HUMAX

IxWebSocketServer
Software Design

Eng Ver 2.0

Ver	Date	Change	Author
1.0	2016 09/21	Draft Version Release	Jongwon Kim, Soyoung Kim
1.01	2017 07/07	Update APIs, First release of English	
1.02	2017 07/20	Update class, sequence, plugin	GyoungChul Jung, Kwonsik Kim
1.03	2017 08/27	Update class, sequence, sdk package	GyoungChul Jung
1.04	2017 09/19	Update class	GyoungChul Jung
1.2	2018 01/08	Update Environment part	GyoungChul Jung
2.0	2018 03/26	A new server supporting the http protocol was applied.	Kwonsik Kim

Contents

Picture Contents

1 Introduction

IX of IXWSS stands for Interface Extension, and WSS stands for WebSocket Server. In other words, IXWSS can be understood as a component that can easily extend the interface between client and server based on WebSocket/HTTP communication protocol. In a client / server architecture, a communication protocol is basically a standard method, but various data formats may be required for data to be transmitted. IXWSS provides an environment that can be developed using individual plugins so that users can directly process data in various formats.

At this point, the following requirements are required to use IXWSS.

- ✓ WebSocket Server in the casting system as a data relay agent between sender and receiver
- ✓ WebSocket Server to support JSON-PRC

2 Overview

2.1 Software Architecture

IXWSS is a component for data processing and transmission/reception between client and server based on communication protocol as described in chapter 1. The overall structure is shown in the figure below.

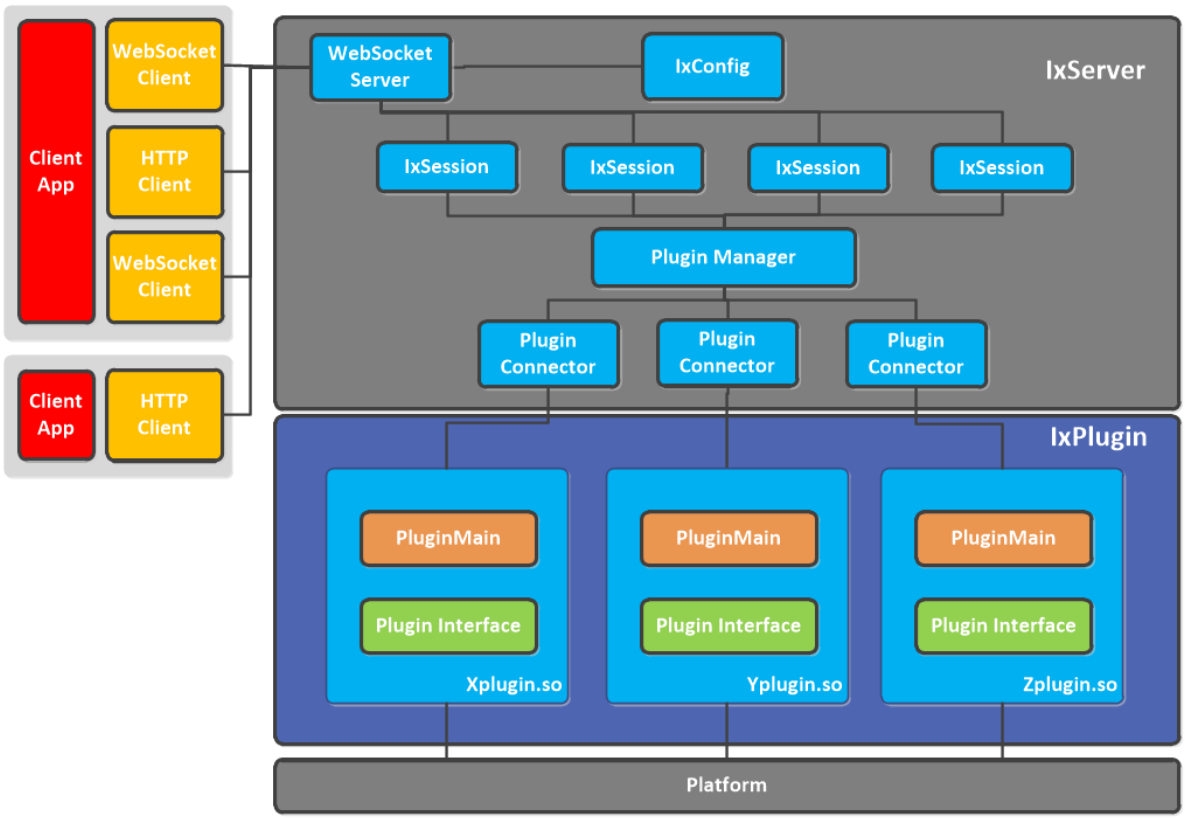


Figure 1. IXWSS Architecture

IXWSS is a component based on WebSocket Server. It creates each session dynamically each time a client is connected, and each session is connected to the client in a 1:1 connection.

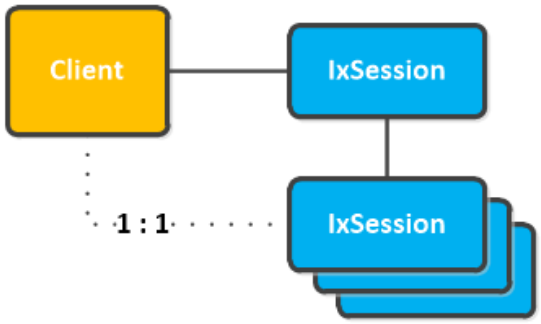


Figure 2. Client & Server

All data from the client is received by each session, and the receive data is passed to plugin loaded at the session start. The advantage of IXWSS is that users can process the data through the plug-in in the form they want. In addition, each plug-in can send any type of data to the client side. In other words, the data received from each session is transferred to the individual plug-in, and the transferred data can be processed in any form in the plug-in. If a response is required to the client side, data can be created and transmitted in a desired form.

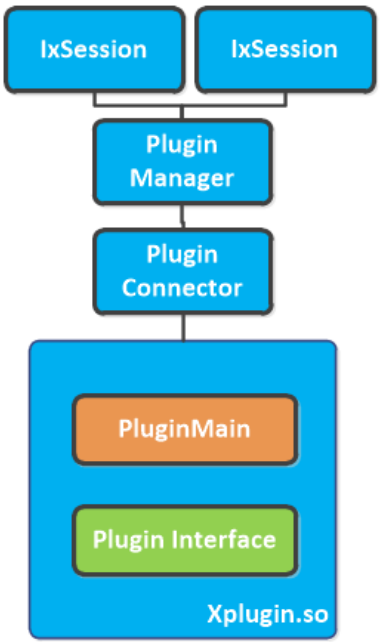


Figure 3. IxSession & PluginConnector

The last module to introduce is IXPlugin. The requirement for various WebSocket-based services are increasing. In order to support theses requirement as a single software component, we have structured the plug-in that can be loaded dynamically by users. We tried

to include all of the following things while considering the plug-in development environment structure.

- ✓ Easy of development
  - ■ Easy interface for developers to easily develop
  - ■ Provides various macros
- ✓ Independent build environment
  - ■ Provide 100% independent build environment with IXWSS
  - ■ Shared Object type output
- ✓ Various Examples
  - ■ Simple Plugin Example
  - ■ JSON-RPC Plugin Example
- ✓ Provides Development Guide
  - ■ Provides the development guide of Plugin

3 IXWSS

3.1 Introduction

IXWSS is a server module that is implemented to load the necessary plugins and communicate with the clients using libwebsockets. It support the multi-port and sub-protocols, and communication data format can be used variously according to plug-in which processes data. The following describes how to run the server and how to connect and communicate with clients.

3.2 How to start IXWSS

IXWSS is distributed as a shared library and runs independently of the plugin. How to start IXWSS is following.

1) Two library are required to configure the IXWSS server.
   A. libwebsocket.so
   B. libixs.so
2) To start server, create a server object.
3) When creating the server object, the folder location where the configure file needed for the server operation exists should be passed as a parameter.
4) The server configuration file and the plug-in configuration file must exist in the folder path of the delivery, and the server is set by the corresponding information when the server operates.
   A. ixconfig.conf: Server & Plugin Configuration

```
# IXServer check this file to load plugin when the client request
#
```

```
# @ field : plugin_path
# description :
#    set the path of ixplugin library full path
# example > #    plugin_path=/usr/browser/unittest/ixplugins/bin
#
# @ field : plugin_map
# description :
#    set the port numerber and prototocol and sub-protocol and ixplugin library file
#    that ixs has to load when the web client request. if you don't want to use sub-protocol
#    about some port , please refer to port 9997 in the example below
# example >
# plugin_path=../ixplugins/bin
# plugin_map=9999:http::libixplugin_simple.so
# plugin_map=9999:ws:jsonrpc:libixplugin_jsonrpc.so
# plugin_map=9998:http::libixplugin_jsonrpc.so
# plugin_map=9997:ws::libixplugin_simple.so
#

plugin_path=../ixplugins/bin
plugin_map=9999:http::libixplugin_simple.so
plugin_map=9999:ws:jsonrpc:libixplugin_jsonrpc.so
plugin_map=9998:http::libixplugin_jsonrpc.so
plugin_map=9997:ws::libixplugin_simple.so
```

5) The server waits for the client to connect.

6) The code above is simplified as follows.

```
const char *conf_path = "/usr/browser/ixwss/config/ixconfig.config";
IXWebSocketServer *server = new IXWebSocketServer();
server->initialize(conf_path);
server->start();
```

3.3 What is the IxSession

When a client connects to a port and sub-protocols that the server allows, the server creates a session to communicate with the client. Internally, it creates a thread to process event and command, and waits for communication between client and plugin. For example, if a client connects to an unauthorized PORT or subprotocol, the server will now allow access, no sessions will be created and an error will be generated.

4 IXWSS Class Design

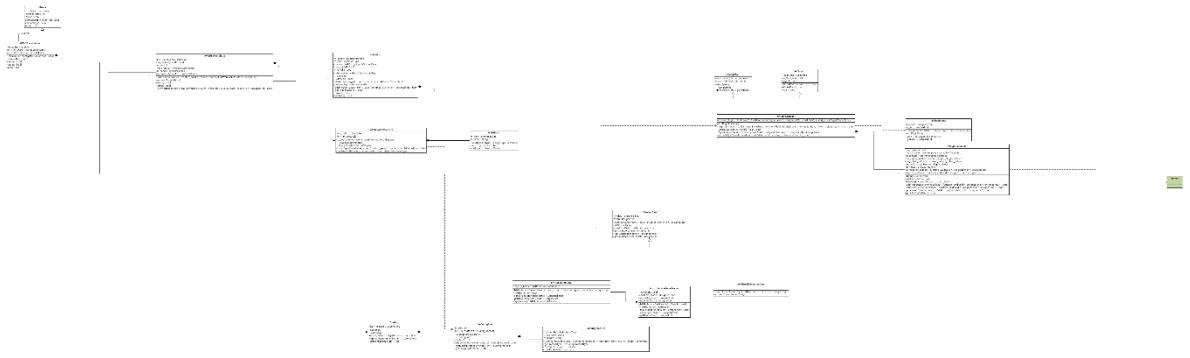
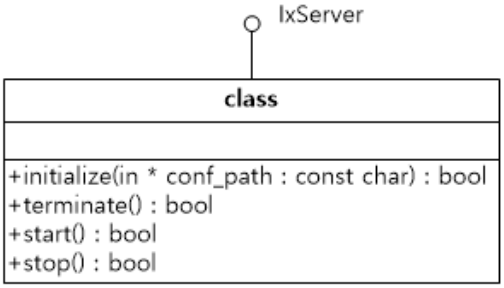


Figure 4. IXWSS Class Diagram

IXWSS consists of the above class modules.
IxWebSocketServer runs the server based on the information set in the configuration files in the folder path that is passed when the server object is created. IxWebSocketVhost handles session management. An IxSession session is created by client connection, and the session communicates with the plugin interface via IxPluginManger.
Each class is described in Section 4.1 through Section 4.11.

4.1 IxServer



Base class for creating IXServer.

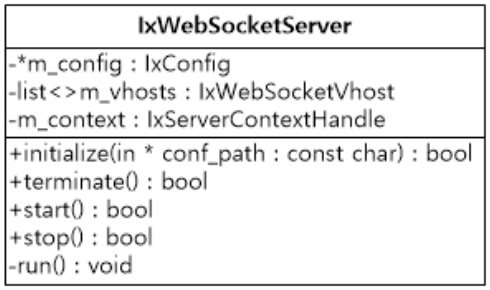
■ Member Function

함수명	initialize
매개변수	const char *conf_path
리턴값	bool
기능정의	Initialize the server with the configuration file.

함수명	start
매개변수	void
리턴값	bool
기능정의	Start the server.

함수명	stop
매개변수	void
리턴값	bool
기능정의	Stop the server.

4.2 IxWebSocketServer



It is an interface for creating an IX Server application by inheriting IxServer and implementing it with libwebsocket.

■ Member Functions

함수명	IxWebSocketServer
매개변수	-
리턴값	-
기능정의	Constructor for IXWebSocketServer.

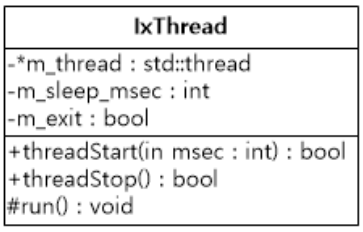
함수명	~IxWebSocketServer
매개변수	-
리턴값	-
기능정의	Destructor for IXWebSocketServer.

함수명	initialize
매개변수	const char *conf_path
리턴값	bool
기능정의	Initialize the server with the configuration file.

함수명	start
매개변수	void
리턴값	bool
기능정의	Start the server.

함수명	stop
매개변수	void
리턴값	bool
기능정의	Stop the server.

4.3 IxThread



A base class for a thread.

■ Member Functions

함수명	**IxThread**
매개변수	-
리턴값	-
기능정의	Constructor for IxThread.

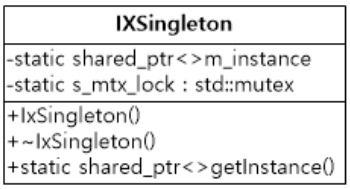
함수명	~IxThread
매개변수	-
리턴값	-
기능정의	Destructor for IxThread.

함수명	start
매개변수	Int msec
리턴값	bool
기능정의	Start the thread.

함수명	stop
매개변수	void
리턴값	bool
기능정의	Stop the thread.

함수명	**run**
매개변수	-
리턴값	-
기능정의	It is called with the sleep interval set by the thread.

4.4 IxSingleton



A template class for creating singleton objects. You can inherit the object, hide the constructor internally, declare getInstance() as friend, and get the singleton object via getInstance().

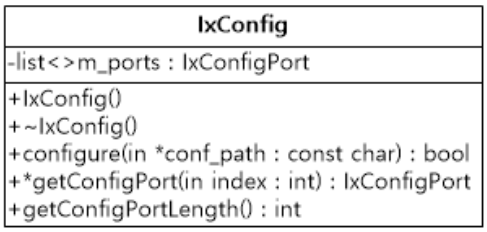
■ Member Functions

함수명	**getInstance**
매개변수	void
리턴값	shared_ptr<T>
기능정의	Gets the singleton object

함수명	IxSingleton
매개변수	-
리턴값	-
기능정의	Constructor for IxSingleton.

함수명	~IxSingleton
매개변수	-
리턴값	-
기능정의	Destructor for IxSingleton.

4.5 IxConfig



It reads the server drive information from the file and creates the data structure. The server activation information includes port number, allowed protocol, and plugin information to be connected to the protocol.

■ Member Functions

함수명	IxConfig
매개변수	-
리턴값	-
기능정의	Constructor for IxConfig.

함수명	~IxConfig
매개변수	-
리턴값	-
기능정의	Destructor for IxConfig.

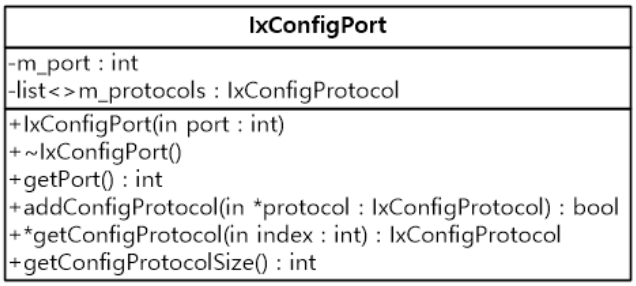
함수명	configure
매개변수	const char *conf_path
리턴값	bool
기능정의	Proceed with the configuration from the configuration file.

함수명	getConfigPort
매개변수	int index
리턴값	const IxConfigPort *

기능정의	Gets a reference to the port configuration object.

함수명	getConfigPortLength
매개변수	void
리턴값	int
기능정의	Gets the number of the port configuration objects.

4.6 IxConfigPort



It is a class that has data structure for port information. The port information includes information such as a port number and a supported protocol.

■ Member Functions

함수명	IxConfigPort
매개변수	int port
리턴값	-
기능정의	Constructor for IxConfigPort.

함수명	~IxConfigPort
매개변수	-
리턴값	-
기능정의	Destructor for IxConfigPort.

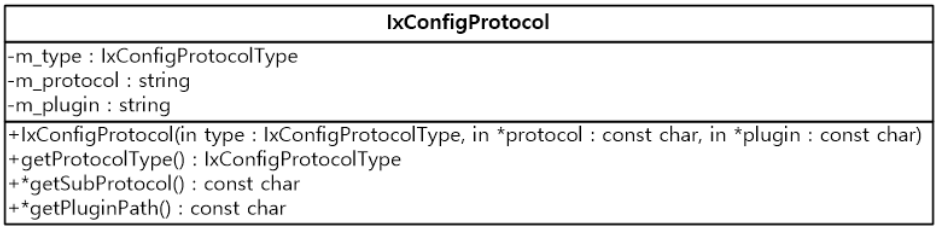
함수명	getPort
매개변수	void
리턴값	int
기능정의	Get the port.

함수명	addConfigProtocol
매개변수	IxConfigProtocol *protocol
리턴값	bool
기능정의	Add an object for protocol configuration.

함수명	getConfigProtocol
매개변수	int index
리턴값	const IxConfigProtocol *
기능정의	Get a reference to an object for protocol configuration.

함수명	getConfigProtocolSize
매개변수	void
리턴값	int
기능정의	Get the number of object references for the protocol configuration.

4.7 IxConfigProtocol



A data structure class that contains protocol information. The protocol information includes the protocol type and sub-protocol, and the plug-in path associated with the protocol. The protocol types are HTTP type and WebSocket type.

■ Member Functions

함수명	IxConfigProtocol
매개변수	IxConfigProtocolType type, const char *protocol , const char *plugin
리턴값	-
기능정의	Constructor for IxConfigPort.

함수명	getProtocolType
매개변수	void
리턴값	IxConfigProtocolType
기능정의	Get the protocol type.

함수명	getSubProtocol
매개변수	void
리턴값	const char *
기능정의	Get the sub-protocol value.

함수명	getPluginPath
매개변수	void
리턴값	const char*
기능정의	Get the plugin path for the protocol.

4.8 IxWebSocketVhost

IxWebSocketVhost
-list<>m_sessions : IxSession -*m_context_handle : void -m_port : int -*m_protocol : IxConnectionInfo -m_handle : IxVhostHandle -m_protocol_handle : IxProtocolHandle
+IxWebSocketVhost(in *context_handle : void, in *protocol : IxConnectionInfo, in port : int) +~IxWebSocketVhost() +start() : bool +stop() : bool +onVhostCallback(in event : IxWebsocketEvent, in *handle : void, in *data : void, in len : unsigned int) : bool

Sets the ports allowed by the server, and manages the protocol corresponding to the ports. The client creates a session for the connection, determines whether the session is valid, and manages the connection with the plug-in corresponding to the session.

■ Member Functions

함수명	IxWebSocketVhost
매개변수	void *context_handle, int port, IxConnectionInfo *info
리턴값	-
기능정의	Constructor for IxWebSocketVhost.

함수명	~IxWebSocketVhost
매개변수	-
리턴값	-
기능정의	Destructor for IxWebSocketVhost.

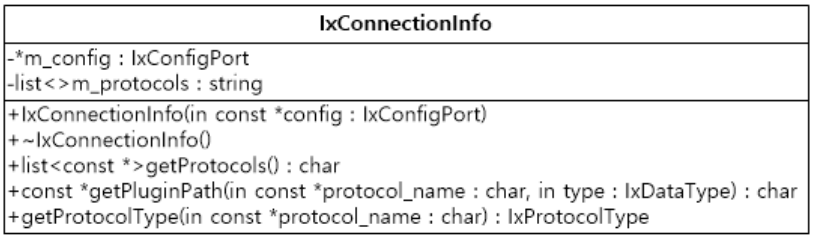
함수명	start
매개변수	void
리턴값	bool
기능정의	Start the Vhost.

함수명	stop

매개변수	void
리턴값	bool
기능정의	Stop the Vhost.

함수명	onVhostCallback
매개변수	LWS IxWebsocketEvent event, void *handle, void *data , unsigned int len
리턴값	bool
기능정의	Receive events from the client.

4.9 IxConnectionInfo



It is a data structure for judging whether the client connection is valid or not. It has information about the protocol and plugin path set in the server.

■ Member Functions

함수명	IxConnectionInfo
매개변수	const IxConfigPort *config
리턴값	-
기능정의	Constructor for IxConnectionInfo.

함수명	~IxConnectionInfo
매개변수	-
리턴값	-

기능정의	Destructor for IxConnectionInfo.

함수명	getProtocols
매개변수	void
리턴값	list<const char *>
기능정의	The protocol list to be set by the configuration is fetched.

함수명	**getPluginPath**
매개변수	const char *protocol_name, IxDataType type
리턴값	const char *
기능정의	Get the plugin path corresponding to the protocol.

함수명	**getProtocolType**
매개변수	const char *protocol_name
리턴값	IxProtocolType
기능정의	Get the protocol type by protocol name.

4.10 IXWebSocketSession

IxSession
-m_handle : IxSessionHandle -m_type : IxSessionType -queue<const *>m_data : IxSessionData -m_is_started : bool -m_is_valid : bool
+IxSession(in handle : IxSessionHandle) +~IxSession() +isStarted() : bool +start(in *plugin_path : const char, in type : IxSessionType) : bool +getHandle() : IxSessionHandle +pushRequestData(in const *data : unsigned char, in len : unsigned int) : bool +popResponseData() : bool +isValid() : bool +setValid() : bool

If the client is allowed to connect, a session is created. Sessions are of the type HTTP, WebSocket, etc. depending on the nature of the client. Through the session, the client connects to the plug-in and sends and receives data.

■ Member Functions

함수명	IxSession
매개변수	IxSessionHandle handle
리턴값	-
기능정의	Constructor for IxSession.

함수명	~IxSession
매개변수	-
리턴값	-
기능정의	Destructor for IxSession.

함수명	isStarted
매개변수	void
리턴값	bool
기능정의	세션이 시작되었는지 확인한다.

함수명	start
매개변수	IxSessionType type, const char *plugin_path
리턴값	bool
기능정의	Check whether the session is started.

함수명	getHandle
매개변수	void
리턴값	IxSessionHandle
기능정의	Gets a reference to lws of the websocket library.

함수명	pushRequestData

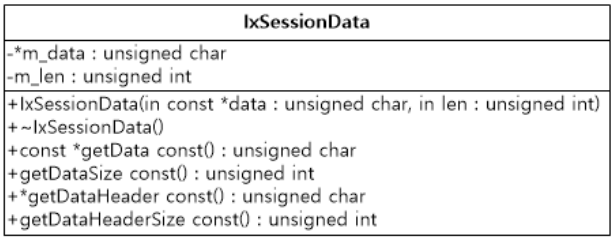
매개변수	const unsigned char *data, unsigned int len
리턴값	bool
기능정의	Request data from the plug-in.

함수명	popResponseData
매개변수	void
리턴값	bool
기능정의	Process any data received from the plug-in.

함수명	isValid
매개변수	void
리턴값	bool
기능정의	Check that the session is available.

함수명	setValid
매개변수	bool valid
리턴값	bool
기능정의	Sets whether the session is available.

4.11 IxSessionData



A session is a data class for storing data to and from a client. It also includes data structures for header data, if necessary.

■ Member Functions

함수명	IxSessionData
매개변수	const unsigned char *data, unsigned int len
리턴값	-
기능정의	Constructor for IxSessionData.

함수명	~IxSessionData
매개변수	-
리턴값	-
기능정의	Destructor for IxSessionData.

함수명	getData
매개변수	void
리턴값	const unsigned char *
기능정의	Get data to process in the session.

함수명	getDataSize
매개변수	void
리턴값	unsigned int
기능정의	Gets the size of the data processed by the session.

함수명	**getDataHeader**
매개변수	void
리턴값	const unsigned char *
기능정의	Get data header to process in the session.

함수명	**getDataHeaderSize**
매개변수	void

리턴값	unsigned int
기능정의	Gets the size of the header data processed by the session.

4.12 IxWebSocketSessionData

IxWebSocketSessionData
+IxWebSocketSessionData(in *data : const char, in len : unsigned int) +~IxWebSocketSessionData()

It inherits IxSessionData and stores the data for the WebSocket client.

■ Member Functions

함수명	IxWebSocketSessionData
매개변수	const unsigned char *data, unsigned int len
리턴값	-
기능정의	Constructor for IxWebSocketSessionData.

함수명	~IxWebSocketSessionData
매개변수	-
리턴값	-
기능정의	Destructor for IxWebSocketSessionData.

4.13 IxHttpSessionData

IxHttpSessionData
-*m_http_header : IxHttpSessionDataHeader
+IxHttpSessionData(in *handle : void, in const *data : unsigned char, in len : unsigned int) +~IxHttpSessionData() +const *getDataHeader const() : unsigned char +getDataHeaderSize const() : unsigned int +*getHeader() : IxHttpSessionDataHeader

It inherits IxSessionData and stores the data for the HTTP client. The data includes header data for the HTTP client.

■ Member Functions

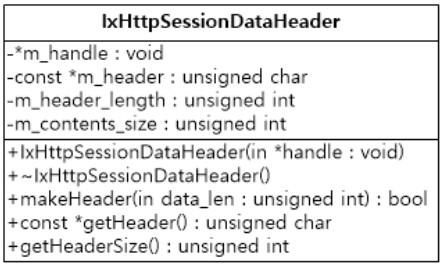
함수명	IxHttpSessionData
매개변수	void *handle, const unsigned char *data, unsigned int len
리턴값	-
기능정의	Constructor for IxHttpSessionData.

함수명	~IxHttpSessionData
매개변수	-
리턴값	-
기능정의	Destructor for IxHttpSessionData.

함수명	getDataHeader
매개변수	void
리턴값	const unsigned char *
기능정의	Get header data of the data to be sent.

함수명	getDataHeaderSize
매개변수	void
리턴값	unsigned int
기능정의	Get the header data size of the data to be sent.

4.14 IxHttpSessionDataHeader



It is a class that stores header data for HTTP client. And generates header data through a handle received from the server.

■ Member Functions

함수명	IxHttpSessionDataHeader
매개변수	const char *data, int len
리턴값	-
기능정의	Constructor for IxHttpSessionDataHeader.

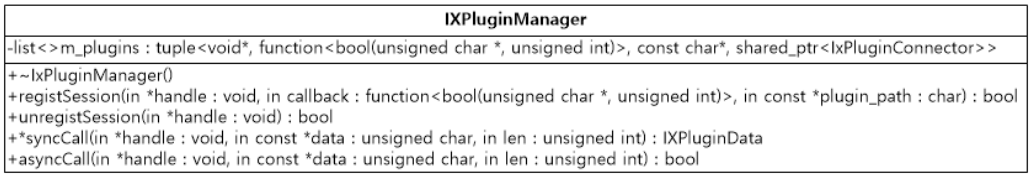
함수명	~ IxHttpSessionDataHeader
매개변수	-
리턴값	-
기능정의	Destructor for IxHttpSessionDataHeader.

함수명	makeHeader
매개변수	void
리턴값	bool
기능정의	Create header data of the data to be sent.

함수명	getData
매개변수	void
리턴값	const char *
기능정의	Get header data of the data to be sent.

함수명	**getDataLength**
매개변수	void
리턴값	int
기능정의	Get the header data size of the data to be sent.

4.15 IxPluginManager



It acts as a thread to manage the connection between the plug-in and the session and to act as a relay for the data request / response.

■ Member Functions

함수명	getInstance
매개변수	void
리턴값	shared_ptr<IxPluginManager>
기능정의	Friend declaration for Getting the singleton object of IxPluginManager.

함수명	~IXPluginAdapter
매개변수	-
리턴값	-
기능정의	Destructor for IxPluginManager.

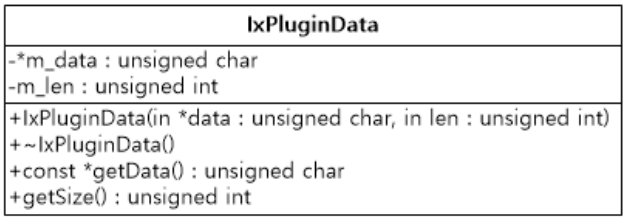
함수명	registSession
매개변수	void *handle, function<bool(unsigned char *data, unsigned int len)> callback , const char *plugin_path
리턴값	bool
기능정의	Registers a listener for receiving plug-in data.

함수명	unRegistSession
매개변수	void *handle
리턴값	bool
기능정의	Unregisters a listener for receiving plug-in data.

함수명	syncCall
매개변수	void *handle, const unsigned char *data, unsigned int len
리턴값	IxPluginData *
기능정의	Receives plugin data as a return.

함수명	asyncCall
매개변수	void *handle, const unsigned char *data, unsigned int len
리턴값	bool
기능정의	Receives plugin data as a registered listener.

4.16 IxPluginData



The data class that the plug-in manager passes.

■ Member Functions

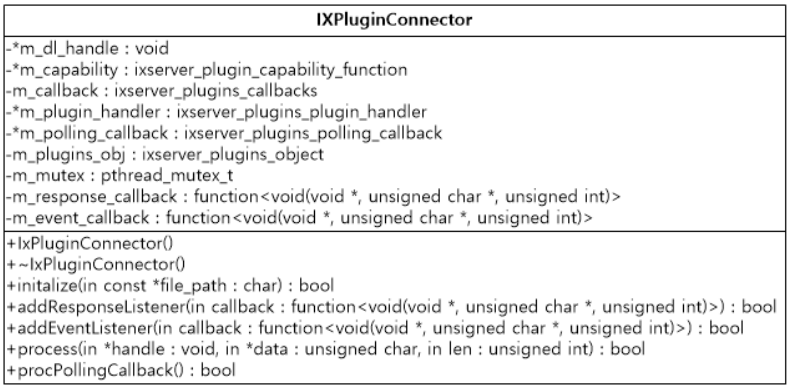
함수명	IxPluginData
매개변수	unsigned char *data, unsigned int len
리턴값	-
기능정의	Constructor for IxPluginData.

함수명	**~IxPluginData**
매개변수	-
리턴값	-
기능정의	Destructor for IxPluginData.

함수명	**getData**
매개변수	void
리턴값	const unsigned char *
기능정의	Get plugin data to send.

함수명	**getSize**
매개변수	void
리턴값	unsigned int
기능정의	Gets the size of the plugin data.

4.17 IxPluginConnector



Loading of one plug-in, request / response of data, and event handling.

■ Member Functions

함수명	IxPluginConnector
매개변수	-
리턴값	-
기능정의	Constructor for IxPluginConnector.

함수명	**~IxPluginConnector**

매개변수	-
리턴값	-
기능정의	Destructor for IxPluginConnector.

함수명	**initialize**
매개변수	const char *file_path
리턴값	bool
기능정의	Initialize the connector.

함수명	**addResponseListener**
매개변수	function<void(void *session_handle, unsigned char *data, unsigned int len)> callback
리턴값	bool
기능정의	Registers a listener for receiving a response to the request.

함수명	**addEventListener**
매개변수	function<void(void *connector_handle, unsigned char *data, unsigned int len)> callback
리턴값	bool
기능정의	Registers a listener for receiving a plugin event.

함수명	**process**
매개변수	void *handle, unsigned char *data, unsigned int len
리턴값	bool
기능정의	Request data from the plug-in.

함수명	**procPollingCallback**
매개변수	void
리턴값	bool

기능정의	Poll the plug-in event. It should be called periodically to receive events.

5 IXPlugin

5.1 What is IXPlugin?

IXWSS is designed to process the request of the client after loading plugin through PluginConnector designed for plugin loading of PluginManger. This chapter describes how to make a plugin and how to load the plugin from the PluginConnector.

	If the plugin fails to load because it does not exist, it will send a plugin loading failure message to IXWSS.

5.2 How does it work?

In case of IXWSS, home directory exists by default and the following files exist under the directory.

- ✓ ixplugin.ini : Shared object mapping table to be loaded for port number, protocol and sub-protocol
- ✓ {xxx}.so : Plugin (share object) created by user

5.3 How to load plugin?

*int ixserver_plugin_capability_function (ixserver_plugins_callbacks *cb,*
*ixserver_plugins_plugin_handler **ret)*

IXWSS calls the above function when the plugin is loaded.
The plugin creator should implement the following two things.

1) Keep the cb parameter received by ixserver_plugins_callbacks internally. This callback parameter is used to pass a response or event from the plugin to the client.
2) ixserver_plugins_plugin_handler assigns each function handler that it requires through the return parameter. Each return parameter is called when IXWSS requires the plugin to operate.

This part is a basic mutual promise for making a plugin, and preparation for plugin operation is completed through this process.

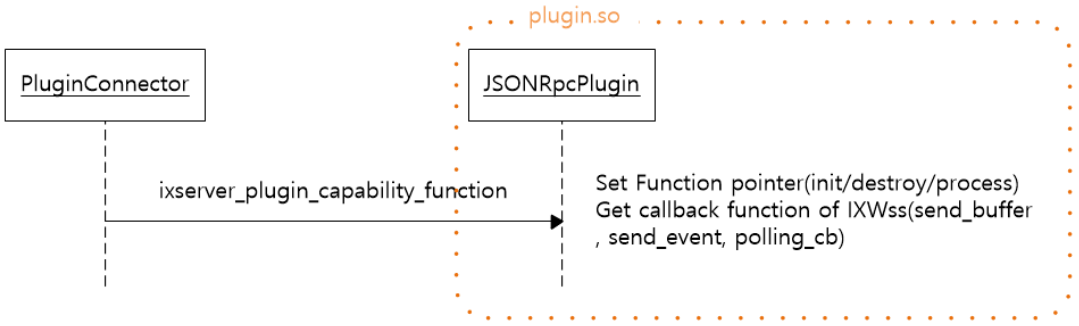


Figure 5. Plugin Load

```
static ixserver_plugins_plugin_handler s_plugin_handler;
static ixserver_plugins_callbacks* s_wss_cb = NULL;

int ixserver_plugin_capability( ixserver_plugins_callbacks *cb, ixserver_plugins_plugin_handler **ret )
{
    s_plugin_handler.process = &json_rpc_plugin_process;
    s_plugin_handler.initialize = &json_rpc_plugin_initialize;
    s_plugin_handler.destroy = &json_rpc_plugin_destroy;
    *ret = &s_plugin_handler;
    s_wss_cb = cb;
}
```

5.4 Initialization and destruction

After the plugin is loaded and the necessary information exchange is completed through ixserver_plugin_capability, the initialize function of the plugin handler is called to initialize the plugin. The plug-in developer can do the basic initialization of the plug-in here.

- ✓ *ixserver_plugins_plugin_handler_initialize*

When using the plugin, it performs a task to cleanup the internal resources of the plugin such as releasing resources before closing the shared object. To do this, IXWSS calls the destroy function before the plugin terminates.

- ✓ *ixserver_plugins_plugin_handler_destroy*

5.5 Data processing and sending data

A process handler is a callback function that receives data passed from a client. The plug-in developer processes the received data in the function and obtains the result.

- ✓ *ixserver_plugins_plugin_handler_process*

Call the this IXWSS callback function to pass the results from the process handler to the client.

- ✓ *ixserver_plugins_callback_send_buffer*

Call the this IXWSS callback function to pass the events from the process handler to the client.

- ✓ *ixserver_plugins_callback_send_event*

5.6 callback functions

In order to be able to transmit events generated by the plugin, a callback function that can poll events generated by the plugin should be registered in the PluginConnector.
When the callback function is registered, the callback function is periodically called to transmit the generated event.

- ✓ *ixserver_plugins_polling_callback*

```
static int json_rpc_plugin_setPollingCallback( ixserver_plugins_object obj )
{
    s_wss_cb->set_polling_cb( obj, json_rpc_plugin_pollingcallback );
    s_polling_callback_initialized = true;
}
```

5.7 ixconfig.conf

```
# IXServer check this file to load plugin when the client request
#
# @ field : plugin_path
# description :
#    set the path of ixplugin library full path
# example >
#    plugin_path=/usr/browser/unittest/ixplugins/bin
#
# @ field : plugin_map
# description :
#    set the port numerber and prototocol and sub-protocol and ixplugin library file
#    that ixs has to load when the web client request. if you don't want to use sub-protocol
#    about some port , please refer to port 9997 in the example below
# example >
# plugin_path=../ixplugins/bin
# plugin_map=9999:http::libixplugin_simple.so
# plugin_map=9999:ws:jsonrpc:libixplugin_jsonrpc.so
# plugin_map=9998:http::libixplugin_jsonrpc.so
# plugin_map=9997:ws::libixplugin_simple.so
#

plugin_path=../ixplugins/bin
plugin_map=9999:http::libixplugin_simple.so
plugin_map=9999:ws:jsonrpc:libixplugin_jsonrpc.so
plugin_map=9998:http::libixplugin_jsonrpc.so
plugin_map=9997:ws::libixplugin_simple.so
```

The process of finding a plug-in is very simple. If the WebSocket Server is listening on a

specific port and a client is connected to the port, the Server will create a Session and pass the necessary plugin path and the callback to receive the event before delivering the data. It finds a shared object to be loaded in the received plugin path, loads it, registers an event callback, and communicates with the client.

5.8 Simple example

The simple example codes are existed in /ixplugins/simple folder.

6 IXPlugin with JSON-RPC

IXWSS can easily extend the interface between client and server by plugin method. IXPlugin is an environment provided for developing plugins directly from client. Various plugins can be made according to requirements. This chapter describes the pre-made JSON-RPC Plugin for customers to refer to when creating various plugins.

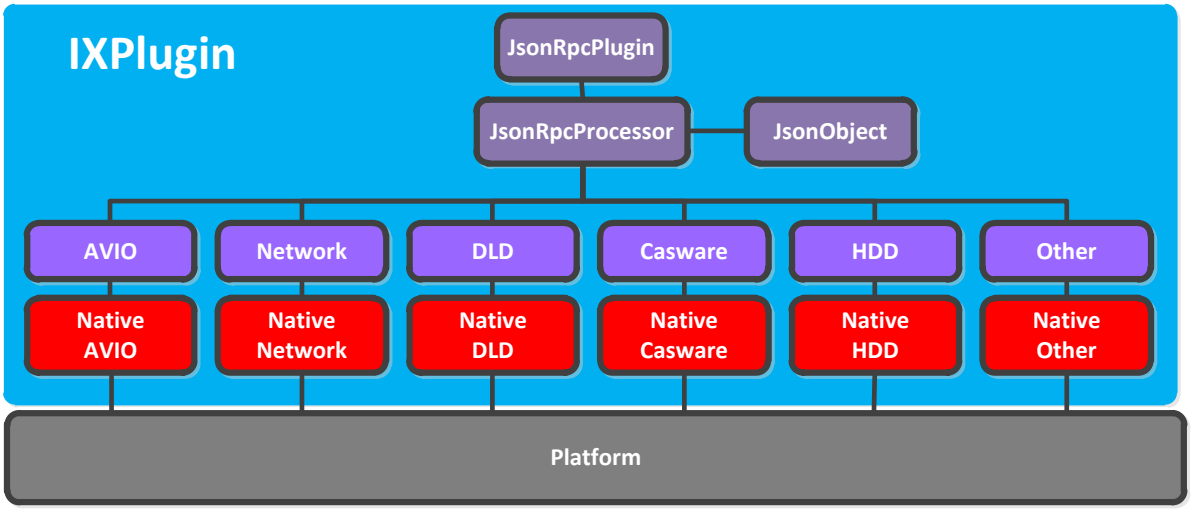


Figure 6. Plugin Structure on JSON-RPC

IXPlugin with JSON-RPC can be represented by the above block. The red highlight is the integration layer to access platform resources.

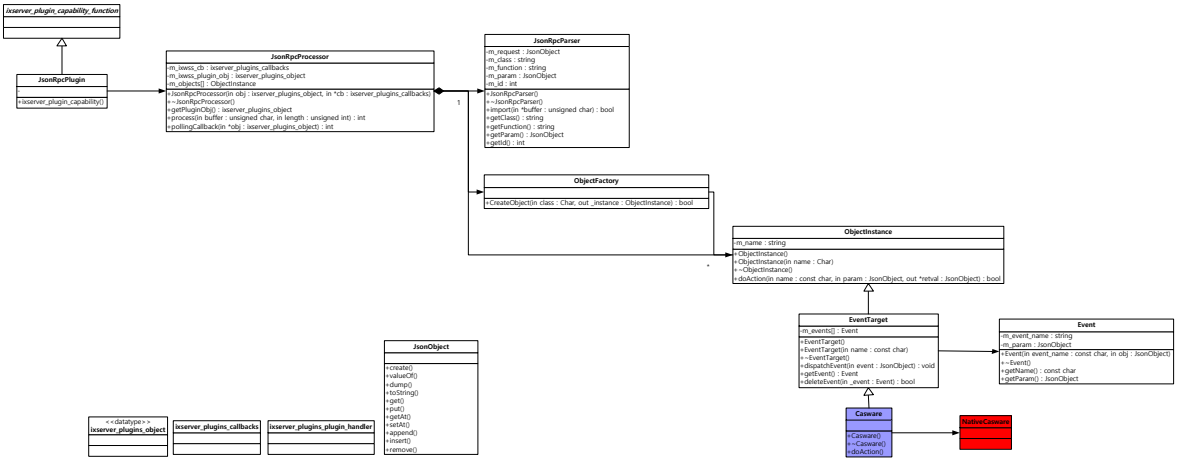


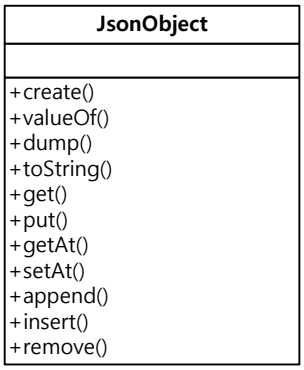
Figure 7. JSON-RPC Plugin Class Diagram

IXPlugin with JSON-RPC consists of the above class modules.

When the plugin is loaded, the ixserver_plugin_Capability () function of JsonRpcPlugin is called to exchange necessary information between the server and the plugin. There is a JsonRpcProcessor class that handles client request and platform events. The json data is managed in the JsonObject class. JsonRpcParser is used to parse json data. The ObjectFactory creates an object to handle client requests. All the generated objects inherit the objectInstance class. Objects that require event processing can manage events by inheriting the EventTarget class.

The classes are described in Sections 6.1 through 6.7.

6.1 JsonObject



It is a data class to manage JSON data.

6.2 JsonRpcPlugin

JsonRpcPlugin
-
+ixserver_plugin_capability()

It is the main class of JSON-RPC Plugin and is responsible for communication with IXWSS. When IXWSS loads this plugin, it calls ixserver_plugin_capability () first.

■ Member Function

Function	ixserver_plugin_capability
Parameter	ixserver_plugins_callbacks *cb, ixserver_plugins_plugin_handler **ret
Return	0/1
Description	When IXWSS loads the plugin, the first function to be called to exchange handlerscb->set_polling_cb : Register polling callback function cb->send_buffer : Pass response data to client cb->send_event : Pass event data to client. ret->initialize : Register plugin initialization function ret->destroy : Register plugin shutdown function ret->process : Register plugin processing function

6.3 JsonRpcProcessor

JsonRpcProcessor
-m_ixwss_cb : ixserver_plugins_callbacks -m_ixwss_plugin_obj : ixserver_plugins_object -m_objects[] : ObjectInstance
+JsonRpcProcessor(in obj : ixserver_plugins_object, in *cb : ixserver_plugins_callbacks) +~JsonRpcProcessor() +getPluginObj() : ixserver_plugins_object +process(in buffer : unsigned char, in length : unsigned int) : int +pollingCallback(in *obj : ixserver_plugins_object) : int

This class handles both the client's request to be sent to the plugin and the response, the event to be delivered to the client.

■ Member Function

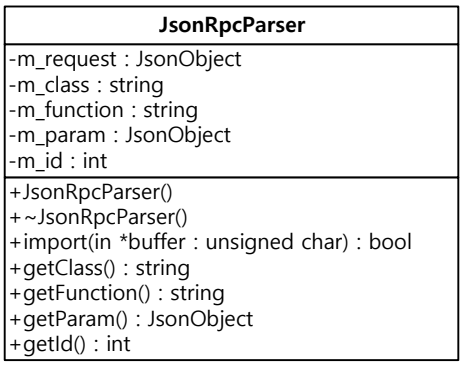
Function	getPluginObj
Return	ixserver_plugin_object

description	Returns the plugin object received from Ixserver.

Function	process
Parameter	unsigned char *buffer, unsigned int length
Return	0/1
Description	It is a function that processes request data transferred from Client.

Function	pollingCallback
Parameter	Ixserver_plugin_object *obj
Return	0/1
description	This function is to be registered as a callback function to be polled, and this function is responsible for sending event data to the client.

6.4 JsonRpcParser



It is a class that parses JSON data delivered from Client according to the specified protocol.

■ Member Function

Function	import
Parameter	unsigned char *buffer
Return	bool

description	Enter the data to be parsed.

Function	getClass
Return	string
description	Returns the class name.

Function	getFunction
Return	string
description	Returns the function name.

Function	getParam
Return	JsonObject
description	Returns the parameter.

Function	getId
Return	Integer
description	Returns the id.

6.5 ObjectFactory

ObjectFactory
+CreateObject(in class : Char, out _instance : ObjectInstance) : bool

This class is responsible for creating the necessary objects.

■ Member Function

Function	CreateObject
Parameter	const char *_method, ObjectInstance *_instance
Return	boolean
Description	Create and return an object that matches the class name.

6.6 ObjectInstance

ObjectInstance
-m_name : string
+ObjectInstance() +ObjectInstance(in name : Char) +~ObjectInstance() +doAction(in name : const char, in param : JsonObject, out *retval : JsonObject) : bool

It is a class that has as a common attribute of objects to be created to perform client's request.

■ Member Function

Function	doAction
Parameter	const char* _name, JsonObject _param, JsonObject *_retval
Return	boolean
Description	As a function to execute the function requested from the client to the module, _name contains the function name that you want to execute. Parameter information is passed to _param. The result of the operation is passed through _retval.

6.7 EventTarget

EventTarget
-m_events[] : Event
+EventTarget() +EventTarget(in name : const char) +~EventTarget() +dispatchEvent(in event : JsonObject) : void +getEvent() : Event +deleteEvent(in _event : Event) : bool

It is a class that is a common attribute of objects to which event data should be transferred to Client.

■ Member Function

Function	dispatchEvent
Parameter	JsonObject event
Return	void
Description	Event data is saved.

Function	getEvent
Return	Event
description	Get event data to be passed to Client.

Function	deleteEvent
Parameter	Event *_event
Return	bool
Description	Delete the event received by the parameter.

6.8 Integration layer

Explains how to use Native interface integration to use platform resourse.

6.8.1 Naming conventions

The name of the Integration layer API consists of the following structure.

native_<InterfaceName>_<FunctionName>
Example: native_network_saveConfig

6.8.2 Return codes

The return code for function execution in the Integration layer is NATIVE_RETURN type. The definition for the type is in NativeTypes.h.

Return Code	Code	Desc.
	NATIVE_RETURN_OK	This is the type that returns when the internal operation of the plugin is completed normally.
	NATIVE_RETURN_ERROR	It is a type that returns when the internal operation of the plugin can not be completed normally.

6.8.3 Data objects

The structure of the data object for exchanging client JSON data with the Platform layer is defined in the Native header. The data object to be returned from the platform will also be defined in Native and the platform layer will maintain the structure that returns the result by filling this value. We designed this structure to make it useful to change

the result object to JsonObject type.

```
typedef struct{
        int id;
        int idInterface;
        char config[20];
        bool enable;
        char routeInterface[20];
        char destination[20];
        char gateway[20];
        int metric;
}NetworkRouteItem;
```

6.8.4 Event handling

Register the callback function from the integration layer to the platform layer to deliver events that occur on the platform. Callback API naming format and listener configuration API example are as follows.

```
native_<InterfaceName>_setListener

//Example:
typedef void (caswareListener)(CaswareEvent evt_type, void* data);
NATIVE_RETURN native_casware_setListener(caswareListener listener);
```

6.8.5 Error handling

NATIVE_RETURN_ERROR is returned if an error occurs in the execution result of the Integration layer.

7 IXWSS & IXPlugin Sequence

7.1 Create IxSession and Plugin loading

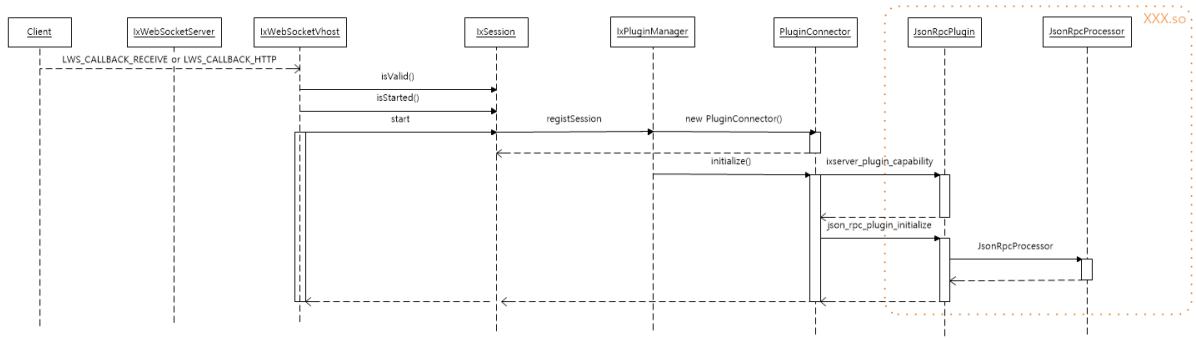


Figure 8. IXWSS sequence diagram - loading

When the client is connected to the server, the plugin is loaded in the following order.

1) IxWebSocketVhost checks the validity and start of Session at the time of receiving data from Client.

2) Session is valid and if it does not start Session is started and register Session through PluginManager, creates PluginConnector, and loads plugin file set through configuration file from PluginConnector.

3) The PluginConnector() calls the ixsever_plugin_capability () function of the loaded plugin.

4) plugin exchanges initialization data in ixserver_plugin_capability () function.

Plugin callback:

- ✓ Register the plugin's polling callback function in PluginConnector.
- ✓ Take the send_buffer function of PluginConnector in plugin.

Plugin handler:

- ✓ Register plugin's initialize function in PluginConnector.
- ✓ Register plugin's process function in PluginConnector.
- ✓ Register plugin's destroy function in PluginConnector.

5) PluginConnector calls the initialize function of loaded plugin.

6) Plugin initialize function creates a processor to handle client request & event and object.

7.2 Client request process

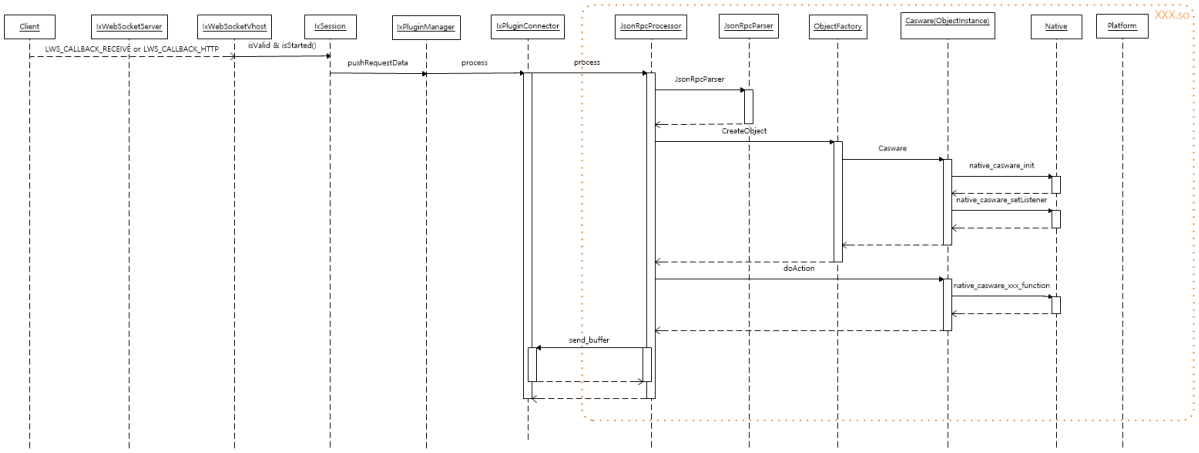


Figure 9. IXWSS sequence diagram - process

When a request is made from a client, it is processed in the following order.

1) The server passes the data passed from the client to the session.

2) Session passes the data through the process function of PluginConnector.

3) PluginConnector calls the process function of the previously registered plugin and passes the data.

4) The processor of the plugin parses the request data using the parser.

5) Get the class name of the request data and request the module to the ObjectFactory.

6) Request function is executed by passing function name and param of request data to doAction () function of Object obtained through ObjectFactory.

7) The return data obtained from the execution result is formatted according to the protocol, passed to the send_buffer callback, and the client receives the result.

7.3 Plugin Event Process

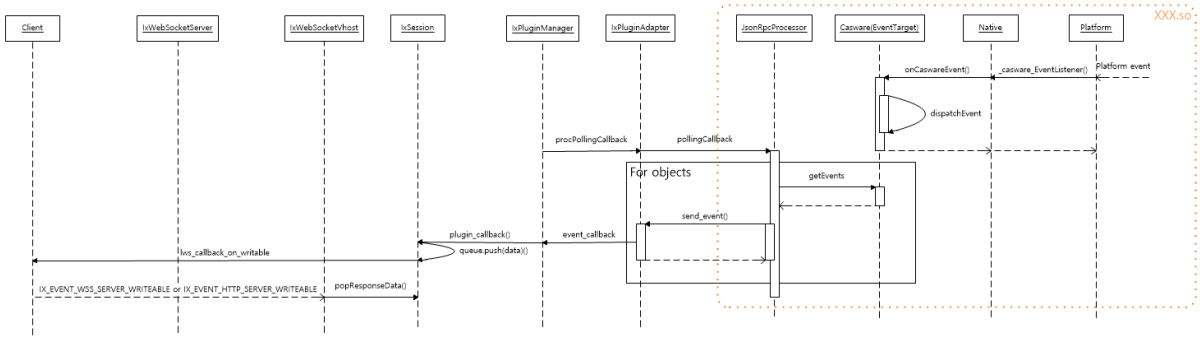


Figure 10. IXWSS sequence diagram – event notify

In order to receive a client event when an event occurs on the platform, the following initialization operation must be performed.

1) Register the event listener from the Native Integration layer to the platform layer.

2) The listener is specified so that the event listener registered in 1) becomes the event listener of OjbectClass.

```
int _casware_EventListener(CaswareEvent evt_type, void* data)
{
    g_CaswareListener(evt_type, data);
}
NATIVE_RETURN native_casware_OnCreate()
{
    Platform_addEventListener(_casware_EventListener)
}
NATIVE_RETURN native_casware_setListener(caswareListener listener)
{
    g_CaswareListener = (caswareListener *)listener;
}
```

When events occur on the platform, they are processed in the following order.

1) The event is delivered to the native event callback registered in the platform.

2) The event is delivered to the event callback of each plugin module registered in the native layer.

3) Put the event in the event queue of the object.

4) PluginConnector is called periodically the plugin's polling callback function.

5) In polling callback function, events accumulated in the objects managed by the processor are delivered to client through send_event function of PluginConnector.

6) The event is stored in the queue of the session and calls the libwebsocket function lws_callback_on_writable.

7) In libwebsocket, LWS_CALLBACK_SERVER_WRITEABLE indicates that the data can be sent, and event is sent to the client by calling lws_write function in session.

8 Environment

8.1 Structure

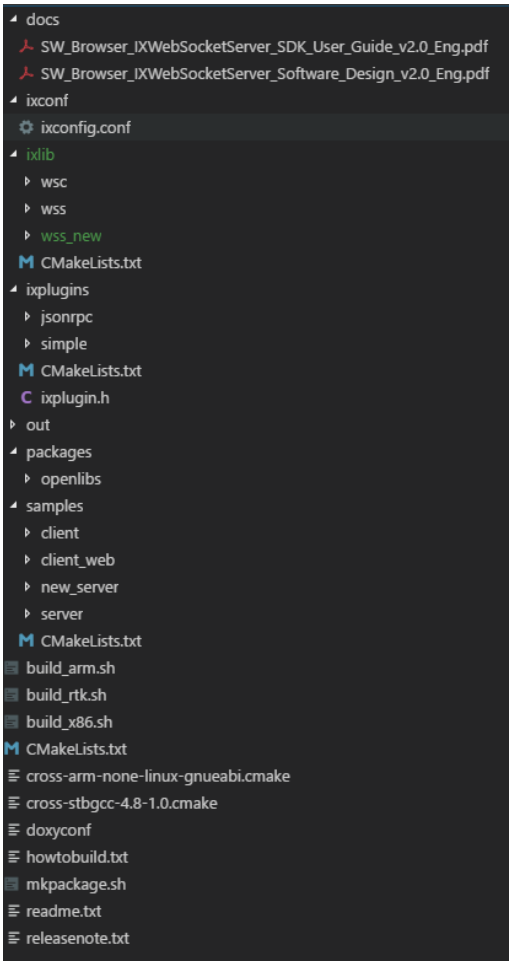


Figure 11. Source Tree

Root	1st Sub	2nd Sub	Desc.
	docs		Software design doc, sdk user guide doc
	ixconf		IXWSS configure file (ixconfig.conf)
	Ixlib	wss	Server source, header files
		wsc	Client source, header files
		wss_new	new server source, header files
	ixplugins	jsonrpc	jsonrpc plugin source, header files
		simple	simple plugin source, header files
	packages	openlibs	open libraries
	sample	server	server sample code

	client	Client sample code
	client_web	web client sample code
build_{xxx}.sh		build script
cross-{xxx}		cmake crosscompile configure
doxyconf		doxygen configure
howtobuild.txt		description of how to build
mkpackage.sh		sdk package script
readme.txt		readme text file
releasenote.txt		History of versions

8.2 How to build IXWSS

- ✓ command: ./build_{xxx}.sh
- ✓ Result : Generate libixserver.so and plugin so files in out / ixwss_sdk folder

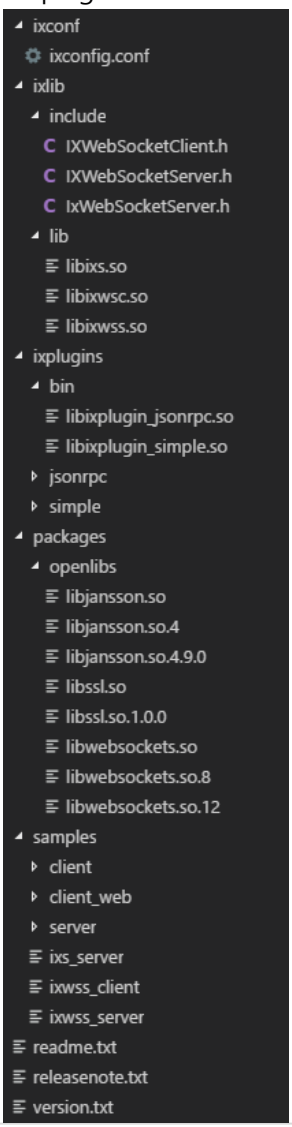


Figure 12. out/ixwss_sdk folder Tree

8.3 How to make IXWSS SDK

- ✓ condition : IXWSS build completed
- ✓ command : ./mkpackage.sh
- ✓ result : Generate ixwss_sdk_{version}_{date}.tar.gz file in out folder.

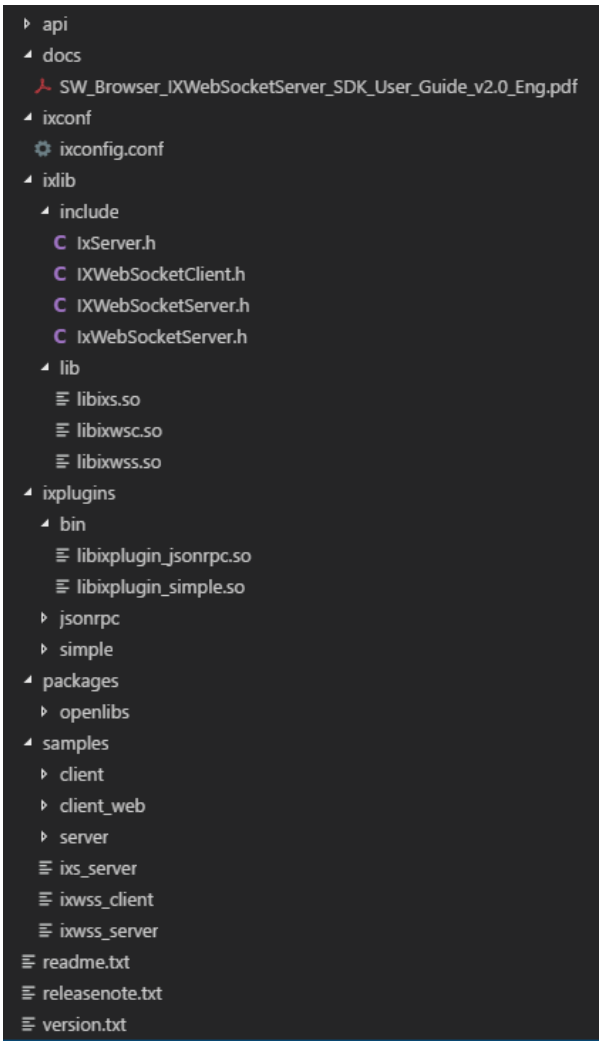


Figure 13. package Tree

8.4 How to test IXWSS on x86

- ✓ command : mkdir out; cd out; cmake ..; make; make install
- ✓ run the server : cd out/ixwss_sdk/sample; ./ixwss_server
- ✓ run the client : cd out/ixwss_sdk/sample; ./ixwss_client
- ✓ run the new server : cd out/ixwss_sdk/sample; ./ixs_server