OIO Service Oriented Infrastructure

OIOSI RASP Library for .NET v 1.01

Component overview

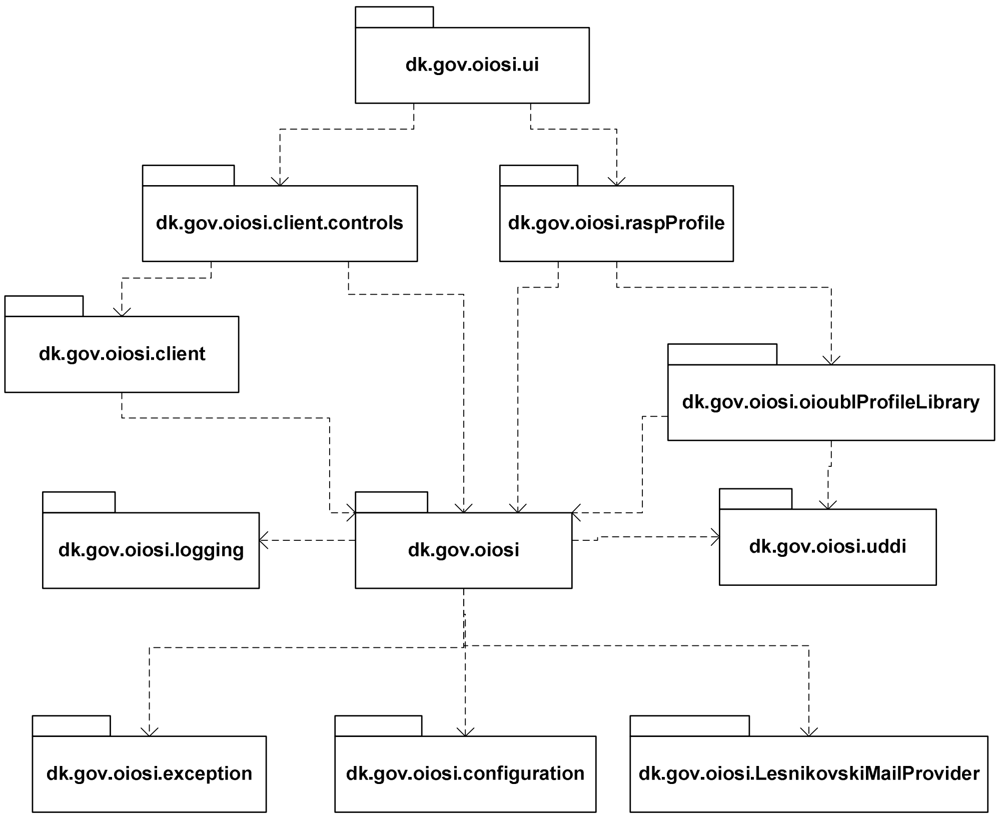
Contents

1 Introduction

The purpose of this document is to describe the main components of the RASP software library for .Net RC1 release. Only the most important classes are presented, to give an overview of the various elements.

2 Overview

Below is an overview of the library structure. ”dk.gov.oiosi” is the RASP library. The diagram includes both library and UI packages.



The main component, **the OIOSI RASP software library** contains the following elements:

3 Release Pack

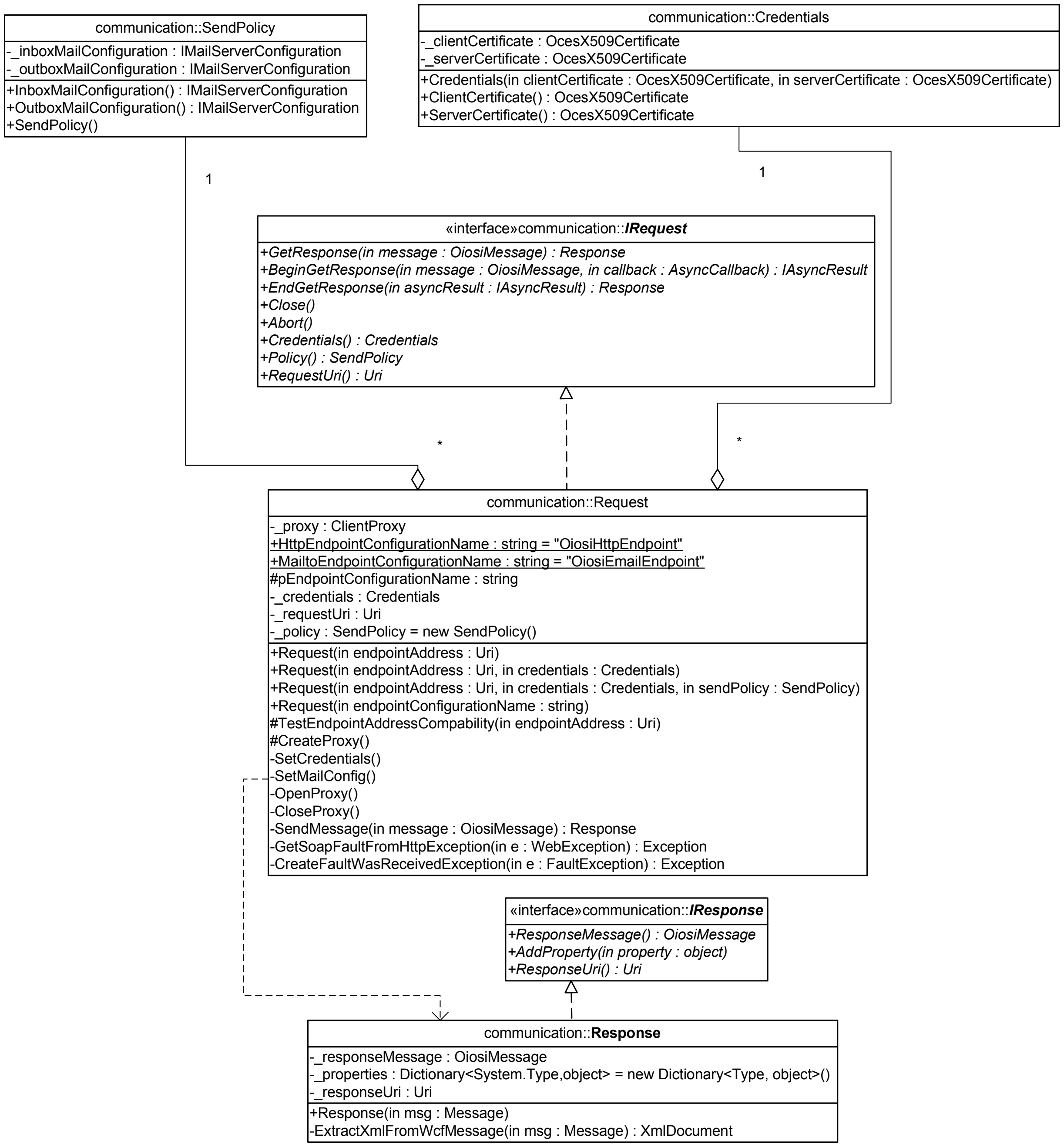
This release pack includes two visual studio solutions, one for RASP Library functionality, and one for RASP client UI code. Both solutions include the RASP library source code. The library functionality will be discussed further in later sections of this document.

4 Components

This section describes each of the high level interfaces to the RASP library.

4.1 Communication, client side

The main client side class is the Request, which enables sending functionality. Request can either be initialized with an URI and some credentials or with the reference to a client endpoint in the configuration file.

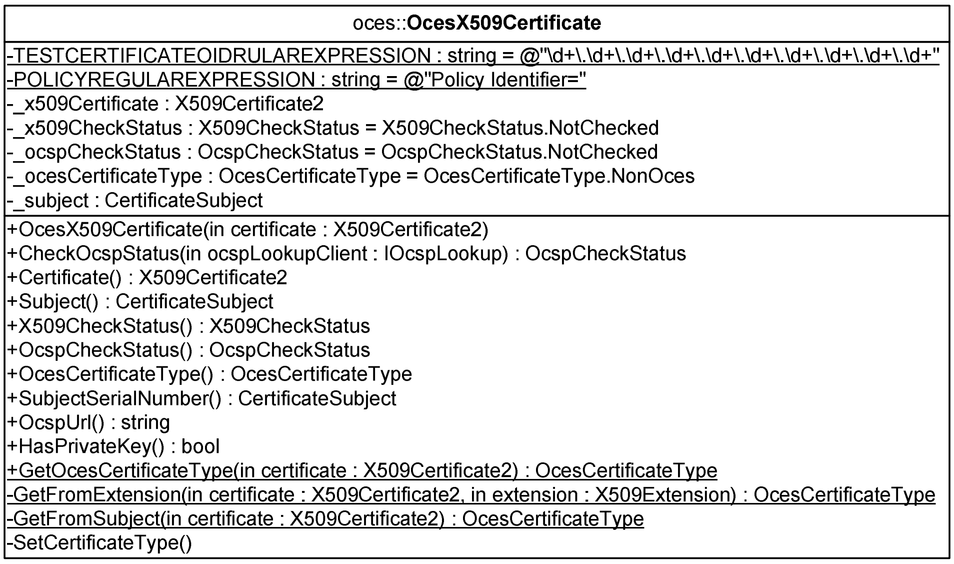

communication::SendPolicy
-_inboxMailConfiguration : IMailServerConfiguration
-_outboxMailConfiguration : IMailServerConfiguration
+InboxMailConfiguration() : IMailServerConfiguration
+OutboxMailConfiguration() : IMailServerConfiguration
+SendPolicy()
communication::Credentials
-_clientCertificate : OcesX509Certificate
-_serverCertificate : OcesX509Certificate
+Credentials(in clientCertificate : OcesX509Certificate, in serverCertificate : OcesX509Certificate)
+ClientCertificate() : OcesX509Certificate
+ServerCertificate() : OcesX509Certificate
1
1
«interface»communication::*IRequest*
+GetResponse(in message : OiosiMessage) : Response
+BeginGetResponse(in message : OiosiMessage, in callback : AsyncCallback) : IAsyncResult
+EndGetResponse(in asyncResult : IAsyncResult) : Response
+Close()
+Abort()
+Credentials() : Credentials
+Policy() : SendPolicy
+RequestUri() : Uri
*
*
communication::Request
-_proxy : ClientProxy
+HttpEndpointConfigurationName : string = "OiosiHttpEndpoint"
+MailtoEndpointConfigurationName : string = "OiosiEmailEndpoint"
#pEndpointConfigurationName : string
-_credentials : Credentials
-_requestUri : Uri
-_policy : SendPolicy = new SendPolicy()
+Request(in endpointAddress : Uri)
+Request(in endpointAddress : Uri, in credentials : Credentials)
+Request(in endpointAddress : Uri, in credentials : Credentials, in sendPolicy : SendPolicy)
+Request(in endpointConfigurationName : string)
#TestEndpointAddressCompability(in endpointAddress : Uri)
#CreateProxy()
-SetCredentials()
-SetMailConfig()
-OpenProxy()
-CloseProxy()
-SendMessage(in message : OiosiMessage) : Response
-GetSoapFaultFromHttpException(in e : WebException) : Exception
-CreateFaultWasReceivedException(in e : FaultException) : Exception
«interface»communication::***IResponse***
+ResponseMessage() : OiosiMessage
+AddProperty(in property : object)
+ResponseUri() : Uri
communication::**Response**
-_responseMessage : OiosiMessage
-_properties : Dictionary<System.Type,object> = new Dictionary<Type, object>()
-_responseUri : Uri
+Response(in msg : Message)
-ExtractXmlFromWcfMessage(in msg : Message) : XmlDocument

If initialized with *Credentials*, two certificates should be given. One for client side encryption and one for validating server side responses.

Request.GetResponse takes a OiosiMessage, which basically is an XML document with metadata attached to it. In return a Response is given, which again is an XML document, the one returned from the called service, coupled with metadata. Metadata can be added to the object using the *AddProperty* method. This collection also holds custom properties added to the message on its way through the WCF stack, i.e. signature validation proof object set by the signature proof interceptor.

4.2 Certificate handling & LDAP utilities

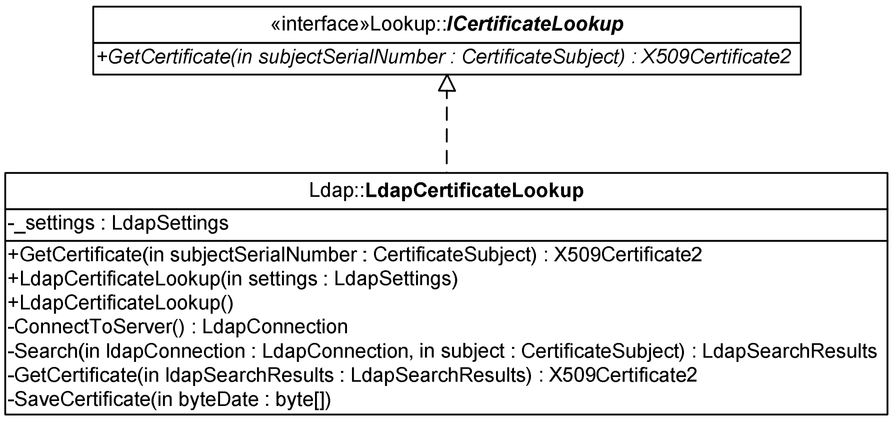
These components are used to validate the client certificates. These are represented by the OcesX509Certificate. The class is also responsible for checking the certificates expiration date, activation date, chain checks etc.



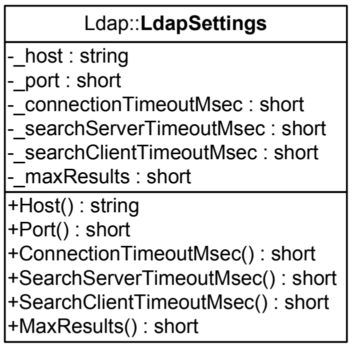
```
oces::OcesX509Certificate
-TESTCERTIFICATEOIDRULAREXPRESSION : string = @"\d+\.\d+\.\d+\.\d+\.\d+\.\d+\.\d+\.\d+\.\d+\.\d+"
-POLICYREGULAREXPRESSION : string = @"Policy Identifier="
-_x509Certificate : X509Certificate2
-_x509CheckStatus : X509CheckStatus = X509CheckStatus.NotChecked
-_ocspCheckStatus : OcspCheckStatus = OcspCheckStatus.NotChecked
-_ocesCertificateType : OcesCertificateType = OcesCertificateType.NonOces
-_subject : CertificateSubject
+OcesX509Certificate(in certificate : X509Certificate2)
+CheckOcspStatus(in ocspLookupClient : IOcspLookup) : OcspCheckStatus
+Certificate() : X509Certificate2
+Subject() : CertificateSubject
+X509CheckStatus() : X509CheckStatus
+OcspCheckStatus() : OcspCheckStatus
+OcesCertificateType() : OcesCertificateType
+SubjectSerialNumber() : CertificateSubject
+OcspUrl() : string
+HasPrivateKey() : bool
+GetOcesCertificateType(in certificate : X509Certificate2) : OcesCertificateType
-GetFromExtension(in certificate : X509Certificate2, in extension : X509Extension) : OcesCertificateType
-GetFromSubject(in certificate : X509Certificate2) : OcesCertificateType
-SetCertificateType()
```

This class has various subclass wich add additional properties, e.g. for getting a CVR number from an employee certificate.

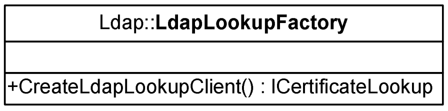
The Ldap lookup uses the ldap protocol to query for a given certificate. It is also responsible for opening and closing connections to the ldap server.


«interface»Lookup::ICertificateLookup
+GetCertificate(in subjectSerialNumber : CertificateSubject) : X509Certificate2
Ldap::LdapCertificateLookup
-_settings : LdapSettings
+GetCertificate(in subjectSerialNumber : CertificateSubject) : X509Certificate2
+LdapCertificateLookup(in settings : LdapSettings)
+LdapCertificateLookup()
-ConnectToServer() : LdapConnection
-Search(in ldapConnection : LdapConnection, in subject : CertificateSubject) : LdapSearchResults
-GetCertificate(in ldapSearchResults : LdapSearchResults) : X509Certificate2
-SaveCertificate(in byteDate : byte[])

The LdapCertificateLookup constructor takes an ldap settings object as a parameter. The setting object is used every time a lookup is performed in the GetCertificate function.


Ldap::LdapSettings
-_host : string
-_port : short
-_connectionTimeoutMsec : short
-_searchServerTimeoutMsec : short
-_searchClientTimeoutMsec : short
-_maxResults : short
+Host() : string
+Port() : short
+ConnectionTimeoutMsec() : short
+SearchServerTimeoutMsec() : short
+SearchClientTimeoutMsec() : short
+MaxResults() : short

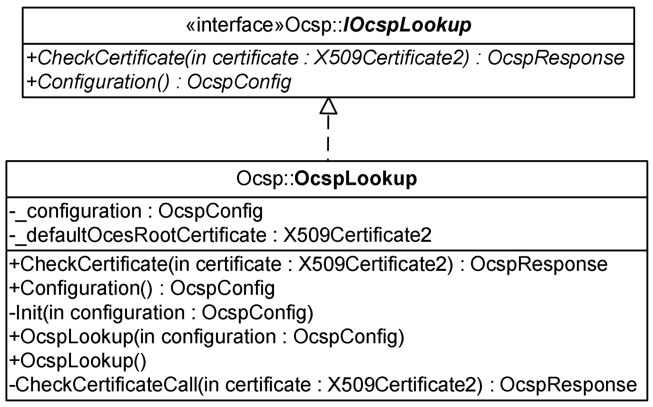
The LdapLookupFactory instantiates classes with the ICertificateLookup interface based on the configuration.



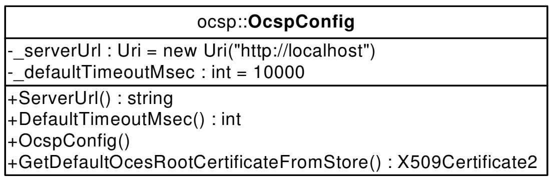
The factory could instantiate either *LdapCertificateLookup* or *LdapCertificateLookupTest*, or you may create your own implementation. You can change this behavior in the *RaspConfiguration.xml* file, in the section **<LdapLookupFactoryConfig>** element by setting the namespace and assembly name of the ICertificateLookup implementation to instantiate. Developers may add their own implementation libraries and instantiate these just by changing the configuration.

4.3 OCSP

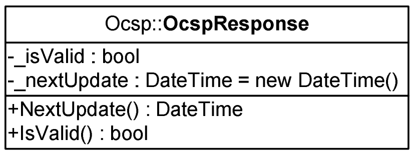
OCSP lookup component is used to check certificate revocation status against an OCSP server.


«interface»Ocsp::*IOcspLookup*
+CheckCertificate(in certificate : X509Certificate2) : OcspResponse
+Configuration() : OcspConfig
Ocsp::**OcspLookup**
-_configuration : OcspConfig
-_defaultOcesRootCertificate : X509Certificate2
+CheckCertificate(in certificate : X509Certificate2) : OcspResponse
+Configuration() : OcspConfig
-Init(in configuration : OcspConfig)
+OcspLookup(in configuration : OcspConfig)
+OcspLookup()
-CheckCertificateCall(in certificate : X509Certificate2) : OcspResponse

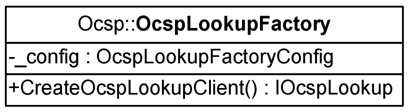
The OcspConfig is used to dynamically set the URL to the OCSP server, timeout, various default settings etc.


ocsp::**OcspConfig**
-_serverUrl : Uri = new Uri("http://localhost")
-_defaultTimeoutMsec : int = 10000
+ServerUrl() : string
+DefaultTimeoutMsec() : int
+OcspConfig()
+GetDefaultOcesRootCertificateFromStore() : X509Certificate2

The response from the lookup contains an isValid boolean, which indicates if the certificate has been revoked, either temporarily or permanently. The response also contains a nextUpdate value, which is used to store the time when newer information will be available, but this has not been implemented for this preview. This is however not implemented with the underlying library that this release implements.



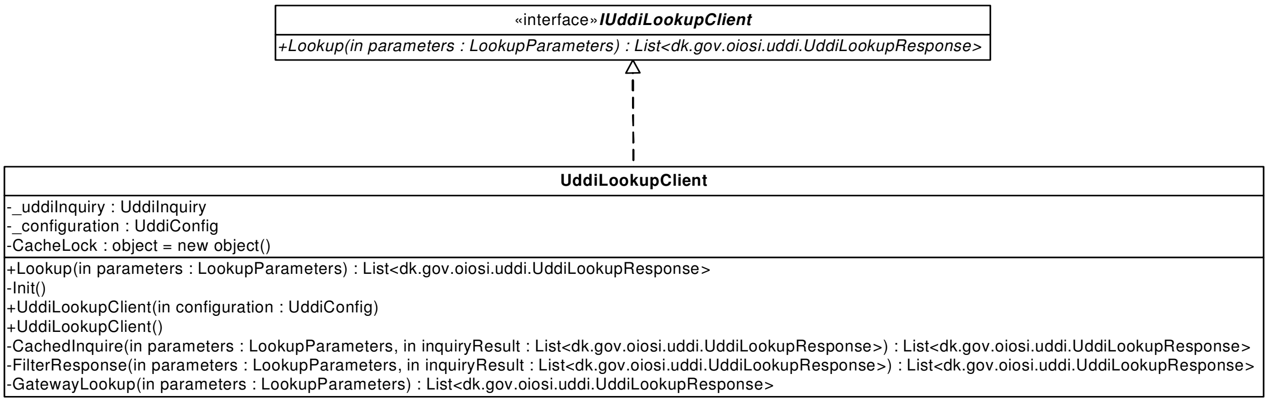
OcspLookupFactory creates an instance of a class with the IOscspLookup interface.


Ocsp::**OcspLookupFactory**
-_config : OcspLookupFactoryConfig
+CreateOcspLookupClient() : IOcspLookup

The factory could instantiate either *OcspLookup* or *OcspLookupTest*, or you can derive your own implementation. You can change this behavior in the *RaspConfiguration.xml* file, in the section **<OcspLookupFactoryConfig>** element by setting the namespace and assembly name of the IOcspLookup implementation to instantiate. Developers may add their own implementation libraries and instantiate these just by changing the configuration.

4.4 UDDI

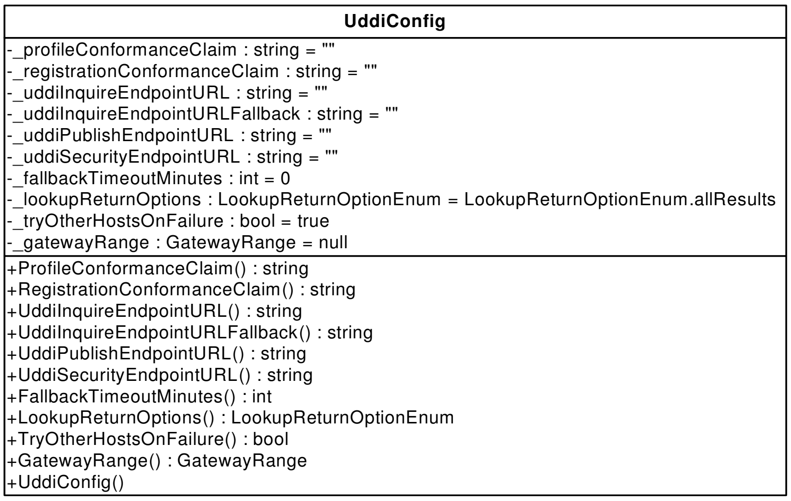
The UDDI is used in the ARS (Address Resolving Service), which is used to resolve business level endpoints to service endpoints. Below you can see the main UDDI classes for lookup.


«interface»IUddiLookupClient
+Lookup(in parameters : LookupParameters) : List<dk.gov.oiosi.uddi.UddiLookupResponse>
UddiLookupClient
-_uddiInquiry : UddiInquiry
-_configuration : UddiConfig
-CacheLock : object = new object()
+Lookup(in parameters : LookupParameters) : List<dk.gov.oiosi.uddi.UddiLookupResponse>
-Init()
+UddiLookupClient(in configuration : UddiConfig)
+UddiLookupClient()
-CachedInquire(in parameters : LookupParameters, in inquiryResult : List<dk.gov.oiosi.uddi.UddiLookupResponse>) : List<dk.gov.oiosi.uddi.UddiLookupResponse>
-FilterResponse(in parameters : LookupParameters, in inquiryResult : List<dk.gov.oiosi.uddi.UddiLookupResponse>) : List<dk.gov.oiosi.uddi.UddiLookupResponse>
-GatewayLookup(in parameters : LookupParameters) : List<dk.gov.oiosi.uddi.UddiLookupResponse>

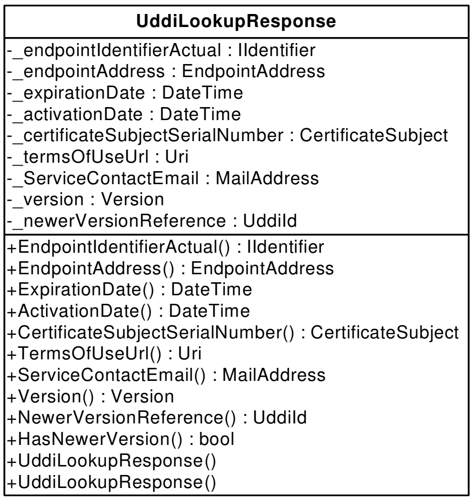
The lookup is based on an endpoint key, like EAN number, and other lookup options like address type etc. You can for example set a process definition (such as an OIOUBL profile definition) as a filter, or the role that a

The component can also handle automatic caching of endpoints of resolving attempts and supports caching policies based on the call result feedback. You can also set a fallback address, so it uses a secondary UDDI registry in case the primary is unavailable.

The client takes a configuration object, that sets attributes like UDDI endpoint url, return options, timeouts etc.

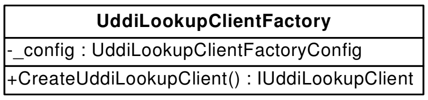

UddiConfig
-_profileConformanceClaim : string = ""
-_registrationConformanceClaim : string = ""
-_uddiInquireEndpointURL : string = ""
-_uddiInquireEndpointURLFallback : string = ""
-_uddiPublishEndpointURL : string = ""
-_uddiSecurityEndpointURL : string = ""
-_fallbackTimeoutMinutes : int = 0
-_lookupReturnOptions : LookupReturnOptionEnum = LookupReturnOptionEnum.allResults
-_tryOtherHostsOnFailure : bool = true
-_gatewayRange : GatewayRange = null
+ProfileConformanceClaim() : string
+RegistrationConformanceClaim() : string
+UddiInquireEndpointURL() : string
+UddiInquireEndpointURLFallback() : string
+UddiPublishEndpointURL() : string
+UddiSecurityEndpointURL() : string
+FallbackTimeoutMinutes() : int
+LookupReturnOptions() : LookupReturnOptionEnum
+TryOtherHostsOnFailure() : bool
+GatewayRange() : GatewayRange
+UddiConfig()

The response from the lookup contains the endpoint address and other parameters.



```
UddiLookupResponse
-_endpointIdentifierActual : IIdentifier
-_endpointAddress : EndpointAddress
-_expirationDate : DateTime
-_activationDate : DateTime
-_certificateSubjectSerialNumber : CertificateSubject
-_termsOfUseUrl : Uri
-_ServiceContactEmail : MailAddress
-_version : Version
-_newerVersionReference : UddiId
+EndpointIdentifierActual() : IIdentifier
+EndpointAddress() : EndpointAddress
+ExpirationDate() : DateTime
+ActivationDate() : DateTime
+CertificateSubjectSerialNumber() : CertificateSubject
+TermsOfUseUrl() : Uri
+ServiceContactEmail() : MailAddress
+Version() : Version
+NewerVersionReference() : UddiId
+HasNewerVersion() : bool
+UddiLookupResponse()
+UddiLookupResponse()
```

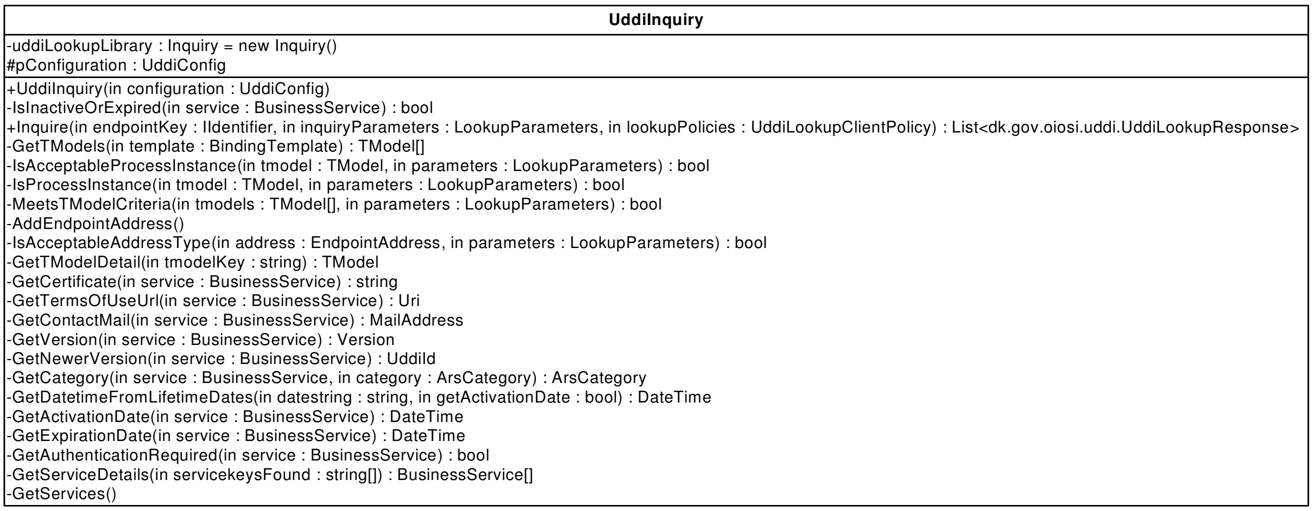
One of the UDDI support classes is the UddiLookupClientFactory, which creates an IUddiLookup implementation, as set in config.



```
UddiLookupClientFactory
-_config : UddiLookupClientFactoryConfig
+CreateUddiLookupClient() : IUddiLookupClient
```

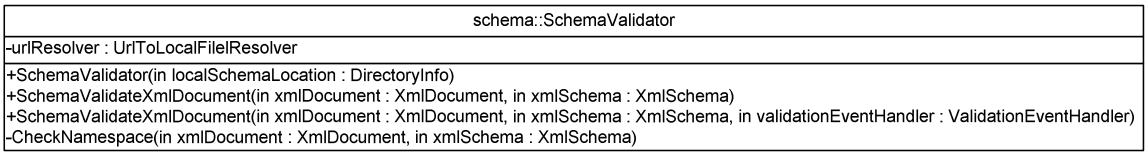
The factory could instantiate either *UddiLookupClient* or *UddiLookupClientTest*. You can change this behavior in the *RaspConfiguration.xml* file, in the section **<UddiLookupFactoryConfig>** element by setting the namespace and assembly name of the IUddiLookupClient implementation to instantiate. Developers may add their own implementation libraries and instantiate these just by changing the configuration.

The UDDI inquiry represents an inquiry to the UDDI inquiry API. The inquiry takes parameters like endpoint key, lookup policies, the UDDI endpoint etc.



```
UddiInquiry
-uddiLookupLibrary : Inquiry = new Inquiry()
#pConfiguration : UddiConfig
+UddiInquiry(in configuration : UddiConfig)
-IsInactiveOrExpired(in service : BusinessService) : bool
+Inquire(in endpointKey : IIdentifier, in inquiryParameters : LookupParameters, in lookupPolicies : UddiLookupClientPolicy) : List<dk.gov.oiosi.uddi.UddiLookupResponse>
-GetTModels(in template : BindingTemplate) : TModel[]
-IsAcceptableProcessInstance(in tmodel : TModel, in parameters : LookupParameters) : bool
-IsProcessInstance(in tmodel : TModel, in parameters : LookupParameters) : bool
-MeetsTModelCriteria(in tmodels : TModel[], in parameters : LookupParameters) : bool
-AddEndpointAddress()
-IsAcceptableAddressType(in address : EndpointAddress, in parameters : LookupParameters) : bool
-GetTModelDetail(in tmodelKey : string) : TModel
-GetCertificate(in service : BusinessService) : string
-GetTermsOfUseUrl(in service : BusinessService) : Uri
-GetContactMail(in service : BusinessService) : MailAddress
-GetVersion(in service : BusinessService) : Version
-GetNewerVersion(in service : BusinessService) : UddiId
-GetCategory(in service : BusinessService, in category : ArsCategory) : ArsCategory
-GetDatetimeFromLifetimeDates(in datestring : string, in getActivationDate : bool) : DateTime
-GetActivationDate(in service : BusinessService) : DateTime
-GetExpirationDate(in service : BusinessService) : DateTime
-GetAuthenticationRequired(in service : BusinessService) : bool
-GetServiceDetails(in servicekeysFound : string[]) : BusinessService[]
-GetServices()
```

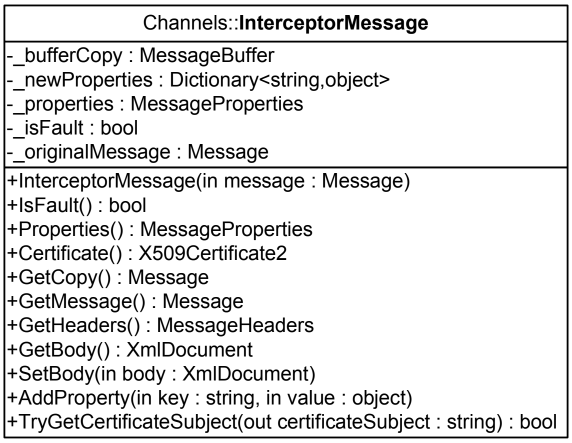
4.5 Validation, schema, client



```
schema::SchemaValidator
-urlResolver : UrlToLocalFileIResolver
+SchemaValidator(in localSchemaLocation : DirectoryInfo)
+SchemaValidateXmlDocument(in xmlDocument : XmlDocument, in xmlSchema : XmlSchema)
+SchemaValidateXmlDocument(in xmlDocument : XmlDocument, in xmlSchema : XmlSchema, in validationEventHandler : ValidationEventHandler)
-CheckNamespace(in xmlDocument : XmlDocument, in xmlSchema : XmlSchema)
```

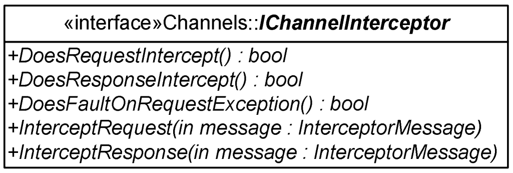
The SchemaValidator class is used by the schema validating interceptor to validate documents, but you may also use it as a stand-alone schema validator. When used by the interceptor, all schema includes or imports are resolved to a single disk location, regardless of the schemaLocation attribute, so that schemas are loaded locally.

4.6 Interceptors

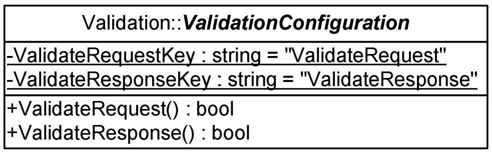


```
Channels::InterceptorMessage
-_bufferCopy : MessageBuffer
-_newProperties : Dictionary<string,object>
-_properties : MessageProperties
-_isFault : bool
-_originalMessage : Message
+InterceptorMessage(in message : Message)
+IsFault() : bool
+Properties() : MessageProperties
+Certificate() : X509Certificate2
+GetCopy() : Message
+GetMessage() : Message
+GetHeaders() : MessageHeaders
+GetBody() : XmlDocument
+SetBody(in body : XmlDocument)
+AddProperty(in key : string, in value : object)
+TryGetCertificateSubject(out certificateSubject : string) : bool
```

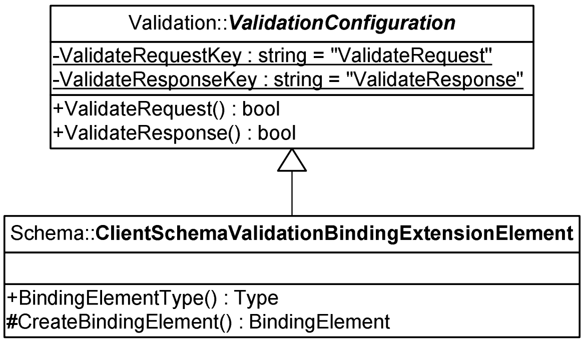
InterceptorMessage is an extension on the Message from the service model. It has extended functionality and is used by the interceptor when calling the IChannelInterceptor.


«interface»Channels::IChannelInterceptor
+DoesRequestIntercept() : bool
+DoesResponseIntercept() : bool
+DoesFaultOnRequestException() : bool
+InterceptRequest(in message : InterceptorMessage)
+InterceptResponse(in message : InterceptorMessage)

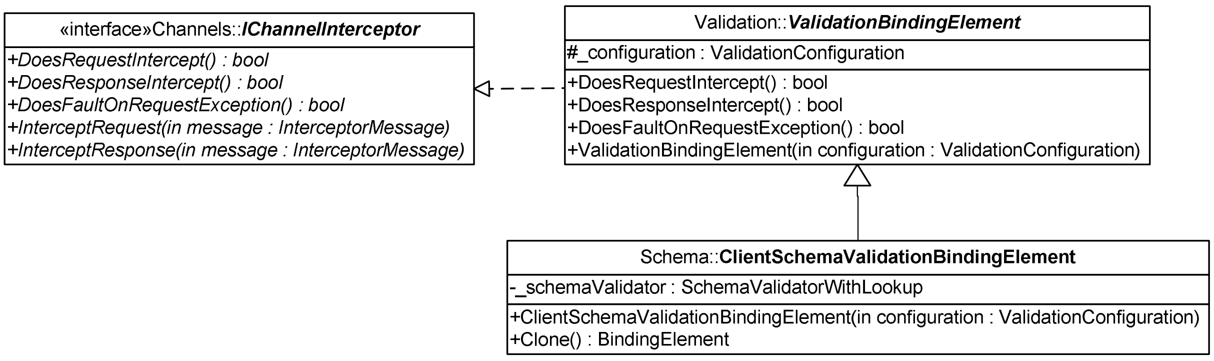
IChannelInterceptor is the interface any interceptor must implement. It is called when the request or response is intercepted.


Validation::ValidationConfiguration
-ValidateRequestKey : string = "ValidateRequest"
-ValidateResponseKey : string = "ValidateResponse"
+ValidateRequest() : bool
+ValidateResponse() : bool

ValidationConfiguration implements basic validation configuration.


Validation::ValidationConfiguration
-ValidateRequestKey : string = "ValidateRequest"
-ValidateResponseKey : string = "ValidateResponse"
+ValidateRequest() : bool
+ValidateResponse() : bool
Schema::ClientSchemaValidationBindingExtensionElement
+BindingElementType() : Type
#CreateBindingElement() : BindingElement

ClientSchemaValidationBindingExtensionElement inherits the ValidationConfiguration and creates the ClientSchemaValidationBindingElement when WCF is building the stack.

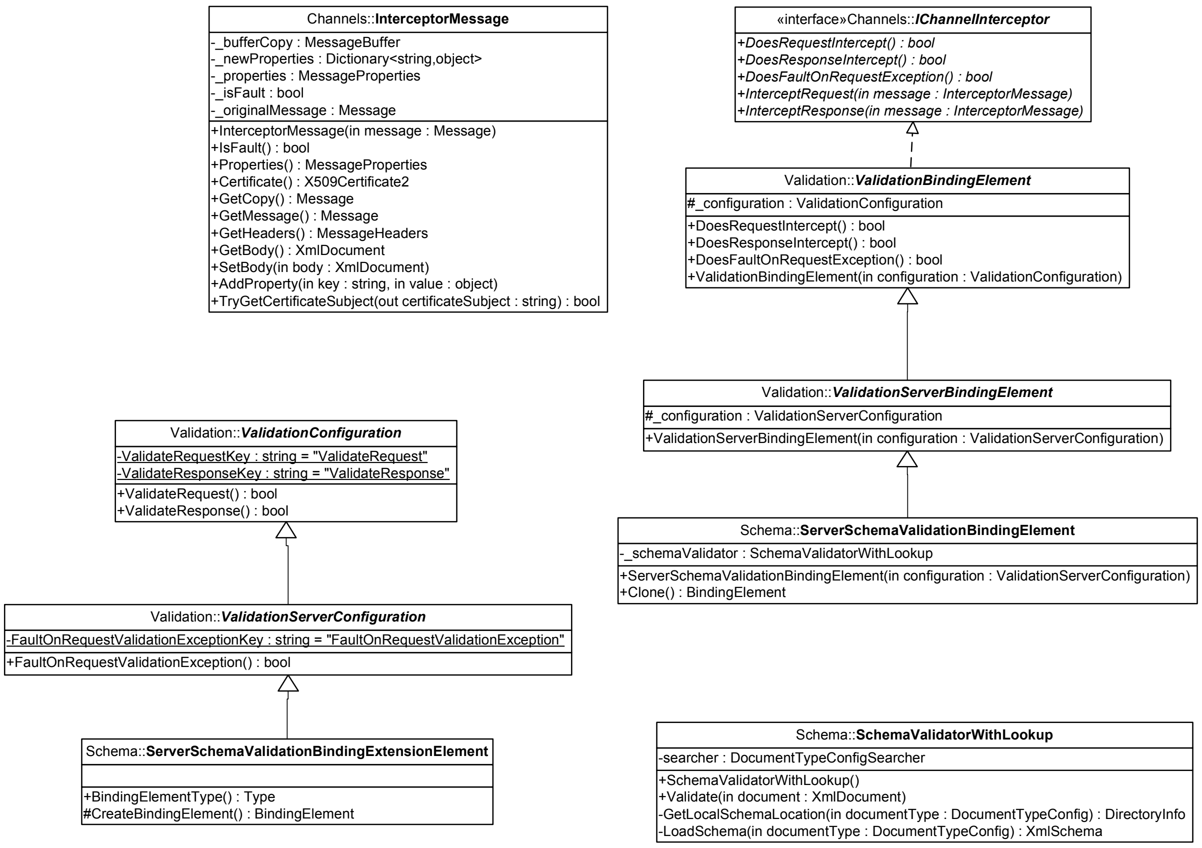

«interface»Channels::IChannelInterceptor
+DoesRequestIntercept() : bool
+DoesResponseIntercept() : bool
+DoesFaultOnRequestException() : bool
+InterceptRequest(in message : InterceptorMessage)
+InterceptResponse(in message : InterceptorMessage)
Validation::ValidationBindingElement
#_configuration : ValidationConfiguration
+DoesRequestIntercept() : bool
+DoesResponseIntercept() : bool
+DoesFaultOnRequestException() : bool
+ValidationBindingElement(in configuration : ValidationConfiguration)
Schema::ClientSchemaValidationBindingElement
-_schemaValidator : SchemaValidatorWithLookup
+ClientSchemaValidationBindingElement(in configuration : ValidationConfiguration)
+Clone() : BindingElement

ValidationBindingElement partly implements the IChannelInterceptor interface. It does not implement the interception methods, but it implements whether the inceptor does intercept request or response. This is determined by the state of ValidationConfiguration.

ClientSchemaValidationBindingElement inherits the ValidationBindingElement and implements the interception methods. Here the actual interception takes place and it calls the SchemaValidationWithLookup to validate whether the intercepted message body is schema valid.

SchemaValidationWithLookup validates an xml document by first finding the corresponding type of XML and then validate the document with the fitting schema.

4.7 Validation, Schema, Server


Channels::InterceptorMessage
-_bufferCopy : MessageBuffer
-_newProperties : Dictionary<string,object>
-_properties : MessageProperties
-_isFault : bool
-_originalMessage : Message
+InterceptorMessage(in message : Message)
+IsFault() : bool
+Properties() : MessageProperties
+Certificate() : X509Certificate2
+GetCopy() : Message
+GetMessage() : Message
+GetHeaders() : MessageHeaders
+GetBody() : XmlDocument
+SetBody(in body : XmlDocument)
+AddProperty(in key : string, in value : object)
+TryGetCertificateSubject(out certificateSubject : string) : bool
«interface»Channels::IChannelInterceptor
+DoesRequestIntercept() : bool
+DoesResponseIntercept() : bool
+DoesFaultOnRequestException() : bool
+InterceptRequest(in message : InterceptorMessage)
+InterceptResponse(in message : InterceptorMessage)
Validation::ValidationBindingElement
#_configuration : ValidationConfiguration
+DoesRequestIntercept() : bool
+DoesResponseIntercept() : bool
+DoesFaultOnRequestException() : bool
+ValidationBindingElement(in configuration : ValidationConfiguration)
Validation::ValidationServerBindingElement
#_configuration : ValidationServerConfiguration
+ValidationServerBindingElement(in configuration : ValidationServerConfiguration)
Validation::ValidationConfiguration
-ValidateRequestKey : string = "ValidateRequest"
-ValidateResponseKey : string = "ValidateResponse"
+ValidateRequest() : bool
+ValidateResponse() : bool
Schema::ServerSchemaValidationBindingElement
-_schemaValidator : SchemaValidatorWithLookup
+ServerSchemaValidationBindingElement(in configuration : ValidationServerConfiguration)
+Clone() : BindingElement
Validation::ValidationServerConfiguration
-FaultOnRequestValidationExceptionKey : string = "FaultOnRequestValidationException"
+FaultOnRequestValidationException() : bool
Schema::ServerSchemaValidationBindingExtensionElement
+BindingElementType() : Type
#CreateBindingElement() : BindingElement
Schema::SchemaValidatorWithLookup
-searcher : DocumentTypeConfigSearcher
+SchemaValidatorWithLookup()
+Validate(in document : XmlDocument)
-GetLocalSchemaLocation(in documentType : DocumentTypeConfig) : DirectoryInfo
-LoadSchema(in documentType : DocumentTypeConfig) : XmlSchema

InterceptorMessage is an extension on the Message from the service model. It has extended functionality and is used by the interceptor when calling the IChannelInterceptor.

IChannelInterceptor is the interface any interceptor must implement. It is called when the request or response is intercepted.

ValidationConfiguration implements basic validation configuration.

ValidationServerConfiguration implement the server specific validation configuration.

ServerSchemaValidationBindingExtensionElement inherits the ValidationServerConfiguration and creates the ServerSchemaValidationBindingElement when wcf is building the stack.

ValidationBindingElement partly implements the IChannelInterceptor interface. It does not implement the interception methods, but it implements whether the inceptor does intercept request or response. This is determined by the state of ValidationConfiguration.

ServerValidationBindingElement inherits the ValidationBindingElement and overrides whether the inceptor should fault or not. This is determined by the ValidationServerConfiguration.

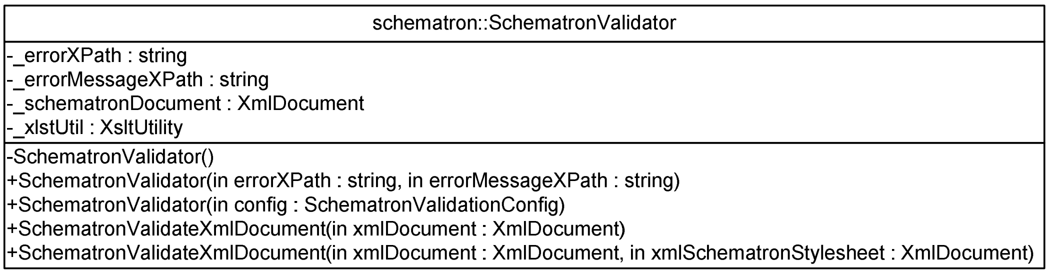
ServerSchemaValidationBindingElement inherits the ServerValidationBindingElement and implements the interception methods. Here the actual interception takes place and it calls the SchemaValidationWithLookup to validate whether the intercepted message body is schema valid.

SchemaValidationWithLookup validates an xml document by first finding the corresponding type of XML and then validate the document with the fitting schema.

The server side interception functions partly as the client side interception. It has the same configuration and same functionality. The only difference is whether it should send a soap fault or propagate the information when the validation fails.

4.8 Validation, schematron

The schematron validation interceptor works as the schema validation interceptor, except that it uses a schematron validation instead of a schema validation.



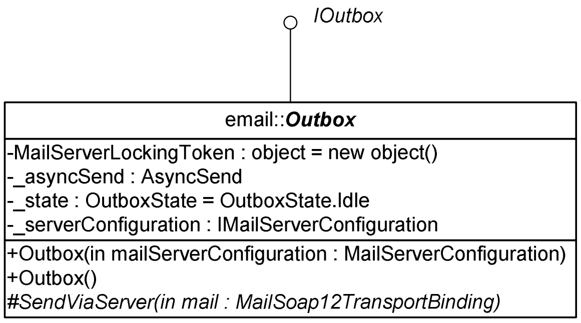
```
schematron::SchematronValidator
-_errorXPath : string
-_errorMessageXPath : string
-_schematronDocument : XmlDocument
-_xlstUtil : XsltUtility
-SchematronValidator()
+SchematronValidator(in errorXPath : string, in errorMessageXPath : string)
+SchematronValidator(in config : SchematronValidationConfig)
+SchematronValidateXmlDocument(in xmlDocument : XmlDocument)
+SchematronValidateXmlDocument(in xmlDocument : XmlDocument, in xmlSchematronStylesheet : XmlDocument)
```

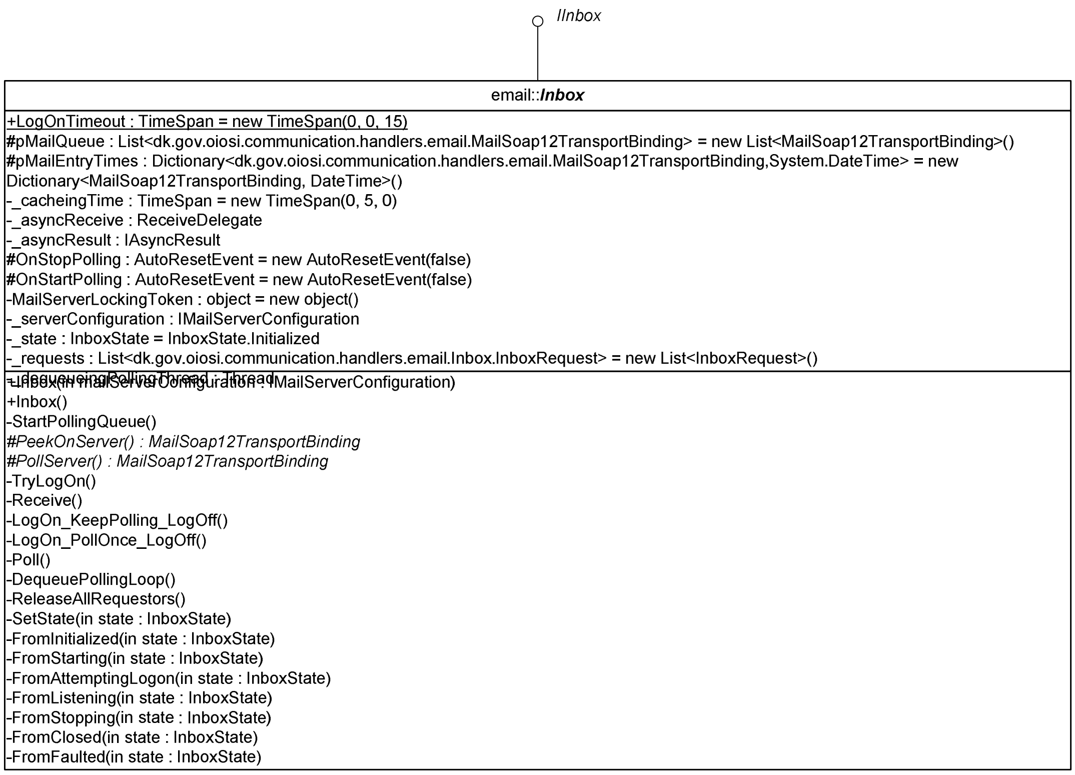
The SchematronValidator class is used by the schematron validating interceptor to validate documents, but you may also use it as a stand-alone schematron validator. Note that the SchematronValidator simply performs a stylesheet transformation of the document, using stylesheets representing schematron validation documents. There exists a stylesheet that transform schematron documents into stylesheets that perform an equivalent validation.

4.9 Email handling

The email handling component contains two virtual mailboxes, one inbox – used for receiving mails, and one outbox – for sending mails. The Rasp library ships with implementations of these base classes, using the Lesnikowski mail library. However one could choose to use their own implementation, to for example gain more control over the mail handling, or to use another licensed product than Lesnikowski (which is not free to distribute).

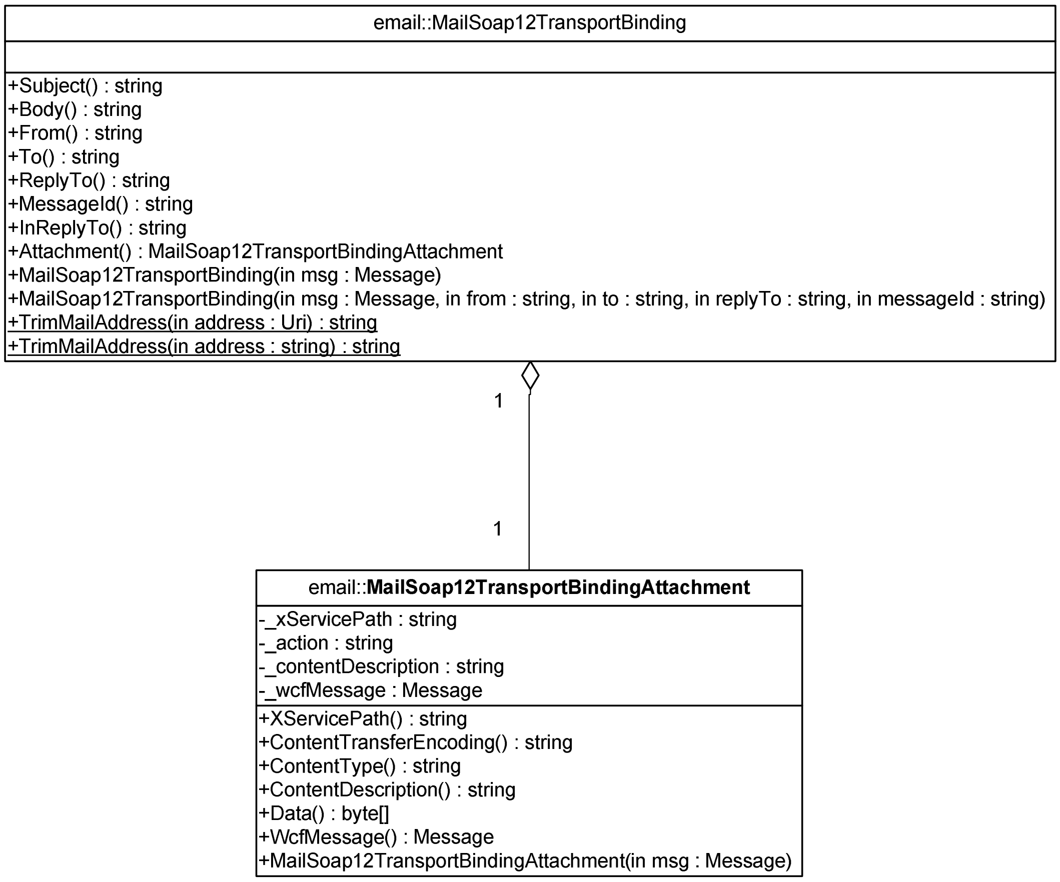
If you choose to do so, the classes Outbox and Inbox should be inherited, and their abstract methods implemented.


IOutbox
email::*Outbox*
-MailServerLockingToken : object = new object()
-_asyncSend : AsyncSend
-_state : OutboxState = OutboxState.Idle
-_serverConfiguration : IMailServerConfiguration
+Outbox(in mailServerConfiguration : MailServerConfiguration)
+Outbox()
#SendViaServer(in mail : MailSoap12TransportBinding)


IInbox
email::*Inbox*
+LogOnTimeout : TimeSpan = new TimeSpan(0, 0, 15)
#pMailQueue : List<dk.gov.oiosi.communication.handlers.email.MailSoap12TransportBinding> = new List<MailSoap12TransportBinding>()
#pMailEntryTimes : Dictionary<dk.gov.oiosi.communication.handlers.email.MailSoap12TransportBinding,System.DateTime> = new Dictionary<MailSoap12TransportBinding, DateTime>()
-_cacheingTime : TimeSpan = new TimeSpan(0, 5, 0)
-_asyncReceive : ReceiveDelegate
-_asyncResult : IAsyncResult
#OnStopPolling : AutoResetEvent = new AutoResetEvent(false)
#OnStartPolling : AutoResetEvent = new AutoResetEvent(false)
-MailServerLockingToken : object = new object()
-_serverConfiguration : IMailServerConfiguration
-_state : InboxState = InboxState.Initialized
-_requests : List<dk.gov.oiosi.communication.handlers.email.Inbox.InboxRequest> = new List<InboxRequest>()
-_dequeueingPollingThread : Thread
+Inbox(in mailServerConfiguration : IMailServerConfiguration)
+Inbox()
-StartPollingQueue()
#PeekOnServer() : MailSoap12TransportBinding
#PollServer() : MailSoap12TransportBinding
-TryLogOn()
-Receive()
-LogOn_KeepPolling_LogOff()
-LogOn_PollOnce_LogOff()
-Poll()
-DequeuePollingLoop()
-ReleaseAllRequestors()
-SetState(in state : InboxState)
-FromInitialized(in state : InboxState)
-FromStarting(in state : InboxState)
-FromAttemptingLogon(in state : InboxState)
-FromListening(in state : InboxState)
-FromStopping(in state : InboxState)
-FromClosed(in state : InboxState)
-FromFaulted(in state : InboxState)

While running the inbox will eat every mail there is on the mail server, and keeps them in memory, for de-queuing.

Both mailboxes deal with MailSoap12TransportBinding objects, that represents an OIOSI compliant mail containing a SOAP message.


email::MailSoap12TransportBinding
+Subject() : string
+Body() : string
+From() : string
+To() : string
+ReplyTo() : string
+MessageId() : string
+InReplyTo() : string
+Attachment() : MailSoap12TransportBindingAttachment
+MailSoap12TransportBinding(in msg : Message)
+MailSoap12TransportBinding(in msg : Message, in from : string, in to : string, in replyTo : string, in messageId : string)
+TrimMailAddress(in address : Uri) : string
+TrimMailAddress(in address : string) : string
1
1
email::**MailSoap12TransportBindingAttachment**
-_xServicePath : string
-_action : string
-_contentDescription : string
-_wcfMessage : Message
+XServicePath() : string
+ContentTransferEncoding() : string
+ContentType() : string
+ContentDescription() : string
+Data() : byte[]
+WcfMessage() : Message
+MailSoap12TransportBindingAttachment(in msg : Message)

The Message object representing your SOAP can be found in the Attachment property, and the other properties should give you the information needed to send a mail (such as subject line, encoding etc).

The Inbox/Outbox implementation would then be referred to from the App.config file in your email transport binding like:

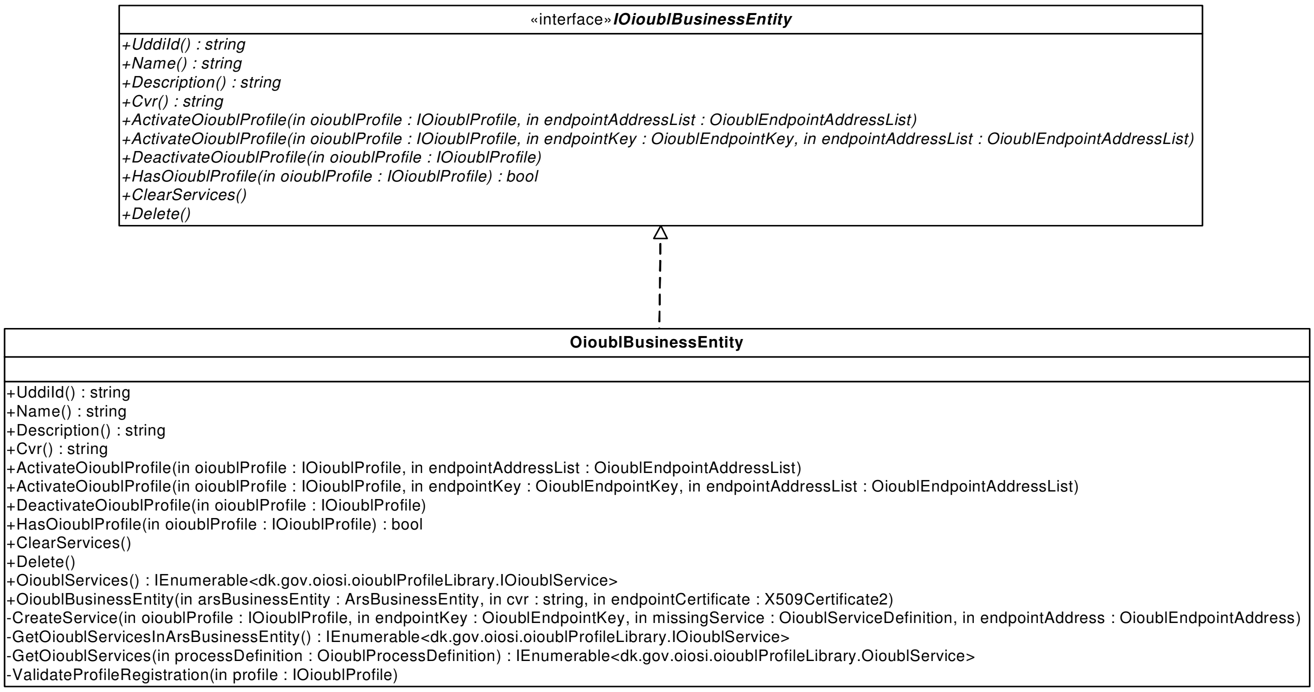
```
<emailTransport outboxImplementation="qualfied name of my outbox class"
inboxImplementation=" qualfied name of my outbox class "
sendingServerAddress="xxx" receivingServerAddress="xxx"
receivingUserName="xxx" receivingPassword="xxx" replyAddress="mailto:xxx" />
```

4.10 OIOUBL Profiling

The OIOUBL Profile library handles the management of OIOUBL Profiles and maps the OIOUBL entities to UDDI v3 entities.

4.10.1 OioublBusinessEntity

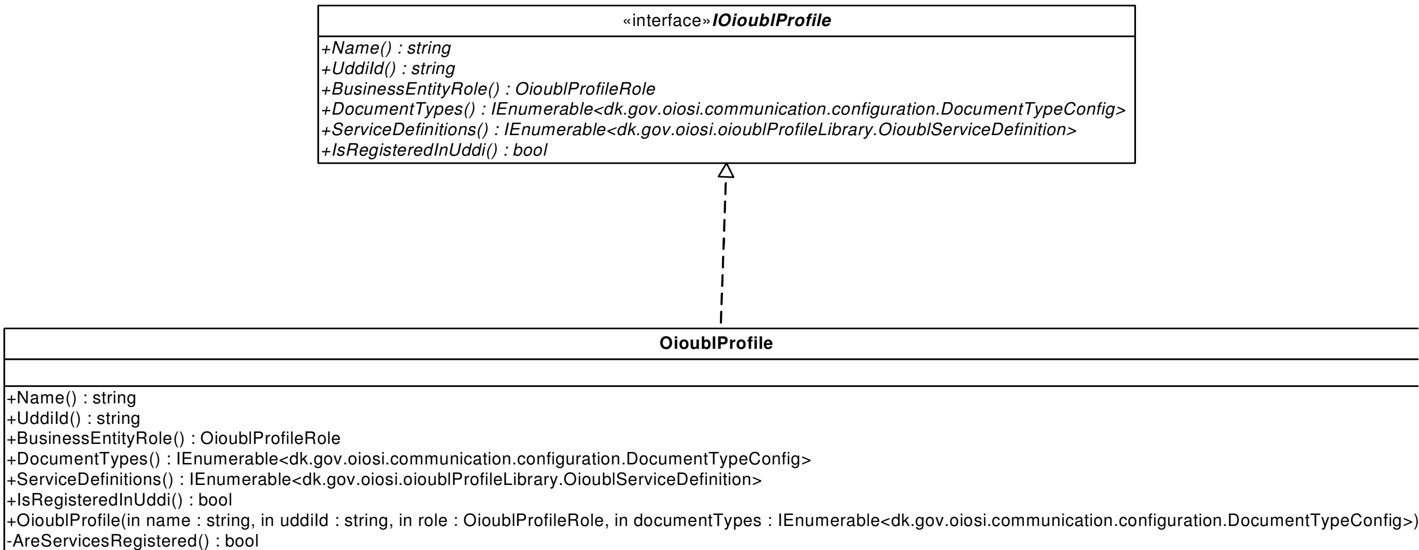
The root OIOUBL class type is the OioublBusinessEntity, which implement the IOioublBusinessEntity:


«interface»IOioublBusinessEntity
+UddiId() : string
+Name() : string
+Description() : string
+Cvr() : string
+ActivateOioublProfile(in oioublProfile : IOioublProfile, in endpointAddressList : OioublEndpointAddressList)
+ActivateOioublProfile(in oioublProfile : IOioublProfile, in endpointKey : OioublEndpointKey, in endpointAddressList : OioublEndpointAddressList)
+DeactivateOioublProfile(in oioublProfile : IOioublProfile)
+HasOioublProfile(in oioublProfile : IOioublProfile) : bool
+ClearServices()
+Delete()

OioublBusinessEntity
+UddiId() : string
+Name() : string
+Description() : string
+Cvr() : string
+ActivateOioublProfile(in oioublProfile : IOioublProfile, in endpointAddressList : OioublEndpointAddressList)
+ActivateOioublProfile(in oioublProfile : IOioublProfile, in endpointKey : OioublEndpointKey, in endpointAddressList : OioublEndpointAddressList)
+DeactivateOioublProfile(in oioublProfile : IOioublProfile)
+HasOioublProfile(in oioublProfile : IOioublProfile) : bool
+ClearServices()
+Delete()
+OioublServices() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.IOioublService>
+OioublBusinessEntity(in arsBusinessEntity : ArsBusinessEntity, in cvr : string, in endpointCertificate : X509Certificate2)
-CreateService(in oioublProfile : IOioublProfile, in endpointKey : OioublEndpointKey, in missingService : OioublServiceDefinition, in endpointAddress : OioublEndpointAddress)
-GetOioublServicesInArsBusinessEntity() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.IOioublService>
-GetOioublServices(in processDefinition : OioublProcessDefinition) : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublService>
-ValidateProfileRegistration(in profile : IOioublProfile)

4.10.2 OioublProfile

OioublProfiles are activated and deactivated resp. on an OioublBusinessEntity. The OioublProfile “knows” which Documents/Service Definitions and Profile/Process Instances make up a complete OioublProfile, and will create or removes the relevant Ars entities (which in return creates or removes UDDI entities).


«interface»IOioublProfile
+Name() : string
+UddiId() : string
+BusinessEntityRole() : OioublProfileRole
+DocumentTypes() : IEnumerable<dk.gov.oiosi.communication.configuration.DocumentTypeConfig>
+ServiceDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublServiceDefinition>
+IsRegisteredInUddi() : bool

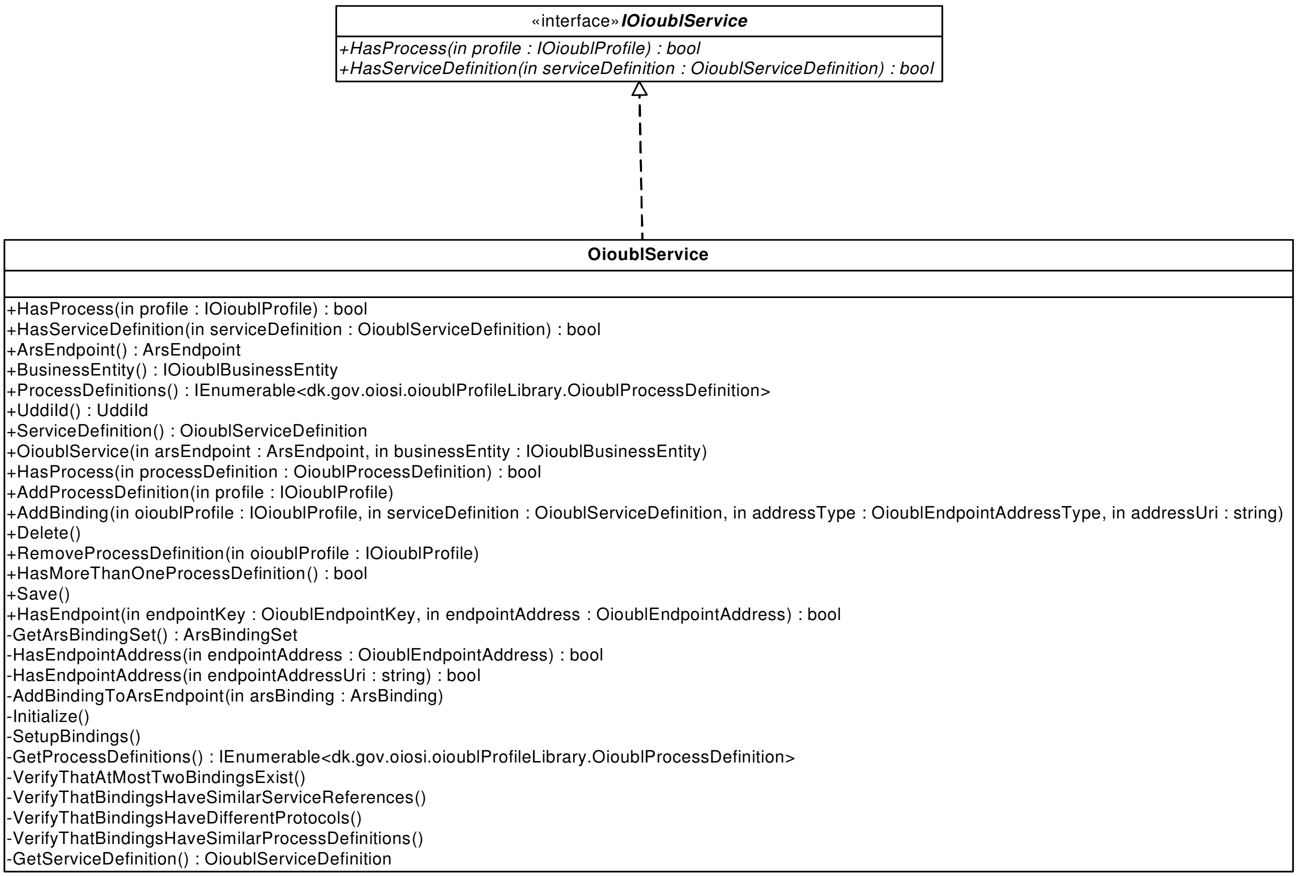
OioublProfile
+Name() : string
+UddiId() : string
+BusinessEntityRole() : OioublProfileRole
+DocumentTypes() : IEnumerable<dk.gov.oiosi.communication.configuration.DocumentTypeConfig>
+ServiceDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublServiceDefinition>
+IsRegisteredInUddi() : bool
+OioublProfile(in name : string, in uddiId : string, in role : OioublProfileRole, in documentTypes : IEnumerable<dk.gov.oiosi.communication.configuration.DocumentTypeConfig>)
-AreServicesRegistered() : bool

4.10.3 OioublService

Wrapper class for the ArsEndpoint class. An oioubl service is similar to an ARS endpoint, but with some built-in enforcements:

- Only two bindings are allowed
- The bindings must reference the same process definitions
- The bindings must reference the same service definition
- The bindings must use different protocols (ie, http, mtp, ...)

When creating an instance based on an existing ARS endpoint, an exception is thrown if these conditions are not met.



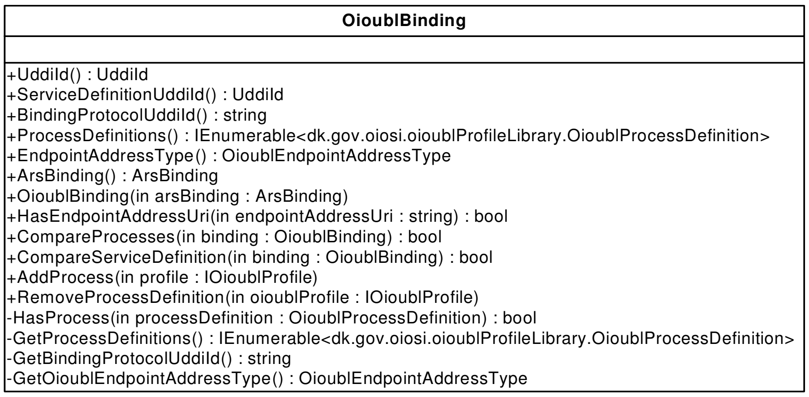
```
«interface»IOioublService
+HasProcess(in profile : IOioublProfile) : bool
+HasServiceDefinition(in serviceDefinition : OioublServiceDefinition) : bool
```

```
OioublService

+HasProcess(in profile : IOioublProfile) : bool
+HasServiceDefinition(in serviceDefinition : OioublServiceDefinition) : bool
+ArsEndpoint() : ArsEndpoint
+BusinessEntity() : IOioublBusinessEntity
+ProcessDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublProcessDefinition>
+UddiId() : UddiId
+ServiceDefinition() : OioublServiceDefinition
+OioublService(in arsEndpoint : ArsEndpoint, in businessEntity : IOioublBusinessEntity)
+HasProcess(in processDefinition : OioublProcessDefinition) : bool
+AddProcessDefinition(in profile : IOioublProfile)
+AddBinding(in oioublProfile : IOioublProfile, in serviceDefinition : OioublServiceDefinition, in addressType : OioublEndpointAddressType, in addressUri : string)
+Delete()
+RemoveProcessDefinition(in oioublProfile : IOioublProfile)
+HasMoreThanOneProcessDefinition() : bool
+Save()
+HasEndpoint(in endpointKey : OioublEndpointKey, in endpointAddress : OioublEndpointAddress) : bool
-GetArsBindingSet() : ArsBindingSet
-HasEndpointAddress(in endpointAddress : OioublEndpointAddress) : bool
-HasEndpointAddress(in endpointAddressUri : string) : bool
-AddBindingToArsEndpoint(in arsBinding : ArsBinding)
-Initialize()
-SetupBindings()
-GetProcessDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublProcessDefinition>
-VerifyThatAtMostTwoBindingsExist()
-VerifyThatBindingsHaveSimilarServiceReferences()
-VerifyThatBindingsHaveDifferentProtocols()
-VerifyThatBindingsHaveSimilarProcessDefinitions()
-GetServiceDefinition() : OioublServiceDefinition
```

4.10.4 OioublBinding

Wrapper class for the ArsBinding class. A binding is the combination of a binding protocol (http, smtp, ...) and a service definition reference. Use the factory to create an instance.

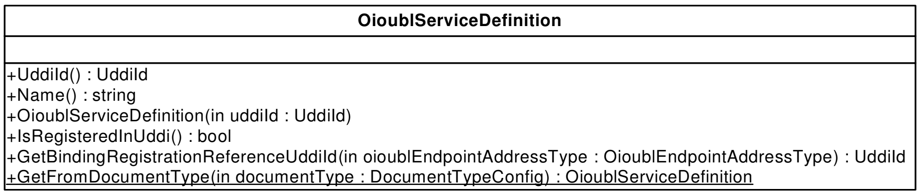


```
OioublBinding

+UddiId() : UddiId
+ServiceDefinitionUddiId() : UddiId
+BindingProtocolUddiId() : string
+ProcessDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublProcessDefinition>
+EndpointAddressType() : OioublEndpointAddressType
+ArsBinding() : ArsBinding
+OioublBinding(in arsBinding : ArsBinding)
+HasEndpointAddressUri(in endpointAddressUri : string) : bool
+CompareProcesses(in binding : OioublBinding) : bool
+CompareServiceDefinition(in binding : OioublBinding) : bool
+AddProcess(in profile : IOioublProfile)
+RemoveProcessDefinition(in oioublProfile : IOioublProfile)
-HasProcess(in processDefinition : OioublProcessDefinition) : bool
-GetProcessDefinitions() : IEnumerable<dk.gov.oiosi.oioublProfileLibrary.OioublProcessDefinition>
-GetBindingProtocolUddiId() : string
-GetOioublEndpointAddressType() : OioublEndpointAddressType
```

4.10.5 OioublServiceDefinition

Wrapper class for the OasisPortTypeRegistration class. An OioublServiceDefinition is similar to a DocumentType. In the UDDI registry the ServiceDefinition is stored as a OasisPortTypeRegistration.



OioublServiceDefinition

```
+UddiId() : UddiId
+Name() : string
+OioublServiceDefinition(in uddiId : UddiId)
+IsRegisteredInUddi() : bool
+GetBindingRegistrationReferenceUddiId(in oioublEndpointAddressType : OioublEndpointAddressType) : UddiId
+GetFromDocumentType(in documentType : DocumentTypeConfig) : OioublServiceDefinition
```