CHAPTER 15

Using the Core Logging API

The Core Logging API (CLA) allows both the Core (the object repository) and MDCs (Multiple Device Controllers) to save logging messages as well as auditing messages. It also allows Core and other MDCs to log to the same file from both Java and C++, and to be accessed seamlessly from either language. MDCs also use CLA to write to logs wherever necessary.

The Core Logging API (CLA) is accessible from both Java and C++. The Core Client Registry (CCR) contains information for each MDC that pertains to accessibility of certain types of logs as well as auditing. It also contains location information for the logs. The CLA accesses the information in the CCR.

Log4cpp is used as the logging API and simple static methods are created for users of the CLA to call. The Log4cpp API remains hidden to users of the CLA. A very simple java JNI interface is created to call the static CLA methods.

The following topics describe the CLA:

About the Core Logging API Structure

The Core Logging API consists of four parts:

1. **CoreLogger**: Communicates with the CCR to determine categories, priorities, and locations for MDC logs. Appropriate appenders are created for each MDC, depending on the location. They are attached to each category of the MDC with the specified priority. Priority levels are (ranked from lowest to highest):
   - DEBUG
   - INFO
   - WARNING
   - ERROR
   - FATAL

   AUDIT priority is used to separate levels from each other.
2. **Logger API**: Contains a series of simple static methods, which CLA users call to create logs. It is the access point for the CLA into the Log4cpp API.

3. **JavaLogger API**: Identical to the Logger API, but accessible from Java.
4. **Auditlog API**: A wrapper over JavaLogger which provides only the JavaLogger auditing feature. Your application can use this API if your code is compiled with JDK1.4 and above.

Using the Core Logging API

The following topics explain how to use each of the main CLA functions:

Initializing the Core Logging API

To initialize the CLA, use:

```
Call Logger::LoadCategories()
```

For Java:

```
call JavaLogger.LoadCategories()
```

Creating Debug Messages

The following information is required: MDC to which the message belongs, its category, the message, and its ID.

```
Call Logger::Debug(mdc, category, message, id)
```

For Java:

```
call JavaLogger.Debug(mdc, category, message, id)
```

Creating Information Messages

The following information is required: MDC to which the message belongs, its category, the message, and its ID.

```
Call Logger::Info(mdc, category, message, id)
```

For Java:

```
call JavaLogger.Info(mdc, category, message, id)
```

Creating Warning Messages

The following information is required: MDC to which the message belongs, its category, the message, and its ID.

```
Call Logger::Warn(mdc, category, message, id)
```

For Java:

```
call JavaLogger.Warn(mdc, category, message, id)
```

Creating Error Messages

The following information is required: MDC to which the message belongs, its category, the message, and its ID.

```
Call Logger::Error(mdc, category, message, id)
```

For Java:

```
call JavaLogger.Error(mdc, category, message, id)
```

Creating Fatal Messages

The following information is required: MDC to which the message belongs, its category, the message, and its ID.

```
Call Logger::Fatal(mdc, category, message, id)
```

For Java:

```
call JavaLogger.Fatal(mdc, category, message, id)
```

Creating Auditing Messages

Auditing messages can be created using Java by using:

```
call JavaLogger.audit(userName, flag, taskid, appid, message)
```

Auditing functionality is not available from C++.

Altering Priority for Category

You can alter the priority level of a category within an MDC using:

```
Call Logger::AlterPriority(mdc, category, newPriority)
```

For Java:

```
call JavaLogger.AlterPriority(mdc, category, newPriority)
```

Adding Logging Location to CCR

From a CCRInterface object, you can use:

```
call addLoggingCategory(mdc, category, priority)
```

From CCRAccess, you can use:

```
CCRAccess addLog mdc category priority
```

Adding Logging Category and Priority to CCR

From a CCRInterface object, you can use:

```
call setLoggingLocation(mdc, location)
```

From CCRAccess, you can use:

```
CCRAccess addLogLocation mdc location
```

About the Core Logging API Interface Design

The following topics describes the functions, fields, methods and arguments of the main Core Logging API components:

- About the Logger Interface
- About the JavaLogger Interface
- About the Auditlog Interface

About the Logger Interface

1. `public static void LoadCategories();`

   Initializes the CLA.

2. `public static void AlterPriority(std::string mdc, std::string category, std::string priority);`

   Alters a category's priority.

Parameter	Type	Purpose
`mdc`	`std::string`	The MDC to which the category belongs.
`category`	`std::string`	The category in the MDC whose priority will be altered.
`priority`	`std::string`	The new priority value.

3. `public static void Debug(std::string mdc, std::string category, std::string message, std::string id);`

   Posts a debug message if DEBUG priority is enabled.

Parameter	Type	Purpose
mdc	std::string	The MDC to which the message belongs.
category	std::string	The category in the MDC to which the message belongs.
message	std::string	The message.
id	std::string	The message ID.

4. `public static void Info(std::string mdc, std::string category, std::string message, std::string id);`

   Posts an info message if INFO priority or less is enabled.

Parameter	Type	Purpose
mdc	std::string	The MDC to which the message belongs.
category	std::string	The category in the MDC to which the message belongs.
message	std::string	The message.
id	std::string	The message ID.

5. `public static void Warn(std::string mdc, std::string category, std::string message, std::string id);`

   Posts a warning message if WARNING priority or less is enabled.

Parameter	Type	Purpose
mdc	std::string	The MDC to which the message belongs.
category	std::string	The category in the MDC to which the message belongs.
message	std::string	The message.
id	std::string	The message ID.

6. `public static void Error(std::string mdc, std::string category, std::string message, std::string id);`

Posts an error if ERROR priority or less is enabled.

Parameter	Type	Purpose
mdc	std::string	The MDC to which the message belongs.
category	std::string	The category in the MDC to which the message belongs.
message	std::string	The message.
id	std::string	The message ID.

7. `public static void Fatal(std::string mdc, std::string category, std::string message, std::string id);`

   Posts a fatal message if FATAL priority or less is enabled.

Parameter	Type	Purpose
mdc	std::string	The MDC to which the message belongs.
category	std::string	The category in the MDC to which the message belongs.
message	std::string	The message.
id	std::string	The message ID.

About the JavaLogger Interface

1. `public static synchronized void LoadCategories();`

   Initializes the CLA.

2. `public static synchronized void AlterPriority(string mdc, string category, string priority);`

   Alters a category's priority.

Parameter	Type	Purpose
mdc	string	The MDC to which the category belongs.
category	string	The category in the MDC whose priority will be altered.
priority	string	The new priority value.

3. `public static synchronized void Debug(string mdc, string category, string message, string id);`

   Posts a debug message if DEBUG priority is enabled.

Parameter	Type	Purpose
mdc	string	The MDC to which the message belongs.
category	string	The category in the MDC to which the message belongs.
message	string	The message.
id	string	The message ID.

4. `public static synchronized void Info(string mdc, string category, string message, string id);`

   Posts an info message if INFO priority or less is enabled.

Parameter	Type	Purpose
mdc	string	The MDC to which the message belongs.
category	string	The category in the MDC to which the message belongs.

Parameter	Type	Purpose
message	string	The message.
id	string	The message ID.

5. `public static synchronized void Warn(string mdc, string category, string message, string id);`

   Posts a warning message if WARNING priority or less is enabled.

Parameter	Type	Purpose
mdc	string	The MDC to which the message belongs.
category	string	The category in the MDC to which the message belongs.
message	string	The message.
id	string	The message ID.

6. `public static synchronized void Error(string mdc, string category, string message, string id);`

   Posts an error message if ERROR priority or less is enabled.

Parameter	Type	Purpose
mdc	string	The MDC to which the message belongs.
category	string	The category in the MDC to which the message belongs.
message	string	The message.
id	string	The message ID.

7. `public static synchronized void Fatal(string mdc, string category, string message, string id);`

   Posts a fatal message if FATAL priority or less is enabled.

Parameter	Type	Purpose
mdc	string	The MDC to which the message belongs.
category	string	The category in the MDC to which the message belongs.
message	string	The message.
Id	string	The message ID.

8. `public static synchronized void audit(string userName, byte flag, int taskid, string appid, string message)`

   Sends an auditing event to be logged.

Parameter	Type	Purpose
userName	string	The user name that posted the auditing message.
flag	byte	The flag of the event to be logged
taskid	int	The task ID.
appid	string	The application ID.
message	string	The auditing message.

9. `public static synchronized void audit(string userName, AccountParam params)`

   Sends an auditing event to be logged

Parameter Name	Type	Purpose
userName	string	The user name that posted the auditing message.
params	AccountParam	The parameters of the audit log.

About the Auditlog Interface

1. `public static synchronized void audit(string userName, byte flag, int taskid, string appid, string message)`

   Sends an auditing event to be logged. Auditlog.java is part of the com.cisco.core.nm.util package.

Parameter	Type	Purpose
userName	string	The user name that posted the auditing message.
flag	byte	The flag of the event to be logged
taskid	int	The task ID.
appid	string	The application ID.
message	string	The auditing message.