Omni-ID, Inc

Software Requirements Specification

**.NET Omni middleware services API**

Release 1.0

Document Revision: 1.06

Date: October 31, 2013

Omni-ID, Inc.

1200 Ridgeway Avenue

Rochester, NY

[www.Omni-ID.com](http://www.Omni-ID.com)

Author: Omni-ID Software Development Team

Contact: John Steeves

John.Steeves@omni-id.com

**

Table of Contents

[1 Introduction 4](#_Toc363128748)

[2 API 5](#_Toc363128749)

[2.1 Establishing the Client API Connection 7](#_Toc363128750)

[2.1.1 C# Example - creating OmniMiddlewareClient class ; supplying event handler 7](#_Toc363128751)

[2.2 Construction and Transmission of Commands 9](#_Toc363128752)

[2.2.1 Transaction ID fields for Commands that Return Result Messages 9](#_Toc363128753)

[2.3 Receiving and Processing Asynchronous Messages 9](#_Toc363128754)

[2.3.1 Event Handler Method 10](#_Toc363128755)

[2.4 Example Command Execution Sequence 10](#_Toc363128756)

[2.5 Application Configuration File **Error! Bookmark not defined.**](#_Toc363128757)

[3 API Command and Event Message Listing 11](#_Toc363128758)

[3.1 Commands 11](#_Toc363128759)

[3.1.1 OmniSystemMetadataInfoRequestCommand 11](#_Toc363128760)

[3.1.2 OmniVisualTagInfoRequestCommand 11](#_Toc363128761)

[3.1.3 OmniGPIStateReportCommand 12](#_Toc363128762)

[3.1.4 OmniGPOStateReportCommand 12](#_Toc363128763)

[3.1.5 OmniGPOStateChangeCommand 13](#_Toc363128764)

[3.1.6 OmniRFIDReaderCommand 13](#_Toc363128765)

[3.1.7 OmniImageUpdateCommand 13](#_Toc363128766)

[3.1.8 OmniImageUrlUpdateCommand 14](#_Toc363128767)

[3.1.9 OmniPageChangeCommand 15](#_Toc363128768)

[3.1.10 OmniPageDeleteCommand 16](#_Toc363128769)

[3.2 Event Messages 17](#_Toc363128770)

[3.2.1 OmniRFIDDetectionMessage 17](#_Toc363128771)

[3.2.2 OmniVisualTagAnnounceEvent 17](#_Toc363128772)

[3.2.3 OmniVisualTagHealthReportMessage 18](#_Toc363128773)

[3.2.4 OmniGPIEventMessage 18](#_Toc363128774)

[3.2.5 OmniCommandErrorResultEvent 18](#_Toc363128775)

[3.2.6 OmniSystemErrorEvent 19](#_Toc363128776)

[4 Class diagrams 20](#_Toc363128777)

[4.1 Command Classes 20](#_Toc363128778)

[4.2 Message Classes 21](#_Toc363128779)

[5 Configuration 22](#_Toc363128780)

[6 Revision History 24](#_Toc363128781)

# Introduction

This document presents the .NET-specific Application Programming Interface (API) for integrating Omni-ID Fixed RFID Reader middleware and Visual Tag middleware into a third party system solution. The API shall be distributed as a .NET version 4.0 Class Library Dynamic Link Library (DLL) file. The API DLL file is one part of an overall Omni-ID middleware and hardware solution.

The API DLL establishes a communication channel with an Omni-ID server application that is responsible for executing commands submitted through the API as well as sending asynchronous messages back to the API DLL for instance to communicate the results of previously submitted commands and to communicate system-initiated messages such as an RFID detection, low battery warning, loss of communication with a fixed reader, and so on.

# System Requirements

The Omni-ID Middleware has the following requirements:

1. Windows Server 2008 R2 x64

–or–

1. Windows 7 x64
2. Microsoft SQL Server 2008 R2 or later

–or–

1. Microsoft SQL Server 2008 Express R2

Note:

Other Windows operating systems should function but are unsupported at this time.

# System Block Diagram

The Omni-ID Middleware software is contains the following software components:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Software Component** | **Description** |
| Middleware .NET API Interface | Middleware.dll | The software interface described in this document. |
| Active Tag Manager | Omni CALCManager service | Provides integration with the Link Gateway to control the ProVIEW tags. |
| Impinj Reader App | OmniImpinjReaderApp | Provides configuration and integration with the RFID readers. |
| Image Generation Service | Omni Image Generator service | Service that renders images for the ProVIEW tags. |
| Log Server | OmniLogServer | Provides unified logging for Middlware. |
| Database Configuration utility | OmniImportConnectionString | Configuration utility to setup database. |
| Middleware database | CALCMan database | Database to store configuration and visual tag states and history. |



Figure - Middleware Block Diagram

# API

The Application Programming Interface provided by the DLL consists of a single method for the API client to send commands to the Omni-ID middleware, and a .NET event object with a specific method signature to communicate asynchronously generated events from the Omni-ID middleware to the API client.

The **PostOmniAPICommand** API method is supplied for the API client to construct and send commands to the Omni-ID middleware service.

The **OmniIDMiddlewareEvent** .NET event allows the .NET API client will receive asynchronously generated events from the Omni-ID middleware service.

## Establishing the Client API Connection

The .NET client code establishes a connection with the Omni-ID middleware service by creating an instance of the OmniMiddlewareClient class, feeding to its constructor the IP address and port number of the CALC Manager Service, the Image Gen Service and Impinj Reader Service.

The .NET client code then ‘wires’ an event handler function to the **OmniIDMiddlewareEvent** .NET event to be able to receive asynchronously generated events.

### OmniMiddlewareClient Contructors

There are currently two OmniMiddleware constructors, a default constructor which will reads the CALCManager service, Image Gen service, and Impinj Reader service IP address/port numbers from the applications app.config file. A second constructor, takes these IP address/port numbers as parameters.

**OmniMiddlewareClient()**

Creates a new OmniMiddlewareClient which will interact with the CALCMan, ImageGen, and Revolution services. The host names of CALCMan, ImageGen, and OmniImpinjReader services will be read for the calling applications application configuration file (app.config). The ClientGUID will also be read from the application configuration file. See other constructor for a description of the ClientGuid.

**Syntax**

public OmniMiddlewareClient()

**Remarks**

Example app.config appSettings are as follows:

key="CALCManServiceHostName" value="local-host:3000"

key="ImageGenServiceHostName" value="local-host:30525"

key="RFIDServiceHostName" value="local-host:3300"

key="MiddlewareClientGuid" value= "F33B072B647448d2BA48230903A2C565"

**OmniMiddlewareClient(String, String, String, String)**

Creates a new OmniMiddlewareClient which will interact with the CALCMan, ImageGen, and OmniImpinjReader services at the given hostnames. These hostnames SHOULD NOT include "http://" and SHOULD include the port number on which the service is running (i.e. "local-host:3000")

**Syntax**

public OmniMiddlewareClient(string calcIp, string imageIp, string rfidIp, string clientGuid)

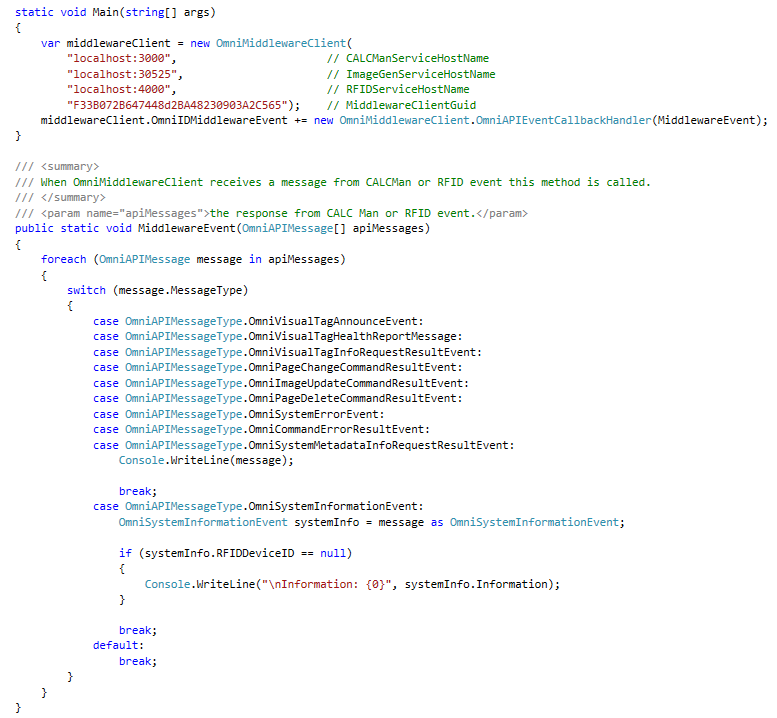
**Parameters**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| *calcIp* | String | Hostname (IP and port) of the CALCMan service. The IP address of the machine running the CALC Manager service and the port is the "CALCManServiceHostName" value in the CALCManService.exe.config file |
| *imageIp* | String | Hostname (IP and port) of the ImageGen service. The IP address of the machine running the ImageGen service and the port is the "ImageGenServiceHostName" value in the CALCManService.exe.config file |
| *rfidIp* | String | Hostname (IP and port) of the OmniImpinj Reader application. The IP address of the machine running the OmniImpinj Reader application, and a default port of 3300 |
| *clientGuid* | String | A unique string that is used to identify the client instance. A client should use the same guid each time, and will be used to create a message queue. GUID strings like "F33B072B647448d2BA48230903A2C565" are recommended. This string should appear in the QueueNames.txt file as well as the CALCMan database for the Impinj reader. It is anticipated that this parameter will be removed in the near future. |

**Remarks**

Example: var middleware = new OmniMiddlewareClient("localhost:3000", "localhost:30525", "localhost:3300", "F33B072B647448d2BA48230903A2C565");

### C# Example - creating OmniMiddlewareClient class ; supplying event handler



## Construction and Transmission of Commands

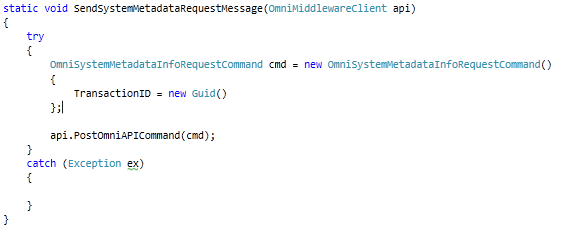
The Omni .NET Middleware DLL interface allows the .NET client to send commands to the Omni Middleware Service by constructing an instance of a class derived from the base command class, **OmniAPICommand** , populating the derived class’s data fields, and sending the command class instance to the Omni Middleware Service using the **PostOmniAPICommand** method of the **OmniMiddlewareClient** API class.

### Transaction ID fields for Commands that Return Result Messages

All classes that derive from **OmniAPICommand** and when executed by the Omni Middleware Service can result in the transmission of an asynchronous response message include a required **TransactionID** field of the .NET type System.Guid. This **TransactionID** field value is used as a ‘correlation ID’ that allows the API client code to match up an asynchronously received message generated by the Omni Middleware Service with the original command message that was sent sometime in the past.

The API client is responsible for setting the value of **TransactionID** field as part of the process of constructing a Command message prior to sending the command message to the Omni Middleware Client.

#### C# Example of Command Construction and Transmission



## Receiving and Processing Asynchronous Messages

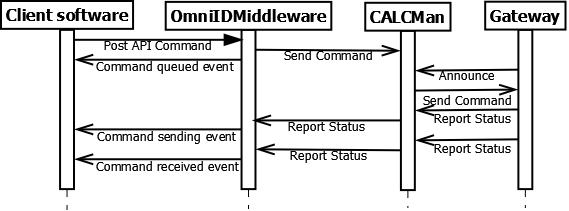
A .NET API client registers to receive asynchronous messages generated by the Omni Middleware Service as shown above in section 2.1.1 by adding an event handler function to the OmniIDMiddlewareEvent .NET event of the OmniMiddlewareClient class.

### Event Handler Method

The event handler method supplied by the .NET API client will receive an array containing one or more instances of classes derived from the OmniAPIMessage class. The event handler should iterate through each OmniAPIMessage class instance provided in the array, and process each message instance as necessary. Each message is processed by determining its type using the **OmniAPIMessageType** enumeration field, and reading the values of the fields of a specific message instance as needed to implement the API client business logic. See Section 2.1.1 for a C# example of this processing.

## Example Command Execution Sequence

The Middleware API is fundamentally asynchronous in nature: once a Client has posted a command to the Middleware and the command is queued, the command is executed and its status updated asynchronously. An example command sequence is shown below:



# API Command and Event Message Listing

This section lists each of the OmniAPICommand types and OmniAPIMessage types available in the API. New command and message types may be added in the future to support new features and functionality.

## Commands

### OmniSystemMetadataInfoRequestCommand

#### Description

This command is issued to retrieve an XML document from the Omni Middleware Service containing a detailed description of system metadata, including information such as what Visual Tag models are supported, and for each Visual Tag model, what Image Templates are available, and for each Image Template what fields along with field type are available, and so on.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.

#### Result Message

1. TransactionID – GUID – same as above
2. SystemMetadataXMLDocument: string – contains the system metadata

### OmniVisualTagInfoRequestCommand

#### Description

This command allows the API client to request information about a specific visual tag by supplying its ID.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.
2. VisualTagUID – string – the unique identifier for the visual tag.

#### Result Message

1. TransactionID – GUID – same as above
2. VisualTagUID - string – the UID of the visual tag
3. VisualTagModel – string – contains the model identifier of the tag such as P4
4. BatteryLevel – int – contains the battery level of the tag
5. CurrentDisplayPage – int – zero-based index indicating which page in memory is currently displayed.

### OmniGPIStateReportCommand

Note: This Command is not yet supported

#### Description

This command is used to request the current state of a General-Purpose Input (GPI) port attached to a specific RFID Reader.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.
2. RFIDDeviceID – string – ID of the fixed RFID reader device.
3. GPIPortNumber – int – the number of the GPI port to report

#### Result Message

1. TransactionID – GUID – same as above
2. RFIDDeviceID – string – same as above
3. GPIPortNumber – int – same as above
4. PortState – GPIOPortState – GPIOPortState.High if port state has changed to high, or GPIOPortState.Low if port state has changed to low.

### OmniGPOStateReportCommand

Note: This Command is not yet supported

#### Description

This command is used to request the current state of a General Purpose Output (GPO) port attached to a specific RFID Reader.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.
2. RFIDDeviceID – string – ID of the fixed RFID reader device.
3. GPOPortNumber – int – the number of the GPO port to report

#### Result Message

1. TransactionID – GUID – same as above
2. RFIDDeviceID – string – same as above
3. GPOPortNumber – int – same as above
4. PortState – GPIOPortState – GPIOPortState.High if port state has changed to high, or GPIOPortState.Low if port state has changed to low.

### OmniGPOStateChangeCommand

Note: This Command is not yet supported

#### Description

This command is used to set the state of a specific GPO port on a specific Fixed RFID Reader.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.
2. RFIDDeviceID – string – ID of the fixed RFID reader device.
3. GPOPortNumber – int – the number of the GPO port to report
4. PortState – GPIOPortState – GPIOPortState.High if port state should be changed to high, or GPIOPortState.Low if port should be changed to low.

#### Result Message

1. TransactionID – GUID – same as above
2. RFIDDeviceID – string – same as above
3. GPOPortNumber – int – same as above
4. Success – bool – True if the state change was successful, false if it failed in some way.

### OmniRFIDReaderCommand

Note: This Command is not yet supported

#### Description

This command is used to start or stop a specific Fixed RFID Reader.

#### Fields

1. TransactionID – GUID – the client-supplied GUID message correlation ID.
2. RFIDDeviceID – string – ID of the fixed RFID reader device.
3. RequestedReaderState – ReaderState – ReaderState.StartReading if the reader should be made to start reading, or ReaderState.StopReading if the reader should stop reading.

#### Result Message

1. TransactionID – GUID – same as above
2. RFIDDeviceID – string – same as above
3. Success – bool – true if the state change was successful, false if it failed in some way.

### OmniImageUpdateCommand

#### Description

This command is used to send the image template parameters necessary to update the image of a specific visual tag.

#### Fields

1. TransactionID – GUID – the command / response correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag
3. ImageTemplateID – string – the identifier of the image generation template to use to generate the image.
4. PageNumber – int – the 1-based index of the page to store the image on the target visual tag.
5. CoralType – CoralTypes – The type of visual tag.
6. ImageTemplateParameters - Dictionary<string,string> - a string dictionary containing the name and value of each image template parameter for the specified image template.
7. GatewayID – string – the unique identifier of the target Gateway
8. IsTransactional – bool – True means this command is part of a group of commands for the specified visual tag. If this command fails then all other commands that are queued for the specified visual tag that have not yet been executed should be cancelled. A group is defined as any other commands for the specified visual tag that are queued at the time this command fails and these command are marked as transactional commands. For example let's suppose an image update and page change command are queued to CORAL 123. The commands are marked as transactional. The image update command fails after all retries. The page change command will be deleted because it is marked as part of the same transaction as the image update command. It does not make sense flip the page if the image download did not succeed.   
   False means if the command fails it has no effect on other commands. False is the default.

#### Result Message

1. TransactionID – GUID – message correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag.
3. CommandQueued – bool – if True, indicates the Omni Middleware Service has received and queued the command for transmission over radio to the target visual tag.
4. CommandSent – bool - if True, indicates the Omni Middleware Service has sent the command over the radio to the target visual tag.
5. CommandReceived – bool - if True, indicates that the target visual tag has received the command via radio, and has successfully processed the command.
6. CommandFailed – bool – If True, indicates that the target visual tag was unreachable via radio or has rejected the image update command.
7. CommandRetrying – bool – If True, indicates the command failed and it is being retried.
8. Information – string – if CommandFailed is True, provides additional textual information about the nature of the failure for diagnostic purposes, otherwise provides more detailed status information.

### OmniImageUrlUpdateCommand

#### Description

This command is used to send an arbitrary image to update the image of a specific visual tag.

#### Fields

1. TransactionID – GUID – the command / response correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag
3. ImageURL – string – a URL to an arbitrary image to be sent to the target tag. The image should be monochrome bitmap files (.bmp) of the correct image dimensions for the targeted visual tag. The syntax of the string is validated against RFC 2396 and RFC 2732. An example local ImageURL is “file:///C:/temp/image.bmp”
4. PageNumber – int – the 1-based index of the page to store the image on the target visual tag.
5. GatewayID – string – the unique identifier of the target Gateway
6. IsTransactional – bool – True means this command is part of a group of commands for the specified visual tag. If this command fails then all other commands that are queued for the specified visual tag that have not yet been executed should be cancelled. A group is defined as any other commands for the specified visual tag that are queued at the time this command fails and these command are marked as transactional commands. For example let's suppose an image update and page change command are queued to CORAL 123. The commands are marked as transactional. The image update command fails after all retries. The page change command will be deleted because it is marked as part of the same transaction as the image update command. It does not make sense flip the page if the image download did not succeed.   
   False means if the command fails it has no effect on other commands. False is the default.

#### Result Message

Returns the same result message as the OmniImageUpdateCommand. See above for details.

### OmniPageChangeCommand

#### Description

This command is used to command a specific visual tag to change its display to an already-stored image in the specified page memory buffer.

#### Fields

1. TransactionID – GUID – the command / response correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag
3. PageNumber – int – the 1-based index of the page to store the image on the target visual tag.
4. GatewayID – string – the unique identifier of the target Gateway
5. IsTransactional – bool – True means this command is part of a group of commands for the specified visual tag. If this command fails then all other commands that are queued for the specified visual tag that have not yet been executed should be cancelled. A group is defined as any other commands for the specified visual tag that are queued at the time this command fails and these command are marked as transactional commands. For example let's suppose an image update and page change command are queued to CORAL 123. The commands are marked as transactional. The image update command fails after all retries. The page change command will be deleted because it is marked as part of the same transaction as the image update command. It does not make sense flip the page if the image download did not succeed.   
   False means if the command fails it has no effect on other commands. False is the default.

#### Result Message

1. TransactionID – GUID – message correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag.
3. CommandQueued – bool – if True, indicates the Omni Middleware Service has received and queued the command for transmission over radio to the target visual tag.
4. CommandSent – bool - if True, indicates the Omni Middleware Service has sent the command over the radio to the target visual tag.
5. CommandReceived – bool - if True, indicates that the target visual tag has received the command via radio, and has successfully processed the command.
6. CommandFailed – bool – If True, indicates that the target visual tag was unreachable via radio or has rejected the command.
7. CommandRetrying – bool – If True, indicates the command failed and it is being retried.
8. Information – string – if CommandFailed is True, provides additional textual information about the nature of the failure for diagnostic purposes, otherwise provides more detailed status information.

### OmniPageDeleteCommand

#### Description

This command is used to command a specific visual tag to delete the image in the specified page memory buffer.

#### Fields

1. TransactionID – GUID – the command / response correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag
3. PageNumber – int – the 1-based index of the page to delete on the target visual tag.
4. GatewayID – string – the unique identifier of the target Gateway
5. IsTransactional – bool – True means this command is part of a group of commands for the specified visual tag. If this command fails then all other commands that are queued for the specified visual tag that have not yet been executed should be cancelled. A group is defined as any other commands for the specified visual tag that are queued at the time this command fails and these command are marked as transactional commands. For example let's suppose an image update and page change command are queued to CORAL 123. The commands are marked as transactional. The image update command fails after all retries. The page change command will be deleted because it is marked as part of the same transaction as the image update command. It does not make sense flip the page if the image download did not succeed.   
   False means if the command fails it has no effect on other commands. False is the default.

#### Result Message

1. TransactionID – GUID – message correlation ID
2. VisualTagUID – string – the unique identifier of the target visual tag.
3. CommandQueued – bool – if True, indicates the Omni Middleware Service has received and queued the command for transmission over radio to the target visual tag.
4. CommandSent – bool - if True, indicates the Omni Middleware Service has sent the command over the radio to the target visual tag.
5. CommandReceived – bool - if True, indicates that the target visual tag has received the command via radio, and has successfully processed the command.
6. CommandFailed – bool – If True, indicates that the target visual tag was unreachable via radio or has rejected the command.
7. CommandRetrying – bool – If True, indicates the command failed and it is being retried.
8. Information – string – if CommandFailed is True, provides additional textual information about the nature of the failure for diagnostic purposes, otherwise provides more detailed status information.

## Event Messages

This section lists the OmniAPIMessage- derived class instances that are not first solicited by sending an OmniAPICommand instance to the Omni Middleware Service, with the exception of the **OmniCommandErrorResultEvent**, which is sent when the Omni Middleware Service rejects a malformed command message that was submitted.

### OmniRFIDDetectionMessage

#### Description

This event message is generated asynchronously and sent to the .NET API client when the Omni Middleware Service records an RFID tag read from a specific RFID reader.

#### Fields

1. RFIDDeviceID – string- the unique system identifier for the fixed RFID reader that generated the RFID read event. This is typically a GUID string.
2. RFIDDeviceAddress – string – the IP or DNS multicast name of the RFID readerer.
3. RFIDDeviceAntennaID – int – the identifier for the specific antenna on the specific RFID reader
4. RFIDTagEPC – string – the EPC code of the RFID tag that was detected.
5. RFIDTagUserMemory – string – Contents of the RFID tag user memory.
6. TimeStamp – DateTime – Timestamp of the RFID read.

### OmniVisualTagAnnounceEvent

#### Description

This event message is generated and sent asynchronously to the .NET API client to report the presence of a specific visual tag.

#### Fields

1. VisualTagUID – string – the unique identifier of the visual tag being reported on

### OmniVisualTagHealthReportMessage

#### Description

This event message is generated and sent asynchronously to the .NET API client to report the health status of a specific visual tag, for instance to communicate that the battery level is low.

#### Fields

1. VisualTagUID – string – the unique identifier of the visual tag being reported on
2. BatteryLevel – int – the battery level of the visual tag.

### OmniGPIEventMessage

#### Description

This event message is generated and sent asynchronously to the .Net API client to report that a specific GPI port on a specific fixed RFID reader has changed state.

#### Fields

1. RFIDDeviceID – string – the unique identifier of the fixed RFID reader that reported the GPI state change event. This is typically a GUID string.
2. RFIDDeviceAddress – string – the IP or DNS multicast name of the RFID readerer.
3. GPIPortNumber – int – the 0-based index of the GPI port that has changed state.
4. PortState – GPIOPortState – GPIOPortState.High if port state has changed to high, or GPIOPortState.Low if port state has changed to low.

### OmniCommandErrorResultEvent

#### Description

The Omni Middleware Service generates this message if it receives a command through the .NET client API that it has rejected, giving diagnostic information.

#### Fields

1. TransactionID - GUID – the command correlation ID.
2. ErrorCode – string – the Omni Middleware Service error code (a listing of which is provided in the Omni System Metadata XML document)
3. Information – string – additional diagnostic information about the error.

### OmniSystemErrorEvent

#### Description

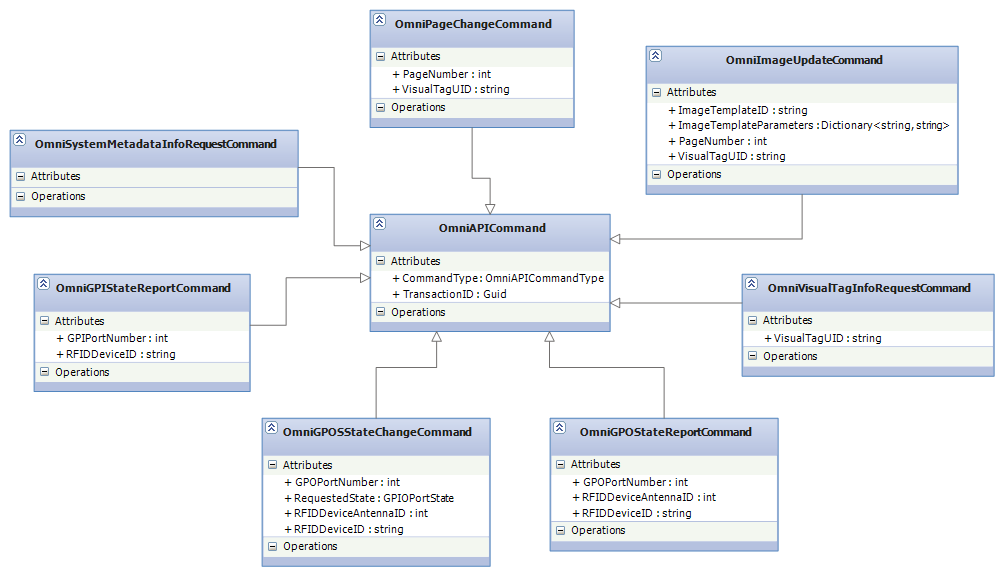
The Omni Middleware Service generates this message if it encounters a system-level error, for example if network communications with a certain fixed reader has been lost.

#### Fields

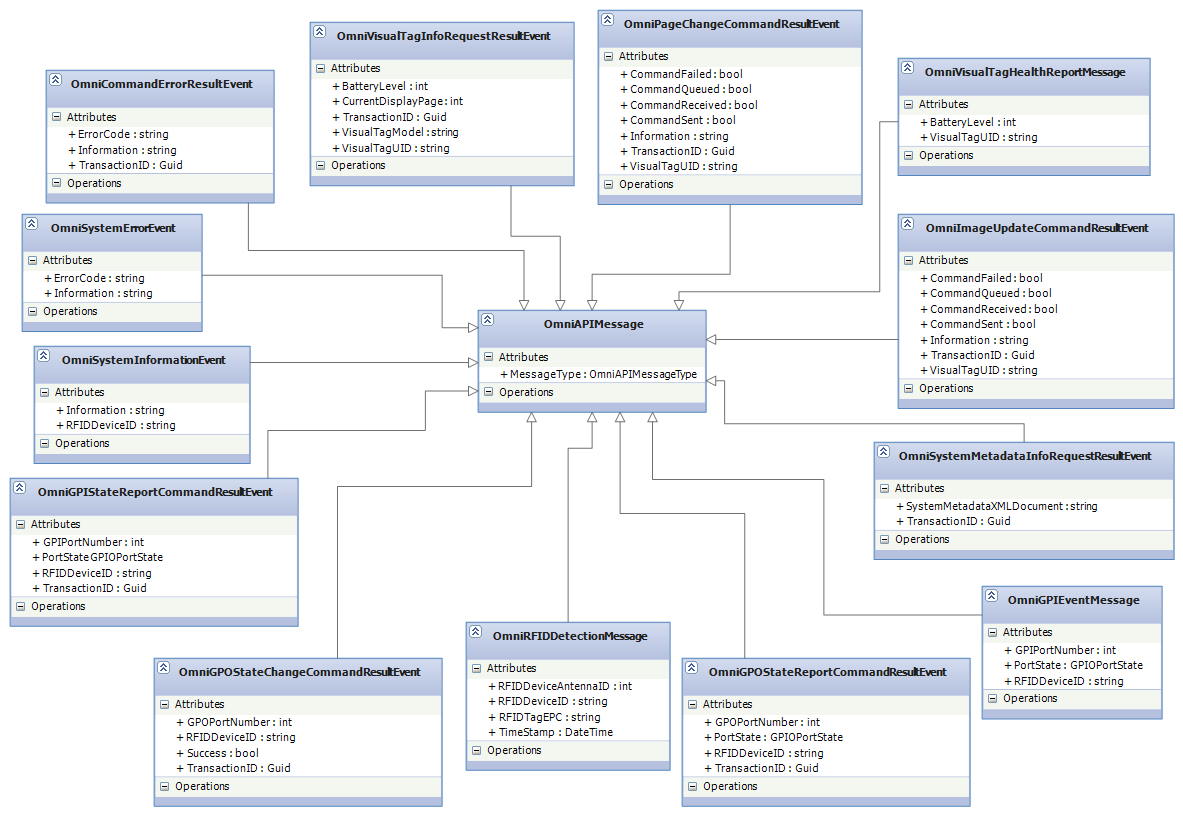
1. ErrorCode – string – the Omni Middleware Service error code (a listing of which is provided in the Omni System Metadata XML document)
2. Information – string – additional diagnostic information about the error.

# Class diagrams

## Command Classes



## Message Classes



# Configuration

## CALC Manager, Omni Impinj Reader App and Middleware Clients

Use the Middleware Client Configurator utility to configure Middleware clients. This utility is available via the start menu. Use the "Add new client" button to configure Middleware clients. In most cases the host name and client host names will be "localhost". The OmniMiddlewareClient constructor and app.config values will be generated for each Middleware client. Changes to the configuration will take effect when the associated service/app is restarted. Use the following rules to create a configuration:

1. All instances of the Omni Impinj reader app must be run on the computer that the OmniSystemScheduler is running on.
2. Host name fields should be set to localhost if the Middleware client is executing on the same machine. If not, it should be set to the machine name the service or app is running on.
3. Client host names should be set to localhost if the Middleware client is executing on the same machine. If not, it should be set to the machine name the service or app is running on.

Example 1:

CALCMan is running on the same computer as the Middleware client.

CALCMan host name: "localhost"

CALCMan client host name: "localhost"

Example 2:

CALCMan is running a computer named "MW\_Server" and the the Middleware client is running on a computer named "MW\_Client"

CALCMan host name: "MW\_Server"

CALCMan client host name: "MW\_Client"

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Revision** | **Change Description** |
| 12/3/2012 | M. Doser | 0.1 | Initial Draft |
| 12/4/2012 | M. Doser | 0.2 | Added current page displayed to OmniVisualTagInfoResult  message. |
| 2/5/2013 | Kyle Mullins | 0.3 | Changed PortState to an enum |
| 2/8/2013 | Kyle Mullins | 0.4 | Added short install/uninstall section |
| 2/19/2013 | Kyle Mullins | 0.5 | Updated data types for some message/command fields |
| 2/20/2013 | Kyle Mullins | 0.6 | Updated class diagrams |
| 4/10/2013 | Kyle Mullins | 0.7 | Added OmniImageUrlUpdateCommand and OmniRFIDReaderCommand |
| 8/1/2013 | Kyle Mullins | 0.8 | Added OmniPageDeleteCommand and sequence diagram, also made some misc updates |
| 8/27/2013 | Aaron Williams | 0.9 | Added CommandRetrying, GatewayID, and IsTransactional |
| 8/16/2013 | Aaron Williams | 1.0 | Added QueueNames.txt configuration |
| 8/23/2013 | B. Huzyk | 1.01 | Added system requirements and system diagram. |
| 8/25/2013 | B. Huzyk | 1.02 | Added new 0 parameter constructor. |
| 8/28/2013 | B. Huzyk | 1.03 | 1. Updated system block diagram section to include actual software component names.  2. Updated OmniImageUpdateCommand to include CoralType field.  3. Updated OmniRFIDDetectionMessage fields. |
| 8/29/2013 | B. Huzyk | 1.04 | Modified OmniMiddlewareClient constructors, creating OmniMiddlewareClient class section, and configuration section. |
| 8/31/2013 | B. Huzyk | 1.05 | Updated OmniImageUrlUpdateCommand ImageUrl parameter documentation.  Changed system block diagram. |
| 11/13/2013 | B. Huzyk | 1.06 | 1. Updated configuration section to reflect new Middleware client configurator utility.  2. Updated RFID events, added RFIDDeviceAddress  Field. |