US DIGITAL DESIGNS

MOTOROLA MCC7500 INTERFACE DESCRIPTION

INTRODUCTION

The USDD to Motorola MCC7500 Interface is the connection between a USDD Communications Gateway and one or more Motorola MCC7500 radio consoles. The interface consists of the Communications Gateway with an available Ethernet network interface, a Gateway Radio Interface (GARI) and the USDD to MCC7500 API Application.

The Communications Gateway is the central control server set for the USDD Phoenix G2 Station Alerting System. The GARI provides streaming transmit audio output from the Communications Gateway along with a push-to-talk output.

The USDD to MCC7500 API Application is a Windows application that must run on an MCC7500 radio console, where it uses the MCC7500 API DLLs and uses the MCC7500 features as a Dispatch User Interface (DUI) program. The Interface application exposes a subset of the console features that USDD has deemed necessary or desirable to support fire alerting and dispatching operations. The Interface also, whenever possible, hides the complexity of the MCC7500 API from the network client by exposing "feature-level" request/response messages on the network that the Interface turns into multiple API function calls and messages.

CONSOLE FEATURES

The MCC7500 features exposed by the Interface are:

- Call Alerts
- Resource Select / Multi Select
- Emergency Call Status (asynchronous notify)
- Inbound Radio Message (asynchronous notify)
- Inbound PTT ID Status (asynchronous notify)
- Resource Busy (asynchronous notify for only resources in a Multi-select Group)

The Interface exposes these features on the network via a TCP/IP socket connection on port 5071. A network client connects to the Interface and exchanges multiple messages over single connection for the duration of the client's lifetime. The format of the messages conforms to the USDD Block Protocol.

AUDIO INTERFACE

The Communications Gateway can transmit audio through the MCC7500 VPM via the USDD Gateway Radio Interface (GARI) module. This module connects to the VPM using the External Paging Encoder Port on the VPM and either Audio Output 1 or 2 on the GARI using a straight-through 8 pin modular patch cord. The GARI provides Transmit Audio, PTT, and Sidetone Audio to the VPM.

shows the connection from the GARI to the VPM and Figure 2 shows the VPM External Paging Encoder port.

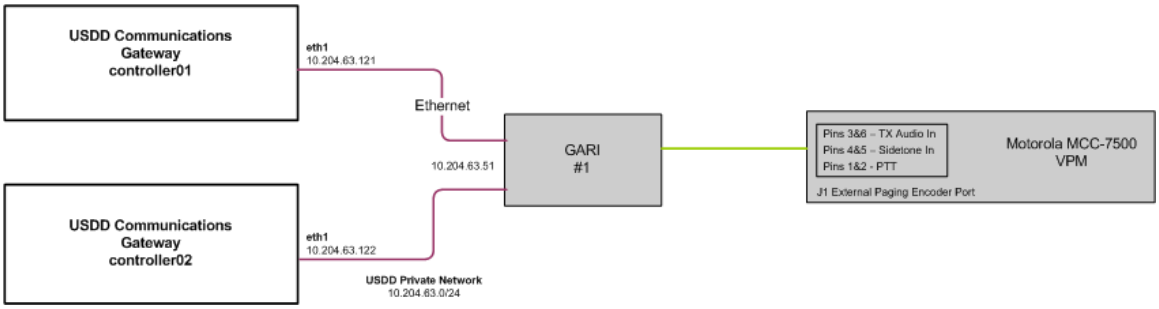

Figure 1

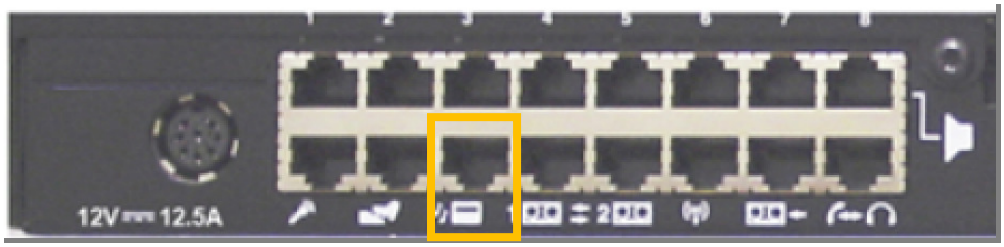

Figure 2

NETWORK INTERFACE

The Communications Gateway communicates with the MCC 7500 API Application through a TCP socket connection to the Application running on the MCC7500 console. To open a TCP connection to the application, the Communications Gateway must be able to reach the MCC7500 console through the trunking system's Customer Enterprise Network firewall which is the external entry point into the trunking system network. **Error! Reference source not found.** shows this connection from the Gateways to the CEN firewall. This diagram is only a logical example and specific implementations may have other

intervening network equipment between the Communications Gateways and the CEN firewall / Console Workstation. USDD will work with customer radio system technical staff to implement a suitable solution.

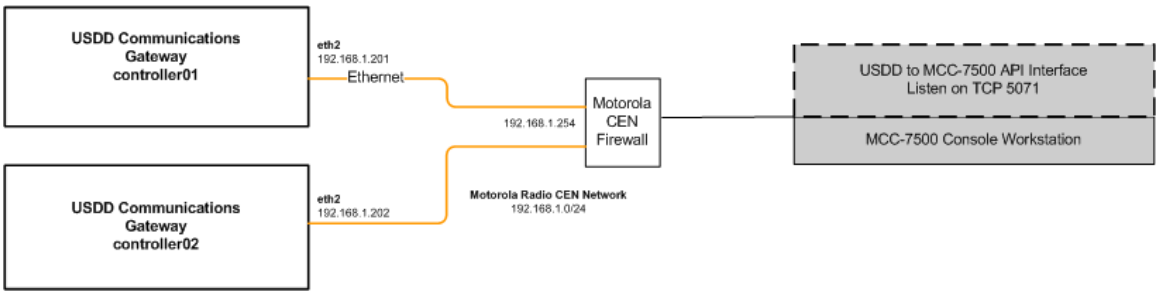

Figure 3

This firewall is managed as part of the trunking systems and customers will need to contact the trunking system administrator to approve and allow access. In installations that USDD has made in the past, a Juniper firewall is used by Motorola as the CEN firewall. An Ethernet cable connection is made from an available Ethernet interface on each Communications Gateway to an available port on the CEN firewall. The trunking system technicians then make rule additions to the firewall to allow the Communications Gateway shared IP address to access the IP address of the MCC7500 console on TCP port 5071. Previous installations have used the 192.168.1.0/24 network for the external CEN network and mapped the internal console IP addresses to addresses in this external network using NAT. The Communications Gateways have also been assigned addresses on this network.

USDD to MCC 7500 API INTERFACE APPLICATION

The USDD MCC7500 Interface program must run on a Windows-based MCC7500 radio console. This console is a computer provided by Motorola as part of the radio console system. The interface application software is installed in a specific application directory on the computer.

Installing or Updating the Application

1. Insert USB flash drive with the application software into an unused USB port on the MCC7500 console computer.

NOTE: If you are updating an existing installation proceed with Step 2. Otherwise skip to Step 6.

2. On the running USDD MCC7500 interface click the window **Close** button in the upper right of the application to close the application.

3. Delete old USDD shortcut from the Desktop.

4. Open Windows Explorer.

5. Open directory `c:\Program Files (x86)\Motorola MCC 7500\bin`. Delete the file `USDDMCC7500NetworkAdapterMFC.exe`. This is the old application.

6. Check for existence of the directory `C:\Program Files (x86)\US Digital Designs\` and create this directory if it does not exist.

7. Right click on the `C:\Program Files (x86)\US Digital Designs` directory, and select **Properties**.
    a. Select the **Security** tab.
    b. Click the **Edit...** button to change permissions (accept any UAC confirmation).
    c. Select the **Users (XXXXXXXXXXXX\Users)** group.
    d. Select the **Full control/Allow** checkbox.
    e. Click **OK** (closing the **Permissions for US Digital Designs** dialog).
    f. Click **OK** (closing the **US Digital Designs Properties** dialog).

8. Copy the new `USDDMCC7500NetworkAdapterMFC.exe` file from the flash drive to the `C:\Program Files (x86)\US Digital Designs` directory.

9. Copy `LaunchUSDDMCC7500NetworkAdapter.bat` from the flash drive to the `US Digital Designs` directory.

10. Right-click and drag `LaunchUSDDMCC7500NetworkAdapterMFC.exe` to Desktop and select **Create shortcut here**.

11. Right-click on this shortcut and select **Properties**.
    a. Click **Change Icon...**
    b. Click **Browse...**
    c. Navigate `to the c:\Program Files (x86)\US Digital Designs` directory and select `USDDMCC7500NetworkAdapterMFC.exe`.
    d. Click **Open**
    e. Select the appropriate app icon showing the USDD Phoenix G2 logo.
    f. Click **OK** (closing **Change Icon** dialog).
    g. Click the **General** tab.
    h. Change the shortcut name (in the first text box) to `USDD`.
    i. Click **OK** (closing the **Shortcut Properties** dialog).

12. Run (double-click) `LaunchUSDDMCC7500NetworkAdapter.bat`.

13. Copy `USDDstartup.bat` from the flash drive to `C:\Users\dispatch\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\`

14. Eject the USB flash drive and remove it from the computer.

Description of Batch Files

The purpose of the batch file in Step #9 is to start the interface application with its working directory set to the `c:\Program Files (x86)\Motorola MCC 7500\bin` directory (giving it access to the MCC7500 API DLLs). This allows us to put a simple shortcut (to the batch file) on the Desktop that will launch the interface application.

This batch file contains:

```
start /D "c:\Program Files (x86)\Motorola MCC 7500\bin"
USDDMCC7500NetworkAdapterMFC.exe
```

The purpose of the batch file in Step #13 is to make Windows run the file at startup (auto launch), and its contents (with the ping command) mimic the contents of a similar Motorola-provided .bat file seen on another customer's consoles and used to start the Motorola console application. The ping command serves only to delay the start of the actual program. This delay on startup is apparently required by the trunking system, and starting the adapter (or the console software itself) too early causes "issues". The use of ping as a simple delay is a known work-around since Windows doesn't have a built-in way to "sleep" in a batch file.

This batch file contains:

```
echo off
ping 127.0.0.1 -n 31 > NUL
start /D "C:\Program Files (x86)\US Digital Designs\"
LaunchUSDDMCC7500NetworkAdapter.bat
```

In short, these two batch files play nicely together, play nicely with the relevant permissions, and launch the interface application with access to the DLLs at a sufficiently late time for the underlying console services, *and* this mechanism mirrors what we found with Motorola's own software on the consoles at a customer site.

Adding Required Registry Keys

The interface stores two types of authentication credentials in the Windows registry. All values are in the `HKLM\Software\US Digital Designs\MCC7500 Interface` key.

NOTE: These keys may have to be installed in the `\Wow6432Node\US Digital Designs\MCC7500 Interface` key if the Motorola applications are installed in the `Program Files (x86)` directory.

Console user credentials - the interface must authenticate with the radio system before it can begin to use any of the API features. To do so, it needs the username and password of the console user (typically, the same username and password used to log in to the MCC7500 console software). These are stored in the `radio_username` and `radio_password` values.

Interface credentials - the interface application requires credentials to authorize incoming network connections from clients. The username and password pairs are stored in values `username1` and `password1`. Multiple credential pairs can be configured by specifying additional values of the basic form `usernameN` and `passwordN` where N is replaced with an incrementing number on each pair. These numbers must be contiguous -- the interface looks for pairs starting with 1 and then counts upwards, stopping when it doesn't find a matching pair.

Adding MSEL Group Registry Keys

The MC7500 console in its default configuration has only three Multi Select (msel) groups. The USDD MCC7500 Interface uses one of them for the Do Multi Select function. It asks the API for a "free" msel group, where "free" is defined as unopen (unselected) and having no members.

It is possible for all three of the Multi Select groups to have members (assigned via the MCC7500 console application, for example). In this case, the interface will be unable to find a "free" msel group, and it will be unable to service the Do Multi Select Request. Because of this, the interface will attempt to free all msel groups when it starts up (i.e., remove all members and close any open group).

NOTE: These keys may have to be installed in the `\Wow6432Node\US Digital Designs\MCC7500 Interface` key if the Motorola applications are installed in the `Program Files (x86)` directory.

This behavior can be inhibited for specific msel groups by creating one or more values in the Windows Registry. In the key:

```
HKLM\Software\US Digital Designs\MCC7500 Interface\
```

Create any of the three values:

```
protect_msel_group1
protect_msel_group2
protect_msel_group3
```

The presence of one of these registry values will protect the corresponding msel group -- the interface will not try and clear it on startup. The registry values can be any type, and have any value (it's the value name that is important).

Logging

The interface application writes its log files to the `C:\temp\` directory. This directory must exist before the application will write log files.

INTERFACE CONNECTION ESTABLISHMENT AND AUTHENTICATION

The USDD MCC7500 Interface accepts TCP/IP connections on port 5071. After connection, all communication with the interface uses the USDD Block Protocol.

After connecting to port 5071, a client MUST send a Login Request block. If the first block received by the interface is not a Login Request block, the interface closes the connection without sending any response.

When the client sends a Login Request block, the interface will respond with a Login Response block. This response will indicate success or failure of the login attempt (i.e., did the credentials match any of those configured in the Windows Registry). If the login attempt failed, the interface will close the connection immediately after sending the Login Response block.

If the interface replies with a Login Response block indicating success, then the connection and interface are now ready to accept other requests (Call Alert, Msel, etc.), and will send asynchronous notification messages to the client (Radio Message, PTT ID, etc.).

The interface ignores any Login Request blocks received on an already-authorized connection.

MCC7500 FEATURES AND THEIR USE

The USDD MCC7500 Interface exposes the following console features over the network:

- Emergency Alarm Status
- Emergency Call Status
- Inbound Radio Message
- PTT ID
- Call Alerts
- Resource Select / Unselect
- Multi Select / Unselect

EMERGENCY CALL STATUS

Upon receipt of an MCA_EMERGENCY_CALL_STATUS message from the API, the interface sends an Emergency Call Status block to each connected client. The Emergency Call Status block will always contain the following sub-blocks:

- Channel Data
- Status

The Emergency Call Status block MAY contain the following sub-blocks, if and only if the values received from the API are non-null:

- Unit ID
- Individual Alias
- Rcv Mode
- Rcv Secure Key
- Zone ID
- Site ID
- Resource
- Initiator

In all cases, the sub-blocks contain the values received from the MCC7500 API.

EMERGENCY ALARM STATUS

Upon receipt of an MCA_EMERGENCY_ALARM_STATUS message from the API, the interface sends an Emergency Alarm Status block to each connected client. The emergency Alarm Status block will always contain the following sub-blocks:

- Channel Data
- Status

The Emergency Alarm Status block MAY contain the following sub-blocks, if and only if the values received from the API are non-null:

- Unit ID
- Individual Alias
- Zone ID
- Site ID
- Resource

In all cases, the sub-blocks contain the values received from the MCC7500 API.

INBOUND RADIO MESSAGE

Upon receipt of an MCA_INB_RADIO_MESSAGE_STATUS message from the API, the interface sends an Inbound Radio Message Status block to each connected client. The Inbound Radio Message Status block will always contain the following sub-blocks:

- Channel Data

- Status
- Radio Message
- Primary Set ID
- Secondary Set ID

The Inbound Radio Message Status block MAY contain the following sub-blocks, if and only if the values received from the API are non-null:

- Unit ID
- Zone ID
- Site ID
- Dest Unit
- Individual Alias
- Destination Alias

In all cases, the sub-blocks contain the values received from the MCC7500 API.

PTT ID

Upon receipt of an MCA_INB_INBOUND_PTT_ID_STATUS message from the API, the interface sends an Inbound PTT ID Status block to each connected client. The Inbound PTT ID Status block will always contain the following sub-blocks:

- Channel Data
- Status

The Inbound PTT ID Status block MAY contain the following sub-blocks, if and only if the values received from the API are no-null:

- Unit ID
- Rcv Mode
- Rcv Secure Key
- individual Alias
- Zone ID
- Site ID

In all cases, the sub-blocks contain the values received from the MCC7500 API.

CALL ALERTS

The client can request the interface to perform Call Alerts on a specified resource and to a set of specified units. The client makes this request by sending a Do Call Alerts Request block to the interface.

The interface will respond with one or more Do Call Alerts Response blocks. The API processing of the Call Alert is a multi-step procedure, with opportunities along the way for incremental status changes to the units and the request itself. The client SHOULD check the Status block in each response to determine if the overall processing has completed yet -- status codes less than 1 (i.e., 0 or negative) indicate the request is complete, and this shall be the final response.

The Do Call Alerts Request block contains the following sub-blocks:

- Request ID - an opaque ID or token of the client's choosing.

- User Data - an opaque block containing data from the client that the interface will echo back to the client in each response. (optional)

- Resource - specifies the resource on which to send the call alerts.

- Seconds - specifies a timeout value in seconds for the interface's handling of this request. This timeout period begins when the interface begins processing this request. In particular, if this request is queued behind other Call Alert requests, this timeout period does not include the time this request spends waiting in the queue. (optional)

- One or more Unit ID blocks - each of which specify a unit to call alert.

The interface will respond with one or more Do Call Alerts Response blocks. These blocks WILL contain the following sub-blocks:

- Request ID - an echo of the ID received in the request.

- Status - a value indicating the overall outcome of the request (see list below).

- One or more Tuple blocks - one per unit ID requested

Additionally, the response block MAY contain the following:

- User Data - if the original request included a User Data block, the interface will return a copy of that data to the client in each response.

Each Tuple block in the response WILL contain the following sub-blocks:

- Unit ID - an echo of the requested unit ID

- Status - a value indicating the outcome of this specific unit (see list below).

The overall request status can have the following values:

0 - COMPLETE_ALL_MEMBERS_SUCCEEDED
This status indicates a successful outcome for all requested units.

1 - PENDING
Internal - not expected to reach the client.

2 - ADDING_MEMBERS
Internal - not expected to reach the client.

3 - PAGING_REQUESTED
Internal - not expected to reach the client.

4 - PAGING_IN_PROGRESS
Internal - not expected to reach the client.

-1 - COMPLETE_SOME_MEMBERS_FAILED
This status indicates the interface was able to execute a call alert via the API, but one or more of the units might have failed. In this case, the client should examine the status of each unit ID to discover which succeeded and which failed.

-2 - FAILED_CHANNEL_URID_NOT_ASSIGNED
This status indicates that the interface could not perform a call alert on the requested resource, because the requested resource URID was not assigned on the console (i.e., not present on the console UI).

-3 - FAILED_NO_UNITS_SPECIFIED
This status indicates that the original request did not contain any units, so there was no work for the interface to do.

-4 - FAILED_NO_UNITS_ADDED
This status indicates that the interface was unable to add any of the requested units to the page queue.

-5 - FAILED_PAGING_REQUEST_FAILED
This status indicates that the API paging request (McaBeginPaging) failed (i.e., it was not accepted by the API).

-6 - FAILED_REQUEST_TIMEOUT

This status indicates that the the client-specified timeout period expired before the API processing of this request could complete. Some units may have successfully received call

alerts. The client should check the status of each unit.

Lance Strong of Phoenix Fire reports that unreachable units cause the API to wait about 10 seconds before responding with a failure, while the other reachable units successfully alert much faster. USDD added this timeout functionality to handle this scenario better. This feature embodies the notion that "if a call alert to a unit will succeed, it will succeed quickly".

The individual unit status can have the following values:

0 - PAGE_SUCCESSFUL
This status indicates that the API reported a successful page for this unit.

1 - ADD_PENDING
This status indicates the interface has queued the unit to add it to the page queue, but hasn't yet presented it to the API.

2 - ADD_IN_PROGRESS
This status indicates the interface has requested the API to add this unit to the page queue, and the API accepted the request.

3 - ADD_SUCCESSFUL
This status indicates the API has successfully added the unit to the page queue.

-1 - ADD_FAILED_IMMEDIATELY
This status indicates the interface has requested the API to add this unit to the page queue, but the API did not accept the request.

-2 - ADD_FAILED
This status indicates the API could not add the unit to the page queue.

-3 - PAGE_FAILED
This status indicates that the call alert for this unit was unsuccessful.

MULTI SELECT

The client can request the interface to select a group of resources. The client makes this request by sending a Do Multi Select Request block to the interface. This block contains the following sub-blocks:

- Request ID - an opaque ID or token of the client's choosing.

- One or more Resource blocks - each of which specify a resource to select.

The interface will respond with a Do Multi Select Response block. This block WILL contain the following sub-blocks:

- Request ID - an echo of the ID received in the request.

- Status - a value indicating the overall outcome of the request (see list below).

- One or more Tuple blocks - one per resource requested

Each Tuple block WILL contain the following sub-blocks:

- Resource - an echo of the requested resource (URID)

- Status - a value indicating the outcome of this specific resource (see list below).

- Busy - an indication whether this resource is busy or idle

- Seconds - a value indicating how many seconds this resource has been in its current busy/idle state.

The overall request status can have the following values:

0 - ALL_SUCCESS
This status indicates that all resources have been selected.

1 - PARTIAL_SUCCESS
This status indicates that the selection process ran to completion, but not all of the resources have been selected successfully.

2 - PENDING
This status indicates that the request was received by the interface.

3 - GROUP_SELECTED
This status indicates that the API has reported the Msel Group is open (i.e., selected).

-1 - NOTHING_TO_DO
This status indicates that none of the requested resources were mapped in the console UI (or there were no resources specified in the request).

-2 - PRIOR_SELECTION_IN_PROGRESS
This status indicates that the interface already has an active selection.

-3 - COULDNT_GET_MSEL_GROUP
This status indicates that the API did not accept the request for a free msel group (McaGetFreeMselGroup).

-4 - GET_FREE_MSEL_GROUP_FAILED
This status indicates that the API request for a free msel group failed.

-5 - GET_FREE_MSEL_GROUP_TIMED_OUT
This status indicates that the interface gave up waiting on the API to respond to the request for a free msel group.

-6 - ADD_RESOURCE_TIMED_OUT
This status indicates that the interface gave up waiting on the API to respond to the request to add a resource to the msel group.

-7 - NO_RESOURCES_ADDED
This status indicates that none of the requested resources were added to the msel group.

-8 - MSEL_OPEN_REQUEST_FAILED
This status indicates that the API did not accept the request to open the msel group.

-9 - MSEL_OPEN_ERROR
This status indicates that the API request to open the msel group failed.

-10 - OPEN_MSEL_GROUP_TIMED_OUT
This status indicates that the interface gave up waiting on the API to respond to the request to open the msel group.

The individual resource status can have the following values:

1 - ADD_PENDING
This status indicates that the interface has queued the resource to add it to the msel group, but hasn't yet presented it to the API.

2 - ADD_REQUESTED
This status indicates that the API has accepted the request to

add the resource to the msel group.

3 - ADDED
This status indicates that the API has successfully added the resource to the msel group.

-1 - URID_NOT_MAPPED
This status indicates that the requested resource (URID) is not assigned on the console (i.e., not present on the console UI).

-2 - ADD_REQUEST_FAILED
This status indicates that the API did not accept the request to add the resource to the msel group.

-3 - ADD_ERROR
This status indicates that the API request to add the resource to the msel group has failed.

-4 - ADD_TIMED_OUT
This status indicates that the interface gave up waiting on the API to respond to the request to add this resource to the msel group.

After a Multi Select Request has succeeded, the interface will send notifications of idle/busy changes on any of the selected resources. It does this by sending a Resource Busy block to the client. This block contains the following sub-blocks:

- Resource - the resource whose idle/busy status has changed.

- Busy - the new idle/busy state of the resource.

After the client unselects the resources, the interface stops sending idle/busy updates.

UNSELECT

The client can request the interface to unselect the resources that have been previously selected by a Multi Select Request. The client makes this request by sending a Do Unselect Request block to the interface. This block contains the following sub-blocks:

- Request ID - an opaque ID or token of the client's choosing.

The interface will respond with a Do Unselect Response block block. This block WILL contain the following sub-blocks:

- Request ID - an echo of the ID received in the request.

- Status - a value indicating the outcome of the request (see list below).

The request status can have the following values:

0 - SUCCESS
This status indicates the interface has closed (unselected) the msel group, and removed all of the resources from it.

1 - PENDING
This status indicates the interface has received the request, but hasn't taken any action on it yet. (this status is not expected to reach the client).

-1 - NO_ACTIVE_SELECTION
This status indicates the interface did not have an active Multi Select.