

## **Project54 Application Manager messaging**

W. Thomas Miller, III

This document describes special messages used by applications for communicating and synchronizing with the Project54 application manager (version 2.3 and higher).

For reference, all Project54 version 2 inter-application messages have a source application name, a destination application name, a message identifier string, and a message body string. A simple code model is:

```
Message(source, destination, id, message_text);
```

Since the application manager is not an application, it does not have an application name. Applications can send messages to the application manager by using the zero length string as the destination string. For example:

```
Message(L"lights", L"", L"any_id", L"STARTUP");
```

This code represents the "lights" application sending the "STARTUP" message to the application manager. *Note that this is not the same as:*

```
Message(L"lights", NULL, L"any_id", L"STARTUP");
```

This sends the message nowhere (i.e. it is discarded by the application manager).

### ***Startup Synchronization Using "STARTUP" and "BROADCAST STARTUP"***

In the Project54 software system all applications are launched almost simultaneously and immediately start processing on their own threads. Thus, there is no guarantee that when a client application first becomes ready at startup, other applications will also be ready to receive and handle messages.

The application manager supports a simple protocol to assist in application startup synchronization. When applications are initially launched, application manager support for inter-application message passing is disabled. When an application finishes initialization and becomes ready for normal operation, it should send a "STARTUP" message to the application manager. For example:

```
Message(self, L"", self, L"STARTUP");
```

As soon as the application manager has received the "STARTUP" message from all applications (or after a relatively long timeout period) the application manager enables inter-application message passing and sends the "BROADCAST STARTUP" message to

all applications. Applications should refrain from sending messages to other applications until after they receive the “BROADCAST STARTUP” message signaling that all applications are ready.

Messages are generally queued first by the application manager before delivery to the destination applications, and then by the individual destination applications. After startup, all applications should therefore always be able to receive messages without message loss. Messages are placed on the application manager’s queue by a thread in the source application, whereas messages are taken off of the application’s queue and processed by a thread in the destination application. Thus, sending a message takes very little time in the source application’s thread, even when processing the message may take significant time.

### ***Shutdown Synchronization Using “BROADCAST SHUTDOWN” and “SHUTDOWN”***

In the Project54 software system a shutdown synchronization procedure is used to ensure that all applications have an opportunity to shut down processing on their own threads.

Shutdown synchronization begins when the “BROADCAST SHUTDOWN” message is sent to all applications. Applications should then begin any application cleanup that may be necessary (closing files, setting final hardware device states, and so forth). Note that the “BROADCAST SHUTDOWN” message can be sent by any application, initiating the shutdown of the entire Project54 software system.

When an application finishes its cleanup operations, it should send a “SHUTDOWN” message to the application manager. For example:

```
Message(self, L””, self, L”SHUTDOWN”);
```

When the application manager receives the “SHUTDOWN” message from an application it marks the application as shutdown and discards any future messages destined for that application. Note that an application may still have messages in its local queue. As soon as the application manager has received the “SHUTDOWN” message from all applications (or after a relatively long timeout period) the application manager calls the exit() API function, immediately killing all remaining executing threads and shutting down the entire process.

### ***Broadcast Messages Using “BROADCAST ...”***

Broadcast messages are messages sent to all application. A message to be broadcast must be sent to the application manager, and must begin with the word “BROADCAST”. For example:

```
Message(self, L””, self, L”BROADCAST SHUTDOWN”);
```

This causes the “BROADCAST SHUTDOWN” message to be sent to all applications.

***Message Sniffers Using "REGISTER MESSAGE SNIFFER" and "UNREGISTER MESSAGE SNIFFER"***

Message sniffer applications receive a copy of all messages delivered by the application manager. An application registers as a sniffer by sending the "REGISTER MESSAGE SNIFFER" message to the application manager. For example:

```
Message(self, L"", self, L"REGISTER MESSAGE SNIFFER");
```

An application removes itself from the list of sniffers by sending the "UNREGISTER MESSAGE SNIFFER" message to the application manager. For example:

```
Message(self, L"", self, L"UNREGISTER MESSAGE SNIFFER");
```

Sniffer applications must check the destination address to distinguish messages intended for it versus messages intended for other applications. When a broadcast message is sent, a sniffer application sees only the copy addressed to it.

***Application Aliases Using "REGISTER APPLICATION ALIAS" and "UNREGISTER APPLICATION ALIAS"***

An application can register additional application names with the application manager. Subsequently, messages destined to the alias name will be delivered to the application which registered the alias. An application registers an alias name by sending the "REGISTER APPLICATION ALIAS" message to the application manager with the message identification field set to the alias name. For example:

```
Message(L"proxy", L"", L"pdalights", L"REGISTER APPLICATION ALIAS");
```

registers the name "pdalights" as an alias name for the "proxy" application. Any subsequent messages destined for "pdalights" will be delivered to "proxy". An alias is an extra name, not a replacement name. An application can register any number of aliases.

An application eliminates an alias name by sending the "UNREGISTER APPLICATION ALIAS" message to the application manager with the message identification field set to the alias name. For example:

```
Message(L"proxy", L"", L"pdalights", L"UNREGISTER APPLICATION ALIAS");
```

removes the name "pdalights" which was previously registered as an alias name for the "proxy" application. Note that an application can not "unregister" its real name, it can only "unregister" previously registered alias names.