[i][7] = new Array(); //REPORT RELATIONAL
SUB-REPORTS

   for each subreport and top-level report in the report
definition
   {
      datalistbox_name[i][7][j] = new Array(); //REPORT SUB-
REPORT
      datalistbox_name[i][7][j][0] = "text" //RELATION
      datalistbox_name[i][7][j][1] = "text" //
[PROBLEM_DEF|TASK_DEF|OBJECT_DEF]
      datalistbox_name[i][7][j][2] = "integer specifying
the definition type
      [PROBLEM_DEF=17|TASK_DEF=18|OBJECT_DEF=19]"; //
DEF_TYPE
      datalistbox_name[i][7][j][3] = "definition name" //
NAME
      datalistbox_name[i][7][j][4] = "main HTML template" /
/MAIN_TEMPLATE
      datalistbox_name[i][7][j][5] = "header HTML template"
//HDR_TEMPLATE
      datalistbox_name[i][7][j][6] = "attribute HTML
template" //ATTR_TEMPLATE
      datalistbox_name[i][7][j][7] = "image HTML template"
//IMG_TEMPLATE
      datalistbox_name[i][7][j][8] = "grouping HTML
template" //GROUP_TEMPLATE
      datalistbox_name[i][7][j][9] = "single attribute HTML
template"
```

```
        //AUTO_ATTR_TEMPLATE
        datalistbox_name[i][7][j][10] = "footer HTML
template" //FTR_TEMPLATE
        datalistbox_name[i][7][j][11] = "attributes available
in this report
        definition" //ATTRS
        datalistbox_name[i][7][j][12] = "grouping attribute"
//GROUP_BY
        datalistbox_name[i][7][j][13] = "report sorting
order" //SORT_ORDER
        datalistbox_name[i][7][j][14] = "custom wslet
referenced in this report"
        //CUSTOM_WSLET
        datalistbox_name[i][7][j][15] = "XML needed for the
custom wslet"
        //XML_CONTENT
        datalistbox_name[i][7][j][16] = "single attribute
spanning HTML template"
        //SPAN_ATTR_TEMPLATE
        datalistbox_name[i][7][j][17] = "label HTML template"
//LABEL_TEMPLATE
        datalistbox_name[i][7][j][18] = "single label HTML
template"
        //AUTO_LABEL_TEMPLATE
        datalistbox_name[i][8] = "is report incremental"
        //INCREMENTAL
        datalistbox_name[i][9] = "increment size"
        //INCREMENT size
        datalistbox_name[i][10] = "report style"
        //STYLE
        datalistbox_name[i][11] = "display order"
        //CUSTOM DISPLAY ORDER
        datalistbox_name[i][12] = "image path"
        //IMAGE PATH
    }
}
```

CCM_LISTBOX section

```
var datalistbox_name = new Object(); //LIST OF LISTBOXES

for each list box in the referenced LIST object
{
    datalistbox_name["listbox_name"] = new Array(); //LISTBOX

    for each list box value in the referenced LISTBOX
object
    {
      datalistbox_name.listbox_name.[i] = new Array(); //a
LISTBOX entry
      datalistbox_name.listbox_name.[i][0] = "integer
position value"; //VALUE
      datalistbox_name.listbox_name.[i][1] = "listbox
text" //TEXT
   }
}
```

CCM_VALUELISTBOX section

```
var datalistbox_name = new Object(); //LIST OF VALUELISTBOXES

for each valuelistbox_name in the referenced LIST object
{
   datalistbox_name["valuelistbox_name"] = new Array(); //
VALUELISTBOX

   for each valuelistbox value in the referenced
VALUELISTBOX object
   {
      datalistbox_name.valuelistbox_name.[i] = new Array();
//VALUELISTBOX
      datalistbox_name.valuelistbox_name.[i][0] = "value
list box value";//VALUE
      datalistbox_name.valuelistbox_name.[i][1] = "value
list box label" //LABEL
   }
}
```

CCM_LIST section

```
var datalistbox_name = new Object(); //LIST OF LISTS

for each list in the referenced LIST object
{
   datalistbox_name["list_name"] = "list name" //LIST NAME
}
```

Use 3 required parameters

None.

Use 3 optional parameters

CCM_USER_PROFILE Look only in the user's profile configuration data for this data list box value. VALUE=`"`*`preferred variable name`*".

CCM_SHARED_PROFILE Only look in the shared configuration data for this data list box value.

JAVASCRIPT_IDENTIFIER When creating the JavaScript data structure, use this name as the root variable name.

OBJECT_BASED Return a JavaScript data structure that is object based instead of array based.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_ONLOAD_DATA VALUE="datalistbox_name, as in
config file entry" -->
<!-- PARAM NAME=OBJECT_BASED -->
<!-- PARAM NAME=JAVASCRIPT_IDENTIFIER VALUE="name of JavaScript
variable to use" -->
<!-- /WSLET -->
```

Output, based on section type:

CCM_QUERY section

```
var datalistbox_name = new Array(); //QUERY DATA
datalistbox_name.folders = "[array of folders]";

        for each query in the referenced LIST object
        {
        datalistbox_name[i] = new Object();
        datalistbox_name[i].queryString = "query string";
        datalistbox_name[i].template = "referenced query
template";
        datalistbox_name[i].name = "query object name";
        datalistbox_name[i].description = "query description";
        datalistbox_name[i].onload_action = "query template form
type";
        datalistbox_name[i].date_last_run = "last time user ran
this query";
        datalistbox_name[i].prompting_query = "XML that defines a
prompting query";
        datalistbox_name[i].folder = "folder this query is part
of";
        }
```

CCM_REPORT section

```
var datalistbox_name = new Array(); //REPORT DATA
datalistbox_name.folders = "[array of folders]";
for each report in the referenced LIST object
        {
```

```
       datalistbox_name[i] = new Object();
       datalistbox_name[i].name = "report name";
       datalistbox_name[i].template = "report template name";
       datalistbox_name[i].onload_action = "report template form
type";
       datalistbox_name[i].query = new Object();
       datalistbox_name[i].query.name = "query object name";
       datalistbox_name[i].query.queryString = "query string";
       datalistbox_name[i].query.template = "referenced query
template";
       datalistbox_name[i].query.onload_action = "query template
form type";
       datalistbox_name[i].query.description = "query
description";
       datalistbox_name[i].query.date_last_run = "last time user
ran this query";
       datalistbox_name[i].query.prompting_query = "XML that
defines a prompting query";
       datalistbox_name[i].export_form = "file extension of the
report, default HTML";
       datalistbox_name[i].max_query = "maximum allowed number
of items queried";
       datalistbox_name[i].max_string = "maximum allowed string
length";
       datalistbox_name[i].description = "report description";
       datalistbox_name[i].sub_reports = new Array(); //REPORT
RELATIONAL SUB REPORTS
       datalistbox_name[i].sub_report[j] = new Object();
       datalistbox_name[i].sub_report[j].relation = "text";
       datalistbox_name[i].sub_report[j].relation_def = "text";
       datalistbox_name[i].sub_report[j].def_type = "integer
specifiying the definition type";
       datalistbox_name[i].sub_report[j].name = "definition
name";
       datalistbox_name[i].sub_report[j].main_template = "main
HTML template";
       datalistbox_name[i].sub_report[j].header_template =
"header HTML template";
       datalistbox_name[i].sub_report[j].attr_template =
"attribute HTML template";
       datalistbox_name[i].sub_report[j].image_template = "image
HTML template";
       datalistbox_name[i].sub_report[j].group_template =
"grouping HTML template";
       datalistbox_name[i].sub_report[j].auto_attr_template =
"single attribute HTML template";
       datalistbox_name[i].sub_report[j].footer_template =
"footer HTML template";
```

```
        datalistbox_name[i].sub_report[j].attributes =
"attributes available in this report";
        datalistbox_name[i].sub_report[j].group_by = "grouping
attribute";
        datalistbox_name[i].sub_report[j].sort_order = "report
sorting order";
        datalistbox_name[i].sub_report[j].custom_wslet = "custom
wslet referenced in this report";
        datalistbox_name[i].sub_report[j].xml_content = "XML
needed for the custom wslet";
        datalistbox_name[i].sub_report[j].span_attr_template =
"single attribute spanning HTML template";
        datalistbox_name[i].sub_report[j].label_tempalte =
"label HTML template";
        datalistbox_name[i].sub_report[j].auto_label_template =
"single label HTML template";
        datalistbox_name[i].sub_report[j].incremental = "true/
false";
        datalistbox_name[i].sub_report[j].increment_size =
"integer of page size";
        datalistbox_name[i].sub_report[j].style = "report
style";
        datalistbox_name[i].sub_report[j].custom_display_order =
"custom_display_order";
        datalistbox_name[i].sub_report[j].image_path =
"image_path";
        datalistbox_name[i].folder = "XML that defines a report";
        }
```

CCM_LISTBOX section

```
var datalistbox_name = new Object() //LIST OF LISTBOXES
datalistbox_name.name = "name of the LIST";
datalistbox_name.elements = new Array();

for each list box in the referenced LIST object
        {
        datalistbox_name.elements[i] = new Object();
        datalistbox_name.elements[i].name = "name of the
listbox";
        datalistbox_name.elements[i].elements = new Array();

        for each value in the referenced LISTBOX object
        {
        datalistbox_name.elements[i].elements[j] = new Object();
        datalistbox_name.elements[i].elements[j].value =
"integer position value";
        datalistbox_name.elements[i].elements[j].text = "listbox
text";
```

```
        }
        }
```

CCM_VALUELISTBOX section

```
var datalistbox_name = new Object() //LIST OF LISTBOXES
datalistbox_name.name = "name of the VALUELISTBOX";
datalistbox_name.elements = new Array();

for each valuelistbox_name in the referenced list object
       {
       datalistbox_name.elements[i] = new Object();
       datalistbox_name.elements[i].name = "name of the
listbox";
       datalistbox_name.elements[i].elements = new Array();

       for each valuelistbox value in the referenced
VALUELISTBOX object
       {
       datalistbox_name.elements[i].elements[j] = new Object();
       datalistbox_name.elements[i].elements[j].value = "value
list box value";
       datalistbox_name.elements[i].elements[j].text = "value
list box label";
       }
       }
```

CCM_LIST section

```
var datalistbox_name = new Object() //LIST OF LISTS
datalistbox_name.name = "name of the LIST of LISTS";
datalistbox_name.elements = new Array();

for each list in the referenced LIST Object
       {
       datalistbox_name.elements[i] = new Object();
       datalistbox_name.elements[i].name = "list name";
       datalistbox_name.elements[i].value = "list value";
       }
```

Use 3 examples

Load the `resolvers` list box object.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_ONLOAD_DATA VALUE=USER_RESOLVERS -->
```

```
<!-- PARAM NAME=CCM_USER_PROFILE -->
<!-- PARAM NAME=JAVASCRIPT_IDENTIFIER VALUE=USER_RESOLVERS -->
<!-- /WSLET -->
```

Configuration file entries:

```
[CCM_LISTBOX]
   [NAME]resolvers[/NAME]
   [TYPE]STANDARD[/TYPE]
   [VALUES]Any|john|bob|sue|joe|tom|bill|jane
   [/VALUES]
[/CCM_LISTBOX]
[CCM_LIST]
   [NAME]USER_RESOLVERS[/NAME]
   [SECTION]CCM_LISTBOX[/SECTION]
   [VALUES]resolvers[/VALUES]
   [APPEND]true[/APPEND]
[/CCM_LIST]
[CCM_DATALISTBOX]
   [NAME]USER_RESOLVERS[/NAME]
   [LIST]USER_RESOLVERS[/LIST]
[/CCM_DATALISTBOX]
```

Output:

```
var USER_RESOLVERS = new Object(); //LIST OF LISTBOXES

USER_RESOLVERS["resolvers"] = new Array(); //LISTBOX
USER_RESOLVERS.resolvers[0] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[0][0] = 0; //VALUE
USER_RESOLVERS.resolvers[0][1] = "Any";

USER_RESOLVERS.resolvers[1] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[1][0] = 1; //VALUE
USER_RESOLVERS.resolvers[1][1] = "john";

USER_RESOLVERS.resolvers[2] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[2][0] = 2; //VALUE
USER_RESOLVERS.resolvers[2][1] = "bob";

USER_RESOLVERS.resolvers[3] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[3][0] = 3; //VALUE
USER_RESOLVERS.resolvers[3][1] = "sue";

USER_RESOLVERS.resolvers[4] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[4][0] = 4; //VALUE
USER_RESOLVERS.resolvers[4][1] = "joe";

USER_RESOLVERS.resolvers[5] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[5][0] = 5; //VALUE
USER_RESOLVERS.resolvers[5][1] = "tom";

USER_RESOLVERS.resolvers[6] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[6][0] = 6; //VALUE
USER_RESOLVERS.resolvers[6][1] = "bill";

USER_RESOLVERS.resolvers[7] = new Array(); //A LISTBOX ENTRY
USER_RESOLVERS.resolvers[7][0] = 7; //VALUE
USER_RESOLVERS.resolvers[7][1] = "jane";
```

Use 4

CCM_QUERY

Obtain the specified query object as a JavaScript data structure.

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_QUERY VALUE="query_name as in a config
file entry" -->
<!-- /WSLET -->
```

Output:

```
var "query_name" = new Array(); //QUERY DATA

query_name[0] = "query string" //QRY_STRING
query_name[1] = "referenced query template" //QRY_TEMPLATE
query_name[2] = "query object name" //NAME
query_name[3] = "query description" //DESCRIPTION
query_name[4] = "query template form type" //ONLOAD_ACTION
query_name[5] = "last time user ran this query" //
DATE_LAST_RUN
```

Use 4 required parameters

None.

Use 4 optional parameters

JAVASCRIPT_IDENTIFIER When creating the JavaScript data structure, use this name as the root variable name. VALUE="*preferred variable name*".

OBJECT_BASED Return a JavaScript data structure that is object based instead of array based.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_QUERY VALUE="query name, as in a config file
entry" -->
<!-- PARAM NAME=OBJECT_BASED -->
<!-- /WSLET -->
```

Output:

```
        var "query_name" = new Object(); //QUERY DATA
        query_name.queryString = "query string";
        query_name.template = "referenced query template";
        query_name.name = "query object name";
        query_name.description = "query description";
        query_name.onload_action = "query template form type";
        query_name.date_last_run = "last time user ran this
query";
        query_name.prompting_query = "XML that defines a
prompting query";
        query_name.folder = "folder this query is part of"; //
also see Tom
```

Use 5

CCM_REPORT

Obtain the specified report object as a JavaScript data structure.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_REPORT VALUE="report_name, as in a config
file entry" -->
!-- /WSLET -->
```

Output:

```
var "report_name" = new Array(); //REPORT DATA

report_name[0] = "report name" //NAME
report_name[1] = new Array(); //REPORT TEMPLATE
report_name[1][0] = "report template name" //NAME
report_name[1][1] = "report template form type" //
ONLOAD_ACTION
report_name[2] = new Array(); //REPORT QUERY
report_name[2][0] = "query object name" //NAME
report_name[2][1] = "query string" //QRY_STRING
report_name[2][2] = "referenced query template" //
QRY_TEMPLATE
```

```
report_name[2][3] = "query template form type" //
ONLOAD_ACTION
report_name[2][4] = "query description" //DESCRIPTION
report_name[2][5] = "date run" //DATE_LAST_RUN
report_name[3] = "file extension of the report, default
HTML" //EXPORT_FORM
report_name[4] = "maximum allowed number of items queried"
//MAX_QUERY
report_name[5] = "maximum allowed string length for
queried attributes"
//MAX_STRING
report_name[6] = "report description" //DESCRIPTION
report_name[7] = new Array(); //REPORT RELATIONAL SUB-REPORTS

for each subreport and top level report in the report
definition
{
   report_name[7][j]     = new Array(); //REPORT SUB-REPORT
   report_name[7][j][0] = "text" //RELATION
   report_name[7][j][1] = "text" //
[PROBLEM_DEF|TASK_DEF|OBJECT_DEF]
   report_name[7][j][2] = "integer specifying the
definition type
   [PROBLEM_DEF=17|TASK_DEF=18|OBJECT_DEF=19]"; //DEF_TYPE
   report_name[7][j][3] = "definition name" //NAME
   report_name[7][j][4] = "main HTML template" //
MAIN_TEMPLATE
   report_name[7][j][5] = "header HTML template" //
HDR_TEMPLATE
   report_name[7][j][6] = "attribute HTML template" //
ATTR_TEMPLATE
   report_name[7][j][7] = "image HTML template" //
IMG_TEMPLATE
   report_name[7][j][8] = "grouping HTML template" //
GROUP_TEMPLATE
   report_name[7][j][9] = "single attr HTML template" //
AUTO_ATTR_TEMPLATE
   report_name[7][j][10] = "footer HTML template" //
FTR_TEMPLATE
   report_name[7][j][11] = "attributes available in this
report definition"
   //ATTRS
   report_name[7][j][12] = "grouping attribute" //GROUP_BY
   report_name[7][j][13] = "report sorting order" //
SORT_ORDER
   report_name[7][j][14] = "custom wslet referenced in this
report"
   //CUSTOM_WSLET
   report_name[7][j][15] = "XML needed for the custom
wslet" //XML_CONTENT
   report_name[7][j][16] = "single attr spanning HTML
template"
   //SPAN_ATTR_TEMPLATE
   report_name[7][j][17] = "label HTML template" //
LABEL_TEMPLATE
   report_name[7][j][18] = "single label HTML template" //
AUTO_LABEL_TEMPLATE
}
```

Use 5 required parameters

None.

Use 5 optional parameters

JAVASCRIPT_IDENTIFIER When creating the JavaScript data structure, use this name as the root variable name. VALUE="*preferred variable name*"

OBJECT_BASED Return a JavaScript data structure that is object based instead of array based.

Wslet call:

```
.<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_REPORT VALUE="report name, as in a config
file entry" -->
<!-- PARAM NAME=OBJECT_BASED -->
<!-- /WSLET -->
```

Output:

```
       var "report_name" = new Object(); //REPORT_DATA
       report_name.name = "report name";
       report_name.template = "report template name";
       report_name.onload_action = "report template form type";
       report_name.query = new Object();
       report_name.query.name = "query object name";
       report_name.query.queryString = "query string";
       report_name.query.template = "referenced query template";
       report_name.query.onload_action = "query template form
type";
       report_name.query.description = "query description";
       report_name.query.date_last_run = "last time user ran
this query";
```

```
       report_name.query.prompting_query = "XML that defines a
prompting query";
       report_name.export_form = "file extension of the report,
default HTML";
       report_name.max_query = "maximum allowed number of items
queried";
       report_name.max_string = "maximum allowed string length";
       report_name.description = "report description";
       report_name.sub_reports = new Array(); //REPORT
RELATIONAL SUB REPORTS

       for each subreport and top level report in the report
definition
       {
report_name.sub_report[j] = new Object();
       report_name.sub_report[j].relation = "text";
       report_name.sub_report[j].relation_def = "text";
       report_name.sub_report[j].def_type = "integer specifiying
the definition type";
       report_name.sub_report[j].name = "definition name";
       report_name.sub_report[j].main_template = "main HTML
template";
       report_name.sub_report[j].header_template = "header HTML
template";
       report_name.sub_report[j].attr_template = "attribute HTML
template";
       report_name.sub_report[j].image_template = "image HTML
template";
       report_name.sub_report[j].group_template = "grouping HTML
template";
       report_name.sub_report[j].auto_attr_template = "single
attribute HTML template";
       report_name.sub_report[j].footer_template = "footer HTML
template";
       report_name.sub_report[j].attributes = "attributes
available in this report";
       report_name.sub_report[j].group_by = "grouping
attribute";
       report_name.sub_report[j].sort_order = "report sorting
order";
       report_name.sub_report[j].custom_wslet = "custom wslet
referenced in this report";
       report_name.sub_report[j].xml_content = "XML needed for
the custom wslet";
       report_name.sub_report[j].span_attr_template = "single
attribute spanning HTML template";
       report_name.sub_report[j].label_tempalte = "label HTML
template";
```

```
       report_name.sub_report[j].auto_label_template = "single
label HTML template";
       report_name.sub_report[j].incremental = "true/false";
       report_name.sub_report[j].increment_size = "integer of
page size";
       }

       report_name.folder = "XML that defines a report";
```

Use 6

CCM_VALUELISTBOX

Obtain the specified value list box object as a JavaScript data structure.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_VALUELISTBOX VALUE="value_listbox_name as
in a config file entry" -->
<!-- /WSLET -->
```

Output:

```
var "value_listbox_name" = new Array(); //a VALUELISTBOX

for each valuelistbox value in the referenced
VALUELISTBOX object
{
   value_listbox_name[i]    = new Array(); //a VALUELISTBOX
entry
   value_listbox_name[i][0] = "valuelistbox value"; //VALUE
   value_listbox_name[i][1] = "valuelistbox label" //LABEL
}
```

Use 6 required parameters

None.

Use 6 optional parameters

JAVASCRIPT_IDENTIFIER When creating the JavaScript data structure, use this name as the root variable name. VALUE="*preferred variable name*".

OBJECT_BASED Return a JavaScript data structure that is object based instead of array based.

Wslet call:

```
<!-- WSLET CODE=PTDataListBox -->
<!-- PARAM NAME=CCM_VALUELISTBOX VALUE="valuelistbox name, as in
a config file entry" -->
<!-- PARAM NAME=OBJECT_BASED -->
<!-- /WSLET -->
```

Output:

```
var valuelistbox_name = new Object();
valuelistbox_name.name = "name of the valuelistbox";
valuelistbox_name.elements = new Array();

       for each valuelistbox value in the referenced
VALUELISTBOX object
       {
       valuelistbox_name.elements[i] = new Object();
       valuelistbox_name.elements[i].value = "value list box
value";
       valuelistbox_name.elements[i].label = "value list box
label";
       }
```

PTDBListBox

PTDBListBox displays database object relationships, such as "`has_duplicate`," in a list box format.

Wslet call:

```
<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=relation_type VALUE="relation_name" -->
<!-- PARAM NAME=attribute_name1 -->
<!-- PARAM NAME=attribute_name2 -->
.
.
.
<!-- PARAM NAME=attribute_nameN -->
<!-- /WSLET -->
```

The possible *`relation_types`* are as follows:

CCM_PROBLEM_PROBLEM

problem-to-problem relationship

CCM_PROBLEM_TASK

problem-to-task relationship

CCM_PROBLEM_OBJECT

problem-to-object relationship

CCM_TASK_PROBLEM

task-to-problem relationship

CCM_TASK_TASK

task-to-task relationship

CCM_TASK_OBJECT

task-to-object relationship

CCM_OBJECT_PROBLEM

object-to-problem relationship

CCM_OBJECT_TASK

object-to-task relationship

CCM_OBJECT_OBJECT

object-to-object relationship

The *object_id* (Select/Option/Value) output for each item in the list box is one of the following:

- `problem_number` for a relation to a problem
- `task_number` for a relation to a task
- `cvid` for a relation to a object.

Attribute values are delimited strings of attributes to view. By default, the delimiter is a space.

Output:

```
<OPTION VALUE=object_id>attribute values</OPTION>
<OPTION VALUE=object_id>attribute values</OPTION>
.
.
.
```

Relationship data formats

PTDBListbox and PTDBReport wslets are similar, but PTDBListbox-style formats have buttons instead of links for additional information.

The report/link style (PTDBReport) is a little easier to use and does not require JavaScript functions; however, it might not give you the output format you want. The list box style (PTDBListbox) takes up less room on the template, but requires a button and some JavaScript code to implement.

Use

```
<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_PROBLEM_PROBLEM VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_PROBLEM_TASK VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_TASK_PROBLEM VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
```

```
<!-- PARAM NAME=CCM_TASK_TASK VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_TASK_OBJECT VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_OBJECT_PROBLEM VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_OBJECT_TASK VALUE="relation_name" -->
<!-- /WSLET -->

<!-- WSLET CODE=PTDBListBox -->
<!-- PARAM NAME=CCM_OBJECT_OBJECT VALUE="relation_name" -->
<!-- /WSLET -->
```

Required parameter

attribute_name Specifies the attribute for which to show the relationship.

You must provide at least one *attribute_name* and, optionally, the attribute order in the attribute array (VALUE="*attribute_order*", where the order is an integer >=1). You can specify up to 25 attributes this way.

Optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="*string*". The default is a space. Attributes are padded with one space before and after.

REPORT_DATE_FORMAT Overrides the default `pt.cfg` entry for the date, for CCM_DATE types. VALUE="*date_format*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

Examples

- Get the attachments associated with the current change request.

  Wslet call:

```
   <!-- WSLET CODE=PTDBListBox -->
   <!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE=attachment -->
   <!-- PARAM NAME=ATTR_DELIMITER VALUE=">>" -->
   <!-- PARAM NAME=attachment_name VALUE=1 -->
   <!-- PARAM NAME=comment VALUE=2 -->
   <!-- PARAM NAME=displayname VALUE=3 -->
   <!-- /WSLET -->
```

Output:

```
   <OPTION VALUE=10101>chart.html >> Chart of Q2 results >>
attachment_1-1
   </OPTION>
   <OPTION VALUE=10100>event.log >> Log contents for June 2007
>> attachment_1-1
   </OPTION>
```

- Get the duplicate change request for the current change request.

```
   <SELECT NAME="DUPLICATE" SIZE=1 ONCHANGE="">
      <!-- WSLET CODE=PTDBListBox -->
      <!-- PARAM NAME=CCM_PROBLEM_PROBLEM VALUE=has_duplicate -
->
      <!-- PARAM NAME=problem_number -->
      <!-- PARAM NAME=problem_synopsis -->
      <!-- /WSLET -->
   </SELECT>
```

Output:

```
   <SELECT NAME="DUPLICATE" SIZE=1 ONCHANGE="">
      <OPTION VALUE=42>42 Username not entered in Submitter
Information "Name"
      field </OPTION>
      <OPTION VALUE=19>19 Calendar not being updated
automatically </OPTION>
      <OPTION VALUE=15>15 Percentages exceeding 100 in tutorial
chart </OPTION>
   </SELECT>
```

PTDBReport

PTDBReport displays database object relationships, such as "has_duplicate," in a report format.

PTDBReport displays the same type of database information as PTDBListBox, but PTDBListBox uses buttons and PTDBReport uses hypertext links.

See "PTDBListBox" on page 281 for information about report types and formats.

Use 1

***relation_type*=ANY relation type**

Displays database object relationships, with no hyperlinks. The *relation_type* value must be a relationship name. See the relationships listed in "PTDBListBox" on page 281.

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=relation_type VALUE="relation_name" -->
<!-- PARAM NAME=attribute_name1 VALUE="attribute_order1" -->
<!-- PARAM NAME=attribute_name2 VALUE="attribute_order2" -->
.
.
.
<!-- PARAM NAME=attribute_nameN VALUE="attribute_orderN" -->
<!-- /WSLET -->
```

Use 1 required parameter

attribute_name Specifies the attribute for which to show the relationship.

You must provide at least one *attribute_name* and, optionally, the attribute order in the attribute array (VALUE="*attribute_order*", where the order is an integer >=1). You can specify up to 25 attributes this way.

Use 1 optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="*string*". The default is a space. Attributes are padded with one space before and after.

OBJECT_DELIMITER Separates each object. "<BR>" is the default if OBJECT_DELIMITER is not specified. VALUE="*delimiter* ". The default is "<BR>".

REPORT_DATE_FORMAT Overrides the default `pt.cfg` entry for the date format in a report, for CCM_DATE types. VALUE="*`date_format`*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

Use 1 examples

- Show several attributes of the change requests associated with a task.

  Wslet call:

```
   <!-- WSLET CODE=PTDBReport -->
   <!-- PARAM NAME=CCM_TASK_PROBLEM VALUE=has_associated_task -
->
   <!-- PARAM NAME=ATTR_DELIMITER VALUE="|" -->
   <!-- PARAM NAME=problem_synopsis VALUE=1 -->
   <!-- PARAM NAME=resolver VALUE=2 -->
   <!-- PARAM NAME=crstatus VALUE=3 -->
   <!-- PARAM NAME=severity VALUE=4 -->
   <!-- PARAM NAME=release VALUE=5 -->
   <!-- /WSLET -->
```

  Output:

```
   Username not entered in Submitter Information "Name" field |
jane |
   assigned | Showstopper | beta<BR>
   Calendar not being updated automatically | jane | assigned |
Showstopper |
   alpha<BR>
   Percentages exceeding 100 in tutorial chart | jane | assigned
| Minor |
   alpha<BR>
```

- Show the attachments associated with a CR.

  Wslet call:

```
   <!-- WSLET CODE=PTDBReport -->
   <!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE=attachment -->
   <!-- PARAM NAME=ATTR_DELIMITER VALUE=">>" -->
   <!-- PARAM NAME=attachment_name VALUE=1 -->
   <!-- PARAM NAME=comment VALUE=2 -->
   <!-- /WSLET -->
```

Output:

```
   chart.html >> Q2 2001 results <BR>event.log >> Log contents
for June 2007 <BR>
```

Use 2

***relation_type*=ANY relation type**

Displays database object relationships with image hyperlinks. When clicked, the hyperlink loads a form (for example, a show or transition form).

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=relation_type VALUE="relation_name" -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE="template_name" -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 2 required parameters

REPORT_ALLOW_MODIFY Causes a form to be loaded.

REPORT_MODIFY_POPUP Specifies the template file for the pop-up form. VALUE="*template_name*"

REPORT_LINK_IMAGE Specifies the image to use for the link. VALUE="*image_name*".

Use 2 optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="*string*". The default is a space. Attributes are padded with one space before and after.

OBJECT_DELIMITER Separates each object. "<BR>" is the default if OBJECT_DELIMITER is not specified. VALUE="*delimiter* ". The default is "<BR>".

REPORT_DATE_FORMAT Overrides the default `pt.cfg` entry for the date format in a report, for CCM_DATE types. VALUE="*date_format*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

REPORT_FORM_TYPE Specifies the form type of the template given by REPORT_MODIFY_POPUP. The default is "*frameset_form*". VALUE="*form_type*".

REPORT_IMAGE_PROMPT Specifies alternative text to use with an image. Most browsers display this text 1) when a user mouses over an image, or 2) if the browser is not capable of displaying images. This parameter is only relevant when you use REPORT_LINK_IMAGE. VALUE="*alternative_text*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*target_window_name*". By default, "NoFrame" is used.

WINDOW_OPTIONS Specifies window decorations for the pop-up window. If the window is already open, the window options are ignored. VALUE="*JavaScript_window_options*". By default, the window options are "status,resizable,width=700,height=600".

attribute_name Specifies the attribute for which to show the relationship.

You can provide one or more *attribute_names* and, optionally, the attributes orders in the attribute array (VALUE="*attribute_order*", where the order is an integer >=1). You can specify up to 25 attributes this way.

Use 3

relation_type=ANY relation type

Displays database object relationships with attribute value hyperlinks. When clicked, the hyperlink loads a form (for example, a show or transition form). The link is always the first of any attributes displayed.

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=relation_type VALUE="relation_name" -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE="template_name" -->
```

```
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE="link_attribute_name"
-->
<!-- /WSLET -->
```

Use 3 required parameters

REPORT_ALLOW_MODIFY Causes a form to be loaded.

REPORT_MODIFY_POPUP Specifies the template file for the pop-up form. VALUE="*template_name*"

REPORT_LINK_SHOW Specifies the attribute to use for the link.

Use 3 optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="*string*". The default is a space. Attributes are padded with one space before and after.

OBJECT_DELIMITER Separates each object. "`<BR>`" is the default if OBJECT_DELIMITER is not specified. VALUE="*delimiter* ". The default is "<BR>".

REPORT_DATE_FORMAT Overrides the default `pt.cfg` entry for the date format in a report, for CCM_DATE types. VALUE="*date_format*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

REPORT_FORM_TYPE Specifies the form type of the template given by REPORT_MODIFY_POPUP. The default is "*frameset_form*". VALUE="*form_type*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*target_window_name*". By default, "`NoFrame`" is used.

WINDOW_OPTIONS Specifies window decorations for the pop-up window. If the window is already open, the window options are ignored. VALUE="*JavaScript_window_options*". By default, the window options are "status,resizable,width=700,height=600".

attribute_name Specifies the attribute for which to show the relationship.

You can provide one or more *attribute_names* and, optionally, the attributes orders in the attribute array (VALUE="*attribute_order*", where the order is an integer >=1). You can specify up to 25 attributes this way.

Use 3 examples

- Show the tasks associated with a CR, providing a link on the task number to see the task in more detail.

  Wslet call:

```
   <!-- WSLET CODE=PTDBReport -->
   <!-- PARAM NAME=CCM_PROBLEM_TASK VALUE=associated_task -->
   <!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
   <!-- PARAM NAME=REPORT_LINK_SHOW VALUE=task_number -->
   <!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE=TaskDetailsView --
>
   <!-- PARAM NAME=ATTR_DELIMITER VALUE="|" -->
   <!-- PARAM NAME=task_synopsis VALUE=1 -->
   <!-- PARAM NAME=status VALUE=2 -->
   <!-- /WSLET -->
```

- Show the change requests associated with a task, where each CR ID is a link.

  Wslet call:

```
   <!-- WSLET CODE=PTDBReport -->
   <!-- PARAM NAME=CCM_TASK_PROBLEM VALUE=has_associated_task -
->
   <!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
   <!-- PARAM NAME=REPORT_LINK_SHOW VALUE=problem_number -->
   <!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE=ProblemReportView
-->
   <!-- PARAM NAME=REPORT_TARGET_FRAME VALUE=AssocProblem -->
   <!-- PARAM NAME=WINDOW_OPTIONS
VALUE="resizable,width=800,height=600" -->
   <!-- PARAM NAME=OBJECT_DELIMITER VALUE=", " -->
   <!-- /WSLET -->
```

Use 4

relation_type=CCM_PROBLEM_OBJECT

Display attachments with image hyperlinks for change request objects.

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE=attachment -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 4 required parameters

REPORT_ALLOW_MODIFY Causes a form to be loaded.

REPORT_VIEW_ATTACHMENT Causes the link to download the attachment.

REPORT_LINK_IMAGE Specifies the image to use for the link. `VALUE="`*`image_name`*`"`.

Use 4 optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="*`string`*". The default is a space. Attributes are padded with one space before and after.

OBJECT_DELIMITER Separates each object. "`<BR>`" is the default if OBJECT_DELIMITER is not specified. VALUE="*`delimiter`* ". The default is "`<BR>`".

REPORT_DATE_FORMAT Overrides the configuration file entry for the date format in a report, for CCM_DATE types. VALUE="*`date_format`*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

REPORT_IMAGE_PROMPT Specifies alternative text to use with an image. Most browsers display this text 1) when a user mouses over an image, or

2) if the browser is not capable of displaying images. This parameter is only relevant when you use REPORT_LINK_IMAGE. VALUE="`alternative_text`".

attribute_name Specifies the attribute for which to show the relationship.

You can provide one or more *attribute_names* and, optionally, the attributes orders in the attribute array (VALUE="`attribute_order`", where the order is an integer >=1). You can specify up to 25 attributes this way.

Use 5

***relation_type*=CCM_PROBLEM_OBJECT**

Display attachments with attribute value hyperlinks for change request objects. The link is always the first of any attributes displayed.

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE=attachment -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE="attribute_name" -->
<!-- /WSLET -->
```

Use 5 required parameters

REPORT_ALLOW_MODIFY Causes a form to be loaded.

REPORT_VIEW_ATTACHMENT Causes the link to download the attachment.

REPORT_LINK_SHOW Specifies the attribute to use for the link. You must provide at least one `link_attribute_name` and, optionally, the attribute order (VALUE="`attribute_order`", where the order is an integer >=1) in the attribute array. You can specify up to 25 attributes. VALUE="`link_attribute_name`".

Use 5 optional parameters

ATTR_DELIMITER Separates the attributes on a single object. VALUE="`string`". The default is a space. Attributes are padded with one space before and after.

OBJECT_DELIMITER Separates each object. "`<BR>`" is the default if OBJECT_DELIMITER is not specified. VALUE="*`delimiter`* ". The default is "`<BR>`".

REPORT_DATE_FORMAT Overrides the configuration file entry for the date format in a report, for CCM_DATE types. VALUE="*`date_format`*".

Note If you request a CCM_DATE-type attribute without specifying REPORT_DATE_FORMAT, the date is formatted using the SHOW_DATE_FORMAT system setting.

attribute_name Specifies the attribute for which to show the relationship.

You can provide one or more *attribute_names* and, optionally, the attributes orders in the attribute array (VALUE="*`attribute_order`*", where the order is an integer >=1). You can specify up to 25 attributes this way.

Use 5 example

Show the attachments associated with a CR as links, allowing them to be downloaded.

Wslet call:

```
<!-- WSLET CODE=PTDBReport -->
<!-- PARAM NAME=CCM_PROBLEM_OBJECT VALUE=attachment -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE=attachment_name -->
<!-- /WSLET -->
```

PTGenerate

PTGenerate is used in the `base_*.html` files to generate the following templates:

- report template
- email submit template
- query template
- show template
- submit template
- transition template

The template generated is determined using values in the installed *`crprocess_name`*`.xml` file.

PTGenerate wslets are the only wslets Rational Change evaluates during package creation; all other wslets are written as text to the generated template file.

You can change the behavior of all templates of a specified type by customizing a new version of the base template. Do so by choosing a base template from the General tab of either the CR Process Edit or the Lifecycles Edit buttons. A base template chosen from the Lifecycle Edit overrides one chosen from the CR Process, only for the one Lifecycle.

Note You are allowed to customize the generated templates after installing a CR Process package (a package that contains a *`crprocess_name`*`.xml` file). Do not, however, remove existing logic from the template

Use 1

GEN_ATTACHMENT_SUBMIT_FORM

Returns a set of HTML inputs of type HIDDEN for each of the required and optional attributes on the form. Used primarily with a visible attachment form.

Template scope: all base templates except query

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_ATTACHMENT_SUBMIT_FORM
VALUE="attachment_form_name" -->
<!-- /WSLET -->
```

Use 1 required parameter

You must include one—and only one—of the following `form_type_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 1 optional parameters

None.

Use 1 example

Return the HTML inputs for the `pt_attachment` attachment form.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_ATTACHMENT_SUBMIT_FORM
VALUE="pt_attachment" -->
<!-- /WSLET -->
```

Output:

```
<FORM NAME="pt_attachment">
   <INPUT NAME="ACTION_FLAG" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="TEMPLATE_FLAG" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="RELATION_NAME" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="token" TYPE=HIDDEN VALUE="NO_ACTION"></INPUT>
```

```
    <INPUT NAME="timestamp" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
    <INPUT NAME="modify_time" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

    <!-- PT Required Attribute Data -->
    <INPUT NAME="problem_synopsis" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
    .
    .
    .
    <!-- PT Optional Attribute Data -->
    <INPUT NAME="hardware" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
    .
    .
    .
</FORM>
```

Use 2

GEN_AUTO_INIT_FUNCTION

Generates the form initialization function for both copy and submit dialogs. All attributes that are user preference attributes set the HTML controls with their values. Non-user-preference attributes are stamped in the function, but are commented-out with double backslash characters.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_COPY_RELATION -->
<!-- /WSLET -->
```

Use 2 required parameters

None.

Use 2 optional parameters

You can include one—and only one—of the following *form_type*_ONLY parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

NO_ATTACHMENTS If this parameter is present, the "init_attachment_form();" function call is not placed inside the form initialization function.

Use 2 example

Return the relation name as a JavaScript variable.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_AUTO_INIT_FUNCTION -->
<!-- /WSLET -->
```

Output:

```
function init_form()
{
    .
    .
    .
}
```

Use 3

GEN_AUTO_USER_PROFILE

Autogenerates CCM_USER_PROFILE data used for submit, copy, and show templates. CR attributes that are flagged as preferences are automatically included. It replaces the GEN_USER_PROFILE wslet call, which is no longer needed.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_AUTO_USER_PROFILE -->
<!-- /WSLET -->
```

Use 3 required parameters

None.

Use 3 optional parameter

COPY _ONLY **S**pecifies the base template is to be used to produce a CR copy form.

SHOW _ONLY Specifies the base template is to be used to produce a CR show form.

SUBMIT Specifies the base template is to be used to produce a CR submit form.

Use 3 example

Generate the wslet for loading CCM_USER_PROFILE data.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_AUTO_USER_PROFILE -->
<!-- /WSLET -->
```

Output:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_USER_PROFILE -->
<!-- PARAM NAME="user_preference_attribute_name"
        .
        .
        .
<!-- /WSLET -->
```

Use 4

GEN_CCM_USER_PROFILE

Returns a PTLoadData Wslet formulated for retrieving CCM_USER_PROFILE data.

All attributes that are user preference attributes will used in the Wslet call. This is the same as GEN_USER_PROFILE.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_CCM_USER_PROFILE -->
<!-- /WSLET -->
```

Use 4 required parameters

None.

Use 4 optional parameters

None.

Use 4 example

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
    <!-- PARAM NAME=GEN_CCM_USER_PROFILE -->
    <!-- /WSLET -->
```

Result:

```
    <!-- WSLET CODE=PTLoadData -->
    <!-- PARAM NAME=CCM_USER_PROFILE -->
    <!-- PARAM NAME="user preference attribute" -->
    .
    .
    .
    <!-- /WSLET -->
```

Use 5

GEN_CCM_USER_UPDATE_FUNCTION

Creates a function on the form that lists all CCM_USER based attributes, and calls the "SUB_ATTR_setListboxValue()" function when invoked. The saving of a custom user list calls the form update function.

This feature is associated with new behavior for the "base.CCM_USER" control. The base CCM_USER control includes a link for setting the user's "custom user list". When the user uses this link to set his/her custom user list preference, all

CCM_USER controls on the current form will be updated to reflect the new list of users.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_CCM_USER_PROFILE -->
<!-- /WSLET -->
```

Use 5 required parameters

GEN_CCM_USER_UPDATE_FUNCTION The value of the Wslet parameter is the name specification of the function to create. VALUE = `users_update_OnClick(newlist)`.

Use 5 optional parameters

You can include one—and only one—of the following `*form_type*_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 5 example

Specify the form type for the template being generated.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_CCM_USER_UPDATE_FUNCTION
VALUE="users_update_OnClick(newlist)" -->
<!-- PARAM NAME=SUBMIT_ONLY -->
<!-- /WSLET -->
```

Output:

```
      function users_update_OnClick(newlist)
      {
         resolver_setListboxValue(newlist);
         .
         .
         .
      }
```

Use 6

GEN_COPY_RELATION

Returns the relation name as a JavaScript variable.

Template scope: copy base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_COPY_RELATION -->
<!-- /WSLET -->
```

Use 6 required parameters

None.

Use 6 optional parameters

None.

Use 6 example

Return the relation name as a JavaScript variable.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_COPY_RELATION -->
<!-- /WSLET -->
```

Output:

```
var Transition_Relation = "sub_problem";
```

Use 7

GEN_FILE_NAME

Returns the name of the file for the current template.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_FILE_NAME -->
<!-- /WSLET -->
```

Use 7 required parameters

None.

Use 7 optional parameters

None.

Use 8

GEN_FROM_STATE

Returns the "from state" as a JavaScript variable.

Template scope: transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_FROM_STATE -->
<!-- /WSLET -->
```

Use 8 required parameters

None.

Use 8 optional parameters

None.

Use 8 example

Return the "from state" as a JavaScript variable.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_FROM_STATE -->
<!-- /WSLET -->
```

Output:

```
var Transition_From_State = "entered";
```

Use 9

GEN_HDR_BGCOLOR

Uses a lifecycle background color in a generated template. The value used is the value specified in the CR Process editor in the edit pages of either the lifecycle or CR Process. If both are specified, the lifecycle value has a higher precedence that the CR Process value.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_HDR_BGCOLOR -->
<!-- /WSLET -->
```

Use 9 required parameters

None.

Use 9 optional parameter

None.

Use 9 example

Use the background color.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_HDR_BGCOLOR -->
<!-- /WSLET -->
```

Use 10

GEN_HDR_TXTCOLOR

Uses a lifecycle text color in a generated template. The value used is the value specified in the CR Process editor in the edit pages of either the lifecycle or CR Process. If both are specified, the lifecycle value has a higher precedence that the CR Process value.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_HDR_TXTCOLOR -->
<!-- /WSLET -->
```

Use 10 required parameters

None.

Use 10 optional parameter

None.

Use 10 example

Use the text color.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_HDR_TXTCOLOR -->
<!-- /WSLET -->
```

Use 11

GEN_INIT_FUNCTION

Creates the form initialization function, which initializes the values of each attribute. Used with the PTLoadData:CCM_USER_PROFILE wslet.

Template scope: email submit, submit, query, and transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_INIT_FUNCTION -->
<!-- /WSLET -->
```

Use 11 required parameters

None.

Use 11 optional parameter

cr_attribute_name Specifies an attribute to uncomment (include in initialization). You can specify more than one attribute. VALUE="*user_profile_attribute_name*".

Use 11 example

Uncomment the submitter_* and product_name attributes in the initialization function.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_INIT_FUNCTION -->
<!-- PARAM NAME=submitter VALUE=user -->
<!-- PARAM NAME=submitter_name VALUE=user_name -->
<!-- PARAM NAME=submitter_company VALUE=user_company -->
<!-- PARAM NAME=submitter_email VALUE=user_email -->
<!-- PARAM NAME=submitter_fax VALUE=user_fax -->
<!-- PARAM NAME=submitter_phone VALUE=user_phone -->
<!-- PARAM NAME=submitter_address VALUE=user_address -->
<!-- PARAM NAME=product_name VALUE=user_product_name -->
<!-- /WSLET -->
```

Output:

```
function init_form()
{
    submitter_address_Set_Value(user_address,true);
    submitter_company_Set_Value(user_company,true);
    submitter_fax_Set_Value(user_fax,true);
    submitter_email_Set_Value(user_email,true);
    submitter_phone_Set_Value(user_phone,true);
    submitter_name_Set_Value(user_name,true);
    submitter_Set_Value(user,true);
    product_name_Set_Value(user_product_name,true);
    .
    .
    .
    init_attachment_form();
}
```

Use 12

GEN_MODIFY_CHECK

Generates a JavaScript function that detects whether the user has made any changes to the form.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_MODIFY_CHECK VALUE="function_name" -->
<!-- /WSLET -->
```

The *function_name* value is optional.

Use 12 required parameters

None.

Use 12 optional parameter

You can include one—and only one—of the following *form_type*_ONLY parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 12 example

Generates the JavaScript function that detects changes to a form.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_MODIFY_CHECK VALUE="hasModifications()" -->
<!-- PARAM NAME=TRANSITION_ONLY -->
<!-- /WSLET -->
```

Output:

```
function hasModifications()
{
    if(IsFormLoading())
        return(false);

    if(priority_is_Modified_by_User())
        return(true);

    if(_COMMENTS_is_Modified_by_User())
        return(true);
    .
    .
    .
    return(false);
}
```

Use 13

GEN_MODIFY_FUNCTION

Generates the change request modification function.

Template scope: show and transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_MODIFY_FUNCTION VALUE="function_name" -->
<!-- /WSLET -->
```

`The` *`function_name`* value is optional.

Use 13 required parameters

None.

Use 13 optional parameters

You can include one—and only one—of the following *`form_type`*`_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

SUBMIT_FORM Specifies the name of the visible HTML form. VALUE="*form_name*".

SUBBUTTONBAR_FRAME Specifies the frame/window name to target with the wait indicator following a modification action. The WAITBUTTONBAR_TEMPLATE is the template that will be loaded. VALUE="*frame_name*"

WAITBUTTONBAR_TEMPLATE Specifies the *tokenless_form* to load following a modification action. VALUE="*template_name*"

TARGET_FRAME Specifies the frame/window name of the window that will be targeted as a result of the modification action. VALUE="*frame_name_to_target*".

TEMPLATE_FLAG Specifies the name of a custom response template that will be loaded as a result of the modification action. A default will be loaded if you do not set this parameter. VALUE="*template_name*".

SUBMIT_ACTION_FLAG Specifies the Rational Change modification "action." VALUE="*action_name*".

CS_SUBMIT_ACTION Specifies the URL/servlet HTTP address. VALUE="*url_name*". Use the SERVLET_PATH wslet, that is,

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=SERVLET_PATH VALUE=PTACTION_SERVLET -->
<!-- /WSLET -->
```

.

Use 13 example

Generate the change request modification function.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_MODIFY_FUNCTION VALUE="modify_OnClick()" --
>
<!-- PARAM NAME=SHOW_ONLY -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="cr_modify" -->
<!-- PARAM NAME=SUBBUTTONBAR_FRAME VALUE="parent.subbuttonbar"
-->
<!-- PARAM NAME=WAITBUTTONBAR_TEMPLATE VALUE="WaitButtonBar" --
>
<!-- PARAM NAME=TARGET_FRAME VALUE="workspace" -->
<!-- PARAM NAME=ACTION_FLAG VALUE="modify_problem" -->
<!-- PARAM NAME=TEMPLATE_FLAG VALUE="BlockStatus" -->
<!-- /WSLET -->
```

Output:

```
function modify_OnClick()
{
       if(IsFormLoading())
       return;

       var SubmitForm = false;

        // PT Optional Attribute Data
        if(problem_synopsis_is_Modified())
       {
       window.document.cr_modify.problem_synopsis.value =
       problem_synopsis_Value();
       SubmitForm = true;
       }
   .
   .
   .
     // Set the change request/problem number

   window.document.cr_modify.problem_number.value =
       problem_number;

   window.document.cr_modify.method = "POST";
   window.document.cr_modify.action =
       "/servlet/com.continuus.webpt.servlet.PTaction";
   window.document.cr_modify.target = "workspace";
   window.document.cr_modify.ACTION_FLAG.value =
"modify_problem";
```

```
    window.document.cr_modify.TEMPLATE_FLAG.value =
"BlockStatus";

    // PT Security Data
    window.document.cr_modify.role.value = role;
    window.document.cr_modify.token.value = token;
    window.document.cr_modify.timestamp.value = timestamp;
    window.document.cr_modify.modify_time.value = modify_time;

    parent.subbuttonbar.location=
        "/servletcom.continuus.webpt.servlet.PTweb?ACTION_FLAG=
        tokenless_form&TEMPLATE_FLAG=WaitButtonBar";
    window.document.cr_modify.submit();
}
```

Use 14

GEN_MODIFY_LABEL

Returns the label for the current show template.

Template scope: show base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_MODIFY_LABEL -->
<!-- /WSLET -->
```

Use 14 required parameters

None.

Use 14 optional parameters

SET_MODIFY_DATA Returns the label as a JavaScript variable called `Transition_Label`; for example:

```
var Transition_Label = "Submit and Assign a CR";
```

Use 15

GEN_ONLOAD_DATA

Returns a PTLoadData:CCM_ONLOAD_DATA wslet for retrieving change request data from the database. Optionally, you can force any number of CR attributes by placing them in the wslet call.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_ONLOAD_DATA -->
<!-- PARAM NAME=form_type_ONLY -->
<!-- /WSLET -->
```

Use 15 required parameters

None.

Use 15 required parameter

You must include one—and only one—of the following *form_type*_ONLY parameters; otherwise, IGNORE_SECURITY (making everything modifiable) is inserted in the generated wslet call:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 15 optional parameter

cr_attribute_name Specifies the name of an optional attribute that will be forced into the wslet call. Needs not be selected from the CR Process editor. You can specify more than one attribute.

Use 15 example

Generate the wslet for loading Change Request Show data, and force the `transition_log` attribute.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_ONLOAD_DATA -->
<!-- PARAM NAME=SHOW_ONLY -->
<!-- PARAM NAME=transition_log -->
<!-- PARAM NAME=_IS_MODIFIABLE -->
<!-- /WSLET -->
```

Output:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_ONLOAD_DATA VALUE="problem_attr_show_form" -
->
<!-- PARAM NAME=problem_synopsis -->
<!-- PARAM NAME=problem_description -->
<!-- PARAM NAME=request_type -->
.
.
.
<!-- PARAM NAME=transition_log -->
<!-- PARAM NAME=_IS_MODIFIABLE -->
<!-- /WSLET -->
```

Use 16

GEN_QUERY_FORM

Generates a list of attribute controls in HTML row format. The list of attributes is determined by which attributes are tagged as queriable in the lifecycle editor.

Template scope: query base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_QUERY_FORM VALUE="form_name" -->
<!-- /WSLET -->
```

Use 16 required parameters

None.

Use 16 optional parameter

CONTROL_WIDTH Specifies the width of all controls that accept a width parameter. VALUE="*integer_width*". The default is 25.

Use 16 example

Generate a query form.

Wslet call:

```
<TABLE WIDTH="100%" CELLSPACING=0 CELLPADDING=0 BORDER=0>
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_QUERY_FORM VALUE="query" -->
<!-- PARAM NAME=CONTROL_WIDTH VALUE="15" -->
<!-- /WSLET -->
</TABLE>
```

Output:

```
<TABLE WIDTH="100%" CELLSPACING=0 CELLPADDING=0 BORDER=0>
    <TR>
    <!-- WSLET CODE=PTInclude -->
    <!-- PARAM NAME="attr_controls/base.CCM_TOGGLE" -->
    <!-- PARAM NAME=SUB_ATTR VALUE="approved_by_architect" -->
    <!-- PARAM NAME=SUB_FORM VALUE="query" -->
    <!-- PARAM NAME=SUB_REQUIRED VALUE=false -->
    <!-- PARAM NAME=SUB_COLUMN_COUNT VALUE=1 -->
    <!-- PARAM NAME=SUB_TYPE_OF_FORM VALUE="QUERY" -->
    <!-- /WSLET -->
    </TR>
    <TR>
    <!-- WSLET CODE=PTInclude -->
    <!-- PARAM NAME="attr_controls/base.CCM_TOGGLE" -->
    <!-- PARAM NAME=SUB_ATTR VALUE="approved_by_dev_mgr" -->
    <!-- PARAM NAME=SUB_FORM VALUE="query" -->
    <!-- PARAM NAME=SUB_REQUIRED VALUE=false -->
    <!-- PARAM NAME=SUB_COLUMN_COUNT VALUE=1 -->
    <!-- PARAM NAME=SUB_TYPE_OF_FORM VALUE="QUERY" -->
    <!-- /WSLET -->
    </TR>
.
.
.
</TABLE>
```

Use 17

GEN_QUERY_FUNCTION

Generates a function that builds a query string for a custom query dialog. The string is an ACcent query string based on the set of attributes that have a value.

Template scope: query base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_QUERY_FUNCTION VALUE="function_name" -->
<!-- /WSLET -->
```

The *function_name* value is optional.

Use 17 required parameters

None.

Use 17 optional parameters

You can include one—and only one—of the following *form_type*_ONLY parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

GEN_QUERY_SUBMIT_FORM Specifies the name of the function to use when building the JavaScript. You can use the function in the `getQueryString()` function. VALUE="*function_name*".

CCM_LIST Specifies the name of the CCM_LIST to which the query object is appended. VALUE="*valid_CCM_LIST_name*".

CCM_DATALISTBOX Specifies the name of the CCM_DATALISTBOX that references the CCM_LIST object. VALUE="*valid_CCM_DATALISTBOX_name*".

Use 17 example

Generate the function that builds the query string.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_QUERY_FUNCTION VALUE="buildQueryString()" -
->
<!-- PARAM NAME=CCM_DATALISTBOX VALUE=CRQUERYCUSTOM -->
<!-- PARAM NAME=CCM_LIST VALUE=CRQueryCustom -->
<!-- PARAM NAME=QUERY_ONLY -->
<!-- /WSLET -->
```

Output:

```
function buildQueryString()
{
   var retval = "(cvtype='problem')";

   if(dev_effort_is_Modified())
   {
      if(dev_effort_has_Value())
      {
      retval += "and(dev_effort='" + dev_effort_Value() + "')";
      }
   }
   .
   .
   .
   return(retval);
}
```

Use 18

GEN_QUERY_NAME

Returns the name of the query object, derived from the CR Process name.

Template scope: query base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_QUERY_NAME -->
<!-- /WSLET -->
```

Use 18 required parameters

None.

Use 18 optional parameters

None.

Use 19

GEN_SECURITY_DATA

Returns a PTLoadData:CCM_SECURITY_DATA wslet call. The result contains the predetermined values for the security data.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SECURITY_DATA -->
<!-- /WSLET -->
```

Use 19 required parameters

None.

Use 19 optional parameters

None.

Use 19 example

Generate the wslet for loading the predetermined security data values.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SECURITY_DATA -->
<!-- /WSLET -->
```

Output:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_SECURITY_DATA -->
<!-- PARAM NAME=CCM_TOKEN -->
<!-- PARAM NAME=CCM_ROLE -->
<!-- PARAM NAME=CCM_TIMESTAMP -->
<!-- PARAM NAME=CCM_USER -->
<!-- PARAM NAME=CCM_VERSION VALUE="current_release_version" -->
<!-- /WSLET -->
```

Use 20

GEN_SUBMIT_FORM

Returns a set of HTML inputs of type HIDDEN for each of the required and optional attributes on the form. Used for submitting attribute data.

Template scope: submit base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SUBMIT_FORM VALUE="submit_form_name" -->
<!-- /WSLET -->
```

Use 20 required parameter

You must include one—and only one—of the following *form_type*_ONLY parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 20 optional parameters

None.

Use 20 example

Return the HTML inputs for the submission form.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SUBMIT_FORM VALUE="cr_submit" -->
<!-- PARAM NAME=SUBMIT_ONLY -->
<!-- /WSLET -->
```

Output:

```
<FORM NAME="cr_submit">
   <INPUT NAME="ACTION_FLAG"    TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="TEMPLATE_FLAG"  TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
```

```
   <INPUT NAME="RELATION_NAME"  TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="problem_number" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   <INPUT NAME="crstatus"       TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- PT Security Data -->
   <INPUT NAME="role"        TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="token"       TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="timestamp"   TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   <INPUT NAME="modify_time" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>

   <!-- PT Required Attribute Data -->
   <INPUT NAME="problem_synopsis" TYPE=HIDDEN
VALUE="NO_ACTION"></INPUT>
   .
   .
   .
   <!-- PT Optional Attribute Data -->
   <INPUT NAME="hardware" TYPE=HIDDEN VALUE="NO_ACTION"></
INPUT>
   .
   .
   .
</FORM>
```

Use 21

GEN_SUBMIT_FUNCTION

Generates the change request submit functions.

Template scope: submit base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SUBMIT_FUNCTION VALUE="function_name" -->
<!-- /WSLET -->
```

The *function_name* value is optional. You must include `crstatus` as a parameter.

Use 21 required parameters

None.

Use 21 optional parameters

You can include one—and only one—of the following `*form_type*_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

NO_ATTACHMENTS Suppresses generation of an attachment submit function.

SUBMIT_FORM Specifies the name of the visible HTML form. VALUE="*form_name*".

ATTACHMENT_FORM Specifies the name of the visible attachment form. VALUE="*form_name*"

SUBBUTTONBAR_FRAME Specifies the frame/window name to target with the wait indicator following a submit action. The WAITBUTTONBAR_TEMPLATE is the template that will be loaded. VALUE="*frame_name*".

WAITBUTTONBAR_TEMPLATE Specifies the `tokenless_form` to load following a submit action. VALUE="*template_name*".

TARGET_FRAME Specifies the frame/window name of the window that will be targeted as a result of the submit action. VALUE="*frame_name_to_target*".

TEMPLATE_FLAG Specifies the name of a custom response template that will be loaded as a result of the submit action. A default will be loaded if you do not set this parameter. VALUE="*template_name*".

SUBMIT_ACTION_FLAG Specifies the Rational Change submit "action." VALUE="*action_name*".

ATTACHMENT_ACTION_FLAG Specifies the Rational Change submit with attachments "action." VALUE="*action_name*"

CS_SUBMIT_ACTION Specifies the URL/servlet HTTP address. VALUE="*url_name*". Use the SERVLET_PATH wslet, that is,

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=SERVLET_PATH VALUE=PTACTION_SERVLET -->
<!-- /WSLET -->
```

.

Use 21 example

Generate the submit functions.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SUBMIT_FUNCTION
VALUE="submit_OnClick(crstatus)" -->
<!-- PARAM NAME=SUBBUTTONBAR_FRAME VALUE="parent.subbuttonbar" -
->
<!-- PARAM NAME=WAITBUTTONBAR_TEMPLATE VALUE="WaitButtonBar" -->
<!-- PARAM NAME=TARGET_FRAME VALUE="pt_status" -->
<!-- PARAM NAME=TEMPLATE_FLAG VALUE="SubmitStatus" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="cr_submit" -->
<!-- PARAM NAME=ATTACHMENT_FORM VALUE="pt_attachment" -->
<!-- PARAM NAME=SUBMIT_ACTION_FLAG VALUE="submit_problem" -->
<!-- PARAM NAME=ATTACHMENT_ACTION_FLAG
VALUE="submit_problem_and_attachment" -->
<!-- /WSLET -->
```

Output:

```
function submit_OnClick(crstatus)
.
.
.
function submit_form(crstatus)
.
.
.
function submit_form_with_attachment(crstatus) //if for
attachments
.
.
.
```

Use 22

GEN_SUBMIT_LABEL

Returns the label for the current submit template.

Template scope: submit base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_SUBMIT_LABEL -->
<!-- /WSLET -->
```

Use 22 required parameters

None.

Use 22 optional parameters

SET_MODIFY_DATA Returns the label as a JavaScript variable called `Transition_Label`; for example:

```
var Transition_Label = "Submit and Assign a CR";
```

Use 23

GEN_TEMPLATE_NAME

Returns information about the template being generated.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TEMPLATE_NAME -->
<!-- /WSLET -->
```

Output:

```
if this is a transition template or a state based show
template
{
   <!-- Lifecycle of Transition: lifecycle_name -->
   <!-- Transition Template : transition/state_template_name
-->
   <!-- Transition Description : transition/
state_description -->
   <!-- Transition Label : transition/state_label -->
}
<!-- Package Name  : package_name -->
<!-- Package Description :  package_description -->
<!-- Process Image  : process_image_name -->

else
<!-- Package Name  : package_name -->
<!-- Package Description :  package_description -->
<!-- Process Image  : process_image_name -->
```

Use 23 required parameters

None.

Use 23 optional parameters

None.

Use 23 example

Return information about the template currently being generated.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TEMPLATE_NAME -->
<!-- /WSLET -->
```

Output:

```
<!-- Lifecycle of Transition: Default Lifecycle -->
<!-- Transition Template    : entered2in_review -->
<!-- Transition Description : Transition the CR to the in_review
state when the CR has been verified. -->
<!-- Transition Label       : Verify -->
<!-- Package Name           : change_lifecycle_process -->
<!-- Package Description    : This is a sample development
Change Request Process. -->
<!-- Process Image          : dev_process.gif -->
```

Use 24

GEN_TO_STATE

Returns the "to state" as a JavaScript variable.

Template scope: transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TO_STATE -->
<!-- /WSLET -->
```

Use 24 required parameters

None.

Use 24 optional parameters

None.

Use 24 example

Return the "to state" as a JavaScript variable.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TO_STATE -->
<!-- /WSLET -->
```

Output:

```
var Transition_To_State = "in_review";
```

Use 25

GEN_TRANSITION_FUNCTION

Generates the change request transition function. You must specify a "to" state; for example, by using the following wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TO_STATE -->
<!-- /WSLET -->
```

Template scope: show and transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TRANSITION_FUNCTION VALUE="function_name"
-->
<!-- /WSLET -->
```

`The` *`function_name`* value is optional.

Use 25 required parameters

None.

Use 25 optional parameters

You can include one—and only one—of the following *`form_type`*`_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

SUBMIT_FORM Specifies the name of the visible HTML form. VALUE="*`form_name`*".

SUBBUTTONBAR_FRAME Specifies the frame/window name to target with the wait indicator following a transition action. The WAITBUTTONBAR_TEMPLATE is the template that will be loaded. VALUE="*frame_name*"

WAITBUTTONBAR_TEMPLATE Specifies the *tokenless_form* to load following a transition action. VALUE="*template_name*"

TARGET_FRAME Specifies the frame/window name of the window that will be targeted as a result of the transition action. VALUE="*frame_name_to_target*".

TEMPLATE_FLAG Specifies the name of a custom response template that will be loaded as a result of the transition action. A default will be loaded if you do not set this parameter. VALUE="*template_name*".

SUBMIT_ACTION_FLAG Specifies the Rational Change transition "action." VALUE="*action_name*".

CS_SUBMIT_ACTION Specifies the URL/servlet HTTP address. VALUE="*url_name*". Use the SERVLET_PATH wslet, that is,

```
<!-- WSLET CODE=PTAdmin -->
<!-- PARAM NAME=SERVLET_PATH VALUE=PTACTION_SERVLET -->
<!-- /WSLET -->
```

Use 25 example

Generate the change request transition function.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TRANSITION_FUNCTION
VALUE="transition_problem_OnClick()" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="cr_transition" -->
<!-- PARAM NAME=SUBBUTTONBAR_FRAME VALUE="parent.subbuttonbar"
-->
<!-- PARAM NAME=WAITBUTTONBAR_TEMPLATE VALUE="WaitButtonBar" --
>
<!-- PARAM NAME=TARGET_FRAME VALUE="workspace" -->
<!-- PARAM NAME=ACTION_FLAG VALUE="transition_problem" -->
<!-- PARAM NAME=TEMPLATE_FLAG VALUE="BlockStatus" -->
<!-- /WSLET -->
```

Use 26

GEN_TRANSITION_LABEL

Returns the label for the current transition template.

Template scope: transition base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TRANSITION_LABEL -->
<!-- /WSLET -->
```

Use 26 required parameters

None.

Use 26 optional parameters

SET_MODIFY_DATA Returns the label as a JavaScript variable called `Transition_Label`; for example:

```
var Transition_Label = "Submit and Assign a CR";
```

Use 27

GEN_USER_PREF_FORM

Generates the visible form elements used in the user submit dialog preferences. It calls the `base_user_preferences.html` user preference base template. This user preference base template is not configurable from the CR Process editor. If needed, customizations can be made to the base template.

Template scope: user preference base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TO_STATE -->
<!-- /WSLET -->
```

Use 27 required parameters

GEN_USER_PREF_FUNCTION he name of the visible HTML form. VALUE = "*`submit form name`*"

PREF_ONLY The parameter that states that the form is for user preferences (admin_form).

Use 27 optional parameters

CONTROL_WIDTH The width value for text and textarea html elements. VALUE ="*numeric value*"

The default is 25.

Use 27 example

Show the CR Process attributes that are flagged to be user preferences.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_USER_PREF_FORM VALUE="user_pref" -->
<!-- PARAM NAME=CONTROL_WIDTH VALUE="15" -->
<!-- PARAM NAME=PREF_ONLY -->
<!-- /WSLET -->
```

result:

A single column list of control wslets.

Use 28

GEN_USER_PREF_FUNCTION

Generates the function used to submit the user submit dialog preferences. It calls the `base_user_preferences.html` user preference base template. This user preference base template is not configurable from the CR Process editor. If needed, customizations can be made to the base template.

Template scope: user preference base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_TO_STATE -->
<!-- /WSLET -->
```

Use 28 required parameters

GEN_USER_PREF_FUNCTION The name of the function to be used. VALUE="*function name*"

Use 28 optional parameters

None.

Use 28 example

Use this function.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_USER_PREF_FUNCTION
VALUE="getPreferenceXML()" -->
<!-- /WSLET -->
```

result:

```
function getPreferenceXML()
{
   var retval = "";

   .
   .
   .

   return(retval);
```

Use 29

GEN_USER_PREF_INIT_FUNCTION

Autogenerates the form initialization function for user preferences dialogs. All attributes that are user preference attributes set the HTML controls with their values.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_USER_PREF_INIT_FUNCTION -->
<!-- /WSLET -->
```

Use 29 required parameters

None.

Use 29 optional parameter

None.

Use 29 example

Generate the wslet for loading CCM_USER_PROFILE data.

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_USER_PREF_INIT_FUNCTION -->
<!-- /WSLET -->
```

Result:

```
function init_form()
    {
        .
        .
        .
    }
```

Use 30

GEN_VISIBLE_FORM

Generates the visible HTML form.

The layout options for this wslet are as follows:

- Single Column
- Multiple Column
- By Order
- By Attribute Web Type.

Template scope: all base templates

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_VISIBLE_FORM -->
<!-- /WSLET -->
```

Use 30 required parameter

You must include one—and only one—of the following `*form_type*_ONLY` parameters:

SHOW_ONLY

or

TRANSITION_ONLY

or

SUBMIT_ONLY

or

QUERY_ONLY

Use 30 optional parameters

Single Column layout optional parameters:

None.

Multiple Column layout optional parameters:

COLUMN_COUNT Specifies the number of columns that the resultant HTML or dialog will have; that is, each row in the table will have this many columns. VALUE="*`integer_value`*" or "`auto`"

COLUMN_COUNT_PER_CONTROL Specifies the number of columns for a Rational Change control; that is, each control must conform to a known column count. VALUE="*`integer_value`*" or "`auto`". By default, the supplied controls conform to a two-column layout: *`label`*:*`value`*.

Note These values must be consistent with the number of columns in the control.

If **COLUMN_COUNT** and **COLUMN_COUNT_PER_CONTROL** parameters are missing, a single column layout (single control per row) will be generated. If "`auto`" is used for **COLUMN_COUNT**, the value in "Attributes per Row" listbox of the lifecycle editor will take effect. If "`auto`" is used for **COLUMN_COUNT_PER_CONTROL** or the parameter is missing, the default value "2" in the system will be used.

By Order layout optional parameters:

ORDER_BY Specifies the percentage of attributes output each time a GEN_VISIBLE_FORM wslet is called. The total of the percentages must be 1.

You can use the GEN_VISIBLE_FORM wslet repeatedly to output the entire attribute control set. VALUE="*`percentage_value`*".

By Attribute Web Type layout optional parameters:

CCM_LISTBOX VALUE="*percentage_value*".

CCM_VALUELISTBOX VALUE="*percentage_value*".

CCM_TEXT VALUE="*percentage_value*".

CCM_STRING VALUE="*percentage_value*".

CCM_BUTTON VALUE="*percentage_value*".

CCM_DATE VALUE="*percentage_value*".

CCM_READONLY VALUE="*percentage_value*".

CCM_HIDDEN VALUE="*percentage_value*".

CCM_TOGGLE VALUE="*percentage_value*".

CCM_RELATION VALUE="*percentage_value*".

CCM_NUMBER VALUE="*percentage_value*".

CCM_USER VALUE="*percentage_value*".

All the parameters listed above specify the percentage of attributes output each time a GEN_VISIBLE_FORM wslet is called. The total of the percentages must be 1.

You can use the GEN_VISIBLE_FORM wslet repeatedly to output the entire attribute control set. VALUE="*percentage_value*".

Use 30 example

Generate the visible HTML form.

Single Column

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_VISIBLE_FORM VALUE="modify" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="modify" -->
<!-- PARAM NAME=SHOW_ONLY -->
<!-- /WSLET -->
```

Multiple Column

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_VISIBLE_FORM VALUE="modify" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="modify" -->
<!-- PARAM NAME=COLUMN_COUNT VALUE=4 -->
<!-- PARAM NAME=COLUMN_COUNT_PER_CONTROL VALUE=2 -->
<!-- PARAM NAME=SHOW_ONLY -->
<!-- /WSLET -->
```

By Order

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_VISIBLE_FORM VALUE="modify" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="modify" -->
<!-- PARAM NAME=SHOW_ONLY -->
<!-- PARAM NAME=ORDER_BY VALUE=".50" -->
<!-- /WSLET -->
```

By Attribute Web Type

Wslet call:

```
<!-- WSLET CODE=PTGenerate -->
<!-- PARAM NAME=GEN_VISIBLE_FORM VALUE="modify" -->
<!-- PARAM NAME=SUBMIT_FORM VALUE="modify" -->
<!-- PARAM NAME=SHOW_ONLY -->
<!-- PARAM NAME=CCM_READONLY VALUE=".50" -->
<!-- PARAM NAME=CCM_HIDDEN VALUE=".50" -->
<!-- /WSLET -->
```

PTInclude

PTInclude includes the contents of one or more files, and optionally performs substitutions.

PTInclude is useful for reusing common JavaScript functions and putting controls on forms. Rational Change pre-processes included files to perform any and all substitutions, after which Rational Change evaluates any included wslets on the included files.

A *control*, or *control-based attribute*, is a file containing the HTML and JavaScript for displaying and manipulating an attribute. Controls are based on attribute Web Types (see "[TYPE]" on page 67), and the standard control for each Web Type is contained in a file named `base.`*`WEB_TYPE`*. Controls reside in the `.../include/attr_controls` directory and are placed on templates using the PTInclude wslet.

You can override the base control for an individual attribute by implementing a new control in a file named *`attribute_name.WEB_TYPE`*.

Use

filename

Includes the named file. The file name is relative to the *`wsconfig_dir`*`/templates/pt/include` directory. You can include multiple files with a single call to PTInclude.

Note The order in which the files are included is indeterminate.

```
<!-- WSLET CODE=PTInclude -->
<!-- PARAM NAME=filename1 -->
<!-- PARAM NAME=filename2 -->
.
.
.
<!-- PARAM NAME=filenameN -->
<!-- /WSLET -->
```

Required parameters

None.

Optional parameter

SUB_* Substitutes the named string with the given substitution value. The substituted string must begin with the prefix "`SUB_`". Multiple substitutions are allowed. VALUE=*`sub_value`*.

Examples

- Put the contents of the `base.CCM_READONLY` control (which is located in the `attr_controls` subdirectory) at the location of the wslet tag.

  Wslet call:

  ```
  <!-- WSLET CODE=PTInclude -->
  <!-- PARAM NAME="attr_controls/base.CCM_READONLY" -->
  <!-- PARAM NAME=SUB_ATTR VALUE="problem_number" -->
  <!-- PARAM NAME=SUB_COLUMN_COUNT VALUE=1 -->
  <!-- PARAM NAME=SUB_TYPE_OF_FORM VALUE="SHOW" -->
  <!-- /WSLET -->
  ```

  Output:

  Rational Change makes the following substitutions in `base.CCM_READONLY` (the control file):

  ```
  SUB_ATTR         --> problem_number
  SUB_COLUMN_COUNT --> 1
  SUB_TYPE_OF_FORM --> SHOW
  ```

  Rational Change includes `base.CCM_READONLY` in the file that called the wslet.

- Put the contents of all four JavaScript files (`formload.js`, `messagebox.js`, `ptalert.js`, and `calendar.js`) at the location of the wslet tags.

  Wslet call:

  ```
  <!-- WSLET CODE=PTInclude -->
  <!-- PARAM NAME=formload.js -->
  <!-- PARAM NAME=messagebox.js -->
  <!-- PARAM NAME=ptalert.js -->
  <!-- PARAM NAME=calendar.js -->
  <!-- /WSLET -->
  ```

  Output:

  The specified files are included in the file.

PTListBox

PTListBox builds select options in customizable list boxes on templates. The select options are based on the values in CCM_LISTBOX data items in configuration files.

PTListBox presets HTML control elements, when templates are parsed, using context data loaded by PTLoadData.

Note PTListBox generates the option elements, not the select element in which a list box is used.

Rational Change pads list boxes to compensate for the Netscape behavior if the list box is reloaded. If the original list box size was small, like 1, and then is reloaded to be of size 10, Netscape 4.7 will have a scrollable drop-down of size 1. Adding padded entries makes the drop-down size more reasonable.

Use 1

CCM_LISTBOX

Get an HTML option element (using a CCM_LISTBOX entry). If context data exists for the list box attribute, the corresponding option is selected.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=CCM_LISTBOX VALUE=listbox_name -->
<!-- /WSLET -->
```

Output:

```
<OPTION VALUE=0>Any</OPTION>
<OPTION VALUE=1>listbox_value1</OPTION>
<OPTION VALUE=2>listbox_value2</OPTION>
.
.
.
<OPTION VALUE=N>listbox_valueN</OPTION>
<OPTION VALUE=-1></OPTION>
```

Use 1 required parameters

None.

Use 1 optional parameter

ALTERNATE_LISTBOX   Allows a different list box to be used than the one defined for the list box attribute. VALUE=*listbox_name*.

Normally, the attribute name and list box name must match. This option allows you to use a list box with a different name than the attribute.

CCM_DATALISTBOX   Allows a data list box to be used instead of the list box attribute. The data list box must reference a list that is of type `CCM_LISTBOX`, and the list must have only one listbox. VALUE=*datalistbox_name*.

APPEND_LISTBOX   Appends the contents of a listbox to the end of the result. Duplicate entries are omitted. VALUE=`valid ccm_`*listbox*

Wslet call example:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=CCM_LISTBOX VALUE=listbox_name -->
<!-- PARAM NAME=ALTERNATE_LISTBOX VALUE=transition_user -->
<!-- PARAM NAME=CCM_DATALISTBOX VALUE=user_resolvers -->
<!-- PARAM NAME=APPEND_LISTBOX VALUE=external_resolvers -->
<!-- /WSLET -->
```

Output:

The contents of the values [CCM_LISTBOX][NAME]EXTERNAL_RESOLVERS[/NAME][VALUES]*your_values*[/VALUES][/CCM_LISTBOX] are merged at the end.

Use 2

DEP_FUNCTION

Generate the JavaScript to load the correct values into a dependent sublist box when the parent list box value changes.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=DEP_FUNCTION VALUE=listbox_name -->
<!-- PARAM NAME=DEP_FORM VALUE=form_name -->
<!-- /WSLET -->
```

Output:

Calls the following functions for each dependent attribute:

```
dep_attr_LoadListBox
    (window.document.form_name.listbox_name,
    DEP_LISTS_dep_attr[listbox_name_ListBoxPosition
        (window.document.submit.listbox_name.options

[window.document.submit.listbox_name.selectedIndex].text)]);
dep_attr_ChangeFocus();
```

Use 2 required parameter

DEP_FORM Specifies the form name for the dependent function. VALUE="*form_name*".

Use 2 optional parameters

None.

Use 3

DEP_LISTS

Get all the sublist boxes values for a dependent (child) attribute as a JavaScript data structure. That is, get a list of sublist boxes, complete with their values.

DEP_LISTS creates a JavaScript data structure that contains all the [SUBLISTBOX][/SUBLISTBOX] sublist box values.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=DEP_LISTS VALUE="dep_attr" -->
<!-- /WSLET -->
```

The data structure is an array of sublist boxes. Each sublist box is an array of options. Each option is a two-element array, where the first element is the option value, and second element is the option text.

Access the data structure as follows:

```
sublistbox option value =
DEP_LISTS_listbox_name[sublistbox_index][option_index][0]
sublistbox option text =
DEP_LISTS_listbox_name[sublistbox_index][option_index][1]
```

Therefore, `DEP_LISTS_`*`listbox_name`*`[1][2][1]` accesses the text of the 3rd option of the 2nd sublist box.

Output:

The Dependent Function wslet creates JavaScript code that calls the dependent attribute's Load Listbox function.

Use 3 optional parameters

None.

Use 4

FROM_CENTRAL

In central server mode, database-specific list boxes normally come from the context database. Instead, if the listbox should always come from the central CR database, this flag can be used. For example, the modifiable_in attribute is closely tied to the database the CRs reside in and thus always passes the FROM_CENTRAL flag. This is ignored in standalone mode.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=CCM_LISTBOX VALUE=SUB_ATTR -->
<!-- PARAM NAME=FROM_CENTRAL -->
<!-- /WSLET -->
```

Output:

Typical PTListBox output, but pulls database-specific values from the central database.

Required parameter

None.

Optional parameters

FROM_CENTRAL

Use 5

RELOAD

Get the list box values of a CCM_LISTBOX entry as a JavaScript data structure. The data structure is an array of options. Each option is a two-element array, where the first element is the option value, and the second element is the option text.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=RELOAD VALUE=listbox_name -->
<!-- /WSLET -->
```

Output:

```
var RELOAD_listbox_name = new Array(); //LISTBOX
RELOAD_listbox_name[0] = new Array(); //LISTBOX entry
RELOAD_listbox_name[0][0] = 0; //VALUE
RELOAD_listbox_name[0][1] = "text_string0" //TEXT
RELOAD_listbox_name[1] = new Array(); //LISTBOX entry
RELOAD_listbox_name[1][0] = 1; //VALUE
RELOAD_listbox_name[1][1] = "text_string1" //TEXT
.
.
.
RELOAD_listbox_name[N] = new Array(); //LISTBOX entry
RELOAD_listbox_name[N][0] = N; //VALUE
RELOAD_listbox_name[N][1] = "text_stringN" //TEXT
```

Use 5 required parameters

None.

Use 5 optional parameters

None.

Example

Load a browser version list box that depends on the browser value.

1. Get the `browser` HTML option element.

   Wslet call:

   ```
   <!-- WSLET CODE=PTListBox -->
   <!-- PARAM NAME=CCM_LISTBOX VALUE=browser -->
   <!-- /WSLET -->
   ```

   Configuration file entries:

```
[CCM_LISTBOX]
   [NAME]browser[/NAME]
   [TYPE]ATTRIBUTE[/TYPE]
   [VALUES]Any|Internet Explorer|Netscape[/VALUES]
   [DEPENDENT_ATTR]browser_version[/DEPENDENT_ATTR]
[/CCM_LISTBOX]

[CCM_LISTBOX]
   [NAME]browser_version[/NAME]
   [TYPE]ATTRIBUTE[/TYPE]
   [VALUES]Any|4.0|5.0|5.5|4.7|6.0|4.5[/VALUES]
   [SUBLISTBOX]BROWSER_VERSION_ANY|BROWSER_VERSION_INTERNET
   EXPLORER|BROWSER_VERSION_NETSCAPE[/SUBLISTBOX]
[/CCM_LISTBOX]

[CCM_LISTBOX]
   [NAME]BROWSER_VERSION_ANY[/NAME]
   [TYPE]SUBLISTBOX[/TYPE]
   [VALUES]Any[/VALUES]
[/CCM_LISTBOX]

[CCM_LISTBOX]
   [NAME]BROWSER_VERSION_INTERNET EXPLORER[/NAME]
   [TYPE]SUBLISTBOX[/TYPE]
   [VALUES]Any|4.0|5.0|5.5[/VALUES]
[/CCM_LISTBOX]

[CCM_LISTBOX]
   [NAME]BROWSER_VERSION_NETSCAPE[/NAME]
   [TYPE]SUBLISTBOX[/TYPE]
   [VALUES]Any|4.5|4.7|6.0[/VALUES]
[/CCM_LISTBOX]
```

Output:

```
<OPTION VALUE=0>Any</OPTION>
<OPTION VALUE=1>Internet Explorer</OPTION>
<OPTION VALUE=2>Netscape</OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
<OPTION VALUE=-1></OPTION>
```

2. Get the JavaScript data structure to reload the browser values.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=RELOAD VALUE=browser -->
<!-- /WSLET -->
```

Output:

```
var RELOAD_browser = new Array(); //LISTBOX
RELOAD_browser[0] = new Array(); //LISTBOX entry
RELOAD_browser[0][0] = 0; //VALUE
RELOAD_browser[0][1] = "Any" //TEXT
RELOAD_browser[1] = new Array(); //LISTBOX entry
RELOAD_browser[1][0] = 1; //VALUE
RELOAD_browser[1][1] = "Internet Explorer" //TEXT
RELOAD_browser[2] = new Array(); //LISTBOX entry
RELOAD_browser[2][0] = 2; //VALUE
RELOAD_browser[2][1] = "Netscape" //TEXT
```

3. Get the browser version sublist box values (dependent list).

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=DEP_LISTS VALUE=browser_version -->
<!-- /WSLET -->
```

Output:

```
   var DEP_LISTS_browser_version = new Object(); //LIST OF
LISTBOXES
   DEP_LISTS_browser_version[0] = new Array(); //LISTBOX
   DEP_LISTS_browser_version[0][0] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[0][0][0] = 0; //VALUE
   DEP_LISTS_browser_version[0][0][1] = "text" //TEXT
   DEP_LISTS_browser_version[1] = new Array(); //LISTBOX
   DEP_LISTS_browser_version[1][0] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[1][0][0] = 0; //VALUE
   DEP_LISTS_browser_version[1][0][1] = "text" //TEXT
   DEP_LISTS_browser_version[1][1] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[1][1][0] = 1; //VALUE
   DEP_LISTS_browser_version[1][1][1] = "text" //TEXT
   DEP_LISTS_browser_version[1][2] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[1][2][0] = 2; //VALUE
   DEP_LISTS_browser_version[1][2][1] = "text" //TEXT
   DEP_LISTS_browser_version[1][3] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[1][3][0] = 3; //VALUE
   DEP_LISTS_browser_version[1][3][1] = "text" //TEXT
   DEP_LISTS_browser_version[2] = new Array(); //LISTBOX
   DEP_LISTS_browser_version[2][0] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[2][0][0] = 0; //VALUE
   DEP_LISTS_browser_version[2][0][1] = "text" //TEXT
   DEP_LISTS_browser_version[2][1] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[2][1][0] = 1; //VALUE
   DEP_LISTS_browser_version[2][1][1] = "text" //TEXT
   DEP_LISTS_browser_version[2][2] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[2][2][0] = 2; //VALUE
   DEP_LISTS_browser_version[2][2][1] = "text" //TEXT
   DEP_LISTS_browser_version[2][3] = new Array(); //LISTBOX
entry
   DEP_LISTS_browser_version[2][3][0] = 3; //VALUE
   DEP_LISTS_browser_version[2][3][1] = "text" //TEXT
```

4. Load the browser version list based on the parent value.

   Wslet call:

```
<!-- WSLET CODE=PTListBox -->
<!-- PARAM NAME=DEP_FUNCTION VALUE=browser -->
<!-- PARAM NAME=DEP_FORM VALUE=submit -->
<!-- /WSLET -->
```

   Output:

```
browser_version_LoadListBox
    (window.document.submit.browser_version,
    DEP_LISTS_browser_version[browser_ListBoxPosition
    (window.document.submit.browser.options
    [window.document.submit.browser.selectedIndex].text)]);
browser_version_ChangeFocus();
```

Use 6

IS_EXTERNAL

Retrieves the external status of a sublist. Displays `true` if the listbox specified is external, `false` if it is not.

Wslet call:

```
<!-- WSLET CODE=PTListBox -->
```

```
<!-- PARAM NAME=IS_EXTERNAL VALUE="browser" -->
<!-- /WSLET -->
```

Output:

```
true
```

Use 6 required parameter

attribute_value Specifies the name of the listbox to be checked.

Use 6 optional parameters

None.

Use 7

GET_ANCESTORS_ARRAY

Retrieves a JavaScript array of all of the attributes above the current attribute in the listbox hierarchy.

Wslet call:

```
<!-- WSLET CODE=PTListbox -->
<!-- PARAM NAME=GET_ANCESTORS_ARRAY VALUE=os_version -->
<!-- /WSLET -->
```

The value returned is an array of name/value pairs of all the attributes above the request attribute in the listbox hierarchy. The name is the actual attribute name (not the alias) and the value is determined by the value of the parent attribute on the form. If the parent value function does not exist, the parent value is null; otherwise it is the value of the parent.

Output:

```
var ancestors = new Array();
ancestors[0] = new Object()
ancestors[0].name = “hardware”;
ancestors[0].value = window.hardware_Value ? hardware_Value() :
null;
ancestors[1] = new Object()
ancestors[1].name = “os”;
ancestors[1].value = window.os_Value ? os_Value() : null;
```

Use 7 required parameter

attribute_value	Specifies the name of the listbox to be checked.

Use 7 optional parameters

None.

PTLoadData

PTLoadData retrieves database data about change requests, tasks, and objects. PTLoadData also makes context data available to other wslets.

PTLoadData retrieves the following:

- security information
- user profile settings
- system information
- the name of the template file being processed

Use 1

CCM_LOGIN_ID

Determines the login ID of the user.

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_LOGIN_ID -->
<!-- /WSLET -->
```

Output:

```
var login_id ='user1';
```

Use 2

CCM_ONLOAD_DATA

Make context data available to other wslets. (*Context data* is data available to the parser.)

Only the attributes specified for `CCM_ONLOAD_DATA` are available as context data to other wslets. By default, the only output of the `CCM_ONLOAD_DATA` action is JavaScript variables for `cvid` and `modify_time` (these are forced in, whether or not they were requested).

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_ONLOAD_DATA VALUE="template_type" -->
<!-- PARAM NAME=attribute_name1 -->
<!-- PARAM NAME=attribute_name2 -->
.
.
.
<!-- PARAM NAME=attribute_nameN -->
<!-- /WSLET -->
```

The template type must be one of the following, and should match the type of template you are using:

- problem_attr_show_form
- problem_show_form
- task_attr_show_form
- task_show_form
- object_attr_show_form
- object_show_form

Use 2 required parameters

None.

Use 2 optional parameters

ALL_STRINGS Output each of the specified attributes as a JavaScript variable. Without this parameter, the requested attributes are simply put into context data.

Use 2 examples

- Loads the contact data.

  Wslet call:

  ```
     <!-- WSLET CODE=PTLoadData -->
     <!-- PARAM NAME=CCM_ONLOAD_DATA
  VALUE="problem_attr_show_form" -->
     <!-- PARAM NAME=product_name -->
     <!-- PARAM NAME=hardware -->
     <!-- /WSLET -->
  ```

- Retrieve the `product_name` and `hardware` attribute values, and make their values JavaScript variables on the form.

  Wslet call:

  ```
     <!-- WSLET CODE=PTLoadData -->
     <!-- PARAM NAME=CCM_ONLOAD_DATA
  VALUE="problem_attr_show_form" -->
     <!-- PARAM NAME=ALL_STRINGS -->
     <!-- PARAM NAME=product_name -->
     <!-- PARAM NAME=hardware -->
  ```

```
<!-- /WSLET -->
```

Output:

```
var cvid = "cvid_value"
var problem_number = "problem_number"
var modify_time = "time_value"
var hardware = "hardware_value"
var product_name = "product_name"
```

Use 3

CCM_PT_SYSTEM

Obtain the requested Rational Change system information.

The following are examples of system attributes:

- `database`

  Specifies the database to which you are connected.
- `dcm_dbid`

  Specifies the DCM database ID.
- `dcm_delimiter`

  Specifies the DCM database delimiter.
- `pt_app`

  Internal use only.
- `pt_app_role`

  Internal use only.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_PT_SYSTEM -->
<!-- PARAM NAME=system_attribute_name1 -->
<!-- PARAM NAME=system_attribute_name2 -->
.
.
.
<!-- PARAM NAME=system_attribute_n -->
<!-- /WSLET -->
```

Output:

```
var variable1="system_attribute_name1_value";
var variable2="system_attribute_name2_value";
.
.
.
var variableN="system_attribute_n_value";
```

Use 3 required parameters

None.

Use 3 optional parameters

None.

Use 3 example

Load the requested Rational Change system information.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_PT_SYSTEM -->
<!-- PARAM NAME=database -->
<!-- PARAM NAME=dcm_dbid -->
<!-- PARAM NAME=dcm_delimiter -->
<!-- PARAM NAME=pt_app -->
<!-- PARAM NAME=pt_app_role -->
<!-- /WSLET -->
```

Output:

```
var dcm_dbid = "text";
var pt_app = "text";
var dcm_delimiter = "text";
var database = "text";
var pt_app_role = "text";
```

Use 4

CCM_SECURITY_DATA

Obtain the requested security information as JavaScript variables.

This call is required on all forms that perform actions against the Rational Synergy database, for example, when loading the main button bar.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_SECURITY_DATA -->
<!-- PARAM NAME=CCM_security_variable1 -->
<!-- PARAM NAME=CCM_security_variable2 -->
.
.
.
<!-- PARAM NAME=CCM_security_variableN -->
<!-- /WSLET -->
```

Output:

```
var variable1="security_variable1_value";
var variable2="security_variable2_value";
.
.
.
var variableN="security_variableN_value";
```

Use 4 required parameters

You must provide one or more of the following security parameters:

CCM_TOKEN Specified the user session ID.

CCM_ROLE Specifies the login role.

CCM_LIFECYCLE Specifies the lifecycle name.

CCM_TIMESTAMP Specifies when the form was loaded.

CCM_USER Specifies the user name.

CCM_VERSION Specifies the Rational Change template version (VALUE="*template_value*").

CCM_DELIMITER Specifies the delimiter value; for example, "|"

CCM_SUB_DELIMITER Specifies the colon.

CRLF **S**pecifies the line separation character sequence for OS.

Use 4 optional parameters

None.

Use 4 example

Load security data.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_SECURITY_DATA -->
<!-- PARAM NAME=CCM_TOKEN -->
<!-- PARAM NAME=CCM_ROLE -->
```

```
<!-- PARAM NAME=CCM_LIFECYCLE -->
<!-- PARAM NAME=CCM_TIMESTAMP -->
<!-- PARAM NAME=CCM_USER -->
<!-- PARAM NAME=CCM_VERSION VALUE="4.0" -->
<!-- PARAM NAME=CCM_DELIMITER -->
<!-- PARAM NAME=CCM_SUB_DELIMITER -->
<!-- PARAM NAME=CRLF -->
<!-- /WSLET -->
```

Output:

```
var token="5877386030747271350";
var timestamp="05/02/2001 08:45:48";
var template_version="4.0";
var role="User";
var user= "text";
var delimiter ="|";
var sub_delimiter =":";
var crlf ="\r\n";
```

Use 5

CCM_TEMPLATE_INFO

Output an HTML comment that shows the name of the template file being processed.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_TEMPLATE_INFO -->
<!-- /WSLET -->
```

Output:

```
<!-- Template Loading Information
Loaded_File = relative_path_in_template_directory
Loaded_File_Path = absolute_path_to_template_directory -->
```

Use 5 required parameters

None.

Use 5 optional parameters

None.

Use 5 example

Show the name of the template file currently being processed.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_TEMPLATE_INFO -->
<!-- /WSLET -->
```

Output:

```
<!-- Template Loading Information
Loaded_File = pt\forms\admin_framework/admin_dflt.html
Loaded_File_Path =
D:\cs_installs\build_15\<context>\webapps\synergy\WEB-
INF\wsconfig\templates
 -->
```

Use 6

CCM_USER_PROFILE

Obtain the requested user profile information, as JavaScript variables, from the user's `username.cfg` file in the Rational Synergy installation area.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_USER_PROFILE -->
<!-- PARAM NAME=profile_attribute_name1 -->
<!-- PARAM NAME=profile_attribute_name2 -->
.
.
.
<!-- PARAM NAME=profile_attribute_n -->
<!-- /WSLET -->
```

Output:

```
var
profile_attribute_name1="profile_attribute_name1_value";
var
profile_attribute_name2="profile_attribute_name2_value";
.
.
.
var profile_attribute_n="profile_attribute_n_value";
```

Use 6 required parameters

None.

Use 6 optional parameters

None.

Use 6 example

Load user profile data.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_USER_PROFILE -->
<!-- PARAM NAME=user_email -->
<!-- PARAM NAME=user_fax -->
<!-- PARAM NAME=user_name -->
<!-- PARAM NAME=user_product_name -->
<!-- PARAM NAME=user_address -->
<!-- PARAM NAME=user_phone -->
<!-- PARAM NAME=user_company -->
<!-- /WSLET -->
```

Output:

```
var user_email = "text";
var user_fax = "text";
var user_name = "text";
var user_product_name = "text";
var user_company = "text";
var user_phone = "text";
var user_address = "text";
```

Use 7

CCM_WINDOW_LABEL

Displays user information in the window caption. The format of the string returned is: *`user_name (user_role); user_database_alias`*.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_WINDOW_LABEL -->
<!-- /WSLET -->
```

Use 7 required parameters

None.

Use 7 optional parameters

None.

Use 7 example

Show user information:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_WINDOW_LABEL -->
<!-- /WSLET -->
```

Output:

```
bob (User): ccm_model
```

Use 8

CCM_USER_PROFILE_READONLY

Determines if the requested user profile attributes are read only. This is dependent on how the LDAP is configured. A JavaScript data structure is returned in place of the wslet call.

Data structure format used is:

```
    var user_preference_RO = new Object();
    user_preference_RO['user_profile_attribute_name'] =
true|false;
    .
    .
    .
```

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_WINDOW_LABEL -->
<!-- /WSLET -->
```

Use 8 required parameters

None.

Use 8 optional parameters

user_profile_attribute An attribute name, must have a CCM_USER_PROFILE setting, and optionally, a CCM_LDAP_MAPPING mapping.

None.

Use 8 example

The install is using the bundled SDS server.

```
[CCM_SYSTEM][LDAP_BUNDLED]true[/LDAP_BUNDLED][/CCM_SYSTEM]
```

- Attributes are defined to be CCM_USER_PROFILE attributes.

```
[CCM_USER_PROFILE]
    [NAME]user_first_name[/NAME]
    [TYPE]CCM_STRING[/TYPE]
[/CCM_USER_PROFILE]
[CCM_USER_PROFILE]
    [NAME]user_last_name[/NAME]
    [TYPE]CCM_STRING[/TYPE]
[/CCM_USER_PROFILE]
[CCM_USER_PROFILE]
    [NAME]user_email[/NAME]
    [TYPE]CCM_STRING[/TYPE]
    [ATTRIBUTE]submitter_email[/ATTRIBUTE]
[/CCM_USER_PROFILE]
```

- Attributes are defined to have CCM_LDAP_MAPPING LDAP mappings.

```
[CCM_LDAP_MAPPING]
    [NAME]user_first_name[/NAME]
    [LDAP_ATTRIBUTE]givenname[/LDAP_ATTRIBUTE]
[/CCM_LDAP_MAPPING]
[CCM_LDAP_MAPPING]
    [NAME]user_last_name[/NAME]
    [LDAP_ATTRIBUTE]sn[/LDAP_ATTRIBUTE]
[/CCM_LDAP_MAPPING]
[CCM_LDAP_MAPPING]
    [NAME]user_email[/NAME]
    [LDAP_ATTRIBUTE]mail[/LDAP_ATTRIBUTE]
    [ALIAS]user_submitter_email[/ALIAS]
[/CCM_LDAP_MAPPING]
```

Wslet call.

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=CCM_USER_PROFILE_READONLY -->
<!-- PARAM NAME=user_first_name -->
<!-- PARAM NAME=user_last_name -->
<!-- PARAM NAME=user_email -->
<!-- /WSLET -->
```

Output:

```
var user_preference_RO = new Object();

user_preference_RO['user_first_name'] = false;
user_preference_RO['user_last_name']  = false;
user_preference_RO['user_email']      = false;
```

Use 9

LOAD_CVID_LIST

Obtains the global form submit variable "*CVID_LIST*" as a JavaScript variable, which is used in the bulk transition feature.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=LOAD_CVID_LIST -->
<!-- /WSLET -->
```

Use 9 required parameters

None.

Use 9 optional parameters

None.

Use 9 example

Obtain the CVID list:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=LOAD_CVID_LIST -->
<!-- /WSLET -->
```

Output:

```
var CVID_LIST = unescape("cvid_1|cvid_2|cvid_3")
```

Use 10

MESSAGE_OF_THE_DAY

Determines the settings of the message of the day and returns the values accordingly.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=MESSAGE_OF_THE_DAY VALUE=JSP/HTML/MOTD_TYPE -->
<!-- /WSLET -->
OR
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=MESSAGE_OF_THE_DAY -->
<!-- /WSLET -->
```

If the value is not supplied then HTML is considered.

Use 10 required parameters

None.

Use 10 optional parameters

None.

Use 10 example

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=MESSAGE_OF_THE_DAY VALUE=JSP/HTML/MOTD_TYPE -->
<!-- /WSLET -->
```

Output:

For value HTML:

```
var isMotdUpdated = false;
var motdToggle = false;
var motdToggle = false;
var motdType = 'text';
```

For value JSP:

```
'true'/'<blank_string>'
```

For value MOTD_TYPE:

```
'text'/'html'
```

Use 11

SERVER_TYPE

Tests which type of Change server is installed. Returns a Javascript boolean variable (`isRemoteServer, isCentralServer`, or `isStandAloneServer`) based on the server type (`STAND_ALONE, CENTRAL_SERVER`, or `REMOTE_SERVER`) passed to the wslet.

Wslet call:

```
<!-- WSLET CODE=PTLoadData -->
<!-- PARAM NAME=SERVER_TYPE VALUE=STAND_ALONE/CENTRAL_SERVER/
REMOTE_SERVER -->
<!-- /WSLET -->
```

Use 11 required parameters

None.

Use 11 optional parameters

None.

Use 11 example

Output:

```
var isRemoteServer = false;
```

PTReport

PTReport retrieves report data.

PTReport is one of the most complex wslets and is used in report template files (that is, main, header, footer, attribute, label, group, and image templates) and inline reports on Show forms.

PTReport can also run spreadsheet-style metrics on the report data. The basic metrics come in the following styles:

- REPORT_ATTR_METRIC

  Creates a report based on the attributes of a single report object (row-based).
- REPORT_COL_METRIC

  Creates a report based on a single attribute across report objects (column-based).
- REPORT_ROW_METRIC

  Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

You must perform metrics on homogeneous attributes; that is, on attributes of the same type.

Use 1

GET_REPORT_DATA

Get the report's configuration data as JavaScript.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=GET_REPORT_DATA -->
<!-- PARAM NAME=config_item_name1 -->
<!-- PARAM NAME=config_item_name2 -->
.
.
.
<!-- PARAM NAME=config_item_nameN -->
<!-- /WSLET -->
```

Use 1 required parameters

config_item_name Specifies the configuration data item. You must specify at least one.

The report configuration items are:

- REPORT_TITLE
- CHOSEN_REPORT
- CHOSEN_QUERY
- QUERY_STRING
- ATTRIBUTES_*i*
- SORT_ORDER_*i*
- CUSTOM_WSLET_*i*
- XML_CONTENT_*i*

Use 1 optional parameters

None.

Use 1 example

Get the report configuration data as JavaScript.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=GET_REPORT_DATA -->
<!-- PARAM NAME=REPORT_TITLE -->
<!-- PARAM NAME=CHOSEN_REPORT -->
<!-- PARAM NAME=CHOSEN_QUERY -->
<!-- PARAM NAME=QUERY_STRING -->
<!-- PARAM NAME=ATTRIBUTES_0 -->
<!-- PARAM NAME=ATTRIBUTES_1 -->
<!-- PARAM NAME=ATTRIBUTES_2 -->
<!-- PARAM NAME=SORT_ORDER_0 -->
<!-- PARAM NAME=SORT_ORDER_1 -->
<!-- PARAM NAME=SORT_ORDER_2 -->
<!-- PARAM NAME=CUSTOM_WSLET_0 -->
<!-- PARAM NAME=CUSTOM_WSLET_1 -->
<!-- PARAM NAME=CUSTOM_WSLET_2 -->
<!-- PARAM NAME=XML_CONTENT_0 -->
<!-- PARAM NAME=XML_CONTENT_1 -->
<!-- PARAM NAME=XML_CONTENT_2 -->
<!-- /WSLET -->
```

Output:

```
var REPORT_TITLE = "text";
var CHOSEN_REPORT = "text";
var CHOSEN_QUERY = "text";
var QUERY_STRING = "text";
var ATTRIBUTES_0 = "text";
var SORT_ORDER_0 = "text";
var CUSTOM_WSLET_0 = "text";
var XML_CONTENT_0 = "text";
var ATTRIBUTES_1 = "text";
var SORT_ORDER_1 = "text";
var CUSTOM_WSLET_1 = "text";
var XML_CONTENT_1 = "text";
var ATTRIBUTES_2 = "text";
var SORT_ORDER_2 = "text";
var CUSTOM_WSLET_2 = "text";
var XML_CONTENT_2 = "text";
```

Use 2

INLINE_IMMEDIATE_REPORT

Runs a Rational Change report and places the contents on a CR show or Task show form.

Template scope: attribute and problem_attr_show_form, problem_show_form, task_show_form, task_attr_show_form, object_show_form, object_attr_show_form

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=CHOSEN_REPORT VALUE="single-item_report_name"
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 2 required parameters

INLINE_IMMEDIATE_REPORT The VALUE is the report to run. The VALUE must be a CCM_REPORT defined in a configuration file. VALUE="*report name*"

Use 2 optional parameters

INLINE_IMMEDIATE_QUERY_STRING The query string to use when running the immediate report. The query defined for the report will be used if this option is omitted. VALUE="*custom query string*"

INLINE_IMMEDIATE_ERROR_VALUE If an error occurs while running the report, this value will be displayed in place of a report. VALUE="*report error message*"

INLINE_IMMEDIATE_EMPTY_STRING If no result is found after running the report, this value will be output in place of a report. VALUE="*no results found message*"

INLINE_IMMEDIATE_NO_MAIN Allows you to over-ride the main report definition with this file. The Rational Change provide default s "sub_report_no_main_rpt.html". An over-ride file (if used) must exist in the CHANGE_APP_HOME/.../wsconfig/templates/pt/reports/*your_file.html*. The file does not have to have a .html extension. VALUE="*over-ride for default*"

The default main file provided by Rational Change has the following PTReport options:

```
    <!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_HEADER --
><!-- /WSLET -->
    <!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_ATTRS --
><!-- /WSLET -->
    <!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_IMAGE --
><!-- /WSLET -->
    <!-- WSLET CODE=PTReport --><!-- PARAM NAME=REPORT_FOOTER --
><!-- /WSLET -->
```

A header, attribute, image, and footer template definitions will be included if they exist for the chosen report.

Use 2 example

Report definition used in bulk transition operation.

```
[CCM_REPORT]
    [NAME]Bulk Transition[/NAME]
    [QUERY]All CRs[/QUERY]
    [PROBLEM_DEF]prob_transition_summary[/PROBLEM_DEF]
    [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
```

```
        [DESCRIPTION]List of problem numbers, status and
synopsis[/DESCRIPTION]
    [/CCM_REPORT]

    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=INLINE_IMMEDIATE_REPORT VALUE="Bulk
Transition" -->
    <!-- /WSLET -->
```

Output:

A Rational Change report.

Use 3

OFFLINE_DATABASES

When running in central server mode, returns an HTML block with a warning message listing the offline databases.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=OFFLINE_DATABASES -->
<!-- PARAM NAME=TRUNCATE
<!-- /WSLET -->
```

Use 3 required parameters

None.

Optional parameter

TRUNCATE CIf present, only a short list of the offline databases are shown. The rest are hidden behind a (JavaScript) "more..." link. This can be used to keep the warning message short when many databases are offline.

Use 4

REPORT_ALLOW_MODIFY

Displays an image hyperlink that runs a single-item report. The form is loaded in the context of a report object; that is, it has access to the current `problem_number`, `task_number`, or `cvid`.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=CHOSEN_REPORT VALUE="single-item_report_name"
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 4 required parameters

CHOSEN_REPORT Causes the hyperlink to run the single-item report.

REPORT_LINK_IMAGE Specifies the image to use for the link. `VALUE="`*`image_name`*`"`.

Use 4 optional parameters

REPORT_TITLE Gives the report a title. By default, the title is "*`object`* Detail Report", where *`object`* is CR, Task, or Object, depending on the subreport type. VALUE="*`report_title`*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*`target_window_name`*". By default, "`NoFrame`" is used.

WINDOW_OPTIONS Defines the window decorations for the pop-up window. If the window is already open, the window options are ignored. By default, the window options are "status,resizable,width=700,height=600". VALUE="*`JavaScript_window_options`*".

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used.

Only applicable when using an date attribute value link. VALUE="*`date_format`*".

REPORT_IMAGE_PROMPT Specifies alternative text to use with an image. Most browsers display this text 1) when a user mouses over an image 2) if the browser is not capable of displaying images. Only applicable when using an image link. VALUE="*`alternative_text`*".

Use 4 example

Use an image as a link to show a detailed report on an object.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=CHOSEN_REPORT VALUE=objectdetail -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="/trapeze/ptimages/
ws_object.gif" -->
<!-- /WSLET -->

<A HREF="javascript:void window.open(new String('/servlet/
com.continuus.webpt.servlet.PTaction?ACTION_FLAG=objectreport&r
ole=User&token=8439147065079165807&CHOSEN_REPORT=%6c&cvid=%31')
,'NoFrame','status,resizable,width=700,height=600')" >
<IMG SRC="/trapeze/ptimages/ws_object.gif" BORDER=0 ALT="View
results">
</A>
```

Use 5

REPORT_ALLOW_MODIFY

Display an attribute hyperlink that runs a single-item report. The form is loaded in the context of a particular report object; that is, it has access to the current `problem_number`, `task_number`, or `cvid`.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=CHOSEN_REPORT VALUE="single-item_report_name"
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE="link_attribute" -->
<!-- /WSLET -->
```

Use 5 required parameters

CHOSEN_REPORT Causes the hyperlink to run the single-item report.

REPORT_LINK_SHOW Specifies the attribute to use for the link. VALUE=*"link_attribute"*.

Use 5 optional parameters

REPORT_TITLE Gives the report a title. By default, the title is "*object* Detail Report", where *object* is CR, Task, or Object, depending on the subreport type. VALUE="*report_title*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*target_window_name*". By default, "`NoFrame`" is used.

WINDOW_OPTIONS Defines the window decorations for the pop-up window. If the window is already open, the window options are ignored. By default, the window options are "status,resizable,width=700,height=600". VALUE="*JavaScript_window_options*".

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used.

Only applicable when using an date attribute value link. VALUE="*date_format*".

Use 5 example

Use the display name of an object as a link to show a detailed report.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=CHOSEN_REPORT VALUE=objectdetail -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE=displayname -->
<!-- PARAM NAME=REPORT_TITLE VALUE="Object Information" -->
<!-- /WSLET -->
```

Output:

```
<A HREF="javascript:void window.open(new String('/servlet/
com.continuus.webpt.servlet.PTaction?ACTION_FLAG=objectreport&r
ole=User&token=8439147065079165807&CHOSEN_REPORT=%6c&REPORT_TIT
LE=%74&cvid=%32'),'NoFrame','status,resizable,width=700,height=
600')" >
test.html-1
</A>
```

Use 6

REPORT_ALLOW_MODIFY

Display an image hyperlink that loads a form (for example, show or transition form). The form is loaded in the context of a particular report object; that is, it has access to the current `problem_number`, `task_number`, or `cvid`.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE="template_name" -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 6 required parameters

REPORT_MODIFY_POPUP Specifies the template to use for the modify form. VALUE=*template_name*.

REPORT_LINK_IMAGE Specifies the image to use for the link. `VALUE="`*`image_name`*`"`.

Use 6 optional parameters (either wslet call)

REPORT_FORM_TYPE Specifies the form type of template specified by REPORT_MODIFY_POPUP. By default, "*`frameset_form`*" is used. VALUE="*`form_type`*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*`target_window_name`*". By default, "`NoFrame`" is used.

WINDOW_OPTIONS Determine the window decorations for the pop-up window. If the window is already open, the window options are ignored. By default, the window options are "status,resizable,width=700,height=600". VALUE="*`JavaScript_window_options`*".

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. Only applicable when using an date attribute value link. VALUE="*`date_format`*".

REPORT_IMAGE_PROMPT Specifies alternative text to use with an image. Most browsers display this text 1) when a user mouses over an image 2) if the browser is not capable of displaying images. Only applicable when using an image link. VALUE="*`alternative_text`*".

Use 6 example

Display an arrow icon as a link, which, when clicked, brings up the transition dialog for that CR.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="/trapeze/ptimages/
ws_trans_arrow.gif" -->
<!-- PARAM NAME=REPORT_IMAGE_PROMPT VALUE="Transition" -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE=ProblemTransitionView
-->
<!-- PARAM NAME=REPORT_TARGET_FRAME VALUE=REPORT_UNIQUE -->
```

```
<!-- PARAM NAME=WINDOW_OPTIONS
VALUE="status,resizable,width=800,height=600" -->
<!-- /WSLET -->
```

Output:

```
<A HREF="javascript:var w = window.open('/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=frameset_form&TEM
PLATE_FLAG=%77&role=User&token=8439147065079165807&problem_numb
er=%31','','status,resizable,width=800,height=600')"
ONMOUSEOVER="window.status = 'Load modification form.';
return(true);"><IMG SRC="/trapeze/ptimages/ws_trans_arrow.gif"
BORDER=0 ALT="Transition"></A>
```

Use 7

REPORT_ALLOW_MODIFY

Display an attribute hyperlink that loads a form (for example, show or transition form). The form is loaded in the context of a particular report object; that is, it has access to the current `problem_number`, `task_number`, or `cvid`.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE="template_name" -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE="link_attribute" -->
<!-- /WSLET -->
```

Use 7 required parameters

REPORT_MODIFY_POPUP Specifies the template to use for the modify form. VALUE=*`template_name`*.

REPORT_LINK_SHOW Specifies the attribute to use for the link. VALUE=*`"link_attribute"`*.

Use 7 optional parameters (either wslet call)

REPORT_FORM_TYPE Specifies the form type of template specified by REPORT_MODIFY_POPUP. By default, "*`frameset_form`*" is used. VALUE="*`form_type`*".

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*target_window_name*". By default, "NoFrame" is used.

WINDOW_OPTIONS Determine the window decorations for the pop-up window. If the window is already open, the window options are ignored. By default, the window options are "status,resizable,width=700,height=600". VALUE="*JavaScript_window_options*".

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. Only applicable when using an date attribute value link. VALUE="*date_format*".

Use 7 example

Display problem number as a link which, when clicked, brings up information view for that change request.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE=problem_number -->
<!-- PARAM NAME=REPORT_MODIFY_POPUP VALUE=ProblemReportView -->
<!-- /WSLET -->
```

Output:

```
<A HREF="javascript:var w = window.open('/servlet/
com.continuus.webpt.servlet.PTweb?ACTION_FLAG=frameset_form&TEM
PLATE_FLAG=%77&role=User&token=8439147065079165807&problem_numb
er=%31','NoFrame','status,resizable,width=700,height=600')"
>1</A>
```

Use 8

REPORT_ALLOW_MODIFY

Display an image hyperlink that downloads an attachment.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
```

```
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="image_name" -->
<!-- /WSLET -->
```

Use 8 required parameters

REPORT_VIEW_ATTACHMENT Causes the link to download the attachment.

REPORT_LINK_IMAGE Specifies the image to use for the link. `VALUE="image_name"`.

Use 8 optional parameter

REPORT_IMAGE_PROMPT Specifies alternative text to use with an image. Most browsers display this text 1) when a user mouses over an image 2) if the browser is not capable of displaying images. Only applicable when using an image link. VALUE="*alternative_text*".

Use 8 example

Display a link to an attachment using an image.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- PARAM NAME=REPORT_LINK_IMAGE VALUE="/trapeze/ptimages/
ws_attachment.gif"
-->
<!-- PARAM NAME=REPORT_IMAGE_PROMPT VALUE="Download attachment"
-->
<!-- /WSLET -->
```

Output:

```
<A HREF="/servlet/
com.continuus.webpt.servlet.PTaction?ACTION_FLAG=view_attachmen
t&role=User&token=8439147065079165807&cvid=%30"
ONMOUSEOVER="window.status = 'Download Attachment.';
return(true);">
<IMG SRC="/trapeze/ptimages/ws_attachment.gif" BORDER=0
ALT="Download attachment">
</A>
```

Use 9

REPORT_ALLOW_MODIFY

Display an attribute hyperlink that downloads an attachment.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE="link_attribute" -->
<!-- /WSLET -->
```

Use 9 required parameters

REPORT_VIEW_ATTACHMENT Causes the link to download the attachment.

REPORT_LINK_SHOW Specifies the attribute to use for the link. VALUE=*"link_attribute"*.

Use 9 optional parameters

None.

Use 9 example

Display a textual link to an attachment using the attachment name.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ALLOW_MODIFY -->
<!-- PARAM NAME=REPORT_LINK_SHOW VALUE=attachment_name-->
<!-- PARAM NAME=REPORT_VIEW_ATTACHMENT -->
<!-- /WSLET -->
```

Output:

```
<A HREF="/servlet/
com.continuus.webpt.servlet.PTaction?ACTION_FLAG=view_attachmen
t&role=User&token=843914706507916580 7&cvid=%30"
ONMOUSEOVER="window.status = 'Download Attachment.';
return(true);">
event.log
</A>
```

Use 10

REPORT_ATTR_ALIAS

Get the label of the named attribute.

Template scope: Any

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ATTR_ALIAS VALUE="attribute_name" -->
<!-- /WSLET -->
```

Use 10 required parameters

None.

Use 10 optional parameters

None.

Use 10 example

Show the `product_name` attribute label.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ATTR_ALIAS VALUE=product_name -->
<!-- /WSLET -->
```

Output:

```
Product Name
```

Use 11

REPORT_ATTR_VALUE_COUNT

Count the number of query items that have the specified attribute value.

Template scope: main, header, footer, and group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ATTR_VALUE_COUNT VALUE="attribute_name"
-->
<!-- PARAM NAME=REPORT_ATTR_VALUE VALUE="attribute_value" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 11 required parameters

REPORT_ATTR_VALUE Specifies the attribute value. VALUE="*attribute_value*".

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 11 optional parameters

REPORT_METRIC_LABEL Label the count using the form "*label*:*count*". VALUE="*label*:*count*".

REPORT_GRP_METRIC Limit the count to items within the current group. This parameter should be used inside the group template.

Use 12

REPORT_ATTR_VALUE_COUNT

Count the number of query items that have the specified date attribute value.

Template scope: main, header, footer, and group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ATTR_VALUE_COUNT VALUE="date_attribute"
-->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 12 required parameters

REPORT_DATATYPE=date Specifies the type of attribute. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 12 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. The date value must be consistent with the REPORT_FROM_DATE_FORMAT declaration in the configuration file or in the format override used in this wslet call. VALUE="*date_value*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_METRIC_LABEL Label the count using the form "*label*:*count*". VALUE="*label*:*count*".

REPORT_GRP_METRIC Limit the count to items within the current group. This parameter should be used inside the group template.

Use 13

REPORT_ATTRS

Process and output the results of the attribute template (defined in a subreport definition) of a report. The attribute template displays the actual results of the query; that is, the attributes of a CR, task, or object. PTReport loops over each item in the query results, parsing the attribute template once for each iteration. Grouping templates and subreports are also processed at this time.

While processing each item in the query results, either of the following can happen:

- If a grouping template was specified, and the value of the group by attribute changes, the grouping template will be parsed and output
- If a query item has associated objects, they are parsed and output

Template scope: main

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ATTRS -->
<!-- /WSLET -->
```

Use 13 required parameters

None.

Use 13 optional parameters

None.

Use 14

REPORT_AUTO_ATTR_NAME

Get the label of an auto attribute.

Template scope: auto attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AUTO_ATTR_NAME -->
<!-- /WSLET -->
```

Use 14 required parameters

None.

Use 14 optional parameter

REPORT_NULL_VALUE Specifies a string to be used if the attribute label is the empty string. VALUE="*string*".

Use 15

REPORT_AUTO_ATTR_VALUE

Get the value of an attribute automatically. This option has a rich set of parameters to affect the attribute presentation, including the ability to create hyperlinks.

Special note when using hyperlinks:

- Unlike the REPORT_ALLOW_MODIFY option, REPORT_AUTO_ATTR_VALUE only allows for textual links, not image links.
- The parameters WINDOW_OPTIONS and REPORT_TARGET_FRAME are shared for all links.

Template scope: auto attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AUTO_ATTR_VALUE -->
<!-- /WSLET -->
```

Use 15 required parameters

None.

Use 15 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_NULL_VALUE Specifies a string to be used if the attribute label is the empty string. VALUE="*string*".

REPORT_ENCODING Performs the substitutions defined by the encoding group. VALUE=*encoding_group*. The default encoding group is "HTML".

REPORT_PRE Surrounds the attribute value in <PRE></PRE> tags, which preserves the line spacing.

or

REPORT_TEXT_AREA Puts the attribute value inside an HTML text area control. This is useful for multi-line attributes, because the text area is a fixed size and scrollable.

Optional parameters for REPORT_TEXT_AREA:

REPORT_ROWS Specifies the number of rows in the text area. VALUE=*number_of_rows.* The default is 5.

REPORT_COLS Specifies the number of columns in the text area. VALUE=*number_of_columns*. The default is 60.

REPORT_WRAP Defines the wrapping behavior of the text area. VALUE=[`virtual`|`physical`]. The default is `virtual`.

AUTO_ATTR_REPORT_LOG Formats the value as a log attribute if the user is on the named attribute. `VALUE="`*`attribute_name`*`"`.

Optional parameters for AUTO_ATTR_REPORT_LOG:

RTF_FORMAT Returns the attribute value in RTF format instead of HTML.

TRANSITION_FONT Specifies font attributes for the transition section header. All HTML <FONT> element attributes are valid. VALUE="*`font_options`*".

MODIFY_FONT Specifies font attributes for the modification section header. All HTML <FONT> element attributes are valid. VALUE="*`font_options`*".

NOTE_FONT Specifies font attributes for the notes section header. All HTML <FONT> element attributes are valid. VALUE="*`font_options`*".

SHOW_TRANSITION_COMMENTS Indicates whether to show transition comments. VALUE=[`TRUE` | `FALSE`].

SHOW_MODIFY_EVENTS Indicates whether to show modify events. VALUE=[`TRUE` | `FALSE`].

SHOW_NOTES Indicates whether to show notes. VALUE=[`TRUE` | `FALSE`].

AUTO_ATTR_ALLOW_MODIFY

and

AUTO_ATTR_MODIFY_POPUP Creates a hyperlink from the attribute value if the user is on the named attribute. Both or neither of the options above should be provided. `VALUE="`*`attribute_name`*`"` and VALUE="*`template_name`*", respectively.

Optional parameter for AUTO_ATTR_ALLOW_MODIFY and AUTO_ATTR_MODIFY_POPUP:

AUTO_ATTR_MODIFY_FORM_TYPE Specifies the form type of template specified by AUTO_ATTR_MODIFY_POPUP. By default, *`frameset_form`* is used. VALUE="*`form_type`*".

AUTO_ATTR_TRANSITION

and

AUTO_ATTR_TRANSITION_POPUP Creates a transition hyperlink from the attribute value if the user is on the named attribute. Both or neither of the options above should be provided. `VALUE="`*`attribute_name`*`"` and VALUE="*`template_name`*".

Optional parameter for AUTO_ATTR_ALLOW_MODIFY and AUTO_ATTR_MODIFY_POPUP:

> **AUTO_ATTR_TRANSITION_FORM_TYPE** Specifies the form type of template specified by REPORT_MODIFY_POPUP. By default, "*`frameset_form`*" is used. VALUE="*`form_type`*".

AUTO_ATTR_VIEW_ATTACHMENT Creates a hyperlink from the specified attribute value to download an attachment if the user is on the named attribute. This option would only be used on an attribute template for objects, since attachments are objects. `VALUE="`*`attribute_name`*`"`.

AUTO_ATTR_CHOSEN_REPORT Creates a hyperlink from the specified attribute value to run a single-item report if the user is on the named attribute. `VALUE="`*`attribute_name`*`"`.

Optional parameter for AUTO_ATTR_CHOSEN_REPORT:

> **REPORT_TITLE** Gives the report a title. By default, the title is "*`object`* Detail Report", where *`object`* is CR, Task, or Object, depending on the subreport type. VALUE="*`report_title`*"

The following two options are only applicable to links, and are shared across all links:

REPORT_TARGET_FRAME Specifies the window in which to display the linked object.You can use the special value REPORT_UNIQUE so that each link is displayed in its own window. VALUE= "*`target_window_name`*". By default, "`NoFrame`" is used.

WINDOW_OPTIONS Determine the window decorations for the pop-up window. If the window is already open, the window options are ignored. By default, the window options are "status,resizable,width=700,height=600". VALUE="*`JavaScript_window_options`*".

Note `WINDOW_OPTIONS VALUE="`*`window_options`*`"` becomes the “features” parameter of the JavaScript `window.open()` function call.

Use 15 example

Get the value of an attribute automatically.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AUTO_ATTR_VALUE -->
<!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->

<!-- PARAM NAME=AUTO_ATTR_ALLOW_MODIFY VALUE=problem_number -->
<!-- PARAM NAME=AUTO_ATTR_MODIFY_POPUP VALUE=ProblemReportView -
->
<!-- PARAM NAME=AUTO_ATTR_MODIFY_FORM_TYPE
VALUE="frameset_form" -->

<!-- PARAM NAME=AUTO_ATTR_TRANSITION VALUE=crstatus -->
<!-- PARAM NAME=AUTO_ATTR_TRANSITION_POPUP
VALUE=ProblemTransitionView -->
<!-- PARAM NAME=AUTO_ATTR_TRANSITION_FORM_TYPE
VALUE="frameset_form" -->

<!-- PARAM NAME=REPORT_TARGET_FRAME VALUE=REPORT_UNIQUE -->
<!-- PARAM NAME=WINDOW_OPTIONS
VALUE="status,resizable,width=800,height=600" -->

<!-- PARAM NAME=AUTO_ATTR_REPORT_LOG VALUE=transition_log -->
<!-- PARAM NAME=TRANSITION_FONT VALUE="COLOR=purple" -->
<!-- PARAM NAME=MODIFY_FONT VALUE="COLOR=red" -->
<!-- PARAM NAME=NOTE_FONT VALUE="COLOR=blue" -->
<!-- PARAM NAME=SHOW_TRANSITION_COMMENTS VALUE=true -->
<!-- PARAM NAME=SHOW_MODIFY_EVENTS VALUE=true -->
<!-- PARAM NAME=SHOW_NOTES VALUE=true -->
<!-- /WSLET -->
```

Output:

The result depends on the attribute (and value). The `transition_log` attribute is formatted in log format, with color section headings. Attributes with null values will return "`&nbsp;`". The `crstatus` attribute will link to a transition form, while `problem_number` will link to a show form. Other values will have HTML encoding performed, if necessary (default).

Use 16

REPORT_AUTO_GENERATE

Generate the attribute values for an item in the query results. This is done by processing the auto attribute template or span attribute template (defined in a configuration entry) once for each attribute of an item in the query results. The order of the attribute values is defined in the [ATTRS] section of the subreport.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AUTO_GENERATE -->
<!-- /WSLET -->
```

Use 16 required parameters

None.

Use 16 optional parameters

None.

Use 17

REPORT_AUTO_GENERATE_LABELS

Generate the attribute labels for a column-style report. This is done by processing the auto label template (defined in a subreport definition) once for each attribute. The order of the labels is defined in the [ATTRS] section of the subreport.

Template scope: label

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AUTO_GENERATE_LABELS -->
<!-- /WSLET -->
```

Use 17 required parameters

None.

Use 17 optional parameters

None.

Use 18

REPORT_AVG

Calculate the average of the specified attributes within a single report object.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
result = (attribute_name1 + attribute_name2 + ... +
attribute_n) / n
```

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- /WSLET -->
```

Use 18 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int`|`float`].

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 18 optional parameters

None.

Use 19

REPORT_AVG

Calculate the average of the specified attribute across report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   result += attribute
   result = result / (# of report objects with non-null,
   non-empty attribute value)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 19 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int`|`float`].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 19 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 20

REPORT_AVG

Calculate the average of the specified attributes for each row, then average all the row subtotals. This is equivalent (except for rounding differences) to calculating the average of all occurrences, both within and across query items.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   result += (attribute_name1 + attribute_name2 + ... +
attribute_n) / n
result = result / (# of report objects with non-null, non-
empty attribute value)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 20 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 20 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 21

REPORT_AVG

Calculate the average difference between the specified date attributes and the *from_date*, within a single report object.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
result = ((attribute_name1 - from_date) + (attribute_name2
- from_date)
+ ... + (attribute_n - from_date)) / n
```

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 21 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 21 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 22

REPORT_AVG

Calculate the average of the differences between the specified date attribute and the *from_date*, across report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   result += (attribute_name - from_date)
result = result / (# of report objects with non-null, non-
empty attribute value)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 22 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 22 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 23

REPORT_AVG

Calculate the average of the differences between the specified date attributes and the *from_date* within a report object, then average those across report objects. That is, get a subtotal of the average of the differences for each row, then get the average of those subtotals.

Note Row-level averages include null and empty values, while the column-level average excludes them.

```
for each report object
   result += (attribute_name1 - from_date) +
(attribute_name2 - from_date)
   + ... + (attribute_n - from_date)
result = result / (# of report objects with non-null, non-
empty attribute value)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 23 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 23 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 24

REPORT_CHART

Produce a chart of the query results.

Template scope: main and image

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_CHART
VALUE=[REPORT_HBAR_CHART|REPORT_VBAR_CHART|REPORT_PIE_CHART|REP
ORT_LINE_CHART] -->
<!-- PARAM NAME=REPORT_CHART_TITLE VALUE="title" -->
<!-- PARAM NAME=REPORT_CHART_ON VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_CHART_WIDTH VALUE="height" -->
<!-- PARAM NAME=REPORT_CHART_HEIGHT VALUE="width" -->
<!-- /WSLET -->
```

Use 24 required parameters

REPORT_CHART_TITLE Specifies the chart title. VALUE="*title*".

REPORT_CHART_ON Specifies the attribute being charted. VALUE="*attribute_name*".

REPORT_CHART_WIDTH Specifies the chart width. VALUE="*height.*"

REPORT_CHART_HEIGHT Specifies the chart height. VALUE="*width*".

Use 25

REPORT_CHECKBOX

Places a check box on the report for each item queried. This check box is used with the bulk transition feature.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_CHECKBOX -->
<!-- /WSLET -->
```

Use 25 required parameters

REPORT_CHECKBOX_VALUE Shows the string beside the check box. If no string is used, only the check box is displayed. VALUE = "*checkbox_description*"

Use 25 optional parameters

None.

Use 25 example

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_CHECKBOX -->
<!-- /WSLET -->
```

Output:

```
<INPUT TYPE=CHECKBOX NAME='bulkTrans' VALUE='entered|12345'
OnClick='TransCheckBox_OnClick(this)'></INPUT>
```

The VALUE is the status and cvid for the queried item. The NAME and OnClick actions are not modifiable.

Use 26

REPORT_COUNT

Get a count of the items in a report.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_COUNT -->
<!-- /WSLET -->
```

Use 26 required parameters

None.

Use 26 optional parameters

None.

Use 26 example

Get a count of the items in a report.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_COUNT -->
<!-- /WSLET -->
```

Use 27

REPORT_DUAL_ATTR_CHART

Produces a chart of the query results using the attribute values of two attributes. The **REPORT_GROUP_DUAL_ATTR_CHART** works identically except that it is used in a grouping template.

Template scope: main, image and group.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DUAL_ATTR_CHART
    VALUE=[REPORT_HBAR_CHART|REPORT_VBAR_CHART|REPORT_PIE_CHART|

REPORT_LINE_CHART|REPORT_STACKED_BAR_CHART|REPORT_STACKED_AREA_
CHART] -->
<!-- PARAM NAME=REPORT_CHART_TITLE VALUE="title" -->
<!-- PARAM NAME=REPORT_ATTR_1 VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_ATTR_2 VALUE="attribute_name" -->
<!-- /WSLET -->
```

Use 27 required parameters

REPORT_ATTR_1 Specifies the attribute whose values are the series in the chart. VALUE="attribute_name". VALUE ="*attribute_name*"

REPORT_ATTR_2 Specifies the attribute whose values are the x-points in the chart. VALUE="attribute_name". VALUE ="*attribute_name*"

Use 27 optional parameterOptional Arguments:

REPORT_CHART_TITLE Specifies the chart title. VALUE="*`title`*".

REPORT_CHART_WIDTH Specifies the chart height. VALUE="*`width`*".

REPORT_CHART_HEIGHT Specifies the chart width. VALUE="*`height`*."

Use 27 example:

Make a stacked bar chart where each x-point is a severity value and the corresponding y-points break down status. The wslet is placed in an image template.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DUAL_ATTR_CHART
VALUE=REPORT_STACKED_BAR_CHART -->
<!-- PARAM NAME=REPORT_CHART_TITLE VALUE="Severity by Status" -
->
<!-- PARAM NAME=REPORT_ATTR_1 VALUE=crstatus -->
<!-- PARAM NAME=REPORT_ATTR_2 VALUE=severity -->
<!-- /WSLET -->
```

Output:

An Applet call to load the requested chart.

```
    <APPLET CODE=javachart.applet.stackColumnApp.class
CODEBASE='/trapeze'
           WIDTH=650 HEIGHT=450>
    <PARAM NAME=CopyrightNotification VALUE="KavaChart is a
copyrighted work,
          and subject to full legal protection">
    <PARAM NAME=dataset0Name VALUE="concluded">
    .
    .
    .
    </APPLET>
```

or

An image tag that points to a chart image produced by a KavaChart servlet.

```
<IMG . . . http://host:port/ . . .
```

Use 28

REPORT_DATE

Get the time at which the query was run.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DATE -->
<!-- /WSLET -->
```

Use 28 required parameters

None.

Use 28 optional parameter

REPORT_DATE_FORMAT Indicates how the date should be formatted. If not provided, the REPORT_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 28 example

Get the time at which the query was run.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DATE -->
<!-- PARAM NAME=REPORT_DATE_FORMAT VALUE="EEEE, MMMM dd, yyyy
h:mm:ss a" -->
<!-- /WSLET -->
```

Output:

```
Friday, May 11, 2001 9:00:49 AM
```

Use 29

REPORT_DETAILS

Get the value of an attribute.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE="attribute_name" -->
<!-- /WSLET -->
```

Use 29 required parameters

None.

Use 29 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_NULL_VALUE Specifies a string to be used if the attribute label is the empty string. VALUE="*string*".

REPORT_ENCODING Performs the substitutions defined by the encoding group. The default encoding group is "HTML". VALUE=*encoding_group*.

REPORT_PRE Surrounds the attribute value in <PRE></PRE> tags, which preserves the line spacing.

or

REPORT_TEXT_AREA Puts the attribute value inside an HTML text area control. This is useful for multi-line attributes, because the text area is a fixed size and scrollable.

Optional parameters for REPORT_TEXT_AREA:

REPORT_ROWS Specifies the number of rows in the text area. VALUE=*number_of_rows*. The default is 5.

REPORT_COLS Specifies the number of columns in the text area. VALUE=*number_of_columns*. The default is 60.

REPORT_WRAP Defines the wrapping behavior of the text area. VALUE=[`virtual`|`physical`]. The default is virtual.

Use 29 example

Get the value of the `problem_synopsis` attribute.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=problem_synopsis -->
<!-- /WSLET -->
```

Output:

```
try to duplicate Pam's problem
```

Get the value of the `problem_description` attribute and define the layout.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DETAILS VALUE=problem_description -->
<!-- PARAM NAME=REPORT_TEXT_AREA -->
<!-- PARAM NAME=REPORT_ROWS VALUE=4 -->
<!-- PARAM NAME=REPORT_COLS VALUE=50 -->
<!-- /WSLET -->
```

Output:

```
<TEXTAREA ROWS=4 COLS=50 WRAP=virtual>text</TEXTAREA>
```

Use 30

REPORT_DIFF

Calculate the difference between the specified attributes (*`attribute_name1`* - *`attribute_name2`*) within a single report object.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF
VALUE="attribute_name1:attribute_name2" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 30 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*`datatype`*"=[`int`|`float`].

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 30 optional parameters

None.

Use 31

REPORT_DIFF

Calculate the difference between the specified date attributes (*attribute_name1* - *attribute_name2* or *attribute_name* - *from_date*) within a single report object.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 31 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 31 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 32

REPORT_DIFF_AVG

Calculate the difference *attribute_name1* - *attribute_name2* for each report object, then average the differences across report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   result += attribute_name1 - attribute_name2
result = result / (# of report objects with non-null, non-
empty values)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_AVG
VALUE="attribute_name1:attribute_name2" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 32 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 32 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 33

REPORT_DIFF_AVG

Calculate the difference *attribute_name1* - *attribute_name2* (or *attribute_name* - *from_date*) for each report object, then average these differences across report object.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   result += attribute_name1 - attribute_name2 (or
attribute_name - from_date)
result = result / (# of report objects with non-null, non-
empty VALUEs)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_AVG
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 33 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 33 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 34

REPORT_DIFF_MAX

Calculate the difference *attribute_name1* - *attribute_name2* for each report object, and compute the maximum of these differences across report objects. If the difference cannot be computed for (for example, if a value cannot be parsed as a numerical value), it is assumed to be 0.

```
for each report object
   determine attribute_name1 - attribute_name2
result = max(attribute_name1 - attribute_name2)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_MAX
VALUE="attribute_name1:attribute_name2" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 34 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 34 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 35

REPORT_DIFF_MAX

Calculate the difference *attribute_name1* - *attribute_name2* (or *attribute_name* - *from_date*) for each report object, then compute the maximum of these differences across report objects:

```
for each report object
   determine attribute_name1 - attribute_name2 (or
attribute_name - from_date)
result = max(attribute_name1 - attribute_name2) (or
max(attribute_name - from_date))
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_MAX
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 35 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 35 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified.
VALUE="*attribute_value*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used.
VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 36

REPORT_DIFF_MIN

Calculate the difference `attribute_name1` - `attribute_name2` for each report object, and compute the minimum of these differences across report objects. If the difference cannot be computed for some reason (for example, a value cannot be parsed as a numerical value), it is assumed to be 0.

```
for each report object
   determine attribute_name1 - attribute_name2
result = min(attribute_name1 - attribute_name2)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_MIN
VALUE="attribute_name1:attribute_name2" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 36 required parameters

REPORT_DATATYPE Specifies the type of data to return.
VALUE="*datatype*"=[`int|float`].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 36 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 37

REPORT_DIFF_MIN

Calculate the difference *attribute_name1* - *attribute_name2* (or *attribute_name* - *from_date*) for each report object, then compute the minimum of these differences across report objects.

```
for each report object
   determine attribute_name1 - attribute_name2 (or
attribute_name - from_date)
result = min(attribute_name1 - attribute_name2)(or
max(attribute_name - from_date))
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_MIN
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 37 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 37 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. In unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 38

REPORT_DIFF_SUM

Calculate the difference `attribute_name1` - `attribute_name2` for each report object, then sum these differences across report objects. If the difference cannot be computed for some reason (for example, a value cannot be parsed as a numerical value), it is assumed to be 0.

```
for each report object
   result += attribute_name1 - attribute_name2
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_SUM
VALUE="attribute_name1:attribute_name2" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 38 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int`|`float`].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 38 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 39

REPORT_DIFF_SUM

Calculate the difference `attribute_name1` - `attribute_name2` (or `attribute_name` - `from_date`) for each report object, then sum these differences across report objects.

```
for each report object
   result += attribute_name1 - attribute_name2 (or
attribute_name - from_date)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_DIFF_SUM
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 39 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 39 optional parameters

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 40

REPORT_FOOTER

Process and output the results of the footer template (defined in a subreport definition) of a report. The footer template typically contains elements like a navigation link.

Template scope: main

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_FOOTER -->
<!-- /WSLET -->
```

Use 40 required parameters

None.

Use 40 optional parameters

None.

Use 41

REPORT_GROUP_BY

Get the current value of the group by attribute.

Template scope: group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_GROUP_BY -->
<!-- /WSLET -->
```

Use 41 required parameters

None.

Use 41 optional parameters

None.

Use 41 example

Get the current value of the group by attribute.Assume the report is grouped by resolver.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_GROUP_BY -->
<!-- /WSLET -->
```

Output:

```
jane
```

Use 42

REPORT_GROUP_CHART

Produce a chart of the query items in the current grouping.

Template scope: group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_GROUP_CHART
VALUE=[REPORT_HBAR_CHART|REPORT_VBAR_CHART|REPORT_PIE_CHART|REP
ORT_LINE_CHART] -->
<!-- PARAM NAME=REPORT_CHART_TITLE VALUE="title" -->
<!-- PARAM NAME=REPORT_CHART_ON VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_CHART_WIDTH VALUE="height" -->
<!-- PARAM NAME=REPORT_CHART_HEIGHT VALUE="width" -->
<!-- /WSLET -->
```

Use 42 required parameters

REPORT_CHART_TITLE Specifies the chart title. VALUE="*title*".

REPORT_CHART_ON Specifies the attribute being charted. VALUE="*attribute_name*."

REPORT_CHART_WIDTH Specifies the chart width. VALUE="*height*".

REPORT_CHART_HEIGHT Specifies the chart height. VALUE="*width*".

Use 43

REPORT_GROUP_COUNT

Return a count of the number of items in the current group (that is, the number of report objects that have the same value for the group by attribute).

Template scope: group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_GROUP_COUNT -->
<!-- /WSLET -->
```

Use 43 required parameters

None.

Use 43 optional parameters

None.

Use 44

REPORT_GROUP_DUAL_ATTR_CHART

Produces a chart of the query results using the attribute values of two attributes in a grouping template.

For more information, see REPORT_DUAL_ATTR_CHART

Template scope: main, image and group.

Use 45

REPORT_HEADER

Process and output the results of the header template (defined in a subreport definition) of a report. The header template typically contains elements like the report title, a print button, the date of the report, and a count of the query results.

Template scope: main

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_HEADER -->
<!-- /WSLET -->
```

Use 45 required parameters

None.

Use 45 optional parameters

None.

Use 46

REPORT_IMAGE

Process and output the results of the image template (defined in a subreport definition) of a report. The image template is typically used for displaying a chart.

Template scope: main

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_IMAGE -->
<!-- /WSLET -->
```

Use 46 required parameters

None.

Use 46 optional parameters

None.

Use 47

REPORT_ITEM_LOCATION_DATABASE

Shows where the report item (CR, task, or object) was found.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_ITEM_LOCATION_DATABASE -->
<!-- /WSLET -->
```

Output:

```
/vol/shark/ccmdb/change 5.2
```

Use 47 required parameters

None.

Use 47 optional parameters

None.

Use 48

REPORT_LABELS

Process and output the results of the label template (defined in a subreport definition) of a report. The label template is intended to produce the column labels for a column-style report.

Template scope: main

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_LABELS -->
<!-- /WSLET -->
```

Use 48 required parameters

None.

Use 48 optional parameters

None.

Use 49

REPORT_LOG

Get the named attribute's value in log format.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_LOG VALUE="log_attribute" -->
<!-- /WSLET -->
```

Use 49 required parameters

None.

Use 49 optional parameters

RTF_FORMAT Returns the attribute value in RTF format instead of HTML.

TRANSITION_FONT Specifies font attributes for the transition section header. All HTML <FONT> element attributes are valid. VALUE="`font_options`".

MODIFY_FONT Specifies font attributes for the modification section header. All HTML <FONT> element attributes are valid. VALUE="`font_options`".

NOTE_FONT Specifies font attributes for the notes section header. All HTML <FONT> element attributes are valid. VALUE="`font_options`".

SHOW_TRANSITION_COMMENTS Indicates whether to show transition comments. VALUE=[`TRUE` | `FALSE`].

SHOW_MODIFY_EVENTS Indicates whether to show modify events. VALUE=[`TRUE` | `FALSE`].

SHOW_NOTES Indicates whether to show notes. VALUE=[`TRUE` | `FALSE`].

Use 49 example

Show the `transition_log` attribute.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_LOG VALUE=transition_log -->
<!-- PARAM NAME=TRANSITION_FONT VALUE="COLOR=purple" -->
<!-- PARAM NAME=MODIFY_FONT VALUE="COLOR=red" -->
<!-- PARAM NAME=NOTE_FONT VALUE="COLOR=blue" -->
<!-- PARAM NAME=SHOW_TRANSITION_COMMENTS VALUE=true -->
<!-- PARAM NAME=SHOW_MODIFY_EVENTS VALUE=true -->
<!-- PARAM NAME=SHOW_NOTES VALUE=true -->
<!-- /WSLET -->
```

Output:

```
<FONT COLOR=purple>Submitted by jane on Friday, May 11, 2001
8:43:33 AM</FONT>
<BR><BR>
<FONT COLOR=purple>Transitioned to in_review by jane on Friday,
May 11, 2001 8:44:43 AM</FONT>
<BR>verify what?<BR>----------<BR>priority: &lt;empty&gt;  --
&gt;  3<BR><BR>
<FONT COLOR=purple>Transitioned to assigned by jane on Friday,
May 11, 2001 9:00:49 AM</FONT>
<BR>afaf<BR>----------<BR>release: &lt;empty&gt;  --&gt;  alpha
<BR>priority: 3  --&gt;  4<BR>resolver: &lt;empty&gt;  --&gt;
jane<BR><BR>
<FONT COLOR=blue>Note added by jane on Thursday, May 24, 2001
10:21:18 AM</FONT>
<BR>just adding a note for all to see<BR>
```

Use 50

REPORT_MAX

Calculate the maximum of the specified attribute within a single report object.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MAX VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 50 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 50 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 51

REPORT_MAX

Calculate the maximum of the specified attribute across report objects.

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MAX VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 51 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int|float|date`].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 51 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 52

REPORT_MAX

Calculate the maximum of the specified attributes across report objects. The maximum of each row is computed, then the maximum across rows is determined.

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MAX VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
```

```
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 52 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 52 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 53

REPORT_MIN

Calculate the minimum of the specified attribute within a single report object.

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MIN VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 53 required parameters

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

Use 53 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 54

REPORT_MIN

Calculate the minimum of the specified attribute across report objects.

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MIN VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 54 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 54 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 55

REPORT_MIN

Calculate the minimum of the specified attributes across report objects. The minimum of each row is computed, then the minimum across rows is determined.

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_MIN VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 55 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 55 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 56

REPORT_QUERYNAME

Get the name of the report's query.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_QUERYNAME -->
<!-- /WSLET -->
```

Use 56 required parameters

None.

Use 56 optional parameters

None.

Use 56 example

Get the name of the report's query.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_QUERYNAME -->
<!-- /WSLET -->
```

Output:

```
DRP - Submitted by Me
```

Use 57

REPORT_QUERYSTRING

Get the report's query string.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_QUERYSTRING -->
<!-- /WSLET -->
```

Use 57 required parameters

None.

Use 57 optional parameters

None.

Use 57 example

Get the report's query string.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_QUERYSTRING -->
<!-- /WSLET -->
```

Output:

```
cvtype='problem' and submitter='%username'
```

Use 58

REPORT_REPORTNAME

Get the report's name.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_REPORTNAME -->
<!-- /WSLET -->
```

Use 58 required parameters

None.

Use 58 optional parameters

None.

Use 58 example

Get the report's name.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_REPORTNAME -->
<!-- /WSLET -->
```

Output:

```
DRP - Summary of CRs Submitted by Me
```

Use 59

REPORT_SPAN_ATTR_NAME

Get the label of a span attribute.

Template scope: span attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SPAN_ATTR_NAME -->
<!-- /WSLET -->
```

Use 59 required parameters

None.

Use 59 optional parameter

REPORT_NULL_VALUE Specifies a string to be used if the attribute label is the empty string. VALUE="*string*".

Use 60

REPORT_SPAN_ATTR_VALUE

Get the value of a span attribute. This option has a rich set of parameters to affect the attribute presentation.

Template scope: span attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SPAN_ATTR_VALUE -->
<!-- /WSLET -->
```

Use 60 required parameters

None.

Use 60 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_NULL_VALUE Specifies a string to be used if the attribute label is the empty string. VALUE="*string*".

REPORT_ENCODING Performs the substitutions defined by the encoding group. VALUE=*encoding_group*. The default encoding group is "HTML".

REPORT_PRE Surrounds the attribute value in <PRE></PRE> tags, which preserves the line spacing.

or

REPORT_TEXT_AREA Puts the attribute value inside an HTML text area control. This is useful for multi-line attributes, because the text area is a fixed size and scrollable.

Optional parameters for REPORT_TEXT_AREA:

> **REPORT_ROWS** Specifies the number of rows in the text area. VALUE=*number_of_rows.* The default is 5.

REPORT_COLS Specifies the number of columns in the text area. VALUE=*number_of_columns*. The default is 60.

REPORT_WRAP Defines the wrapping behavior of the text area. VALUE=[virtual|physical]. The default is virtual.

AUTO_ATTR_REPORT_LOG Formats the value as a log attribute if the user is on the named attribute. VALUE="*attribute_name*".

Optional Parameters for AUTO_ATTR_REPORT_LOG:

RTF_FORMAT Returns the attribute value in RTF format instead of HTML.

TRANSITION_FONT Specifies font attributes for the transition section header.All HTML <FONT> element attributes are valid. VALUE="*font_options*".

MODIFY_FONT Specifies font attributes for the modification section header. All HTML <FONT> element attributes are valid. VALUE="*font_options*".

NOTE_FONT Specifies font attributes for the notes section header. All HTML <FONT> element attributes are valid. VALUE="*font_options*".

SHOW_TRANSITION_COMMENTS Indicates whether to show transition comments. VALUE=[TRUE|FALSE].

SHOW_MODIFY_EVENTS Indicates whether to show modify events. VALUE=[TRUE|FALSE].

SHOW_NOTES Indicates whether to show notes. VALUE=[TRUE|FALSE].

Use 60 example

Get the value of a span attribute.

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SPAN_ATTR_VALUE -->
<!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->
<!-- PARAM NAME=AUTO_ATTR_REPORT_LOG VALUE=transition_log -->
<!-- PARAM NAME=TRANSITION_FONT VALUE="COLOR=purple" -->
<!-- PARAM NAME=MODIFY_FONT VALUE="COLOR=red" -->
<!-- PARAM NAME=NOTE_FONT VALUE="COLOR=blue" -->
<!-- PARAM NAME=SHOW_TRANSITION_COMMENTS VALUE=true -->
<!-- PARAM NAME=SHOW_MODIFY_EVENTS VALUE=true -->
<!-- PARAM NAME=SHOW_NOTES VALUE=true -->
<!-- /WSLET -->
```

Output:

The result depends on the attribute (and value). The `transition_log` attribute will be formatted in log format, with color section headings. Attributes with null values will return "`&nbsp;`". Other values will have HTML encoding performed, if necessary (default).

Use 61

REPORT_SUB_ATTR_VALUE_COUNT

Count the number of associated objects across report objects that have the specified attribute value.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_ATTR_VALUE_COUNT
VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_ATTR_VALUE VALUE="attribute_value" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 61 required parameters

REPORT_ATTR_VALUE Specifies the attribute value. VALUE="*attribute_value*".

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 61 optional parameter

REPORT_METRIC_LABEL Label the count using the form "*label*:*count*". VALUE="*label*".

Use 62

REPORT_SUB_ATTR_VALUE_COUNT

Count the number of associated objects across report objects that have the specified date attribute value.

Template scope: main, header, footer, and group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_ATTR_VALUE_COUNT
VALUE="date_attribute" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 62 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 62 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE="*attribute_value*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_METRIC_LABEL Label the count using the form "*label*:*count*". VALUE="*label*".

Use 63

REPORT_SUB_AVG

Calculate the average of the specified attribute across the associated objects of the report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   for each associated object of the current report object
      result += attribute
result = result / (# of non-null, non-empty associated
objects)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_AVG VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 63 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int|float`].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 63 optional parameters

None.

Use 64

REPORT_SUB_AVG

Calculate the average of the specified attributes across the associated objects of the report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   for each associated object of the current report object
      result += (attribute_name1 + attribute_name2 + ... +
attribute_n) / n
result = result / (# of non-null, non-empty associated
objects)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 64 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 64 optional parameters

None.

Use 65

REPORT_SUB_AVG

Calculate the average of the differences between the specified date attribute and the *from_date* across the associated objects of the report objects.

```
for each report object
   for each associated object of the current report
object
      result += attribute_name - from_date
result = result / (# of non-null, non-empty associated
objects)
```

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_AVG VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
```

```
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 65 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*`date_units`*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 65 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE=`"`*`attribute_value`*`"`. The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*`date_format`*".

Use 66

REPORT_SUB_AVG

Calculate the average of the differences between the specified date attributes and the *`from_date`* across the associated objects of the report objects.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   for each associated object of the current report object
      result += ((attribute_name1 - from_date) +
(attribute_name2 - from_date) +
      ... + (attribute_n - from_date)) / n
result = result / (#of non-null, non-empty associated
objects)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_AVG VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 66 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 66 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE="*attribute_value*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 67

REPORT_SUB_COUNT

Calculate the number of associated objects belonging to the report objects.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_COUNT -->
<!-- /WSLET -->
```

Use 67 required parameters

None.

Use 67 optional parameter

None.

Use 68

REPORT_SUB_DIFF_AVG

Calculate the average of the differences of two subreport attributes, across all subreport objects at the same level.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   for each immediate subreport object
      result += attribute_name1 - attribute_name2
result = result / (# of subreports with non-null, non-
empty values)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_AVG
VALUE=attribute_name1:attribute_name2 -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 68 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 68 optional parameters

None.

Use 69

REPORT_SUB_DIFF_AVG

Calculate the average of the differences of two subreport date attributes (or one date attribute and a specified date), across all subreport objects at the same level.

Note The query excludes null values; that is, only query items that have a non-null, non-empty attribute value are factored into the average.

```
for each report object
   for each immediate subreport object
      result += attribute_name1 - attribute_name2 (or
attribute_name -
      from_date)
result = result / (# of subreports with non-null, non-
empty values)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_AVG
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 69 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 69 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. In unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 70

REPORT_SUB_DIFF_MAX

Calculate the maximum difference of two subreport attributes, across all subreport objects at the same level. If the difference cannot be computed for some reason (for example, a value cannot be parsed as a numerical value), it is assumed to be 0.

```
for each report object
   for each immediate subreport object
      determine attribute_name1 - attribute_name2
result = max(attribute_name1 - attribute_name2)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_MAX
VALUE=attribute_name1:attribute_name2 -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 70 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 70 optional parameters

None.

Use 71

REPORT_SUB_DIFF_MAX

Calculate the maximum difference of two subreport date attributes (or one date attribute and a specified date), across all subreport objects at the same level.

```
for each report object
   for each immediate subreport object
      determine attribute_name1 - attribute_name2 (or
attribute_name -
      from_date)
result = max(attribute_name1 - attribute_name2) (or
max(attribute_name - from_date))
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_MAX
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 71 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 71 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. In unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 72

REPORT_SUB_DIFF_MIN

Calculate the minimum difference of two subreport attributes, across all subreport objects at the same level. If the difference cannot be computed for some reason (for example, a value cannot be parsed as a numerical value), it is assumed to be 0.

```
for each report object
   for each immediate subreport object
      determine attribute_name1 - attribute_name2
result = min(attribute_name1 - attribute_name2)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_MIN
VALUE=attribute_name1:attribute_name2 -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 72 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 72 optional parameters

None.

Use 73

REPORT_SUB_DIFF_MIN

Calculate the minimum difference of two subreport date attributes (or one date attribute and a specified date), across all subreport objects at the same level:

```
for each report object
   for each immediate subreport object
      determine attribute_name1 - attribute_name2 (or
attribute_name -
      from_date)
result = min(attribute_name1 - attribute_name2) (or
max(attribute_name - from_date))
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_MIN
VALUE=[attribute_name|attribute_name1:attribute_name2] -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 73 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 73 optional parameters

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. In unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 74

REPORT_SUB_DIFF_SUM

Calculate the sum of the differences between two subreport attributes, across all subreport objects at the same level.

```
for each report object
   for each immediate subreport object
result += attribute_name1 - attribute_name2
```

If the difference cannot be computed for some reason (for example, a value cannot be parsed as a numerical value), it is assumed to be 0.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_SUM
VALUE="attribute_name1":attribute_name2 -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 74 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 74 optional parameters

None.

Use 75

REPORT_SUB_DIFF_SUM

Calculate the sum of the differences between two subreport date attributes (or one date attribute and a specified date), across all subreport objects at the same level.

```
for each report object
   for each immediate subreport object
      result += attribute_name1 - attribute_name2 (or
attribute_name -
      from_date)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_DIFF_SUM
VALUE=[attribute_name|attribute_name1:attribute_name2 -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 75 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 75 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. This parameter is only used if a single attribute (rather than an attribute pair) is specified. VALUE="*from_date*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the FROM_DATE parameter. In unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 76

REPORT_SUB_MAX

Calculate the maximum of the specified attribute across the associated objects of the report objects.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_MAX VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 76 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 76 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 77

REPORT_SUB_MAX

Calculate the maximum of the specified attributes across the associated objects of the report objects.

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_MAX VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 77 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 77 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 78

REPORT_SUB_MIN

Calculate the minimum of the specified attribute across the associated objects of the report objects.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_MIN VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 78 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 78 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the REPORTS_DATE_FORMAT configuration entry is used. VALUE="*date_format*".

Use 79

REPORT_SUB_MIN

Calculate the minimum of the specified attributes across the associated objects of the report objects.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_MIN VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 79 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float|date].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 79 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the

REPORTS_DATE_FORMAT configuration entry is used.
VALUE="*date_format*".

Use 80

REPORT_SUB_SUM

Calculate the sum of the specified attribute across the associated objects of the report objects.

```
for each report object
   for each associated object of the current report
object
      result += attribute_name
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_SUM VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 80 required parameters

REPORT_DATATYPE Specifies the type of data to return.
VALUE="*datatype*"=[int|float].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 80 optional parameters

None.

Use 81

REPORT_SUB_SUM

Calculate the sum of the specified attributes across the associated objects of the report objects:

```
for each report object
   for each associated object of the current report
object
```

```
        result += attribute_name1 + attribute_name2 + ... +
attribute_n
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 81 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 81 optional parameters

None.

Use 82

REPORT_SUB_SUM

Calculate the sum of the differences between the specified date attribute and the *from_date* across the associated objects of the report objects.

```
for each report object
   for each associated object of the current report object
      result += attribute_name - from_date
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_SUM VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 82 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 82 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE="*attribute_value*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 83

REPORT_SUB_SUM

Calculate the sum of the differences between the specified date attributes and the *from_date* across the associated objects of the report object:

```
for each report object
   for each associated object of the current report
object
      result += (attribute_name1 - from_date) +
(attribute_name2 -
      from_date) + ... + (attribute_n - from_date)
```

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUB_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 83 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 83 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE="*attribute_value*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 84

REPORT_SUM

Calculate the sum of the specified attributes within a single report object.

```
sum(attribute_i)
```

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 84 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[`int`|`float`].

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 84 optional parameters

None.

Use 85

REPORT_SUM

Calculate the sum of the specified attribute across report objects.

```
for each report object
   result += attribute
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 85 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 85 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 86

REPORT_SUM

Calculate the sum of the specified attributes within a report object, across report objects. That is, get a subtotal for each row, then add all the rows together.

```
for each report object
   result += attribute_name1 + attribute_name2 + ... +
attribute_n
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE="datatype" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 86 required parameters

REPORT_DATATYPE Specifies the type of data to return. VALUE="*datatype*"=[int|float].

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 86 optional parameter

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 87

REPORT_SUM

Calculate the sum of the differences between the specified date attributes and the *from_date*, within a single report object.

```
sum(attribute_i - from_date)
```

Template scope: attribute

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ATTR_METRIC -->
<!-- /WSLET -->
```

Use 87 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ATTR_METRIC Creates a report based on the attributes of a single report object (row-based).

Use 87 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. If unspecified, the current date is used. VALUE="*from_date*".

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

Use 88

REPORT_SUM

Calculate the sum of the differences between the specified date attribute and the *from_date*, across report objects.

```
for each report object
   result += attribute _name- from_date
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_name" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_COL_METRIC -->
<!-- /WSLET -->
```

Use 88 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_COL_METRIC Creates a report based on a single attribute across report objects (column-based).

Use 88 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limit the count to items within the current group. This parameter should be used inside the group template.

Use 89

REPORT_SUM

Calculate the sum of the differences between the specified date attributes and the *from_date* within a report object, across report objects; that is, get a subtotal of the sum of the differences for each row, then add all the rows together:

```
for each report object
   result += (attribute_name1 - from_date) +
(attribute_name2 - from_date)
   + ... + (attribute_n - from_date)
```

Template scope: main, header, footer, group

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_SUM VALUE="attribute_list" -->
<!-- PARAM NAME=REPORT_DATATYPE VALUE=date -->
<!-- PARAM NAME=REPORT_DATE_UNITS VALUE="date_units" -->
<!-- PARAM NAME=REPORT_ROW_METRIC -->
<!-- /WSLET -->
```

Use 89 required parameters

REPORT_DATATYPE=date Specifies the type of data to return. VALUE=`date`.

REPORT_DATE_UNITS Specifies the date units to return. VALUE="*date_units*"=`[years|quarters|months|weeks|days|hours|minutes|seconds]`

REPORT_ROW_METRIC Creates a report based on the attributes of each report object, across all report objects (over each row, across all rows).

Use 89 optional parameters

REPORT_FROM_DATE Specifies the value of the date attribute. VALUE="*from_date*". The default is today's date.

REPORT_FROM_DATE_FORMAT Specifies the date format of the from date parameter. If unspecified, SUBMIT_DATE_FORMAT is used. VALUE="*date_format*".

REPORT_GRP_METRIC Limits the metric to items within the current group. This parameter should be used inside the group template.

Use 90

REPORT_TITLE

Get the title of the report.

Template scope: main, header, footer

Wslet call:

```
<!-- WSLET CODE=PTReport -->
<!-- PARAM NAME=REPORT_TITLE -->
<!-- /WSLET -->
```

Output:

`report_title_string`

Use 90 required parameters

None.

Use 90 optional parameters

None.

Examples

- Build a column-style report using the auto generate features (REPORT_AUTO_GENERATE_LABELS and REPORT_AUTO_GENERATE) of PTReport. With auto generate, attributes need not be known in advance, and the report templates are independent of the attributes that are reported on.

  Configuration file entries:

  ```
      [CCM_REPORT]
          [NAME]Column Format[/NAME]
          [RPT_TEMPLATE]ColumnCRReport[/RPT_TEMPLATE]
          [QUERY]All CRs[/QUERY]
          [PROBLEM_DEF]column_cr[/PROBLEM_DEF]
          [EXPORT_FORMAT]HTML[/EXPORT_FORMAT]
          [DESCRIPTION]Custom report format...[/DESCRIPTION]
      [/CCM_REPORT]

      [CCM_PROBLEM]
          [NAME]column_cr[/NAME]
          [MAIN_TEMPLATE]user_framework/column_rpt.html[/
  MAIN_TEMPLATE]
  ```

```
       [HDR_TEMPLATE]user_framework/common_hdr.html[/
HDR_TEMPLATE]
       [ATTR_TEMPLATE]user_framework/column_attr.html[/
ATTR_TEMPLATE]
       [LABEL_TEMPLATE]user_framework/custom_label.html[/
LABEL_TEMPLATE]
       [AUTO_LABEL_TEMPLATE]user_framework/
custom_auto_label.html
       [/AUTO_LABEL_TEMPLATE]
       [SPAN_ATTR_TEMPLATE]user_framework/custom_span_attr.html
       [/SPAN_ATTR_TEMPLATE]
       [AUTO_ATTR_TEMPLATE]user_framework/column_auto_attr.html
       [/AUTO_ATTR_TEMPLATE]
       [FTR_TEMPLATE]user_framework/common_ftr.html[/
FTR_TEMPLATE]
       [ATTRS]problem_number:0:false|crstatus:1:false|
       problem_synopsis:2:false[/ATTRS]
       [SORT_ORDER]problem_number:intb:A[/SORT_ORDER]
    [/CCM_PROBLEM]
```

Template files:

`user_framework/custom_label.html`:

```
<TR>
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_AUTO_GENERATE_LABELS -->
    <!-- /WSLET -->
</TR>
```

`user_framework/column_auto_label.html`:

```
<TD ALIGN=left VALIGN=center BGCOLOR=lightgrey>
<B>
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_AUTO_ATTR_NAME -->
    <!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->
    <!-- /WSLET -->
</B>
</TD>
```

`user_framework/column_attr.html`:

```
<TR>
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_AUTO_GENERATE -->
    <!-- /WSLET -->
</TR>
```

`user_framework/column_auto_attr.html`:

```
<TD ALIGN=left VALIGN=center>
   <!-- WSLET CODE=PTReport -->
   <!-- PARAM NAME=REPORT_AUTO_ATTR_VALUE -->
   <!-- PARAM NAME=REPORT_NULL_VALUE VALUE="&nbsp;" -->
   <!-- /WSLET -->
</TD>
```

Suppose the main template contains the following PTReport wslet calls:

```
<TABLE CELLSPACING=0 CELLPADDING=2 BORDER=1>
   <!-- WSLET CODE=PTReport -->
   <!-- PARAM NAME=REPORT_LABELS -->
   <!-- /WSLET -->

   <!-- WSLET CODE=PTReport -->
   <!-- PARAM NAME=REPORT_ATTRS -->
   <!-- /WSLET -->
</TABLE>
```

The resulting HTML will be:

```
<TABLE CELLSPACING=0 CELLPADDING=2 BORDER=1>
<TR>
   <TD ALIGN=left VALIGN=center BGCOLOR=lightgrey><B>CR ID</
B>
   </TD>
   <TD ALIGN=left VALIGN=center BGCOLOR=lightgrey><B>Status
   </B></TD>
   <TD ALIGN=left VALIGN=center
   BGCOLOR=lightgrey><B>Synopsis
   </B></TD>
</TR>

<TR>
   <TD ALIGN=left VALIGN=center>1</TD>
   <TD ALIGN=left VALIGN=center>assigned</TD>
   <TD ALIGN=left VALIGN=center>First in the db</TD>
</TR>
<TR>
   <TD ALIGN=left VALIGN=center>2</TD>
   <TD ALIGN=left VALIGN=center>assigned</TD>
   <TD ALIGN=left VALIGN=center>go figure</TD>
</TR>
<TR>
   <TD ALIGN=left VALIGN=center>3</TD>
   <TD ALIGN=left VALIGN=center>assigned</TD>
   <TD ALIGN=left VALIGN=center>durgie</TD>
```

```
    </TR>
</TABLE>
```

- Get a count of the number of CRs with the specified status value. This is achieved by grouping on the crstatus attribute.

  Configuration file entries:

```
    [CCM_PROBLEM]
        .
        .
        .
        [GROUP_TEMPLATE]user_framework/my_grp.html[/
GROUP_TEMPLATE]
        [ATTRS]problem_number|crstatus|problem_synopsis[/ATTRS]
        [GROUP_BY]crstatus[/GROUP_BY]
        [SORT_ORDER]problem_number:intb:A[/SORT_ORDER]
    [/CCM_PROBLEM]
```

  Template file:

  `user_framework/my_grp.html`

  Wslet call:

```
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_GROUP_COUNT -->
    <!-- /WSLET -->
    CRs have the following value:
    <!-- WSLET CODE=PTReport -->
    <!-- PARAM NAME=REPORT_GROUP_BY -->
    <!-- /WSLET -->"
```

  Output:

```
    3 CRs have the value "assigned"
```

PTString

Retrieves resource strings, attribute aliases, role-based text, and supplemental data sent in with a request. In the first two capacities, PTString replaces a CCM_ATTRIBUTE from a configuration file with an alias or with role-based text. If no alias or role-specific value is defined, the attribute name itself is returned.

The precedence of CCM_ATTRIBUTE values is as follows (from greatest to least):

- role-based
- ALIAS
- NAME

If both an alias and role-based text are present, the role-based text is used. If neither is present, the attribute name is used; that is, no substitution is performed.

Supplemental data allows user-defined parameters to be sent in with an HTTP request and retrieved on a form. Each piece of supplemental data is named with the prefix "SUPP_".

Use 1

CCM_STRING

Get the role-based text or alias for the specified attribute entry.

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=CCM_STRING VALUE=attribute_name -->
<!-- /WSLET -->
```

Use 1 required parameters

None.

Use 1 optional parameters

ENCODE Sets the value to be returned as a JavaScript string.

Use 1 examples

Suppose you have the following attribute entries:

```
[CCM_ATTRIBUTE]
    [NAME]Problem_Identifier[/NAME]
    [ALIAS]Change Request[/ALIAS]
    [User]CR[/User]
    [TYPE]CCM_STRING[/TYPE]
[/CCM_ATTRIBUTE]

[CCM_ATTRIBUTE]
    [NAME]customer[/NAME]
    [TYPE]CCM_LISTBOX[/TYPE]
[/CCM_ATTRIBUTE]
```

The following would occur for a user logged in with the *User* role.

Wslet call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=CCM_STRING VALUE=Problem_Identifier -->
<!-- /WSLET -->
```

Output:

```
CR
```

Wslet call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=CCM_STRING VALUE=customer -->
<!-- /WSLET -->
```

Output:

```
customer
```

The following would occur for a user logged in with the *Admin* role.

Wslet call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=CCM_STRING VALUE=Problem_Identifier -->
<!-- /WSLET -->
```

Output:

```
Change Request
```

Wslet call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=CCM_STRING VALUE=customer -->
<!-- /WSLET -->
```

Output:

```
customer
```

Use 2

SUPPLEMENTAL_DATA

Get the named piece of supplemental data, which should have been passed in as a request parameter. Note that only request parameters whose names begin with the prefix "SUPP_" are available.

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=SUPPLEMENTAL_DATA VALUE=SUPP_param_name -->
<!-- /WSLET -->
```

Use 2 required parameters

None.

Use 2 optional parameters

ENCODE Sets the value as a JavaScript variable. The name of the variable is SUPP_*param_name* unless JAVASCRIPT_IDENTIFIER is specified.

JAVASCRIPT_IDENTIFIER Specifies the name to use for the JavaScript variable. Only applicable when using the ENCODE option.

DFLT_VALUE Speficies a value to return if the requested piece of supplemental data did not exist.

ESCAPE Instructs the wslet to return the value in escaped format.

Use 2 examples

Suppose that SUPP_hi was passed as a request parameter and has the value "hello, world".

Wslet Call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=SUPPLEMENTAL_DATA VALUE=SUPP_hi -->
<!-- /WSLET -->
```

Output:

```
hello, world
```

Wslet Call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=SUPPLEMENTAL_DATA VALUE=SUPP_hi -->
<!-- PARAM NAME=ENCODE -->
<!-- /WSLET -->
```

Output:

var SUPP_hi = unescape("%68%65%6c%6c%6f%2c20%77%6f%72%6c%64");

Wslet Call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=SUPPLEMENTAL_DATA VALUE=SUPP_hi -->
<!-- PARAM NAME=ENCODE -->
<!-- PARAM NAME=JAVASCRIPT_IDENTIFIER VALUE=x -->
<!-- PARAM NAME=DFLT_VALUE VALUE=null -->
<!-- /WSLET -->
```

Output:

var x = unescape("%68%65%6c%6c%6f%2c20%77%6f%72%6c%64");

Wslet Call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=SUPPLEMENTAL_DATA VALUE=SUPP_hi -->
<!-- PARAM NAME=ESCAPE -->
<!-- /WSLET -->
```

Output:

%68%65%6c%6c%6f%2c20%77%6f%72%6c%64

Use 3

RESOURCE_STRING

Allows a named resource string bundle to be used when retrieving a resource string. If BUNDLE is specified, then string will be read from a resource bundle with that name from the `wsconfig/messages` directory. The correct bundle is automatically selected based on the locale.

Wslet Call:

```
<!-- WSLET CODE=PTString -->
<!-- PARAM NAME=RESOURCE_STRING VALUE=productName -->
<!-- /WSLET -->
```

Output:

```
Rational Change
```

Use 3 required parameters

None.

Use 3 optional parameters

REPORT_ENCODING The name of an encoding group (for example, HTML), as defined in a config file.

DEFAULT The value to use in the event no value was found for the indicated key.

BUNDLE The name of the resource bundle to look for in the string. Do not specify locale or end with ".properties"; only the base name. The proper locale and file name are determined automatically. If not specified, the default file is used.

PTText

PTText fetches an attribute value.

PTText is primarily used to preset HTML control elements, making it unnecessary to initialize them with the `OnLoad()` function.

Attributes whose values are available on a form are called *context data*; that is, within the context of a particular form, the attributes' data is available. Context data is made available through the CCM_ONLOAD action of the PTLoadData wslet.

Use 1

CCM_LOG_FORMAT

Get the formatted value of a log attribute.

Wslet call:

```
<!-- WSLET CODE=PTText -->
<!-- PARAM NAME=CCM_TEXT VALUE="attribute_name" -->
<!-- PARAM NAME=CCM_LOG_FORMAT -->
<!-- /WSLET -->
```

Use 1 required parameters

None.

Use 1 optional parameters

RTF_FORMAT Return the attribute value in RTF format instead of HTML.

TRANSITION_FONT Specifies font attributes for the transition section header. All HTML `<FONT>` element attributes are legal. VALUE="*font_options*".

MODIFY_FONT Specifies font attributes for the modification section header. All HTML `<FONT>` element attributes are legal. VALUE="*font_options*".

NOTE_FONT Specifies font attributes for the notes section header. All HTML `<FONT>` element attributes are legal. VALUE="*font_options*".

SHOW_TRANSITION_COMMENTS Indicates whether or not to show transition comments. VALUE=[TRUE|FALSE].

SHOW_MODIFY_EVENTS Indicates whether or not to show modify events. VALUE=[TRUE|FALSE].

SHOW_NOTES Indicate whether or not to show notes. VALUE=[TRUE|FALSE].

Use 1 example

Fetch the `transition_log` value.

Wslet call:

```
<!-- WSLET CODE=PTText -->
<!-- PARAM NAME=CCM_TEXT VALUE=transition_log -->
<!-- PARAM NAME=CCM_LOG_FORMAT -->
<!-- PARAM NAME=TRANSITION_FONT VALUE="COLOR=green" -->
<!-- PARAM NAME=MODIFY_FONT VALUE="COLOR=yellow" -->
<!-- PARAM NAME=NOTE_FONT VALUE="COLOR=pink" -->
<!-- PARAM NAME=SHOW_TRANSITION_COMMENTS VALUE="true" -->
<!-- PARAM NAME=SHOW_MODIFY_EVENTS VALUE="true" -->
<!-- PARAM NAME=SHOW_NOTES VALUE="true" -->
<!-- /WSLET -->
```

Output:

```
Submitted by jane on Wednesday, May 02, 2001 11:30:01 AM (in
green)
Note added by jane on Wednesday, May 02, 2001 11:30:25 AM (in
pink) adding a note
Transitioned to assigned by jane on Wednesday, May 02, 2001
11:31:07 AM (in green)----------
release: <empty> --> alpha
priority: <empty> --> 2
release_priority: <empty> --> 2
res
Modifications made by jane on Wednesday, May 02, 2001
11:31:21 AM (in yellow)
pgroup: <empty> --> usability
priority: 2 --> 4
Transitioned to resolved by jane on Wednesday, May 02, 2001
11:31:37 AM (in green)
Transitioned to concluded by jane on Wednesday, May 02, 2001
11:31:51 AM (in green)
```

Use 2

HTML_ENCODE

Retrieves the value of the named attribute, and is encoded if indicated.

Wslet call:

```
<!-- WSLET CODE=PTText -->
<!-- PARAM NAME=CCM_LISTBOX VALUE=severity -->
<!-- PARAM NAME=HTML_ENCODE -->
<!-- /WSLET -->
```

Output:

Showstopper

Use 2 required parameters

None.

Use 2 optional parameters

None.

Use 3

web_type

Put the value of an attribute into a JavaScript variable.

Possible values for *`web_type`* are as follows:

- **CCM_BUTTON**

  Puts attribute data on the form as a <SELECT></SELECT>.
- **CCM_DATE**

  Puts attribute data on the form as a <INPUT></INPUT> of type TEXT.
- **CCM_LISTBOX**

  Puts attribute data on the form as a <SELECT></SELECT>.
- **CCM_STRING**

  Puts attribute data on the form as <INPUT></INPUT> of type TEXT.
- **CCM_TEXT**

  Puts attribute data on the form as a <TEXTAREA></TEXTAREA>.
- **CCM_TOGGLE**

  Puts attribute data on the form as a <INPUT></INPUT> of type CHECKBOX.
- **CCM_VALUELISTBOX**

  Puts attribute data on the form as a <SELECT></SELECT>.

Wslet call:

```
<!-- WSLET CODE=PTText -->
<!-- PARAM NAME=web_type VALUE="attribute_name" -->
<!-- PARAM NAME=ENCODE -->
<!-- /WSLET -->
```

Use 3 required parameter

ENCODE Sets a JavaScript variable to the value.

Use 3 optional parameter

REPORT_DATE_FORMAT Indicates how a date attribute should be formatted (only applicable to CCM_DATE attributes). If not provided, the

SHOW_DATE_FORMAT configuration entry is used.
VALUE="*date_format*".

HTML_ENCODE Indicates if HTML meta-characters (e.g., < and >) should be encoded as entity references. This parameter does not take a value.

Use 3 examples

- Fetch the `problem_synopsis` value.

  Wslet call:

  ```
  <!-- WSLET CODE=PTText -->
  <!-- PARAM NAME=CCM_STRING VALUE=problem_synopsis -->
  <!-- PARAM NAME=ENCODE -->
  <!-- /WSLET -->
  ```

  Output:

  ```
  var problem_synopsis = "text";
  ```

- Fetch the `request_type` value.

  Wslet call:

  ```
  <!-- WSLET CODE=PTText -->
  <!-- PARAM NAME=CCM_LISTBOX VALUE=request_type -->
  <!-- PARAM NAME=ENCODE -->
  <!-- /WSLET -->
  ```

  Output:

  ```
  var request_type = "text";
  ```

- Fetch the `resolver` value.

  Wslet call:

  ```
  <!-- WSLET CODE=PTText -->
  <!-- PARAM NAME=CCM_STRING VALUE=resolver -->
  <!-- /WSLET -->
  ```

  Output:

  ```
  jane
  ```

- Fetch the `est_completion_date` value.

  Wslet call:

  ```
  <!-- WSLET CODE=PTText -->
  <!-- PARAM NAME=CCM_DATE VALUE=est_completion_date -->
  <!-- PARAM NAME=REPORT_DATE_FORMAT VALUE="MM/dd/yyyy" -->
  <!-- /WSLET -->
  ```

  Output:

  ```
  09/23/2001
  ```

PTToggle

PTToggle sets HTML input elements of type `CHECKBOX`. Returns the string "`CHECKED`" if the attribute is true, and "" (empty string) otherwise.

Note Context data is only available after calling PTLoadData.

Use

Determine whether a toggle should be checked (if true) or not (if false) based on an attribute's value.

Wslet call:

```
<!-- WSLET CODE=PTToggle -->
<!-- PARAM NAME=CCM_TOGGLE VALUE="attribute_name" -->
<!-- /WSLET -->
```

Required parameters

None.

Optional parameters

None.

Example

Set a toggle if the attribute `approved_by_manager` is true.

Wslet call:

```
<!-- WSLET CODE=PTToggle -->
<!-- PARAM NAME=CCM_TOGGLE VALUE=approved_by_manager -->
<!-- /WSLET -->
```

Output:

```
CHECKED
```

PTValueListBox

PTValueListBox builds the select options for a list box that has both labels and values.

PTValueListBox only generates the option elements, not the select element that contains them. The named value list box must be defined as a CCM_VALUELISTBOX in a configuration file.

Use

Get a CCM_VALUELISTBOX entry as HTML option elements.

Wslet call:

```
<!-- WSLET CODE=PTValueListBox -->
<!-- PARAM NAME=CCM_VALUELISTBOX VALUE="valuelistbox_name" -->
<!-- /WSLET -->
```

Required parameters

None.

Optional parameters

None.

Example

Return option elements for the `chart_type` attribute.

Wslet call:

```
<!-- WSLET CODE=PTValueListBox -->
<!-- PARAM NAME=CCM_VALUELISTBOX VALUE=chart_type -->
<!-- /WSLET -->
```

Output:

```
<OPTION VALUE=line>Line</OPTION>
<OPTION VALUE=stacked_area>Stacked Area</OPTION>
<OPTION VALUE=stacked_bar>Stacked Bar</OPTION>
<OPTION VALUE=vertical_bar>Vertical Bar</OPTION>
```

RunBsfScript

Allows a user to run a BSF script on any page that allows WSLETS to be executed. On show forms, this WSLET has the ability to send CR data to the script for processing.

Use

After the script is run, a string value from the script is returned. It is the responsibility of the script writer to ensure that all data returned from the script is safe for HTML.

Wslet call:

```
<!-- WSLET CODE=RunBsfScript -->
<!-- PARAM NAME="SCRIPT_NAME" VALUE="test.js" -->
<!-- PARAM NAME="%problem_number" -->
<!-- PARAM NAME="%crstatus" -->
<!-- PARAM NAME="%notThere" -->
<!-- PARAM NAME="testArg1" VALUE="93472673" -->
<!-- PARAM NAME="testArg2" VALUE="my second argument" -->
<!-- PARAM NAME="%problem_description" -->
<!-- /WSLET -->
```

Output:

The output is whatever the script returns in the 'returnValue' variable.

Required parameters

SCRIPT_NAME Name of a script in the `wsconfig\scripts` directory.

Optional parameters

%attribute_name Allows for CR data to be passed to the script on show forms only.

argument Allows user to pass hard-coded arguments to the script.

WsletTimer BEGIN/END

WsletUpdate BEGIN/END measures how long wslet execution takes on the server, which is useful for performance testing and debugging. It does not return anything; it merely logs a timed operation in the application log. Timing is logged to `event.log`.

Use

To use, you surround one or more wslets with a BEGIN/END pairing of WsletTimers. The BEGIN wslet captures the start time for a named event. The END wslet computes how long it has been since the start wslet for the event, and logs it, provided a minimum threshold (if provided) was exceeded. The event name can be anything descriptive but should match for a given pair. If the enclosed wslets take longer than the given threshold, a message is logged.

Wslet call:

```
<!-- WSLET CODE=WsletTimer -->
<!-- PARAM NAME=BEGIN VALUE="various wslets" -->
<!-- /WSLET -->

    <!-- WSLET CODE=PTInclude -->
    <!-- PARAM NAME=EventUtils.js --><!-- /WSLET -->
    <!-- WSLET CODE=PTString -->
    <!-- PARAM NAME=RESOURCE_STRING VALUE='lbl00740' -->
    <!-- /WSLET -->

<!-- WSLET CODE=WsletTimer -->
<!-- PARAM NAME=END VALUE="various wslets" -->
<!-- PARAM NAME=THRESHOLD_MS VALUE="50" -->
<!-- /WSLET -->
```

Output:

```
various wslets took 60 ms for user ‘john’
```

Required parameters

BEGIN A descriptive name for the event being recorded.

END An event matching a previously declared BEGIN tag.

Optional parameter

THRESHOLD_MS An integer value specifying an upper limit on how long the operation can take until it is logged.

WsletUpdate TDS_DOWN

WsletUpdate TDS_DOWN shows if the directory server is available or not.

Use

This Javascript variable indicates if the directory server is available or not. If the server is down, true is shown; else false.

Wslet call:

```
<!-- WSLET CODE=WsletUpdate -->
<!-- PARAM NAME=TDS_DOWN -->
<!-- /WSLET -->
```

Output:

```
var tds_down = false;
```

Required parameters

None.

Optional parameters

None.

10 Wslets index

This index contains a complete list of wslets and the parameters that determine their uses.

Wslets and Parameters

Appendix: Notices

Copyright license

Trademarks

Index