DIMETRA APPLICATION PROGRAMMING INTERFACE (API) TRAINING

Dimetra Control Interface

MCC 7500

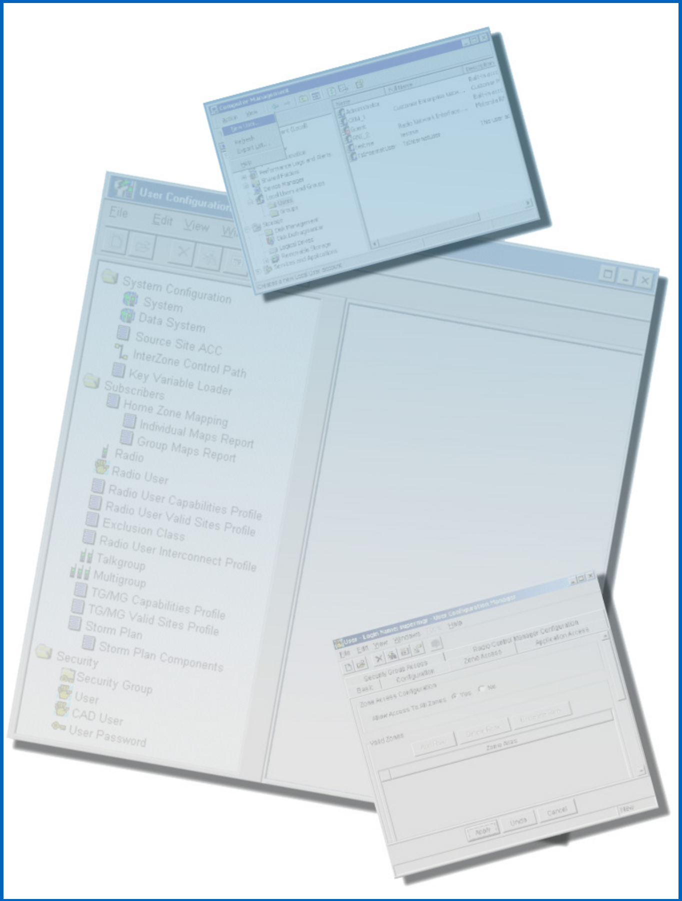
Course Structure

Module 1 - Course Introduction

Module 2 – MCC 7500 Overview

Module 3 – MCC 7500 API Structures

Module 4 – Summary



MCC 7500 Overview

Overview

MCC7500 API used to interface with the new dispatch subsystem that will eventually replace the older Elite consoles

MCC7500 API Capabilities

- Monitor voice communication on several talkgroups
- Transmit voice on single or multiple talkgroups
- Set up patch calls
- Receive status messages
- Handle emergency calls

Changes from Centracom Gold Series to MCC 7500

Added authentication

- User access permission to system

Aliasing

- Maintain by NM, no more ADM & On Line Alias API, extension in length (16chars), Unicode compliant

Unit ID presentation

- No more BCD conversion

Changes (Cont)

Changes in process control

- Due to authentication

New mechanisms

- Notified of db updates (DUN API)

Changes in audio processing

- IP network to control real-time call & audio

Changes in DUI Types

- Master and slaves (authentication)

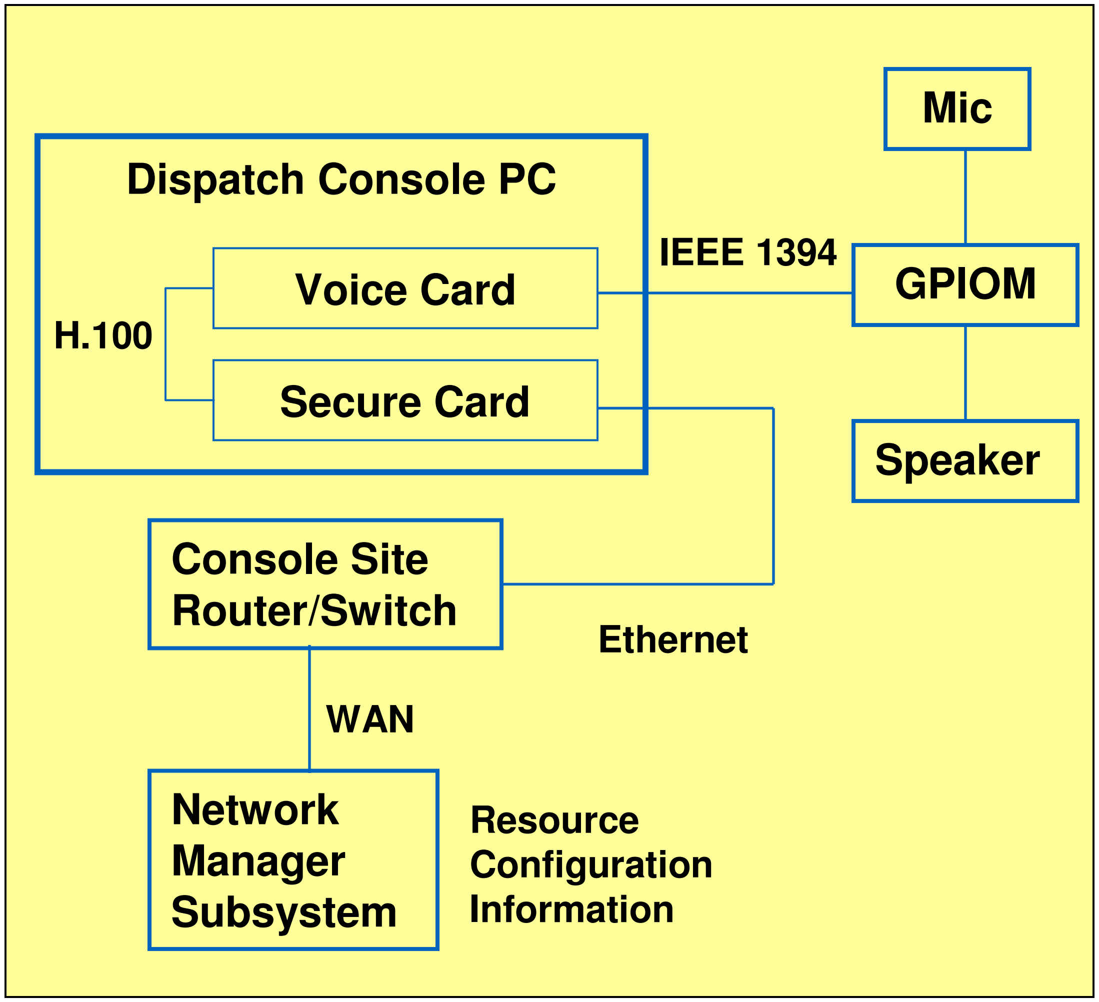
MCC 7500 Setup

Dispatch Console PC

Console Site Router/Switch

Network Manager Subsystem

General Purpose I/O Module (GPIOM)


Mic
Dispatch Console PC
IEEE 1394
GPIOM
Voice Card
H.100
Secure Card
Speaker
Console Site Router/Switch
Ethernet
WAN
Network Manager Subsystem
Resource Configuration Information

MCC7500 Software Development Kit (SDK)

The API is provided as C library (.lib) and header files (.h)

- Available as an SDK download

Documentation for API is available with SDK

Dynamic Link Libraries (DLLs) required to execute program

- Install via MCC7500 custom setting/support files

MsgSim is used to monitor and simulate events

- Similar to CEBsim for Elite

NMSim is to simulate the Network Manager Subsystem for resource configuration data

MCC7500 API IDs & Aliases

For Dimetra, Unit ID (ISSI) given in Binary Coded Decimal (BCD) format

- Eg. 123410 represented as 123416

Trunking resources (eg talkgroups, etc) are identified by Unified Resource IDs (URID)

All aliases (talkgroups, unit alias, etc) in MCC7500 support international characters

Therefore, alias strings are encoded in UTF-8 format

Organization of MCC7500 API

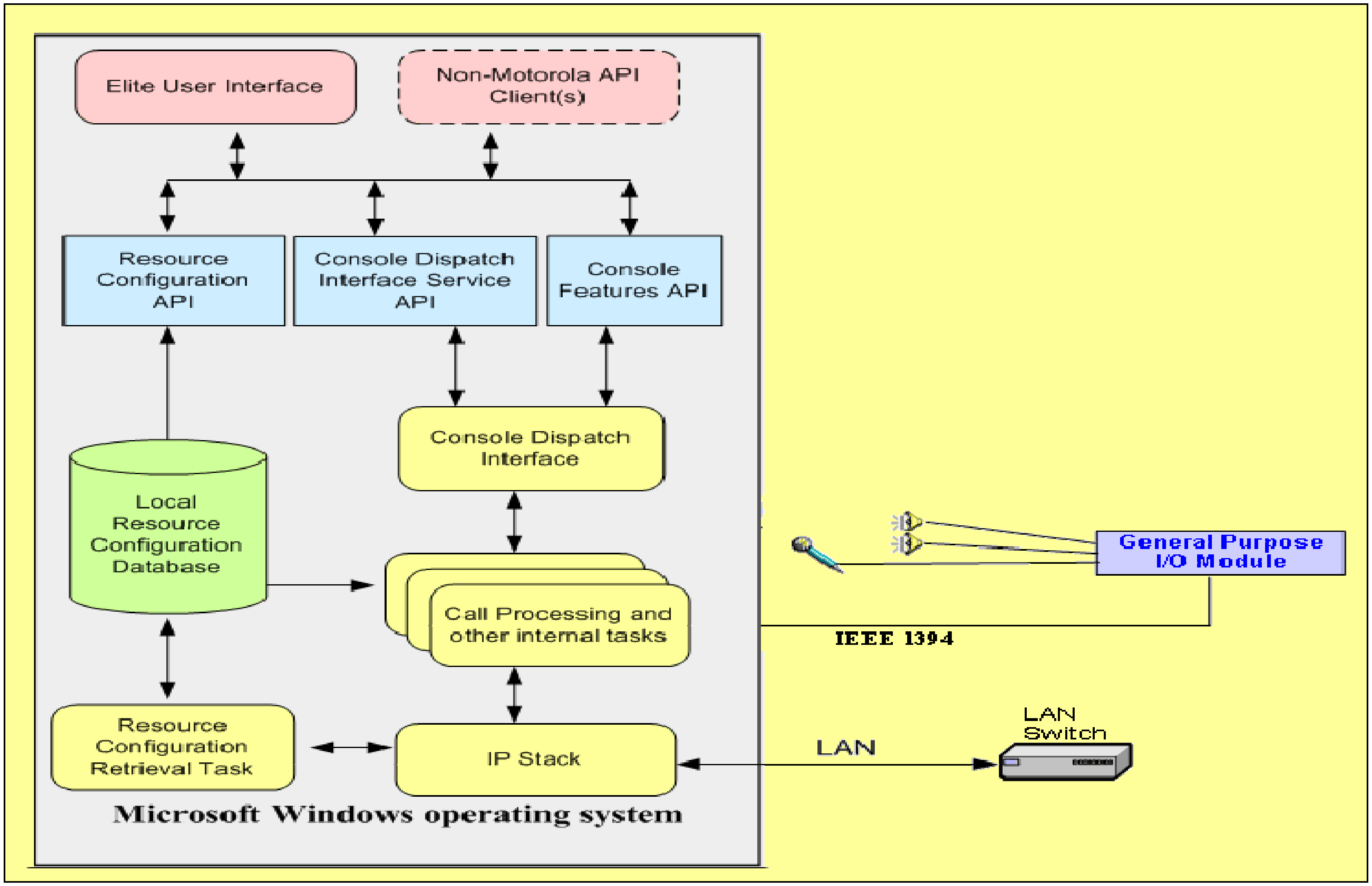
Split into several API groups

- Resource Configuration (RC) API
- Console Dispatch Interface (CDI) API
- Console Features API

Each API group supports several Features

Each Feature consists of several Functions and Responses

MCC7500 Software Architecture


Elite User Interface
Non-Motorola API Client(s)
Resource Configuration API
Console Dispatch Interface Service API
Console Features API
Console Dispatch Interface
Local Resource Configuration Database
Call Processing and other internal tasks
Resource Configuration Retrieval Task
IP Stack
Microsoft Windows operating system
General Purpose I/O Module
IEEE 1394
LAN
LAN Switch

Functions and Responses

Functions are C function calls

Responses are messages sent to the application via the CDI API

Note: All API groups have functions but not all have responses

Understanding Response Messages

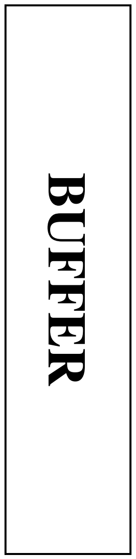
Response messages from the CDI in the form of a buffer

First WORD in buffer identifies response type

Must overlay corresponding ‘struct’ as defined in header file to interpret fields

- e.g. use C++ reinterpret_cast

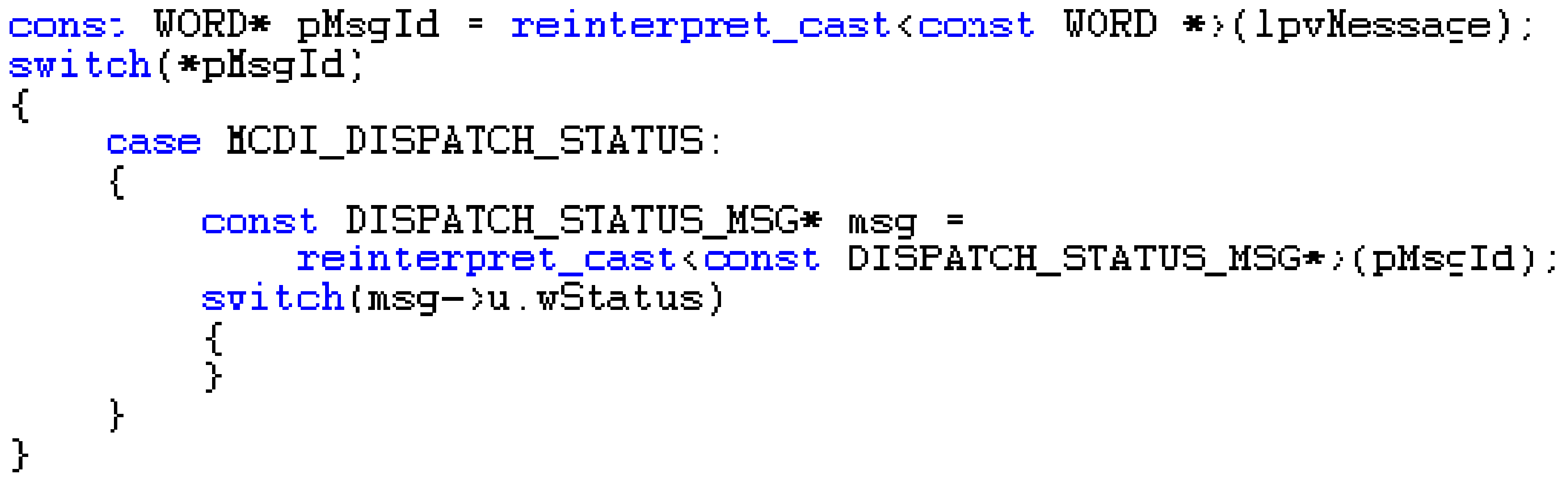
Understanding Response Messages (cont)



```
typedef struct tagDispatchStatus
{
      WORD wMsgId;
      union
      {
            WORD wStatus;
      } u;

} DISPATCH_STATUS_MSG;
```

Understanding Response Messages (cont)



```
const WORD* pMsgId = reinterpret_cast<const WORD *>(lpvMessage);
switch(*pMsgId)
{
    case MCDI_DISPATCH_STATUS:
    {
        const DISPATCH_STATUS_MSG* msg =
            reinterpret_cast<const DISPATCH_STATUS_MSG*>(pMsgId);
        switch(msg->u.wStatus)
        {
        }
    }
}
```

Handling Response Messages

Putting all the task (response messages) in a single switch-case statement would be impractical

The message ID are grouped based on their functions

The upper significant 10-bits indicates the group

Thus, response messages received can be routed to respective functions when the message IDs are masked using MCA_MSG_MASK which is 0xFFC0

Eg: MCA_GENERAL_XMIT_MSG group has messages such as MCA_GENERAL_XMIT_STATUS, MCA_END_GENERAL_XMIT_ERR, etc.

Console Dispatch Interface API

CDI API Overview

CDI Services API provides a communication framework such that a DUI Client can communicate with the MCC 7500 system

Communication consists of API requests by the DUI to the MCC 7500 system, and API responses from the MCC 7500 system to the DUI

CDI is the communication Server and processes are the clients of CDI

Obtain Response message from API

- Mostly from Console Features API and CDI API

Function names prefixed with Mcdi (Motorola Console Dispatch Interface)

Header files in Mcdi subfolder

Library file mcdi.lib

CDI Components

Registration

CDI Status

Client Activation

Authentication

Message Retrieval

System Database Updating

Audio Processing Status

Client Registration

- To allow dispatch application logon into CDI
- To allow CDI to monitor dispatch application to ensure proper interaction between CDI clients
- Registration requires a unique ID, client ID
- CDI will periodically request registration status from all registered clients to ensure alive
- Purpose of the client ID is to provide single entry point to communications services of CDI
- Deregistration is needed before shutting down

Registration functions

Registration

- McdiRegisterClientEx
- Registration as DUI client type for dispatch consoles

Deregistration

- McdiDeregisterClient

Recommended to register upon application initialize and deregister upon application shutdown only

Activation

Notify dispatch application of establishment or disruption of communication with console platform

Upon receiving McdiActivateClient message, the application should complete its initialization and call McdiInService. After this, application can start using the real-time dispatch functions

Similarly, upon receiving McdiDeactivateClient, application should clean-up, call McdiOutOfService and stop using the dispatch functions

Activation (cont)

Activation messages:

- McdiActivateClient
- McdiDeactivateClient

Functions:

- McdiInService
  - Notify system that it is now providing services or is “In Service”
- McdiOutOfService
  - Notify system that services is terminated

Authentication

Authentication is performed after the client registration

It is to authenticate to the Network Manager to allow the console to perform dispatch operation and also download configuration data

Authentication by console operator’s username and password. It can be different from Windows Domain password

- McdiRequestAuthentication
- McdiAuthenticationStatus

CDI Status Feature

Provides information to dispatch application on whether communication via CDI can be made

3 errors which affect communications between CDI & dispatch application

- Notify dispatch application with unregistered status of the sending dispatch application, on message transmit failure, prior to registration
- Notify dispatch application with the unregistered status of the receiving client when message is sent to an unregistered client
- Request dispatch application to shut down for unrecoverable error in the console position

Message Retrieval Feature

Two ways to use CDI API

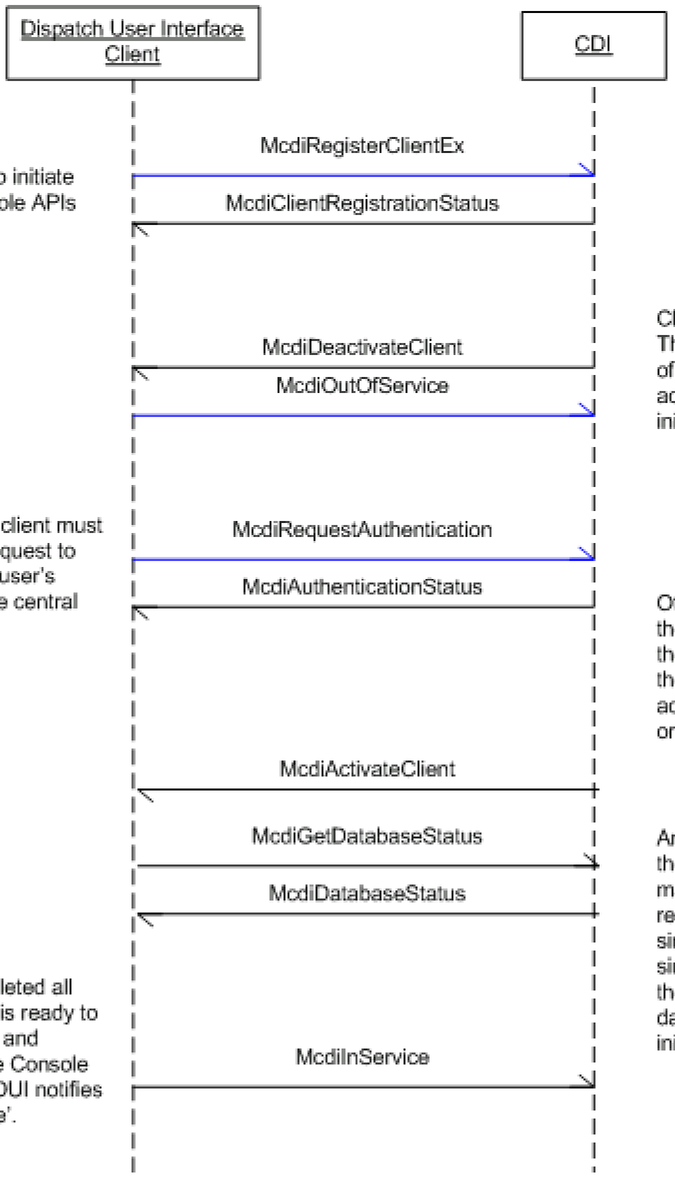
- Dedicated API call (Polling mechanism)
  - Must made this API call to retrieve message
  - Periodically check for arrival of new messages
  - MCDI_BLOCKED
    - API function wont return until a message is received
  - MCDI_NON_BLOCKED
    - API function return immediately
    - Returning a message or no message retrieved
- Callback function
  - CDI will invoke this function when it has message to deliver to dispatch application
  - Must provide at registration time

CDI API Initialization

In addition to the registration process similar to Elite API, MCC7500 requires 2 additional steps:

- Authentication
  - Need to provide username and password
  - To authenticate user instead of just authenticating the machine
- Activation
  - A method to activate/deactivate clients and other MCC7500 software process
  - To inform the system that a particular client is ready for its service/task and use of API function

CDI Initialization



Client registers with CDI to initiate communication with Console APIs

CDI puts the client in a known state. The DUI client must confirm it is out of service before it will receive an activation request to proceed with initialization.

After registration, the DUI client must make an authentication request to initiate the retrieval of the user's configuration data from the central system server.

Other API messages may be received by the DUI client between registration and the activation message. During this time, the DUI should not process or initiate any activity from the console features (MCA) or resource configuration (MCDB) APIs.

An McdiDatabaseStatus message indicating the readiness of the configuration database may have arrived at any time between registration and the activation message. To simplify the initialization process, DUIs can simply request the database status as part of their activation processing. Once the database is ready, the client may proceed with initialization.

When the client has completed all initialization activities and is ready to begin sending commands and processing status from the Console Dispatch API (MCA), the DUI notifies the CDI that it is 'in service'.

Resource Configuration API

RC API Overview

Provides means to retrieve the resources and capabilities from the Local Resource Configuration Database (DB) that the console operator has access to

Database updated by the Resource Configuration Retrieval Application at each operator position to keep the database synchronized with the master Network Manager database

Data availability depends on permissions setup by Network Manager

RC API Overview (cont)

Read only access to resource configuration data

May be used to obtain resource IDs and resource capabilities

Results returned via function parameters

- Does not use CDI responses

Function names prefixed with Mcdb (Motorola Console DataBase)

Header files in Mcdb subfolder

Library file mcdb32.lib

Database Access

Before RC data can be accessed, database connection must be opened

Open database

- McdbOpenDatabaseEx ()
- Open before any queries are made
- McdiRequestAuthentication () prior to open database

Close database

- McdbCloseDatabase ()
- Close after all queries are made

Recommendation

- Do not access the RC API when processing time-critical dispatch operations, so to avoid affecting communications system performances
- Cache all necessary data into memory before use

Queries

Simple Query

- Query result is a single value
- McdbGet<XXX>

List Query

- Query result is a list of values
- Start query
  - McdbCreate<XXX>Query
  - Eg. McdbCreateZoneQuery, McdbCreateTalkChannelQuery
- Continue Query
  - McdbGetNext<XXX>
  - Eg. McdbGetNextZone, McdbGetNextTalkChannel
- End query
  - McdbDestroyQuery

Database Update Notification Overview

A feature introduced to allows application to retrieve efficiently only the data from the database that has changed since last time the application retrieved that data

McdiDatabaseUpdateNotification will inform client which data groups in the database has changed

DUI application needs to keep track of the data group version number for the updates

Provides selective database update notification messages

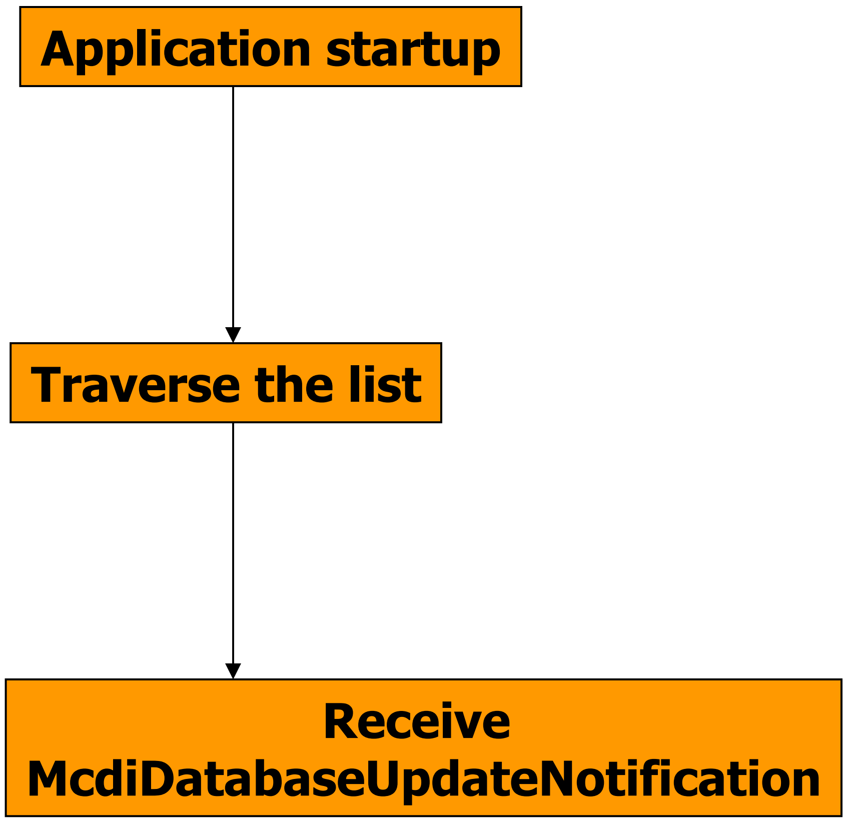
Using query functions that has ‘ByVersion’ suffix such as McdbCreateResCapabilityQueryByVersion

DUN Procedures (Pic)

“Create query” function,
value 0 for prev ver num

“Get next” function,
Store the largest ver num
for future usage

“Create query” function,
Pass the version number
in prev step,
Store the largest ver num
for the list of data



Console Features API

Console Features API Overview

Real time monitoring and control of communication resources

2 types of API messages

- API Functions
  - a request for something to be done by the system
  - will return a result
- API Responses
  - API Status Responses
    - indicate some condition has occurred in the system
    - May be of previous API function call, API called parallelly by a console position or system updates
  - API Error Responses
    - Indicate error condition that has resulted from trying to execute an API function
  - API Warning Responses
    - Inform console position of a condition in the system

Most functions give result through CDI response messages

Console Features API Overview (Cont)

Function names prefixed with Mca (Motorola Console Application)

Header files in Mca subfolder

Library file mca.lib

Resource Assignment

Resources need to be assigned before it can be monitored or controlled by the position

A Logical Channel ID (LC_ID) will be given for each assigned resource

All programmatic interaction with the resource will be via the LC_ID

Transmit

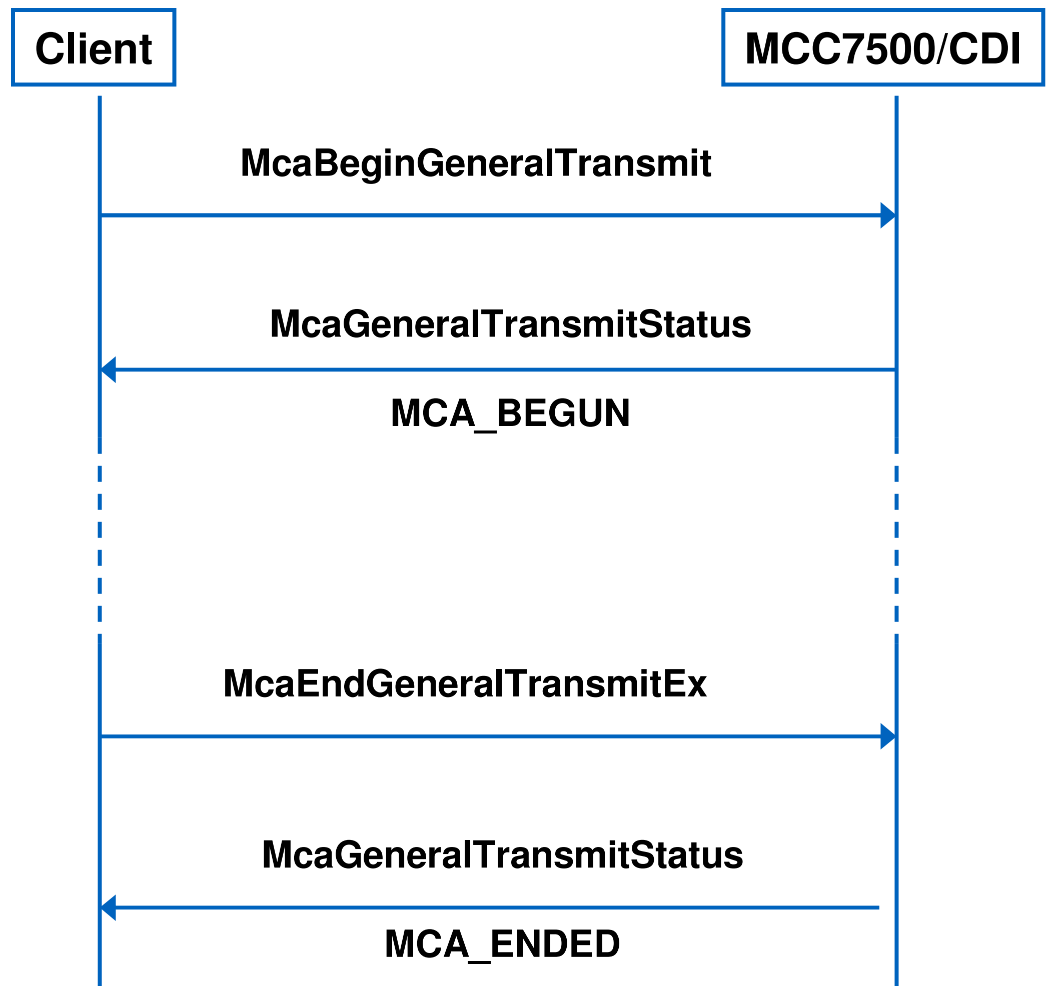
General Transmit

- Transmit on currently selected resource
- Same effect as pushing red button on CIE
- Functions:
  - McaBeginGeneralTransmit
  - McaEndGeneralTransmitEx
- Responses:
  - McaGeneralTransmitStatus: MCA_BEGUN, MCA_ENDED

Instant transmit

- Transmit on a particular resource
- Supply LC_ID as parameter

General Transmit Operation



Summary

SUMMARY SLIDE

Dimetra Dispatcher Overview

MCC 7500 SDK Structures

- Console Dispatch Interface API
- Resource Configuration API
- Console Feature API

THANK YOU…