Joint Tactical Radio System (JTRS) Standard
Audio Port Device
Application Program Interface (API)

Version: 1.3.4
29 July 2010

Statement A- Approved for public release; distribution is unlimited (29 July 2010)
## REVISION HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Authorization</th>
<th>Description</th>
<th>Last Modified Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td></td>
<td>Initial release</td>
<td>20-December-2005</td>
</tr>
<tr>
<td></td>
<td>ICWG Approved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td></td>
<td>Update outline format, add extensions</td>
<td>02-February-2006</td>
</tr>
<tr>
<td></td>
<td>ICWG Approved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td>Removed Vocoder Extension.</td>
<td>22-February-2006</td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td>Updates based on Increment 4 ICWG. Added audio sample stream support, removed deprecated operations.</td>
<td>09-May-2006</td>
</tr>
<tr>
<td></td>
<td>ICWG Approved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.1</td>
<td></td>
<td>Updated Figure to be consistent with the Vocoder Service API.</td>
<td>07-August-2006</td>
</tr>
<tr>
<td></td>
<td>ICWG Approved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.1.1</td>
<td></td>
<td>Preparation for public release</td>
<td>29-March-2007</td>
</tr>
<tr>
<td>1.3.2</td>
<td></td>
<td>Errata: In AudioSampleStreamExt.idl corrected #ifndef/#define, i.e. changed __AUDIO_SAMPLE_STREAM_DEFINED to __AUDIO_SAMPLE_STREAM_EXT_DEFINED</td>
<td>02-April-2008</td>
</tr>
<tr>
<td>1.3.3</td>
<td></td>
<td>Correction: Section A.5.4.2: Clarified complex tone profile description</td>
<td>29-June-2010</td>
</tr>
<tr>
<td></td>
<td>ICWG Approved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.4</td>
<td></td>
<td>Preparation for public release</td>
<td>29-July-2010</td>
</tr>
</tbody>
</table>
Table of Contents

A. AUDIO PORT DEVICE ..................................................................................................8

B. AUDIO SAMPLE STREAM EXTENSION ................................................................30
Table of Contents

A. AUDIO PORT DEVICE ........................................................................................................... 8
A.1 Introduction ......................................................................................................................... 8
  A.1.1 Overview ....................................................................................................................... 8
A.1.2 Service Layer Description .............................................................................................. 9
  A.1.2.1 Audio Port Device Port Connections ..................................................................... 9
A.1.3 Modes of Service ............................................................................................................. 9
A.1.4 Service States .................................................................................................................. 10
A.1.5 Referenced Documents ................................................................................................. 11
  A.1.5.1 Government Documents ...................................................................................... 11
  A.1.5.2 Commercial Standards ......................................................................................... 11
A.2 Services ............................................................................................................................... 12
  A.2.1 Provide Services ......................................................................................................... 12
  A.2.2 Use Services .............................................................................................................. 12
  A.2.3 Interface Modules ....................................................................................................... 13
  A.2.3.1 Audio Port Device ............................................................................................... 13
  A.2.4 Sequence Diagrams ................................................................................................... 14
A.3 Service Primitives and Attributes .................................................................................... 15
  A.3.1 Audio::AudibleAlertsAndAlarms ............................................................................. 16
    A.3.1.1 createTone Operation ...................................................................................... 16
    A.3.1.2 startTone Operation ....................................................................................... 17
    A.3.1.3 stopTone Operation ....................................................................................... 18
    A.3.1.4 stopAllTones Operation ............................................................................... 19
    A.3.1.5 destroyTone Operation .................................................................................. 20
A.3.2 Audio::AudioPTT_Signal ............................................................................................ 21
  A.3.2.1 setPTT Operation ............................................................................................... 21
A.4 IDL .................................................................................................................................. 22
  A.4.1 Audio IDL ................................................................................................................... 22
A.5 UML .................................................................................................................................. 24
  A.5.1 Data Types .................................................................................................................. 25
  A.5.2 Enumerations ............................................................................................................. 25
    A.5.2.1 Audio::AudibleAlertsAndAlarms::ToneDescriptor ........................................ 25
  A.5.3 Exceptions ................................................................................................................... 25
    A.5.3.1 Audio::AudibleAlertsAndAlarms::InvalidToneProfile .................................... 25
    A.5.3.2 Audio::AudibleAlertsAndAlarms::InvalidToneId .......................................... 26
  A.5.4 Structures .................................................................................................................... 26
    A.5.4.1 Audio::AudibleAlertsAndAlarms::SimpleToneProfile .................................... 26
    A.5.4.2 Audio::AudibleAlertsAndAlarms::ComplexToneProfile ................................. 26
  A.5.5 Unions ........................................................................................................................ 27
    A.5.5.1 Audio::AudibleAlertsAndAlarms::ToneProfileType ........................................ 27
Appendix A.A Abbreviations and Acronyms .......................................................................... 28
Appendix A.B Performance Specification .............................................................................. 29

B. AUDIO SAMPLE STREAM EXTENSION ............................................................................... 30
B.1 Introduction ......................................................................................................................... 30
Lists of Figures

FIGURE 1 – AUDIO PORT DEVICE PORT DIAGRAM ................................................................. 9
FIGURE 2 – AUDIO PORT DEVICE STATE DIAGRAM ............................................................. 10
FIGURE 3 – AUDIO PORT DEVICE INTERFACE CLASS DIAGRAM .................................... 13
FIGURE 4 – AUDIO ALERTS AND ALARMS CLASS DIAGRAM .......................................... 13
FIGURE 5 – AUDIO_PTT_SIGNAL CLASS DIAGRAM ........................................................... 13
FIGURE 6 – AUDIO PORT DEVICE COMPONENT DIAGRAM ............................................... 24
FIGURE 7 – AUDIO SAMPLE STREAM EXTENSION PORT DIAGRAM .................................. 31
FIGURE 8 – AUDIO SAMPLE STREAM EXTENSION STREAMING STATE DIAGRAM ............ 32
FIGURE 9 – AUDIO SAMPLE STREAM EXTENSION INTERFACE CLASS DIAGRAM .......... 36
FIGURE 10 – SAMPLE_STREAM INTERFACE DIAGRAM .................................................... 37
FIGURE 11 – SAMPLE_STREAM_CONTROL INTERFACE DIAGRAM ............................. 38
FIGURE 12 – SAMPLE_MESSAGE_CONTROL INTERFACE DIAGRAM ............................ 38
FIGURE 13 – AUDIO SAMPLE STREAM EXTENSION COMPONENT DIAGRAM .................. 41
List of Tables

TABLE 1 – AUDIO PORT DEVICE PROVIDE SERVICE INTERFACE ......................................................... 12
TABLE 2 – AUDIO PORT DEVICE USE SERVICE INTERFACE ............................................................ 12
TABLE 3 – AUDIO PORT DEVICE PERFORMANCE SPECIFICATION .................................................. 29
TABLE 4 – AUDIO SAMPLE STREAM EXTENSION PROVIDE SERVICE INTERFACE ......................... 34
TABLE 5 – AUDIO SAMPLE STREAM EXTENSION USE SERVICE INTERFACE .................................. 35
TABLE 6 – AUDIO SAMPLE STREAM EXTENSION PERFORMANCE SPECIFICATION ...................... 43
A. AUDIO PORT DEVICE

A.1 INTRODUCTION

The Audio Port Device supports methods and attributes that are specific to the Audio Port hardware (HW) device it represents. The Audio Port Device provides the ability to control alert and alarm tones and to notify the device user of a Push-To-Talk (PTT) signal.

The Audio Port Device also provides a base configuration interface. It should be noted that this base Audio Port Device may be extended with the use of the extension (see B.1).

This document defines a common set of Audio Port Device provide services and interfaces required by most Joint Tactical Radio (JTR) Sets.

The Audio Port Device acts as “device adapter”. It is used by Common Object Request Broker Architecture (CORBA) components (e.g., waveform application components) to access JTR Set Audio Port HW.

A.1.1 Overview

a. Section A.1, Introduction, contains the introductory material regarding the overview, service layer description, modes, states, and referenced documents of this document.

b. Section A.2, Services, specifies the interfaces for the component, port connections, and sequence diagrams.

c. Section A.3, Service Primitives and Attributes, specifies the operations that are provided by the Audio Port Device.

d. Section A.4, IDL.

e. Section A.5, UML.

f. Appendix A.A, Abbreviations and Acronyms.

g. Appendix A.B, Performance Specification.
A.1.2 Service Layer Description

A.1.2.1 Audio Port Device Port Connections

Figure 1 shows the port connections for the Audio Port Device.

Note: All port names are for reference only.

![Audio Port Device Port Diagram]

**Audio Port Device Provides Ports Definitions**

- **audio_alertalarm_wf_provides_port** is provided by the Audio Port Device to control the alert and alarm tones by waveform.

**Audio Port Device Uses Ports Definitions**

- **audio_ptt Uses_port** is used by the Audio Port Device to notify the Device User of PTT signal.

A.1.3 Modes of Service

Not applicable.
A.1.4 Service States

A.1.4.1.1 Audio Port Device State Diagram

The Audio Port Device states are illustrated in Figure 2. The Audio Port Device states ensure that received operations are only executed when the Audio Port Device is in the proper state. The five states of the Audio Port Device are as follow:

• CONSTRUCTED - The state transitioned to upon successful creation.
• INITIALIZED - The state transitioned to upon successful initialization.
• ENABLED - The state transitioned to upon successful start.
• DISABLED - The state transitioned to upon successful stop.
• RELEASED - The state transitioned to upon successful release.

The Audio Port Device transitions between states in response to the initialize, start, stop and releaseObject operations.

![Audio Port Device State Diagram](image)

Figure 2 – Audio Port Device State Diagram
A.1.5 Referenced Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein.

A.1.5.1 Government Documents

A.1.5.1.1 Specifications

A.1.5.1.1.1 Federal Specifications
None

A.1.5.1.1.2 Military Specifications
None

A.1.5.1.2 Other Government Agency Documents


A.1.5.2 Commercial Standards

None
### A.2 SERVICES

#### A.2.1 Provide Services

The Audio Port Device provides service consists of the Table 1 service ports, interfaces, and primitives, which can be called by other client components.

<table>
<thead>
<tr>
<th>Service Group (Port Name)</th>
<th>Service (Interface Provided)</th>
<th>Primitives (Provided)</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio_alertalarm_wf_in_port</td>
<td>Audio::AudibleAlertsAndAlarms</td>
<td>createTone()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>startTone()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stopTone()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stopAllTones()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>destroyTone()</td>
</tr>
</tbody>
</table>

#### A.2.2 Use Services

The Audio Port Device use service set consists of the Table 2 service ports, interfaces, and primitives. Since the Audio Port Device acts as a client with respect to these services from other components, it is required to connect these ports with corresponding service ports applied by the server component. The Audio Port Device uses the port name as the connectionID for the connection.

<table>
<thead>
<tr>
<th>Service Group (Port Name)</th>
<th>Service (Interface Used)</th>
<th>Primitives (Used)</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio_ptt_out_port</td>
<td>Audio::AudioPTT_Signal</td>
<td>setPTT()</td>
</tr>
</tbody>
</table>
A.2.3 Interface Modules

A.2.3.1 Audio Port Device

A.2.3.1.1 AudibleAlertsAndAlarms Interface Description
The AudibleAlertsAndAlarms interface provides tone and beep creation and their storage capability to the device user.

A.2.3.1.2 AudioPTT_Signal Interface Description
The AudioPTT_Signal interface is defined in the Audio Port Device for the use by the waveforms, so that the Audio Port Device can signal an event to the waveform when a Push-To-Talk (PTT) event is received from the Audio Port HW.
A.2.4 Sequence Diagrams

None
A.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in Section A.5. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.
A.3.1 Audio::AudibleAlertsAndAlarms

A.3.1.1 createTone Operation

The createTone operation provides the capability of creating a tone or beep with the specified profile, for future use by the device user.

A.3.1.1.1 Synopsis

unsigned short createTone(in ToneProfileType toneProfile) raises(InvalidToneProfile);

A.3.1.1.2 Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>toneProfile</td>
<td>A structure containing the elements used to create the tone/beep for the Audio Port Device.</td>
<td>ToneProfileType (see A.5.3)</td>
</tr>
</tbody>
</table>

A.3.1.1.3 State

ENABLED CF::Device::operationalState.

A.3.1.1.4 New State

This operation does not cause a state change.

A.3.1.1.5 Return Value

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned short</td>
<td>An identification number associated with the tone/beep.</td>
<td>0 – 64k</td>
</tr>
</tbody>
</table>

A.3.1.1.6 Originator

Service User

A.3.1.1.7 Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneProfile</td>
<td>A CORBA exception is raised when the tone/beep cannot be generated due to invalid attributes in toneProfileType structure.</td>
</tr>
</tbody>
</table>

(see A.5.3.1)
A.3.1.2 *startTone Operation*

The *startTone* operation provides the user the ability to start the generation of a previously created tone/beep to the device user.

A.3.1.2.1 **Synopsis**

```c
void startTone(in unsigned short toneId) raises(InvalidToneId);
```

A.3.1.2.2 **Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
<th>Units</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>toneId</td>
<td>A tone ID associated with the tone/beep for sending to the Audio Port HW.</td>
<td>unsigned short</td>
<td>ID number</td>
<td>1 – 65535</td>
</tr>
</tbody>
</table>

A.3.1.2.3 **State**

ENABLED CF::Device::operationalState.

A.3.1.2.4 **New State**

This operation does not cause a state change.

A.3.1.2.5 **Return Value**

None

A.3.1.2.6 **Originator**

Service User

A.3.1.2.7 **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneId</td>
<td>A CORBA exception is raised when the tone/beep identification number is invalid.</td>
</tr>
</tbody>
</table>

(see A.5.3.2)
A.3.1.3 *stopTone* Operation

The *stopTone* operation provides the device user the ability to stop generation of a previously started tone.

### A.3.1.3.1 Synopsis

```c
void stopTone(in unsigned short toneId) raises(InvalidToneId);
```

### A.3.1.3.2 Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
<th>Units</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>toneId</td>
<td>A tone ID associated with the tone which needs to be stopped.</td>
<td>unsigned short</td>
<td>ID number</td>
<td>1 – 65535</td>
</tr>
</tbody>
</table>

### A.3.1.3.3 State

ENABLED CF::Device::operationalState.

### A.3.1.3.4 New State

This operation does not cause a state change.

### A.3.1.3.5 Return Value

None

### A.3.1.3.6 Originator

Service User

### A.3.1.3.7 Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneId (see A.5.3.2)</td>
<td>A CORBA exception is raised when the tone/beep identification number is invalid.</td>
</tr>
</tbody>
</table>
A.3.1.4  *stopAllTones* Operation

The *stopAllTones* operation provides the device user the ability to stop generation of all previously started tones.

A.3.1.4.1  Synopsis

    void stopAllTones();

A.3.1.4.2  Parameters

None

A.3.1.4.3  State

ENABLED CF::Device::operationalState.

A.3.1.4.4  New State

This operation does not cause a state change.

A.3.1.4.5  Return Value

None

A.3.1.4.6  Originator

Service User

A.3.1.4.7  Exceptions

None
A.3.1.5  *destroyTone Operation*

The *destroyTone* operation provides the device user the ability to destroy the previously created tone/beep to prevent the future use.

A.3.1.5.1  **Synopsis**

```c
void destroyTone(in unsigned short tonelId) raises(InvalidToneId);
```

A.3.1.5.2  **Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
<th>Units</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>tonelId</td>
<td>A tone/beep ID associated with the tone/beep which needs to be destroyed.</td>
<td>unsigned short</td>
<td>ID number</td>
<td>1 – 65535</td>
</tr>
</tbody>
</table>

A.3.1.5.3  **State**

ENABLED CF::Device::operationalState.

A.3.1.5.4  **New State**

This operation does not cause a state change.

A.3.1.5.5  **Return Value**

None

A.3.1.5.6  **Originator**

Service User

A.3.1.5.7  **Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneId</td>
<td>A CORBA exception is raised when the tone/beep identification number is invalid.</td>
</tr>
</tbody>
</table>
A.3.2 Audio::AudioPTT_Signal

A.3.2.1 setPTT Operation

The setPTT operation is used to inform the downstream components of the push to talk signal.

A.3.2.1.1 Synopsis

void setPTT(in boolean PTT):

A.3.2.1.2 Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
<th>Valid Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTT</td>
<td>Indicates whether the push to talk signal has been received from the Audio Port HW.</td>
<td>boolean</td>
<td>TRUE = push to talk signal has been received from Audio Port HW; FALSE = push to talk signal has not been received from Audio Port HW.</td>
</tr>
</tbody>
</table>

A.3.2.1.3 State

ENABLED CF::Device::operationalState.

A.3.2.1.4 New State

This operation does not cause a state change.

A.3.2.1.5 Return Value

None

A.3.2.1.6 Originator

Service Provider

A.3.2.1.7 Exceptions

None
A.4 IDL

A.4.1 Audio IDL

/*
** Audio.idl - JTRS Base Audio Service Set
*/
#ifndef __AUDIO_DEFINED
#define __AUDIO_DEFINED
#endif
#ifndef __JTRSCORBATYPES_DEFINED
#include "JtrsCorbaTypes.idl"
#endif

module Audio
{
    // Push to Talk Control
    interface AudioPTT_Signal
    {
        void setPTT( in boolean PTT );
    };

    interface AudibleAlertsAndAlarms
    {
        exception InvalidToneProfile
        {
            boolean complexTone;  // changed to FALSE ComplexToneProfile structure invalid
            boolean simpleTone;    // changed to FALSE SimpleToneProfile structure invalid
            string msg;            // message exception location
        };

        exception InvalidToneId
        {
            string msg;            // message exception location
        };

        struct SimpleToneProfile
        {
            unsigned short frequencyInHz;  // frequency in Hz
        };
    }
}
unsigned short durationPerBurstInMs; // duration of tone per burst in milliseconds
unsigned short repeatIntervalInMs;  // The repeat interval in milliseconds
};

enum ToneDescriminator
{
    COMPLEX_TONE, // Select ComplexToneProfile
    SIMPLE_TONE  // Select SimpleToneProfile
};

struct ComplexToneProfile
{
    JTRS::ShortSequence toneSamples;  // tone samples
    unsigned short  numberOfRepeats; // number times to repeat samples
};

union ToneProfileType switch ( ToneDescriminator )
{
    case COMPLEX_TONE:
        ComplexToneProfile complexTone; // tone described by ComplexToneProfile
    case SIMPLE_TONE:
        SimpleToneProfile simpleTone;  // tone described by SimpleToneProfile
};

unsigned short createTone( in ToneProfileType toneProfile )
raises (InvalidToneProfile);

void startTone( in unsigned short toneId )
raises (InvalidToneId);

void stopTone( in unsigned short toneId )
raises (InvalidToneId);

void destroyTone( in unsigned short toneId )
raises (InvalidToneId);

void stopAllTones();

#endif
A.5 UML

This section contains the device component UML diagram and the definitions of all data types referenced (directly or indirectly) by A.3 Service Primitives and Attributes.

![UML Diagram]

Figure 6 – Audio Port Device Component Diagram
A.5.1 Data Types

None

A.5.2 Enumerations

A.5.2.1 Audio::AudibleAlertsAndAlarms::ToneDescriminator

The ToneDescriminator enumeration type defines attributes to determine if the tone/beep selection is the ComplexToneProfile or a SimpleToneProfile. This enumeration type is used by ToneProfileType to make a selection between the two profiles.

```cpp
enum ToneDescriminator
{
    COMPLEX_TONE,
    SIMPLE_TONE
};
```

<table>
<thead>
<tr>
<th>Enum</th>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToneDescriminator</td>
<td>COMPLEX_TONE</td>
<td>Select ComplexToneProfile.</td>
</tr>
<tr>
<td></td>
<td>SIMPLE_TONE</td>
<td>Select SimpleToneProfile.</td>
</tr>
</tbody>
</table>

A.5.3 Exceptions

A.5.3.1 Audio::AudibleAlertsAndAlarms::InvalidToneProfile

The InvalidToneProfileType exception is raised when any of the attributes in either ComplexToneProfile structure or SimpleToneProfile structure are out of range.

```cpp
exception InvalidToneProfile
{
    boolean complexTone;
    boolean simpleTone;
    string msg;
};
```

<table>
<thead>
<tr>
<th>Exception</th>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneProfile</td>
<td>complexTone</td>
<td>This attribute will be changed to FALSE if the elements of the ComplexToneProfile structure are either invalid or out of range.</td>
<td>boolean</td>
</tr>
<tr>
<td></td>
<td>simpleTone</td>
<td>This attribute will be changed to FALSE if the elements of the SimpleToneProfile structure are either invalid or out of range.*</td>
<td>boolean</td>
</tr>
<tr>
<td></td>
<td>msg</td>
<td>A message of type string indicating that the exception has occurred.</td>
<td>string</td>
</tr>
</tbody>
</table>

Note: (*) The valid range for the frequencyInHz attribute of the SimpleToneProfile will be specified by the platform.
A.5.3.2 Audio::AudibleAlertsAndAlarms::InvalidToneId

The InvalidToneId exception is used by the startTone, stopTone, and destroyTone operations to indicate that the tone cannot be started, stopped, or destroyed due to an invalid tonelid.

```cpp
exception InvalidToneId
{
    string msg;
};
```

<table>
<thead>
<tr>
<th>Exception</th>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidToneId</td>
<td>Msg</td>
<td>A message of type string indicating that the exception has occurred.</td>
<td>string</td>
</tr>
</tbody>
</table>

A.5.4 Structures

A.5.4.1 Audio::AudibleAlertsAndAlarms::SimpleToneProfile

The SimpleToneProfileType structure defines attributes, which describes the tone or beep for the Audio Port Device. A beep is created by setting the repeatIntervalInMs attribute of the SimpleToneProfile to 0.

```cpp
struct SimpleToneProfile
{
    unsigned short frequencyInHz;
    unsigned short durationPerBurstInMs;
    unsigned short repeatIntervalInMs;
};
```

<table>
<thead>
<tr>
<th>Struct</th>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
<th>Units</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleToneProfile</td>
<td>frequencyInHz</td>
<td>The frequency in Hz.</td>
<td>unsigned short</td>
<td>Hz</td>
<td>50 - 4000</td>
</tr>
<tr>
<td></td>
<td>durationPerBurstInMs</td>
<td>The duration of tone per burst in milliseconds.</td>
<td>unsigned short</td>
<td>ms</td>
<td>1 - 65535</td>
</tr>
<tr>
<td></td>
<td>repeatIntervalInMs</td>
<td>The repeat interval in milliseconds.</td>
<td>unsigned short</td>
<td>ms</td>
<td>0* – 65535</td>
</tr>
</tbody>
</table>

Note: (*) The repeatIntervalInMs of range 1-65535 will indicate the repeat interval for the tone in milliseconds. A repeatIntervalInMs of value 0 will indicate the creation of a beep.

A.5.4.2 Audio::AudibleAlertsAndAlarms::ComplexToneProfile

The ComplexToneProfileType structure defines attributes to generate complex tones for the Audio Port Device.

```cpp
struct ComplexToneProfile
{
    JTRS::ShortSequence toneSamples;
    unsigned short numberOfRepeats;
};
```

<table>
<thead>
<tr>
<th>Struct</th>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComplexToneProfile</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ComplexToneProfile*  |  toneSamples  |  The tone samples of type ShortSequence (sequence of unsigned short). Tone samples are provided as a sequence of 16 bit linear Pulse Code Modulation (PCM) sampled at 8 kHz.  |  JTRS::ShortSeq (See JTRS CORBA Types [1])

numberOfRepeats  |  The number of repeats for the tone samples.  |  unsigned short

Note: (*) All Complex tones combined can only occupy 3MB or 196608 samples (24 seconds).

A.5.5 Unions

A.5.5.1 Audio::AudibleAlertsAndAlarms::ToneProfileType

The ToneProfileType CORBA::Union type defines attributes, which describes the tone/beep for the Audio Port Device.

```c
union ToneProfileType switch ( ToneDescriminator )
{
    case COMPLEX_TONE:
        ComplexToneProfile complexTone;
    case SIMPLE_TONE:
        SimpleToneProfile simpleTone;
}
```

<table>
<thead>
<tr>
<th>Union</th>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToneProfileType</td>
<td>COMPLEX_TONE</td>
<td>The tone described by the ComplexToneProfileType.</td>
<td>ComplexToneProfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(See A.5.4.2)</td>
</tr>
<tr>
<td></td>
<td>SIMPLE_TONE</td>
<td>The tone/beep described by the SimpleToneProfileType.</td>
<td>SimpleToneProfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(See A.5.4.1)</td>
</tr>
<tr>
<td></td>
<td>ToneDescriminator</td>
<td>Used to make a selection between the ComplexToneProfile or SimpleToneProfile types.</td>
<td>ToneDescriminator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(See A.5.2.1)</td>
</tr>
</tbody>
</table>
# Appendix A.A  Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>CF</td>
<td>Core Framework</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>Hz</td>
<td>Hertz</td>
</tr>
<tr>
<td>ICWG</td>
<td>Interface Control Working Group</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>IDL</td>
<td>Interface Definition Language</td>
</tr>
<tr>
<td>JPEO</td>
<td>Joint Program Executive Office</td>
</tr>
<tr>
<td>JTRS</td>
<td>Joint Tactical Radio System</td>
</tr>
<tr>
<td>ms</td>
<td>Millisecond</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation</td>
</tr>
<tr>
<td>PTT</td>
<td>Push to Talk</td>
</tr>
<tr>
<td>SCA</td>
<td>Software Communications Architecture</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>WF</td>
<td>Waveform</td>
</tr>
</tbody>
</table>
Appendix A.B  Performance Specification

Table 3 provides a template for the generic performance specification for the Audio Port Device API which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 1.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst Case Command Execution Time for audio_alertalarm_wf_provides_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_ptt_uses_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Note: (*) These values should be filled in by individual developers.
B. AUDIO SAMPLE STREAM EXTENSION

B.1 INTRODUCTION

The Audio Sample Stream Extension is based upon the Audio Port Device API (see A.1). It extends the functionality of the common JTRS audio device to provide, consume, and control audio samples to/from the Audio Port Hardware. It retains the methods and attributes defined in the base Audio Port Device API.

B.1.1 Overview

a. Section B.1, Introduction, contains the introductory material regarding the overview, service layer description, modes, states, and referenced documents of this document.

b. Section B.2, Services, specifies the interfaces for the component, port connections, and sequence diagrams.

c. Section B.3, Service Primitives and Attributes, specifies the operations that are provided by the Audio Sample Stream Extension.

d. Section B.4, IDL.

e. Section B.5, UML.

f. Appendix B.A, Abbreviations and Acronyms.

B.1.2 Service Layer Description

B.1.2.1 Audio Sample Stream Extension Port Connections

Figure 7 shows the port connections for the Audio Sample Stream Extension.

Note: All port names are for reference only. Ports in black are defined in the base Audio Port Device API (see Figure 1 – Audio Port Device Port Diagram).

Audio Sample Stream Extension Provides Ports Definitions

audio_sample_stream_provides_port is provided by the Audio Port Device to consume packets through the pushPacket operation.

audio_sample_stream_ctrl_provides_port is provided by the Audio Port Device to set the payload size by the Device User.

audio_sample_msg_ctrl_provides_port is provided by the Audio Port Device to manage the message flows.

Audio Sample Stream Extension Uses Ports Definitions

audio_sample_stream_uses_port is used by the Audio Port Device to set the payload size of the incoming packets from the Device User.

audio_sample_stream_ctrl_uses_port is used by the Audio Port Device to push packets to the Device User.
B.1.3 Modes of Service
Not applicable.

B.1.4 Service States

B.1.4.1.1 Audio Sample Stream Extension Streaming State Diagram

The AudioPort Device streaming states are illustrated in Figure 8.

The two streaming states of the AudioPort Device are as follow:
  • STREAMING - The state transitioned to when the endOfStream indicator in the Packet::StreamControlType of the pushPacket operation is set to FALSE.
  • NOT STREAMING - The state transitioned to upon successful startup and when the endOfStream indicator in the Packet::StreamControlType of the pushPacket operation is set to TRUE.

See Packet API [2] for the definition of Packet::StreamControlType.

Figure 8 – Audio Sample Stream Extension Streaming State Diagram
B.1.5 Referenced Documents

The following documents are additional references not already defined in the base API.

B.1.5.1 Government Documents

B.1.5.1.1 Specifications

B.1.5.1.1.1 Federal Specifications
None

B.1.5.1.1.2 Military Specifications
None

B.1.5.1.2 Other Government Agency Documents


B.1.5.2 Commercial Standards

None
## B.2 SERVICES

### B.2.1 Provide Services

The Audio Sample Stream Extension provides service consists of the Table 4 service ports, interfaces, and primitives, which can be called by other client components. Detailed definition of the interfaces and services shaded in grey is provided by separate documentation identified in the table.

<table>
<thead>
<tr>
<th>Service Group (Port Name)</th>
<th>Service (Interface Provided)</th>
<th>Primitives (Provided)</th>
<th>Parameter Name or Return Value</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Packet::PayloadStatus</td>
<td>getMaxPayloadSize()</td>
<td>Return Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>getMinPayloadSize()</td>
<td>Return Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>getDesiredPayloadSize()</td>
<td>Return Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>getMinOverrideTimeout()</td>
<td>Return Value</td>
</tr>
<tr>
<td>audio_sample_stream_ctrl_provides_port</td>
<td>Audio::StreamControl</td>
<td>Packet::PayloadControl</td>
<td>setMaxPayloadSize()</td>
<td>maxPayloadSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>setMinPayloadSize()</td>
<td>minPayloadSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>setDesiredPayloadSize()</td>
<td>desiredPayloadSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>setMinOverrideTimeout()</td>
<td>minOverrideTimeout</td>
</tr>
<tr>
<td>audio_sample_msg_ctrl_provides_port</td>
<td>Audio::SampleMessageControl</td>
<td>DevMsgCtl::DeviceMessageControl</td>
<td>txActive()</td>
<td>Return Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rxActive()</td>
<td>Return Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>abortTx()</td>
<td>Return Value</td>
</tr>
</tbody>
</table>
### B.2.2 Use Services

The *Audio Sample Stream Extension* use service set consists of the Table 5 service ports, interfaces, and primitives. Since the *Audio Port Device* acts as a client with respect to these services from other components, it is required to connect these ports with corresponding service ports applied by the server component. The *Audio Sample Stream Extension* uses the port name as connectionID for the connection. Detailed definition of the interfaces and services shaded in grey is provided by separate documentation specified in the table.

<table>
<thead>
<tr>
<th>Service Group (Port Name)</th>
<th>Service (Interface Provided)</th>
<th>Primitives (Provided)</th>
<th>Parameter Name or Return Value</th>
<th>Valid Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Packet::UshortStream [2]</td>
<td>getMaxPayloadSize()</td>
<td>Return Value</td>
<td>1 to 16383</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getMinPayloadSize()</td>
<td>Return Value</td>
<td>0 to 512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getDesiredPayloadSize()</td>
<td>Return Value</td>
<td>1 to 16383</td>
</tr>
<tr>
<td></td>
<td>Packet::PayloadStatus [2]</td>
<td>setMaxPayloadSize()</td>
<td>maxPayloadSize</td>
<td>1 to 16383</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setMinPayloadSize()</td>
<td>minPayloadSize</td>
<td>0 to 512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setDesiredPayloadSize()</td>
<td>desiredPayloadSize</td>
<td>1 to 16383</td>
</tr>
<tr>
<td>audio_sample_stream_ctrl_uses_port</td>
<td>Audio::StreamControl</td>
<td>setMinOverrideTimeout()</td>
<td>minOverrideTimeout</td>
<td>0 to 50</td>
</tr>
<tr>
<td></td>
<td>Packet::PayloadControl [2]</td>
<td>setMaxPayloadSize()</td>
<td>maxPayloadSize</td>
<td>1 to 16383</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setMinPayloadSize()</td>
<td>minPayloadSize</td>
<td>0 to 512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setDesiredPayloadSize()</td>
<td>desiredPayloadSize</td>
<td>1 to 16383</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setMinOverrideTimeout()</td>
<td>minOverrideTimeout</td>
<td>0 to 50</td>
</tr>
</tbody>
</table>
B.2.3 Interface Modules

B.2.3.1 Audio Port Device

B.2.3.1.1 Audio Sample Stream Extension

The interface class diagram for the Audio Sample Stream Extension is provided in Figure 9. Interfaces defined in grey are specified in the Device Message Control API [1] or the Packet API [2].

![Interface Class Diagram](image)

Figure 9 – Audio Sample Stream Extension Interface Class Diagram

B.2.3.1.2 SampleStream Interface Description

The interface design of SampleStream is shown in Figure 10. It extends the Packet::UshortStream interface defined in the Packet API [2] to provide the ability to status the audio sample packet sizes and to push audio sample packets to the Audio Port HW.
B.2.3.1.3 SampleStreamControl Interface Description

The interface design of SampleStreamControl is shown in Figure 11. It extends the Packet::PayloadControl interface defined in the Packet API [2] to provide the ability to configure the audio sample packet sizes.
B.2.3.1.4 SampleMessageControl Interface Description

The interface design of SampleMessageControl is shown in Figure 12. It extends the DevMsgCtl::DeviceMessageControl interface defined in the DeviceMessageControl API [1] to determine whether the supporting component is actively processing transmit or receive traffic. It also provides the ability to abort the transmission.

![SampleMessageControl Interface Diagram](image)

B.2.4 Sequence Diagrams

None
B.3 SERVICE PRIMITIVES AND ATTRIBUTES

There are no additional service primitivites and attributes than those defined in the Packet API [2] and the Device Message Control API [1].
B.4 IDL

B.4.1 Audio Sample StreamExt IDL

/*
** AudioSampleStreamExt.idl - JTRS Audio Extension Service Set
*/
#ifndef __AUDIO_SAMPLE_STREAM_EXT_DEFINED
#define __AUDIO_SAMPLE_STREAM_EXT_DEFINED

#ifndef __PACKET_DEFINED
#include "Packet.idl"
#endif

#ifndef __DEVICEMESSAGECONTROL_DEFINED
#include "DeviceMessageControl.idl"
#endif

module Audio
{
    // Packet Consumer
    interface SampleStream : Packet::UshortStream
    {
    }

    // Packet Provider Control
    interface SampleStreamControl : Packet::PayloadControl
    {
    }

    // Abort Messaging
    interface SampleMessageControl : DevMsgCtl::DeviceMessageControl
    {
    }

    #endif //__AUDIO_SAMPLE_STREAM_EXT_DEFINED
B.5 UML

This section contains the Audio Sample Stream Extension component UML diagram.

![Audio Sample Stream Extension Component Diagram](image)

**Figure 13 – Audio Sample Stream Extension Component Diagram**

B.5.1 Data Types

None

B.5.2 Enumerations

There are no additions to the base API.

B.5.3 Exceptions

There are no additions to the base API.

B.5.4 Structures

There are no additions to the base API.
Appendix B.A  Abbreviations and Acronyms

The following lists additional abbreviations and acronyms not defined in the base API Appendix A.A, Abbreviations and Acronyms.

| Tx  | Transmit       |
| Rx  | Receive        |
Appendix B.B  Performance Specification

Table 6 provides a template for the generic performance specification for the *Audio Sample Stream* Extension API which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 7.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst Case Command Execution Time for pushPacket() on audio_sample_stream_provides_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for pushPacket() on audio_sample_stream_uses_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_sample_stream_provides_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_sample_stream_uses_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_sample_stream_ctrl_provides_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_sample_stream_ctrl_uses_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Worst Case Command Execution Time for audio_sample_msg_ctrl_provides_port</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Note: (*) These values should be filled in by individual developers.