Cisco Collaboration Server Dynamic Content Adapter

Version 2.01

Administration and Configuration Guide

Contains instructions on administering and configuring the
Dynamic Content Adapter (DCA) 2.01 on all supported platforms.

Table of Contents

About DCA Documentation

Welcome to the Administration and Configuration Guide for the Cisco Collaboration Server Dynamic Content Adapter (DCA), version 2.01. This guide contains instructions for administering and configuring the DCA 2.01 on all supported platforms.

Audience

This guide is intended for individuals who configure and administer the DCA. It assumes that the reader is generally familiar with how the DCA works, and has knowledge of:

- Web architecture and Web collaboration
- The platforms and operating systems on which the DCA runs
- Cisco Collaboration Server

Other DCA 2.01 Documentation

The following documentation is available for the DCA 2.01.

Document	Primary Audience	Description
DCA Product Overview Guide	All DCA users	High-level product information. Includes: • How the DCA works • DCA features list • Supported platforms • SPLIT content collaboration issues • Deployment map Available formats: HTML and PDF
DCA Installation and Integration Guide	DCA installers, Collaboration Server administrators, system integrators	Instructions for getting the DCA installed and running. Includes: • System requirements • Installation instructions (all supported platforms) • Licensing information • Instructions for integrating the DCA with

Document	Primary Audience	Description
		Collaboration Server Available formats: HTML and PDF
DCA Administration and Configuration Guide	System integrators, DCA and Collaboration Server administrators, Web architects	Instructions for post-installation DCA configuration, customization, and administration. Includes: • DCA administration information • DCA configuration and customization information • Collaboration Toolbar configuration Available formats: HTML and PDF
DCA Parser API Javadoc	DCA customization specialists	Specification for the DCA parser API. Available formats: HTML and PDF
Collaboration Toolbar Online Help	Collaboration agents	Instructions for Collaboration agents on using the DCA Collaboration Toolbar to share Web content. Available formats: HTML
DCA Reference Card	All DCA users	Hard-copy reference card included with the DCA CD. Includes quick start information on how to access DCA documentation, the DCA Admin Tool, and Cisco support services. Available formats: Hard copy only
Release Notes	All DCA users	The DCA Release Notes contain up-to-date information on known issues and workarounds and any special instructions not covered in this guide. Available formats: PDF

To Access DCA Documentation

You can access all DCA documentation from the DCA Getting Started Page. To access the Getting Started Page:

- Before installation: From the top level of the DCA CD, open `getstart.htm`.
- After installation: In a Web browser, enter `http://<DCAservername>/dca-doc`. Or, if you have local access to the DCA server, you can also access `getstart.htm` from the DCA root directory.

DCA documentation is also available for download from the technical publication section of Cisco's Web site, www.cisco.com.

DCA Document Conventions

DCA documentation uses the following conventions:

Note	Indicates information of particular interest or significance.
Caution	Indicates the possibility of an adverse condition, such as poor or improper performance, data loss, or a security risk.
<>	Indicates a variable. For example: <DCAservername> represents the name of your DCA server. When prefaced by "Press," a bracketed term represents a keystroke. For example, "Press <Enter>" means to press the Enter key.

About Administering and Configuring the DCA

This guide groups the tasks and concepts related to setting up and maintaining the Dynamic Content Adapter into two categories: administration and configuration. Generally speaking:

- *DCA Administration* involves monitoring the DCA to ensure that it is functioning properly, and to ascertain details on server activity such as active session information.
- *DCA Configuration* involves changing the DCA's out-of-box configuration to achieve a specific goal; for example, to configure the DCA to use SSL, or to modify the Collaboration Toolbar interface.

Both categories contain information that is appropriate to all readers. For example, both administrators and configuration specialists will need to know how to use the DCA Admin Tool. And all users should understand the Collaboration Toolbar and how it works.

Note also that it is quite possible for the same person to perform both administration and configuration duties for the DCA.

See Also

For related information, see:

About Administering the DCA

About Configuring the DCA

Part 1: Administering the DCA

About Administering the DCA

Administering the DCA involves monitoring the DCA to ensure that it is functioning properly, and to ascertain details on server activity such as active session information. In a practical sense, however, once it is functioning well with your Web site, the DCA should need little in the way of regular monitoring or administration.

DCA administration tasks may include:

Using the Admin Tool

The Admin Tool is a Web-based interface you use to administer and configure the DCA. The Admin Tool gives you remote access to DCA configuration files, logs, server statistics, and to current session and participant information.

Maintaining DCA Sessions

The Admin Tool allows you to view details on current DCA sessions and session participants. While rarely necessary, you can also use the Admin Tool to terminate a session or participant.

Monitoring DCA Server Activity

The DCA includes these mechanisms for monitoring your DCA server:

- Server Statistics: You can view statistics on the number and type of requests sent to and served from your DCA server.
- Log Files: The DCA includes system logs you can use to monitor DCA activity and to locate and debug errors.
- Remote Monitoring: The DCA includes an API that can be used to send events (for example, exceptions, requests served) to a third-party monitoring tool.

See Also

For related information, see:

About the DCA Admin Tool
About DCA Sessions

Section I. Managing DCA Sessions

About DCA Sessions

Each DCA session carries a unique ID that is based on its corresponding CCS session ID. Each DCA participant receives a participant cookie that identifies him or her to the session. Users must present the correct combination of a URL and DCA session ID, or URL and participant cookie, to access pages from the DCA.

DCA sessions require little in the way of regular administration. They initiate and terminate automatically in conjunction with their corresponding CCS session.

How a DCA Session Begins

In a typical Call Center configuration, DCA sessions are initiated as follows:

1. A caller on a Web site submits a request for assistance by clicking a link or submitting a form. The request is sent to a Collaboration Server agent, who responds to the request, initiating a CCS session.

2. The Collaboration Toolbar is loaded into a browser window for both the agent and the caller.

3. As the Toolbar loads, a DCA session is automatically established and a session ID created.

4. The agent and caller are each sent a DCA participant cookie which uniquely identifies and associates them with the session.

5. The Web page from which the caller requested assistance is loaded into the browser window containing the Collaboration Toolbar.

The agent and caller can now begin sharing content.

How Participants Exit a Session

There is no formal mechanism (such as a logout button) that participants use to exit a DCA session. From a practical standpoint, participants exit a DCA session by exiting their CCS session, or by closing their browser window. Behind the scenes however, the DCA will continue to regard them as DCA session participants until their DCA participant cookie expires.

Administrators can use the Admin Tool to remove a participant from a session. However, because the Admin Tool interface lists participants by ID (a randomly

generated number) there may be no way to know which participant you are removing from session. (The Admin Tool DOES indicate whether a participant is an agent or caller.)

How a DCA Session Ends

A DCA session can end in any of these ways:

CCS Session Ends: A DCA session automatically terminates when its corresponding CCS session ends. This is the most common way of ending a DCA session. Note that as long as the CCS session remains active, in most cases the DCA session will remain active as well.

Connection to CCS Fails: In the rare instance that the DCA's connection to CCS fails (for example, due to network outage) the DCA will terminate inactive sessions based on a configurable session timeout period. Inactive sessions (that is, those in which a URL has not been pushed for a prescribed amount of time) are terminated regardless of the number of participants they contain. The default DCA session timeout period is 30 minutes.

Sessions with Zero Participants: Although an unlikely scenario, the DCA will automatically terminate any session with zero participants.

Termination by Administrator: Administrators can use the Admin Tool to terminate sessions. However, because the Admin Tool interface lists sessions by ID (which, like participant IDs, are randomly generated) there may be no way to know which participant's sessions are being terminated.

Joining and Switching Between Sessions

Once a DCA session has been established, additional participants can join as follows:

Joining a Session in Progress

Additional participants can join a DCA session already in progress simply by joining its corresponding CCS session. There is no limit to the number of participants a session can include.

Switching Between Sessions

Multi-session Collaboration Server agents can participate in multiple DCA sessions, just as they do multiple CCS sessions. Agents switch between DCA sessions

automatically when they switch between their corresponding CCS sessions.

Note: Multi-session agents receive only one DCA participant cookie, regardless of the number of sessions they are engaged in.

See Also

For related information, see:

How to View Active Sessions
How to Manually Terminate a Session
How to View Session Participant Information
How to Manually Remove a Participant From a Session
How to Configure the Session Timeout Period
How to Configure the Participant Timeout Period

How to View Active Sessions

You can use the Admin Tool to view active DCA sessions. The Admin Tool allows you to:

- View a list of all active DCA sessions
- View details of a particular session

To View Active Sessions and Session Details

To view a list of all active DCA sessions, or details on a particular session:

6. In the Admin Tool, select Sessions. All current DCA sessions are listed, along with their Session ID, start time, and number of participants.
7. To view additional information on a particular session, click its Session ID. The Admin Tool displays the session's:
    - Creation Time: The time this session was initiated
    - Expiration Time: The time this session will expire, IF the connection to the Collaboration server has failed. It is calculated as: Last access time + the DCA session timeout value (the default for this is 30 minutes).
    - Last Access Time: The time of the most recent request made to the DCA during this session.
    - Number of Participants: The current number of session participants.
    - Cookies: Cookies accumulated during this session.
    - Script Properties: Session-specific properties visible to scripts in the DCA parser.

See Also

For related information, see:

About DCA Sessions
How to View Active Sessions
How to Manually Terminate a Session
How to View Session Participant Information
How to Manually Remove a Participant From a Session

How to Manually Terminate a Session

Under normal circumstances, it should not be necessary to terminate a DCA session manually. DCA sessions end automatically when their corresponding CCS session ends.

However, as necessary, you can also a terminate session manually using the Admin Tool. Note that when you terminate a session:

- Participants DO NOT receive notification that the session was terminated.
- Because the Admin Tool interface lists sessions by ID, there may be no way to know which agent's session you are terminating.
- All state information and pages related to the session are removed from the DCA and Copy Server.

To Manually Terminate a Session

To manually terminate a DCA session:

1. In the Admin Tool, select Sessions. All current DCA sessions are listed, along with their Session ID, start time, and number of participants.
2. Click the ID of the session you want to terminate.
3. Click Terminate Session.

See Also

For related information, see:

About DCA Sessions
How to View Active Sessions
How to Configure the Session Timeout Period

How to Configure the Session Timeout Period

A DCA session automatically terminates when its corresponding CCS session ends. This is the most common way of ending a DCA session. As long as the CCS session remains active, in most cases the DCA session will remain active as well.

In the rare instance that the DCA's connection to CCS fails (for example, due to network outage) the DCA will terminate inactive sessions based on a configurable session timeout period. Inactive sessions (that is, those in which a URL has not been pushed for a prescribed amount of time) are terminated regardless of the number of participants they contain.

The default timeout period for inactive sessions is 1800000 milliseconds (30 minutes). You can modify this period to be shorter or longer, as desired.

To Modify the Session Timeout Period

To modify the DCA session timeout:

1. In a text editor, open the uiadapter-props.xml file, located at: <DCArootdirectory>\uiserver\WEB-INF\properties.
2. For sessionTimeout, specify a value in milliseconds.
3. Save the file.
4. Restart the DCA.

See Also

For related information, see:

About DCA Sessions
How to Stop and Start the DCA

How to View Session Participant Information

You can use the Admin Tool to view DCA session participants. The Admin Tool allows you to:

- View a list of all participants engaged in active DCA sessions
- View details on any participant

To View Session Participants and Participants Details

To view a list of all active DCA session participants, or details on any participant:

1. In the Admin Tool, select Participants. All participants currently engaged in DCA sessions are listed. For each participant, the Admin Tool displays:
    - Participant ID: The participant's ID for this DCA session. This is a random number generated by the DCA server.
    - Expiration Time: The time this participant's DCA cookie will expire, calculated as: Last access time + the DCA participant timeout value (the default for this is 60 minutes).
    - Sessions: The ID(s) of 0 or more DCA session(s) in which the participant is engaged. No sessions indicates that while the participant's session has terminated, the participant's DCA cookie has not yet expired. Multiple sessions indicates a multi-session agent.
2. To view additional information on a particular participant, click Participant ID. The Admin Tool displays the participant's:
    - Expiration Time: The time this participant's DCA cookie will expire, calculated as: Last access time + the DCA participant timeout value (the default for this is 60 minutes).
    - User Agent: The user-agent header string for the browser type/version to which this participant is mapped (via browser mapping). If browser mapping is not in effect, displays Not Found.
    - Active Session: For multi-session agents engaged in multiple sessions, the DCA session in which they are active.
    - Sessions: The ID(s) of 0 or more DCA session(s) in which the participant is engaged. No sessions indicates that while the participant's session has terminated, the participant's DCA cookie has not yet expired. Multiple sessions indicates a multi-session agent.

- Properties: Participant properties that can be created and accessed through scripts in the DCA parser.

See Also

For related information, see:

About DCA Sessions

How to Manually Remove a Participant From a Session

How to Configure the Participant Timeout Period

How to Manually Remove a Participant From a Session

Under normal circumstances, it should not be necessary to remove a participant manually from a DCA session. Participants are removed automatically when a session terminates. Inactive participants are dropped from sessions based on a configurable timeout period. And of course CCS agents can hang up on callers through the Agent Desktop.

However, as necessary, you can remove participants from sessions manually using the Admin Tool. Note that when you remove a participant from a session:

- The participant DOES NOT receive notification.
- Because the Admin Tool interface lists participants by ID, there may be no way to know who it is you are removing from session.

To Manually Remove a Participant From a Session

To manually remove a participant from a session:

1. In the Admin Tool, select Participants. All users currently engaged in DCA sessions are displayed.
2. Click the ID of the participant you want to terminate.
3. Click Terminate Participant.

See Also

For related information, see:

About DCA Sessions
How to View Session Participant Information
How to Manually Remove a Participant From a Session
How to Configure the Participant Timeout Period

How to Configure the Participant Timeout Period

A DCA participant cookie will expire automatically after a period of inactivity in which no requests are sent to the DCA. When a user's participant cookie expires, he is automatically removed from his DCA session.

The default timeout period for participant cookies is 3600 seconds (60 minutes). You can modify this period to be shorter or longer, as desired.

To Modify the Participant Timeout Period

To modify the DCA participant timeout period:

1. In the Admin Tool, select Configuration.
2. Click ProxyProperties.
3. For ParticipantTimeout, specify a value in seconds.
    - Make sure that the interval you specify is equal to or greater than the DCA's session timeout interval.
    - A high number helps prevent participants whose PC clock time is fast from receiving cookies that expire prematurely.
4. Click Submit.

See Also

For related information, see:

About DCA Sessions
How to Use the Proxy Properties File

Section II. Monitoring the DCA

How to View DCA Server Statistics

You can use the Admin Tool to view statistics on the number and type of requests sent to and served from your DCA server.

To View DCA Server Statistics

To view server statistics:

In the Admin Tool, select Server Status. The Server Statistics page displays the information about your DCA server:

Statistic:	**Description:**
Total Hits	Number of requests made to the DCA server: This number: • Includes requests satisfied by the DCA cache or Copy Server. • In addition to pages, can include resources not parsed by the DCA (for example, files called through JavaScript). • Excludes Administration requests.
Total POSTs	Number of POST hits made against the DCA server. A POST submitted by a DCA session participant results in one GET to the DCA for each other session participant. For example, in a session with 3 participants, a POST submitted by participant A will cause one GET each by participants B and C.
Total GETs	Number of GET hits made against the DCA server.
Requests sent to actual server	Number of page requests sent from the DCA to the Web content server(s).
Requests served from response cache	Number of page requests served from the Output cache.
Requests sent to Copy Server	Number of page requests served from the Copy Server.
Exceptions	Number of exceptions recorded since the last time Server Statistics was reset. Includes both internal (DCA) and external (non-DCA) exceptions. Note: Because external [non-DCA] exceptions are included in this total (such as errors returned from Web content servers), this count may show exceptions occurring even when the DCA itself is functioning error-free.

To Reset Server Statistics

To reset all current server statistics to 0:

In the Admin Tool, with the statistics page displayed, click Reset Counters.

Note: Server statistics are also reset automatically any time you restart the DCA Server.

See Also

For related information, see:

How to Access the Admin Tool
About DCA Caching

About DCA Logs

The DCA includes two system logs you can use to monitor DCA activity and to locate and debug errors.

- The **Trace Log** can include information on system Trace events, system error events, and Web content accessed through the DCA. You can set the Trace Log's system Logging Level and Content Logging Levels as desired.
- The **Error Log** only includes information on system Error events. Its logging level is not modifiable.

Log File Name and Persistence Conventions

DCA log files use the following default naming and storage conventions. These conventions can be reconfigured as desired:

- **Default File Names:** The default base file name for the Error Log is `errorLog.log`. The default base file name for the Trace Log is `traceLog.log`.
- **Old Log Files:** By default, log files will store up to a maximum of 1000 entries. When that number is reached, a new log file is started and the older file is renamed using an iteration of the base file name (for example, `traceLog1.log`, `traceLog2.log`, etc.).

  By default, the DCA will store up to 10 of these older log files. When that number is reached, the oldest file is deleted automatically each time a new file is created.

- **Effects of Server Restarts:** By default, whenever the DCA is restarted, all existing log files are renamed with a time stamp (for example, `traceLog20020610135818.log)`. Log files renamed with a timestamp are NOT subject to automatic deletion, and will continue to accumulate. This feature can be turned off, if desired.

System Events in DCA Logs

Generically speaking, an event is a notification that a business behavior occurred (for example, a request was sent or a session was initiated). DCA logs can track these event types:

Event Type:	Description:	Associated Log Levels:
Error	An error is a message that indicates an application is likely to have done something wrong. Each error has an associated severity.	GLOBAL_EVENT EXTERNAL_EVENT STATIC_FIELD
Trace	A trace is a message that indicates why or how an application did something. A trace exists for debugging and diagnostic purposes. Trace messages result from instrumented code.	EXTERNAL_STATE INTERNAL_STATE CODE_BRANCH METHOD_CALL CODE_MARKER CLASS_DUMP LOCAL_DUMP

See Also

For related information, see:

How to View a Log File
How to Delete a Log File
How to Modify Log File Properties
About Logging Levels
How to Set the Trace Log Logging Levels

How to View a Log File

You can open and view the Error and Trace logs:

- In the Admin Tool
- In any text editor

To View Logs in the Admin Tool

To view a DCA log in the Admin Tool:

1. In the Admin Tool, select Logging > Select and View Logs. All log files are displayed.
2. Click the link for the log file you want to view. By default, Trace Log file names begin with "traceLog" (for example (`traceLog`) and Error Log files names begin with "errorLog" (for example, `errorLog`). Older log files will contain an iterative number or a timestamp in their name.

To View Logs in a Text Editor

If you have local network access to files on the DCA Server, you can also open DCA log files in any text editor. Log files are located at: `<DCArootdirectory>\webapp\WEB-INF\Cisco\logs`.

See Also

For related information, see:

How to Access the Admin Tool
About DCA Logs
How to Delete a Log File
How to Modify Log File Properties
About Logging Levels
How to Set the Trace Log Logging Levels

How to Delete a Log File

By default, the DCA will store up to 10 (each) Error and Trace log files. When this maximum number is reached, the oldest file is deleted automatically when a new file is created. As necessary, you can also delete old log files manually.

To Delete a Log File

To delete a log file:

1. In Admin Tool, select Logging > Select and View Logs.
2. Click the check box to the left of one or more files you want to delete.
3. Click Delete Checked Files.

See Also

For related information, see:

How to Access the Admin Tool
About DCA Logs
How to Modify Log File Properties

How to Modify Log File Properties

As desired, you can modify the default properties for Error and Trace log files. Editable properties include maximum size, default name, storage location, and the persistence level of old files.

To Modify Log File Properties

To modify log file properties:

1. In a text editor, open the DCALog.properties file. It is located at: <DCArootdirectory>\webapp\WEB-INF\Cisco\logs.
2. Edit any of the properties listed in the table below. To modify:
   - Error Log properties, edit the properties under the comment heading: `# errorFileAdapter setup`.
   - Trace Log properties, edit the properties under the comment heading: `# traceFileAdapter setup`.
3. Save the file.
4. Restart the DCA.

DCA Log Property Descriptions

While the `DCALog.properties` file contains many settings affecting log behavior, Cisco recommends that you limit editing of the file to those properties listed below.

Property:	**Description:**	**Default:**
dirName	Defines the directory in which the DCA stores log files. You can change the default to point to any local directory.	<DCArootdirectory>\webapp\WEB-INF\Cisco\logs
fileName	Defines the base name for Error and Trace log files.	errorLog.log and traceLog.log
maxNumberOfEntries	Defines the maximum number of entries that can be stored in the Trace or Error log. When this number is reached, a new log file is started and the old log file is renamed using an iteration of the base file name (for example: traceLog1.log	1000 **Note:** By default, this parameter does not appear in the file. You must manually add it

Property:	Description:	Default:
dirName	Defines the directory in which the DCA stores log files. You can change the default to point to any local directory.	<DCArootdirectory>\webapp\WEB-INF\Cisco\logs
	traceLog2.log, etc.). Bear in mind that a substantial increase to the log's maximum number of entries can result in significantly larger file sizes.	under errorFileAdapter setup or traceFileAdapter setup, as desired.
maxNumberOfFiles	Defines the maximum number of old log files the DCA will retain. When this number is reached, the oldest file is deleted automatically when a new log file is created.	10
renameFileAtInit	Determines whether existing log files are renamed with a timestamp (for example, traceLog_20020610135818.log) when the DCA server is restarted . **Note:** Log files renamed with a timestamp are NOT subject to automatic deletion and will continue to accumulate. Therefore, consider available disk space when setting this parameter.	Yes **Note:** By default, this parameter does not appear in the file. You must manually add it under errorFileAdapter setup or traceFileAdapter setup, as desired. Acceptable values are Yes and No.

See Also

For related information, see:

About DCA Logs

How to Stop and Start the DCA

About Logging Levels

For the Trace Log, you can specify both the:

- System Logging Level: The System Logging Level determines which system messages appear in the Trace Log.
- Content Logging Level: The Content Logging Level determines what, if any, Web content information appears in the Trace Log.

Note: The DCA's Error Log logging level is not modifiable.

What is the System Logging Level?

The System Logging Level determines which system messages appear in the Trace Log. The least verbose message level, GLOBAL_EVENT, reports only serious system errors. The most verbose level, LOCAL_DUMP, reports all system activity.

A more verbose System Logging Level setting provides you with more information, but also causes your log to fill up more quickly. Cisco recommends that the System Logging Level's verbosity be set no higher than INTERNAL_STATE (the default) except when you are debugging a problem.

Available System Logging Levels

You can set the Trace Log to any of these System Logging Levels:

Level:	**Reports:**
GLOBAL_EVENT	Serious system errors only.
EXTERNAL_EVENT	Problems that could affect multiple users.
STATIC_FIELD	Abnormal conditions that could potentially be errors.
EXTERNAL_STATE	Changes to external visible state.
INTERNAL_STATE (DCA default)	Common system events.
CODE_BRANCH	Troubleshooting information.
METHOD_CALL	Verbose troubleshooting information.

Level:	Reports:
CODE_MARKER	Code path markers.
CLASS_DUMP	Very verbose troubleshooting information.
LOCAL_DUMP	Maximum verbose troubleshooting information.

What is the Content Logging Level?

The Content Logging Level determines what, if any, Web content information appears in the Trace Log. Web content can include:

- HTTP response headers
- HTML source code
- DCA response headers
- DCA HTML code
- Cookie information

Caution: Logging Web content presents significant security considerations. Content logging can expose sensitive information such as credit card numbers, passwords, and so on. Consult your company's privacy statement and security advisor before enabling content logging.

In addition, Web content can dramatically increase the size of your log. Cisco recommends that you log web content only when troubleshooting a problem. At other times, the Content Logging Level should be set to NO_CONTENT_LOGGING (the default).

Available Content Logging Levels

The following table describes available Content Logging Levels for the Trace Log:

Level:	Reports:
NO CONTENT LOGGING (DCA default)	No web content information.
LOG ORIGINAL RESPONSE CONTENT	Original unmodified HTML
LOG MODIFIED RESPONSE	Parsed and reconstructed HTML

Level:	Reports:
NO CONTENT LOGGING (DCA default)	No web content information.
CONTENT	
LOG ORIGINAL RESPONSE HEADERS	Original unmodified headers
LOG MODIFIED RESPONSE HEADERS	Parsed and reconstructed headers
LOG ALL RESPONSE HEADERS	Original unmodified and parsed and reconstructed headers
LOG ENTIRE ORIGINAL RESPONSE	Original unmodified HTML and headers
LOG ENTIRE MODIFIED RESPONSE	Parsed and reconstructed HTML and headers
LOG ALL CONTENT	All available web content information.

See Also

For related information, see:

How to Access the Admin Tool

About DCA Logs

How to Set the Trace Log Logging Levels

How to Set the Trace Log Logging Levels

As desired, you can modify the amount of system information and Web content information recorded in the Trace Log.

To Set the System Logging Level

To set the Trace Log's System Logging Level:

1. In the Admin Tool, select Logging > Set Logging Level.
2. Click the System Logging Level you want to use.

Note: A more verbose System Logging Level setting provides you with more information, but also causes your log to fill up more quickly. Cisco recommends that the System Logging Level's verbosity be set no higher than INTERNAL_STATE (the default) except when you are debugging a problem.

To Set the Content Logging Level

To set the Trace Log's Content Logging Level:

1. In the Admin Tool , select Logging > Set Content Logging Level.
2. Click the Content Logging Level you want to use.

Note: Web content can dramatically increase the size of your log. Cisco recommends that you log web content only when troubleshooting a problem.

See Also

For related information, see:

How to Access the Admin Tool
About DCA Logs
About Logging Levels

About Remote Monitoring

The DCA 2.01 includes a Remote Monitoring API you can use to send events (for example, exceptions, session statistics, and so on) from the DCA to a third-party remote listening tool. Monitoring the DCA can allow you to detect errors before they result in poor performance for users.

You can monitor the DCA for any of these event types:

- Exceptions (errors)
- Server Statistics (hits, requests, etc. to the DCA)
- Periodic Checks (current sessions, whether server is running, etc.)
- Property Files Modified (instances of property file modification)

Multiple remote listeners can subscribe to different events and each listener can implement the actions to be performed when a particular event is occurred.

The Remote Monitor API

The Remote Monitor API uses the following Java class files, located at: `<DCArootdirectory>\webapp\WEB-INF\classes\com\cisco\ics\monitor`:

- `IMonitorListener`: Use this interface to implement remote monitoring of the DCA.
- `IMonitorHost`: Use this interface to interact with the host.
- `MonitoredEvent`: Superclass of all monitored events.

See Also

For related information, see:

How to Implement Remote Monitoring
How to Use the RemoteMonitor Properties File

How to Implement Remote Monitoring

The Remote Monitor API uses the following Java class files, located at: `<DCArootdirectory>\webapp\WEB-INF\classes\com\cisco\ics\monitor`:

- `IMonitorListener`: Use this interface to implement remote monitoring of the DCA.
- `IMonitorHost`: Use this interface to interact with the host.
- `MonitoredEvent`: Superclass of all monitored events

To Implement a Remote Listener

To implement a remote listener:

1. Create and compile your listener class, as desired referencing the Remote Monitor API described in the next section.

   A sample listener class file, `SampleMonitorListener.java`, is included with the DCA at `<DCArootdirectory>\webapp\WEB-INF\classes\com\cisco\ics\monitor`.

2. Place your files in the appropriate location on the DCA server -- `<DCArootdirectory>\webapp\WEB-INF\classes` (for uncompiled class files) or `<DCArootdirectory>\webapp\WEB-INF\lib` (if you have compiled your class files into a jar file).
3. Add your listener to the list of subscribed listeners in the `RemoteMonitor.properties` file.

com.cisco.ics.monitor

Interface IMonitorListener

public interface **IMonitorListener**

Interface implemented by listeners of monitored events.

Method Summary -- IMonitorListener	
void	**eventOccurred**(MonitoredEventevent) Called when a monitored event occurs.
void	**init**(IMonitorHost host) Called during startup of the DCA when the listener is loaded.
void	**shutdown**() Called when the server is being shut down, allowing the listener implementation to perform any necessary cleanup.

Method Detail -- IMonitorListener

eventOccurred

```
public void eventOccurred(MonitoredEvent event)
```

Called when a monitored event occurs. The provided event object contains more information about the event being reported.

Parameters: `event` - Monitored event generated by DCA

init

```
public void init(IMonitorHost host)throws MonitorListenerException
```

Called during startup of the DCA when the listener is loaded. The listener will have already been subscribed to all events at the point this method is called.

Parameters: `host` - Description of Parameter

Throws: MonitorListenerException - Description of Exception

Method Detail -- IMonitorListener
shutdown `public void` **`shutdown`**`()` Called when the server is being shut down, allowing the listener implementation to perform any necessary cleanup.

com.cisco.ics.monitor

Interface IMonitorHost

public interface **IMonitorHost**

Interface presented to monitor listeners for interaction with the host.

Field Summary -- IMonitorHost	
`static int`	**`ERROR`** Constant indicating an error-only logging level.
`static int`	**`INFORM`** Constant indicating a moderate level of logging information.
`static int`	**`VERBOSE`** Constant indicating a verbose logging level.
Method Summary -- IMonitorHost	
`void`	**`addNewListener`**`(IMonitorListener listener)` Adds a new monitor listener.
`java.io.File`	**`getConfigurationFileDirectory`**`()` Returns the directory in which any monitor listener configuration files may

Field Summary -- IMonitorHost	
	be found.
void	**logError**(java.lang.Throwable error, int severity) Logs a single error (with stack trace) in the host's log.
void	**logMessage**(java.lang.String message, int severity) Logs a single message in the host's log.
void	**removeListener**(IMonitorListener listener) Removes a listener from the host's listener collection.
void	**subscribe**(IMonitorListener listener, java.lang.Object eventType) Subscribes a listener to a single event type
void	**subscribe**(IMonitorListener listener, java.lang.Object eventType, EventFilterfilter) Subscribes a listener to a single event type with the given event filter
void	**subscribeAll**(IMonitorListener listener) Subscribes a listener to all event types
void	**subscribeAll**(IMonitorListener listener, EventFilter filter) Subscribes a listener to all event types with the given filter
void	**unsubscribe**(IMonitorListener listener, java.lang.Object eventType) Unsubscribes a listener from a single event.
void	**unsubscribeAll**(IMonitorListener listener) Unsubscribes a listener from all events, causing it to be removed.

Field Detail -- IMonitorHost
ERROR

Field Detail -- IMonitorHost

`public static final int` **`ERROR`**

Constant indicating an error-only logging level

INFORM

`public static final int` **`INFORM`**

Constant indicating a moderate level of logging information

VERBOSE

`public static final int` **`VERBOSE`**

Constant indicating a verbose logging level

Method Detail -- IMonitorHost

addNewListener

`public void` **`addNewListener`**`(IMonitorListener listener)`

Adds a new monitor listener. A single monitor listener will only be added once.

Parameters: `listener` - New listener being added

getConfigurationFileDirectory

`public java.io.File` **`getConfigurationFileDirectory`**`()`

Returns the directory in which any monitor listener configuration files may be found.

Returns: The configuration file directory

Method Detail -- IMonitorHost

logError

```
public void logError(java.lang.Throwable error, int severity)
```

Logs a single error (with stack trace) in the host's log. Depending on the host's log configuration, the message may or may not appear if it does not meet or exceed the verbosity threshold.

Parameters: `error` - Description of Parameter; `severity` - Description of Parameter

logMessage

```
public void logMessage(java.lang.String message, int severity)
```

Logs a single message in the host's log. Depending on the host's log configuration, the message may or may not appear if it does not meet or exceed the verbosity threshold.

Parameters: `message` - Description of Parameter; `severity` - Description of Parameter

removeListener

```
public void removeListener(IMonitorListener listener)
```

Removes a listener from the host's listener collection. Shutdown will be called on the listener during this method.

Parameters: `listener` - Listener to remove

subscribe

```
public void subscribe(IMonitorListener listener, java.lang.Object
eventType)
```

Subscribes a listener to a single event type.

Parameters: `listener` - Listener being subscribed; `eventType` - Event to which listener is being subscribed (from MonitoredEvent)

Method Detail -- IMonitorHost

subscribe

```
public void subscribe(IMonitorListener listener, java.lang.Object
eventType, EventFilter filter)
```

Subscribes a listener to a single event type with the given event filter.

Parameters: `listener` - Listener being subscribed; `eventType` - Event to which listener is being subscribed (from MonitoredEvent); `filter` - Description of Parameter

subscribeAll

```
public void subscribeAll(IMonitorListener listener)
```

Subscribes a listener to all event types.

Parameters: `listener` - Listener being subscribed

subscribeAll

```
public void subscribeAll(IMonitorListener listener, EventFilter filter)
```

Subscribes a listener to all event types with the given event filter.

Parameters: `listener` - Listener being subscribed; `filter` - Description of Parameter

unsubscribe

```
public void unsubscribe(IMonitorListener listener, java.lang.Object
eventType)
```

Unsubscribes a listener from a single event. May cause the listener to be removed if it is no longer subscribed to any events.

Parameters: `listener` - Listener being unsubscribed; `eventType` - Event to which listener is being subscribed (from MonitoredEvent)

Method Detail -- IMonitorHost
unsubscribeAll `public void` **`unsubscribeAll`**`(IMonitorListener listener)` Unsubscribes a listener from all events, causing it to be removed.r. **Parameters:** `listener` - Listener being unsubscribed

com.cisco.ics.monitor

Class MonitoredEvent

public abstract class **MonitoredEvent**

Superclass of all monitored events.

Field Summary -- MonitoredEvent	
`static java.lang.object`	**`CONFIGCHANGE_EVENT_TYPE`** Type of configuration change event.
`static java.lang.object`	**`EXCEPTION_EVENT_TYPE`** Type of exception event.
`static java.lang.object`	**`PERIODIC_EVENT_TYPE`** Type of periodic event.
Constructor Summary -- MonitoredEvent	
`MonitoredEvent``(java.lang.Object source, java.lang.String message)` Constructor for the MonitoredEvent object.	

Field Summary -- MonitoredEvent
MonitoredEvent(java.lang.Object source, java.lang.String message, boolean waitForDelivery) Constructor for the MonitoredEvent object.

Method Summary -- MonitoredEvent	
static java.lang.object	**getEventType()** Gets the type of this event.
static java.lang.string	**getMessage()** Gets the event message.

Field Detail -- MonitoredEvent
PERIODIC_EVENT_TYPE public static final java.lang.Object **PERIODIC_EVENT_TYPE** Type of periodic event
EXCEPTION_EVENT_TYPE public static final java.lang.Object **EXCEPTION_EVENT_TYPE** Type of exception event
CONFIGCHANGE_EVENT_TYPE public static final java.lang.Object CONFIGCHANGE_EVENT_TYPE Type of configuration change event

Constructor Detail -- MonitoredEvent

MonitoredEvent

```
public MonitoredEvent(java.lang.Object source, java.lang.String message)
```

Constructor for the MonitoredEvent object.

Parameters: `source` - Description of Parameter; `message` - Description of Parameter

MonitoredEvent

```
public MonitoredEvent(java.lang.Object source, java.lang.String message,
boolean waitForDelivery)
```

Parameters: `source` - Event source object; `waitfordelivery` - Set to true to wait to return until event is delivered to all listeners; `message` - Event message

Method Detail -- MonitoredEvent

getEventType

```
public java.lang.object getEventType()
```

Gets the type of this event.

Returns: The eventType value

getMessage

```
public java.lang.string getMessage()
```

Gets the event message.

Returns: The message value

See Also

For related information, see:

About Remote Monitoring
How to Use the RemoteMonitor Properties File

How to Use the RemoteMonitor Properties File

After you create a remote listener class, use the `RemoteMonitor.properties` file to add it to the list of remote listeners that monitor DCA events.

To Add a Remote Listener Class

To add a remote listener class:

1. In the Admin Tool, select Configuration > RemoteMonitor.
2. For each listener you want to add, on a separate line, specify its Java class name as follows:

   ```
   listener1.class = <class name>

   listener2.class = <class name>

   listener3.class = <class name>
   ```

3. If desired, you can also modify `ServerStatInterval` to change the interval between which DCA statistics are sent to listeners.

   Specify the value in milliseconds. The default is 300000 (5 minutes).

4. Click Submit.

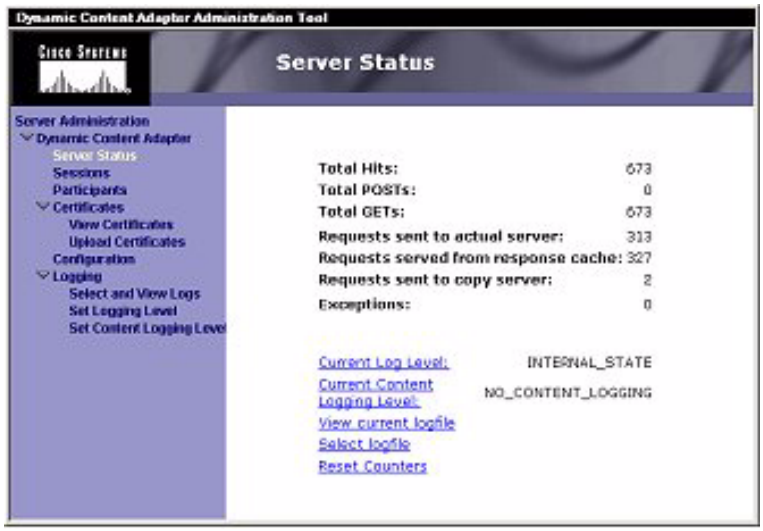
See Also

For related information, see:

How to Implement Remote Monitoring
How to Access the Admin Tool

Section III. Using the DCA Admin Tool

About the DCA Admin Tool



The DCA Admin Tool is a Web-based interface you use to administer and configure the DCA. You can use the Admin Tool to:

- Configure the DCA / edit DCA properties files
- Monitor and manage DCA sessions
- View DCA server statistics
- View DCA logs
- Maintain CA Root certificates
- Confirm the version and build number of your DCA server

The Admin Tool interface consists of an HTML frameset you can open in any DCA-supported Web browser. The DCA must be running in order for you to use the Admin Tool.

You can access the Admin Tool remotely over the Internet; physical access to the DCA server is not necessary.

See Also

For related information, see:

How to Access the Admin Tool

You can access the Admin Tool locally or remotely over the Internet; physical access to the DCA server is not necessary. The DCA must be running in order for you to use the Admin Tool.

To Access the Admin Tool

To access the Admin Tool:

1. Open a Web browser and in the Address line, enter: `http://<DCAservername>/dca-admin` (case-sensitive). Or, if your DCA server is set up to use SSL, you can enter `https` as the protocol.
2. Press <Enter>. The Admin Tool Login screen opens.
3. Enter your DCA Admin username and password.

   The default values for these are `Admin` and `Admin` (case-sensitive).

4. Depending on your localization configuration, you may also have the option of choosing a different on-screen language to use for the current session.
5. Click Login.

See Also

For related information, see:

About the DCA Admin Tool
How to Change the Admin Tool Username and Password
How to Limit Admin Tool Access
How to Localize the Admin Tool

How to Change the Admin Tool Username and Password

Cisco strongly recommends that you change the Admin Tool username and password soon after installing the DCA. The default Admin username and password are:

- Default Username: Admin
- Default Password: Admin

To Change the Admin Username and Password

To change the Admin Tool user name and password:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. In AdminUsername, enter a new username. The new username:
   - Is case-sensitive.
   - Has no minimum/maximum length.
   - Cannot include spaces.
3. In AdminPassword, replace the current password (shown encrypted) with a new one. The new password:
   - Is case-sensitive.
   - Has no minimum/maximum length.
   - Cannot include spaces.
4. Click Submit.

Caution: DO NOT attempt to change the Admin password by editing the `Proxy.properties` file directly in a text editor: The DCA Login window requires an encrypted password. A password created via a text editor is not encrypted.

If You Forget Your Admin Password

If you forget your Admin password and cannot access the Admin Tool:

1. Open the `Proxy.properties` file in a text editor.

2. Delete the value for AdminPassword. DO NOT specify a new password.

3. Save the file.

4. Login to the Admin Tool, leaving the Password field in the Login screen blank. Once in the Admin Tool, you can set a new password.

See Also

For related information, see:

About the DCA Admin Tool
How to Access the Admin Tool
How to Use the Proxy Properties File

How to Limit Admin Tool Access by IP Address

You can limit access to the Admin Tool to one or more IP addresses, or to a range of addresses. By default, access to the Admin Tool is accessible to all IP addresses.

To Specify IP Address Restrictions:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. In AdminHosts, enter one or more IP addresses. Separate multiple addresses with semi-colons. You can specify a range of addresses as follows. To limit access to:
   - Addresses beginning with 161, enter: 161
   - Addresses beginning with 161.44, enter: 161.44
   - Addresses 161.44.248.1 and 162.44.248.1, enter: 161-162.44.248.1
   - Addresses 161.44.248.1 through 161.44.248.50, enter: 161.44.248.1-50
3. Click Submit.

To Remove IP Address Restrictions

To remove IP address restrictions, delete any address values specified for the AdminHosts property.

See Also

For related information, see:

About the DCA Admin Tool
About Admin Tool and DCA Server Security
How to Access the Admin Tool
How to Change the Admin Tool Username and Password

Part 2: Configuring the DCA

About Configuring the DCA

In most cases, you can use the DCA as-is immediately after installing and integrating it with CCS. Unless you will be using SSL, post-installation configuration is usually not required to get the DCA up and running. After you've had a chance to use the DCA with your Web site, you may want to take advantage of its configuration options. They allow you to tune DCA performance, troubleshoot, and customize the DCA interface.

Common DCA configuration options include:

Configuring the DCA for SSL

To share secure content through the DCA, you must first configure the DCA to use SSL.

Configuring DCA for Top Performance

The DCA contains a range of configurable features that allow you to ensure users a responsive and trouble-free Web collaboration experience. As necessary, you can:

- Modify DCA cache settings
- Modify the DCA's default parser configuration
- Specify the source and type of content served through the DCA

Customizing the DCA

You can customize various aspects of the DCA to suit you needs or preferences. Customization tasks may include:

- Customizing the Collaboration Toolbar configuration
- Localizing the DCA interface and parser
- Creating mappings for browser-dependant content
- Specifying you own custom error pages

See Also

For related information, see:

About DCA Properties Files
About DCA Performance
About Using SSL with the DCA
About the DCA Collaboration Toolbar
About Localizing the DCA
About Browser Mapping
About Custom Error Pages

Section IV. DCA Properties Files

About DCA Properties Files

You configure most aspects of DCA behavior through settings in its properties files. DCA properties include default settings that should work well for most users. However, as necessary you can make changes to:

- Tune DCA server performance
- Customize the content requested and served through the DCA
- Customize the DCA interface
- Define DCA session and participant expiration periods
- Localize the DCA
- Troubleshoot the DCA

Editable Properties Files

The following table lists all editable DCA properties files. Most DCA properties files are located at: `<DCArootdirectory>\webapp\WEB-INF\Cisco\properties\`. Unless explicitly stated elsewhere in this guide, DCA files outside of this directory are non-configurable and should not be modified.

Except where noted, editable DCA properties files can be accessed and edited through the Admin Tool interface. This is the method Cisco recommends for editing the DCA properties files.

File:	**Description:**
BrowserMap	Use the BrowserMap properties file to map individual browser types and versions to a list of common denominator browsers (part of browser mapping configuration).
Charset	Use the Charset properties file to manually map HTML content encodings to Java encodings (part of DCA parser localization configuration).
CharsetDefault	Use the CharsetDefault properties file to specify a default Java encoding to use with encoded documents whose coding method is unspecified (part of DCA parser localization configuration).

File:	Description:
BrowserMap	Use the BrowserMap properties file to map individual browser types and versions to a list of common denominator browsers (part of browser mapping configuration).
dca_admin, admin_ui, and fault_strings	Use the dca_admin, admin_ui, and fault_strings properties files to localize the Admin Tool interface. They are located at: <DCArootdirectory>\uiserver\WEB-INF\properties\default\dcaadmin. **Note:** These files are not available through the Admin Tool interface; they can only be edited through a text editor.
dca_console_controls	Use the dca_console_controls properties file to localize the Collaboration Toolbar. It is located at: <DCArootdirectory>\uiserver\WEB-INF\properties\default\dca. **Note:** This file is not available through the Admin Tool interface; it can only be edited through a text editor.
DCALog	Use the DCALog properties file to configure file naming and storage conventions for DCA log files. **Note:** This file is not available through the Admin Tool interface; it can only be edited through a text editor.
Encoding	Use the Encoding properties file to specify equivalent Java encoding methods that should NOT cause document parsing to restart when the mapped encoding changes (part of DCA parser localization configuration).
Page	Use the Page properties file to specify pages requested via a POST that should always be requested from a Web content server, even if a cached copy of the page exists on the DCA.
Proxy	Use the Proxy properties file to specify a number of basic, application-wide behaviors for the DCA.
RemoteMonitor	Use the RemoteMonitor properties file to specify one or more remote listeners that monitor DCA events.
Responses	Use the Responses properties file to designate custom response pages to display in place of standard server error pages when those pages fail to return message text.
toolbar_config	Use the toolbar_config.xml file to configure the controls that appear in the Collaboration Toolbar. It is located at: <DCArootdirectory>\uiserver\WEB-INF\properties\default\dca **Note:** This file is not available through the Admin Tool interface; it can only be edited through a text editor.
TrustedHosts	Use the TrustedHosts properties file to specify hosts whose certificates should be trusted, even if the certificates are expired, unsigned, or otherwise invalid.
UserAgents	Use the UserAgents properties file to define the user-agent headers sent by the to Web content servers, indicating which browser is being used (part of browser mapping configuration).

Non-Editable Properties Files

The following DCA properties files, although they appear in the `properties` directory, should NOT be edited. These files are not listed in the Admin Tool interface.

- CSCconfigOpt.cfg
  - logger.properties
  - logManager.properties
  - logOutputAdapter.properties
  - MessageAdapter.properties

See Also

For related information, see:

How to Edit DCA Properties Files
About DCA Directory Structures
How to Use the BrowserMap Properties File
How to Use the Charset Properties File
How to Use the CharsetDefault Properties File
How to Use the Encoding Properties File
How to Use the Page Properties File
How to Use the Proxy Properties File
How to Use the RemoteMonitor Properties File
How to Use the Responses Properties File
How to Use the TrustedHosts Properties File
How to Use the UserAgent Properties File
How to Modify Log File Properties
How to Configure the Collaboration Toolbar
How to Localize the Collaboration Toolbar
How to Localize the Admin Tool

How to Edit DCA Properties Files

DCA properties files can be modified:

- In the Admin Tool
- In a text editor

Except when otherwise instructed in this guide, Cisco recommends that you edit properties files through the Admin Tool interface. The Admin Tool reduces the possibility of editing errors and allows you to edit properties files remotely over the Internet using a Web browser.

For instructions on editing a specific properties file, see its description elsewhere in this guide.

To Edit Properties Files Using the Admin Tool

Cisco strongly recommends that you use the DCA Admin Tool to edit properties files. Editing properties files through the Admin Tool:

- Reduces the possibility of syntax error.
- Allows you to edit properties files remotely over the Internet using a Web browser.
- Selectively lists only those properties files that are configurable..

To edit DCA properties from the Admin Tool:

1. In the Admin Tool, select Configuration.
2. Click the name of the file you want to edit.
3. Make the desired edits. Some properties files only allow you to edit values for existing properties. Others may allow you to define new properties.
4. Click Submit.

Editing Properties Files Using a Text Editor

While not generally recommended, there are isolated instances in which you may

need to edit a properties file using a standard text editor. This requires the ability to access the specific file on the DCA Server.

DCA Properties that Must be Edited in a Text Editor

You may need to edit a properties file in a text editor to:

Configure the Collaboration Toolbar: The files used to configure the Collaboration Toolbar are not accessible through the Admin Tool. They must be edited in a text editor.

Edit DCA Log File Properties: The `DCALog.properties` file that controls behaviors of the DCA Error and Trace logs is not accessible through the Admin Tool. It must be edited in a text editor.

Reset the DCA Admin password: If you have forgotten your DCA Admin password, you can reset the password to null by opening the `Proxy.properties` file in a text editor and deleting the current password. This allows you to login using no password. Once in the Admin Tool, you can specify a new password.

Modify a property's description: To edit the description of a property that displays in the Admin Tool interface, open the property's file in a text editor and modify its comment text. You can do this for any file whose properties consist of name-value pairs, including: `BrowserMap`, `Proxy`, `RemoteMonitor`, and `UserAgents`.

Enable DCA proxy server properties: The DCA installer assumes you will not use a proxy for your DCA Server. Therefore, the properties used to configure a proxy (in `Proxy.properties`) are commented out and are not visible in the Admin Tool. To enable these properties, open Proxy.properties in a text editor and remove the comments.

Text File Editing Notes

When editing properties files in a text editor, be careful that you:

- Do Not Delete Blank Lines

  When editing a properties file whose settings consist of name-value pairs (for example, `Proxy.properties)`, be careful not to delete the blank lines that separate individual properties. These blank lines are required for correct parsing of the file.

- Do Not Edit Lists of Comma Separated Values

  Do not modify any line that contains a list of comma separated values enclosed in

brackets, as shown below. These lists comprise the range of acceptable values for a property:

```
# (value1, value2, value3, etc.)
```

See Also

For related information, see:

How to Access the Admin Tool
How to Change the Admin Tool Username and Password
How to Modify Log File Properties
About DCA Properties Files
How to Configure the Collaboration Toolbar

How to Use the Proxy Properties File

The `Proxy.properties` file is the main configuration file for the DCA. It contains a variety of properties used to set up, administer, customize, and improve performance for the DCA.

To Edit the Proxy.properties File

To edit the `Proxy.properties` file:

1. In the Admin Tool, select Configuration > Proxy.
2. Edit properties as needed.
3. Click Submit.

Note: Changes to properties whose named appear in **bold** (in the table below) also require you to restart the DCA server.

Property Descriptions

The following table lists individual properties in the `Proxy.properties` file. Properties are listed in alphabetical order, not the order in which they appear in the file. Edits to properties whose named appear in **bold** require a DCA server restart to take effect.

Property	**Description**	**Default Value**
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
AdminPassword	Login password for the Admin Tool. Displays as encrypted. Password: • Is case-sensitive. • Has no maximum/ minimum length. • Cannot include spaces.	Admin

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
AdminUsername	Login username for the Admin Tool. Username: • Is case-sensitive. • Has no maximum/ minimum length. • Cannot include spaces.	Admin
BrowserMapping	Enables browser mapping.	True
ContentLogLevel	Specifies what, if any, Web content information is stored in the DCA Trace log.	NO_CONTENT_LOGGING
CopyServerPages	Specifies URLs that can be stored on the DCA, either in the cache or on the Copy Server. Specify URLs by entering one or more string patterns to compare against incoming URLs. URLS that match the pattern can be stored on the DCA. URLs that do not match the pattern are not stored and must always be retrieved from a Web content server. Note that non-stored pages are still parsed by the DCA. Typically, all pages requested through the DCA should be stored. Not storing pages essentially cancels the benefit of the DCA. However, this property can be used for those instances in which particular pages should not be stored (due to security reasons, for example). Use semi-colon delimited Perl5 regular expression syntax. Examples: All: .* With cisco.com: .*cisco.com.* With cisco.com or yahoo.com: .*cisco.com.*;.*yahoo.com.*	.* (all)
CopyServerPollAttempts	Number of times the DCA will attempt to retrieve a page from the Copy Server before requesting it from a Web content server.	5
CopyServerPollInterval	Interval between Copy Server poll attempts (in milliseconds).	1000

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
CopyServerRoot	Copy Server RMI URL (the location of the Copy Server). Set at installation.	//localhost/CopyServer
DoFirstOfSessionRedirect	Specifies whether to allow a first-of-session redirect to a URL that includes a DCA session ID. If True, and if an initial DCA URL does not include a session ID, the DCA will autogenerate a session ID and append it to the URL.	True
enableDCACCSConnection	Specifies whether to allow a connection between the DCA and CCS server(s). The connection allows for DCA-CCS session integration and integrated reporting.	False
ExcludedFileTypes	Specifies non-text file types that should not be requested through the DCA. Requests for files of specified types are redirected to a Web content server. While non-text file types are not stored by the DCA, requests for them are still routed through the DCA unless the file type is specified here. **Note**: ExcludedFileTypes supercedes any specification in the DCA parser regarding which file types should be requested through the DCA.	zip;gif;jpg;jpeg;pdf;doc
HttpProxyPort	If the DCA will use a proxy server for outbound connections, the proxy server's HTTP port. For more information on proxy servers, see the *DCA Installation and Integration Guide*.	None Note: Because it is rarely used, by default, this property is not visible in the Admin Tool. To make visible, use a text editor to uncomment the entry.
HttpProxyServer	If the DCA will use a proxy server for outbound connections, specify the proxy server name (fully-qualified DNS or IP address).	None Note: Because it is rarely used, by default, this property is not visible in the Admin Tool. To make visible, use a text editor to uncomment the entry.
HttpsProxyPort (ignored if HttpProxyServer is null)	If the DCA will use a proxy server for outbound SSL connections, the proxy server's HTTPS port. For more information on proxy servers, see the *DCA Installation and Integration Guide*.	None Note: Because it is rarely used, by default, this property is not visible in the Admin Tool. To make visible, use a text editor to uncomment the entry.

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
HttpsProxyServer (ignored if HttpProxyServer is null)	If the DCA will use a proxy server for outbound SSL connections, specify the proxy server name (fully-qualified DNS or IP address).	None Note: Because it is rarely used, by default, this property is not visible in the Admin Tool. To make visible, use a text editor to uncomment the entry.
LogLevel	Specifies the system logging (message priority) level used by the DCA Trace log.	INTERNAL_STATE
LongWaitTime	Specifies the interval that pages submitted to the Copy Server remain available (in seconds).	600
MatcherPoolSize	Initial size of the pool available for pattern matching. Increases as number of processors increases (n * 15 where n = number of processors). Increases as necessary to provide a sufficient number of matchers	15
MaxOutputCacheSize	Maximum number of pages stored in the Output cache. The Output cache stores reconstructed pages. Cisco recommends constraint in modifying this property. A significant increase to the default may increase demands on memory without a measurable performance benefit.	50
MaxResponseCacheSize	Maximum number of pages stored in the Response cache. The Response cache stores parsed pages. Cisco recommends constraint in modifying this property. A significant increase to the default may increase demands on memory without a measurable performance benefit.	50
ParserPoolSize	Initial size of the pool available for parsing html files. Increases as number of threads increases (n * 4 where n = number of threads). Increases as necessary to provide a sufficient number of parsers.	30

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
ParticipantTimeout	Specifies the duration interval of participant cookies (in seconds). Should be equal to or greater than SessionTimeout. A high number helps prevent participants whose PC clock time is fast from receiving cookies that expire prematurely.	3600
ProxyBasePath	DCA Server path used in DCA URL requests. Case-sensitive. Appears in URL requests as: http://<DCAservername>**/DCA**/http://<someURL> **Caution:** DO NOT EDIT unless advised by a certified Cisco representative.	/DCA
ResponseURLFilename	Name of the file containing customized pages (to display in place of standard internet server error pages when those pages return no text). Stored in <DCArootdirectory>\webapp\WEB-INF\Cisco\properties.	Responses.properties
RootCAFile	Name of the file that stores root Certificate Authority certificates used to verify secure server certificates.	cafile.txt
SavedCookieDomains	Domains whose cookies should be returned to a DCA participant PC as well as the DCA (normally, cookies are returned and stored only on the DCA Server). Returned cookies are only valid for the DCA Server. Use semi-colon delimited Perl5 regular expression syntax. Examples: All: .* With cisco.com: .*cisco.com.* With cisco.com or yahoo.com: .*cisco.com.*;.*yahoo.com.*	
SetProxyCookieDomains	Specifies whether a domain is available to associate with DCA participant cookies. If True, the DCA will attempt to set a domain on the cookie.	True

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
SSLVersion	By default, the DCA supports SSL versions 2 and 3 concurrently. Some older Web servers running SSL 2.01 may have difficulty handling this SSL negotiation. Symptoms of the problem include: • Users encounter server errors when attempting to access pages from the SSL server; • An exception appears in the DCA log bearing the message "SSL 2 bad mac decode." To alleviate this problem, set the value for SSLVersion to ssl_v2. Note however that with this setting, the DCA cannot share secure content from Web servers running SSL 3.0. A more complete resolution to the problem is to upgrade Web servers that experience this problem. The full list of acceptable values for this parameter include: ssl_v23 (the default -- SSL versions 2 and 3 supported) ssl_v2 (SSL version 2 only supported) ssl_v3 (SSL version 3 only supported) tls_v1 (Transport Level Security version 1 supported)	ssl_v23
SubmitExpireRemove ThreadInterval	Specifies how often the DCA removes references to expired pages (in seconds). **Note:** This property does not control the actual removal of pages from the DCA caches or Copy Server. Pages are removed from a cache based upon the cache's maximum size. Pages are removed from the Copy Server based on its CleanAge property.	60

Property	Description	Default Value
AdminHosts	Limits access to the DCA Admin Tool to specified IP addresses, or to a range of addresses.	
SubmitExpireTimeout	Specifies how quickly pages stored by the DCA expire (in seconds). Once a page expires, it is unavailable to users, even if the page is still present in the cache or on the Copy Server. If your site is slow to return pages, you may need to increase the timeout period to prevent pages from expiring too quickly.	40

See Also

For related information, see:

How to Change the Admin Tool Username and Password
How to Limit Admin Tool Access
How to Configure Browser Mapping
How to Configure the DCA Cache
How to Configure the Copy Server
How to Configure the Session Timeout Period
How to Configure the Participant Timeout Period
About DCA Proxying
How to Set the Trace Log Logging Levels
How to Maintain Root Certificates
About SSL Versions
How to Access the Admin Tool
How to Stop and Start the DCA

Section V. Performance Configuration

About DCA Performance

The most important factors affecting DCA performance are processor speed and available memory. Beyond these, DCA performance is most directly affected by the factors listed below.

- DCA Server Load: Factors related to DCA server load, such as the number of concurrent sessions and complexity of pages being requested affect DCA performance.
- Cache configuration: The DCA cache stores copies of pages requested during a DCA session. This eliminates the need for multiple requests to the content server for pages viewed more than once during a session. You can configure the size of the cache and how long pages remain within it.
- SSL: In a DCA/SSL configuration, every DCA session is conducted in SSL mode and each resource on a page is delivered through the DCA. This includes non-text files that would normally come directly from a Web content server. Thus SSL can have a significant effect on DCA performance.
- Files requested through the DCA: The DCA has several configuration options that allow you to specify which files should be requested and stored from the DCA, and which should always be retrieved directly from a Web content server.
- Parser complexity: Out of the box, the DCA parser is configured to search for and reformat HTML links only. However, you can customize the parser to match any kind of document text and to modify pages in any number of ways.

Improving DCA performance is rarely a matter of simple reconfiguration. While it's possible to achieve improved performance through DCA configuration (for example, by changing the cache size), other factors that affect performance may require broader solutions.

See Also

For related information, see:

About DCA Server Load
About DCA Caching
About SSL and DCA Performance
About DCA Proxying
How to Configure the DCA Cache
How to Configure the Static Cache
How to Configure the Copy Server
About Customizing the Parser

About DCA Server Load

The following factors related to server load can affect DCA performance.

Number of Sessions

Each DCA license is valid for a single DCA server and allows an unlimited number of seats. However, performance considerations may impose a practical limit on the number of concurrent sessions the DCA can effectively handle.

As a rough guideline, the DCA 2.01 Minimum configuration should deliver adequate performance for 40 concurrent sessions. The Enterprise configuration should deliver adequate performance for 100 or more concurrent sessions. (Minimum and Enterprise hardware specifications are described in the *DCA 2.01 Installation and Integration Guide*.) Note, however, that these limits can be higher or lower, subject to other factors such as DCA cache size, size and complexity of content, number simultaneous requests, use of SSL, and so on.

Complexity of Pages

The relative complexity of pages being requested through the DCA can affect its performance. Items affecting a page's complexity include:

Number of parsable items: The DCA parses each item on a page that references a URL (in one way or another). This can include text links, image maps, forms, frames, and so on.

Page size: Page size specifically refers to the size of the page's text content

When the number of concurrent DCA sessions is low, the effect of large or complex pages on DCA performance is minimal. However, the effect can become more pronounced as the number of concurrent sessions increases.

Simultaneous Page Requests

Simultaneous page requests refers to the number of DCA users who request a unique page (for example, by clicking a link) at precisely the same time. Note that multiple requests for the same cached page represents only one request.

Also, the frequency with which users request pages affects DCA performance. A relatively short average interval between requests (15-30 seconds) puts more strain on the DCA than a longer one (60-120 seconds).

Other Server Load Performance Factors

Other load-related factors that can affect DCA performance include:

Running other applications on the DCA Server: This is not recommended. The server resources should be dedicated to the DCA and its Web server software.

Logging Level: When under load, the DCA writes a large amount of log entries. Setting the Log Level to a lower level can conserve some CPU power, though the affect is minimal.

See Also

For related information, see:

About DCA Performance
About DCA Sessions
About DCA Load Test Results
The DCA Installation and Configuration Guide

About DCA Caching

The DCA includes these mechanisms for storing the content it receives from Web content servers:

- The DCA Cache
- The Copy Server
- The Static Cache

The DCA Cache

Pages returned to the DCA from a Web content server are stored in the DCA cache. The DCA cache allows the DCA to store and return pages quickly from memory. The DCA cache only stores the text of served pages; the DCA cache does not store non-text file types such as images, executables, archives, PDFs and so on.

The Response and Output Cache

There are actually two DCA caches: the Response cache and the Output cache.

- **Response Cache:** When a page is initially returned to the DCA from a Web content server, the DCA parses the page, redirects all links on the page to the DCA, and then stores the page as a Java object in its Response cache.
- **Output Cache:** After a page is stored in the Response cache, the DCA reconstructs it and passes it to the Output cache, where it is available to users. Subsequent requests for the page are satisfied directly from the Output cache.

Cache Size and Page Expiration

The size of each cache is determined by a configurable property. By default, each cache stores up to 50 pages. When the maximum size is reached, new pages coming into the cache cause older pages to be forced out. When an unexpired page is forced from the Response cache, it is written to the Cisco Copy Server, where it remains available to users until it expires.

Pages stored in the DCA cache expire based on a configurable timeout. Once a page has expired, it is unavailable, even if it still remains in the cache. Subsequent requests for an expired page are satisfied from a Web content server.

The Copy Server

The Cisco Copy Server is a mechanism that allows the DCA to store pages that have been forced from the DCA cache but have not yet expired. In this way, the pages still remain available to users. Unlike the DCA Cache, which stores pages in memory, the Copy Server writes pages to disk. The Copy Server is installed automatically when you install the DCA.

Although there is usually no reason to modify the Copy Server's default settings, you can change certain defaults if desired. Note that Copy Server configuration does not significantly affect DCA performance.

How the DCA Searches for Stored Pages

When a user requests a previously accessed page during a DCA session, the DCA searches in this order for a stored copy of the page in:

1. The Output cache
2. The Response cache
3. The Copy Server

If an unexpired copy of the page is not available from any of these sources, the request is then forwarded to a Web content server.

The Static Cache

In non-SSL (HTTP) mode, non-text files (such as images) are typically not requested through the DCA: they are requested and delivered to participants directly from a Web content server. The DCA does not cache non-text files when used in non-SSL mode.

In SSL (HTTPS) mode, however, non-text files are always and automatically requested and served through the DCA; this is to avoid the security warnings participants would otherwise receive due to content derived from different sources.

To speed performance, when used in secure mode, the DCA stores non-text files in a special cache named the Static cache. You can modify the size and other attributes of the Static cache.

See Also

For related information, see:

About DCA Performance
About SSL and DCA Performance
About DCA Proxying
How to Configure the DCA Cache
How to Configure the Static Cache
How to Configure the Copy Server

About SSL and DCA Performance

To share secure content through the DCA, the Collaboration Toolbar must be loaded in secure mode (HTTPS). The setting that controls this resides on the Collaboration Server and makes a DCA/CCS server combination either full-time SSL or full-time non-SSL.

In a full-time SSL configuration, every DCA session is conducted in SSL mode, regardless of whether the actual pages requested are secure. In addition, to prevent security warnings, each resource on a page is delivered through on the DCA, including non-text files such as images and multimedia that would normally come directly from a Web content server. These non-text files are stored in the DCA's Static cache.

Thus, SSL imparts additional overhead to the DCA that can affect performance. This overhead is difficult to quantify, but may significantly affect the number of concurrent sessions the DCA can effectively handle.

See Also

For related information, see:

About DCA Performance
About Using SSL with the DCA
About DCA Caching
About DCA Proxying
How to Configure the Static Cache
How to Configure the Collaboration Toolbar to Use SSL

About DCA Proxying

The DCA parses each link on a Web page it receives to point back to itself. This means that each file request emanating from a parsed page is sent first to the DCA. If the file in question is not found there (in the cache) the request is sent on to a Web content server.

While this is normally desirable behavior, there may be instances in which for performance, security, or other reasons you want certain files to be requested or served directly from a Web content server. The DCA has several mechanisms that allow you to control this.

Text versus Non-Text Files

From a DCA perspective, *text files* include only files that can be served as text from a Web content server to a browser (for example, html, jsp, asp, etc.). The DCA parses and caches text files.

All other files (for example, images, PDFs, Flash, Word Documents, etc.) are regarded are *non-text*. The DCA does not parse non-text files and in a typical configuration does not cache them.

Text File Proxying

In most cases, it is desirable for the DCA to cache all text files (i.e., Web pages) it retrieves during a DCA session. Caching pages prevents the DCA from having to request and parse the same page multiple times if users request it more than once.

However, there may be instances in which you prefer particular pages not be stored, or that requests always be satisfied from a Web content server. To control this:

- The `CopyServerPages` property in the `Proxy.properties` file allows you to specify URLS that should not be cached by the DCA. Although these pages will continue to be *requested* through and parsed by the DCA, they will always be served from a Web content server.
- The `Page.properties` file allows you to specify pages requested via a form POST that should always be requested from a Web content server, even if a cached copy of the page exists.

From a performance standpoint, the list of pages to be always requested and delivered from a Web content server should be kept to a minimum.

Non-Text File Proxying

Because the DCA does not parse or store non-text files (images, zips, PDFs, etc.), there is usually no reason to request them through it. Requesting non-text files through the DCA comprises an extra layer of request routing that is unnecessary and can adversely affect performance.

To stop non-text files from being requested through the DCA, you can use the `ExcludedFileTypes` property in the `Proxy.properties` file. It lets you specify file types that should always be requested directly from a Web content server. The default setting for `ExcludedFileTypes` already includes a number of common non-text file types. `ExcludedFileTypes` supercedes any specification in the DCA parser regarding which file types should be requested through the DCA.

Note: `ExcludedFileTypes` does not affect DCA sessions conducted using SSL; in DCA sessions that use SSL, ALL files, text and non-text, are always requested through the DCA to avoid the security warnings that would otherwise occur.

See Also

For related information, see:

About DCA Performance
About DCA Caching
About SSL and DCA Performance
How to Use the Page Properties File
How to Use the Proxy Properties File

How to Use the Page Properties File

Use the `Page.properties` file to specify pages requested through a form POST that should always be requested from a Web content server.

Pages Updated Via a POST Command

Your Web site may contain pages requested via POST that should always be requested from a Web content server, even if a copy of the page already exists in the DCA cache.

For example, suppose your site uses a frameset which displays two pages: A and B. Users enter data in A, click a link that results in a POST, and the posted data updates the data displayed in B. During a DCA session, if a cached copy of B already exists in the DCA cache, clicking the link in A would return the cached copy of B, rather than an updated version.

To resolve this problem, in the `Page.properties` file you can specify B as a page that must always be returned from a Web content server.

To Specify Pages that Must Come from a Web Server

To specify POST pages that should always be requested from a Web content server:

1. In the Admin Tool, select Configuration > Page.
2. After `mustbesubmitted=`, enter one or more URL patterns. URLs containing these patterns must always be returned from a content server.
   - To include wildcards in a pattern, use valid Perl5 regular expression syntax.
   - Separate multiple patterns with semi-colons.
   - Example: `mustbesubmitted=.asp;cisco.com` (URLs containing `.asp` or `cisco.com` are always requested from a content server).
3. Click Submit.

See Also

For related information, see:

How to Configure the DCA Cache

Under normal circumstances, the DCA cache should function adequately using its default settings. Cisco recommends that you use the DCA with its default cache sizes for a time before modifying these values. As necessary, however, a judicious increase may deliver better performance.

To Configure the DCA Cache

To configure the DCA cache:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. Modify values for the MaxResponseCacheSize, MaxOutputCacheSize, and/or the SubmitExpireTimeout parameters, as desired.
3. Save the file.
4. Restart the DCA.

DCA Cache Property Descriptions

The DCA cache has these editable properties:

Property:	**Description:**	**Default:**
MaxResponseCacheSize	Maximum number of pages stored in the response cache.	25
MaxOutputCacheSize	Maximum number of pages stored in the output cache.	25
SubmitExpireTimeout	Specifies how quickly pages stored by the DCA expire.	40

Cache Configuration Considerations

When modifying the cache size, bear the following in mind:

Don't devote resources to storing expired pages

Pages stored by the DCA expire based on a configurable period (the default is 40 seconds). Once a page expires, it is unavailable to users. This is true even if the page still exists in the cache. The cache size, therefore, must be big enough to store unexpired pages (that is, pages that have been accessed within the past 40 [default] seconds), but not so big that the DCA devotes resources to storing a large number of expired pages.

Participants in a given DCA session can reference only a few pages in any 40 second interval. Therefore, the point of increasing the cache is to store pages for an increasing number of concurrent DCA sessions, not to make a large number of pages available for users in a single session.

Why not increase both the page expiration time and the cache size?

There is value in having pages expire. The nature of SPLIT content is that it changes, often based upon the time it is requested. By allowing pages to expire in a reasonable period, page content is refreshed from a Web content server at the rate intended by the site's designers.

Increasing the cache size places additional demands on memory

It is possible to increase the cache size to a point where any performance benefit from the larger cache is offset by degradation due to insufficient memory. When increasing the cache size, you may need to make a corresponding increase to Virtual Machine memory.

Calculating the correct cache size

There is no simple formula for calculating the correct cache size. It is a matter of experimentation and observation. However, as a rough indicator, you may want to try multiplying the average number of concurrent DCA sessions you expect by two (with two representing an agent and a client). The result may indicate a reasonable cache size.

For example, if you estimate the average number of concurrent DCA sessions to be 20, and you multiply this by two, you arrive at a cache size of 40.

Bear in mind that this is only a rough indicator. Large results should be adjusted downward.

See Also

For related information, see:

About DCA Performance
About DCA Caching

How to Configure the Static Cache

During a secure DCA session, non-text files (images, multimedia, etc.) returned from a Web content server are stored in the DCA's static cache. Under normal circumstances, the Static Cache should function adequately using its default settings. As necessary, however, you can modify the amount of memory and disk space available to the Static cache.

To Configure the Static Cache

To configure the Static Cache:

1. In a text editor, open the `web.xml` file, located at: `<DCArootdirectory>\webapp\WEB-INF\.`
2. For the HttpCache servlet, modify values for the maxMemorySize, maxFileSize, and/or the cleanOnExit parameters, as desired.
3. Save the file.
4. Restart the DCA.

Static Cache Property Descriptions

The Static cache has these editable properties:

Property:	**Description:**	**Default:**
maxMemorySize	Maximum amount of memory the Static cache can use to store objects. Objects forced from the cache's memory are written to disk.	10240 (10 Mb)
maxFileSize	Amount of disk space the Static cache can use to store objects.	10240000 (1 Gb)
cleanOnExit	Determines whether objects written to disk by the Static cache are automatically deleted when the DCA server is stopped.	true

See Also

For related information, see:

About DCA Performance
About DCA Caching
About SSL and DCA Performance

How to Configure the Copy Server

Although there is usually no reason to modify the Copy Server's default settings, you can change certain defaults if desired. For example, you can modify settings that determine how long pages remain on the Copy Server). Note that Copy Server configuration does not significantly affect DCA performance.

Copy Server properties are set in two places:

- The ServletExec Admin Tool
- The `Proxy.properties` file

Each gives you access to a different set of Copy Server properties.

To Configure the Copy Server in ServletExec

To edit the Copy Server's properties in ServletExec:

1. Open the ServletExec Admin Tool.
2. In the Admin Tool navigation frame, click Servlets > Configure.
3. Click CopyServer.
4. Make changes, as desired.
5. Click Submit.
6. Restart the DCA.

ServletExec Copy Server Properties

The following Copy Server properties can be modified using the ServletExec Admin Tool.

Property	Description	Default
CaptureCookies	(Obsolete) Specifies whether the Copy Server stores cookies for use in future DCA Snaplet sessions.	True

Property	Description	Default
CleanAge	Age of stored pages to be deleted by cleanup (in minutes).	60
CleanInterval	Specifies the interval between cleanups (in minutes).	60
CleanOnInit	Specifies whether the Copy Server should remove all existing pages at startup.	True
CleanThread	Specifies whether the Copy Server should use a cleanup thread.	True
CopyRoot	The location for pages stored by Cisco Copy Server. Set at installation do not edit.	<DCArootdirectory>\WLCopies
LogLevel	Controls the quantity of output written to ServletExec's event log file. For production, 0 or 1 are the most practical settings. During development, 3 or 4 may be used to view additional information. 0 - Startup and shutdown info messages, all error conditions 1 - Additional Startup and shutdown info messages 2 - Cyclical Copy Server info messages (count of copies removed, etc.) 3 - Cyclical Copy Cleanup info messages (count of copies removed, etc.) 4 - Additional cleanup info (directories processed, etc.) 5 - Not Used 6 - First level debug 7 - Second level debug 8 - Third level 9 - Extreme Verbosity	0
Properties	Name of Cisco Copy Server properties file. This file must be located in <DCArootdirectory>\properties.	CSConfigOpt.cfg
ValidateSession	Specifies whether the Copy Server should validate each page request using cookie information from the requester. Must be False to use the Copy Server with the DCA.	False
WebLineRoot	The parent directory for the DCA properties files directory. Set at installation do not edit.	<DCArootdirectory>\Cisco

To Configure the Copy Server in the Proxy Properties File

To edit the Copy Server's properties in the `Proxy.properties` file:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. Edit properties as needed.
3. Click Submit.

Proxy Copy Server Properties

The following Copy Server properties can be modified in the `Proxy.properties` file.

Property	Description	Default Value
CopyServerPages	Specifies URLs that can be stored on the DCA, either in the cache or on the Copy Server. Specify URLs by entering one or more string pattern to compare against incoming URLs. URLS that match the pattern can be stored on the DCA. URLs that do not match the pattern are not stored and must always retrieved from a Web content server. Use semi-colon delimited Perl5 regular expression syntax. Typically, all pages requested through the DCA should be stored. Not storing pages essentially cancels the benefit of the DCA. However, this property can be used for those instances in which particular pages should not be stored (due to security reasons, for example).	.* (all)
CopyServerPollAttempts	Number of times the DCA will attempt to retrieve a page from the Copy Server before requesting it from a Web content server.	5
CopyServerPollInterval	Interval between Copy Server poll attempts (in milliseconds).	1000
CopyServerRoot	Copy Server RMI URL (the location of the Copy Server). Set at installation.	<DCAservername>\copyserver

See Also

For related information, see:

About DCA Caching

About the DCA Admin Tool

How to Use the Proxy Properties File

How to Use the ServletExec Admin Tool

About DCA Load Test Results

Cisco Systems conducted tests to measure the DCA 2.01's response time in satisfying page requests under an increasing concurrent session load. The results of these tests are described below.

DCA response time reflects the combination of the DCA's page parse time and the extra "hop" the DCA interposes between a browser and Web content server. Page parse time is recorded in the DCA log. When testing DCA performance on your own site, consider factors outside of the DCA that also affect response time (for example, network load and Web content server load). Cisco suggests that you test pages both with and without the DCA.

Windows 2000 Load Test Results

The test described below was conducted on a DCA 2.01 Windows 2000 configuration.

The Test Environment

In our test, virtual users requested eight different pages of varying complexity from the DCA. To simulate the typical interval between page requests in a real DCA session, pages were requested at intervals ranging from 15 to 120 seconds.

Tested pages were located inside Cisco's corporate domain and requests were made over Cisco's corporate local area network. To ensure that readings were not affected by Internet latency, no requests were made to outside servers.

Test Server Specifications:

- Processor: Dual 866 MHZ Pentium 3
- Memory: 1 Gb
- Virtual Machine Memory: 768 Mb

Test Results

The following table lists the average amount of time, in seconds, the DCA took to satisfy page requests for a variety of pages under different concurrent session loads.

# of Sessions:	Page A 54kb with 90kb in images	Page B 56kb with 43kb in images	Page C 50kb with 65kb in images	Page D 39kb with no images	Page E 15kb with no images	Page F 30kb with 8kb in images	Page G 13kb with 25kb in images	Page H 12kb with 9kb in images
20	3.4	3.2	2.6	0.2	0.4	0.2	0.2	0.1
40	5.1	4.9	3.1	0.5	0.3	1.4	0.2	0.1

See Also

For related information, see:

About DCA Performance
About DCA Server Load

Section VI. Parser Configuration

About Customizing the DCA Parser

The DCA parser's out-of-box configuration has been designed and tested to deliver trouble-free performance on a wide range of Web content. For many users, therefore, there may be no need to modify the parser's default functionality. Furthermore, many content-related collaboration issues can be addressed more expediently using other DCA configuration options.

However, as desired you can customize the DCA parser in any manner you choose. The parser allows full customization, even to the extent replacing all default parsing functionality with your own. Parser customization allows to you to add to the DCA parsing process literally any behavior that can be attained through JavaScript.

Reasons for Customizing the Parser

Common reasons to customize the parser include:

- **Troubleshooting parsing errors**: The most obvious reason to modify the parser is to correct parsing errors. Examples of parsing errors include:

  Improperly parsed links: Some Web pages may contain hard-to-find links that do not parse correctly using the parser's out-of-the-box configuration. Parsing errors of this type often manifest themselves as JavaScript errors. JavaScript errors that occur only when a page is accessed through the DCA almost always indicate a parsing problem.

  Frameproofed Pages: Because the Collaboration Toolbar uses a frameset, DCA users cannot collaborate on frameproofed pages. The DCA's out-of-box configuration addresses this by parsing out many common frameproofing techniques. Frameproofing methods not handled by the parser's default configuration and which cause a page to "break out" of the Collaboration Toolbar frameset may require parser modification

- **Adding functionality or content to pages**: You can use parser configuration to insert code or content into pages served through the DCA.

- **Customizing pages for different users**: You can configure the parser to reformat pages differently for different users, including agents and callers.

Background Knowledge

Customizing the DCA parser can be a complex process with a distinct learning curve. In particular, creating expressions that match exactly the pieces of document text

you want to match can be very challenging. It is a good idea to allow for a period of learning and experimentation.

To work with parser files, you should first be familiar with the following:

- The design and function of the DCA parser. This includes:
    i. How, when, and why the DCA parses pages.
    ii. The elements, attributes, and values used in the parser.
    iii. The parser's Script environment, along with its objects and interfaces (the parser API).
    iv. The contents of the base-level parser files that ship with the DCA.
- XML syntax: DCA parser files are specified in XML. While an advanced knowledge of XML is not necessary, there are several syntactical requirements that you should be aware of.
- Perl syntax: The Accept statements you include in parser rules to match text are written as Perl5 regular expressions.
- JavaScript and Java: The scripts you include in parser rules to modify content are written in JavaScript and reference Java objects.

Planning Parser Customization

Because modification to the parser can have broad implications, consider carefully changes you plan to make prior to implementing them. It is strongly suggested that you do not modify the base-level parser files that ship with the DCA. Instead, make and modify copies, or extend these parsers (into other files) as necessary.

Among the things you should consider when modifying the parser:

- Have a clear sense of what it is you want to achieve. Are you correcting a problem? Adding a behavior? How will your changes function in different environments?
- Parser modification can be labor-intensive. Is there an alternate method (e.g., property file configuration, Web site modification) that can achieve the results you want more expediently?
- Determine whether modification to an existing rule can achieve the results you want, or whether you need a new rule altogether.
- Know when in the parsing process you want the modification to happen. Do you want your scripts to run when text is matched? When the document is being rendered?

- If creating an Accept statement to match text, have a clear sense of what document text it is that you want to match. Be sure that your regular expression matches exactly and only the text you want it to match.

- Consider how the script portion of a rule is used in the larger parser context. For example, scripts can alter the environment used by subsequently run scripts. Could your scripts adversely affect scripts that run later? What will become of your script once it is in the client?

See Also

For related information, see:

About the Parsing Process
About the Parsing Engine
About Parser Scripts
About the Parser Script Environment
About Extending the Script Environment
The DCA Parser API Guide

About the Parsing Process

Page Parsing During in a DCA Session

The first and most important job of the DCA parser is to reformat links on pages requested during a DCA session to point to the DCA (or, in the case of relative links to non-text file types such as images, to point to the Web content server). This is what makes Web collaboration through the DCA possible.

The process occurs like this:

1. A user in a DCA collaboration session clicks a link or enters a URL, and the request is sent to the DCA.

2. The DCA forwards the request to a Web content server which returns a page to the DCA.

3. After receiving the page, the DCA parses it to locate all HTML links.

4. The DCA parser reformats all found links to point to the DCA.

5. The DCA reconstructs the page and returns it to the requesting user. Subsequently, when a participant clicks a link on the page, the request is sent to the DCA and the requested page is put through the same parsing process.

   A copy of the parsed page is also stored in the DCA's output cache. This can be used to satisfy subsequent requests for the page.

Other Link Reformatting Methods

In addition to clicking a link on a parsed page, DCA participants can also request a page by:

- Typing a URL into the Collaboration Toolbar's Address Bar.

- Clicking a link in the Script Area of the CCS Agent Desktop.

The DCA reformats *these* URLs not through the parser but through JavaScript functions in Collaboration Toolbar and Agent Desktop respectively. This reformatting occurs automatically when the URL request is submitted.

What Links Does the DCA Parse?

The DCA can only parse links that it is able to find. In order for the DCA to be able to find a link:

- **The link must be in a file served through the DCA**

  The DCA can only parse pages that have been served through the DCA -- that is, that have been accessed through the Collaboration Toolbar or from the Script Area on the CCS Agent Desktop. Examples of pages not served through the DCA include pages accessed from a browser's bookmark menu or Address Bar (for this reason, its recommended that you make these browser window features unavailable to users).

- **The link must be in a text file**

  The DCA can only parse text files -- that is, files that can be served as text from a Web content server to a browser. Examples of text files include html, js, jsp, asp, and so on. All other files are regarded are *non-text* and are not parsed the DCA. Examples of these include image files, Flash, PDFs, and MS Word documents.

- **The link must be locatable by the DCA parser's pattern matching**

  The DCA can only parse links that can be located by the parser's pattern matching. By default, the DCA parses any link or tag it can find that is used to make a request (i.e., for a file). This includes links in A, LINK, FORM, IMG, FILE, and SCRIPT tags.

  The parser's default configuration has been widely tested and should be sufficient to located most links. Any links not found can be addressed through appropriate modification to the parser's pattern matching.

DCA URL Format

When the DCA parses a link, it reformats it to point back to the DCA. This ensures that content subsequently requested through that link is served through the DCA.

The URL format used by the DCA is hard-coded and cannot be modified; the DCA applies it automatically and it is essentially transparent to users. Agents do not need to consider it when entering URLs. Configuration specialists do not need to specify it when modifying the parser or other aspects of the DCA.

DCA URL Format

The DCA formats links as follows:

- Before reformatting: `http://www.cisco.com/myfile.htm`
- After reformatting:
  `http://www.mydca.com/DCA/http/www.cisco.com/myfile.htm`

In the example above, the first part of the URL represents the address of the DCA server. The second part represent the path to the DCA on the server. Finally, the third part represents the WWW path to the document being requested.

Parsing Stages

DCA document parsing consists of two stages:

1. In the *decomposition* (first) stage, a document is parsed according to a collection of rules and regular expressions. These rules and regular expressions are used to match text representing links, tags, or other pieces of document text to be modified in the document.
2. In the *rendering* (second) stage, the document is reassembled to produce a new document. It is at this stage that text matched during the first stage is reformatted or otherwise modified according to scripts you provide to the parser.

A document is *decomposed* only once -- when it is initially returned to the DCA from a Web content server. However, depending on how often and by whom it is requested from the DCA, it may be *rendered* multiple times. These multiple renderings can allow you to reformat a document differently at different times and for different users. For example, you can reformat a document differently agents and callers.

Match Outcomes

Different pieces of text matched during the decomposition stage may require different courses of action:

- In the most common scenario, the matched text will represent a link to a Web page or other text file that must be reformatted to point to the DCA.
- Other links, such as those for images and other non-text files, may need to be modified to point to their original Web content server.
- Additional document sections may require other actions; these are often discovered in the field, where rewriting scripts allows new behaviors to be

created as necessary.

Client-Side Parsing

The DCA includes both a server-side and a client-side version of its parser. The client-side parser allows the DCA to correctly parse and, if necessary, modify content generated on a client. Users download the client parser transparently when they load the Collaboration Toolbar at the start of a DCA session.

The client-side parser allows the DCA to correctly parse content generated on the client (for example, links created through JavaScript document.write statements after a page loads in the client).

How the Client-Side Parser Loads

On the DCA server, the XML files that comprise the parser are used to create a combination of Java and JavaScript. The resulting objects are then used to parse content on the server.

When a user loads the Collaboration Toolbar at the start of a DCA session, these same objects are automatically downloaded to the client where they are reconstituted into a client-side version of the DCA parser. The client-side parser is available whenever the Collaboration Toolbar is loaded -- without the Toolbar, client-side parsing cannot occur.

When Client-Side Parsing Occurs

Out-of-the box, DCA client-side parsing occurs on demand when a JavaScript document.write statement results in link creation on a client. This is enabled through a rule in the parser that reformats document.write statements (server-side) to call the client-side parser when a user loads a page. While a document.write statement is the most common scenario in which client-side parsing is desirable, if necessary you can configure client-side parsing to occur at other times as well.

See Also

For related information, see:

About Customizing the Parser

About DCA Proxying

About the Parsing Engine

About the Parsing Engine

The DCA parser is actually a parsing engine comprised of a hierarchical collection of one or more XML-based parsers. Within the hierarchy, individual parsers extend from other parsers, inheriting and overriding behavior as needed. Parsers can be targeted at content based on protocol, host name, port, document path, and content type. When the DCA server is started, these files are processed through an XML parser which combines them into a single parsing engine.

By customizing the parser, you can control the handling of all documents served through the DCA. Parser customization allows you to:

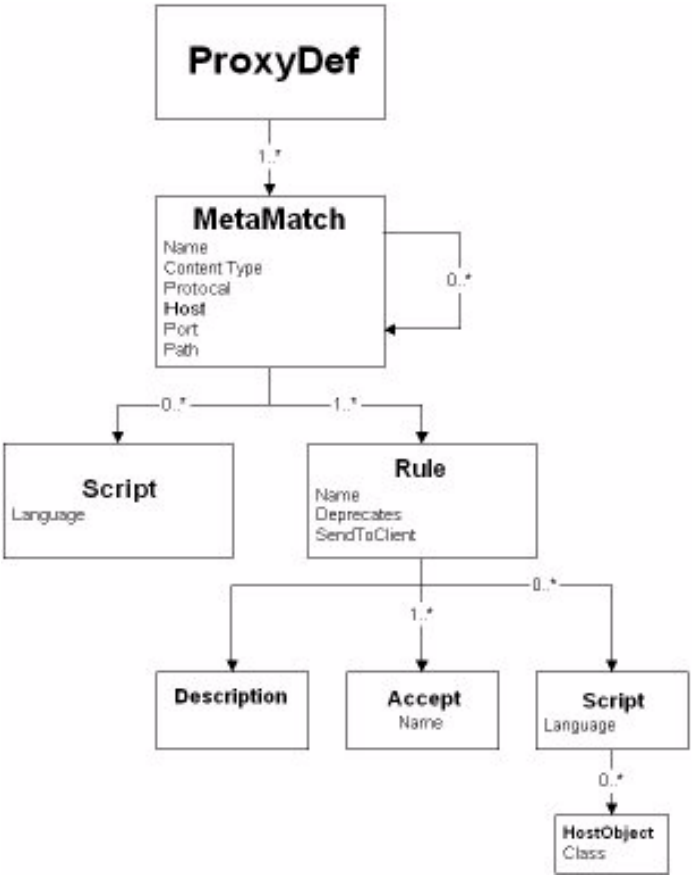
- Specify how documents are parsed;
- Specify behaviors to reformat/modify that content before it is sent back from the server.

Parser Structure

Each parser is made up of one or more rules bound into a single MetaMatch tagset. A MetaMatch, therefore, is a wrapper object that corresponds to a single parser. You can target a parser for use with a specific content type, protocol, host, etc. by setting its MetaMatch attributes.

Rules are the core elements that define a parser's behavior. Each Rule consists of one or Accept elements -- regular expressions used to match text in a parsed document, and one or more Script elements -- which define actions to be taken based on the matched text.

The parser includes a published Java API whose object classes can be used to extend the script environment.



Parser Extensibility

The DCA parser is extensible. You can create a hierarchy of parsers in multiple files that descend from one another. Descendent parsers can inherit, extend, and override behavior defined in other parsers, as necessary.

Base-Level Parsers

The DCA ships with three base-level parser files. They provide parsing for HTML and JavaScript (a parser for VBScript is not provided with the DCA but could be added, if necessary).

The DCA's base-level parser files are located at: `<DCArootdirectory>\webapp\WEB-INF\Cisco\properties`. They include:

- `proxy_rules.xml`. The DCA's root parser.
- `proxy_rules_include_js.xml` and `proxy_rules_include_text.xml`. These files inherit from `proxy_rules.xml`, and constitute two separate parsers, for JavaScript and HTML respectively.

Caution: Cisco strongly recommends that you do not edit the DCA's base-level parsers. These parsers have been carefully constructed to address a wide range of conditions and even a small change can have broad implications for the parsing process. Instead, if modification to out-of-box parsing is necessary, create and modify copies of these parsers. Or you can create descendant parsers that modify behavior defined in the base parsers as necessary.

Creating Additional Parsers

You can create as many additional parsers as you need. Generally, the farther a parser is extended the more specialized it should become (to address the needs of particular situations). While it is possible to include multiple parsers in a single XML file, for easier organization Cisco recommends following a standard of one parser per file.

Targeting Parsers

Typically you will want to have different parsers for different types of Web content. The base-level parsers that ship with the DCA provide parsing for text/HTML and JavaScript. In addition to content type, parsers can be targeted for use with specific protocols, hosts, ports, and document paths.

Creating and Storing Parser Files

While you can include multiple parsers in a single file, for easier editing and organization it's generally a good idea that each parser be stored in its own XML file. Parser files can be stored anywhere on your DCA server, although its recommended that you store them either in the same directory as the base-level parsers, or in a subdirectory.

The ProxyDef Element

The outermost element in the DCA parser is ProxyDef -- the proxy definition element. It resides in the root parser file (`proxy_rules.xml`) and can contain MetaMatch elements only. The proxy definition element corresponds directly with one RuleEngine object in Java. The RuleEngine is created and then provided by the RuleEngineFactory class. You can store named RuleEngines that are loaded from different files and they will coexist in storage via the RuleEngineFactory.

Parser File Syntax

DCA parser files are specified in XML. XML is a well-formed language with exacting syntax. When working with parser files, be sure that you:

- **Close tags properly**

  XML requires Open and Close tags for all elements. Failure to close a tag in an XML file will prevent the file from parsing properly. If necessary, examine the base-level parser files that ship with the DCA for examples.

- **Wrap Accept and Script values in a CDATA tag**

  XML includes an element (CDATA) that allows you to pass what would otherwise be considered illegal characters through the XML parser. To prevent errors, all Accept and Script values should be wrapped in a CDATA tag, as shown in this example:

```
<ACCEPT name="matchHREFs" subgroups="1,3">

    <![CDATA[<A\s([^><]*?)HREF\s*=\s*'\s*([^']*?)\s*'([^><]*?)>]]>

</ACCEPT>
```

See Also

For related information, see:

About the MetaMatch Element

A MetaMatch is a wrapper object that corresponds to a single parser. MetaMatches are comprised of rules, which in turn define the actions and behaviors of the parser. MetaMatches can also contain their own scripts that exist independent of rules. You can target a MetaMatch for use with a specific content type, protocol, host, etc. based on an inbound request's meta data provided in the HTTP header.

MetaMatch Structure

The following example demonstrates the structure of a simple MetaMatch containing a single rule:

```
-------------------------------------------------------------------------------------


<METAMATCH name="myMetaMatch" protocol="" host="" port="" path=""
contenttype="text/plain,text/html">

<RULE name="MyRule" deprecates="HisRule">

<ACCEPT name="matchHREFs" subgroups="1,3">
    <![CDATA[<A\s([^><]*?)HREF\s*=\s*'\s*([^']*?)\s*'([^><]*?)>]]>
</ACCEPT>

<SCRIPT_ONRENDER language="javascript">
  <![CDATA[<HOSTOBJECT class="com.cisco.proxy.rules.JSTagURL" />
    var groupNum = 2;
    var newURL = new TagURL(match. getGroup(groupNum),
data.getTargetURL());
    var newStr = match.setGroupJS(groupNum,
newURL.buildCompleteLink(data));]]>
</SCRIPT_ONRENDER>

</RULE>

</METAMATCH>

-------------------------------------------------------------------------------------
```

Storing MetaMatches in Multiple Files

You can include as many MetaMatches as you want in a single XML file. However, for

easier editing and organization it's generally a better idea to create individual files for each. When you store MetaMatches in multiple files:

- Place the body of the MetaMatch (that is, every part of it EXCEPT the actual METAMATCH tagset) in its own XML file. Enclose the contents of the file in an INCLUDEGROUP tagset as shown below. Any text outside of the INCLUDEGROUP tagset will be ignored by the XML parser when it assembles the DCA parser from its respective files.

```
<INCLUDEGROUP>
 <-- rules and/or scripts-->
<INCLUDEGROUP>
```

- Insert the METAMATCH tagset into a separate, existing parser file that will serve as the parent for your MetaMatch. Then add an INCLUDE tag that points to the previous file containing the body of the MetaMatch.

```
<METAMATCH name="my_MetaMatch" contenttype="text/plain,text/html>
 <INCLUDE SRC="some_subdirectory/my_MetaMatch.xml" />
</METAMATCH>
```

  - It should not matter at which point in the parent you position your METAMATCH tags. The relative positions of individual MetaMatches become irrelevant once the individual parser files are processed (by the XML parser) into a single parsing engine.

  - You can place MetaMatch XML files anywhere on your DCA server (although a subdirectory somewhere within the DCA root is probably best). Use standard URL file syntax when specifying file paths in INCLUDE tags.

MetaMatch Attributes

Metamatches can carry these attributes:

Attribute	Description	Required?	Example
name	The name of the MetaMatch. Must be unique.	Yes	<METAMATCH name="myMetaMatch" >
contenttype	Document content types(s) to be targeted by this parser. Comma-separated.	No	<METAMATCH name="myMetaMatch" contenttype="text/plain,text/html">
host	Content server host(s) to be targeted by this parser. Comma-separated.	No	<METAMATCH name="myMetaMatch" host="mydomain.com", "yourdomain.com" >

Attribute	Description	Required?	Example
path	Content server path(s) used by inbound request to be targeted by this parser. Comma-separated.	No	<METAMATCH name="myMetaMatch" path=/sub-directory/sub-sub-directory/">
port	Content server port(s) to be targeted by this parser. Comma-separated.	No	<METAMATCH name="myMetaMatch" port="80" >
protocol	Web protocol(s) used by inbound request to be targeted by this parser. Comma-separated.	No	<METAMATCH name="myMetaMatch" protocol="http" >

MetaMatch Sub-Elements

You can include the following sub-elements within a MetaMatch:

Element	Description	Example
INCLUDE		<INCLUDE SRC="some_subdirectory/myMetaMatch.xml" />
METAMATCH	Metamatches can be nested within one another across files.	<METAMATCH name="myMetaMatch" contenttype="text/plain,text/html"> <INCLUDE SRC="some_subdirectory/myMetaMatch.xml" /> </METAMATCH>
RULE	Defines text in the parsed document to be matched and consequent actions or behaviors.	<RULE> rule contents </RULE>
SCRIPT	Defines a script used by the MetaMatch. Once loaded, the script is compiled when the RuleEngine initializes. • JavaScript is the only supported script language. • Enclose in a CDATA tag to prevent it from being parsed as XML. • Maximum of 1 script per MetaMatch Script event.	<SCRIPT_ONAFTERRENDER language="javascript" sendtoclient="false"> script contents </SCRIPT_ONAFTERRENDER>

MetaMatch Script Events

Metamatches can contain their own scripts independent of rules. You can use these

Script events within a MetaMatch:

Script Event	Description	Example
OnBeforeParse	Occurs immediately prior to parsing. Allows modification to document about to be parsed.	<SCRIPT_ONBEFOREPARSE language=JavaScript>
OnAfterParse	Occurs after parsing but prior to rendering. Allows modification to parsed document object.	<SCRIPT_ONAFTERPARSE language=JavaScript>
OnBeforeRender	Occurs after parsing but prior to rendering. Similar to OnAfterParse, but per-request, so it can use requesting user attributes.	<SCRIPT_ONBEFORERENDER language=JavaScript>
OnAfterRender	Occurs after rendering. Allows one-time insertion of text into outgoing document.	<SCRIPT_ONAFTERRENDER language=JavaScript>

A MetaMatch can contain a maximum of one script per event type.

See Also

For related information, see:

About the Parsing Engine
About the Rule Element
About Parser Scripts
About the Parser Script Environment
About MetaMatch Scripts

About the Rule Element

A rule is the object in which you define:

A. What text a parser should look for in documents it parses, and...

B. What to do if and when it finds it.

For example, you might create a rule that tells the parser to search pages for text that causes links to launch a new browser window (e.g., "target=_blank") and remove it when found.

A rule consists of one or more regular expressions (used to match text in a parsed document) and scripts (used to define consequent actions or behaviors). You can create different rules for text structures with dissimilar reformatting requirements (for example, HREFs and document.writes). Conversely, you can create a rule that matches an array of text structures whose reformatting needs are similar.

You can specify that a rule deprecates other rules defined elsewhere in the parser. You can also specify whether or not a rule should be available for use in the client-side parser.

Rule Composition

Most rules consists of two basic parts:

- One or more Accept elements: An Accept element contains a regular expression that you use to match text strings in a parsed page. This matched text can then be modified or acted upon in some way, as specified in the Script element. Accept statements must be specified in Perl5 syntax.
- One or more Script elements: A script element defines behaviors to be used when rendering a parsed document, including any action to be taken on text matched by the Accept element. JavaScript is the only script language supported by the DCA.

In addition to Accept and Script elements, rules can also contain a Description element, as explained in the Rule Elements table below.

There are no required parts to a Rule. For example, in theory, you could create a rule that has no Accept Script element. In practice, however, without an Accept element, a rule Script would have no match to act upon.

Rule Example

The following example shows a simple rule containing a single Accept statement. In the example note how:

- The rule (named MyRule) deprecates another rule named HisRule.
- The Accept and Script elements are both wrapped in CDTATA tags (to prevent an XML parsing error).

```
----------------------------------------------------------------------------------------

<RULE name="MyRule" deprecates="HisRule">

<ACCEPT name="matchHREFs" subgroups="1,3">
    <![CDATA[<A\s([^><]*?)HREF\s*=\s*'\s*([^']*?)\s*'([^><]*?)>]]>
</ACCEPT>

<SCRIPT_ONRENDER language="javascript">
  <![CDATA[<HOSTOBJECT class="com.cisco.proxy.rules.JSTagURL" />
    var groupNum = 2;
    var newURL = new TagURL(match. getGroup(groupNum),
data.getTargetURL());
    var newStr = match.setGroupJS(groupNum,
newURL.buildCompleteLink(data));]]>
</SCRIPT_ONRENDER>

</RULE>

----------------------------------------------------------------------------------------
```

Rule Attributes

Rules can carry these attributes:

Attribute	Description	Required?	Example
name	The name of the Rule. Must be unique.	Yes	<RULE name="myrule">
deprecates	Rule(s) to exclude from the tree this Rule exists in. Allows you to override Rules inherited from a parent MetaMatch. Comma separated.	No	<RULE name="myrule" deprecates="hisrule, herrule">

Attribute	Description	Required?	Example
sendtoclient	Specifies whether the Rule should be available for use in client-side parsing.	No	<RULE name="myrule" sendtoclient="FALSE">

Rule Sub-Elements

You can include the following sub-elements within a rule:

Element	Description	Example
DESCRIPTION	Internal text used for the Rule object's description. • Primarily for administration or debugging. • Maximum of 1 per rule.	<DESCRIPTION> A brief description of the rule. </DESCRIPTION>
ACCEPT	Perl5 regular expression used to match text to be acted upon (e.g., a link to be reformatted) in the parsed document. • Enclose in a CDATA tag to prevent it from being parsed as XML. • Create subgroups as desired.	<ACCEPT name="matchHREFs" subgroups="1,3"> <![CDATA[<A\s([^><]*?)HREF\s*=\s*'\s*([^']*?)\s*'([^><]*?)>]]> </ACCEPT>

SCRIPT	Defines a script used by the Rule. Once loaded, the script is compiled when the RuleEngine initializes.
	• JavaScript is the only supported script language.
	• Enclose in a CDATA tag to prevent it from being parsed as XML.
	• Maximum of 1 script per rule Script event.
		<SCRIPT_ONRENDER language="javascript">
		<![CDATA[<HOSTOBJECT class="com.cisco.proxy.rules.JSTagURL" />
		var groupNum = 2;
		var newURL = new TagURL(match.getGroup(groupNum), data.getTargetURL());
		var newStr = match.setGroupJS(groupNum, newURL.buildCompleteLink(data));]]>
		</SCRIPT_ONRENDER>

Rule Script Events

You can use these Script events within a rule:

Script Event	Description	Example

Script Event	Description	Example
On_Match	The script executes immediately each time a valid match for the rule's Accept statement is found.	<SCRIPT_ON_MATCH language=JavaScript>
On_Render	If a valid match for the rule's Accept statement is found, the Script executes when the page is rendered.	<SCRIPT_ON_RENDER language=JavaScript>

A rule can contain a maximum of one script per event type.

- The `On_Match` event is called when a match is found as a document is being decomposed, and can be used to alter the parsing process from that point forward. Because a page is decomposed only once, the `On_Match` event is the best place to call scripts used to modify matched text for all users (for example, to insert additional text or code on a page).
- The `On_Render` event is called when a document is being reassembled to send to a client. Because a page may be rendered multiple times (e.g., for different users), it's usually best to limit use of the `On_Render` event to scripts used for output reformatting (for example, to modify HREFs to point to the DCA).

See Also

For related information, see:

About the Parsing Engine
About the Rule Accept Element
About Parser Scripts
About the Parser Script Environment
About Extending the Script Environment
About Rule Scripts

About the Accept Element

In a Rule, you specify document text to be matched in one or more Accept elements. An Accept element contains a regular expression that you use to match text strings in a parsed page. This matched text can then be modified or acted upon in some way, as specified in the Script element. Accept statements must be specified in Perl5 syntax.

```
<ACCEPT name="matchHREFs" subgroups="1,3">
    <![CDATA[<A\s([^><]*?)HREF\s*=\s*'\s*([^']*?)\s*'([^><]*?)>]]>
</ACCEPT>
```

Accept Attributes

You can use the following attributes with the Accept element:

Attribute:	Description:	Required?	Example:
name	The name of the Accept statement. Must be unique.	Yes	<ACCEPT name="myaccept">
subgroups	Comma-delimited list of regular expressions that will be recursively parsed. Subgroups appearing in an Accept statement are numbered left to right. Specify subgroups: <initialgroupnumber>, <numberof subgroups>	No	<ACCEPT name="myaccept" subgroups="1,3">

Writing Accept Statements

In writing Accept statements, care must be taken to create rules that only target the expected text, no more, no less. This can be very difficult to do, and is the most time-consuming aspect of parser customization.

When writing expressions for an Accept element:

You specify an Accept statement as a Perl5 regular expression. Expressions must be syntactically correct, or the rule will not be included in parser. One online resource for Perl5 syntax is at: http://www.mit.edu:8001/perl/perlre.html.

There is no limit on the number of Accept elements you can include in a single rule. Pieces of document text matched through different Accept statements can be modified by a single script in the rule.

Expressions must be syntactically correct or the rule will not be included in the parser.

Wrap expressions in CDATA tags to avoid XML parsing errors. For example: `<![CDATA[<my_expression>]]>`.

Using Subgroups in Accept Statements

Your can include subgroups within the regular expressions in your Accept statements. These subgroups, numbered right to left, can be reparsed, allowing the same text to be recursively parsed multiple times.

For example, consider FORMs. Each FORM includes an open and close tag and an action that consists of a link. This link must be parsed to point to the DCA. At the same time, within a form there may be additional HREFs different in structure than the FORM action, but which also require parsing. Recursive parsing could be used in this instance to search for and modify a FORM action link on the initial pass, and then additional links within the form tag on subsequent passes.

When creating subgroups:

- Within your Accept statement, enclose subgroups in parentheses, as shown earlier in the Accept example.
- In the ACCEPT tag, include a subgroup attribute that specifies the number to assign to the first subgroup, and the total number of subgroups in the expression. For example: `<ACCEPT name="myaccept" subgroups="1,3">`.

See Also

For related information, see:

About the Parsing Engine
About the Rule Element
About Parser Scripts

About Parser Scripts

The most common use of scripts in the DCA parser is to modify document text matched by rule Accept statements -- namely, to reformat link text to point to the DCA. But you can use scripts within the parser to modify documents any way you choose -- including to add content or behaviors, and to customize pages for specific users or contexts.

Scripts can be called from both the Rule and MetaMatch elements. Each of these elements supports its own set of script events. This allows you to call a script at virtually any point in the parsing process, and to apply it to either a particular piece of matched text, or to a document as a whole.

Parser scripts execute in a hierarchical script environment. You can extend the script environment to provide additional functionality by adding Host objects -- Java classes whose public methods then become available to your scripts. JavaScript is only the scripting language supported in the parser.

See Also

For related information, see:

About the Parsing Engine
About the Rule Element
About the Parser Script Environment
About Rule Scripts
About MetaMatch Scripts
The DCA Parser API Guide

About the Parser Script Environment

DCA parser scripts execute in a script environment. Just as parsers can extend from one another, parser scripts environments can also extend from one another. Thus, you can set properties in top-level script environments that propagate to child environments.

Each time a DCA user submits a new document request, a new script environment is created to handle that request. This new environment is a child of the parser's script environment. Changes made in that environment pertain to that environment only -- they do not propagate back to the parent and do not affect future requests.

You can set properties that are server-wide and available for every request (for properties that must be used repeatedly). Or you can set properties that are only available to a specific request (for those that are only useful in a certain contexts).

Server and Client Script Environments

On the DCA server, parser scripts are executed in a shared script environment that resembles the environment in a browser. On a client, the browser itself provides the JavaScript environment.

Because the DCA script environment is shared, variables set in scripts are globally visible. Script properties can also be made global which will make them available to scripts executed both on the server and the client.

You can set global script properties using the Java RuleEngine and ScriptEnvironment objects. To set a global script property:

1. From the current RuleEngine, call getScriptEnvironment() to get the ScriptEnvironment. The returned object is the shared environment used for the scripts making up the RuleEngine.

2. Call defineProperty() on the environment object.

As desired, you can also define properties so that they are not visible in the client. To hide a property on the client:

3. From the current RuleEngine, call getScriptEnvironment() to get the ScriptEnvironment. The returned object is the shared environment used for the scripts making up the RuleEngine.

4. Call defineProperty() on the environment object.

5. Provide a JSScriptObject.DONTENUM attributes value.

Script Execution on the Server and Client

When creating scripts, make certain that they function correctly both on the server and in client browsers. Primarily, this requires that any host objects be present in the client.

Some scripts may only be appropriate to execute on the server. For example, a script that changes the document character set is only necessary on the server, since the client will already automatically pick up any character set changes.

If necessary, you can specify that certain scripts not be sent to the client. To prevent a script from going to the client, set its `sendtoclient` attribute to FALSE.

Script Environment Properties

Two special tags, Script_Prop and Script_Init, allow you to define script properties and to execute script in environments as they are created.

SCRIPT_PROP

Use Script_Prop to define a property by an executed script fragment. Script_Prop can be used within the ProxyDef element and in MetaMatch elements. You can use one instance per element.

Example: `<SCRIPT_PROP name="DCA_CONTENT_FRAME" value="'content'"/>`

SCRIPT_INIT

Use Script_Init to execute a script that defines functions or properties for use in other scripts. Script_Init can be used within the ProxyDef element and in MetaMatch elements. You can use one instance per element.

Example:

```
<SCRIPT_INIT><![CDATA[
function myfunction()
{function definition}
]]></SCRIPT_INIT>
```

See Also

For related information, see:

About the Parsing Engine
About Parser Scripts
About Extending the Script Environment
The DCA Parser API Guide

About Extending the Script Environment

The default script environment on the DCA server does not necessarily include all of the objects useful for script writing. Other objects must be made available to scripts in the script environment before their methods can be called. (One example of this is TagURL, a commonly used object that helps reformat partial URLs based on request context.)

You can extend the script environment to provide additional functionality by adding Host objects on the server side -- Java classes whose public methods then become available to your scripts.

Using The HostObject Tag

You make Java objects available to scripts using the HostObject tag. The HostObject tag has a single attribute -- Class -- whose value is the fully qualified class name of the Java object providing the JavaScript functions. It is only necessary to add a HostObject once to expose it to all scripts in a particular script environment (although adding the same HostObject multiple times will not cause adverse effects).

```
-----------------------------------------------------------------

<SCRIPT_ONRENDER language="javascript">
<HOSTOBJECT class="com.cisco.proxy.rules.JSTagURL" />
var groupNum = 2; var newURL = new TagURL(match.getGroup(groupNum),
data.getTargetURL()); match.setGroupJS(groupNum,
newURL.buildCompleteLink(data));
</SCRIPT_ONRENDER>

-----------------------------------------------------------------
```

Java objects used in the script environment should extend `org.mozilla.javascript.ScriptableObject`. Each function to be exposed to JavaScript must conform to the `jsFunction_<function name>` naming convention. For more information on these types of objects, see the Rhino documentation. One source for Rhino documentation can be found at: http://www.mozilla.org/rhino/

The functionality provided by script host objects on the server can be provided by JavaScript objects in the client. Equivalent JavaScript objects provided with the DCA out-of-the-box are NetURL, RequestData, and TagURL.

DCA Session and Participant Objects

Script access to the DCA's Session and Participant objects is limited. You can access these objects on the server using the methods of RequestData. They are not accessible in the client. The properties of the Session and Participant objects, however, are available in the script environment on both the server and client.

The script properties DCA sets by default are described below. The `isConsoleSession`, `navigator`, and `role` values for active sessions/participants are viewable in the Admin Tool.

Property	**Description**	**Values**
isConsoleSession	Whether the session is using the DCA Collaboration Toolbar.	true, false, undefined
isMultiAgent	Whether the user is multi-session capable. Non-multi-session-capable users can only be in one session at a time.	true, false
navigator	Current User-Agent header for the session	
role	DCA user's role, as defined in the toolbar_config.xml file.	agent, caller, peer

DCA script properties objects are of the type `java.util.Properties` (or a subclass) and should be accessed through that API. They include:

- scriptProperties: Session-specific script properties. Some session properties are script-visible, some are not.
- participantProperties: Per-participant properties.
- participantSessionProperties: Participant per-session properties. Useful only to multi-session participants.
- requestScriptProperties: Additional script properties provided as request parameters. Additional properties may be specified by adding the parameter "dcaScriptVars" to a request with a value of a ':' delimited string of name/value pairs.

See Also

For related information, see:

About the Parsing Engine
About Parser Scripts
About the Parser Script Environment

The DCA Parser API Guide

About Rule Scripts

The most common place to employ scripts are within a Rule element. Rule scripts are applied based on text matched by a rule's Accept statement. The standard behavior of a rule script is to take the parsed match provided in match, change the value of a subgroup, then set that value again. This operation is repeated many times during a single document reformatting to produce an entirely new document.

The scripts provided with the standard DCA installation are a good place to start when looking for Rule script examples.

Rule Script Events

Each rule has two events for which scripts can be written. A rule can contain a maximum of one script per event type.

- **ON_MATCH**: Scripts written for the `On_Match` event are called when a rule is initially matched during document parsing.

  The `On_Match` event is usually the best place to call scripts that modify matched text. Scripts called from `On_Match` can be used to alter parsing process from that point forward.

- **ON_RENDER**: Scripts written for the `On_Render` event are called when a document is being rendered to be sent to a client.

  Generally, scripts called from the `On_Render` event should be limited to output formatting (including modification of links to point to the DCA). `On_Render` is also the appropriate place to call scripts that format a document differently for different users (for example, one that formats a page differently depending on whether the requester is a caller or agent).

The Match and Data Objects

The script blocks associated with rules in the DCA parser are similar in nature to JavaScript functions (and in the client, they are exactly that). Each function is called with two arguments, `Data` and `Match`. `Data` and `Match` are Java objects put into the script environment and are available to scripts at runtime. They are essential to any rule script functionality.

- The **`Data`** object provides request information to the script, such as session ID

and the URL of the requested page. It can be used to look up almost every property related to DCA configuration and request information.

The `Data` object's Java class is different on the server and client: `com.cisco.proxy.rules.RequestData` (on server); `com.cisco.proxy.rules.LightweightRequestData` (on client).

- The **`Match`** object represents the text matched by an Accept statement during parsing. It is used to extract matched text and, as necessary, change the contents of the match. When rendered in the final output, this produces a different string than what was in the original document.

  The `Match` object's Java class is `com.cisco.proxy.rules.ParsedMatch`.

Other Script Objects

In addition to those described in the previous section, several other objects are provided for convenience in the script environment. These include:

- `TagURL` provides help for handling URL related formatting.
- On the server, `alert()` provides for output to the Collaboration Toolbar.

The interfaces for these and additional objects are described in the *DCA Parser API Specification*.

See Also

For related information, see:

About the Parsing Engine
About the Rule Element
The DCA Parser API Guide

About MetaMatch Scripts

In addition to Rule scripts, you can also include scripts within a MetaMatch element. Scripts specified in a MetaMatch are independent of rules and have their own unique script environment. Rather than being applied based on text matched by an Accept statement, MetaMatch scripts are applied to the entire document.

MetaMatch Script Events

You can create scripts for the following events within a MetaMatch. There is a maximum of one script per event type.

Script Event	Description	Example
OnBeforeParse	Occurs immediately prior to parsing. Allows modification to document about to be parsed.	<SCRIPT_ONBEFOREPARSE language=JavaScript>
OnAfterParse	Occurs after parsing but prior to rendering. Allows modification to parsed document object.	<SCRIPT_ONAFTERPARSE language=JavaScript>
OnBeforeRender	Occurs after parsing but prior to rendering. Similar to OnAfterParse, but per-request, so it can use requesting user attributes.	<SCRIPT_ONBEFORERENDER language=JavaScript>
OnAfterRender	Occurs after rendering. Allows one-time insertion of text into outgoing document.	<SCRIPT_ONAFTERRENDER language=JavaScript>

See Also

For related information, see:

About the Parsing Engine

About the MetaMatch Element

About the Parser Script Environment

The DCA Parser API Guide

About Parser Troubleshooting

The following table describes some basic symptoms of incorrect parsing and possible root causes. Suggestions for troubleshooting the parser are described in the sections that follow.

Symptom	Likely Cause	Solution
Page links not modified correctly to point to the DCA.	Confirm that links are specified in (or ultimately result in) HTML. - if Yes - Confirm the page is being served via the DCA. - if Yes - Parser as specified is not finding links.	Modify or correct errors in the parser, as necessary.
JavaScript error on page when the page is viewed in DCA.	Confirm that the error only occurs when the page is viewed in the DCA. If so, the cause is most likely an error in the parser.	Modify or correct errors in the parser, as necessary.
Page "breaks out" of Collaboration Toolbar Frame.	Frameproofed page not handled correctly by parser.	Remove frameproofing from page, either by parser modification or changes to page.
Parser script fails to run.	If runtime error occurs when viewing page, error in the script. - or - If no runtime error occurs, the Rule or MetaMatch in which the script appears possibly failed to compile.	Correct script or parser element error, as necessary.
Links not parsed in any of the following file types: Flash, PDFs, applets, or other non-text files.	Not an error -- the DCA parses links defined in HTML and JavaScript only. L	None
Weird characters displayed on page (e.g., ???).	Parser encoding not properly set up for the content being served.	Localize the parser's encoding configuration, as necessary.
Improperly formatted text appears in the Collaboration Toolbar Remote Control participant list, or in messages displayed by the toolbar.	Toolbar encoding not properly set up for your locale.	Localize the Toolbar's encoding configuration, as necessary.

Parser Troubleshooting

Parser errors generally manifest themselves in one of two ways: an expected behavior (e.g., link parsing, script execution, etc.) fails to occur; or a JavaScript error occurs when a page is viewed through the DCA.

When troubleshooting parser errors, you may need to

- Confirm that the error only occurs when the page is viewed through the DCA.
- Check the DCA log file for errors compiling Rules or Rule Accept elements. Rules and Accept statements that fail to compile are ignored by the DCA parser at runtime.
- Check log file log for exceptions (scripting errors, server bugs, etc.)
- Verify that the regular expressions specified in your Accept statements are properly formatted to select the document text they are intended to select.
- Compare the parsed page's HTML to its original (unparsed) HTML.

If and when improperly parsed text is identified, consider whether it would best be covered under a modified version of an existing Rule, or whether a new Rule altogether should be created.

Caution: Cisco strongly recommends that you do not edit the DCA's base-level parsers. These parsers have been carefully constructed to address a wide range of conditions and even a small change can have broad implications for the parsing process. Instead, if modification to out-of-box parsing is necessary, create and modify copies of these parsers. Or you can create descendant parsers that modify behavior defined in the base parsers as necessary.

Parser Compile Errors

Individual DCA parser XML specification files are compiled into a single Rule Engine when the DCA server is started. At that time, any errors that prevent individual parsers from compiling are recorded as stack trace exceptions in the DCA log. Individual parsers that fail to compile are thereafter ignored by the DCA during runtime. Thus, if defined parser behavior does not appear to be occurring, the first step is to check the DCA log file that contains the latest DCA startup information.

DCA parser files are specified in XML. XML is a well-formed language with exacting syntax. When working with parser files, be sure that you:

- Close tags properly: XML requires Open and Close tags for all elements. Failure to close a tag in an XML file will prevent the file from parsing properly.

- Wrap Accept and Script values in CDATA tags: To pass what would otherwise be considered illegal characters though the XML parser, wrap Accept and Script values in CDATA tags.

Script Errors

Parser scripts that contain illegal characters, incorrect syntax, or other errors will, in all likelihood, still compile correctly assuming that the Rule or MetaMatch element in which they appear is otherwise syntactically correct. However, there is a good probability that the script will cause runtime errors on pages viewed through the DCA.

Script errors are most easily debugged in the client. The Netscape JS console (javascript:) is a good utility for debugging.

To debug script errors in the client:

1. Enable debugging in the client-side parser applet by opening the `parser.jsp` file located at `<DCArootdirectory>\webapp\client` and adding the following parameter to the applet: `<param Debug="true">`

2. Reload the Collaboration Toolbar, open the page on which errors occurred, and examine messages in the Java console.

See Also

For related information, see:

About the Parsing Engine
About the MetaMatch Element
About the Rule Element
About Parser Scripts
About Localizing the DCA Parser
About DCA Logs
Cisco Support for the DCA
DCA 2.01 Release Notes

Section VII. SSL Configuration

About Using SSL with the DCA

To share secure content through the DCA, the Collaboration Toolbar must be loaded via HTTPS (that is, via SSL over HTTP). This in turn causes all content requested through the DCA to be delivered using HTTPS.

The property that determines how the Toolbar is loaded resides on the Collaboration Server. It has only two settings: one that causes the Toolbar to load always in secure (HTTPS) mode; and one that causes it to load always in non-secure-(HTTP) mode. Thus, a particular DCA/CCS server combination is either full-time SSL or full-time non-SSL.

In a full-time DCA SSL configuration, all sessions are conducted in SSL mode regardless of whether the actual pages requested are secure. Furthermore, to prevent security warnings, all page resources (including images and other non-text files) are requested and returned through the DCA (normally, non-text files are requested and delivered to users directly from the Web content server). To speed performance, in SSL mode, non-text files are stored and returned to users from the DCA's Static cache.

Configuring the DCA to Use SSL

Configuring the DCA to share secure content using SSL is relatively simple. It includes these tasks:

- **Step 1: Configure the DCA's Web server software for SSL**

  Configure your DCA's Web server software by installing an appropriate server certificate and configuring SSL, as desired.

- **Step 2: Configure the DCA Collaboration Toolbar for SSL**

  Configure the DCA Collaboration Toolbar by setting the value of the `wlServer.dca.DCAPROTOCOL` property in the `wlServer.properties` file to HTTPS. The `wlServer.properties` file is located on your CCS server.

Step 3: Upload / Maintain CA Root Certificates

Your DCA server requires a range of CA root certificates to verify secure server certificates sent to the DCA during secure sessions. The DCA installation automatically includes root certificates from most well-known Certificate Authorities. As necessary, you can upload additional certificates, as well as delete expired, invalid, or unwanted certificates.

Step 4: Specify "Trusted" Hosts

The DCA allows you to specify hosts that it should "trust" over an SSL connection, even if the host a certificate that is unsigned, expired, or otherwise invalid.

Note: The communications link between the DCA and CCS used by the Agent Reporting and Management (ARM) service can also be configured to use SSL. This ensures that data sent from the DCA to CCS are encrypted. For more information, see the *DCA Installation and Integration Guide*.

See Also

For related information, see:

The DCA Installation and Integration Guide
How to Configure the Collaboration Toolbar to Use SSL
About SSL and DCA Performance
About SSL Versions
How to Maintain Root Certificates
How to Maintain Trusted Hosts

How to Configure the Collaboration Toolbar for SSL

To share secure content through the DCA, you must configure the Collaboration Toolbar to use SSL. Failure to do this will cause users to receive a denial of service error when they attempt to access secure content through the toolbar.

Note: Configuring the Collaboration Toolbar to use SSL takes place on your Collaboration server, not the DCA.

To configure the Collaboration Toolbar for SSL

To configure the toolbar to use SSL:

1. On your Collaboration Server, open the `wlServer.properties` file in a text editor. `wlServer.properties` is located at: `<CCSrootdirectory>\servlet\properties`.
2. Set the value of `wlServer.dca.DCAPROTOCOL` to `https`.
3. Save the file.
4. Restart the Web server on your Collaboration Server.

See Also

For related information, see:

About Using SSL with the DCA

About SSL Versions

By default, the DCA concurrently supports Web servers running SSL versions 2 and 3. Some older Web servers running SSL 2.01 may have difficulty handling this SSL negotiation. When a problem occurs, the symptoms include:

- Users encounter server errors when attempting to access pages from the SSL server;
- An exception appears in the DCA log bearing the message "SSL 2 bad mac decode."

You can alleviate this problem by setting the value for the SSLVersion parameter in the `Proxy.properties` file to ssl_v2. Note however that with this setting, the DCA cannot share secure content from Web servers running SSL 3.0. A more complete resolution to the problem is to upgrade Web servers that experience this problem.

See Also

For related information, see:

How to Use the Proxy Properties File

How to Maintain Root Certificates

If you will be sharing secure content, your DCA server requires a range of root certificates from common Certificate Authorities (CAs). These are used to verify secure server certificates sent to the DCA during secure sessions.

The DCA installation automatically includes root certificates from most well-known Certificate Authorities. You can view the list of these from the Admin Tool. As necessary, you can upload additional certificates. You can also delete expired, invalid, or unwanted certificates.

To View Installed CA Certificates

To view the list of CA certificates currently on your server:

1. In the Admin Tool, select Certificates > View Certificates.
2. To view details of a particular certificate, click its name in the list of CA certificates. For each certificate, the Admin Tool lists:

   Issuer: The name of the Trusted Root Certification Authority.

   Common Name: The certificate's common (friendly) name.

   Valid Not Before: The certificate's initial date of validity.

   Valid Not After: The certificate's expiration date.

To Add a Certificate

As necessary, you can add additional certificates. To add a certificate:

3. As necessary, download (from a CA Web site) or export (from your browser) the certificate file. With the appropriate software, you can also create your own certificate file.
4. In the Admin Tool, select Certificates > Upload Certificates.
5. Browse to the certificate file. The file must be in PEM (Privacy Encrypted Mail) format.

6. Click Upload File.

To Delete a Certificate

As necessary, you can delete CA certificates that are expired, invalid, or otherwise unwanted. To delete a certificate:

1. In the Admin Tool, select Certificates > View Certificates.
2. Check the box to the left of the certificate(s) you want to delete.
3. Click Remove Checked CAs.

Caution: When you delete a certificate, it is permanently removed from your server. If you delete a certificate by mistake, you must reacquire it (for example, by downloading it from the CA's Web site) and upload it to the DCA.

See Also

For related information, see:

About Using SSL with the DCA
How to Maintain Trusted Hosts
How to Access the Admin Tool

How to Maintain Trusted Hosts

The DCA allows you to specify hosts that it should "trust" over an SSL connection, even if the host a certificate that is unsigned, expired, or otherwise invalid.

To Maintain Trusted Hosts

You maintain your list of trusted hosts in the `TrustedHosts.properties` file.

To add or remove a trusted host:

1. In the Admin Tool, select Configuration > TrustedHost.
2. To add a trusted host, enter the host's Common Name exactly as it appears in the host's certificate. This may be as a DNS entry (for example: `<hostname>.cisco.com`), or as an IP address.
3. To remove a trusted host, delete its entry in the file.
4. Click Submit.

See Also

For related information, see:

About Using SSL with the DCA
How to Access the Admin Tool
How to Maintain Root Certificates
How to Stop and Start the DCA

How to Use the TrustedHosts Properties File

Use the `TrustedHosts.properties` file to specify trusted hosts -- that is, hosts that the DCA should "trust" over an SSL connection, even if the host's certificate is unsigned, expired, or otherwise invalid.

To Add or Delete a Trusted Host

To add or delete a trusted host:

1. In the Admin Tool, select Configuration > TrustedHost.
2. To add a trusted host, enter the host's Common Name exactly as it appears in the host's certificate. This may be as a DNS entry (for example: `<hostname>.cisco.com`), or as an IP address.
3. To remove a trusted host, delete its entry in the file.
4. Click Submit.

Note: If you have removed a trusted host, you must also restart the DCA.

See Also

For related information, see:

How to Maintain Trusted Hosts
About Using SSL with the DCA
How to Access the Admin Tool

Section VIII. Server Security Configuration

About DCA Server Security

The DCA includes the following mechanisms to allow you to secure access your DCA server:

DCA Server Security / Firewall Support

DCA Ports and Firewalls

The server on which you install your DCA is typically deployed in a demilitarized zone (DMZ) outside your corporate firewall. For security purposes, you may want to close server ports unused by the DCA.

If you are deploying the DCA behind a firewall, you must enable access to the ports used by the DCA. By default, the DCA uses the following ports:

- HTTP Port: 80/TCP (default)
- HTTPS Port: 443/TCP (default)

In a typical deployment, your DCA and Collaboration servers are located within the same DMZ. However, if a firewall will stand between the DCA and CCS, you must also enable the following ports on that firewall for the DCA-CCS connection:

- RMI Registry Port: 1099/TCP

  The port the DCA uses to register a connection instance with CCS.

- RMI Connection Port: TCP (no default, defined when you create the DCA-CCS connection)

  The port CCS uses to connect to and communicate with the DCA.

For more information on the DCA-CCS connection, see the *DCA Installation and Integration Guide*.

DCA-CCS Connection

The DCA-CCS connection establishes a communications link between the DCA and CCS for the Agent Reporting and Management (ARM) service. This communication:

- Makes information on Web content shared during DCA sessions available for CCS reports.

- Allows the automatic cleanup of DCA sessions when their associated CCS sessions are terminated.

If desired, the DCA-CCS connection can be configured to use SSL so that data sent from the DCA to CCS are encrypted. For more information, see the *DCA Installation and Integration Guide*.

IIS Lockdown Tool

On Windows 2000 platforms, you may choose to run the IIS Lockdown Tool on your DCA/IIS server. The IIS Lockdown Tool is a wizard provided by Microsoft that tightens security be shutting down potential avenues of entry on your IIS server.

Caution: There are specific Lockdown Tool settings that must be used for the DCA. Failure to use these settings can prevent the DCA from functioning properly. Read About the IIS Lockdown Tool for more information.

DCA Session Security

Each DCA collaboration session is assigned a unique session ID, and each session participant receives a participant cookie. To access pages stored in the DCA cache, users must present the correct combination of a URL and DCA session ID, or URL and participant cookie. In addition, pages that are stored in the DCA cache are automatically removed from the cache, based upon a configurable time-out period. All of these measures are in place to ensure that only authorized session participants can access pages stored in the DCA.

Admin Tool Security

Because it allows you to perform system configuration and management functions, access to the DCA Admin Tool should be limited to appropriate individuals. You can restrict access to the Admin Tool based on:

- Username and password

- One or more specific IP addresses

See Also

For related information, see:

How to Change the Admin Tool Username and Password
How to Limit Admin Tool Access
About the IIS Lockdown Tool
About DCA Sessions
DCA Installation and Integration Guide.

About the IIS Lockdown Tool

On Windows 2000 platforms, you may choose to run the IIS Lockdown Tool on your DCA/IIS server. The IIS Lockdown Tool is a wizard that reduces IIS's vulnerability to virus attack by turning off unnecessary IIS features. It is available at: http://www.microsoft.com/windows2000/downloads/recommended/iislockdown/default.asp.

Running the IIS Lockdown Tool on Your DCA Server

The IIS Lockdown Tool should be run only AFTER the DCA has been installed on your server. If the Lockdown Tool has already been run, you must uninstall it prior to installing the DCA. Failure to first remove the Lockdown Tool will prevent the DCA from installing properly.

After the DCA is installed, you can reinstall the Lockdown Tool using the settings described below.

Note: The information in this guide is based on the Lockdown Tool, version 2.1. For information on subsequent versions, consult your Cisco representative.

Detecting a Previous Lockdown Tool Installation

To detect whether the Lockdown Tool was previously run on your server, check the `<Windowsrootdirectory>\System32\inetsrv` directory for the presence of files named `oblt-rep.log` and/or `oblt-log.log`. The presence of either of these files indicates that the Lockdown Tool has been run.

Uninstalling the Lockdown Tool

If necessary, uninstall the Lockdown Tool by running it a second time. Running the Lockdown Tool wizard over a current installation causes it to uninstall. When run over a current installation, the wizard displays a message asking you to confirm the uninstall. If instead of this message the wizard displays a license agreement, the previous installation has already been removed.

Installing the Lockdown Tool - DCA Settings

When you install the Lockdown Tool, you are prompted to select a template that

most closely matches the role of your server. Select the PROXY SERVER template. This template automatically uses the settings required by the DCA. Specifically, these include:

- Services: Web Service (HTTP) enabled.
- File Permissions for Anonymous Users: Writing to content directories enabled.
- Virtual Directories: Scripts directory enabled.
- URL Scan Filter: Not installed.

See Also

For related information, see:

About Admin Tool and DCA Server Security
The DCA Installation and Integration Guide

Section IX. Collaboration Toolbar Configuration

About the DCA Collaboration Toolbar

In a DCA session, agents and callers share Web content using the DCA's Collaboration Toolbar -- a simple set of sharing controls that the DCA adds to CCS. The Collaboration Toolbar loads automatically in its own browser window at the start of a session.

In most cases, the Collaboration Toolbar extends CCS's content sharing capability. It can be configured separately for agents and callers, so that each has access to a different set of controls. Typically, callers are given access to fewer content sharing features. Online Help is available for agents.

The Collaboration Toolbar Interface

The Collaboration Toolbar includes a full set of navigation and content sharing controls. These must be used in place of similar controls that normally appear in a participant's Web browser or in the Collaboration Agent and Caller desktops.

To ensure this, the DCA-CCS Updater (run during integration) automatically:

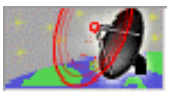
- Hides or disables CCS Agent and Caller desktop controls that conflict with the

Collaboration Toolbar.

- Configures the Agent Desktop to always use an external shared view window. It also removes that window's native browser menus and toolbars.

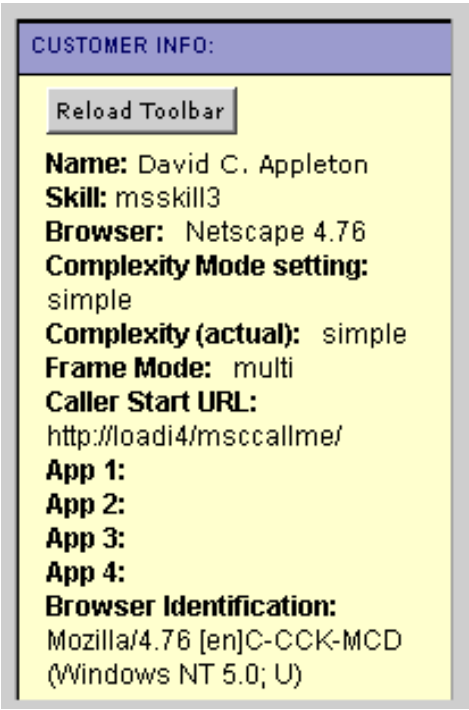
Collaboration Toolbar Controls

The Collaboration Toolbar contains the following navigation and page sharing controls. Because it is fully configurable, you can decide which controls to make available to agents and callers.

Control	Description	Icon
Back-Next buttons	Use to browse back or forward to previously viewed pages.	
Refresh button	Use to refresh the current page from the Web content server.	
Address bar	Use to enter the URL for a new page.	
Page Share button	Use to share the currently displayed page with other session participants.	
Form Share button	Use to share the data you have entered in an online form. Click the Form Share button to turn on Form Share. Form data is then shared automatically as you type it. To turn off Form Share, click the button again.	
Follow Me Button	Use to turn on/off Follow Me mode. In Follow Me mode, each page you navigate to is sent to other participants. To turn off Follow Me mode, click the Follow Me button again.	
Help button	Use to display the Collaboration Toolbar online Help.	
Remote Control	Use to access the Caller Remote Control toolset. Depending on the configuration, this may include a: • Page Share button • Form Share button • Follow Me button Clicking a button in Caller Remote Control triggers a corresponding action in a selected participant's browser. For example, clicking the Page Share button causes the caller's current page to be shared with other session participants.	
Notifier	Displays when you share a page to let you see when the page has finished loading in participant browsers.	

Loading and Unloading the Collaboration Toolbar

The Collaboration Toolbar loads automatically in its own browser window at the start of a DCA session. There are several ways that it can be accidentally unloaded (that is, closed), at which point participants will be unable to share Web content. These include:



- Single-session agents: Using the White Board or App Share features (if available) in the Agent Desktop.
- All agents and callers: Closing the Collaboration Toolbar's browser window; using navigation commands on the browser's pop-up (right mouse button) menu.

As a safeguard against accidental unloading, the DCA-CCS Updater creates a button on the Caller Information Page in the Agent Desktop agents can use to reload the toolbar. The button, labeled Reload Toolbar, reloads the toolbar for both agents and callers, reopening a new browser window if necessary.

Agents can also reload the Toolbar for themselves only by clicking the External View button in the Agent Desktop.

When the Toolbar is reloaded, it automatically displays the last shared page.

See Also

For related information, see:

About Toolbar Changes to CCS
How to Configure the Collaboration Toolbar
How to Use the Collaboration Toolbar
How to Localize the Collaboration Toolbar

About Toolbar Changes to Collaboration Server

The DCA Collaboration Toolbar does more than add a new interface through which agents and callers share Web content in Collaboration Server. It also:

- Extends CCS's content sharing capability
- Modifies the CCS Agent and Caller desktops

Toolbar Changes to CCS Content Sharing Capability

The DCA Collaboration Toolbar extends standalone CCS's content sharing capability as shown in the table below:

	CCS Single-Session Agent:		**CCS Multi-Session Agent:**	
	Alone:	**With DCA:**	**Alone:**	**With DCA:**
Automatic Form Share	No (manual Form Share only)	Yes	No	Yes
Caller Remote Control	Yes	Yes	No	Yes
Content Sharing in Popup Windows	No	Yes	No	Yes
Follow Me Browsing	Yes	Yes	No	Yes

Toolbar Changes to the CCS Caller and Agent Desktops

Integrating the DCA and the Collaboration Toolbar with CCS causes a number of changes to controls and settings in the CCS Agent and Caller desktops. These changes are described below.

In the CCS Administration desktop, the Agent Role settings that control features removed by the DCA are automatically disabled to prevent modification. For more information, see the DCA Installation and Integration Guide.

Change to CCS:	**Explanation:**
Content Sharing controls removed from Agent Desktop In DCA-integrated CCS, content sharing controls are removed from the CCS Agent Desktop.	To prevent conflicts with controls on the Collaboration Toolbar, integrating the DCA with CCS automatically removes the following controls (if present) from the Agent Desktop: • For single-session agent configurations: On the Sharing tab, the Follow Me, Form Share, Page Share, Address Bar, White Board, and App Share controls. From the In Session tab, the Set Leader button. From the Script area, the Caller Remote Control Panel. Note: The Broadcast Mode button (in the tab common area) is not removed but its behavior is deprecated-- it no longer has any effect. • For multi-session agent configurations: Page Share button.
Content Sharing controls removed from Caller Desktop In DCA-integrated CCS, content sharing controls are removed from the CCS Caller Desktop.	To prevent conflicts with controls on the Collaboration Toolbar, integrating the DCA with CCS automatically removes the following controls (if present) from the Caller Desktop: the Follow Me, Form Share, and Page Share and App Share controls.
External View Required In DCA-integrated CCS, agents must use an external shared view to display shared content.	In standalone Collaboration, agents can choose to display shared content within the CCS Agent Desktop frameset (*internal view*) or in a separate browser window (*external view*). In DCA-integrated CCS, agents always use an external shared view. While the External View button remains visible, clicking it simply reloads the Collaboration Toolbar
Caller Complexity Mode set to Simple In DCA-integrated CCS, caller complexity mode is set to Simple by default.	In standalone CCS, callers must download the CCS Caller applet in order to share complex content. Callers are presented with a security warning as the applet downloads. The DCA Collaboration Toolbar eliminates the need for the Caller applet. Therefore, during DCA-CCS integration the DCA automatically sets the CCS Caller Complexity Mode to Simple. This prevents the Caller applet from downloading to the client. One important consideration regarding Simple mode is that it prevents agents and callers from sharing non-Web content; namely, it prevents them from using the CCS White Board and App Share features. If you want agents to be able to use these features, you must manually re-enable White Board and/or App Share after integrating the DCA with CCS (for more, see *the DCA Installation and Integration Guide*).

See Also

For related information, see:

About the DCA Collaboration Toolbar

How to Configure the Collaboration Toolbar

How to Use the Collaboration Toolbar

How to Localize the Collaboration Toolbar

How to Configure the Collaboration Toolbar

The DCA Collaboration Toolbar can be configured separately for agents and caller. This allows you to specify which controls are available to each.

- *Agents* are defined as persons who access Collaboration using the Agent Control Panel.
- *Callers* are defined as persons who access Collaboration using the Caller Control Panel.

The Default Toolbar Configuration

By default, the following Collaboration Toolbar controls are available to agents and callers:

- Agents: Back, Next, Refresh, Address, Page Share, Follow Me, Form Share, Help, and Notifier.
- Callers: Back, Next, Refresh, and Address.

To Configure the Collaboration Toolbar

To configure the Collaboration Toolbar:

1. Open `toolbar_config.xml` in a text editor. It is located at: located at `<DCArootdirectory>\uiserver\WEB-INF\properties\default\dca\`. In the file, user groups for agents and callers are defined in the file as Name-Value pairs, where `Name` is the name of the group and `Value` contains a list of comma-separated controls available to that group.
2. Assign controls to agents and users as desired (a list of valid control names appears in the table below). When editing the file:
   - Be careful that each group definition remains wrapped in a Property tag. For example: `<property name="caller" value="back,next,refresh,address" />`
   - The only valid user groups are `agent` and `caller`. You *can* create a sub-group that bundles multiple controls, and then assign it to agents and/or callers (see example below).
3. Save the file.

4. Restart the DCA.

Example

Here is a sample Collaboration Toolbar configuration file. It uses a sub- group named "base" to group together several controls.

```
<properties>
<property name="base" value="back,next,refresh,address" />
<property name="agent"
value="base,pageshare,followforms,followme,remote,notifier,help" />
<property name="caller" value="base" />
</properties>
```

Toolbar Control Properties

This table lists all Collaboration Toolbar controls, their use, and their respective property names.

Control	Description	Property Name
Address Bar/Go Button	Use to enter the URL for a new page.	address
Back Button	Use to browse back to the next page.	back
Caller Remote Control	Use to display the Caller Remote Control set. **Caution:** It's recommended that Caller Remote Control NOT be made available to callers.	remote
Follow Me Button	Use to turn on/off Follow Me mode.	followme
Form Share Button	Use to turn on/off Form Share.	followforms
Help	Use to display Collaboration Toolbar Help. **Caution:** Because Collaboration Toolbar Help is written specifically for agents, it's recommended that the Help button NOT be made available to callers.	help
Message Text	Use to cause a brief confirmation message in the Toolbar each time a user executes an action, such as selecting control (e.g., "Automatic Form Share is On."). Note: Message Text does not include notification of when a page download is complete.	messages
Next Button	Use to browse forward to the next page.	next

Control	Description	Property Name
Notifier	Allows a participant to see when a sent page has finished loading in another participant's browser. **Caution:** It's recommended that Notifier NOT be made available to callers.	`notifier`
Page Share Button	Use to share the currently displayed page.	`pageshare`
Refresh Button	Use to refresh the current page from the Web content server.	`refresh`

Special Configuration Notes

This section contains additional information on configuring the Collaboration Toolbar.

Restricting Controls to Callers

It's recommended that the following controls not be made available to callers:

- Remote Control: Remote Control was designed for use strictly by agents. Giving callers access to Remote Control would allow them to trigger Web content sharing controls on Agent Desktops.
- Help: Because Collaboration Toolbar Help is written specifically for agents, it's recommended that the Help button NOT be made available to callers.
- Notifier: While it does no harm, callers are unlikely to understand the significance of the Notifier and may find it distracting.

Caller Remote Control

The following guidelines apply to configuring the Caller Remote Control feature:

- Though not enforced by the software, it is strongly recommended the Caller Remote Control be made available to agents only.
- Caller Remote Control automatically includes the same set of content sharing controls (Page Share button, Follow Me button, and/or Form Share button) that has been otherwise configured for the group. There is no manual method of specifying which controls should appear in Caller Remote Control.

See Also

For related information, see:

About the DCA Collaboration Toolbar
About Toolbar Changes to CCS
How to Use the Collaboration Toolbar
How to Localize the Collaboration Toolbar
How to Stop and Start the DCA

Section X. Localization Configuration

About Localizing the DCA

The DCA 2.01 includes the following localizable features:

- **DCA Collaboration Toolbar:** You can localize the language of button labels and messages displayed in the DCA Collaboration Toolbar. In some instances, you also may need to map the toolbar's Java to HTML encoding to support clients whose locales use multi-byte and other non-Western character sets.
- **DCA Admin Tool:** You can localize the language of menus, labels, and messages displayed in the DCA Admin Tool.
- **DCA Parser:** The DCA parser converts HTML content to a Java string, and then back to HTML. For documents encoded in non-Latin 1 character sets, you can (and should) specify which Java encoding method the parser should use based on the document's HTML encoding method. The DCA parser supports multi-byte character sets.
- **DCA Default Error Page**: The DCA ships with a default error page that displays when a user enters an invalid URL or when a Web server is inaccessible. As desired, you can modify the text of the page and/or create additional localizations.

Prepackaged Localization

Out of the box, the DCA 2.01 automatically supports Admin Tool and Collaboration Toolbar localization for the following languages. Localization for the Admin Tool and Collaboration Toolbar is delivered in the form of Java resource bundles. The appropriate resource bundle is automatically used based on the locale setting of the end-user's PC (in the case of the Collaboration Toolbar) or the DCA server (in the case of the Admin Tool).

The Admin Tool and Collaboration Toolbar resource bundles that ship with the DCA 2.01 include:

- English
  - French
  - Spanish
  - German
  - Korean
  - Chinese (Simplified)

To localize the Admin Tool and Collaboration Toolbar for additional languages, you

will need to create additional resource bundles.

Note: If improperly encoded text appears in the Collaboration Toolbar's Remote Control Participant drop-down list, or in messages displayed by the toolbar, it may indicate an additional need to localize the Collaboration Toolbar's Java to HTML encoding method.

Non-Localizable Features

The following DCA features cannot be localized in the current version:

- The DCA log files
- DCA Collaboration Toolbar Online Help
- Certain "hard-coded" labels appearing in the DCA Admin Tool interface

See Also

For related information, see:

About the DCA Collaboration Toolbar
How to Localize the Collaboration Toolbar
About the DCA Admin Tool
How to Localize the Admin Tool
About Localizing the DCA Parser
About Custom Error Pages

How to Localize the Collaboration Toolbar

Localizing the Collaboration Toolbar consists of:

- Localizing toolbar text: This process allows you to convert toolbar messages and button labels into a local language.
- Localizing the toolbar encoding method: To support clients from locales that use multi-byte and other non-Western character sets, you *may* need to map the toolbar's Java to HTML encoding method(s). The need for this configuration is indicated by improperly encoded text appearing in the Remote Control Participant drop-down list or in messages displayed by the toolbar.

Localization support is not currently available for the Collaboration Toolbar Online Help.

Localizing Collaboration Toolbar Text

Localizing the Collaboration Toolbar language consists of converting all toolbar messages and button labels into a local language. You do this by creating one or more locale-specific versions of the file in which toolbar text is stored, replacing the default (English) text in the file with an alternate language. The toolbar then references the appropriate locale-specific file based on individual users' locale settings.

Prepackaged Localization

Out of the box, the Collaboration Toolbar is automatically localized for the following languages:

- English
    - French
    - Spanish
    - German
    - Korean
    - Chinese (Simplified)

To localize the Collaboration Toolbar for additional languages, you will need to create additional resource bundles as described below.

To Create Additional Localization Resource Bundles

To create additional localization resource bundles for the Collaboration Toolbar:

1. In a text editor, open the `dca_console_controls.properties` file. This is the file in which default (U.S. English) toolbar text is stored. It is located at: `<DCArootdirectory>\uiserver\WEB-INF\properties\default\dca`.

2. Modify the file, rewriting message and button text in an alternate language, as desired. In the file:

   Message Text: Message text is the text the toolbar displays when a user selects a particular control. For example, when a user selects the Follow Me button, by default the toolbar displays the message "Follow Me is now on."

   Button Text: Most buttons have properties for both label and mouse-over text. For example, The Back button includes these default values: `back=Back` (label text), `back_mo=Go To Previous Page` (mouse-over text). Note that some buttons have no text associated with them.

3. After editing the file, save it under a locale-specific name by appending a lower-case two-letter suffix that represents the valid ISO language code (as defined by ISO-639) for the language into which you are translating. For example, to localize the file for French, rename the file: `dca_console_controls_fr.properties`.

   To specify a locale that includes both a language and country, include an upper-case two-letter ISO country code in the suffix, as defined by ISO-3166. For example, to localize the file for Belgian French, rename the file: `dca_console_controls_fr_BE.properties`.

   You can find a full list of language and country codes at a number of Web sites, including http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt and http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

4. IF you modified the file to include non-Latin-1 characters, run the file through the *native2ascii* utility (available in your `<DCArootdirectory>\jdk\bin` directory).

   The *native2ascii* utility converts files containing non-Latin-1 characters to Unicode, necessary for the DCA to read them. You can read instructions for using the *native2ascii* utility here: http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html

5. Restart the DCA.

Localizing the Collaboration Toolbar Encoding Method

To support DCA users whose locales use multi-byte and other non-Western character

sets, you *may* need to map the Collaboration Toolbar's Java to HTML encoding method(s).

The need for this configuration is indicated by the appearance of improperly encoded text in the Remote Control Participant drop-down list, or in messages displayed by the toolbar. Improperly encoded characters typically appear as question marks.

You can wait to see if the problem exists before proceeding with this configuration. If no mappings are specified, the toolbar uses your DCA server's platform default encoding method.

To Localize the Collaboration Toolbar Encoding Method

To localize the Collaboration Toolbar encoding method:

1. Open the locale_encoding.xml file located at: <DCArootdirectory>\uiserver\WEB-INF\properties\default\dca.

2. Specify one or more language to Java encoding mappings. These should be based on the language(s) appropriate to the locale settings on your users client machines. Specify mappings as follows:

   - `<property name="<language_code>" value="<Java_encoding_method>" />`

   - Example: `<property name="ja" value="SJIS" />`

   - If you want, you can specify a default language/Java encoding mapping, as follows: `<property name="default" value="<Java_encoding_method>" />` UTF8 is a good encoding method to use as a default.

   - You can find a full list of language codes at a number of Web sites, including http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt. A list of Java encoding methods is available at: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html.

3. Save the file.

4. Open the `encoding_charsets.xml` file located at: `<DCArootdirectory>\uiserver\WEB-INF\properties\default\dca.`

5. Specify one or more Java to HTML mappings. These should be based on the language to Java mappings you defined in the `locale_encoding.xml` file. Specify mappings as follows:

   - `<property name="<Java_encoding_method>" value="<HTML_encoding_method>" />`

   - Example: `<property name="SJIS" value="Shift_JIS" />`

   - You can find a list of HTML encoding methods at:

http://www.w3.org/International/O-charset.html

6. Save the file.
7. Restart the DCA.

See Also

For related information, see:

About Localizing the DCA
About the DCA Collaboration Toolbar
How to Stop and Start the DCA

How to Localize the Admin Tool

Localizing the DCA Admin Tool consists of converting the menu names field labels, and message text that appear in the Admin Tool into a local language. You do this by creating one or more locale-specific versions of the files in which Admin Tool text is stored, replacing the default (English) text in the files with an alternate language. The Admin Tool then references the appropriate locale-specific files based on the DCA server's locale settings.

Prepackaged Localization

Out of the box, the Admin Tool is automatically localized for the following languages:

- English
  - French
  - Spanish
  - German
  - Korean
  - Chinese (Simplified)

To localize the Admin Tool for additional languages, you will need to create additional resources as described below.

To Create Additional Localization Resource Bundles

Localizable Admin Tool text is stored in three different files. You localize these files by replacing their default (English) text with an alternate language and then saving them under locale-specific names.

To create additional localization resource bundles for the Admin Tool:

1. In a text editor, open the file you want to edit. All are located at: `<DCArootdirectory>\uiserver\WEB-INF\properties\default\dcaadmin`. To localize the Admin Tool's:
   - Field labels and messages, open: `dca_admin.properties`
   - Menu names, open: `admin_ui.properties`
   - Error messages resulting from a failed Admin Tool login attempt, open: `fault_strings.properties`

2. Modify the file, rewriting messages and labels in an alternate language, as desired.

3. After editing the file, save it under a locale-specific name by appending a lower-case two-letter suffix that represents the valid ISO language code (as defined by ISO-639) for the language into which you are translating. For example, to localize the `dca_admin.properties` file for French, rename it: `dca_admin_fr.properties`.

   To specify a locale that includes both a language and country, include an upper-case two-letter ISO country code in the suffix, as defined by ISO-3166. For example, to localize the file for Belgian French, rename the file: `dca_admin_fr_BE.properties`.

   You can find a full list of language and country codes at a number of Web sites, including http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt and http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

4. IF you modified the file to include non-Latin-1 characters, run the file through the *native2ascii* utility (available in your `<DCArootdirectory>\jdk\bin` directory).

   The *native2ascii* utility converts files containing non-Latin-1 characters to Unicode, necessary for the DCA to read them. You can read instructions for using the *native2ascii* utility here: http://java.sun.com/j2se/1.3/docs/tooldocs/tools.html

5. Restart the DCA.

Localization Notes

The following additional guidelines apply to Admin Tool localization:

- **Date and number formats** are determined by the locale setting on the DCA's server.

- **Logging Levels and the property file descriptions** appearing on the Admin Tool's Configuration page cannot be localized. Property file names and the contents of property files (with the exception of comment text) cannot be localized.

- **On the Configuration page, the field labels** that display when you view the `Proxy.properties, RemoteMonitor.properties`, and `UserAgent.properties` files derive from comment text in the files themselves. To localize them, you must edit the comments in the respective files.

Selecting a Non-Local Language for the Current Session

The Admin Tool login screen contains a Language drop-down list that automatically displays all localizations you have defined for the Admin Tool. As desired, users can select the localization option they want to use in their current session. Otherwise, the platform default is used, based upon the DCA server's localization setting. If localization files for the platform default have not been defined, English (US) is used.

See Also

For related information, see:

About Localizing the DCA
About the DCA Admin Tool
How to Stop and Start the DCA

About Localizing the DCA Parser

The DCA parser works by converting HTML content to a Java string, and then back to HTML. To parse a document correctly, it must use a Java encoding method appropriate to the document's HTML encoding method.

For documents that use standard Western encoding, the parser requires no encoding configuration. However, if you will be sharing content that uses other encoding methods (including multi-byte character sets), you may need to define the parser's:

- HTML to Java encoding mapping
- Default Java encoding method
- Equivalent Java encoding methods

HTML to Java Encoding Mapping

Most HTML encoding method names do not match their Java equivalents. To ensure correct parsing of documents, the DCA needs to be told which HTML and Java encodings to use with one another. You do this by mapping HTML to Java encoding methods in the `Charset.properties` file.

Default Java Encoding Methods

Most Web pages contain information in a response header or meta tag that indicates their HTML encoding method. If this information is missing, a platform default encoding method is used (in the case of the DCA parser, the platform default would be the server's locale setting).

The `CharsetDefaults.properties` file allows you to override the platform default and specify your own default Java encoding method for the DCA parser. You can specify a range of pages -- for example, those whose URL includes a particular country code. You can also specify different defaults for different ranges of pages.

Equivalent Java Encoding Methods

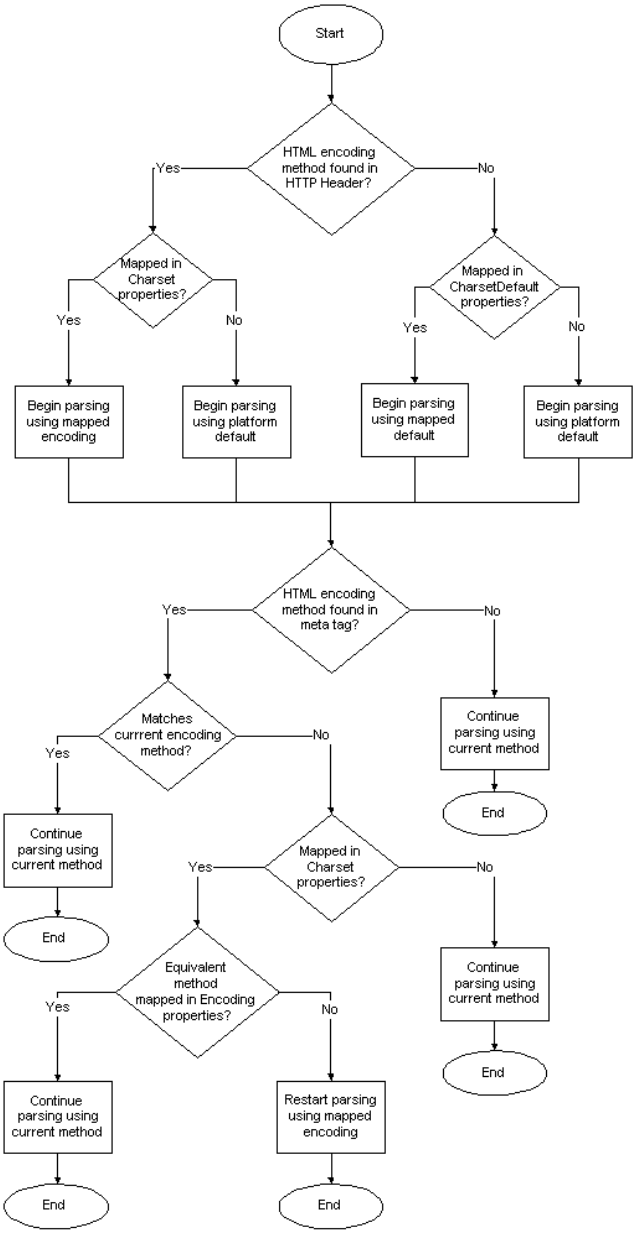
An HTML document's encoding method can be specified either in its HTTP header, or in a meta tag within the document. Because the DCA parser chooses its Java encoding method BEFORE it begins parsing a document, the later discovery of a

different encoding method specified in a document's meta tags will cause parsing to restart.

While under normal circumstances this behavior is correct, there may be situations in which it would not be preferable to restart parsing -- namely, if the new encoding method would return essentially the same output as the current. You can use the `encoding.properties` file to specify equivalent Java encoding methods that should NOT cause document parsing to restart if the mapped encoding changes.

How the DCA Chooses an Encoding Method

The diagram at right describes the process the DCA uses to determine which Java encoding method to use with a particular HTML document. As illustrated, the determination is based on the document's HTML encoding information and settings in the three properties files affecting encoding.



See Also

For related information, see:

About Localizing the DCA
How to Use the Charset Properties File
How to Use the CharsetDefault Properties File
How to Use the Encoding Properties File

How to Use the Charset Properties File

Use the `Charset.properties` file to map HTML encoding methods to their Java equivalents. This helps ensure correct parsing of documents by the DCA.

About HTML to Java Encoding Mapping

The DCA parser works by converting HTML content to a Java string, and then back to HTML. To parse a document correctly, it must use a Java encoding method appropriate to the document's HTML encoding method. However, because most HTML encoding method names do not match their Java equivalents, the DCA needs to be told which HTML and Java encodings to use with one another.

Note that this configuration is only necessary if:

- You DO NOT use standard Western encoding, or...
- You will be parsing pages whose encoding method is different than your server's platform default encoding method.

To Map HTML and Java Encoding Methods

To map an HTML encoding method to a Java encoding method:

1. In the Admin Tool, select Configuration > Charset.
2. On a new line, add an HTML to Java encoding mapping as follows:

   ```
   <HTML encoding method>=<Java encoding method>
   ```

3. Click Submit.

Note: `Charset.properties` can be edited while the DCA server is running. Restarting the server after adding a new mapping is not necessary.

To Reference HTML and Java Encoding Method Names

Lists of HTML encoding methods are available at:

http://www.w3.org/International/O-charset.html

A list of Java encoding methods is available at:

http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html

See Also

For related information, see:

About Localizing the DCA Parser
How to Use the CharsetDefault Properties File
How to Use the Encoding Properties File
How to Access the Admin Tool

How to Use the CharsetDefault Properties File

Use the `CharsetDefaults.properties` file to specify your own default Java encoding method for the DCA parser. This default encoding method will override the platform default.

About Default Encoding Methods

Most Web pages contain information in a response header or meta tag that indicates their encoding method. If this information is missing, a platform default encoding method is used (in the case of the DCA parser, the platform default would be the server's locale setting).

To Specify a Default Encoding Method

To specify a default encoding method:

1. In the Admin Tool, select Configuration > CharsetDefault.
2. On a new line, add a page to default Java encoding method mapping, as follows. You can specify a range of pages -- for example, those whose URL includes a particular country code. You can also specify different defaults for different ranges of pages:

   ```
   <Perl 5 regular expression representing one or more HTML
   documents>=<Java encoding method>
   ```

3. Click Submit.

Note: `CharsetDefaults.properties` can be modified while the DCA server is running. Restarting the server after editing the file is not necessary.

Example

In the example below, pages returned from `www.toyota.co.jp` whose encoding method is unspecified will use JISAutoDetect (Japanese) as their Java encoding method:

```
http.?://www.toyota.co.jp.*=JISAutoDetect
```

See Also

For related information, see:

About Localizing the DCA Parser
How to Use the Charset Properties File
How to Use the Encoding Properties File
How to Access the Admin Tool

How to Use the Encoding Properties File

Use the `Encoding.properties` file to specify equivalent Java encoding methods that should NOT cause document parsing to restart if the mapped encoding changes.

About Encoding Methods and Parsing Restarts

An HTML document's encoding method can be specified either in its HTTP header, or in a meta tag within the document. Because the DCA parser chooses its Java encoding method BEFORE it begins parsing a document, the later discovery of a different encoding method specified in a document's meta tags will cause parsing to restart.

While under normal circumstances this behavior is correct, there may be situations in which it would not be preferable not to restart parsing -- namely, if the new encoding method would return essentially the same output as the current. You can use the `encoding.properties` file to specify equivalent Java encoding methods that should NOT cause document parsing to restart if the mapped encoding changes.

To Specify Equivalent Java Encoding Methods

To specify equivalent Java encoding methods:

1. In the Admin Tool, select Configuration > Encoding.
2. On a new line, specify equivalent Java encoding methods, as follows:

   ```
   <Java encoding method 1>=<Java encoding method 2>
   ```

3. To specify additional equivalents to that method, add additional lines to the file, as follows:

   ```
   <Java method 1>=<Java method 2>

   <Java method 1>=<Java method 3>

   <Java method 1>=<Java method 4>
   ```

4. Click Submit.

Note: `Encoding.properties` can be edited while the DCA server is running.

Restarting the server after adding a new mapping is not necessary.

Example

```
JISAutoDetect=ShiftJIS
```

In the example above, the JISAutoDetect and ShiftJIS Java encoding methods are specified as equivalents. If the DCA begins parsing a page using JISAutoDetect, and then discovers an HTML encoding specification in the document's meta tags which is mapped to ShiftJIS (in the `Charset.properties` file), it checks `encoding.properties` to see if JISAutoDetect and ShiftJIS are equivalent methods. If they are, document parsing will not restart.

See Also

For related information, see:

About Localizing the DCA Parser
How to Use the Charset Properties File
How to Use the CharsetDefault Properties File
How to Access the Admin Tool

Section XI. Browser Mapping Configuration

About Browser Mapping

Some Web sites use a requester's browser type and version to determine what content to deliver, what layout to use, or to confirm a user's identity. For example, users with older browsers may be directed to pages with simpler content. Netscape and Internet Explorer users may receive different style sheets for the same page. During a particular session, a secure site may check a user's browser type as part of its identity confirmation. Any of these situations can cause problems in a Web collaboration session.

To correct this, the DCA includes a browser mapping feature. Browser mapping lets you match the different browsers in use during a DCA session to a single one which represents a highest common denominator. This becomes the browser used in HTTP header requests the DCA sends to Web content servers.

You can configure browser mapping to address both server-side and client-side browser detection.

How Does Browser Mapping Work?

When participants join a DCA session, the DCA identifies their browsers, compares the mappings that have been set up for those browsers, and then determines the highest common denominator browser. This highest common denominator becomes the browser the DCA present itself as when making requests to Web content servers.

Each time a new participant joins a DCA session, the DCA re-examines the browsers in use and, as necessary, determines a new highest common denominator browser. Therefore, it is possible for the highest common denominator browser to change over the course a DCA session. Note that the remapping of browsers does not occur as participants *leave* a session.

Note: Until a user makes an initial request for content to the DCA, that user's browser is not considered by the DCA for the purposes of browser mapping. The initial request can include: clicking a link on a parsed page; receiving a page share from another user; entering a URL in the Collaboration Toolbar Address Bar.

Example

Imagine a DCA session in which:

- Participant A's browser is IE5.0 and IE5.0 is mapped to: IE5.0, IE4.01, and IE4.0.

- Participant B's browser is NS4.7 and NS4.7 is mapped to: NS4.7, NS4.0, and IE4.0.

Because IE4.0 is the highest common denominator among the two lists, it is the browser type/version the DCA will present itself as to Web servers.

Unmapped Browsers

If a DCA session includes a participant whose browser type is not mapped (that is, has no common denominator browsers defined for it), that browser is ignored for the purposes of browser mapping. When this occurs the DCA determines and uses the highest common denominator based on the browsers in use by the remaining participants.

If *all* of the participants in a DCA session use unmapped browsers, browser mapping is effectively not occurring, and each participant may receive content specific to his or her browser.

See Also

For related information, see:

How to Configure Browser Mapping
How to Use the BrowserMap Properties File
How to Use the UserAgent Properties File

How to Configure Browser Mapping

Browser mapping lets you to match the different browsers in use during a DCA session to a single one which represents a highest common denominator.

You can configure browser mapping to address both server-side and client-side browser detection.

To Configure Browser Mapping

By default, browser mapping is enabled in the DCA and pre-defined mappings exist for a number of common browser/platform combinations. As necessary, you can configure the DCA to create mappings for additional browsers, as well as to modify existing mappings.

To configure browser mapping:

1. In the `UserAgents.properties` file, specify browsers for which you want to create mappings.
2. In the `BrowserMap.properties` file, create mappings for the browsers you specified in the `UserAgents.properties` file.

To Disable / Re-enable Browser Mapping

By default, browser mapping is enabled in the DCA. Under ordinary circumstances, there should be no particular need to disable it; even for sites that contain no browser-dependant content or layout, browser mapping should not adversely affect performance. If desired, however, browser mapping can be disabled.

To disable browser mapping:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. Set BrowserMapping to False.
3. Click Submit.

To re-enable browser mapping:

To re-enable browser mapping after disabling it:

1. In the Admin Tool, select Configuration > ProxyProperties.
2. Set BrowserMapping to True.
3. Click Submit.

See Also

For related information, see:

About Browser Mapping
How to Use the BrowserMap Properties File
How to Use the UserAgent Properties File
How to Access the Admin Tool

How to Use the BrowserMap Properties File

Use the `BrowserMap.properties` file to add or edit mappings of individual browsers to lists of common denominator browsers. This helps ensure that users with different browsers receive the same web content.

To Create a New Browser Mapping

To create a new browser mapping:

1. In the Admin Tool, select Configuration > BrowserMap.
2. In the Add a New Property - Comment field, enter a simple descriptive label for the browser. For example: `My IE 6.0 Browser`
3. In the Name field, enter a Perl 5 regular expression that represents the browser's user-agent string. Use Java properties rules for escaped characters, etc. For example:

   ```
   Mozilla/4.0\\s*\\(compatible;\\s*MSIE\\s+6.0.*
   ```

   Note: You can enter an expression that represents a single browser, or a range of browsers (for example all IE version 5.0 browsers). A fairly comprehensive list of user-agent header strings can be found at: http://www.highermind.org/design/user_agent/.

4. In the Value field, enter one or more common denominator browsers. Common denominators:
   - Must be defined in the `UserAgent.properties` file in order to work.
   - Must be separated by semi-colons.
   - Should be ordered from most to least capable.
   - Should include the browser itself (that is, remember to include IE 4.0 as a common denominator for IE 4.0).
   - Example: `IE401;IE40;NS3;IE3`
5. Click Submit.

To Edit an Existing Browser Mapping

You can edit an existing browser mapping by modifying its list of common denominator browsers.

To edit a browser mapping:

1. In the Admin Tool, select Configuration > BrowserMap.
2. In a browser's Value field , add, edit, or delete its common denominator browsers, as necessary. Common denominators:
   - Must be defined in the `UserAgent.properties` file in order to work
   - Must be separated by with semi-colons.
   - Should be ordered from most to least capable.
   - Should include the browser itself (that is, remember to include IE 4.0 as a common denominator for IE 4.0).
3. Click Submit.

Note: To completely remove a browser, or to edit its label, you must open and edit `BrowserMap.properties` in a text editor.

See Also

For related information, see:

About Browser Mapping
How to Configure Browser Mapping
How to Use the UserAgent Properties File
How to Access the Admin Tool

How to Use the UserAgent Properties File

Use the `UserAgent.properties` file to define browsers you plan to specify as common denominator browsers (for use in browser mapping).

About User-Agent Header Strings

Every HTTP request sent to a Web server includes in its header a user-agent string. The user-agent string is what identifies the requesting browser's type and version to a Web content server. Under normal conditions, the information in this string originates from the requesting browser itself. However, because browser mapping allows the DCA to mimic a browser not currently in use, the user-agent string for that browser must come from a different source. This is accomplished through `UserAgent.properties`.

About appname, appversion, and appcodename

Client-side browser detection is accomplished through scripts that determine a browser's type and version based on its appname, appversion, and appcodename properties. Specifying these properties for the common denominator browsers you identify in `UserAgent.properties` allows browser mapping to work with sites that employ client-side browser detection.

To Define a Common Denominator Browser

To define a common denominator browser:

1. In the Admin Tool, select Configuration > UserAgents.
2. In the Add a New Property - Comment field, enter the label you want to appear over the item in the Admin Tool. For example: `My NS 3.03 Browser`
3. In the Name field, enter an ID for the browser. The ID is how you will later reference the browser as a common denominator in `BrowserMap.properties`. For example: `NS303`
4. In the Value field, enter the browser's user-agent string.
   - Example: `Mozilla/3.03Gold (WnNT; I)`
   - A fairly comprehensive list of user-agent header strings can be found at: http://www.highermind.org/design/user_agent/.

5. Optionally, to allow browser mapping to work with client-side browser detection, to the end of the user-agent string, append the browser's appname, appversion, and appcodename as shown below:

   - `<user-agent string>=<appName>=<appVersion>=<appCodeName>`
   - Example: `Mozilla/3.03Gold (WnNT; I)=Netscape=3.03Gold(WnNT; I)=Mozilla`
   - More information on appname, appversion, and appcodename properties can be found at: http://msdn.microsoft.com/library/default.asp.

6. Click Submit.

See Also

For related information, see:

About Browser Mapping
How to Configure Browser Mapping
How to Use the BrowserMap Properties File
How to Access the Admin Tool

Section XII. Popup Window Sharing Configuration

How to Configure Popup Window Sharing

Out of the box, the DCA is automatically configured to allow users to collaborate on the contents of popup windows. As desired, you can modify the DCA's default configuration to disallow popup window sharing, or to only allow sharing of specific popups.

To Turn Off Popup Window Sharing

For sites that do not require it, you can turn off popup window sharing. Once this is done, users will be unable to collaborate on the contents of popup windows.

To turn off popup window sharing:

1. Open the proxy_rules_include_text.xml file located at: <DCArootdirectory>\webapp\WEB-INF\Cisco\properties.
2. Uncomment the following line of code in the file:

   ```
   // groupStr+="var _DCA_stopSharingPopups=true;";
   ```

3. Restart the DCA.

To Target Specific Popups for Sharing

For sites that do not require it, you can turn off popup window sharing. Once this is done, users will be unable to collaborate on the contents of popup windows.

To turn off popup window sharing:

1. Turn off popup window sharing as described in the previous section.
2. For popup windows whose content you want to share though the DCA, replace the JavaScript "window.open()" call normally used to open the window with the following:

   ```
   if(typeof _DCA_windowOpen!="undefined")_DCA_windowOpen(arguments);

    else window.open();
   ```

Note that any popup window opened during a DCA session by methods other than

this code will not be sharable. This code will not adversely affect the normal behavior of windows in which it is inserted. Its syntax ensures that windows it calls will open normally in non-DCA contexts.

Additional Guidelines for Popup Window Sharing

The following guidelines can help you minimize issues with popup window sharing. For additional caveats pertaining to popup window sharing, see the *DCA 2.01 Release Notes*.

Do Not Use window.open(this.href)

Using window.open(this.href) to open a popup results in a URL that is parsed twice by the DCA, therefore containing a duplicate DCA thread. In SSL mode, because the duplicate thread causes the DCA to attempt to access itself, the URL will fail to load in the window. (In non-SSL mode this URL, while undesirable, will nevertheless function correctly).

For example, a popup called using:

```
<a href="http://www.somesite.com" OnClick="window.open(this.href);
return false;">
```

will result in a URL formatted as:

```
http://mydcaserver.mydomain.com/DCA/http/mydcaserver.mydomain.com/DCA/ht
tp/www.somesite.com)
```

Do not use (this.href) to specify URLs in window.open calls. Specify an actual URL instead.

Use Unique Window Names

In some browsers, attempts to share a popup window that has the same name but a different location than an earlier shared popup window may result in a timing-related JavaScript error. This occurs when the earlier window is closed on the agent desktop but remains open on the caller side. The current popup window binds to the original window handle in the caller. When the code executes to examine the existing window's content, the browser attempts to deliver and execute code in a partially loaded window with invalid content.

To avoid this problem, use unique window names in popup window open commands, OR, have agents leave popup windows open during a session.

Specify a Location (URL) in window.open Events

In some browsers, users may receive errors when sharing popup windows with unspecified locations. The error can manifest itself as a blank window and/or a JavaScript error message. Error messages can appear at any time: when the window opens, when it is in use, or as it is being closed. (DCA page sharing functionality will continue to function correctly once the caller has acknowledged the error messages.)

To avoid this problem, always specify a location (URL) in window.open events.

See Also

For related information, see:

How to Use the Collaboration Toolbar

Section XIII. Custom Error Page Configuration

About Custom Error Pages

When an Internet user encounters an error condition, the Web server or browser returns an error message that describes the problem to the user. For example, the standard 404 error condition returns the message "The Web server cannot find the requested URL..."

On occasion, however, an error condition either returns no message, or returns a URL that lists the error code but contains no explanatory text. In a collaboration environment, this can leave both agent and user confused and wondering what to do next.

To address this, the DCA contains a feature called Custom Error Pages. It allows you to specify a file or URL to display when an error condition fails to return message text to the user. For example, when the standard 500 error condition (internal server error) fails to return message text, you may want to display a page containing special instructions to users on how to proceed.

Configuring Custom Error Pages

You configure the DCA to use custom error pages by specifying the error code you want to capture, and the URL or file you want to display when that code returns a page without text. You do this in the `Responses.properties` file.

DCA Default Error Page

Out of the box, the `Responses.properties` file references a DCA default error page (`DefaultError.html`) that displays under the following conditions:

- The user has entered an invalid URL.
- The Web server the user is attempting to access is inaccessible.
- The user is attempting to access a secure (SSL) page and the DCA Collaboration Toolbar not is configured properly for SSL.
- The user is attempting to access a secure (SSL) page and the DCA server not configured for SSL (i.e., valid server certificate not installed).

Default Error Page Localization

Out of the box, the message displayed by `DefaultError.html` is localized into English, French, Spanish, German, Korean, and Chinese (Simplified). The correct localized version of the message is called as follows:

1. `DefaultError.html` calls `responseError.jsp` (located at `<DCArootdirectory>\uiserver\default\dca\responseError.jsp`). `responseError.jsp` determines the user's locale and then returns text from a locale-specific properties file.

2. The locale-specific properties files, named `errorResponses_<languagecode>.properties` are located at: `<DCArootdirectory>\uiserver\WEB-INF\properties\default\dca`.

You can modify the text of the default error message by editing the appropriate localized properties file.

You can add additional localizations by creating additional localized versions of `errorResponses.properties`.

DCA-Specific Error Codes

In addition to standard browser/server error conditions, there are three DCA-specific error conditions for which you may want to define error pages. Out of the box, the DCA calls English version error pages (`InvalidKey.html, SessionLimit.html`, and `KeyExpired.html`) that identify these conditions:

997	Missing or corrupt DCA license key.
998	Maximum session limit reached. Cannot initiate a new session. (Note: Valid DCA licenses support an unlimited number of seats; this error condition will only occur in test environments when temporary licenses are used).
999	DCA license key expired.

See Also

For related information, see:

How to Use the Responses Properties File

About HTTP Error Codes

How to Use the Responses Properties File

Use the `Responses.properties` file to specify custom error pages (URLs or files) that you want to display in place of standard error pages IN THE EVENT that those pages fail to return message text to the user. (Error pages SHOULD always return message text, but this sometimes does not occur.)

To Specify a Custom Error Page

To specify a custom error page:

1. In the Admin Tool, select Configuration > Responses.
2. Specify custom pages as shown in the table below. You can specify:
   - URLs and/or files to be returned for specific error conditions.
   - A default URL or file to be returned for non-specified conditions. The DCA ships with a default error page (`DefaultError.html`) that displays when a user enters an invalid URL or when a Web server is inaccessible. Out of the box, `DefaultError.html` is localized into English, French, Spanish, German, Korean, and Chinese (Simplified).
3. Click Submit.

Examples

The following table gives examples of how to specify different response types:

To Specify:	**Then:**	**Example:**

A response URL	For each error condition you want to capture, enter:
	
	`ResponseCode=<HTTPresponsecode>`
	
	`ResponseUrl=<URL>`
		This example redirects users to a page named "404custom.html" when a 404 server error occurs:
		
		`ResponseCode=404`
		
		`ResponseUrl=http://www.mysite.com/404custom.html`

A response file	For each error condition you want to capture, enter:	This example redirects users to a file named "whoops.txt" when a 400 server error occurs:

To Specify:	Then:	Example:

	`ResponseCode=<HTTPresponsecode>`
	
	`ResponseUrl=File:<filename>`
	
	(file location relative to <DCArootdirectory>\webapp\WEB-INF\Cisco)
		`ResponseCode=400`
		
		`ResponseUrl=File:whoops.txt`

A default response URL or File	Enter either:
	
	`DefaultURL=<URL>`
	
	OR
	
	`DefaultFile=File:<filename>`
	
	(file location relative to <DCArootdirectory>\webapp\WEB-INF\Cisco)
		This example redirects users to a default file"mybad.txt" for all server error conditions EXCEPT those explicitly defined elsewhere in Responses.properties:
		
		`ResponseUrl=File:mybad.txt`
		
		Note: The DCA includes a file (DefaultError.html) which it uses as the default error response.

To Change the Responses.properties File Name

You can specify a different file to use in place of `Responses.properties` by editing the ResponseURLFilename property in the `Proxy.properties` file.

See Also

For related information, see:

About Custom Response Pages
How to Access the Admin Tool
About HTTP Error Codes

Part 3: Reference

How to Use the Collaboration Toolbar

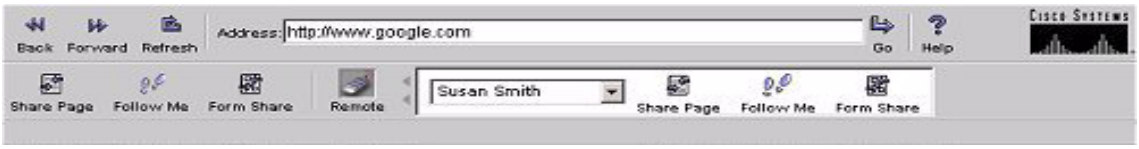
The following instructions on how to use the DCA Collaboration Toolbar are excerpted from the Collaboration Toolbar online Help for CCS agents. They are included in *this* guide as an aid to understanding the Collaboration Toolbar, and as an example of the Toolbar documentation available to agents.

Note that the Toolbar's online Help is by necessity generic; it describes all Toolbar features that can be available to agents, even though some of these you may ultimately choose to make unavailable. Its topics include:

- What is the Collaboration Toolbar?
- How to Open Web Pages
- How to Share Web Pages
- How to Share Content in popup Windows
- How to Share Form Data
- How to Use Caller Remote Control
- How to Reload the Toolbar

What is the Collaboration Toolbar?

The Collaboration Toolbar is a simple set of tools for sharing Web content in a Web collaboration session. It loads automatically in its own browser window at the start of a session, and is available to both agents and callers.



The Collaboration Toolbar includes its own navigation and content sharing controls (Back/Forward buttons, Address Bar, Share page button, etc.). These must be used in place of similar controls that normally appear in your Web browser or Agent Desktop. Your company can configure the Toolbar separately for agents and callers. Typically, callers are given access to fewer content sharing features.

How to Open Web Pages

You open a Web page with the Collaboration Toolbar much as you would in a regular browser window. Note that, unless you are in Follow Me mode, opening a page doesn't automatically share it with other participants in your session -- other participants don't receive a page until you click the Share Page  button.

To Open a Page

To open a Web page:

- Method 1: Type the page's URL in the Collaboration Toolbar's Address Bar and click the Go  button or press <Enter>.
- Method 2: From a different Web page, click a link to the page you want to see.
- Method 3: If one has been created, click a link to the page in your CCS Agent Desktop Script area.
- Method 4: To go back to a recently viewed page, use the Toolbar's Back or Next  buttons.

How to Share Web Pages

Sharing a page means to send that page to other participants in your Collaboration session.

To Share a Page

There are three different ways to share a Web page from the Collaboration Toolbar. The method you choose will depend on whether you want to share only the current page, and whether the page contains an online form whose data you want to share.

To share a page:

- Method 1, **Page Share**: To share only the page you are currently viewing, click the Share Page  button.
- Method 2, **Follow Me**: To share the current page AND every page you navigate to from that point on, click the Follow Me  button to turn on Follow Me mode.

To turn off Follow Me mode, click the button again.

- Method 3, **Form Share**: To share the current page AND any form data you have entered on the page, click the Form Share  button. Form data is then shared automatically as you type it. To turn off Form Share, click the button again.

How to Share Content in Popup Windows

The Collaboration Toolbar lets you share content in JavaScript popup windows. A JavaScript popup is an additional browser window that opens as the result of a JavaScript command on a Web page. Some popup windows open automatically when you open a page. Others open as the result of an action, such as clicking a link, or positioning your mouse pointer over a certain place on a page.

Note: Not all additional browser windows are JavaScript popups. Windows opened using your browser's Open Link in New Window command, or through an HTML target specified in an HREF tag, are not JavaScript popups. You cannot share content in these windows. If necessary, consult your administrator for help determining which popups windows can be shared.

Sharing Content in Popup Windows

To share the contents of popup windows with a caller:

1. Open the popup(s) on your computer.
2. Use any page sharing command (Page Share, Follow Me, or Form Share) on the Collaboration Toolbar. When you do this:
   - All popups windows on your desktop *opened during the current session* are opened for the caller. There is no restriction on the number of popups you can have open at one time.
   - The contents of those popup windows are shared. The Collaboration Toolbar automatically shares the content from ALL popups *opened during the current session*. There is not way to selectively share the contents of a single window.

Switching Between Sessions (Multi-Session Agents)

For multi-session agents sessions, when you switch back and forth between different sessions:

- Any popups opened during a session close automatically when you switch to a different session.

- When you later switch back to that session, the closed popups reopen automatically. When they reopen, they display content current as of the last time you used a page sharing command.

Sharing Popups from Callers

If available, you can use the Caller Remote Control feature to share the contents of popup windows from a caller's desktop to your own.

Navigating in Popup Windows

Most popup windows do not have their own toolbar (e.g., Address bar, Forward and Back buttons, etc.). However, if you do encounter a toolbar in a popup window, DO NOT USE IT. Restrict your navigation within popup windows to page links.

Closing Popup Windows

- Closing a popup window on your desktop does not close the window for other participants. You cannot automatically close popups for callers; they remain open until the callers themselves close them.
- If a caller closes a popup window, and that popup is still open on your desktop, you can reopen it for the caller by using any content sharing command.

How to Share Form Data

To share data you enter in an online form, you use the Form Share command. The Form Share command automatically shares both form data and the Web page on which the form appears. Therefore, you can use the command even if and a caller are not initially on the same page.

Bear in mind, it is only necessary to share form data when you want other participants to SEE the information you enter. If you are completing a form simply to proceed to another Web page, it may not be necessary to share the form.

To Share Form Data

To share form data:

- Click the Form Share  button. This turns on Form Share mode. In Form Share mode, data you enter in an online form is shared automatically as you type it. Data entered in the form prior to clicking the Form Share button is also shared.

Turning on Form Share also automatically turns on Follow Me mode.

- To turn off Form Share mode, click the Form Share button a second time. Turning off Form Share also automatically turns off Follow Me mode.

Form Share and Browser Types

Form Share works slightly differently depending on the browser type of the person completing the form. If your browser is:

- Netscape: Data entered in a particular form field is shared only after you shift focus away from the field (by tabbing out or clicking elsewhere on the page).
- Internet Explorer: Form data is shared as soon as it is entered.

How to Use Caller Remote Control

The Caller Remote Control feature lets you, the agent, share content from a caller's browser with other session participants, including yourself. Depending on your configuration, you may be able to use caller remote control to:

- Share a caller's current page.
- Turn on caller-led Follow Me.
- Share data a caller has entered in an online form.

Note: If you're already familiar with Remote Control in standalone Collaboration, you'll notice that the Collaboration Toolbar's Caller Remote Control works somewhat differently:

In standalone Collaboration, you use Remote Control to turn on or off content sharing controls in the caller interface. Callers then use these controls to share content.

With the Collaboration Toolbar, you use remote control to retrieve content from the caller, but not to enable or disable buttons for the caller. For example, you can use remote control to trigger a page share or a form share, but not to turn the caller's Page Share button on or off (in a typical configuration, the caller may not even have a Page Share button).

The Caller Remote Control Toolset

The content sharing buttons available to you in Caller Remote Control will be the same as those available to you elsewhere in the Collaboration Toolbar. For example,

if your configuration includes Share Page and Form Share buttons, but not a Follow Me button, that same combination of buttons will be available in Caller Remote Control.

To Use Caller Remote Control

To use Caller Remote Control:

1. Click to expand the Remote Control toolset.
2. From the session participant drop-down list, select the caller whose content you want to share.

   The list displays each participant (except yourself) in the current session. The list updates each time a caller joins or leaves the session.

3. Select the content sharing method you want to use:

   - To share the caller's current page only, click the Remote Control's Page Share  button.
   - To turn on caller-led Follow Me, click the Remote Control's Follow Me  button. When you do this, the caller's current page and every page the caller navigates to from that point on is shared.
   - To share the caller's current page and data the caller has entered in an online form, click the Remote Control's Form Share  button.

Special Usage Notes: Follow Me and Form Share

When using Follow Me or Form Share, it's important to remember that these sharing methods are "modal." This means:

- They apply to only one session participant at a time.
- Once they have been applied to a participant, they remain "on" until you either turn them off or apply them to a different session participant. Simply closing the Remote Control toolset DOES NOT turn off Follow Me or Form Share.
- Turning on Form Share automatically turns on Follow Me as well. Turning off Form Share also turns off Follow Me.

How to Reload the Toolbar

The Collaboration Toolbar loads automatically in its own browser window at the start of a Collaboration session. There are several ways that it can be accidentally unloaded (that is, closed), at which point you will be unable to share Web content. These include:

- For single-session agents: Using the White Board or App Share features in the Agent Desktop (if present).
- For all agents and callers: Closing the Collaboration Toolbar's browser window; using navigation commands on the browser's popup (right mouse button) menu.

If you or a caller accidentally unload the toolbar, you can reload it as described below.

To Reload the Toolbar

If necessary, to reload the toolbar:

On the top of the Caller Information Page in the Agent Desktop, click the Reload Toolbar button. This will reload the toolbar for all session participants, opening a new browser window if necessary. When the Collaboration Toolbar is reloaded, it automatically displays the last shared page.

Note: You can also reload the Toolbar for yourself only by clicking the External View  button on your Agent Desktop.

See Also

For related information, see:

About the DCA Collaboration Toolbar
About Toolbar Changes to CCS
How to Configure the Collaboration Toolbar
How to Configure Popup Window Sharing

How to Stop and Start the DCA

Stopping and/or starting the DCA is a simple matter of stopping or re-starting you DCA server's Web server software.

To Stop the DCA

Stop the DCA on a Windows server by stopping the IIS Admin and WWW Publishing Services:

1. From the Windows Start menu, select Settings > Control Panel > Admin Tools > Services.
2. Right-click IIS Admin Service.
3. From the popup menu, select Stop.
4. A dialog opens listing dependant services that will also be shut down. This list should include the WWW Publishing Service. Click OK.

To Start the DCA

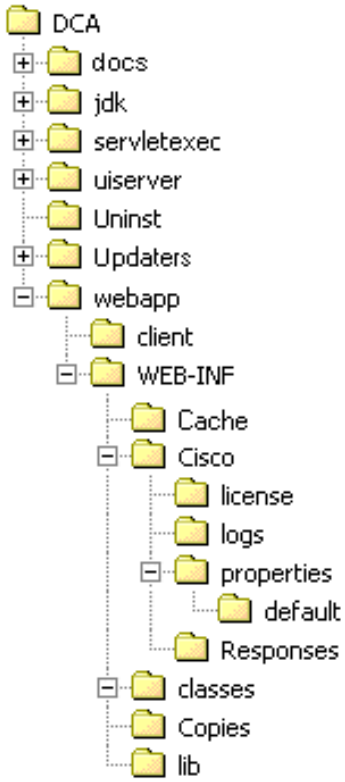
1. Start the DCA on a Windows server by starting the IIS WWW Publishing Services:
2. From the Windows Start menu, select Settings > Control Panel > Admin Tools > Services.
3. Right-click WWW Publishing service.
4. From the popup menu, select Start.

Note: Services whose Startup Type is set to Automatic will also restart automatically when you reboot the server.

About DCA Directory Structures

The following table shows the DCA 2.01 directory tree that installs on your server, and gives a brief description of the files stored in each directory. Note especially that:

- All configurable DCA files are located within `<DCArootdirectory>\webapp\WEB-INF\Cisco`. Do not modify DCA files outside of this directory unless specifically instructed to do so.
- Files used to update Cisco Collaboration Server to integrate it with the DCA are located in the `<DCArootdirectory>\Updaters` directory.
- The DCA installs the JDK and ServletExec within the DCA root directory.

DCA Directory Tree:	Directory Description:
	**DCA:** DCA root directory. **DCA\docs and subdirectories:** DCA user documentation. **DCA\jdk and subdirectories:** Java Development Kit program files. **DCA\servletexec and subdirectories:** ServletExec program files. **DCA\uiserver and subdirectories:** Cisco UIServer program files. Provides communication between the DCA Server and the Collaboration Toolbar. **DCA\Uninst:** DCA uninstall information used by the DCA Uninstaller. **DCA\Updaters and subdirectories:** DCA-CCS Updater files. Includes individual copies of DCA-modified CCS configuration files. **DCA\webapp\client:** JSP pages used by Collaboration Toolbar. **DCA\webapp\WEB-INF:** Container for DCA program and configuration files. **DCA\webapp\WEB-INF\Cache:** Files stored by DCA Static Cache. **DCA\webapp\WEB-INF\Cisco:** Stores DCA configuration directories. **DCA\webapp\WEB-INF\Cisco\license:** Stores DCA license file.

	DCA\webapp\WEB-INF\Cisco\logs: Stores DCA log files. **DCA\webapp\WEB-INF\Cisco\properties and subdirectories:** Stores DCA configurable properties files. **DCA\webapp\WEB-INF\Cisco\Responses:** Stores user-defined custom error pages. **DCA\webapp\WEB-INF\classes and subdirectories:** DCA program files. **DCA\webapp\WEB-INF\Copies:** Default directory used to store Copy Server pages. **DCA\webapp\WEB-INF\lib:** DCA program files.

See Also

For related information, see:

About DCA Properties Files
The DCA Installation and Integration Guide

How to Confirm the DCA Build Number

As necessary, you can use the DCA Admin Tool to confirm the version and build number of your DCA Server software. You may need to do this prior to installing a patch, or when troubleshooting a problem with the Cisco Technical Assistance Center.

To Confirm the DCA Build Number

To confirm the version and build number of your DCA Server:

In the Admin Tool, select Server Administration. Your DCA version and build number are displayed.

See Also

For related information, see:

How to Access the Admin Tool
Cisco Support for the DCA

How to Use the ServletExec Admin Tool

While usually not necessary, you may on occasion want to adjust settings in ServletExec (for example, to adjust your server's Virtual Machine memory or to modify Copy Server properties).

To Access the ServletExec Admin Tool

ServletExec includes a GUI administration tool that can be used to adjust ServletExec settings. Access to the ServletExec Admin Tool is limited to the local machine.

To access the admin tool:

In a Web browser, navigate to `http://localhost/servlet/admin`. For example: `http://myserver/servlet/admin`

See Also

For related information, see:

How to Access the Admin Tool
How to Change the Admin Tool Username and Password
How to Limit Admin Tool Access
About DCA Sessions
How to View DCA Server Statistics
About DCA Logs
About DCA Properties Files
How to Maintain Root Certificates
How to Confirm the DCA Build Number

How to Modify Java Virtual Machine Memory

For performance reasons you may need or want to increase the amount of memory available to the Java Virtual Machine on your DCA server. Note that Cisco recommends JVM memory never be lower than 512 Mb, the default set by the DCA is installation.

To Change Virtual Machine memory

To change the amount of memory available to Java Virtual Machine on your DCA server:

1. Open the ServletExec Admin Tool.
2. In the Admin Tool navigation frame, click Virtual Machine > Settings.
3. In Max Heap Size, adjust the amount of available memory.
4. Click Submit.
5. Restart IIS.

See Also

For related information, see:

How to Use the ServletExec Admin Tool
The DCA Installation and Integration Guide

About HTTP Error Codes

Below are error codes for common HTTP errors, and for several errors particular to the DCA. You can use the `Responses.properties` file to specify custom error pages for these errors.

Standard Browser Error Codes

400	Requested syntax was wrong.
401	The request included an Authorization field, but the client (browser) did not supply one.
403	Request for a file that is presently set to forbidden, so the file can't be sent.
404	Server can't find the requested URL.
405	Server didn't understand the METHOD supplied by the client.
406	Resource found, but not sent. Type of resource incompatible based on information passed between server and client.
407	Proxy authentication required.
408	Request timeout. The client did not produce a request within the time that the server was prepared to wait.
409	Conflict. The request could not be completed due to a conflict with the current state of the resource.
410	Resource no longer available, server has no forwarding URL.

Standard Server Error Codes

500	Server encountered an internal error and can't continue.
501	Server does not support the method being requested.
502	Server attempted to retrieve resource from another server or gateway with this secondary server failing to return a valid response to the calling server.
503	Server is too busy and unavailable to process the request.
504	Secondary server timed-out before the request could be completed.
505	HTTP version not supported.

DCA Error Codes

997	Missing or corrupt DCA license key.
998	Maximum session limit reached. Cannot initiate a new session. (Note: Valid DCA licenses support an unlimited number of seats; this error condition will only occur in test environments when temporary licenses are used).
999	DCA license key expired.

See Also

For related information, see:

About Custom Error Pages

How to Use the Responses Properties File

How to Verify Your Server's IP Address

To verify your server's IP address on a Windows 2000 server:

1. From a command line, enter: `ipconfig`
2. Press <Enter>.

How to Verify Your Server's DNS Entry

Your server's DNS entry is the combination of its host name and domain (for example, myserver@cisco.com) as defined on its DNS server.

You can reference your server's DNS entry by performing the steps listed below. However, this information should be verified with your system administrator.

To Verify Your Server's DNS Entry

To verify your server's DNS entry on a Windows 2000 server:

1. From a DOS prompt, enter: `nslookup` followed by your server's IP address. For example: `nslookup 161.44.160.9`
2. Press <Enter>.

How to Verify Your Server's Port Settings

To verify your Web server's HTTP and HTTPS (SSL) port settings on a Windows 2000 server:

1. From the Windows Start Menu, select Settings > Control Panel > Admin Tools > Internet Services Manager.
2. Double-click the computer name.
3. Right-click Default Web Site.
4. From the popup menu, select Properties.
5. Select the Web Site tab.
6. Click Advanced.

DCA Glossary

The following important terms appear in the DCA documentation.

A

Admin Tool

The DCA Web interface for administering, monitoring and configuring the DCA.

agent

An individual using the Agent Desktop to access Collaboration Server (for example, an agent in a call center).

App Share

A Collaboration feature that allows you to share an open application on your desktop with other session participants.

C

caller

An individual using the Caller Control Panel to access Collaboration Server (for example, a call-center customer).

caller complexity mode

A Collaboration Server setting which determines if and when the CCS caller applet is downloaded to caller machines. The presence or absence of the caller applet in turn affects the type of content agents and callers can share.

The DCA automatically sets caller complexity mode to "simple." Simple mode, prevents the caller applet from downloading, thus eliminating security warnings otherwise displayed to callers. However, it also prevents participants from using two of CCS's advanced content sharing features -- App Sharing and White Boarding.

Caller Remote Control

A Collaboration feature that allows agents to trigger a content sharing event (for example, page share) from a caller's browser.

CCS

See Collaboration Server.

Cisco Collaboration Server

The Cisco Collaboration Server, an application that provides online Web site assistance through features like online chat, callback, desktop sharing, and Web content collaboration.

client-side processing

Includes Java applets, JavaScript and Visual Basic scripts, plugins, and a number of other mechanisms used to alter content after it has been received by the client browser.

collaboration

The act of sharing a view to Web content among multiple users.

Collaboration Toolbar

The DCA's Collaboration Toolbar is a set of Web tools that you use to view and share Web content in a

Collaboration session. The Collaboration Toolbar is used by both agents and callers. It loads automatically in the Collaboration Server's Shared View area at the start of a Collaboration session.

complexity

See *caller complexity mode*.

cookie

Data that can be used to identify a user to a Web server. Each time the user returns to that server, the user's browser presents the same cookie. The cookie can be updated as the user inputs information or performs actions on the site, from session to session.

Copy Server

A DCA service used to store unexpired pages after they have been forced from the DCA cache.

custom error

DCA feature in which a user-defined html file is returned when a DCA user encounters a Web browser or server error condition.

D

DCA

The Dynamic Content Adaptor, an application that extends the Cisco Collaboration Server's (CCS) functionality to allow successful sharing of pages that contain SPLIT content.

DCA cache

Pages returned to the DCA from a Web content server are initially stored in the DCA cache where they are available to satisfy subsequent user requests. Pages can expire in the cache, after which they are unavailable. Unexpired pages that are forced from the cache due to space constraints are stored in the Copy Server.

DCA server

The server on which the DCA is installed. The DCA cannot be installed on the same server as Cisco Collaboration Server.

dynamic content

Web content that is generated dynamically based on factors like the time it was requested or information provided by the user. Includes live, interactive, personalized, and transactional content.

F

Follow Me Browsing

A Collaboration feature that allows participants to browse a Web site together. Each page the session leader navigates to is automatically sent to other participants.

Form Share

A Collaboration feature that allows you to share data you have entered in an online form with to other participants in your session.

frameproofed pages

Web pages with JavaScript code that prevents the pages from opening in a frame of another Web page.

G

GET

Method of submitting an online form in which form data are passed in a query string appended to a URL. This makes GETs subject to a URL's maximum size limitation (typically, 2Kb). See also, POST.

I

interactive content

Content that is contingent upon a user's interaction with a Web page.

L

live content

Content that changes based on factors such as the time it was requested or actions the user has performed previously.

O

output cache

After a page is stored in the DCA's Response cache, the DCA reconstructs it and passes it to the Output cache, where it is available to users. Subsequent requests for the page are satisfied directly from the Output cache.

P

Page Share

A Collaboration feature that allows you to send the page you are currently viewing to other participants in your session.

participant

Any individual engaged in a DCA or CCS session (i.e., an agent or client).

personalized content

Content that is specific to the user requesting it. A Web content server typically determines what Personalized content to display based on a user cookie or other state information.

POST

Method of submitting an online form in which form data are attached to the end of the POST request in its own object. See also, GET.

properties files

A set of files you use to configure and customize the DCA. Properties files can be edited through the Admin Tool.

R

remote control

See *Caller Remote Control*.

response cache

When a page is initially returned to the DCA from a Web content server, the DCA parses the page, redirects all links on the page to the DCA, and then stores the page as a Java object in its Response cache.

S

secure socket layer

A standard technology that encrypts content sent between Web browsers and servers. When Cisco Collaboration Server is SSL-enabled, the padlock in the upper right corner of the agent desktop is engaged.

session

A successful connection between an agent and a caller. Collaboration agents using the multi-session chat desktop can participate in multiple online chat sessions.

secure content

Content that is access-controlled to a specific user or group of users. Access can be controlled in a number of ways, for instance by a login password or by a user's IP address.

Shared View

The Collaboration Shared View is either a frame or full browser window that displays the content that agents and callers are sharing. Collaboration Toolbar users must use an external shared view.

SPLIT content

An acronym for secure, personalized, live, interactive, and transactional Web content that cannot be shared by simple URL sharing.

T

transactional content

Content that is returned based on data a user submits to a back-end system, for example, by submitting an order form.

W

White Board

A Collaboration feature that allows you to share a drawing program with other session participants.

Cisco Support for the DCA

The following resources are available to DCA users:

Online Resources

Additional DCA information is available online at:

- Latest DCA user documentation: www.cisco.com
- Technical tips:www.cisco.com/warp/customer/640/
- Known issues and workarounds: www.cisco.com/cgi-bin/Support/Bugtool/home.pl (listed as: Cisco Collaboration Server Dynamic Content Adapter)

Note: Some resources on the Cisco Web site require you to have an account. Register for an account at: www.cisco.com/register/

To Open a Technical Assistance Call

You can get technical assistance with the DCA by contacting Cisco's Technical Assistance Center (TAC).

Providing Information to TAC

To assist you in troubleshooting a problem, the Cisco TAC may ask you to provide the following. You can expedite matters by having it available when you contact them:

1. Copies of your DCA Trace and Error log files. To ensure that your log files contain information on an error:
   a. Set the DCA log's logging level to Local Dump (its most verbose level).
   b. Restart the DCA.
   c. Repeat the actions that caused the error.
2. URLs of the page(s) on which the error occurred if they are external to your Web site.

3. Copies of your DCA parser XML files if you have customized the DCA parser.

To Contact the Cisco TAC

To open a request for technical assistance with the DCA, contact TAC at:

Online:	www.cisco.com/tac/
Email:	tac@cisco.com (please include "Dynamic Content Adapter" in the Subject line)
Telephone:	In North America: 1.800.553.2447 Outside North America: 1.408.526.7209

Copyright

Doc Version: 2.00112002

Index