Administration

Administration is an essential feature of any enterprise system. Once started, a system must remain operational for long periods of time with no downtime so it must provide ways for an administrator to manage it at runtime. This applies to both changes and updates to the application as well as providing information concerning its health. The more flexible and informative a system, the better an administrator will be able to ensure it runs efficiently and detect any issues with the system quickly.

VXML Server has been designed to afford maximum flexibility for administrators to control how it runs and to monitor vital statistics of its health. Administrators can add, remove and change applications deployed, are able to get information on the system and the applications, and even change the behavior of the system or components, without requiring a restart of VXML Server.

This chapter details the administration functions and statistics exposed by VXML Server and the methods by which these functions can be accessed and processed.

- Introduction to VXML Server Administration, on page 1
- Administration Information, on page 4
- Configuration Updates, on page 8
- Administration Functions, on page 12
- VXML Server Metrics, on page 22

Introduction to VXML Server Administration

VXML Server exposes three methods for an administrator to control it and obtain information. Each method is accessed differently and exposes different levels of functionality or information.

- The first method, and the most flexible, is the JMX-compatible management interface.
- The second method is through the use of administration scripts.
- The third method is using the system information web page.

The topics that follow discuss these three methods.

JMX Management Interface

Java Management Extensions (JMX) is a Java technology specifically designed for managing Java applications. It is part of the standard Java Virtual Machine and defines a standard interface for clients and servers. An application that to be managed by JMX will register MBeans to the JMX context. An MBean can be used to

expose information about the system that an administrator can fetch (for example, the total simultaneous calls on the system). An MBean can also be used to expose a function that an administrator can run (for example, to suspend an application). A client application communicates with the server through the JMX interface to allow administrators access to the information and function that is exposed.

VXML Server, is a server application, exposes many informational MBeans for information regarding itself as well as the applications deployed on it. It also exposes administrative MBeans for controlling important administrator functions. It does this in a fully JMX-complaint manner so that any JMX-compatible client will be able to interface with VXML Server to gain access to the information and functions. One such client is JConsole, which is a client bundled with JDKs provided by Sun Microsystems and others. Some JVMs and application servers provided by other companies may utilize alternative JMX-compatible clients that should work as well.

It is also possible for a developer to create their own custom MBeans for exposing functions or information that will then be viewed by a JMX-compatible client alongside the MBeans exposed by VXML Server. See the *Programming Guide for Cisco Unified CVP VXML Server and Cisco Unified Call Studio* for more on creating custom MBeans.

VXML Server is configured to be JMX enabled when using Tomcat. The JMX port number by default is 9696.

Once VXML Server is started, a JMX client can then be launched and configured to point to the machine on which VXML Server runs, whether it be on the same machine or a remote one. Once connected, the client provides a graphical interface for displaying the information and functions. The client will be able to display information about the JVM itself and typically the Java application server will publish its own set of MBeans. VXML Server information will be displayed where the MBeans are listed in its own *domain*. The domain is typically rendered in a tree structure and will list global information and functions (that is, information having to do with VXML Server itself) as well as information on the deployed voice applications. Detailed explanations of the individual MBeans are provided in the following sections.

To address security, JMX client consoles require the proper security certificates (if JMX security is enabled on the VXML server) and the client attempts to connect to a remote server. Certificates are not required to connect to a local VXML Server because the client already has access to the local system. For details on securing communication and accessing secured communications, see the *Configuring and Modifying Unified CVP Security* section in Configuration and Administration Guide for Cisco Unified Customer Voice Portal.

Of the available administration interfaces, the JMX interface for VXML Server provides the greatest functionality and flexibility. It does, however, require the JVM to have JMX active and a JMX-compatible client. It also has a higher risk and overhead due to this flexibility.

Administration Scripts

Most of the administration functions and some of the information about VXML Server are provided via command-line scripts that can be run by an administrator manually or an automated system directly. The administrator scripts do not use the JMX interface described in the previous section and are functional by default without requiring any configuration on the administrator's part. The included scripts act as the client. The scripts are provided in two forms: batch scripts for Microsoft Windows (ending in `.bat`) and shell scripts for Unix (ending in `.sh`).

Scripts are provided to run global functions (on VXML Server itself) or functions for individual applications. The scripts used for global administrator functions are found in the `admin` directory of VXML Server. The scripts used for individual application administration are located in the `admin` directory of each application.

The provided scripts are primarily used to expose VXML Server functions to administrators such as loading a new application, updating an existing application, suspending VXML Server, and so on. Some scripts provide

information, such as the number of active simultaneous calls on the server. This chapter describes in detail all available scripts and their functionality.

Security is an important concern when it comes to administration functions that are accessed from the command-line. Unified CVP sets up these precautions to allow only the appropriate people access to these scripts:

- By providing scripts or batch files (as opposed to through a graphical or web interface), the administrator must be logged into the machine in order to access them. Accessing these programs is as secure as the remote login process (such as SSH) and the permissions given to these scripts or the entire `admin` folder.
- VXML Server will only accept commands from the local machine, so even scripts stored on one machine cannot issue commands to an instance of VXML Server running on another machine. These two precautions ensure that only authorized administrators can access these functions.

Because the global administration scripts are stored in a different location from application scripts, each directory can be assigned different permissions. That way an administrator can be given access to the global administration scripts while still allowing the application scripts to be accessed by voice application developers.

Every administration script can be configured to ask for confirmation before the action is taken, to prevent the accidental processing of the script. By default the confirmations are on. They can be turned off by passing the command-line argument *noconfirm* to the script. This action can be useful if the administration scripts are run by automated systems such as cron jobs.

While not as flexible as the JMX interface, administration scripts provide easy access to VXML Server functions for both administrators and automated systems out of the box. The risk potential is similar to that of the JXM interface although there is less overhead because JMX is not enabled.

Note When you migrate scripts from Call Studio release version 12.6(2), which runs on JDK 1.8, to version 15.0, which runs on JDK 17.0, you must recompile your custom applications. This action is required to address the deprecation of various XML APIs and standard Java packages, and to ensure full compatibility with the modular architecture introduced in JDK 17.0.

System Information Page

The system information page provides basic information about VXML Server including the license information, the deployed gateway adapters and applications, the status of information on the application server on which VXML Server is running, and some miscellaneous system and Java information such as the version and memory usage. It does not provide the ability to run any functions, it is meant to be a quick way to check relevant information. It is also the easiest of the three methods to obtain information because all that is needed is a web browser. The system information page can be seen by pointing a web browser to the URL:

```
http://[HOST][:PORT]/CVP/Info
```

Where:

- HOST is the host name of the machine on which VXML Server is installed.
- PORT is the port the application server is configured to listen on. The default port for HTTP is 7000.

The first time you access the server information page you must configure a username and password. After you create the username and password you can log in and view the page.

The system information page is the easiest and safest way of obtaining administrative information, although it is also the least flexible.

Administration Information

Using the tools listed, an administrator can obtain a significant amount of information regarding VXML Server and the applications that are deployed on it. This information aids the administrator in determining the health of the system, detecting signs of issues that should be caught early, and debugging issues as they occur.

Much of the information made available by VXML Server can be found only through the JMX interface as that is the strength of JMX. Some of the more important information is available by using scripts and some of the static information is available through the system information page.

Application and System Status

VXML Server provides functions for reporting the status of a specific voice application or all voice applications running on the system. They are provided as functions to allow the administrator to query VXML Server to get the latest information immediately.

The application status function reports the following information:

- Whether the application is running, suspended, or has been suspended before being slated for removal.
- How many active sessions are currently visiting the application. Active sessions are defined as the number of callers that are interacting with the application at the time the status script is called.
- How many sessions are waiting to end. When an active caller ends their application visit, VXML Server delays the closing of the corresponding session to allow the completion of the session accessed by the final logger and end of call class actions. A session waiting to end does not take up a license port. The amount of time a session remains open after a call ends is a VXML Server configuration option .
- How many open sessions are experiencing the most recent past version of the application. Open sessions are the sum of active callers visiting the application and those sessions that are in the process of ending. The reason open sessions are listed here is because both active and ending sessions do need access to session information and an administrator would need to know when it is safe to disable any systems that the old application configuration depends on. This information is helpful for an administrator when performing an application update or suspension in order to determine when the function is complete. See the following sections for more on updating and suspending applications.
- How many callers are on hold waiting to get into the application. A call that is received when the system has used up all the allowed sessions defined in the license will hear a message asking them to stay on the line. This call then checks if a license session has become available and then lets the call into the application.

The VXML Server status function provides an easy-to-read report with the following information:

- If VXML Server itself has been suspended, this fact is listed first. See the following sections for more on suspending VXML Server.
- The total number of concurrent active callers visiting applications on this instance of VXML Server, how many concurrent sessions the license allows, the number of available ports (the license sessions minus the active callers), and the number of callers on hold (which would only appear if the number of current callers exceeds the number of license sessions).

Note This data is not available in case of a standalone deployment.

- How many active callers, sessions ending, and callers on hold for each application currently deployed on the system. This data is the same as would be displayed by the application-specific status function.

Note No on hold column will appear unless there are callers on hold.

- Whether each application is running or suspended.

JMX Interface

To get an application's status using the JMX interface, use a JMX client connected to the server to navigate to the `VoiceApplication/APPNAME/Command` MBean, where `APPNAME` is the name of the application to update. The **Operations** tab of this MBean will list a function named *status*. Pressing **Status** button will display a dialog box with the application status. To get the status of all applications using the JMX interface, navigate to the `Global/Command` MBean and click the function named *status* in the **Operations** tab. Pressing **Status** button will display a dialog box with the status of each application deployed on VXML Server in a table.

Administration Scripts

The script for obtaining an application status is found in the `admin` folder of the application to be updated. Windows users should use the script named `status.bat` and Unix users should use the script named `status.sh`. The script for obtaining the status of all applications is found in the `admin` folder of VXML Server. Windows users should use the script named `status.bat` and Unix users should use the script named `status.sh`. The scripts do not take any parameters.

VXML Server Information

VXML Server reports information about itself that is static so the administrator knows exactly what is installed. The following information is reported:

- The exact name and version of VXML Server.
- The expiration date, number of ports, and the supported gateway adapters listed in the VXML Server license.

Note The gateway adapter list is not a comprehensive list of the adapters installed on VXML Server but is a list of the gateway adapters the license allows the system to use.

- A detail of the version numbers of all components included with VXML Server. This information can be helpful for tracking changes made to individual components of the software installed at different times and this detailed information will typically be requested by Cisco support representatives when a question is raised about the software.

  The components whose versions are displayed are:

- The VXML Server web application archive (WAR) and the components residing within %CVP_HOME%\VXMLServer\.

Note This version is different from the VXML Server product version as that is a version for the whole system and this one is only for the WAR file.

- The core VXML Server elements, Say It Smart plug-ins, and loggers (both application and global) included with the software.
- The Gateway Adapters installed on the system.

JMX Interface

To obtain VXML Server information using the JMX interface, navigate to the `Info` MBean. The attributes tab displays all the information listed. To see all the gateway adapters supported in the license, you must open the value for the *LicensedGWAdapters* attribute (in JConsole this is done by double-clicking the value). The same procedure applies for obtaining the component versions by opening the value for the *ComponentVersions* attribute.

Administration Scripts

The only VXML Server information available by using a script is the versions of the components installed on VXML Server, though the name and version of VXML Server is displayed when it initializes and the license ports is always displayed using the global status script.

The script is found in the admin folder of VXML Server. Windows users should use the script named `getVersions.bat` and Unix users should use the script named `getVersions.sh`. In order to report on the version of the VXML Server web application archive (WAR), the script should be passed as an argument the full path of the WAR location (for example, %CVP_HOME%\VXMLServer\Tomcat\webapps\).

System Information Page

The same information is displayed in the system information page at the top of the table. It will also provide a list of the applications deployed on VXML Server as well as information on the application server, operating system, and Java memory usage.

Server Status Checks

Many load balancers can be configured to periodically access a URL that is used to determine if a server is running. These load balancers make a request to the URL, and if a response comes back within an acceptable time period, they consider the server available to handle connections. To determine the health of VXML Server, include the parameter `probe=true` in the request URL, using one of the following formats:

```
http://[DOMAIN][:PORT]/CVP/Server?probe=true
```

OR:

```
http://[DOMAIN][:PORT]/CVP/Server?application=[APPLICATION]&probe=true
```

The first URL format (*without* the `application` parameter) results in a simple HTML page with the following text if the VXML Server is accessible and is not suspended:

```
The Cisco Unified CVP VXML Server is up and running
```

However, if it is suspended (using the `suspendServer` administrative script), it will respond with:

```
The Cisco Unified CVP VXML Server is running, however it has been suspended.
```

This URL format has several optional parameters that may be used in conjunction with it:

- `activeCalls=true`

  This optional parameter causes the response HTML to include information about how many call sessions are active on the VXML Server instance. This is formatted as illustrated in the following example:

  ```
  running;activeCalls=12;
  ```

- `onHoldCalls=true`

  This optional parameter causes the response HTML to include information about how many call sessions are in an *on hold* status on the VXML Server instance. This is formatted as illustrated in the following example:

  ```
  running;onHoldCalls=3;
  ```

- `activeCalls=true&onHoldCalls=true`

  Specifying both of the optional parameters results in both data items being returned, as illustrated in the following example:

  ```
  running;activeCalls=77;onHoldCalls=0;
  ```

The second URL format (*with* the `application` parameter), results in a VoiceXML page which includes a `<submit>` to the listed voice application. If that VoiceXML page is returned, then VXML Server is accessible. This format is intended for use with load balancers that require the probe URL to match the URL through which actual content is retrieved. This format cannot be used to obtain additional information (that is, active and on-hold calls).

VXML Gateway Adapter

Gateway adapters are small plug-ins installed on VXML Server that provide compatibility with a particular Voice Browser. Once installed, all Unified CVP voice elements (and all custom voice elements not using browser-specific functionality) work on that Voice Browser.

Starting from Release 10.5 onwards VXML Server supports the following gateway adapters:

- **Cisco DTMF:** Generates the grammar for DTMF detection at Cisco Gateway.
- **VXML 2.1 with Cisco DTMF:** Generates the grammar for DTMF detection at Cisco Gateway using VXML 2.1 tags.
- **Nuance 10:** Generates the grammar for Speech and DTMF detection on the Nuance 10 server.
- **VXML 2.1 with Nuance 10:** Generates the grammar for Speech and DTMF detection on the Nuance 10 server using VXML 2.1 tags
- **Speech:** Generates the grammar for Speech and DTMF detection on the SpeechWorks server.
- **VXML 2.1 with Speech:** Generates the grammar for Speech and DTMF detection on the SpeechWorks server using VXML 2.1 tags

Note Nuance 10 or Speech adapter can process the grammars that are present on Nuance 10 or Speech server respectively; however, Cisco DTMF adapter can process the grammars that are present locally on the Cisco IOS gateway.

The following table provides the gateway adapter mapping to be used, while migrating from older version of Call Studio to Release 10.5(1) and onwards.

Adapters Prior to Release 10.5	New adapters from Release 10.5 onwards
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 with Cisco DTMF	Cisco DTMF
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 VoiceXML 2.1 with Cisco DTMF	VXML 2.1 with Cisco DTMF
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 with Nuance 8.5	Nuance 10
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 VoiceXML 2.1 with Nuance 8.5	VXML 2.1 with Nuance 10
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 with OSR 3	Nuance 10
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 VoiceXML 2.1 with OSR 3/Nuance 9	VXML 2.1 with Nuance 10
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 with Speech	Speech
Cisco Unified CVP 4.1/7.0/8.0/8.5/9.0 VoiceXML 2.1 with Speech	VXML 2.1 with Speech

Configuration Updates

When an administrator monitors a VXML Server installation, they want to be aware of any warning signs that the system is overloaded. In these scenarios, it is advantageous if the administrator can alter a few settings to better handle the given load without worrying about updating or suspending applications or shutting down the Java application server. These chnages may enable a system to better handle spikes in call activity with no adverse effects. To this end, VXML Server exposes some of its configuration options and allows an administrator to change them at runtime. It also allows the administrator to change some application settings values for deployed application.

Note It is *important* that the administrator be very careful when altering these configuration options at runtime as improperly chosen values could make the system unstable and achieve the opposite effect than desired.

The ability to change VXML Server configuration options and application settings is available only through the JMX interface. The configuration options are exposed as attributes of an MBean, one for the VXML Server configuration options and one for each application's settings. Those attributes that allow their values to be changed will have editable values. When a new value is given, it takes affect immediately with no confirmation so it is important to ensure that the value entered is correct. There is some simple validation that takes place by VXML Server and if the value entered is inappropriate (such as entering -1 where a positive integer is required), the change will not take place and the original value will remain unchanged. The administrator will know that their entry was accepted if the value does not revert back.

Note It is *very important* that any changes made to these attributes are **not persisted**. The changes affect VXML Server in memory and do not affect the XML files that hold these values. As a result, should the Java application server or the VXML Server web application be restarted or for application-specific attributes the application is updated, the attributes will revert back to the values specified in their respective XML files.

VXML Server Configuration Options

To view the VXML Server configuration options using the JMX interface, navigate to the `Global/Configuration` MBean.

There are five attributes listed.

- The first, named *LoggerEventQueueSize*, will show the current size of the queue that holds logger events waiting to be sent to loggers and is not editable.
- The *next three* are related in that they control aspects of the logger thread pool.
- The final configuration option deals with a period of time VXML Server waits after a caller ends their call before the call session is invalidated. All of these options affect the performance of the system and are defined fully in VXML Server Configuration.

Use the following table to reference the JMX attribute name with the `global_config.xml` tag name.

JMX Attribute Name	Tag Name
LoggerMaximumThreadPoolSize	`<maximum_thread_pool_size>` in the `<logger>` tag
LoggerMinimumThreadPoolSize	`<minimum_thread_pool_size>` in the `<logger>` tag
LoggerThreadKeepAliveTime	`<keep_alive_time>` in the `<logger>` tag
SessionInvalidationDelay	`<session_invalidation_delay>`

Tuning Logger Options

The most important indication of whether VXML Server is encountering issues with loggers is the *LoggerEventQueueSize* attribute. In order to prevent logging from holding up calls, all logging is done in separate threads. The threads are managed within a thread pool, which has a maximum and minimum value. When VXML Server starts up, the thread pool allocates the minimum number of threads. As calls begin to be handled, they generate logger events, which are put into a queue of events.

The activation of a logger event also prompts VXML Server to request a thread from the pool and in that thread have the appropriate logger handle the top most event in the queue. The length of time this thread handles the event depends on the logger, but the event is typically handled in a very short period of time, measured in milliseconds. However as call volume on the system increases, more threads are used simultaneously to handle the increase in logger events added to the queue.

As more threads are needed, the thread pool grows until it reaches the maximum number of threads allowed. At that point the queue would grow until threads become available. Threads that complete their work and cannot find new logger events to handle because the queue is empty will be garbage collected after a certain amount of time being idle (this is governed by the *LoggerThreadKeepAliveTime* option).

Under typical operation, the logger event queue size should not be a large number (one might see it set to 0 to 10 most of the time). There could be spikes where the queue grows quickly but with plenty of available threads to handle the events, the queue size should shrink rapidly. The administrator should take note if the queue size shows a high number, though should be very wary if this number seems to grow over time (minutes, not seconds). A growing queue size is an indication that either the load on the system is too high for the thread pool to handle (which is more likely the smaller the maximum thread pool size is set) or for some reason loggers are taking longer to do their logging. In the latter case this could be due to a slow database connection, overloaded disk IO or other reasons. Regardless of the cause, a growing queue is a warning sign that if the call volume is not reduced, the Java application server is at risk of encountering memory issues and, in the worst case, running out of memory.

It is for this reason that choosing an appropriate maximum thread pool size is important. While the temptation to give the maximum number of threads a very high number this can also cause problems on the system as severe as memory issues. Using too many threads could cause what is called *thread starvation* where the system does not have enough threads to handle standard background processes and could exhibit unpredictable inconsistent behavior and could also cause the Java application server to crash.

The JMX interface supports the ability to change the maximum and minimum thread pool size at runtime. The administrator should only do this if they believed the change could avert an issue listed above. For example, if the system is encountering a temporary spike in activity and the administrator sees the LoggerEventQueueSize attribute report a growing number, then they can increase the maximum thread pool size to potentially allow for a more rapid handling of the queued events. Once the queue shrinks to a manageable number the maximum thread pool size can then be changed back to its original value.

The maximum number of threads set by default in VXML Server is sufficient to handle a very heavy load without issues so the administrator is urged to use caution when changing these values.

Session Invalidation Delay Option

The session invalidation delay option is also an important value that an administrator could be tuned should they see the need. A brief explanation of what this option does is warranted . When a caller ends the call by either disconnecting, going to another application, or the application ends up the connection with the caller, VXML Server must perform some final clean up of the call session. This is primarily for processing logging events that occurred when the call ended. Additionally, application developers can configure their applications to run code at the end of a call to perform their own clean up operations. In sophisticated applications this could involve closing database connections or generating call detail records. These end-of-call operations can take a non-trivial amount of time and may require access to information about the call session, such as element or session data. As a result, VXML Server waits for a preset period of time after a call ends before it invalidates the session, allowing all activities requiring additional time to complete. This period of time is governed by the `SessionInvalidationDelay` attribute and is measured in seconds.

It is important to understand the consequences of changing this value. If too low a time is given then there could be situations where the system under load cannot handle the end of call tasks in the given time and the global error log may see many errors containing the Java exception `IllegalStateException` which occurs when attempting to access data from an invalidated call session. One has to understand that system resources are limited and when it is under load what may have taken 100ms to complete could take longer and depending on what it is that needs to be done, could take much longer.

The administrator should refrain from the temptation of making this number too large. This is because while a call session is still valid but not representing a live call, all that information remains in memory. This may not be much but could be significant depending on the amount of data stored in element and session data by the application. Even though the session has not been invalidated, since the call has ended, VXML Server is ready and will accept new calls, which will allocate additional memory. Under high load, the Java application server could encounter memory issues if call sessions remain in memory for too long a period.

The JMX interface supports the ability to change the session invalidation delay at runtime. The administrator would increase this setting if `IllegalStateException` errors appear in the logs. They would lower the value if the JVM memory usage stays close to the maximum after each garbage collection. Keep in mind that there are many potential causes for JVM memory utilization to rise and is certainly not limited to this cause.

The default value of the session invalidation delay is sufficient to handle a heavy load without issues so the administrator is urged to be cautious when changing this value.

Application Configuration Options

To view the configuration options of an application using the JMX interface, navigate to the `VoiceApplication/APPNAME/Command` MBean, where `APPNAME` is the name of the application. There are four attributes listed:

- **DefautAudioPath** - Shows the audio path defining where the audio files are located (assuming the application was designed to take advantage of it).
- **GatewayAdapter** - Shows the gateway adapter that the application is using and is not editable. It is for informational purposes only.
- **SessionTimeout** - Shows the length of time, in seconds, of inactivity to consider a call session timed out.
- **SuspendedAudioFile** - The path for the audio file to play to callers when calling into an application that is suspended.

An administrator may choose to change the default audio path of an application at runtime should there be a need to change the audio callers hear quickly. One use case would be if the server that hosts the audio files is being restarted and the administrator wishes all audio to be fetched from a backup server.

Note The effectiveness of this change will be based on how consistently the application was designed to use the default audio path and also if the application explicitly sets the default audio path itself, which would override the value passed here.

An administrator may choose to change the session timeout value at runtime as part of the process of debugging a problem. Under ordinary circumstances no session should time out because the voice browser and VXML Server should be in constant communication regarding when a call starts and ends. An administrator experiencing some sessions timing out may choose to increase this attribute to see if it resolves the issue and if not, should look into network issues. The administrator should be careful not to set this value too small a number because there is a risk that an ordinary call could time out due to the caller visiting a particularly large VoiceXML page or taking their time entering a long DTMF input. Too large a number will mean that sessions that are no longer valid will remain in memory longer and the administrator would not be able to see which sessions are timed out until the timeout period elapsed.

An administrator may choose to change the suspended audio file at runtime if the application needed to be suspended due to a specific reason. For example, if a weather event required an application to be suspended, the administrator could point the suspended audio message to a recording explaining why the application is suspended rather then just pointing to a generic message. The administrator is taking advantage of the fact that this change is not persisted since it is expected that the event that caused the application's suspension is temporary.

Administration Functions

VXML Server exposes several functions that allow an administrator to make both small and large changes to the applications and VXML Server at runtime. They are divided into two categories: those that affect a specific application and those that affect all applications running on VXML Server. An administrator can use the JMX interface as well as administration scripts to run these functions.

Each administrator function, when activated, prompts VXML Server to send a logger event reporting the function and its result so that any loggers listening to these events can log the information. The logs will then maintain a history of administration activity that can be analyzed later.

Administrator functions include the ability to add, update, and remove applications as well as suspend both an application and VXML Server itself. This section describes all functions available.

Graceful Administration Activity

Administration functions are used primarily to alter an application, whether it be to update its contents or suspend its activity. Whenever changes are made to a live system handling callers, a concern is how these changes affect live callers. A robust, reliable system should strive for maximum uptime and minimal disruptions of live calls, and VXML Server does this by implementing a process for managing changes.

In the process, existing callers continue to experience the application as it existed before the change, while new callers experience the change. Only after all existing callers have naturally ended their calls will the change apply to all live callers. At this time, VXML Server will perform any necessary cleanup required to remove the old application configuration. In this manner, changes can be made to applications at any time; the administrator does not need to worry about the impact of the change on live callers as the transition will be handled.

Due to the interactive nature, when using administration scripts to perform graceful functions, the script will display a count down of callers that are actively visiting the application as they end their calls. This is provided because an aid to the administrator in determining how many callers are still experiencing the application before the change. Command-line arguments passed to the scripts can turn off this countdown if desired.

When using the JMX interface or if the countdown is turned off in the administration script, the only way to track the number of callers that are still experiencing the old configuration would be to get the system status.

Applications Update

Occasionally, an application will need to be updated. Possible changes can be small, such as renaming an audio file or altering a TTS phrase, or large, such as adding another item to a menu and creating a new call flow branch. They can involve simple configuration changes or may involve new or changed Java class files. While most changes are implemented during development time, there is a requirement to support updating an application at runtime.

The update functionality acts gracefully in that any callers on the system, at the time of update continue to experience their calls as if the application had not been updated, while new callers experience the updated voice application. In this manner, there is no downtime when a change is implemented for an application, the callers are handled as expected.

VXML Server exposes an update function for every application deployed. This Server will update only that application. It also has a function that updates all applications at once.

When updating individual voice applications note these guidelines:

- The update applies only to those resources controlled by VXML Server. These include the application settings and call flow, element configurations, Unified CVP decision elements, and Java classes placed in the `java/application` directory of the application. The following changes are not managed by VXML Server and therefore will not be updated:
  - Java classes placed anywhere else (including the `common` folder).
  - XML content passed to VXML Server by using the XML API.
  - The content of VoiceXML insert elements.
  - Other applications that the updated application transfers to or visits as part of a subroutine.
  - External back-end systems such as web services and databases (including the user management database).
  - Web servers hosting static content used by the application such as audio or grammar files.

  When each of these resources become unavailable or change, all callers would be affected. For small changes such as a revised audio file, this situation may be acceptable. For large-scale changes that span multiple systems, this might cause problems such as callers who are visiting an application when the update is made experiencing an error because a database is down.

  For large changes, the application should be suspended and the changes made once all callers have left the system (see the following section on suspending applications). Once the application is fully suspended, the administrator is free to make the changes and when done, the application should be updated followed by resuming it from its suspended state. This way, no caller will be in the system when the changes are made. The only disadvantage to this approach is that it will make the application unavailable for a period of time as opposed to a transparent change if the update feature alone is used. This may be a necessary compromise considering the consequences.
- When the update occurs, the event created by VXML Server to send to any loggers that are listening will reflect when the update function was run, not when it completed.
- If an error occurs during the update process, for example, due to an incorrectly configured XML file, a description of the error is displayed and sent to any loggers listening to the appropriate logger events and the update is cancelled.
- If an updated VXML server application is transferred from OAMP to VXML Servers while calls are in progress, the following message appears in the log:

  ```
  %CVP_8_5_VXML-3-VXML_INTERNAL_ERROR:DatafeedMgr.handleActEvent()
  java.lang.NullPointerException]
  ```

Note In this case, calls are not disconnected, but the related VXML events during the update will not be sent to the reporting server.

JMX Interface

To update an application using the JMX interface, use a JMX client connected to the server to navigate to the `VoiceApplication/<APPNAME>/Command` MBean, where `APPNAME` is the name of the application to update.

The **Operations** tab of this MBean will list a function named *updateApplication*. Pressing **updateApplication** button will cause the application to be updated and the result of the update will be displayed in a dialog box.

The administrator should be aware that there is no confirmation when this function is called, the update happens immediately once the function is processed.

Note While the function returns immediately, the old application may still be active if there were calls visiting the application at the time of the update. Only when all existing callers end will the old application configuration be removed from memory. To determine when that occurs, use the status function.

To update all applications at once using the JMX interface, navigate to the `Global/Command` MBean and click the function named *updateAllApps* in the *operations* tab. The results are displayed in a dialog box, listing each application updated. As with the application-specific update, use the status function to determine if there are callers experiencing old versions of the applications.

Administration Scripts

The scripts for updating an application are found in the `admin` folder of the application to be updated. Windows users should use the script named `updateApp.bat` and Unix users should use the script named `updateApp.sh`.

The script will first ask for confirmation of the desired action to prevent accidental processing of the script. To turn off the confirmation, pass the parameter *noconfirm*. By default, the script does not return to the command prompt until all pre-update callers are finished. Interrupting the countdown will not stop the update process, only the visual countdown. To turn off the countdown, pass the parameter *nocountdown*. If the countdown is interrupted or the script is passed with the nocountdown parameter then the only way to determine how many callers are experiencing the old application is to run the status script for the system, which displays this information.

The script to update all applications is found in the admin folder of VXML Server. Windows users should use the script named `updateAllApps.bat` and Unix users should use the script named `updateAllApps.sh`. The script behavior is the same as if the update script for each application deployed on VXML Server were run in series.

The `updateAllApps` script also displays a confirmation prompt, which can be turned off by passing the `noconfirm` parameter. Unlike the `updateApp` script, the `updateAllApps` script does not display a countdown of callers, it lists all the applications that are updated. The administrator would need to run the status function to determine how many callers are visiting the old versions of the applications.

Applications Suspension

There are many situations when an application needs to be temporarily suspended. There could be scheduled maintenance to the network, the voice application could have an expiration date (say it runs a contest that must end at a specific time), or the application is to be turned off while enterprise-wide improvements are made. There may also be situations where all applications are to be put in suspension if modifications are being made that affect all applications. In each of these situations, a caller would need to be played a designer-specified message indicating that the application has been temporarily suspended, followed by a hang-up. This is preferable to not answering or taking down the system, which would cause a cryptic outage message to be played.

First, the application designer defines the suspended message in the Application Settings pane in Builder for Call Studio. When the suspend order is given, VXML Server produces a VoiceXML page containing this suspended audio message to all new calls followed by a hang-up. Because VXML Server allows all calls

currently on the system to finish as usual when the command was issued, existing callers are unaware of any changes. VXML Server will keep track of the active callers visiting the application and make that information available for the administrator to access. Only when this number reaches 0 will it be safe for the administrator to perform the system maintenance that required the suspension.

VXML Server exposes suspend and resume functions for every application deployed that acts on just that particular application. It also exposes a function that will suspend VXML Server itself, which has the effect of suspending all applications. A separate resume function resumes VXML Server that restores the previous state of each application. So if an application was already suspended when VXML Server was suspended, resuming VXML Server leaves the application in a suspended state.

There are a few items to note when suspending a voice application:

- Only when all existing callers have exited the system will the application be officially suspended. Depending on the average length of calls to the voice application, this may take some time. The application status will appear as suspended since new callers cannot enter the application, and they will hear the suspended audio message.
- If changes were made to an application while it was suspended, the application should first be updated before being resumed (see the previous section on the update administration function).
- The suspension applies only to those resources under the control of VXML Server. External resources such as databases, other web servers hosting audio or grammar files, or servers hosting components through XML documents over HTTP are accessed at runtime by VXML Server. If any of these resources become unavailable while there are still presuspension callers on the system, those calls will encounter errors that will interrupt their sessions. Any maintenance made to backend systems should be initiated after the application status shows that all presuspended callers are finished with their calls.
- When the suspension occurs, the event created by VXML Server to send to any loggers that are listening will reflect when the suspend function was run, not when it completed.
- If an error occurs during suspension, a description of the error is displayed and sent to any loggers listening to the appropriate logger events and the update is cancelled.
- Suspending a voice application still requires VXML Server (and the Java application server) to be running in order to produce the VoiceXML page containing the suspended message. If the application server itself requires a restart, there are four possible ways to continue to play the suspended message to callers. Remember to run the suspend function before any of these actions are taken as this is the prerequisite.

  The solutions are listed in order of effectiveness and desirability:

  - **Load balance multiple instances of VXML Server**—In a load-balanced environment, one machine can be shut down, restarted, or reconfigured while the rest continue serving new calls. Once removed from the load-balance cluster, a machine will not receive new call requests. Eventually, all existing callers will complete their sessions, leaving no calls on the machine removed from the cluster. That machine can then be safely taken down without affecting new or existing callers.
  - **Use a web server as a proxy**—In a smaller environment, a web server can be used as a proxy for an application server so that when that application server becomes unreachable, the web server itself can return a static VoiceXML page containing the suspended message to the voice browser. The web server does not need to be on the same machine as the application server. Once the web server is configured, VXML Server can be suspended to locate all existing callers, then the application server can be taken down and the proxy server will take over producing the suspended message VoiceXML page. The disadvantage of this approach is that the web server setup is done outside of Unified CVP and if the suspended message changes, it needs to be changed in both the Builder for Call Studio and the web server configuration.

- **Redirect the voice browser**—The voice browser can be configured to point to another URL for calls coming on the specific number. This can point to another machine running VXML Server or even just a web server with a single static VoiceXML document playing the suspended message. A separate file would be needed for each application. This is a manual process and requires another machine with at least a web server (it can be on the same machine which would allow the Java application server to be restarted but would not allow the machine itself to be restarted).

JMX Interface

To suspend an application using the JMX interface, use a JMX client connected to the server to navigate to the `VoiceApplication/APPNAME/Command` MBean, where `APPNAME` is the name of the application to be suspended. The **Operations** tab of this MBean will list a function named *suspendApplication*. Pressing **suspendApplication** button causes the application to be suspended and the result is displayed in a dialog box. To resume the application, select the function named *resumeApplication*. The result is displayed in a dialog box.

The administrator should be aware that there is *no confirmation* when these functions are called, the suspension and resumption occurs immediately after you run the functions.

Note While the suspend function returns immediately, the application may still be active if there were calls visiting the application at the time of the suspension. Only when all existing callers end their calls will the application be fully suspended and the administrator is safe to take down any resources that the application depends on. To determine when all calls have ended, use the status function.

To suspend VXML Server itself using the JMX interface, navigate to the `Global/Command` MBean and click the function named *suspendVXMLServer* in the **operations** tab. The results will be displayed in a dialog box. As with the application-specific suspension, use the application-specific status function to determine if there are callers still visiting the applications. Click the function named *resumeVXMLServer* to resume VXML Server and restore the previous states of the applications.

Administration Scripts

The scripts for suspending and resuming applications are found in the `admin` folder of the application to be suspended. Windows users should use the script named `suspendApp.bat` and Unix users should use the script named `suspendApp.sh`. To resume the application, use the script named `resumeApp.bat` or `resumeApp.sh`.

It is possible to suspend all applications at once by accessing a script found in the `admin` folder of VXML Server. Windows users should use the script named `suspendServer.bat` and Unix users should use the script named `suspendServer.sh`. To restore all applications to their original status, use the script named `resumeServer.bat` or `resumeServer.sh`.

Note These scripts do not resume all applications; they only restore the administrator-specified status of each application. So if an application was already suspended when the server was suspended, resuming the server leaves the application in a suspended state.

Applications Addition

When VXML Server starts up, it will load all applications that have been deployed to its `applications` folder. A new application that is created in Builder for Call Studio and deployed to a machine on which VXML Server is already running cannot begin accepting calls until VXML Server loads the new application. To load the application, run the deploy application function. If the application is already deployed, running this function will do nothing. If multiple new applications are to be deployed together, one can run the deploy all applications function and all new applications will be deployed, and leave existing applications untouched.

JMX Interface

To deploy all new applications using the JMX interface, use a JMX client connected to the server to navigate to the `Global/Command` MBean and click the function named *deployAllNewApps* in the **Operations** tab. Pressing **deployAllNewApps** button displays a dialog box with the status of each application's deployment.

Alternatively, to deploy a single new application, first use the function named *listAllNewApps* in the **operations** tab to get a list of new application names. Then use the *deployNewApp* function to deploy the desired application by name.

Administration Scripts

The script for deploying a specific application is found in the `admin` folder of the application to be deployed. Windows users should use the script named `deployApp.bat` and Unix users should use the script named `deployApp.sh`. The script for deploying all new applications at once is found in the `admin` folder of VXML Server. Windows users should use the script named `deployAllNewApps.bat` and Unix users should use the script named `deployAllNewApps.sh`.

Applications Removal

VXML Server demonstrate two administrative functions to handle the removal of application from memory at runtime. Determining which function to use will depend on the operating system and whether the application being removed is actively handling calls.

The first method involves running the release application function of the application to be removed. This function prompts VXML Server to first suspend the application then remove it from memory when all the active callers at the time the function was run, have naturally ended their sessions. It suspends the application first to prevent new callers from entering the application. Once all active callers are done visiting the application the folder of the application can be deleted (or moved) from the VXML Server `applications` folder. This function affects only a single application so if multiple applications are to be removed using this method, the administrator would have to run this function for each application.

Note On the Microsoft Windows operating system, a user attempting to delete an application folder after the releaseApp function is called may be prevented from doing so by the operating system if the application references Java application archive (JAR) files placed within the `java/application/lib` or `java/util/lib` directories. This is due to the system keeping an open file handle for JAR files that will not be released until a garbage collection event occurs. As a result, the administrator will have to wait until the garbage collector activates before being able to delete the directory. The time to wait will be determined by how often garbage collection is run. A rule of thumb is that a high load system or one with a small amount of memory will encounter garbage collection often, a low volume system or one with a large amount of memory will take longer.

The second method supports the ability to delete multiple applications at once. This time one must first delete (or move) the folders holding the desired applications to be deleted. After which, the flushing of all old applications function is run and VXML Server will suspend and then remove from memory of all the applications that it no longer finds in the `applications` folder. As with the other method, the application is not removed from memory until all callers have ended their visits.

These issues can occur with the second method:

- If an application relies on files found within its folder at runtime, there may be problems with existing callers reaching a point where these files are needed and they will not be found.
- This process may not work on Microsoft Windows because Windows will not allow the deletion of a folder when resources within it are open. For example, the application loggers may have open log files located within the application's `logs` folder. This may work if no loggers are used or the only loggers used are those that do not manage files stored in the `logs` folder.

JMX Interface

To delete an application using the JMX interface, use a JMX client connected to the server to navigate to the `VoiceApplication/<APPNAME>/Command` MBean, where `APPNAME` is the name of the application to update. The **Operations** tab of this MBean will list a function named *releaseApplication*. Pressing **releaseApplication** button will cause the application to be suspended and then removed from memory when all active callers visiting the application at the time the function was completed.

The administrator should be aware that there is no confirmation when this function is called; the application is suspended and removed from memory immediately once the function is completed.

Note While the function returns immediately, the application will remain active if there were calls visiting the application at the time of the release. Only when all existing callers end, the call will the application be removed from memory. To determine if there are active callers, use the status function.

To delete all applications whose folders have been removed from the `applications` folder of VXML Server using the JMX interface, navigate to the `Global/Command` MBean and click the function named *releaseAllOldApps* in the **operations** tab. The results will be displayed in a dialog box, listing each application deleted. As with the application-specific update, use the status function to determine when the callers finish their visits to the applications.

Administration Scripts

The scripts for deleting an application are found in the `admin` folder of the application to be updated. Windows users should use the script named `releaseApp.bat` and Unix users should use the script named `releaseApp.sh`.

The script first asks for confirmation of the desired action to prevent accidental processing. To turn off the confirmation, pass the parameter *noconfirm*. By default, the script does not return to the command prompt until all callers are finished with their calls. Interrupting the countdown will not stop the release process. To turn off the countdown, pass the parameter *nocountdown*. If the countdown is interrupted or the script is passed the *nocountdown* parameter, then the only way to determine how many callers are actively in the application is to run the status script for the system.

The script to release all applications whose folders have been removed from the `applications` folder of VXML Server is found in the `admin` folder of VXML Server. Windows users should use the script named `flushAllOldApps.bat` and Unix users should use the script named `flushAllOldApps.sh`. All applications whose folders have been removed will be suspended and when their active calls have ended will be removed from memory.

The `flushAllOldApps` script also displays a confirmation menu which can be disabled by passing it the *noconfirm* parameter. Unlike the `releaseApp` script, the `flushAllOldApps` script does not display a countdown of active callers, it will list all the applications that were deleted. The administrator would need to run the `status` function to determine how many callers are actively in the applications.

Common Classes Update

When performing an application update, all the data and Java classes related to an application will be reloaded. Java classes placed in the `common` folder of VXML Server are not included in the application update. VXML Server provides a separate administrative function to update the `common` folder.

There are a few items to note about this function:

- The update affects all applications that use classes in the `common` folder, so running this function could affect applications that have not changed. Therefore, take precaution when running this function.
- The update affects *all* classes in the `common` folder, whether they were changed or not. This is usually not a issue unless those classes contain information in them that reloading would reset (such as static variables).
- Due to the fact that this function reloads classes that affect all applications, and those classes may themselves prompt the loading of configuration files from each application that uses those classes, the function may take some time to complete depending on the number of classes in the `common` folder and the number and complexity of the deployed applications.
- Changes are immediate, and are not done. Because this potentially affects all applications, the administrator must be aware of this.

JMX Interface

To update common classes using the JMX interface, use a JMX client connected to the server to navigate to the `Global/Command` MBean and click the function named *updateCommonClasses* in the **operations** tab. The results will be displayed in a dialog box.

Administration Scripts

The script for updating common classes is found in the `admin` folder of VXML Server. Windows users should use the script named `updateCommonClasses.bat` and Unix users should use the script named `updateCommonClasses.sh`. The script will ask for confirmation of the desired action to prevent accidental processing of the script. To disable the confirmation, pass the parameter *noconfirm*.

Global and Application Data Function

Global data holds information that applications decide to share across other applications deployed on VXML Server. Application data holds information that applications decide to share across all calls to the application. The VXML Server JMX interface provides the ability for an administrator to view the contents of these variables, change their values, and even create new variables.

This functionality provides an administrator direct access to live data that is being created on the system and can provide them some control of how applications operate. This is only possible when the application designers design that functionality into the applications. For example, an application designer for a utility company can build their application to look for the existence of a global data variable reporting a power outage. The administrator then creates the global data variable when a power outage occurs and automatically the applications will start reporting the power outage to callers. The administrator can then delete the global data

variable to signify the power has been restored. While this same functionality could be achieved with a database, this is a simpler approach to handle predictable situations without the need to use a database.

Global Data Access

To access global data using the JMX interface, navigate to the `Global/Data` MBean. The **Attributes** tab lists all the global data variable names in an attribute named *AllGlobalDataNames* (the value may need to be expanded in order to see all the global data names). The **Operations** tab lists four functions that can be processed by the administrator for global data:

- **setGlobalData**—This function allows the administrator to create a new global data variable. The function requires the name of the variable and the value. Click **setGlobalData** button to set the global data and the result will appear in a dialog box.

Note If there exists global data with the same name it will be overridden.

- **removeGlobalData**—This function allows the administrator to delete a global data variable. The function takes one input: the name of the global data variable to delete. Click **removeGlobalData** button to remove the global data and the result will appear in the dialog box.
- **removeAllGlobalData**—This function allows the administrator to delete all global data, whether it was created by the administrator or applications. Click **removeAllGlobalData** button to remove all global data and the result will appear in the dialog box.

  Be careful when using this function because it can affect the performance of applications that rely on global data.
- **getGlobalData**—This function allows the administrator to retrieve the value of a global data variable. The function takes one input: the name of the global data variable to retrieve. Click **getGlobalData** button to display a dialog box showing the value of the global data.

Application Data Access

To access application data using the JMX interface, navigate to the `VoiceApplication/APPNAME/Data` MBean, where `APPNAME` is the name of the application whose application data is to be accessed. The *Attributes* tab lists all the application data variable names in an attribute named *AllApplicationDataNames* (the value may need to be expanded in order to see all the application data names). The **Operations** tab lists four functions that can be processed by the administrator for application data:

- **setApplicationData**—This function allows the administrator to create a new application data variable. The function takes two inputs, the first being the name of the variable and the second being the value. Click **setApplicationData** button to set the application data and the result will appear in a dialog box.

Note If there is already application data with the same name it will be overridden.

- **removeApplicationData**—This function allows the administrator to delete a application data variable. The function takes one input: the name of the application data variable to delete. Click **removeApplicationData** button to remove the application data and the result will appear in the dialog box.

- **removeAllApplicationData**—This function allows the administrator to delete all application data, whether it was created by the administrator or applications. Click **removeAllApplicationData** button to remove all application data and the result will appear in the dialog box. Be careful with this function as it could affect the performance of the application.
- **getApplicationData**—This function allows the administrator to retrieve the value of a application data variable. The function takes one input: the name of the application data variable to retrieve. Click **getApplicationData** button to display a dialog box with the value of the application data.

Administrator Log Access

VXML Server includes various default loggers, including administration history loggers that store a history of the administration activity taken, such as when VXML Server started up, when an application was updated, the results of the suspension of VXML Server, and so on. These logs, which are rotated daily, are useful to an administrator as an audit history of administrator activity. As a convenience, the JMX interface exposes methods for the administrator to access the contents of these logs instead of viewing the files in a text editor.

Note The application designer and administrator has the ability to define any loggers desired for the applications as well as for VXML Server, including removal of the default administration history loggers. If this is done, then these functions will return error messages that explain that the log files could not be found.

To view an application's administration history log using the JMX interface, use a JMX client connected to the server to navigate to the `VoiceApplication/<APPNAME>/Command` MBean, where `APPNAME` is the name of the application to view. The **Operations** tab of this MBean lists functions named *retreiveAdminHistoryToday* and *retreiveAdminHistoryAll*. Clicking the first opens up a scrollable window listing the contents of the administration history log file from the day the function is called. Clicking the second opens up a scrollable window listing the contents of all administration history logs concatenated.

To view VXML Server administration history log using the JMX interface, navigate to the `Global/Command` MBean. The **operations** tab of this MBean list functions with the same name and functionality as the application functions do except that the files accessed are for the global administration history.

Administration Function Reference

The following tables list all the administration functions provided by VXML Server and whether they are available from the JMX interface and via script.

Application-Level Functions

Function	JMX	Script	Description
Suspend Application	Yes	Yes	Suspends the application in which the function belongs.
Resume Application	Yes	Yes	Resumes the application in which the function belongs.
Deploy Application	No	Yes	Prompts VXML Server to load the application in which the function belongs (does nothing if the application is already deployed).

Update Application	Yes	Yes	Prompts VXML Server to reload into memory the configuration of the application in which the function belongs.
Release Application	Yes	Yes	Prompts VXML Server to remove from memory the application in which the function belongs so that its folder can be deleted.

VXML Server-Level Functions

Function	JMX	Script	Description
Suspend Server	Yes	Yes	Suspends all applications deployed on VXML Server.
Resume Server	Yes	Yes	Restores the status of each application to the original state at the time VXML Server was suspended.
Deploy All New Applications	Yes	Yes	All applications deployed to VXML Server because the last time the application server started up or the deploy all new applications function was called are now loaded into memory and can handle calls.
List All New Applications	Yes	Yes	Lists the names of all new voice applications so that their names may be known to be deployed using Deploy New Application.
Deploy New Application	Yes	No	Loads and deploys the specified voice application.
Flush All Old Applications	Yes	Yes	When called, all applications in VXML Server whose folders were deleted are removed from memory.
Update All Applications	Yes	Yes	Prompts each application deployed on VXML Server to load its configuration from scratch from the application files.
Update Common Classes	Yes	Yes	Reloads all classes deployed in the `common` directory of VXML Server.

VXML Server Metrics

The more information administrators have, the better they will be able to determine the health of the system. VXML Server provides a significant amount of information on various metrics to allow the administrator to understand what is going on within the system. Provided with this information, the administrator will be able to react quickly to situations that could degrade the stability of the system.

The information falls into three categories: aggregate information, information on peaks, and average information. Aggregate information, such as the total number of calls handled, is helpful in determining how much work VXML Server has done so far. Peak information, such as the maximum concurrent calls occurring in the last 10 minutes, is very helpful in understanding how load is distributed on the system and can help the administrator understand how the volume is changing. Average information, such as the average HTTP request completion time, helps the administrator compare current metrics against historical averages.

The metrics maintained by VXML Server is available only through the JMX administration interface. To view the metrics, navigate to the `Global/Metrics` MBean. The **Operations** tab lists 15 separate functions that the administrator can call to obtain very specific information concerning how the system is running as well as how it has performed in the past. Many of the functions take a time duration as an input. It will display information of the specified period up to a maximum of 60 minutes.

The following list describes each function and the information it returns:

- **totalCallsSinceStart**—Returns the total cumulative number of calls handled by VXML Server since it launched. This number will continually rise and only resets only when VXML Server or the Java application server is restarted.
- **maxConcurrentCallsInLast**—Returns the most number of simultaneous callers that occurred in the last X minutes where X is entered by the administrator (maximum of 60 minutes) and when the maximum was reached. This count is helpful in determining how close the call volume reached the license limit on simultaneous callers. Knowing when the maximum value is reached can be very helpful in determining if call volume is rising. For example, if the peak call volume for the last 10 minutes was achieved very close to the present time, that would indicate that call volume is rising.
- **avgConcurrentCallsInLast**—Returns the average number of simultaneous callers encountered in the last X minutes where X is entered by the administrator. This data is helpful in determining if a peak was an isolated occurrence or a sign of a trend. For example if the maximum number of concurrent calls in the last hour was 100 but the average is 10, then there is less to be alarmed about since the 100 peak did not last long and can be attributed to a temporary spike. If the average were 90, then this would indicate that the call volume is very steady.
- **maxReqRespTimeInLast**—Returns the maximum time, in milliseconds, it took VXML Server to produce an HTTP response in the last X minutes where X is entered by the administrator and when the maximum was reached. A voice browser makes an HTTP request to VXML Server, which then must respond with a VoiceXML page. A large response time is cause for concern because a slow performing system will cause callers to think that the application has encountered errors. In extreme cases, this response time might cause the voice browser to time out a request and end a call with an error.
- **avgReqRespTimeInList**—Returns the average time, in milliseconds, that it took to produce an HTTP response in the last X minutes where X is entered by the administrator. This value gives the administrator a good idea of how long it takes VXML Server to handle responses given the call volume. This value could help the administrator decide if the system is overloaded and is beginning to affect the perception of callers regarding the responsiveness of the application. It also establishes a baseline to compare with the maximum response time. A maximum response time significantly higher than the average might be an indication that there is a problem with an external resource accessed by a custom element such as a database or web service and the few calls that visited that element suffered from bad performance.

  This function can also help determine if the maximum response time was an isolated event or an indication of a trend. For example if the maximum response time was 500 ms, which occurred near the present, the average was 400 ms, the fact that the peak was 500 ms is not alarming because the average is so high. In this situation, the administrator may choose to throttle down the calls being handled by the system to bring the response times back down to more acceptable levels.
- **timeoutCallsInLast**—Returns the total number of calls that ended with a timeout in the last *X* minutes where *X* is entered by the administrator. More specifically, this counts calls where the result action of the end category is timeout. See section *The Application Activity Logger* for more on the different results and how ended values. Under usual circumstances a call should never time out. Many different types of conditions can yield session timeouts on VXML Server and so knowing if there are timeouts in the last period of time would tell the administrator how widespread these issues are.

- **failedCallsInLast**—Returns the total number of calls that ended with an error in the last *X* minutes where *X* is entered by the administrator. More specifically, this counts calls where the result action of the end category is error. See section *The Application Activity Logger* for more on the different results and how ended values. This helps the administrator determine how widespread a bug or other issue that caused a call to end in an error is. For example, if the last 60 minutes yielded only one failed call, while the issue should be investigated, it may not be a symptom of a larger more prevalent issue.
- **timeoutCallsSinceStart**—Returns the total number of calls that ended with a timeout since VXML Server launched. More specifically, this counts calls where the result action of the end category is timeout. See section *The Application Activity Logger* for more on the different results and how ended values. This information is good to compare with the number of timed-out calls in the past *X* minutes because if the numbers are close it might mean that the issue that is causing the timeouts is a recent occurrence. It also gives an indication of the stability of the system and allows the administrator to calculate the percentage of calls that had encountered timeouts.
- **failedCallsSinceStart**—Returns the total number of calls that ended with an error since VXML Server launched. More specifically, this counts calls where the result action of the end category is error. See section *The Application Activity Logger* for more on the different results and how ended values. This information is good to compare with the number of failed calls in the past *X* minutes because if the numbers are close it could mean that the issue that is causing the errors is a recent occurrence. It also gives an indication of the stability of the system is and allows the administrator to calculate the percentage of calls that had errors.
- **maxLoggerEventQueueSizeInLast**—Returns the largest the logger event queue received in the last *X* minutes where *X* is entered by the administrator and when the maximum was reached. For an explanation of the logger queue, see the section titled Tuning Logger Options earlier in this chapter. This value helps the administrator to understand, in an abstract way, how much VXML Server is logging. While it is not unusual for this number to be large, the administrator can track a trend, and if this number continually increases it might be an indication that the system cannot handle the logger event load and might eventually result in memory problems. The time when the maximum was reached can help indicate if VXML Server is able to handle the incoming stream of logger events.
- **maxLoggerThreadCountInLast**—Returns the most simultaneous threads VXML Server was using to handle loggers in the last *X* minutes where *X* is entered by the administrator and when the maximum was reached. For an explanation of the logger thread pool, see the section titled Tuning Logger Options earlier in this chapter. This is another indication of whether VXML Server can keep up with the stream of logger events because if the number is close to the maximum thread pool size it is an indication that VXML Server has almost reached its limit in handling events. When the maximum was reached helps determine if this is happening recently. When all of the threads in the pool are actively handling logger events, the logger event queue will rise rapidly. So, if this value is at the maximum thread pool size, then the `maxLoggerEventQueueSizeInLast` function will display rapidly increasing queue sizes.
- **callTransferRate**—Returns the percentage of calls that ended in a blind telephony transfer. More specifically, this counts calls where the how action of the end category is `call_transfer`. This can help the administrator determine what percentage of callers decided to speak to an agent rather than complete the call in the automated voice application.
- **callAbandonRate**—Returns the percentage of calls that ended with the caller disconnecting the call. More specifically, this counts calls where the how action of the end category is *hangup*. See section *The Application Activity Logger* for more on the different results and how ended values. Despite the name, a caller ending the call is not necessarily a bad situation because the caller might end the call right before the application disconnects on the caller and the end category would still be *hangup*. This value would therefore be a good indication of how callers interact with the applications on the system.

- **callCompleteRate**—Returns the percentage of calls that ended ordinarily. More specifically, this counts calls where the result action of the end category is *normal*. See section *The Application Activity Logger* for more on the different results and how ended values. This function does not count calls into a suspended application, calls ending in an error or timeout, or calls ending due to an element manually invalidating the session. It is expected that this percentage be close to 100 percent.
- **averageCallDuration**—Returns the average duration of all calls handled by VXML Server, in seconds. This information helps the administrator determine if a particular call being analyzed represents a typical call since a particularly long call might indicate a caller having trouble with the application and a short call might indicate caller frustration with the application.