BlackBerry Enterprise Server

Transcoder API

Version: 4.1 SP5 to 5.0 SP4

Development Guide

BlackBerry®

Published: 2012-09-28
SWD-20120928161246473

Contents

Overview

1

You can use the Transcoder API to implement a custom transcoding scheme to encode and decode the Gateway Message Envelope (GME) packets that are exchanged between BlackBerry devices and the BlackBerry Enterprise Server. By default, GME packets are encoded by the standard BlackBerry transport layer encryption. If you implement a transcoding scheme, GME packets are encoded by BlackBerry transport layer encryption and by the transcoder.

If you permit third-party applications on devices to use the transcoder, those applications, if not functioning correctly, might affect the security, usability, and performance of the BlackBerry Enterprise Server, and might cause the loss of device data. Verify the security and reliability of the applications that you permit to use the transcoder.

The Transcoder API is compatible with BlackBerry Enterprise Server 4.1 SP5 or later and BlackBerry Device Software 4.5 or later.

Encoding order

GME packets are encoded by BlackBerry transport layer encryption and by the transcoder implementation. GME packets can be encoded using one of the following options:

Encoding order	**BlackBerry Enterprise Server version required**	**BlackBerry Device Software version required**
GME packets are encoded first by the transcoder implementation, and then by BlackBerry transport layer encryption. This is the default encoding order.	4.1 SP5 or later	4.5 or later
GME packets are encoded first by BlackBerry transport layer encryption, and then by the transcoder implementation. Configuring your environment in this way can make it easier for you to verify the integrity of GME packets that use the transcoder.	5.0 SP4 or later	7.1.0.9 or later

The first option is the only option available in BlackBerry Enterprise Server 4.1 SP5 to 5.0 SP3, and BlackBerry Device Software earlier than 7.1.0.9 (starting with BlackBerry Device Software 4.5). If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and BlackBerry 7.1.0.9 or later, you can configure which encoding order you want to use. If the version requirements for the second method are not met, the first method is used by default.

Transcoder interfaces

2

Research In Motion provides a Java interface and a C++ header file. These files define the methods required to encode and decode messages. The BlackBerry device uses the Java interface; the BlackBerry Enterprise Server uses the C++ header file. The device and BlackBerry Enterprise Server pass all incoming and outgoing data messages to the transcoder implementation. Since two threads can call the encoding or decoding method at the same time, the implementation must be synchronized on every method call.

Device-side interface

The transcoder implementation on the BlackBerry device filters which messages to encode or decode using the willTrancode() method and ensures that a corresponding decoding method exists on the BlackBerry Enterprise Server before it encodes the message.

The transcoder implementation is responsible for any key management that the encoding scheme requires.

The device calculates a Cyclic Redundancy Check (CRC) of the message before the transcoder implementation encodes the message. The device uses the CRC to check that the corresponding decoding process is accurate and that the transmission is error free. The CRC is not used to prevent malicious transcoder implementations.

If encoding succeeds, the code provided in the Transcoder API prepends the identifier of the transcoder to the front of the encoded data. The standard BlackBerry decryption process checks this identifier before it passes the data to the decoder.

The transcoder implementation is responsible for providing useful exception handling and logging. If the transcoder implementation throws an exception, the message is dropped and a failure is logged. The logs display a Globally unique identifier (GUID) to distinguish them from other log entries.

Device-side Transcoder methods to implement

You must implement the following methods for the device-side Transcoder implementation:

Method	Description
Transcoder() constructor	This constructor creates an instance of a transcoder.
getID()	This method returns a byte that uniquely identifies this instance of the transcoder.

Method	Description
willTranscode()	This method queries the transcoder with a context to see if the transcoder will perform an operation in the supplied context. This method filters calls to the encode and decode methods.
encode()	This method encodes the data in an input stream and outputs the encoded data into an output stream.
decode()	This method decodes the data in an input stream and outputs the decoded data into an output stream.

For more information, see Class: Device-side Transcoder.

Server-side interface

The transcoder implementation on the BlackBerry Enterprise Server filters the messages to encode and decode, throws exceptions, logs any encoding and decoding errors, and provides any key management required as part of the encoding scheme.

Set up the implementation file

1. Visit http://www.blackberry.com/go/transcoderapiheaderfile to download the BESTranscoderAPI.h file.
2. In the main file of the server-side transcoder implementation, include the following line: `#include "BESTranscoderAPI.h"`.
3. To use the helper macro provided by the Transcoder API, include the following line: `DEFINE_BES_TRANSCODER_DLL`. This macro ensures that the required function definitions are present.

Server-side Transcoder methods to implement

You must implement the following methods for the server-side transcoder implementation:

Method	Description
LoadDLL()	This function is called once when the BlackBerry Enterprise Server loads the DLL.
FreeDLL()	This function is called once when the BlackBerry Enterprise Server unloads the DLL.
GetID()	This function returns a byte that uniquely identifies this instance of the transcoder.

Method	Description
WillTranscode()	This function queries the transcoder with a context to see if the transcoder will perform an operation in the context. This function filters calls to the Encode and Decode methods.
Encode()	This function encodes the data in an input stream and outputs the encoded data into an output stream.
Decode()	This function decodes the data in an input stream and outputs the decoded data into an output stream.

For more information, see Class: BlackBerry Enterprise Server-side Transcoder.

Transcoding data on the BlackBerry device

3

Registering the transcoder implementation on the BlackBerry device

Before the transcoder implementation can encode or decode messages, it must register with the TranscoderManager on the BlackBerry device by calling the TranscoderManager register() method. The TranscoderManager is available to signed clients.

For devices running software earlier than BlackBerry 7.1.0.9, the TranscoderManager can recognize only one transcoder implementation. If you attempt to register two or more transcoder implementations, the device registers the first implementation only.

For devices running BlackBerry 7.1.0.9 or later, the TranscoderManager can recognize more than one transcoder implementation. You configure the Primary Transcoder IT policy rule in the IT policy that is assigned to user accounts to specify the primary transcoder implementation. The primary transcoder is used to encode the GME packets that devices send and to decode the packets that devices recieve, while secondary transcoders can only decode GME packets that devices receive.

If a Transcoder implementation fails to register with the TranscoderManager, the TranscoderManager throws a TranscoderRegistrationException.

For more information on the TranscoderManager register() method, see Method: register().

Process flow: Encoding data on the BlackBerry device

1. The BlackBerry device prepares a message to send.
2. If a registered transcoder implementation exists on the device, the device calculates a CRC of the datagram and uses the transcoder implementation to encode both the CRC and the datagram.

3. The trancoder implementation prepends the Transcoder ID to the encoded data.
4. The device processes the encoded data using the standard BlackBerry encryption and compression process.

This process flow uses the default encoding order. If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and devices running BlackBerry 7.1.0.9 or later, you can configure the environment to encode data using BlackBerry transport layer encryption first, and then using the transcoder implementation.

Process flow: Decoding data on the BlackBerry device

1. The BlackBerry device receives the datagram and decrypts and decompresses the data using the standard BlackBerry decryption and decompression process.
2. If the Transcoder ID prepended to the data matches the Transcoder implementation registered on the device, the transcoder implementation decodes the data.
3. The device calculates the CRC of the decoded datagram. If the CRC matches the CRC appended to the decoded datagram, the transcoding succeeds.

This process flow uses the default encoding and decoding order. If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and devices running BlackBerry 7.1.0.9 or later, you can configure the environment to encode data using BlackBerry transport layer encryption first, and then using the transcoder implementation. If you use this encoding order, when devices receive data, the data is decoded using the transcoder implementation first, and then using BlackBerry transport layer encryption.

Transcoding data on a BlackBerry Enterprise Server

4

Registering the transcoder implementation on the BlackBerry Enterprise Server

On the BlackBerry Enterprise Server, the Transcoder implementation is a DLL. The location of the Transcoder DLL is specified by the following registry key:

- 32-bit operating system: HKEY_LOCAL_MACHINE\SOFTWARE\Research In Motion\BlackBerry Enterprise Server \Dispatcher\Transcoder
- 64-bit operating system: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Research In Motion\BlackBerry Enterprise Server\Dispatcher\Transcoder

A string value named *Transcoder* holds the full path and file name of the DLL. Only one transcoder implementation can be registered on the BlackBerry Enterprise Server.

On the BlackBerry Enterprise Server, the BlackBerry Dispatcher service is designed to handle BlackBerry transport layer encryption. On startup of the BlackBerry Enterprise Server, the BlackBerry Dispatcher service loads the Transcoder DLL specified by the registry key. If a transcoder implementation is specified, the BlackBerry Dispatcher sets a flag that alerts the BlackBerry encryption process to send data to the transcoder implementation. The BlackBerry Dispatcher must be restarted to load the transcoder implementation; it does not support dynamic loading and unloading of transcoder implementations.

It is a best practice to restart the BlackBerry Dispatcher during low usage periods, as the restart temporarily impacts the flow of data to and from devices.

Process flow: Encoding data on the BlackBerry Enterprise Server

1. The BlackBerry Enterprise Server prepares a datagram to send.

2. If a registered trancoder implementation exists on the BlackBerry Enterprise Server, the BlackBerry Enterprise Server calculates a CRC of the datagram and uses the transcoder implementation to encode both the CRC and the datagram.
3. The transcoder implementation prepends the Transcoder ID to the encoded data.
4. The BlackBerry Enterprise Server processes the encoded data using the standard BlackBerry encryption and compression process.

This process flow uses the default encoding order. If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and devices running BlackBerry 7.1.0.9 or later, you can configure the environment to encode data using BlackBerry transport layer encryption first, and then using the transcoder implementation.

Process flow: Decoding data on the BlackBerry Enterprise Server

1. The BlackBerry Enterprise Server receives a datagram and decrypts and decompresses the data using the standard BlackBerry decryption and decompression process.
2. If the Transcoder ID prepended to the data matches the transcoder implementation registered on the BlackBerry Enterprise Server, the transcoder decodes the data.
3. The BlackBerry Enterprise Server calculates the CRC of the decoded datagram. If the CRC matches the CRC appended to the decoded datagram, the transcoding succeeds.

This process flow uses the default encoding and decoding order. If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and devices running BlackBerry 7.1.0.9 or later, you can configure the environment to encode data using BlackBerry transport layer encryption first, and then using the transcoder implementation. If you use this encoding order, when the BlackBerry Enterprise Server receives data, the data is decoded using the transcoder implementation first, and then using BlackBerry transport layer encryption.

Security on the BlackBerry Enterprise Server

5

A transcoder implementation can access the Transcoder API if it has been digitally signed by the RIM signing authority system. The RIM signing authority system uses the RIM Cryptographic API key to authorize and authenticate transcoder implementation code. You must use the BlackBerry Signing Authority Tool to sign the .cod files for your transcoder implementation.

To read the documentation for the BlackBerry Signing Authority Tool, and to download the tool, visit BlackBerry Java 7.1 SDK - BlackBerry Signing Authority tool.

For a transcoder implementation to register successfully with the TranscoderManager on BlackBerry devices, you must configure the following rules in the IT policy that is assigned to user accounts:

- The Primary Transcoder IT policy rule: Specifies the primary transcoder and the encoding order that you want to use. This IT policy rule is required only for BlackBerry Enterprise Server 5.0 SP4 or later, and devices running BlackBerry 7.1.0.9 or later. See Configuring the primary transcoder and the encoding order.
- The Security Transcoder Cod File Hashes IT policy rule: Specifies the transcoder implementations that are permitted to register with the TranscoderManager on devices. This IT policy rule prevents unauthorized third-party encoding schemes from registering with the TranscoderManager on devices. See Permit devices to register the transcoder implementation.

For more information about IT policy rules, and configuring and assigning IT policies, visit www.blackberry.com/go/serverdocs to read the *BlackBerry Enterprise Server Policy Reference Guide* and the *BlackBerry Enterprise Server Administration Guide*.

Configuring the primary transcoder and the encoding order

If your organization's environment uses BlackBerry Enterprise Server 5.0 SP4 or later, and users have BlackBerry devices running BlackBerry 7.1.0.9 or later, you must configure the Primary Transcoder IT policy rule in the IT policy that is assigned to user accounts. This rule specifies the following settings:

- The primary transcoder implementation: Multiple transcoder implementations can be registered on devices; you must specify the primary transcoder that you want to use to encode the GME packets that devices send and to decode the

packets that devices receive. Other transcoders are considered secondary, and can only be used to decode GME packets that devices receive.
- The encoding order: Whether GME packets are encoded first by the transcoder implementation and then by BlackBerry transport layer encryption (the default behavior), or first by BlackBerry transport layer encryption and then by the transcoder implementation.

Devices that run BlackBerry 7.1.0.9 or later cannot use the transcoder implementation unless you configure the Primary Transcoder IT policy rule and assign the IT policy to the appropriate user accounts.

If your organization's environment uses earlier versions of the BlackBerry Enterprise Server or BlackBerry Device Software, you do not need to specify this information. In earlier versions, you do not need to specify a primary transcoder (only one transcoder can be registered on devices), and the default encoding order is the only option.

Configure the primary transcoder and the encoding order

Before you begin:
- Ask your organization's BlackBerry Administration Service administrator to perform this task, or ask for access to an administrator account that you can use to perform this task.
- Determine whether you will modify an IT policy that is already assigned to user accounts, or whether you will create a new IT policy to assign to user accounts.

1. In the BlackBerry Administration Service, on the **BlackBerry solution management** menu, expland **Policy**.
2. Click **Manage IT policies**.
3. Click the IT policy that you want to change.
4. Click **Edit IT policy**.
5. On the **Security** tab, in the **Primary Transcoder** field, specify the primary transcoder and the encoding order using one of the following:
    - If you want GME packets to be encoded first by the transcoder implementation and then by BlackBerry transport layer encryption, type ***<hash>* inside *<delay>***, where *<hash>* is the primary transcoder application's eldest sibling module, and *<delay>* is the number of minutes before non-transcoded packets are dropped after transcoding starts. *<delay>* is optional. If you do not specify the delay, the default value of 15 minutes is used.

      Example: 6b856777430b09f906fecad4d156da52ec2b6033 inside 20

    - If you want GME packets to be encoded first by BlackBerry transport layer encryption and then by the transcoder implementation, type ***<hash>* outside *<delay>***, where *<hash>* is the primary transcoder application's eldest sibling module, and *<delay>* is the number of minutes before non-transcoded packets are dropped after transcoding starts. *<delay>* is optional. If you do not specify the delay, the default value of 15 minutes is used.

      Example: 6b856777430b09f906fecad4d156da52ec2b6033 outside 20

6. Click **Save all**.

After you finish: Permit devices to register the transcoder implementation.

Permit devices to register the transcoder implementation

To permit BlackBerry devices to register the transcoder implementation, you must configure the Security Transcoder Cod File Hashes IT policy rule in the IT policy that is assigned to user accounts. This IT policy rule identifies a transcoder implementation as valid and permitted for use on devices.

Before you begin:

- Ask your organization's BlackBerry Administration Service administrator to perform this task, or ask for access to an administrator account that you can use to perform this task.
- Determine whether you will modify an IT policy that is already assigned to user accounts, or whether you will create a new IT policy to assign to user accounts.
- If your organization's environment uses BlackBerry Enterprise Server 5.0.4 or later, and devices that run BlackBerry 7.1.0.9 or later, see Configuring the primary transcoder and the encoding order.
- Retrieve the hash or hashes of the transcoder implementation with the command **javaloader siblinginfo** *<implementation_file.cod>* .

1. In the BlackBerry Administration Service, on the **BlackBerry solution management** menu, expland **Policy**.
2. Click **Manage IT policies**.
3. Click the IT policy that you want to change.
4. Click **Edit IT policy**.
5. On the **Security** tab, in the **Security Transcoder Cod File Hashes** field, specify the hash or hashes of the .cod files for the transcoder implementation. Specify each hash in hexadecimal, delimited by semi-colons. For example, b02845188bd3e2b7d60307ba721505c92600e7a5;6b856777430b09f906fecad4d156da52ec2b603.
6. Click **Save all**.

After you finish: If you created a new IT policy, assign the IT policy to the appropriate user accounts.

Transcoder API Reference

Class: Device-side Transcoder

Constant: CONTEXT_SERVICE_RECORD_UID

This constant is the hash table key for retrieving an array of service record UID string(s) from the context object. The retrieved object will never be null.

When a message is sent from the device, this array can contain 0 or more items. Each item in this array will correspond to a service record in the service book. The service record(s) for the transmission can be retrieved using the UID string(s) in this array.

When a message is sent to the device, this array can contain 0 or 1 item(s). There will be 1 item only if the transmission contains a bounded service record.

```
public static final int CONTEXT_SERVICE_RECORD_UID = 0;
```

Constant: CONTEXT_GME_ADDRESS_STRING

This constant is the hash table key for retrieving the GMEAddress string from the context. The string will be formatted according to: [CID]([/UID[(REDIRECT)][:PIN[(REDIRECT)]]])*[$SRC[(REDIRECT)]]

The retrieved object will never be null.

```
public static final int CONTEXT_GME_ADDRESS_STRING = 1;
```

Constructor: Transcoder() Constructor

This constructor creates an instance of a Transcoder.

```
public Transcoder(byte id)
```

Parameter

id	The unique id of this Transcoder instance. Cannot be 0.

Returns

This constructor returns a new Transcoder instance identified by id.

Method: getID()

```
public final byte getID()
```

Parameters

None.

Returns

This method returns a byte that uniquely identifies this Transcoder.

Method: willTranscode()

This method queries the Transcoder with a context to see if the Transcoder will perform an operation in the supplied context.

This method is used to filter calls to encode and decode.

```
public boolean willTranscode(IntHashtable context)
```

Parameter

context	The context to be checked. The keys, `CONTEXT_SERVICE_RECORD_UID` and `CONTEXT_GME_ADDRESS_STRING`, used to obtain the context-sensitive information in this hash table, are described in Class: Device-side Transcoder.

Returns

This method returns true if transcoding will be performed in the supplied context; false otherwise.

Method: encode()

This method encodes the data in the input stream and outputs the encoded data into the output stream.

This method should check whether the recipient of the message has a corresponding decoder implemented before encoding the message.

```
public boolean encode(
                InputStream input,
                OutputStream output,
                IntHashtable context)
```

Parameters

input	The message to encode.
output	The output message is stored in this output stream.
context	The context of the message. The list of keys, `CONTEXT_SERVICE_RECORD_UID` and `CONTEXT_GME_ADDRESS_STRING`, used to obtain the context-sensitive information in this hash table are described in Class: Device-side Transcoder.

Returns

This message returns true if encoding was successful; false otherwise.

Method: decode()

This method decodes the data in the input stream and outputs the decoded data into the output stream.

This method should check whether the recipient of the message has a corresponding encoder implemented before decoding the message.

```
public boolean decode(
                InputStream input,
                OutputStream output,
                IntHashtable context)
```

Parameters

input	The message to decode.
output	The output message is stored in this output stream.
context	The context of the message. The list of keys, `CONTEXT_SERVICE_RECORD_UID` and `CONTEXT_GME_ADDRESS_STRING`, used to obtain the context-sensitive information in this hash table are described in Class: Device-side Transcoder.

Class: TranscoderManager

Method: register()

This method registers a Transcoder implementation. A Transcoder implementation must be registered using this method before it will be used to encode or decode any messages.

After a Transcoder implementation is registered, any messages sent from the device are passed into the Transcoder implementation's encode() method for encoding. Similarly, any messages sent to the device are passed into the Transcoder implementation's decode() method for decoding.

```
public static void register(Transcoder transcoder) throws
TranscoderRegistrationException
```

Parameter

transcoder	The Transcoder to register.

Exceptions

IllegalArgumentException	Thrown if Transcoder is null or the Transcoder's getID() method returns 0.
TranscoderRegistrationException	Thrown if registration cannot be completed. Registration will fail if one of the following conditions exists: • The Transcoder implementation to register is null. • Another Transcoder implementation is already registered. • The calling application does not have permission to register a Transcoder implementation. The permission to register a Transcoder implementation is granted by the BlackBerry Enterprise Server administrator.

Returns

None.

Class: BlackBerry Enterprise Server-side Transcoder

Function: LoadDLL()

This function is called once when the DLL is loaded by the BlackBerry Enterprise Server.

```
int __cdecl LoadDLL()
```

Parameters

None.

Returns

This function returns 0 if loading was successful or a non-zero integer representing an error code.

Function: FreeDLL()

This function is called once when the DLL is unloaded by the BlackBerry Enterprise Server.

```
void FreeDLL()
```

Parameters

None.

Returns

None.

Function: GetID()

```
unsigned char __cdecl GetId()
```

Parameters

None.

Returns

This function returns an unsigned char identifier for the Transcoder. The indentifier cannot be 0.

Function: WillTranscode()

This function queries the Transcoder to determine if transcoding will be performed in the supplied context.

```
int __cdecl WillTranscode(const TranscoderContext *const context)
```

Parameter

context	The context of the message.

Returns

This method returns 0 if encoding is to occur, or a non-zero integer otherwise. TRANSCODE_ERROR (-1) is reserved.

Function: Encode()

This function encodes messages sent from the BlackBerry Enterprise Server if this Transcoder has been registered with the BlackBerry Enterprise Server. The Transcoder implementation should check if a corresponding decoder has been implemented on the receiver before it encodes the message.

```
int __cdecl Encode(
                TranscoderInputStream *const input,
                TranscoderOutputStream *const output,
                const TranscoderContext *const context)
```

Parameters

input	The message to encode.
output	The output message is stored in this output stream.
context	The context of the message.

Returns

This method returns 0 if the encoding is successful, and a non-zero integer otherwise. If a non-zero integer is returned, the message is dropped. Note that TRANSCODE_ERROR (-1) is reserved.

Function: Decode()

This function decodes messages sent to the BlackBerry Enterprise Server if this Transcoder implementation has been registered with the BlackBerry Enterprise Server.

The Transcoder implementation should check if the message was encoded with a corresponding encoder before it decodes the message.

```
int __cdecl Decode(
                    TranscoderInputStream *const input,
                    TranscoderOutputStream *const output,
                    const TranscoderContext *const context)
```

Parameters

input	The message to decode.
output	The output message is stored in this output stream.
context	The context of the message.

Returns

This function returns 0 if the encoding was successful, and a non-zero integer otherwise. If a non-zero integer is returned, the message will be dropped. Note that TRANSCODE_ERROR (-1) is reserved.

Class: TranscoderInputStream

Function: Read(unsigned char *, long)

This function reads data from the stream consuming the data.

```
virtual long Read(unsigned char * pData, long Length) = 0;
```

Parameters

pData	The buffer to fill.
Length	The number of bytes to read.

Returns

This method returns the number of bytes read. This may be less than `Length` if the end of the stream is reached.

Function: Read(char)

This function reads data from the stream.

```
virtual bool Read(char *cp) = 0;
```

Parameter

cp	The character read.

Returns

This function returns true if a character was read; false otherwise.

Function: Read(unsigned char)

This function reads data from the stream.

```
virtual bool Read(unsigned char *cp) = 0;
```

Parameter

cp	The byte read.

Returns

This function returns true if a byte was read; false otherwise.

Function: Peek()

This function reads data from the stream, without altering the read location.

```
virtual long Peek(unsigned char * pData, long Length) = 0;
```

Parameters

pData	The buffer to fill.
Length	The number of bytes to read.

Returns

This function returns the number of bytes read. This may be less than `Length` if the end of the stream is reached.

Function: GetLength()

This function retrieves the length of the stream.

```
virtual long GetLength() const = 0;
```

Returns

This function returns the number of bytes left to be consumed in the stream.

Class: TranscoderOutputStream

Function: Write(unsigned char const *, long)

This function writes data to the stream.

```
virtual long Write(unsigned char const * pData, long Length) = 0;
```

Parameters

pData	The buffer to write.
Length	The number of bytes to write.

Returns

This function returns the number of bytes written.

Function: Write(char)

This function writes data to the stream.

```
virtual bool Write(char ch)
```

Parameter

ch	The char to write.

Returns

This function returns true if the char was written; false otherwise.

Function: Write(unsigned char)

This function writes data to the stream.

```
virtual bool Write(unsigned char ch)
```

Parameter

ch	The byte to write.

Returns

This function returns true if the byte was written; false otherwise.

Struct: TranscoderContext

The TranscoderContext Structure provides parameters for the Encode and Decode functions.

```
    struct TranscoderContext
                {
                /** The version number of the TranscoderContext struct */
                const int version;
                /** The BES user id */
                int nUserId;
                /** The email address of the target */
                const char * szEmail;
                /** The PIN of the target */
                const char * szDeviceId;
                /** The GME content type */
                const char * szContentId;
                };
```

Transcoder error codes

7

BlackBerry device transcoder error codes

Exceptions generated by the transcoder implementation on the BlackBerry device are added to the event log on the device.

Error message	Description
RegF	Transcoder registration failed.
EnFR	Transcoder encoding failed and false is returned by the encode method.
EnTC	Transcoder encoding failed and the exception thrown by the encode method is caught.
DeFR	Transcoder decoding failed and false is returned by the decode method.
DeTC	Transcoder decoding failed and the exception thrown by the decode method is caught.
DeVE	Transcoder decoding failed because the version of the Transcoder is unsupported.
DeRC	Transcoder decoding failed because the redundancy check failed.
NDel	Transocder decoding failed because the Transcoder ID is invalid.

BlackBerry Enterprise Server transcoder error messages

Exceptions generated by the transcoder implementation on the BlackBerry Enterprise Server are added to the BlackBerry Dispatcher (DISP) log.

The following are examples of error strings that might appear in the DISP logs:

- Unable to load the specified transcoder DLL
- Unable to load a required function in the transcoder DLL

- Transcoding Failed: Transcoder ID id encountered error error during encode
- Transcoding Failed: Transcoder ID id encountered error error during decode
- Transcoding Failed: Invalid Transcoder ID
- Transcoding Failed: Expected Transcoder ID id, Received Transcoder ID id
- Transcoding Failed: Could not read CRC
- Transcoding Failed: CRC Mismatch
- Transcoding Failed: Could not write CRC
- Transcoding Failed: Invalid data format when reading TLE
- Transcoding Failed: Invalid data format when writing TLE
- Transcoding Failed: Transcoder ID id threw uncaught exception during encode
- Transcoding Failed: Transcoder ID id threw uncaught exception during decode

Glossary

8

CRC

Cyclic Redundancy Check

GME

Gateway Messaging Envelope

GUID

Globally unique identifier

UID

Unique identifier

Legal notice