Integration Service API

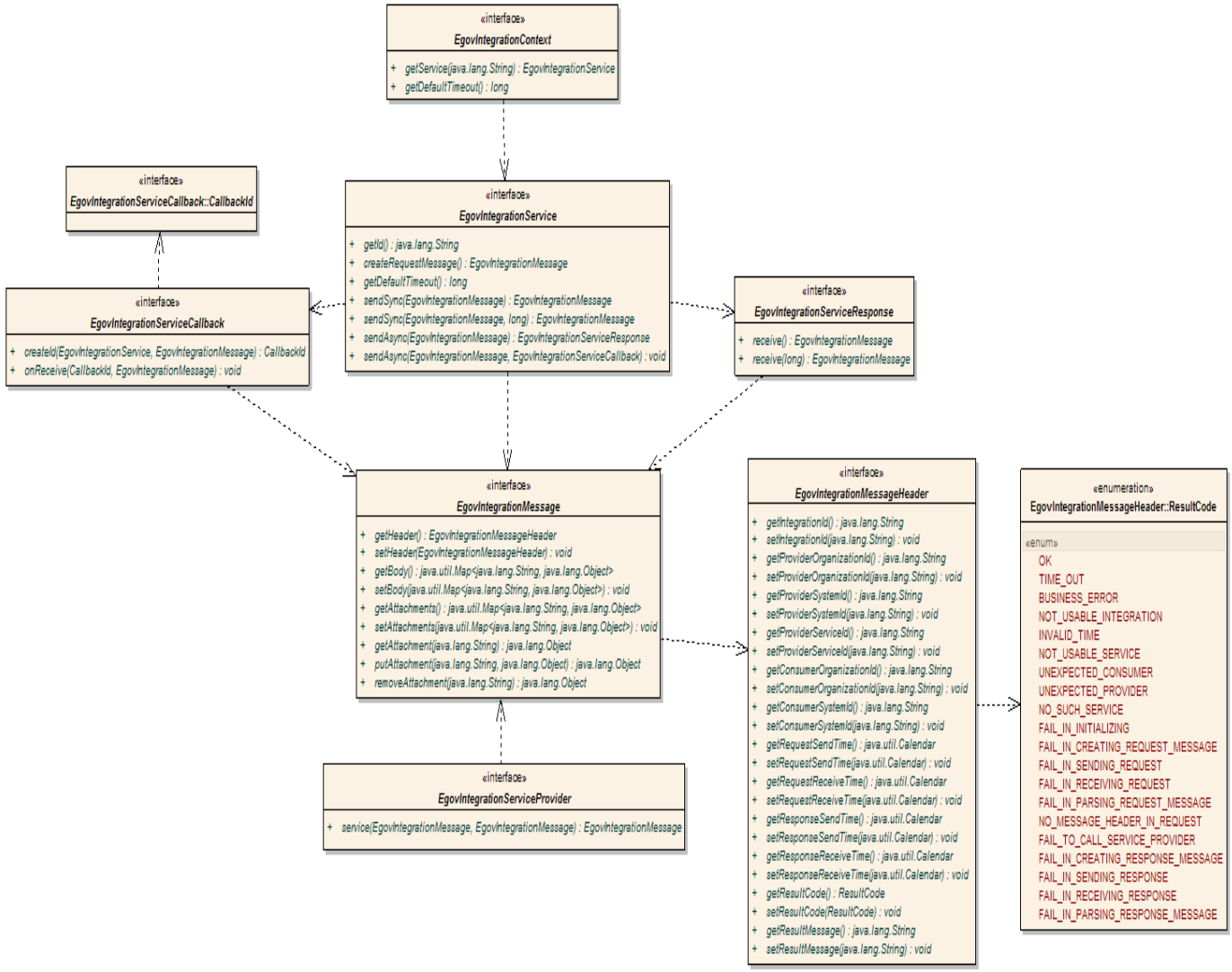
Summary

Integration service API provides an interface to use and provide the connection service.

Description

Composition

Integration service API is composed as following:



Components	Description
EgovIntegrationContext	Manage setting for connected service and EgovIntegrationService object.
EgovIntegrationMessage	Define standard message sent and received through connected service.
EgovIntegrationMessageHeader	Define standard message header sent and received through connected service.
EgovIntegrationMessageHeader::ResultCode	Enumeration containing connected service result code.
EgovIntegrationService	Used to call connected service.
EgovIntegrationResponse	If calling connected service asynchronously, use to receive response message.
EgovIntegrationServiceCallback	If calling connected service asynchronously, Callback

	interface to receive response message
EgovIntegrationServiceCallback::CallbackId	If calling the connected service using Callback asynchronously, interface that indicates the ID for connecting request message and response message.
EgovIntegrationServiceProvider	Used to provide connected service.

EgovIntegrationContext

EgovIntegrationContext manages the configuration of connection service and EgovIntegrationService object. To use the connection service, it should attain EgovIntegrationService object using the getService method of the EgovIntegrationContext.
Following is the example of checking the real name using the resident number and name.

```
package itl.sample;

import javax.annotation.Resource;

import egovframework.rte.itl.integration.EgovIntegrationContext;
import egovframework.rte.itl.integration.EgovIntegrationService;

public class EgovIntegrationSample
{
    @Resource(name = "egovIntegrationContext")
    private EgovIntegrationContext egovIntegrationContext;

    public boolean verifyName(final String name, final String residentRegistrationNumber)
    {
        // Get connected service object with connection ID as "INT_VERIFY_NAME".
        EgovIntegrationService service = egovIntegrationContext.getService("INT_VERIFY_NAME");

        // Create request message
         EgovIntegrationMessage requestMessage = service.createRequestMessage();

        // Create request message
         requestMessage.getBody().put("name", name);
        requestMessage.getBody().put("residentRegistrationNumber", residentRegistrationNumber);

        // Request service
         EgovIntegrationMessage responseMessage = service.sendSync(requestMessage);

        // Return result
        return responseMessage.getBody().get("result");
    }
}
```

Metadata corresponding to example is as shown below.

INTEGRATION						
ID	**PROVIDER_SERVICE_KEY**	**CONSUMER_SYSTEM_KEY**	**DEFAULT_TIMEOUT**	**USING_YN**	**VALIDATE_FROM**	**VALIDATE_TO**
'INT_VERIFY_NAME'	'SERVICE_VERIFY_NAME'	'SYSTEM_CONSUMER'	5000	'Y'	NULL	NULL

ORGANIZATION	
ID	**NAME**
'ORG00001'	'Requesting institution'
'ORG00002'	'Providing institution'

SYSTEM				
SYSTEM_KEY	**ORGANIZATION_ID**	**SYSTEM_ID**	**SYSTEM_NAME**	**STANDARD_YN**

'SYSTEM_CONSUMER'	'ORG00001'	'SYS00001'	'Request system'	'Y'
'SYSTEM_PROVIDER'	'ORG00002'	'SYS00001'	'Response system'	'Y'

SERVICE								
SERVICE_KEY	**SYSTEM_KEY**	**SERVICE_ID**	**SERVICE_NAME**	**REQUEST_MESSAGE_TYPE_ID**	**RESPONSE_MESSAGE_TYPE_ID**	**SERVICE_PROVIDER_BEAN_ID**	**USING_YN**	**STANDARD_YN**
'SERVICE_VERIFY_NAME'	'SYSTEM_PROVIDER'	'SRV00001'	'VerifyName'	'REQ_VERIFY_NAME'	'RES_VERIFY_NAME'	'serviceVerifyName'	'Y'	'Y'

RECORD_TYPE		
RECORD_TYPE_ID	**RECORD_TYPE_NAME**	**PARENT_RECORD_TYPE_ID**
'REQ_VERIFY_NAME'	'RequestVerifyName'	NULL
'RES_VERIFY_NAME'	'ResponseVerifyName'	NULL

RECORD_TYPE_FIELD		
RECORD_TYPE_ID	**RECORD_FIELD_NAME**	**RECORD_FIELD_TYPE_ID**
'REQ_VERIFY_NAME'	'name'	'string'
'REQ_VERIFY_NAME'	'residentRegistrationNumber'	'string'
'RES_VERIFY_NAME'	'result'	'boolean'

EgovIntegrationMessage

EgovIntegrationMessage is composed of a header view, body view and attachment file(s).

Header View

The method to access header view from EgovIntegrationMessage is as following:

Method Summary	
EgovIntegrationMessageHeader	getHeader()
void	setHeader(EgovIntegrationMessageHeader header)

Body View

The method to access body view from EgovIntegrationMessage is as following:

Method Summary	
Map<String, Object>	getBody()
void	setBody(Map<String, Object> body)

The body part of EgovIntegrationMessage can be configured in the following values.

- Wrapper object(Boolean, Byte, Short, Integer, Long, Float, Double) of Java Primitive Type
- BigInteger, BigDecimal
- String
- Calendar
- List<Object>
- Map<String, Object>

Attachment

The method to access attachment file from EgovIntegrationMessage is as following:

Method Summary	
Map<String, Object>	getAttachments()

void	setAttachments(Map<String, Object> attachments)
Object	getAttachment(String name)
void	putAttachment(String name, Object attachment)
Object	removeAttachment(String name)

EgovIntegrationMessageHeader

EgovIntegrationMessageHeader contains below information.

Attribute Name	Data Type	Description
IntegrationId	String	Connection ID
ProviderOrganizationId	String	Connection ID
ProviderSystemId	String	Connection ID
ProviderServiceId	String	Connection ID
ConsumerOrganizationId	String	Connection ID
ConsumerSystemId	String	Connection ID
RequestSendTime	Calendar	Time of sending request
RequestReceiveTime	Calendar	Time of receiving request
ResponseSendTime	Calendar	Time of sending answer
ResponseReceiveTime	Calendar	Time of receiving answer
ResultCode	ResultCode	Result code
ResultMessage	String	Result message

EgovIntegrationMessageHeader defines get/set method for the above attributes.

EgovIntegrationMessageHeader::ResultCode

ResultCode Attribute of EgovIntegrationMessageHeader contains one value in the following connected service result codes.

Code Name	Korean name	Value	Description
OK	Normal termination	"0000"	If connection normally terminates
TIME_OUT	Timeout occurs	"0001"	If Timeout occurs in Client end during connection
BUSINESS_ERROR	Task error occurs	"0002"	If task error occurs in Server end during connection
NOT_USABLE_INTEGRATION	Unused connection	"1000"	If using flag of connection definition(IntegrationDefinition) is false
INVALID_TIME	Not connection available time	"1001"	If the timing requesting connection is not between validateFrom and validateTo of connection definition(IntegrationDefinition) • Connection available condition = (validateFrom == null || validateFrom.compareTo(now) <= 0) && (validateTo == null || now.compareTo(validateTo) <= 0) * now : Calendaer = current time
NOT_USABLE_SERVICE	Unused	"1002"	If the using flag value of provision

	service		service(ServiceDefinition) registered in connection definition (IntegrationDefinition) is false
UNEXPECTED_CONSUMER	Unexpected connection requestor	"1003"	If the request system code value of connection definition(IntegrationDefinition) registered in connection provision system(Server) does not match the request system code value of request message header
UNEXPECTED_PROVIDER	Unexpected connection requestor	"1004"	If the system code registered in the connection provision system(Server) does not match the available system code value of request message header
NO_SUCH_SERVICE	Unavailable service	"1005"	If requesting the service not registered in connection provision system(Server)
FAIL_IN_INITIALIZING	Failed to initialize connected service	"1006"	If failed to initialize connection provision service
FAIL_IN_CREATING_REQUEST_MESSAGE	Failed to create request message	"2000"	If an error occurs when creating request message from client
FAIL_IN_SENDING_REQUEST	Failed to send request message	"2001"	If an error occurs when transmitting request message from client
FAIL_IN_RECEIVING_REQUEST	Failed to receive request message	"3000"	If an error occurs when receiving request message from server
FAIL_IN_PARSING_REQUEST_MESSAGE	Failed to analyze request message	"3001"	If an error occurs when analyzing request message from server
NO_MESSAGE_HEADER_IN_REQUEST	Lack of request message header	"3002"	IF standard message header does not exist in request message received from the server
FAIL_TO_CALL_SERVICE_PROVIDER	Failed to call available service modules	"3003"	If an error occurs when calling the service provision module from the server
FAIL_IN_CREATING_RESPONSE_MESSAGE	Failed to create response message	"4000"	If an error occurs when creating response message from server
FAIL_IN_SENDING_RESPONSE	Failed to send response message	"4001"	If an error occurs when transmitting response message from server
FAIL_IN_RECEIVING_RESPONSE	Failed to receive response message	"5000"	If an error occurs when receiving response message from client
FAIL_IN_PARSING_RESPONSE_MESSAGE	Failed to	"5001"	If an error occurs when analyzing

	analyze response message		response message from client

EgovIntegrationService

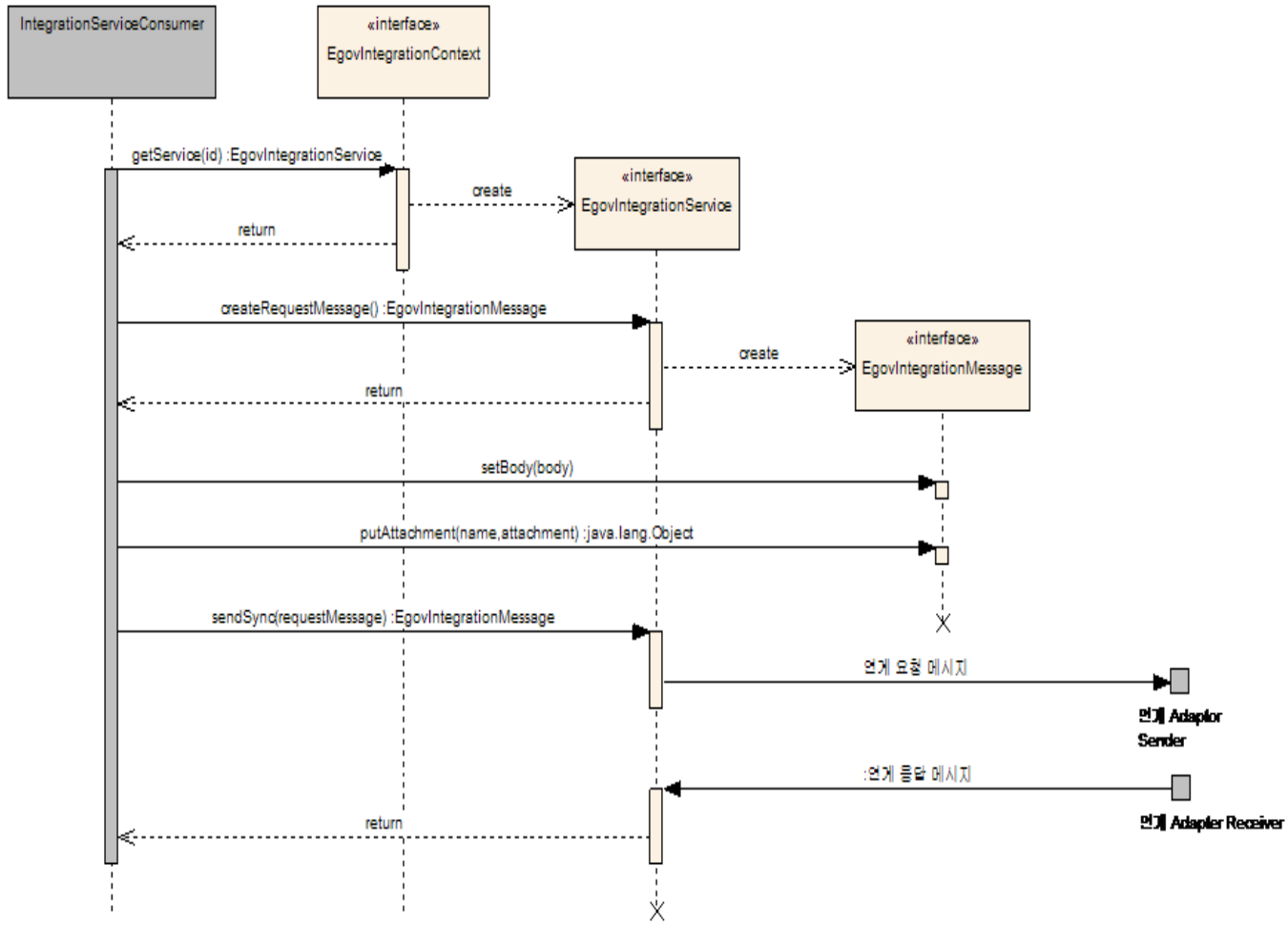
EgovIntegrationService supports the synchronized calling and non-synchronized calling.

- Synchronized method
- Asynchronized method

Synchronized method

sendSync method of EgovIntegrationService calls connected service in synchronization type.

...




```
public class EgovIntegrationSample
{
    ...

    public boolean verifyName(final String name, final String residentRegistrationNumber)
    {
        // Create and write request message after obtaining EgovIntegrationService object from
EgovIntegrationContext.
        ...

        // Call connected service synchronously (timeout = 5000 millisecond)
        EgovIntegrationMessage responseMessage = service.sendSync(requestMessage, 5000);
```

```
        // Response result processing
        ...
    }

    ...
}
```

If using a default timeout value registered in connection registration information of Metadata or EgovIntegrationContext, the timeout value can be omitted.

```
        ...

        EgovIntegrationMessage responseMessage = servicd.sendSync(requestMessage);

        ...
```

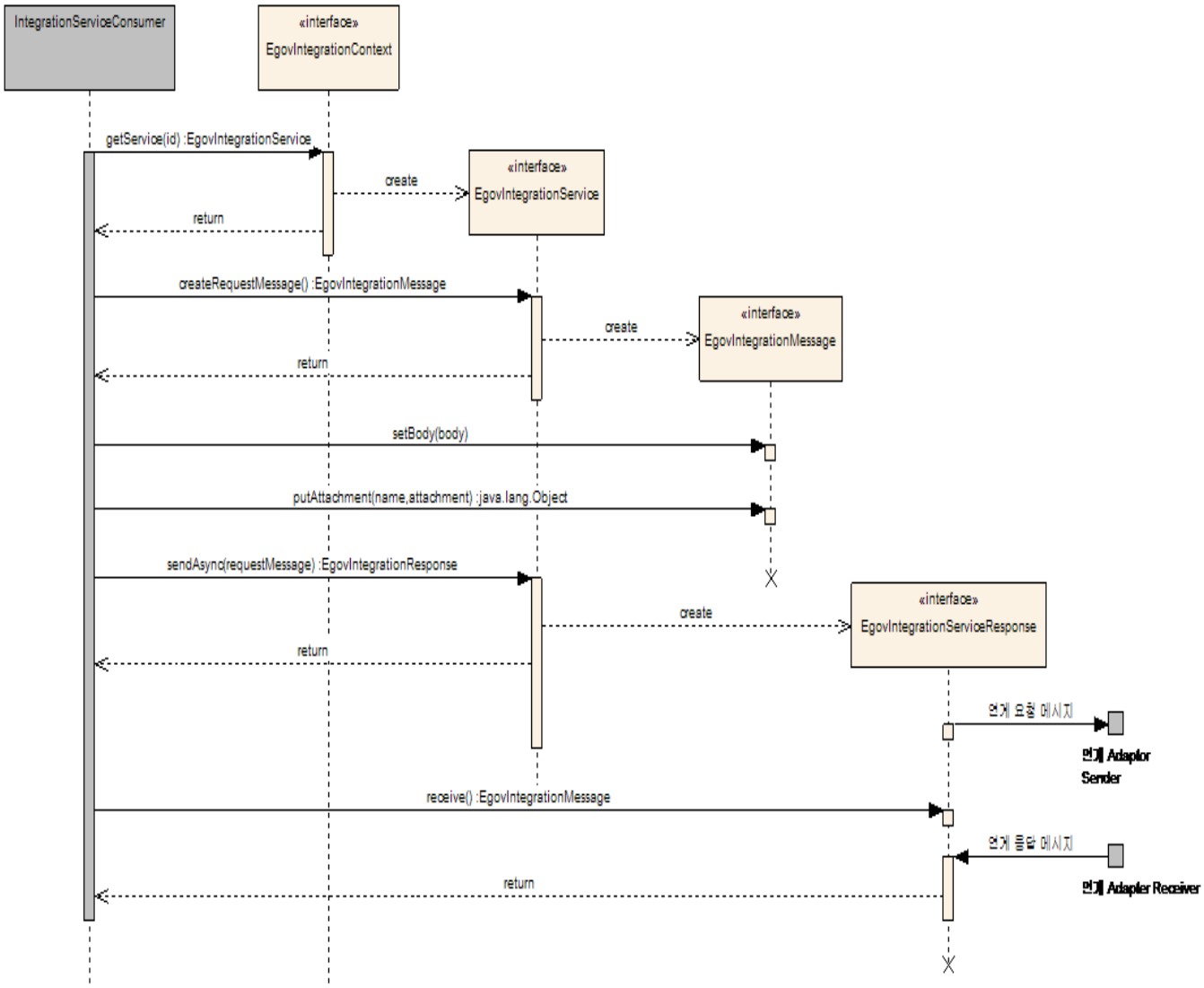
Asynch method

The sendAsync method of the EgovIntegrationService calls the connection service using asynch metthod. There are two sendAsync methods.

- Using sendAsync with Response
- Using sendAsync with Callback

Using sendAsync with Response

The following appendix describes the meta-data tables in detail, along with many of the design decisions that were made when creating them. When viewing the various table creation statements below, it is important to realize that the data types used are as generic as possible. Spring Batch provides many schemas as examples, which all have varying data types due to variations in individual database vendors' handling of data types.



...

```
public class EgovIntegrationSample
{
    ...

    public boolean verifyName(final String name, final String residentRegistrationNumber)
    {
        // After obtaining EgovIntegrationService object from EgovIntegrationContext, create and write request message.
        ...

        // call connected service asynchronously
        EgovIntegrationServiceResponse response = service.sendAsync(requestMessage);

        // Perform required tasks before receiving response message using response object
         ...

        // receive response message using response object(timeout = 5000 millisecond)
         EgovIntegrationMessage responseMessage = response.receive(5000);

        // response message processing
        ...
    }

    ...
}
```

If using default timeout value registered in connection registration information of Metadata or EgovIntegrationContext, timeout value can be omitted.

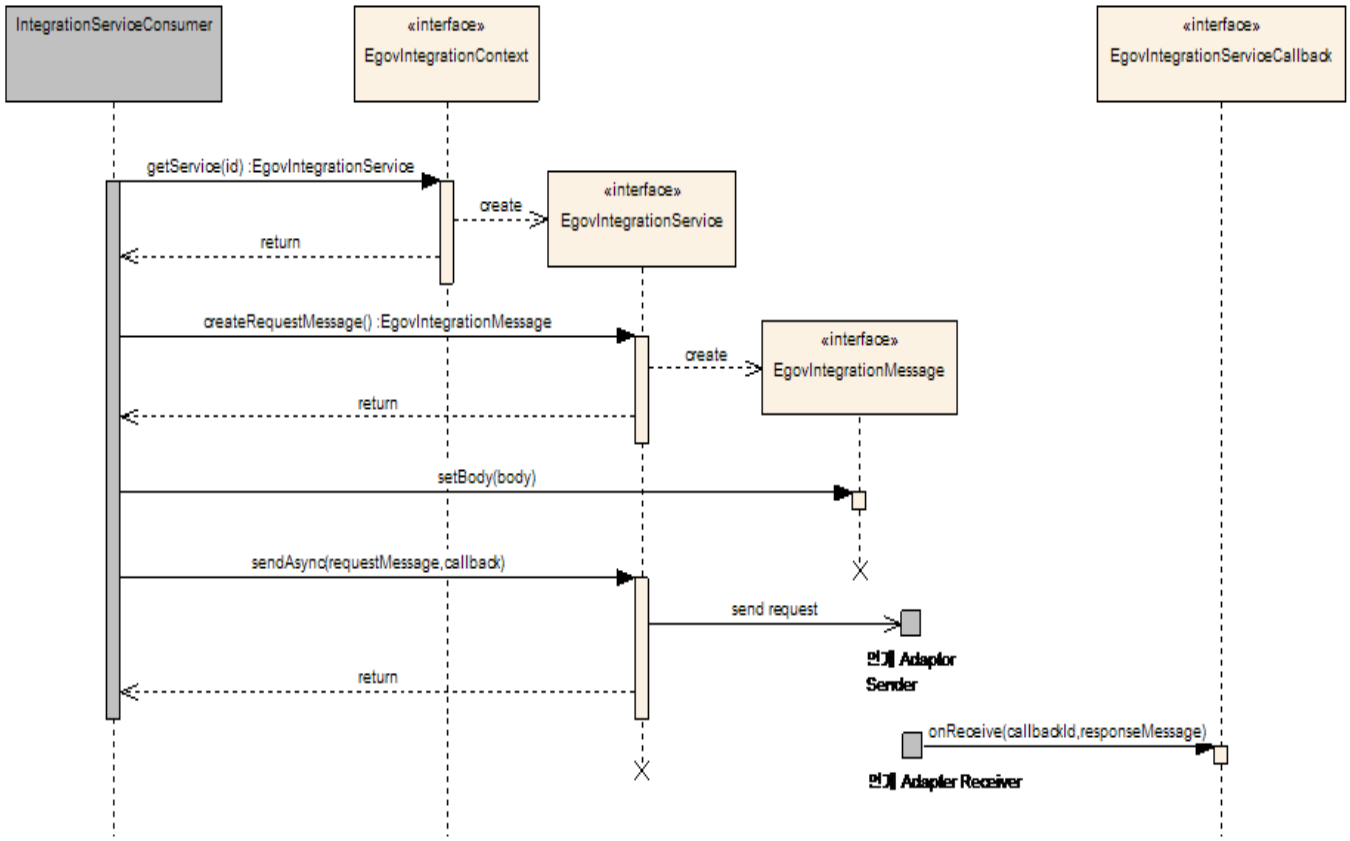
```
        ...

        // receive response message using response object
        EgovIntegrationMessage responseMessage = response.receive();

        ...
```

Using sendAsync with Callback

Asynch calling method using EgovIntegrationServiceCallback. Asynchronous calling of Callback type is used when it is okay to entrust the processing for response to Callback object while task module requesting connected service performs requests only.




```
...

public class EgovIntegrationSample
{
    ...

    @Resource(name = "verifyNameServiceCallback")
    private EgovIntegrationServiceCallback callback;

    public boolean verifyName(final String name, final String residentRegistrationNumber)
    {
        // After obtaining EgovIntegrationService oject from EgovIntegrationContext, create and write request message.
        ...

        // call connected service asynchronously
         service.sendSync(requestMessage, callback);
    }
```

```
    ...
}
package itl.sample;

import egovintegration.rte.itl.integration.EgovIntegrationServiceCallback;
import egovintegration.rte.itl.integration.EgovIntegrationServiceCallback.CallbackId;

public class VefiryNameServiceCallback
{
    public CallbackId createId(EgovIntegrationService service, EgovIntegrationMessage
requestMessage)
    {
        // This method is called in connection adaptor or solution implementing EgovIntegrationService.
        // Create and return CallbackId using service and request message.
        // Created CallbackId is used to identify relevant service and request message when receiving
response message.

        // Create CallbackId
        CallbackId callbadkId = ...

        return callbackId;
    }

    public vod onReceive(CallbackId callbackId, EgovIntegrationMessage responseMessage)
    {
        // This method is called when the response message for processing arrives or is called by
connection Adaptor or solution.

        // response message processing
         ...
    }
}
```

EgovIntegrationServiceProvider

EgovIntegrationServiceProvider interface is the interface to provide the connection service which the model that provides the service should be implemented through this interface.
Following example is the job module providing the identification of real name using name and resident registration number as well as Spring Framework Configuration XML. (Metadata is same as the setting of EgovIntegrationContext example.)

```
package itl.sample;

import egovframework.rte.itl.integration.EgovIntegrationMessage;
import egovframework.rte.itl.integration.EgovIntegrationServiceProvider;

public class ServiceVerifyName implements EgovIntegrationServiceProvider
{
    public void service(EgovIntegrationMessage requestMessage, EgovIntegrationMessage
responseMessage)
    {
        String name = requestMessage.getBody().get("name");
        String residentRegistrationNumber =
requestMessage.getBody().get("residentRegistrationNumber");

        // Check real name
         boolean result = varifyName(name, residentRegistrationNumber);

        responseMessage.getBody().put("result", result);
    }
}
```

```
...
<bean id="serviceVerifyName" class="itl.sample.ServiceVerifyName"/>
...
```

Reference

N/A