



HD Radio™
MPS PAD SDK v1.0

Rev. A
March 20, 2003

Copyright © iBiquity Digital Corporation.

All rights reserved.

The contents of this publication may not be reproduced in any media without the express written permission of iBiquity Digital Corporation.

Trademarks

“iBiquity” and “iBiquity Digital” are trademarks of iBiquity Digital Corporation.

“HD Radio” is a trademark of iBiquity Digital Corporation.

The iBiquity Digital corporate logo and HD Radio logo are trademarks of iBiquity Digital Corporation.

All other trademarks, whether claimed or registered, are the exclusive property of their respective owners.

iBiquity Digital Corporation

8865 Stanford Boulevard

Suite 202

Columbia, MD 21045

410-872-1530 (Phone)

410-872-1531 (FAX)

info@ibiquity.com

Table of Contents

TABLE OF CONTENTS	III
LIST OF FIGURES	IV
LIST OF TABLES	IV
OVERVIEW	V
Purpose.....	v
Scope.....	v
Audience.....	v
Introduction.....	v
Documents Contents.....	v
REFERENCED DOCUMENTS	VI
ACRONYMS, SYMBOLS, AND CONVENTIONS	VII
Acronyms.....	vii
Conventions.....	vii
1. MAIN PROGRAM SERVICE	1
1.1. Overview.....	1
2. APPLICATION PROGRAMMING INTERFACE	2
2.1. Overview.....	2
2.2. System Requirements.....	2
2.3. Installation.....	2
2.4. COM Interface.....	2
2.4.1. Core Interface Properties.....	2
2.4.2. Control Interface Properties.....	4
2.4.3. Interface Methods.....	4
2.5. Coding Examples.....	5
2.5.1. Exception Error Handling.....	5
2.5.2. Transfer Method.....	6
2.5.3. Reference Identifier Methods.....	7
2.5.4. Reset Message Method.....	9
2.5.5. File I/O Methods.....	10
3. PAD TOOLS	12
3.1. Overview.....	12
3.2. MPS PAD Generator.....	12
3.2.1. System Requirements.....	12
3.2.2. Installation.....	12
3.2.3. User interface.....	13
3.2.4. Composer.....	13
3.2.5. Transfer Contents.....	14
3.2.6. Control.....	14
3.2.7. Status.....	14
3.3. MPS PAD Monitor.....	14
3.3.1. System Requirements.....	14
3.3.2. Installation.....	14

3.3.3. User Interface	15
3.3.4. Capturing PAD.....	16
3.3.5. Configuration Options	16
3.3.6. Statistics.....	17

List of Figures

Figure 1-1 MPS System Flow.....	1
Figure 2-1 MPS PAD COM Component Server	2
Figure 3-1 MPS PAD Test Configurations	12
Figure 3-2 MPS PAD Generator Main Interface.....	13
Figure 3-3 MPS PAD Monitor Main Interface.....	15
Figure 3-4 MPS PAD Monitor Detailed View	16
Figure 3-5 MPS PAD Monitor Statistics	17

List of Tables

Table 2-1 Core PAD Properties	3
Table 2-2 Control PAD Properties	4
Table 2-3 Core PAD Methods	5
Table 2-4 Additional PAD Methods	5

Overview

Purpose

The purpose of this document is to provide a technical reference for the MPS PAD SDK.

Scope

The scope of this document is limited to the creation and transmission of MPS PAD.

Audience

The document is intended for software developers interested in creating applications that produce program associated data for the HD Radio™ Main Program Service.

Introduction

The HD Radio™ MPS PAD SDK is a software development kit that makes creating Program Associated Data (PAD) applications easy and fast. The MPS PAD Application Programming Interface (API) allows applications to input data content and send it to HD Radio transmission equipment without having to worry about the details of data encoding and transfer protocols. The SDK also includes the MPS PAD Generator and PAD Monitor test applications, allowing full validation and demonstration of your MPS PAD application. These test applications allow you to simulate MPS PAD broadcasting on your private development LAN, stream-lining the development LAN, accelerating development and integration cycles.

Documents Contents

This document includes the following:

- Overview the Main Program Service
- Detailed description of MPS PAD API
- Description of MPS PAD Generator and Monitor Tools

Referenced Documents

- [1] iBiquity Digital Corporation, “HD Radio™ Air Interface Design Description – Main Program Service Data” Doc. No. SY_IDD_1028s

Acronyms, Symbols, and Conventions

Acronyms

Acronym	Definition
API	Application Programming Interface
MPS	Main Program Service
PAD	Program Associated Data
SDK	Software Development Kit

Conventions

Unless otherwise noted, the following conventions apply to this document:

- Information enclosed in braces { } is either unavailable at the present time or subject to change.
- All vectors are indexed starting with 0.
- The element of a vector with the lowest index is considered to be first.
- In drawings and tables, the leftmost bit is considered to occur first in time in time.
- Bit 0 of a byte or word is considered the least significant bit.
- When presenting the dimensions of a matrix, the number of rows is given first (e.g., an $n \times m$ matrix has n rows and m columns).
- In timing diagrams, earliest time is on the left.
- Binary numbers are presented with the most significant bit having the highest index.
- In representations of binary numbers, the least significant bit is on the right.

1. Main Program Service

1.1. Overview

The Main Program Service (MPS) is a direct extension of traditional analog radio programming. MPS allows the transmission of existing analog radio-programming in both analog and digital formats, which enables smooth transition from analog to digital radio. Radio receivers that are not HD Radio enabled can continue to receive the traditional analog radio signal, while HD Radio receivers can receive both digital and analog signals via the same frequency band. In addition to digital audio, MPS includes program associated data (PAD) that correlates with the audio programming. The PAD contains information which describes or complements the audio. PAD includes items such as title, artist, and album.

The Main Program Service audio and data are synchronized at the broadcast studio. That is, the PAD is transmitted so that radio receivers can acquire it at the same time the audio program that it describes is being heard by radio listeners.

The MPS audio is a continuous stream of audio content, identical to traditional analog audio programming. The PAD is a message-based content that is sent from the PAD source, such as a studio automation system, into the HD Radio broadcast transmission system as the audio program content changes (e.g., when a new song, program feature, or announcement occurs). The HD Radio broadcast system continuously transmits the PAD. This allows receivers to tune to the radio station at arbitrary points and still receive the program-associated data along with the audio.

At the HD Radio receiver, the encoded MPS audio and PAD is acquired, decoded, and delivered to the appropriate destination, such as audio speaker, display, or storage media. Figure 1-1 shows the flow of MPS.

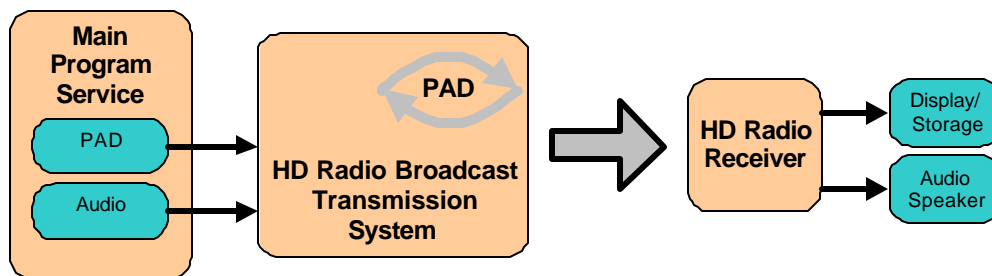


Figure 1-1 MPS System Flow

2. Application Programming Interface

2.1. Overview

The MPS PAD API is a software application programming interface intended for applications that generate PAD source. The API enables PAD applications to easily create and transmit PAD content. Applications simply pass desired PAD contents, such as title and artist name, to the API and invoke PAD API functions to transfer the PAD to the HD Radio broadcast transmission systems. Applications are required to have IP network connectivity to the HD Radio broadcast transmission system.

The MPS PAD API is implemented using the Microsoft™ COM (Common Object Model) component server technology. As a stand alone and fully distributable software component, the *MPS PAD COM component server* can be easily integrated into new and existing software systems.

2.2. System Requirements

The MPS PAD API requires use of either Microsoft Windows™ operating system 2000, NT 4.0, or XP.

2.3. Installation

To install the MPS PAD API, run the MPS PAD SDK installation program and check the MPS PAD API selection box in the install shield option menu. By default, the MPS PAD API will be installed. The Install Shield program will setup and register the MPS PAD COM component server DLL.

2.4. COM Interface

The MPS PAD COM component server is an Automation Server that implements both IDispatch and IUnknown interfaces, and supports client applications developed in different development environments including Visual C++, Visual Basic, and Visual J++.

Figure 2-1 depicts the interface structure of the MPS PAD COM server.

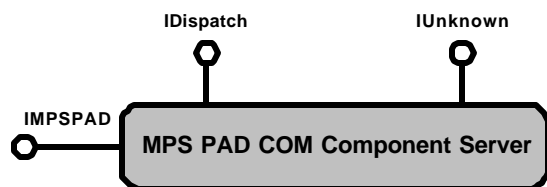


Figure 2-1 MPS PAD COM Component Server

2.4.1. Core Interface Properties

The MPS PAD COM component server provides an interface for MPS PAD creation, manipulation and transmission. The core PAD features include *Title*, *Artist*, *Album*, *Genre*, *Comment* and *Commercial* information. Features such as the *Comment* and *Commercial* consist of multiple properties. The *Comment* feature allows additional free-form text to be added to the PAD message. It consists of two properties: *comment title* and *comment*. The *Commercial* feature consists of multiple properties, which describe product or service information used for commercial advertisement and sale. The commercial properties include: *Price*, *Expiration Date*, *Contact URL*, *Received As*, *Name of Seller*, and *Product Description*.

Table 2-1 gives a description of the core properties.

	Category	Property	Description	Input Type (Visual C++)	Content Description By Programming Category		
					Music	Talk	Announcement
1.	General	title	One-line Title Name	BSTR	Song title	Talk Topic	Announcement or Advertisement Title
		artist	Performer, Originator, Author, Sponsor	BSTR	Artist Name	Show Host	Author/Sponsor
		album	Content Source	BSTR	Album Name	Show Name	Sponsor Name
		genre	Categorization of content. This is an enumerated field of predefined types based on ID3v2 categorization.	BSTR	e.g., (8) Jazz, (17) Rock, (32) Classical	(101) Speech	(101) Speech
2.	Comment	commentTitle	One-line Title for Comment Description	BSTR	Comment Title	Comment Title	Comment Title
		comment	Comment Description. Detailed explanation, user callback information or further information.	BSTR	web site, contact or other info	Talk Show call-in number, or other show info	Announcement or Advertising statement, Point-of-sale or more info
3.	Commercial	price	Price of merchandise	BSTR	The commercial frame facilitates advertisement and sale of products and services. When using the commercial feature, all commercial properties are required.		
		validUntil	Expiration date for transaction	DATE			
		contactURL	URL identifier used to contact the seller. Can be used to initiate purchase transaction via an external return channel, such as a cellular phone network.	BSTR			
		recvAs	Method in which merchandise is received (e.g., over the internet). Values range from 0-8 00 Other 01 Standard CD album with other songs 02 Compressed audio on CD 03 File over the Internet 04 Stream over the Internet 05 As note sheets 06 As note sheets in a book with other sheets 07 Music on other media 08 Non-musical merchandise	SHORT			
		sellerName	Text identifying seller	BSTR			
		productDesc	Textual description of advertisement	BSTR			

Table 2-1 Core PAD Properties

A property can be removed by setting its BSTR value to empty string (""). To remove the commercial category, all commercial BSTR properties must be set to empty string (""). The *recvAs* property must be set to a value of negative one (-1) and the *validUntil* property can be ignored.

2.4.2. Control Interface Properties

In addition, to the core PAD properties, control properties are provided to support PAD content transfer and import/export of PAD to ID3v2 format. The Control Interface Properties are listed in Table 2-2.

Control Property	Input Type (Visual C++)	Description																															
1. destAddr	BSTR	Property to set or get destination IP address used to transfer PAD to the broadcast transmission system. Can be set to explicit IP address (e.g., 0.0.0.0) or hostname string.																															
2. msgLength	SHORT	Property to get the length of the PAD message. This property is read-only. The maximum PAD message size for a single PAD transfer is 1024 bytes, where a byte is an 8-bit octet. This property can be used check the size of the PAD message. Note, the <i>effective received</i> PAD message can exceed 1024 bytes, by using the PADLINK reference identifier to transfer a PAD message, as multiple messages.																															
3. referenceCount	SHORT	Property to get the number of reference IDs contained in PAD message. This property is read-only.																															
4. id3Tag	VARIANT	Property to set or get ID3v2.3 formatted tag. This property can be used to import or export PAD directly from or to an ID3v2 tag. The Mapping of core PAD properties to ID3v2 are as follows: <table border="1" data-bbox="646 871 1344 1360"> <thead> <tr> <th>Property</th> <th>ID3 Frame</th> <th>ID3v2 Field</th> </tr> </thead> <tbody> <tr> <td>Title</td> <td>TIT2</td> <td>Info</td> </tr> <tr> <td>Artist</td> <td>TPE1</td> <td>Info</td> </tr> <tr> <td>Album</td> <td>TALB</td> <td>Info</td> </tr> <tr> <td>Genre</td> <td>TCO3</td> <td>Info</td> </tr> <tr> <td rowspan="2">Comment</td> <td rowspan="2">COMM</td> <td>short description field</td> </tr> <tr> <td>Content field</td> </tr> <tr> <td rowspan="6">Commercial</td> <td rowspan="6">COMR</td> <td>Price</td> </tr> <tr> <td>Valid Until</td> </tr> <tr> <td>Contact URL</td> </tr> <tr> <td>Received as</td> </tr> <tr> <td>Name of seller</td> </tr> <tr> <td>Description</td> </tr> <tr> <td rowspan="2">Reference Identifier</td> <td rowspan="2">UFID</td> <td>Owner Identifier</td> </tr> <tr> <td>Identifier</td> </tr> </tbody> </table>	Property	ID3 Frame	ID3v2 Field	Title	TIT2	Info	Artist	TPE1	Info	Album	TALB	Info	Genre	TCO3	Info	Comment	COMM	short description field	Content field	Commercial	COMR	Price	Valid Until	Contact URL	Received as	Name of seller	Description	Reference Identifier	UFID	Owner Identifier	Identifier
Property	ID3 Frame	ID3v2 Field																															
Title	TIT2	Info																															
Artist	TPE1	Info																															
Album	TALB	Info																															
Genre	TCO3	Info																															
Comment	COMM	short description field																															
		Content field																															
Commercial	COMR	Price																															
		Valid Until																															
		Contact URL																															
		Received as																															
		Name of seller																															
		Description																															
Reference Identifier	UFID	Owner Identifier																															
		Identifier																															

Table 2-2 Control PAD Properties

2.4.3. Interface Methods

The MPS PAD COM component server contain methods for transferring PAD content to an HD Radio broadcast transmission system, basic file I/O functions used in conjunction with importing and exporting of PAD in ID3v2 format, and creation of Reference Identifiers.

Reference Identifiers are used to correlate or link multiple PAD messages that have been transmitted separately into a single PAD message on the receiver device. In addition, reference identifiers can be used for future identification, cross-reference or correlation of PAD content with other content. Multiple Reference Identifiers can be contained in a single PAD message. Each Reference Identifier contains two fields. The first field is the Owner Identifier string, followed by the second field, which contains a unique numeric identifier that can be up to 64 bytes long. All Reference Identifiers contained in a PAD message must have unique Owner Identifiers. Use of Reference Identifiers is optional.

If Owner Identifier is set to PADLINK, then the identifier contains a unique message identifier, which allows a single PAD message to be broken into one or more PAD messages for broadcast transmission.

Each individual PAD message is a complete decodable message, but may only contain a subset of the PAD content. For example, the title, artist and album may be transmitted in one PAD message and the commercial information may be transmitted separately. Both messages contain a common unique identifier, which is the mechanism for correlating the complete PAD message. Collectively the messages represent a single PAD message. The combined messages are treated as one PAD message on the receiver device. The numeric identifier for the PADLINK reference contains a unique number ranging from 0 to 65,535. This number is unique only to the Main Program Service instance of a given broadcast station. Table 2-3 shows the core PAD methods.

	Method Name	Input Parameter	Output Parameter	Return Parameter	Description
1.	Transfer	None	None	HRESULT	Method to send MPS PAD to HD Radio Broadcast Transmission System
2.	addReference	BSTR ownerId, VARIANT refId	None	HRESULT	Method to add reference ID pair
3.	getReferenceByOwner	BSTR ownerID,	VARIANT *pRefID	HRESULT	Method to retrieve reference ID pair identified by the owner ID string
4.	getFirstReference	None	SHORT *pStatus, BSTR *pOwnerId, VARIANT *pRefID	HRESULT	Method to retrieve the first Reference ID pair
5.	getNextReference	None	SHORT *pStatus, BSTR *pOwnerId, VARIANT *pRefID	HRESULT	Method to retrieve the next Reference ID pair after the first
6.	removeReference	BSTR ownerID	None	HRESULT	Method to remove reference ID pair identified by the owner ID string
7.	removeAllReferences	None	None	HRESULT	Method to remove all reference ID pairs

Table 2-3 Core PAD Methods

In addition to the Core MPS PAD Methods defined above, methods are provided to reset the PAD message values, and import/export PAD in ID3v2 format. Additional methods are shown in Table 2-4.

	Method Name	Input Parameter	Output Parameter	Return Parameter	Description
1.	resetMsg	None	None	HRESULT	Method to clear all PAD content from the message. In general, all values are persistent from one invocation (transfer) to the next and must be explicitly removed or updated if not desired in the transfer of MPS PAD.
2.	loadFile	BSTR filename	None	HRESULT	Method to read ID3v2 formatted tag from a file and import it value into PAD message
3.	saveFile	BSTR filename	None	HRESULT	Method to export PAD message to a file as an ID3v2 formatted tag

Table 2-4 Additional PAD Methods

2.5. Coding Examples

This section contains coding samples using the MPS PAD API with Visual C++ and Visual Basic.

2.5.1. Exception Error Handling

All methods throw exception errors and these exceptions have to be caught and handled by the client software.

2.5.2. Transfer Method

Syntax: **HRESULT transfer()**

Parameters:

None

Visual Basic sample code

```
Private Sub TestTransfer()  
On Error Goto Err_Handler  
Dim MPS as new MPSPAD  
' set dest hostname  
MPS.destAddr = "HDRADIOHOST"  
' set MPS PAD values through property set  
MPS.artist = "ABC"  
MPS.album = "The Best of ABC"  
  
' Send MPS PAD  
MPS.transfer  
Exit Sub  
Err_Handler:  
' Handle error  
End Sub
```

Visual C++ sample code

```
IMPSPAD mpsPAD;  
try  
{  
    // set dest hostname and dest port number at run time  
    mpsPAD.SetDestAddr("HDBROADCASTHOST");  
    mpsPAD.SetDestPort(5500);  
  
    // set MPS PAD values through property set  
    mpsPAD.SetTitle("Test Title");  
    mpsPAD.SetArtist("Test");  
    Or  
    // Load MPS PAD from a data file  
    BSTR bstrFileName("test.dat");  
    mpsPAD.loadFile(bstrFileName);  
  
    // send MPS PAD  
    mpsPAD.transfer();  
}  
catch(...)  
{  
}
```

2.5.3. Reference Identifier Methods

Syntax: **HRESULT addReference(BSTR bstrOwner, VARIANT refID)**

Parameters:

bstrOwner - owner ID string
 refID - byte array contains the reference ID

Visual Basic sample code

```
Private Sub TestAddRef()
On Error Goto Err_Handler
Dim MPS as new MPSPAD
' set owner id string
Dim strOwner as String
strOwner = "PADLINK"
' set reference ID
Dim refID(1) as Byte
refID(0) = 78
refID(1) = 123
MPS.addReference strOwner, refID
Exit Sub
Err_Handler:
' Handle error
End Sub
```

Syntax: **HRESULT getReferenceByOwner(BSTR bstrOwner, VARIANT *pRefID)**

Parameters:

bstrOwner - owner ID string
 pRefID - pointer to the returned reference ID byte array

Visual Basic sample code

```
Private Sub TestGetRef()
On Error Goto Err_Handler
Dim MPS as new MPSPAD
' set owner id string
Dim strOwner as String
strOwner = "PADLINK"
' get reference ID
Dim refID as Variant
MPS.getReferenceByOwner strOwner, refID
' display reference ID in hex
Dim strRef As String
For idx = 0 To UBound(refID)
strRef = strRef & Hex(refID(idx)) & " "
Next idx
txtRef.Text = strRef
Exit Sub
Err_Handler:
' Handle error
End Sub
```

Syntax: **HRESULT** **getFirstReference**(short *pStatus, BSTR *pbstrOwner, VARIANT *pRefID)

Parameters:

pStatus - pointer to the returned status, 1 – successful, 0 – failed
 pbstrOwner - pointer to the returned owner ID string
 pRefID - pointer to the returned reference ID byte array

Visual Basic sample code

```
Private Sub TestGetRef()
On Error Goto Err_Handler
  Dim MPS as new MPSPAD
  refCount = 0
  Dim strOwner as String
  Dim refID as Variant
  Dim status as Integer
  MPS.getFirstReference status, strOwner, refID
  While (status = 1)
    ' display reference ID
    refCount = refCount + 1
    MPS.getNextReference status, strOwner, refID
  Wend
Exit Sub
Err_Handler:
  ' Handle error
End Sub
```

Note: The Get First Reference ID method has to be called first prior to calling the Get Next Reference ID method.

Syntax: **HRESULT** **getNextReference**(short *pStatus, BSTR *pbstrOwner, VARIANT *pRefID)

Parameters:

pStatus - pointer to the returned status, 1 – successful, 0 – failed
 pbstrOwner - pointer to the returned owner ID string
 pRefID - pointer to the returned reference ID byte array

VB sample code:

```
Private Sub TestGetRef()
On Error Goto Err_Handler
  Dim MPS as new MPSPAD
  refCount = 0
  Dim strOwner as String
  Dim refID as Variant
  Dim status as Integer
  MPS.getFirstReference status, strOwner, refID
  While (status = 1)
    ' display reference ID
    refCount = refCount + 1
    MPS.getNextReference status, strOwner, refID
  Wend
Exit Sub
Err_Handler:
  ' Handle error
End Sub
```

Syntax: **HRESULT removeReference(BSTR bstrOwner)**

Parameters:

bstrOwner - owner ID string that identifies the Reference ID pair which is to be removed

Visual Basic sample code

```
Private Sub TestRemoveRef()
On Error Goto Err_Handler
Dim MPS as new MPSPAD
' set owner id string
Dim strOwner as String
strOwner = "PADLINK"
' gett reference count
IstResult.AddItem "Ref Count before removal = " & CStr(MPS.referenceCount)
' remove reference owned by PADLINK
MPS.removeReference strOwner
IstResult.AddItem "Ref Count after removal = " & CStr(MPS.referenceCount)
Exit Sub
Err_Handler:
' Handle error
End Sub
```

Syntax: **HRESULT removeAllReferences()**

Parameters: none

Visual Basic sample code

```
Private Sub TestRemoveRef()
On Error Goto Err_Handler
Dim MPS as new MPSPAD
' gett reference count
IstResult.AddItem "Ref Count before removal = " & CStr(MPS.referenceCount)
' remove reference owned by www.abc.com
MPS.removeAllReferences
IstResult.AddItem "Ref Count after removal = " & CStr(MPS.referenceCount)
Exit Sub
Err_Handler:
' Handle error
End Sub
```

2.5.4. Reset Message Method

Syntax: **HRESULT resetMsg ()**

Parameters: none

Visual Basic sample code

```
Private Sub TestReset()
On Error Goto Err_Handler
Dim MPS as new MPSPAD
' set dest hostname
MPS.destAddr = "HDRADIOHOST"
' set MPS PAD values through property set
MPS.artist = "ABC"
MPS.album = "The Best of ABC"

' Reset PAD contents
MPS.resetMsg
Exit Sub
Err_Handler:
' Handle error
End Sub
```

2.5.5. File I/O Methods

Syntax: **HRESULT saveFile(BSTR *filename)**

Parameters:

filename – a pointer to the string contains the full path and file name of the data file

Visual Basic sample code

```
Private Sub TestSaveData()  
On Error Goto Err_Handler  
    Dim MPS as new MPSPAD  
    MPS.artist = "ABC"  
    MPS.album = "The Best of ABC"  
    Dim datafile as String  
    datafile = "test.dat"  
    MPS.saveFile datafile  
Exit Sub  
Err_Handler:  
    ' Handle error  
End Sub
```

Visual C++ sample code

```
IMPSPAD mpsPAD;  
  
try  
{  
    // Set ID3 frame data  
    mpsPAD.SetTitle("Test Title");  
    mpsPAD.SetArtist("Test");  
  
    // Save ID3 data to a data file  
    BSTR bstrFileName("test.dat");  
    mpsPAD.saveFile(bstrFileName);  
}  
catch(...)  
{  
}
```

Syntax: **HRESULT loadFile(BSTR *filename)**

Parameters:

filename – a pointer to the string contains the full path and file name of the data file

Visual Basic sample code

```
Private Sub TestLoadData()  
On Error Goto Err_Handler  
    Dim MPS as new MPSPAD  
    Dim datafile as String  
    datafile = "test.dat"  
    MPS.loadFile datafile  
    ' display data  
    Exit Sub  
Err_Handler:  
    ' Handle error  
End Sub
```

Visual C++ sample code

```
IMPSPAD mpsPAD;  
  
try  
{  
    // Load data from a ID3 data file  
    BSTR bstrFileName("test.dat");  
    mpsPAD.loadFile(bstrFileName);  
    // Display data in a listbox  
    m_ListBox.AddString(mps PAD.GetComment());  
}  
catch(...)  
{  
}
```

3. PAD Tools

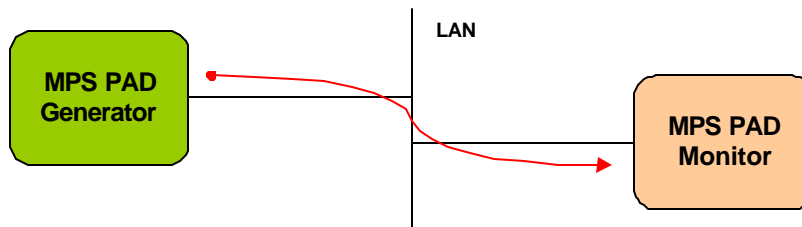
3.1. Overview

The MPS PAD SDK includes software tools that allow PAD application developers to test and verify their application. Two tools are provided: The MPS PAD Generator and the MPS PAD Monitor.

The MPS PAD Generator is a fully compliant MPS PAD application. Internally, the MPS PAD Generator uses the same MPS PAD COM component server supplied with the SDK to create and transfer PAD messages. The MPS PAD Generator is an end-user tool where MPS PAD messages can be constructed via a graphical user interface and transmitted to the HD Radio broadcast transmission system or sent directly to the MPS PAD Monitor.

The MPS PAD Monitor simulates a receiver application that accepts MPS PAD messages. The MPS PAD Monitor is an end-user tool that can be used in conjunction with the MPS PAD Generator or a custom PAD application. With the PAD Monitor, a compliant PAD application can verify that it's generating valid PAD messages and analyze PAD traffic. The Figure 3-1 shows the MPS PAD Monitor configurations.

Configuration 1 - MPS PAD Generator to Monitor



Configuration 2 - Custom PAD Application to MPS PAD Monitor

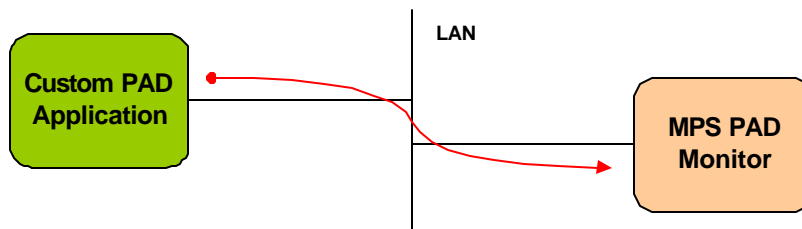


Figure 3-1 MPS PAD Test Configurations

3.2. MPS PAD Generator

3.2.1. System Requirements

The MPS PAD Generator requires use of either Microsoft Windows™ operating system 2000, NT 4.0, or XP.

3.2.2. Installation

To install the MPS PAD Generator, run the MPS PAD SDK installation program and check the MPS PAD Generator selection box in the install shield option menu. If the default options are used, the MPS PAD Generator will be installed. Once installed, the MPS PAD Generator can be found in the *Start*→*Programs*→*HD Radio*→*MPS PAD SDK* submenu.

3.2.3. User interface

The MPS PAD Generator’s main user interface is shown in Figure 3-2. Most of the MPS PAD features are visible from the main interface. The interface is divided into the following regions: Composer, Transfer Contents, Control and Status.

The primary functions in the main user interface are described below. For additional information, refer to the *Help* menu option in the MPS PAD Generator.

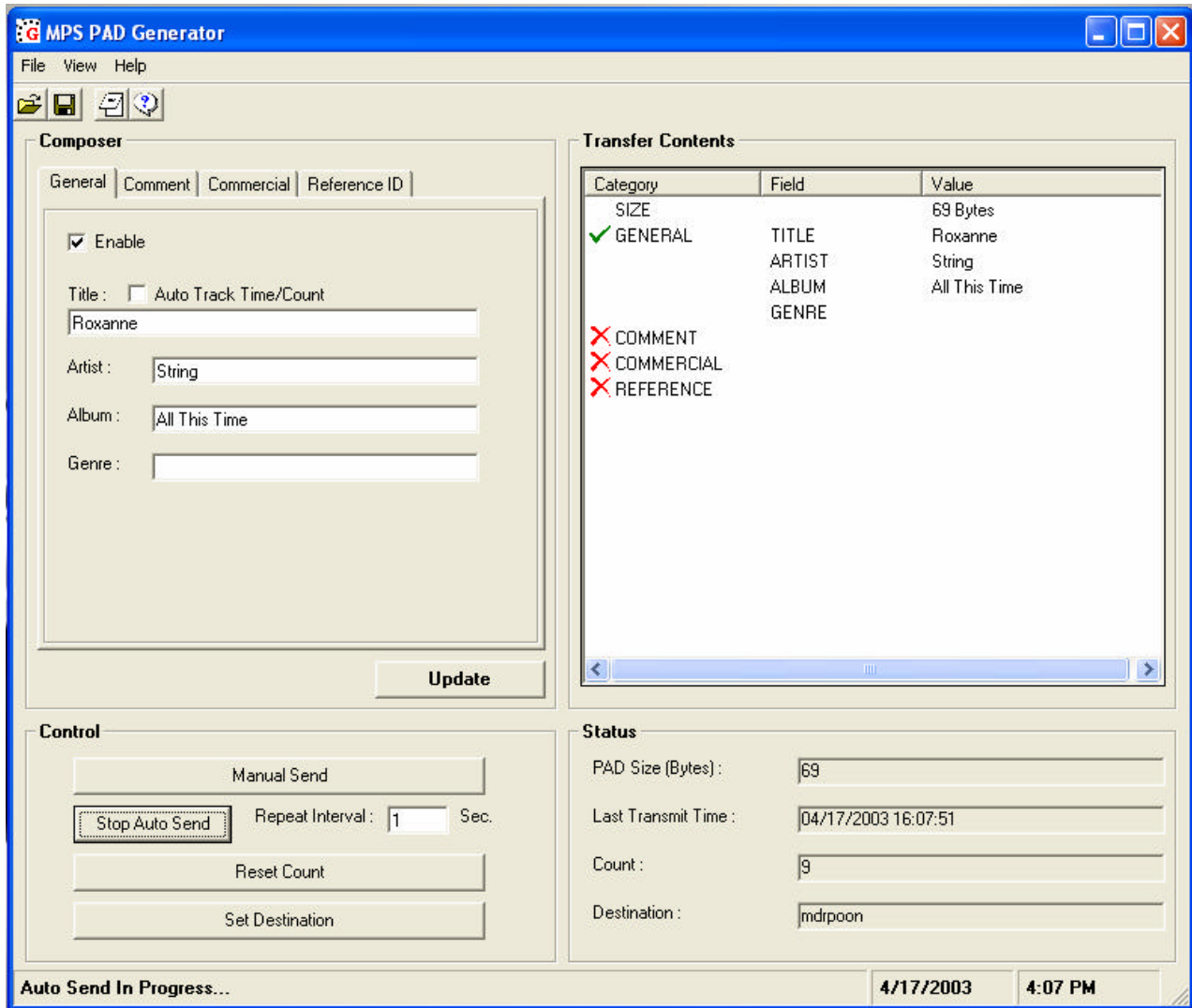


Figure 3-2 MPS PAD Generator Main Interface

3.2.4. Composer

The Composer region is where the MPS PAD message content is defined. To add content, select the desired content category tab (e.g., General, Comment, Commercial, or Reference ID) and enter data in appropriate fields. Once data has been entered, the *Update* button must be pressed before PAD changes can be transmitted. Once the *Update* button is selected, changes appear in the Transfer Contents region.

Each content category section has an *Enable* checkbox; this box must be checked if the content is to be transmitted. Note, a \checkmark or \times symbol is displayed in the Transfer Contents area indicating which content sections are enabled for transfer.

The *Title* field within the General Category has a special *Auto Track* option. When this option is checked, the current time and an incremental counter will be added to the Title field along with any other text. Each time a PAD message is transmitted the counter value is incremented and the current time is retrieved from the system clock transmitted in the Title field of the outgoing message. Auto Track allows the MPS PAD Monitor to approximate end-to-end transit delay of PAD messages.

3.2.5. Transfer Contents

The Transfer Contents region displays the PAD message that will be transmitted. This area shows the category, field, and field value. A *Ů* or *X* symbol is displayed in the Transfer Contents area indicating which content sections are enabled for transfer. At least one category must be enabled to transmit PAD.

3.2.6. Control

The Control region is where transmission of PAD messages is managed. PAD messages can be transmitted manually or automatically. Manual transmission transfers a PAD message (as represented in the Transfer Contents region) each time the *Manual Send* button is pressed. Automatic transmission continuously sends a PAD message at a specified repeat interval. To initiate automatic transmission of PAD messages, set the desired repeat interval, using the *Repeat Interval* field and press the *Start Auto Send* button. Once automatic transmission has been started, the *Start Auto Send* button will be changed to *Stop Auto Send*. To stop automatic transmission, press the *Stop Auto Send* button. The manual send can still be used while automatic transmission is in progress. A log of transmitted PAD messages can be viewed by selecting the *View→Transfer Log* menu.

Prior to Transmitting PAD messages, the PAD message destination IP address must be specified. The actual IP address value will depend on the HD Radio broadcaster's network IP configuration. For testing with the MPS PAD monitor the destination IP address should be set to the IP address of the PC hosting the MPS PAD Monitor. The destination IP address can be set by using the *Set Destination* button.

Each time a PAD message is transmitted a counter is incremented to track the number of messages that have been sent. The counter value can be reset by using the *Reset Count* button. This counter is the same counter used by the auto track feature, which was discussed previously in the Composer section.

3.2.7. Status

The status region shows the following information related to PAD transmission:

- PAD Size – length in bytes of the last transmitted PAD message
- Last Transmit Time – date and time stamp the last PAD message was sent
- Count – Incremental counter identifying the number of transmitted PAD messages
- Destination – IP network destination address specifying where PAD message is transmitted

3.3. MPS PAD Monitor

3.3.1. System Requirements

The MPS PAD Monitor requires use of either Microsoft Windows™ operating system 2000, NT 4.0, or XP.

3.3.2. Installation

To install the MPS PAD Monitor, run the MPS PAD SDK installation program and check the MPS PAD Monitor selection box in the install shield option menu. If the default options are used, the MPS PAD Monitor will be installed. Once installed, the MPS PAD Monitor can be found in the *Start→Programs→HD Radio→MPS PAD SDK* submenu.

3.3.3. User Interface

The MPS PAD Monitor's main user interface is shown in Figure 3-3. The main window shows a list of received (or captured) PAD messages. An abbreviated view of the general PAD fields (e.g., Title, Artist, and Album) is shown for each message. In addition to the PAD messages, the actual capture start and ending events are also shown. For the start and end capture events, the title field displays *Begin Capturing* and *End Capturing*, respectively.

A detailed view of a PAD message can be seen by double-clicking on the PAD message of interest or by selecting the PAD message of interest and selecting *View→PAD Details*. The Detailed View window is shown in Figure 3-4. The Detailed View has a *Sync* option which is used to control whether the detail view is updated as PAD messages are received or displays only the PAD message of interest. To view PAD messages details as they are captured select the *Sync Yes* option. Otherwise, to view only a message of interest, select the *Sync No* option.

The detailed view is capable of displaying PAD messages that are correlated (or linked together) by PAD Reference Identifiers. For example, if a PAD message is transmitted as two separate PAD messages, where one message contains the artist & title information, and the other PAD message contains commercial information, then the *PADLINK* reference identifier (contained in both messages) will have a common unique numeric identifier, which indicates that the two messages collectively constitute a single PAD message. If both messages are received, the MPS PAD Monitor can display the two messages as a single message using the *Merge* option or create navigation links between the two messages. The *Links* region of the MPS PAD Detail View window shows the controls for navigating and merging PAD messages.

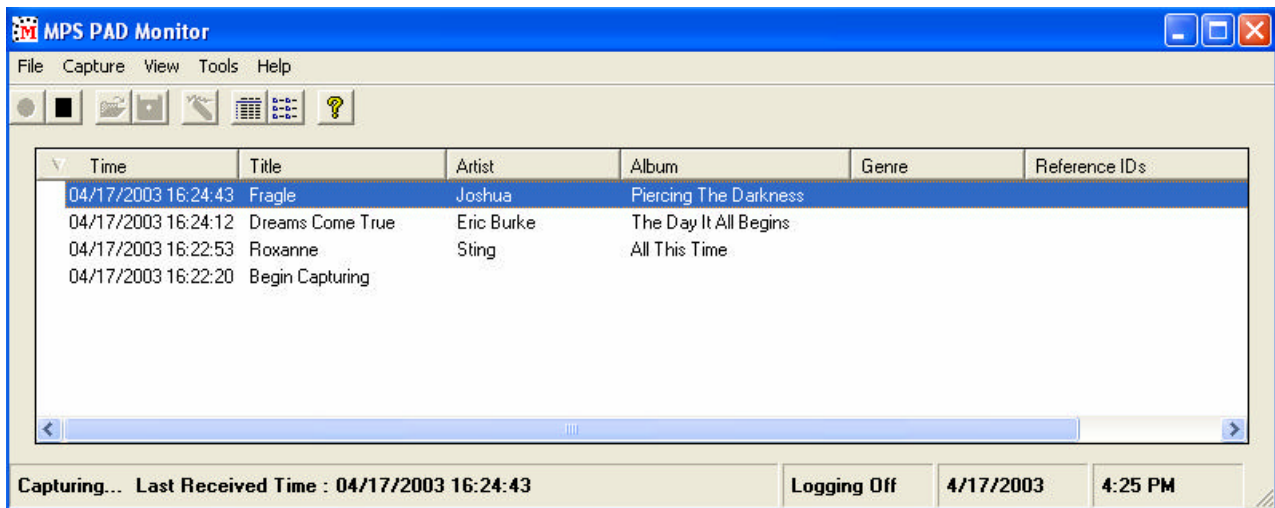


Figure 3-3 MPS PAD Monitor Main Interface

The details of each PAD messages can also be viewed using the Detailed View as shown in Figure 3-4.

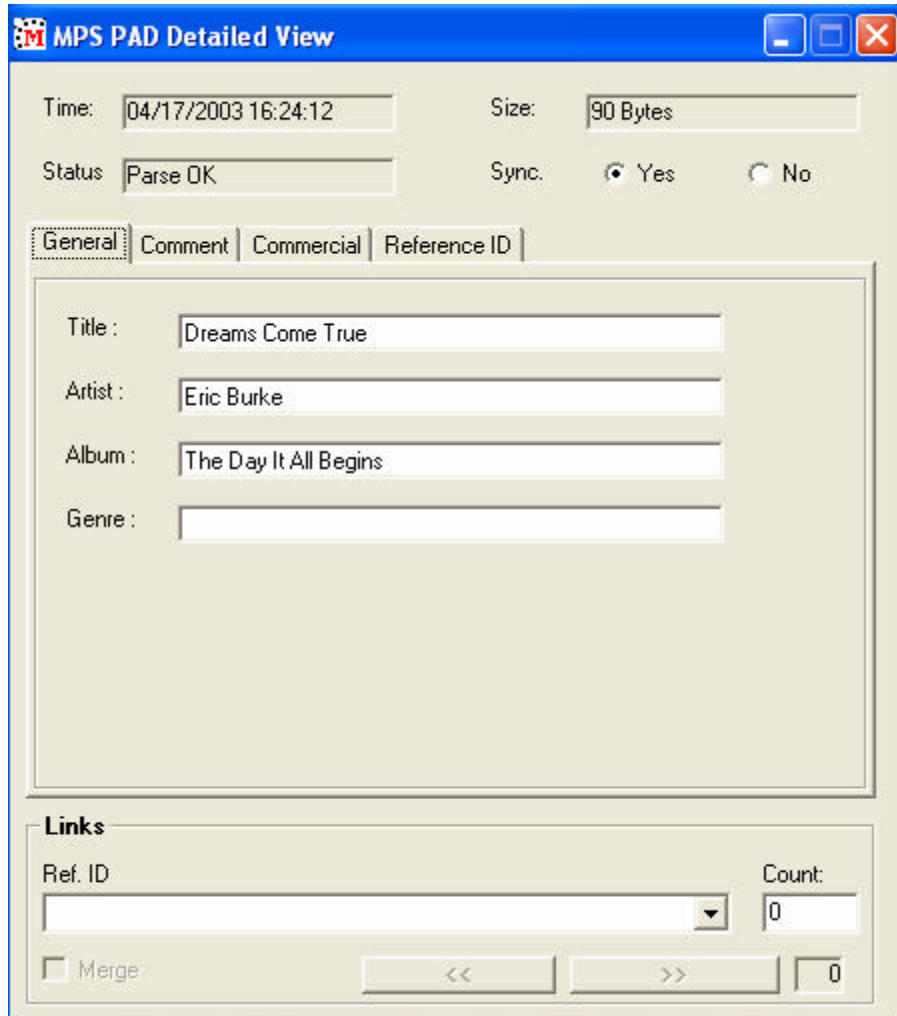


Figure 3-4 MPS PAD Monitor Detailed View

3.3.4. Capturing PAD

Capturing is initiated by selecting the *Capture→Begin Capture* menu option or by using the start capture icon from the main interface. Conversely, the MPS PAD Monitor will stop receiving messages when the *Capture→End Capture* menu option or the stop capture icon is selected from the main interface. Captured messages are displayed in the main interface window as they are received. If the *Sync* option has been enabled, the detailed view window will also display the most recently received message. If logging is enabled, each received PAD message will also be logged to file. Logging is discussed in the Configuration Options section.

3.3.5. Configuration Options

The PAD Monitor has the following configuration options: Direct Connection, Logging, Start Up View, and History Window Size. The options are accessed via the *Tools→Options* menu.

- Direct Connection – Direct connection specifies whether the PAD Generator or custom PAD Application is communicating directly with the MPS PAD Monitor (e.g., over a LAN). This option should always be checked.

- Logging – If Logging is enabled, captured PAD messages are saved to the specified log file, where they can latter be viewed using the *File→Open Log*. If logging is *not* enabled, captured PAD messages that are displayed in the main interface can be saved by using *File→Save Log*.
- Start Up View – By default the list view (main interface) is shown at startup. This option allows the *Detail View* to also be displayed at startup.
- History Window Size – The History Window Size specifies the maximum number of PAD messages contained in the list view of the main interface. The History Window Size is also used to compute PAD traffic average statistics. The History Window Size can range from 16-10,000.

3.3.6. Statistics

The MPS PAD Monitor calculates statistics for captured PAD messages. Each time a capture is initiated, statistics can be reset (e.g., cleared) or they can be accumulated with the previous capture. The following statistics are available:

- Total Capture Time – Total capture time in hours, minutes, and seconds
- Total Captured Messages – Total number of PAD messages captured
- Average Size – Average size of PAD message in bytes
- Average Rate – Average data rate at which PAD messages are received in bytes per second
- Average Arrival Time – Average arrival time between PAD messages in seconds
- Average Transmit Delay – Average PAD message delay from PAD transmission to reception. This option uses the auto track feature of the MPS PAD Generator. If auto track is not enabled, this statistic will not be available.

Statistics can be viewed by selecting *View→Statistics* menu or the statistics icon of the main interface. Statistic averages can be computed over the entire capture time or over the history window size by selecting the *Entire Capture* or *History Window* option from the Statistics window. The statistics window is shown in Figure 3-5.

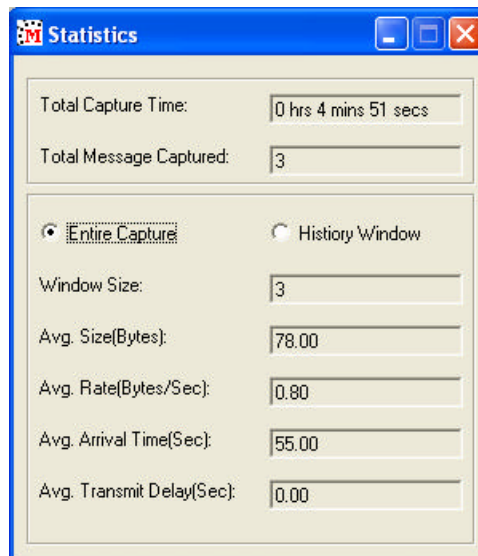


Figure 3-5 MPS PAD Monitor Statistics

For additional information on the MPS PAD Monitor, refer to the *Help* menu option in the MPS PAD Monitor.