

# Architecture Design Document

Version	Date	Document Number	
1.0	2007/08/31		1/23

# - Contents -

<b>1 ABSTRACTION</b> .....	<b>3</b>
1.1 DOCUMENT NAME .....	3
1.2 OBJECTIVE .....	3
1.3 REQUIREMENTS.....	3
1.3.1 <i>Functional Requirements</i> .....	3
1.3.2 <i>Non-Functional Requirements</i> .....	3
<b>2 ARCHITECTURE</b> .....	<b>4</b>
2.1 CONVERTING RPC PROTOCOL.....	4
2.1.1 <i>[Notes] About the requirements which realize calling methods by the synchronous and asynchronous ways</i> .....	5
2.2 CREATING TARGET OBJECT .....	6
2.2.1 <i>[Notes] About which the architecture of creating is applied</i> .....	6
2.2.2 <i>Creating by the target object creation interface</i> .....	7
2.2.3 <i>Creating by the factory pattern</i> .....	9
2.3 LOOKING UP BY THE TARGET OBJECT CREATION INTERFACE .....	11
2.3.1 <i>[Notes] About which the architecture of looking up is applied</i> .....	11
2.3.2 <i>[Restrictions] About the restriction of the target object used as the candidate for looking up</i> .....	11
2.3.3 <i>Looking up by the target object search interface</i> .....	12
2.3.4 <i>Looking up by the factory pattern</i> .....	14
2.4 DELETING TARGET OBJECT.....	16
2.5 CROSS REFERENCE OF OBJECT REFERENCES.....	18
2.6 THE INTERFACE FOR USING FROM THE SCRIPT LANGUAGE.....	21
2.7 CODE GENERATOR .....	22

## 1 Abstraction

### 1.1 Document Name

This document name is "Architecture Design Document".

### 1.2 Objective

In the Linux desktop community, there are various component technologies (UNO, QtDBus, XPCOM, etc.) in which RPC protocols differ. So, reuse of components is to be given up, because application cannot be developed combining the components developed with different component technology. This is the big factor of the cause that Linux is not applied to the desktop from desktop application vendors which are asking for reducing development cost.

The Objective of Common (or Shared) Desktop Infrastructure is solving The factor. For that objective, mutual use between the components made with different component technologies are enabled by realizing the function which mediates the difference of RPC protocols. Consequently, the reusability of components increases. Therefore, the factor is solvable.

### 1.3 Requirements

The functional requirements and the non-functioning requirements for realizing the objective are written below.

#### 1.3.1 Functional Requirements

- (1) It is possible to call methods between the components made with different component technologies.**
- (2) It is possible to refer to object references mutually between the components made with different component technologies.**
- (3) It is possible to generate (activate) the remote objects of the component made with different component technologies.**
- (4) It is possible to search the remote objects of the component made with different component technologies.**
- (5) It is possible to delete the remote objects of the component made with different component technologies.**
- (6) It is possible to call methods by the synchronous and asynchronous ways both.**
- (7) It is possible to supply the interface for using components from script languages, such as JavaScript.**

#### 1.3.2 Non-Functional Requirements

- (1) It is possible to reuse components even if the engineers don't have high computing skill.**
- (2) It is possible to use the development environment of Common Desktop Infrastructure even if the engineer don't have high computing skill.**
- (3) It is unnecessary to change the specification of the existing component technologies.**
- (4) It is unnecessary to program in order to add the existing component on Common Desktop Infrastructure.**

## 2 Architecture

### 2.1 Converting RPC Protocol

There is RPC protocol conversion between the components made with different component technologies in the main functions for which Common Desktop Infrastructure is asked. The caller protocol converts into the callee protocol via conversion to abstract protocol.

Since realizing the protocol conversion without changing the existing component technologies is required, protocol conversion parts are realized in the exterior of the existing components. The following service processes realize protocol conversion.

1. Existing Component RPC Protocol -> Common Desktop Infrastructure RPC Protocol (RPC Routing Service of A Component)
2. Common Desktop Infrastructure RPC Protocol -> Existing Component RPC Protocol (RPC Routing Service of B Component)

The image of the relation between the above mentioned service and components, and RPC protocol conversion processing are as follows.

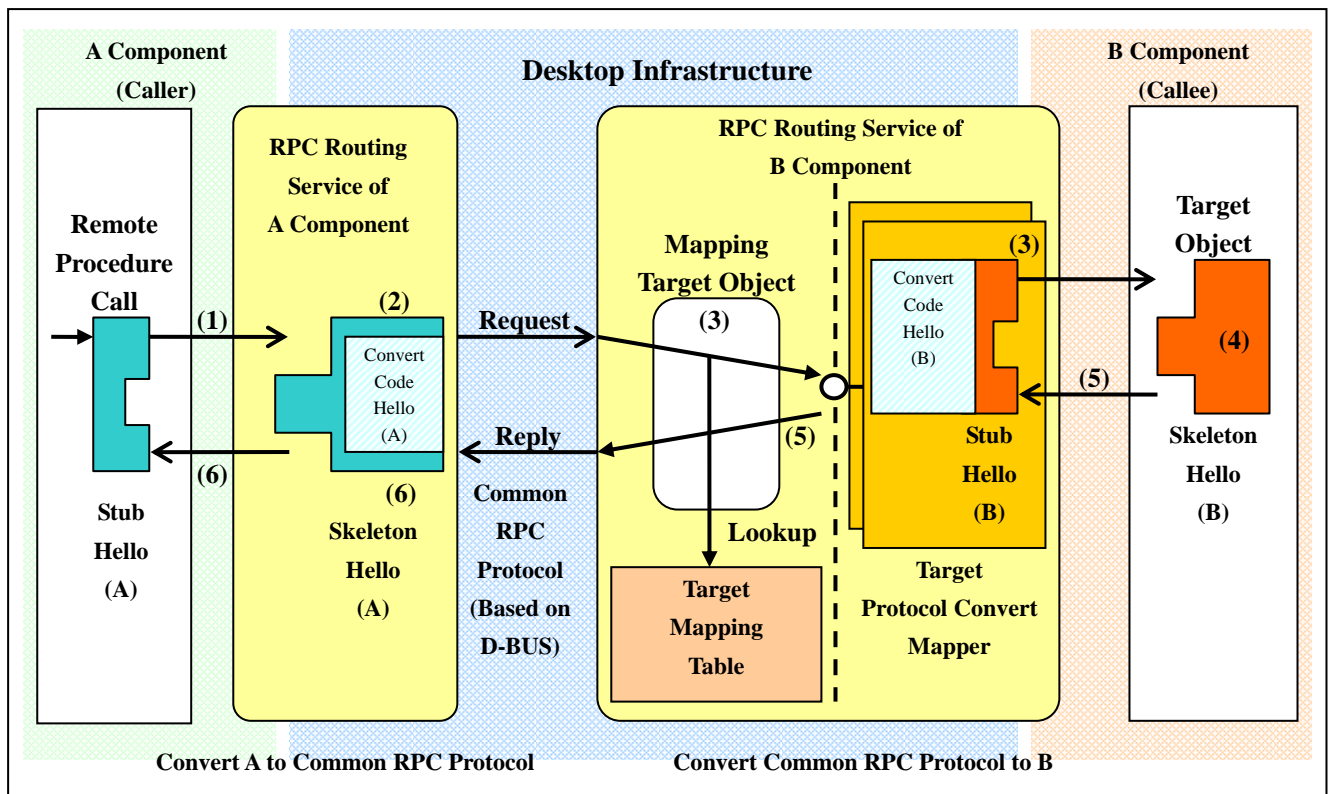


Figure 1 Convert RPC Protocol

The flow of protocol conversion processing is as follows.

## Architecture Design Document

No	Input	Process	Output
(1)	Function Call of Stub Hello (A)	1. RPC is requested to the skeleton Hello (A) which existing on RPC Routing Service of A component, according to the specification of the RPC protocol of A component.	RPC of Skeleton Hello (A) is transmitted to RPC Routing Service of A component.
(2)	The remote function call of Skeleton Hello (A) is received.	1. The arguments etc. of the called function are converted into common RPC protocol through conversion codes, and these are transmitted to RPC Routing Service of B component.	The function call converted into the common RPC protocol is transmitted to RPC routing service of B component.
(3)	The function call request by the common RPC protocol is received.	1. Target Protocol Mapper holding the remote object used as callee and conversion codes are specified, by searching Target Mapping Table which use Target Protocol Mapper ID which is contained in the received function call request as the key. 2. The arguments are taken out from the function call request by using specified Target Protocol Conversion Mapper. 3. The function in the target object of callee is called via the stub.	The function call RPC of Skeleton Hello (B) is transmitted to the target object.
(4)	The function call of Skeleton Hello (B) is received.	1. The target object which received the request of the function call returns return values and the values of the argument specified OUT as the response.	The response of a function call of Skeleton Hello (B) is transmitted to Stub Hello (B).
(5)	The response from Skeleton Hello (B) is received.	1. Return values and the values of the argument specified OUT are acquired, and the values are converted into the data types of common RPC protocol from the response which received. 2. As a response of the function call requested by the common RPC protocol, the return values and the values of arguments converted as mentioned above are returned.	The response of the function call required by the common RPC protocol is transmitted to Skeleton Hello (A).
(6)	The response of the function call requested by the common RPC protocol is received.	1. Return values and the values of the arguments specified OUT are acquired from the response which received, and these are converted into the data type of the RPC protocol of A component. 2. As a response of a function call of Stub Hello (A), the return values and the values of arguments converted as mentioned above are returned.	The response of the function call is transmitted to Stub Hello (A).

### 2.1.1 [Notes] About the requirements which realize calling methods by the synchronous and asynchronous ways

Since the architecture which doesn't change the specification of caller component technologies is adopted, the requirement is realizable by using a synchronous/asynchronous communication of the existing protocol as it is.

In the specification of the synchronous/asynchronous common RPC protocol of Common Desktop Infrastructure, the call function implementation (conversion code Hello (A)) of Skeleton Hello (A) is the same as the usual function call. Therefore, when the asynchronous call which needs the waiting processing for a response is adopted as a common RPC protocol, structure of the conversion code Hello (A) and RPC routing service of A component becomes complicated. So, only synchronous call is adopted as the common RPC protocol of Common Desktop Infrastructure this time.

However, when remarkable performance deterioration occurs by adopting synchronous function call, it is necessary to change common RPC protocol to asynchronous function call. Therefore, it is necessary to verify performance in advance.

## 2.2 Creating Target Object

Creating target object processes to create the object of callee component which consists of different protocol, after creating the object of caller component.

In the creating processing, it is necessary to create the object of callee component (target object) and to correlate the object with the object of caller component. The following two ways can be considered as methods of realizing the processes.

1. Creating by the target object creation interface.  
The interface for requiring target object creation is attached to the object of caller component corresponding to the target object.
2. Creating by the factory pattern.  
The target object and the object of caller component corresponding to the target object is created by using factory pattern.

After this, architectures of creating by the target object creation interface and creating by using the factory pattern are specified.

### 2.2.1 [Notes] About which the architecture of creating is applied.

Adopt the architecture using factory pattern except for the case where there is special reason, such as there is no means to return object reference to remote.

2.2.2 Creating by the target object creation interface

In the creation by using interface of target objects, the interface which creates target object is inherited to caller skeleton class with the same method as target object class, and implementation code is embedded to it.

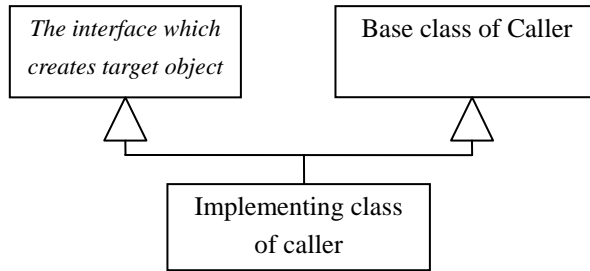


Figure 2 Relation between the interfaces of creation target object and the skeleton class

The image of creation process by using the interface which creates target object is as follows.

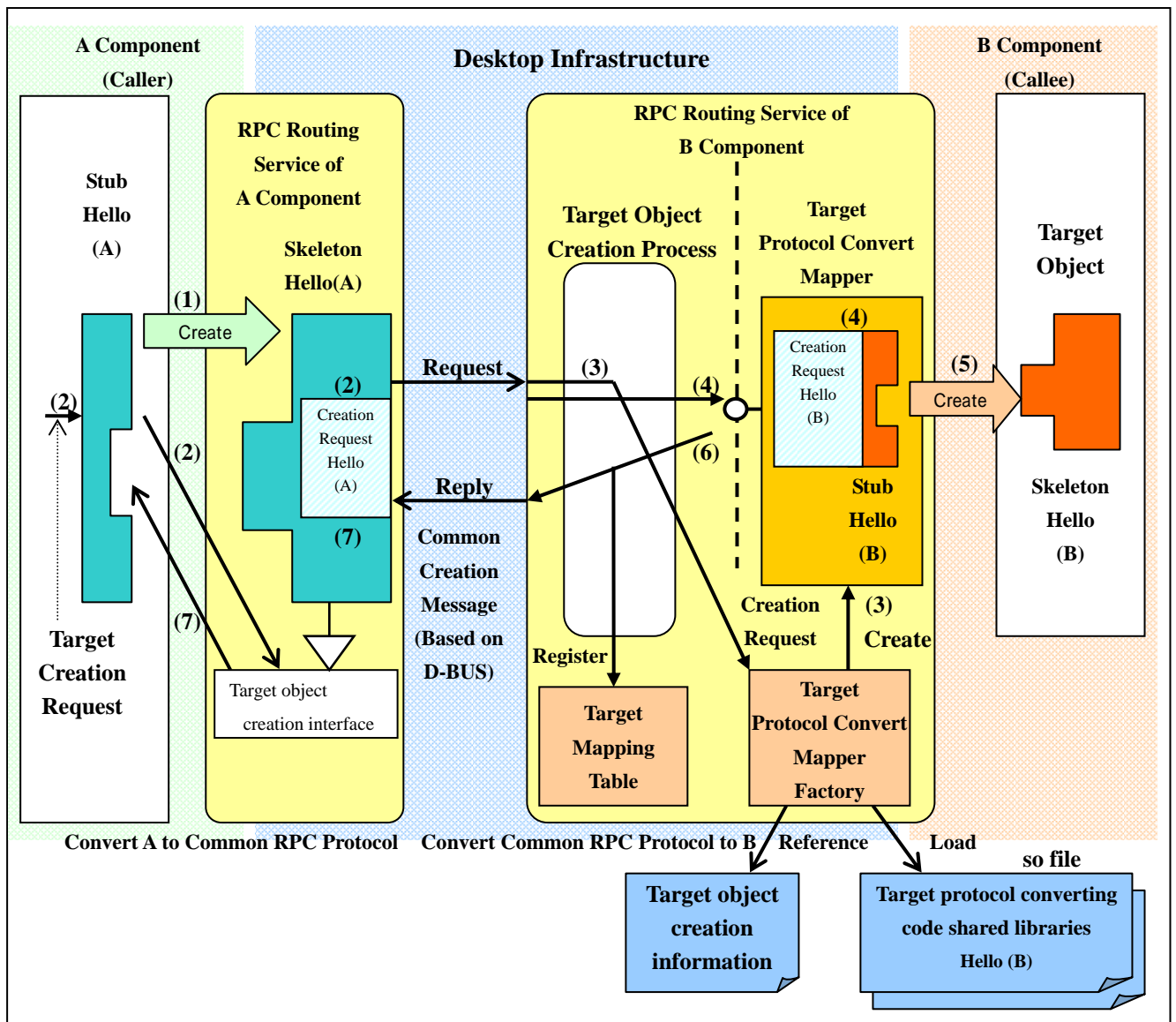


Figure 3 Creation process by using the interface which creates target object

The flow of creation process by using the interface which creates target object is as follows.

## Architecture Design Document

No	Input	Process	Output
(1)	Initialization of Stub Hello (A).	<ol style="list-style-type: none"> <li>1. Skeleton Hello (A) correspond to Stub Hello (A) is created, according to the specification of the RPC protocol of A component. When RPC Routing Service of A Component wasn't started, it is necessary to activate it.</li> </ol>	Creating Skeleton Hello (A).
(2)	The target creation request function of Stub Hello (A) is called.	<ol style="list-style-type: none"> <li>1. The target creation request function of Skeleton Hello (A) is called. In this case, when using the target object from other clients, the object name for discriminating the target object is passed to an argument.</li> <li>2. The target creation request function implemented in Skeleton Hello (A) creates creation request message in common RPC protocol, and transmits it to RPC routing service of B component.</li> </ol>	The creation request message in common RPC protocol is transmitted to RPC routing service of B component.
(3)	The creation request message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>1. The class name of the target object and component type (ex. UNO, QtDBus, etc.) are acquired from the creation request message in common RPC protocol, and creation of Target Protocol Convert Mapper is required to Target Protocol Convert Mapper factory.</li> <li>2. Target Protocol Mapper factory searches target object creation information by using the class name of the target object and component type as the key.</li> <li>3. The shared library name of the target protocol conversion code which creates Target Protocol Convert Mapper is acquired as a searched result, and the shared library is loaded.</li> <li>4. The Target Protocol Convert Mapper creation function contained in the shared library is called, and the Target Protocol Convert Mapper is created. (* Target Protocol Convert Mapper convert common RPC protocol into the protocol of the target object.</li> </ol>	Target Protocol Convert Mapper corresponding to the target object is created.
(4)	Target Protocol Convert Mapper corresponding to the target object.	<ol style="list-style-type: none"> <li>1. The Skeleton Hello (B) is created corresponding to the Stub Hello (B), according to the specification of the RPC protocol of B component, by using Target Protocol Convert Mapper. When the process to create the target object wasn't started, it is necessary to activate it.</li> </ol>	Creation of the target object is required.
(5)	The target object creation request is received.	<ol style="list-style-type: none"> <li>1. The target object is created.</li> </ol>	The target object is created.
(6)	The target object creation reply is received.	<ol style="list-style-type: none"> <li>1. The ID of Target Protocol Convert Mapper which used for target object creation is saved on Target Mapping Table. This ID is used at the time of RPC and remote object deletion.</li> <li>2. The creation reply message in common RPC protocol is created, and transmits it to RPC routing service of a component.</li> </ol>	The creation reply message in common RPC protocol is transmitted to Stub Hello (A).
(7)	The creation reply message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>1. The ID of Target Protocol Convert Mapper is acquired from reply message, and it is saved. This ID is used at the time of RPC and remote object deletion.</li> </ol>	The reply is transmitted to Stub Hello (A).

2.2.3 Creating by the factory pattern

In the creation by using factory pattern, the factory object is permanently stationed in RPC Routing Service of the caller Component. And clients send the target object creation request to the factory object. As the result, clients acquire the stub to the skeleton related with the target object.

The image of creation process by using factory pattern is as follows.

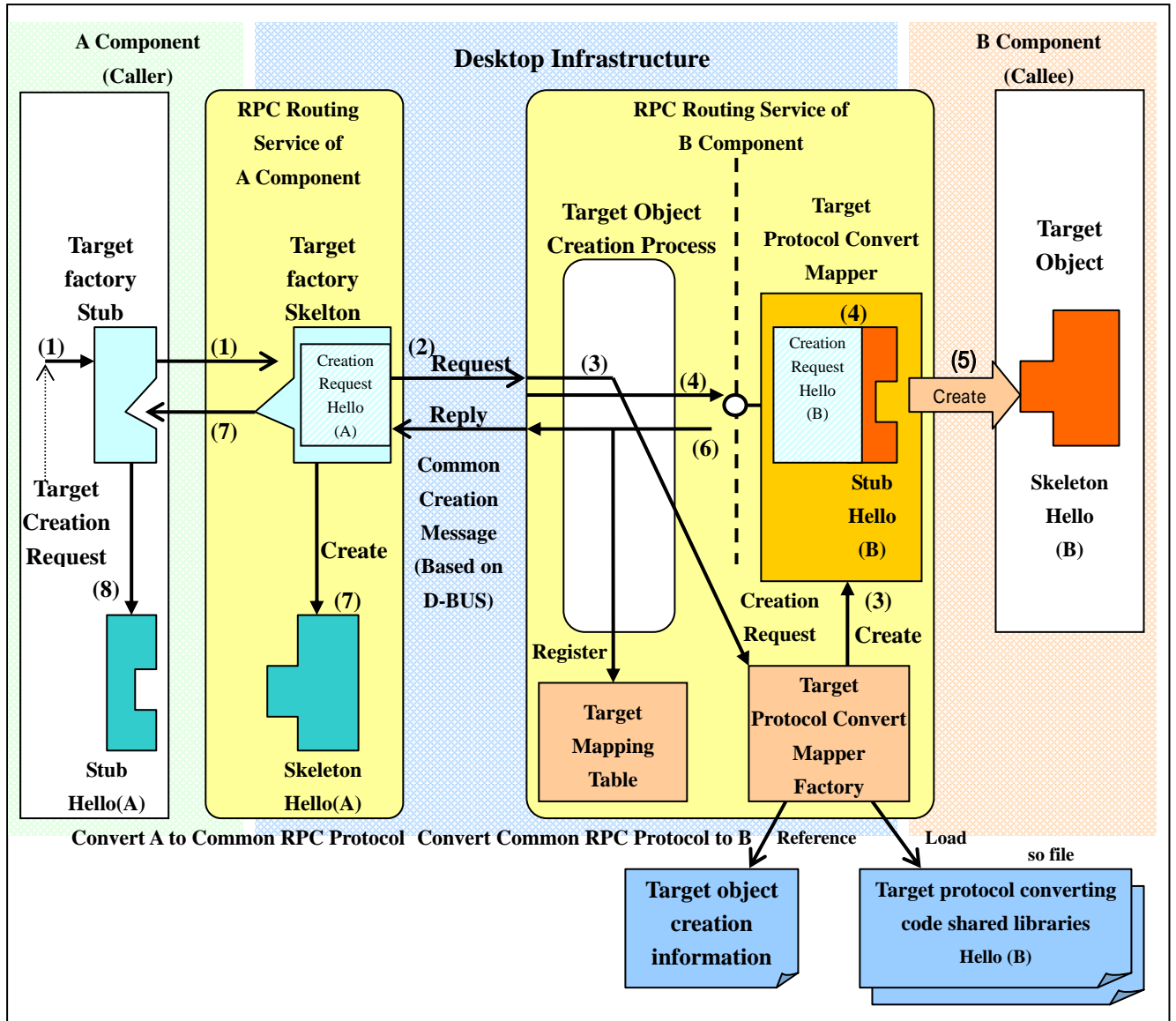


Figure 4 Creation process by using factory pattern

The flow of creation process by using factory pattern is as follows.

Architecture Design Document

No	Input	Process	Output
(1)	The target creation request is received from the target factory stub.	1. The target creation remote function is called to the target factory skeleton, according to the specification of the RPC protocol of a component. When RPC Routing Service of a Component wasn't started, it is necessary to activate it.	The target creation function of the target factory skeleton is called.
(2)	The target creation function of the target factory skeleton is called.	1. The target creation request function of the target factory skeleton creates creation request message in common RPC protocol, and it is transmitted to RPC Routing Service of B Component.	The creation request message in common RPC protocol is transmitted to RPC routing service of B component.
(3)	The creation request message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>1. The class name of the target object and component type (ex. UNO, QtDBUS, etc.) are acquired from the creation request message in common RPC protocol, creation of Target Protocol Convert Mapper is required to Target Protocol Conversion Mapper factory.</li> <li>2. Target Protocol Mapper factory searches target object creation information by using target the class name of the target object and component type as the key.</li> <li>3. The shared library name of the target protocol conversion code which creates Target Protocol Convert Mapper is acquired as a searched result, and the shared library is loaded.</li> <li>4. The Target Protocol Convert Mapper creation function contained in the shared library is called, and the Target Protocol Convert Mapper is created. (* Target Protocol Convert Mapper convert common RPC protocol into the protocol of the target object.</li> </ol>	Target Protocol Convert Mapper corresponding to the target object is created.
(4)	Target Protocol Convert Mapper corresponding to the target object.	1. The Skeleton Hello (B) is created corresponding to the Stub Hello (B), according to the specification of the RPC protocol of B component, by using Target Protocol Convert Mapper. When the process to create the target object wasn't started, it is necessary to activate it.	Creation of the target object is required.
(5)	The target object creation request is received.	1. The target object is created.	The target object is created.
(6)	The target object creation reply is received.	<ol style="list-style-type: none"> <li>1. The ID of Target Protocol Convert Mapper which used for target object creation is saved on Target Mapping Table. This ID is used at the time of RPC and remote object deletion.</li> <li>2. The creation reply message in common RPC protocol is created, and transmits it to RPC Routing Service of A component.</li> </ol>	The creation reply message in common RPC protocol is transmitted to Stub Hello (A).
(7)	The creation reply message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>1. The ID of Target Protocol Convert Mapper is acquired from reply message.</li> <li>2. Skeleton Hello (A) is created, and The ID of Target Protocol Convert Mapper is sated to it. This ID is used at the time of RPC and remote object deletion.</li> <li>3. The object reference of Skeleton Hello (A) is returned as the reply of the target creation request.</li> </ol>	The reply of the target creation request is transmitted to the target factory stub.
(8)	The reply of the target creation request is received.	1. The object reference of Skeleton Hello (A) is acquired from the reply, and the stub of Hello (A) is created.	Stub Hello (A) is created.

## 2.3 Looking up by the target object creation interface

Looking up target object is searching the target object on callee which already exists from caller component, and enabling it to use the target object.

In the looking up process mentioned above, the looking up process which finds out the target object of callee component, and the process which relates it with the object of caller component are needed. The following two methods can be considered as the means to realize these processes.

1. Looking up by the target object search interface.  
The interface for requiring target object search is attached to the object of caller component corresponding to the target object.
2. Looking up by the factory pattern.  
The target object and the object of caller component corresponding to the target object is looked up by using factory pattern.

After this, architectures of looking up by the target object creation interface and looking up by using the factory pattern are specified.

### 2.3.1 [Notes] About which the architecture of looking up is applied.

Adopt the architecture using factory pattern except for the case where there is special reason, such as there is no means to return object reference to remote.

### 2.3.2 [Restrictions] About the restriction of the target object used as the candidate for looking up.

Only the target object matched with Target Protocol Convert Mapper which exists in RPC Routing Service of B Component is made applicable to lookup. This means making applicable to only lookup the target object which was created via Common Desktop Infrastructure.

This restriction is avoidable by making RPC Routing Service of B Component to create Target Protocol Convert Mapper corresponding to the target object which is created without going via Common Desktop Infrastructure. However, it does not realize in this term.

2.3.3 Looking up by the target object search interface

In the looking up by using interface of target objects, the interface which lookup target object is inherited to caller skeleton class with the same method as target object class, and implementation code is embedded to it.

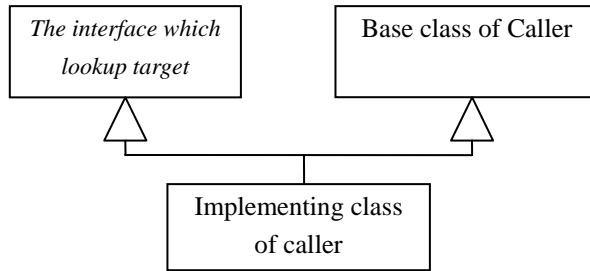


Figure 5 Relation between the interfaces of looking up target object and the skeleton class

The image of looking up process by using the interface which lookup target object is as follows.

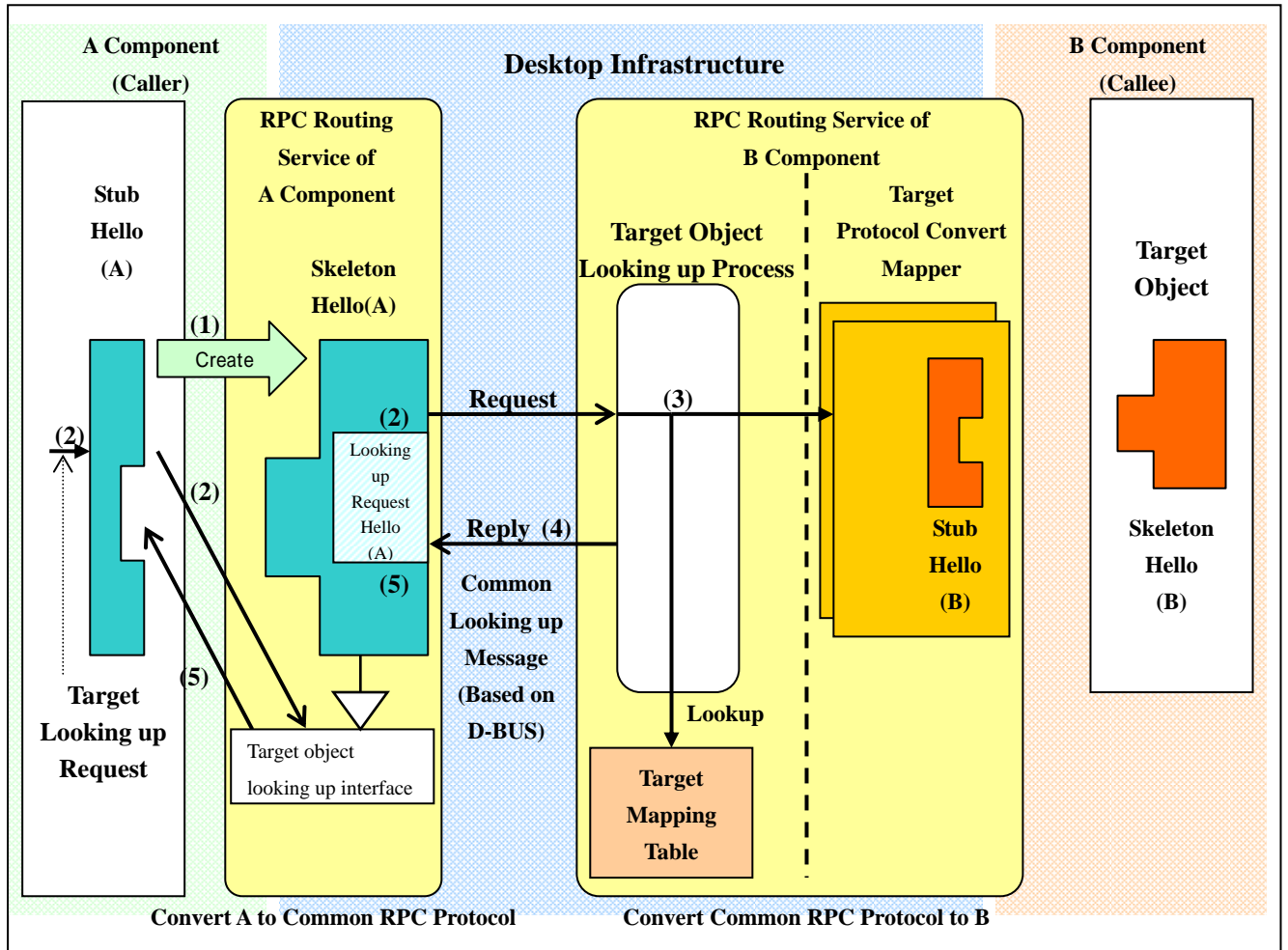


Figure 6 Looking up process by using the interface which lookup target object

The flow of looking up process by using the interface which lookup target object is as follows.

## Architecture Design Document

No	Input	Process	Output
(1)	Initialization of Stub Hello (A).	<ol style="list-style-type: none"> <li>1. Skeleton Hello (A) correspond to Stub Hello (A) is created, according to the specification of the RPC protocol of A component. When RPC Routing Service of A Component wasn't started, it is necessary to activate it.</li> </ol>	Creating Skeleton Hello (A).
(2)	The target looking up request function of Stub Hello (A) is called.	<ol style="list-style-type: none"> <li>1. The target looking up request function of Skeleton Hello (A) is called. In this case, the object name for discriminating the target object is passed to an argument.</li> <li>2. The target looking up request function implemented in Skeleton Hello (A) creates looking up request message in common RPC protocol, and transmits it to RPC Routing Service of B component.</li> </ol>	The looking up request message in common RPC protocol is transmitted to RPC routing service of B component.
(3)	The looking up request message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>1. The class name of the target object, component type (ex. UNO, QtDBus, etc.) and object name are acquired from the looking up request message in common RPC protocol.</li> <li>2. Target object looking up process searches Target Mapping Table by using the class name, component type and the object name as the key.</li> <li>3. The Target Protocol Convert Mapper is acquired as a searched result.</li> </ol>	Target Protocol Convert Mapper is acquired.
(4)	The Target Protocol Convert Mapper	<ol style="list-style-type: none"> <li>1. The ID is acquired from Target Protocol Convert Mapper.</li> <li>2. The reply message corresponding to looking up request message is created. And the reply message is transmitted after that the ID of Target Protocol Convert Mapper is setted to the message.</li> </ol>	The looking up reply message is transmitted.
(5)	The looking up message is received.	<ol style="list-style-type: none"> <li>1. The ID of Target Protocol Convert Mapper is acquired from reply message, and it is saved in Skeleton Hello (A). This ID is used at the time of RPC and remote object deletion.</li> </ol>	The reply is transmitted to Stub Hello (A).

2.3.4 Looking up by the factory pattern

In the looking up by using factory pattern, the factory object is permanently stationed in RPC Routing Service of the caller Component. And clients send the target object looking up request to the factory object. As the result, clients acquire the stub to the skeleton related with the target object.

The image of looking up process by using factory pattern is as follows.

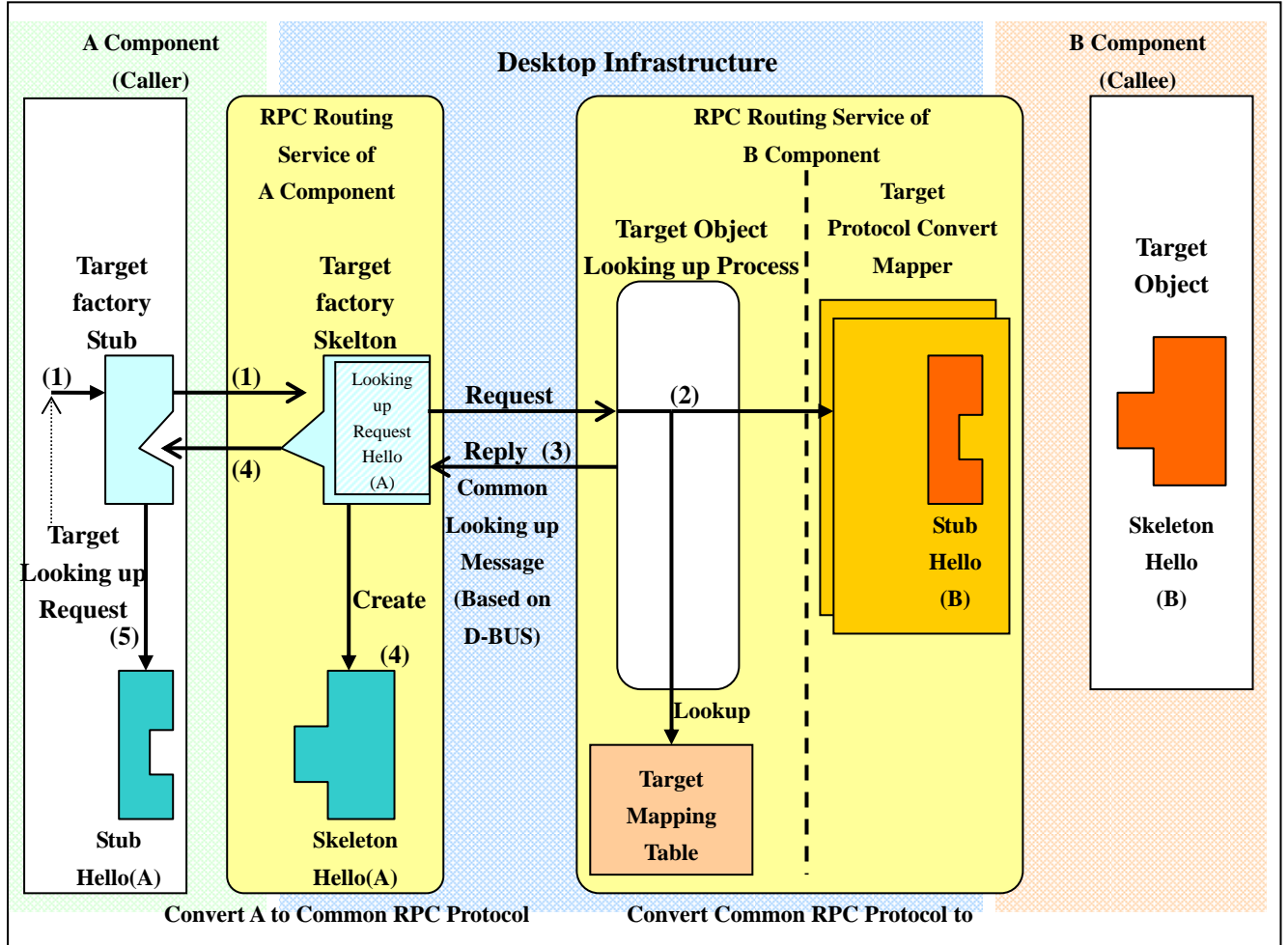


Figure 7 Looking up process by using factory pattern

## Architecture Design Document

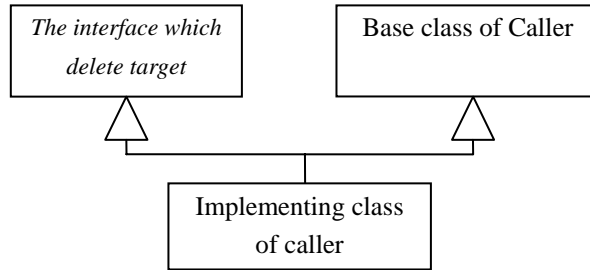
The flow of looking up process by using factory pattern is as follows.

No	Input	Process	Output
	The target looking up function of the target factory stub is called.	<ol style="list-style-type: none"> <li>The target looking up request function of the target factory skeleton creates looking up request message in common RPC protocol, and it is transmitted to RPC Routing Service of B Component. In this case, the object name for discriminating the target object is passed to an argument.</li> </ol>	The looking up request message in common RPC protocol is transmitted to RPC Routing Service of B component.
(2)	The looking up request message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>The class name of the target object, component type (ex. UNO, QtDBus, etc.) and object name are acquired from the looking up request message in common RPC protocol.</li> <li>Target object looking up process searches Target Mapping Table by using the class name, component type and the object name as the key.</li> <li>The Target Protocol Convert Mapper is acquired as a searched result.</li> </ol>	Target Protocol Convert Mapper is acquired.
(3)	The Target Protocol Convert Mapper	<ol style="list-style-type: none"> <li>The ID is acquired from Target Protocol Convert Mapper.</li> <li>The reply message corresponding to looking up request message is created. And the reply message is transmitted after that the ID of Target Protocol Convert Mapper is setted to the message.</li> </ol>	The looking up reply message is transmitted.
(4)	The looking up message is received.	<ol style="list-style-type: none"> <li>The ID of Target Protocol Convert Mapper is acquired from reply message.</li> <li>Skeleton Hello (A) is created, and The ID of Target Protocol Convert Mapper is sated to it. This ID is used at the time of RPC and remote object deletion.</li> <li>The object reference of Skeleton Hello (A) is returned as the reply of the target looking up request.</li> </ol>	The reply of the target looking up request is transmitted to the target factory stub.
(5)	The reply of the target looking up request is received.	<ol style="list-style-type: none"> <li>The object reference of Skeleton Hello (A) is acquired from the reply, and the stub of Hello (A) is created.</li> </ol>	Stub Hello (A) is created.

## 2.4 Deleting Target Object

Deleting target object is deletion the object on caller component, deletion the target object on callee component.

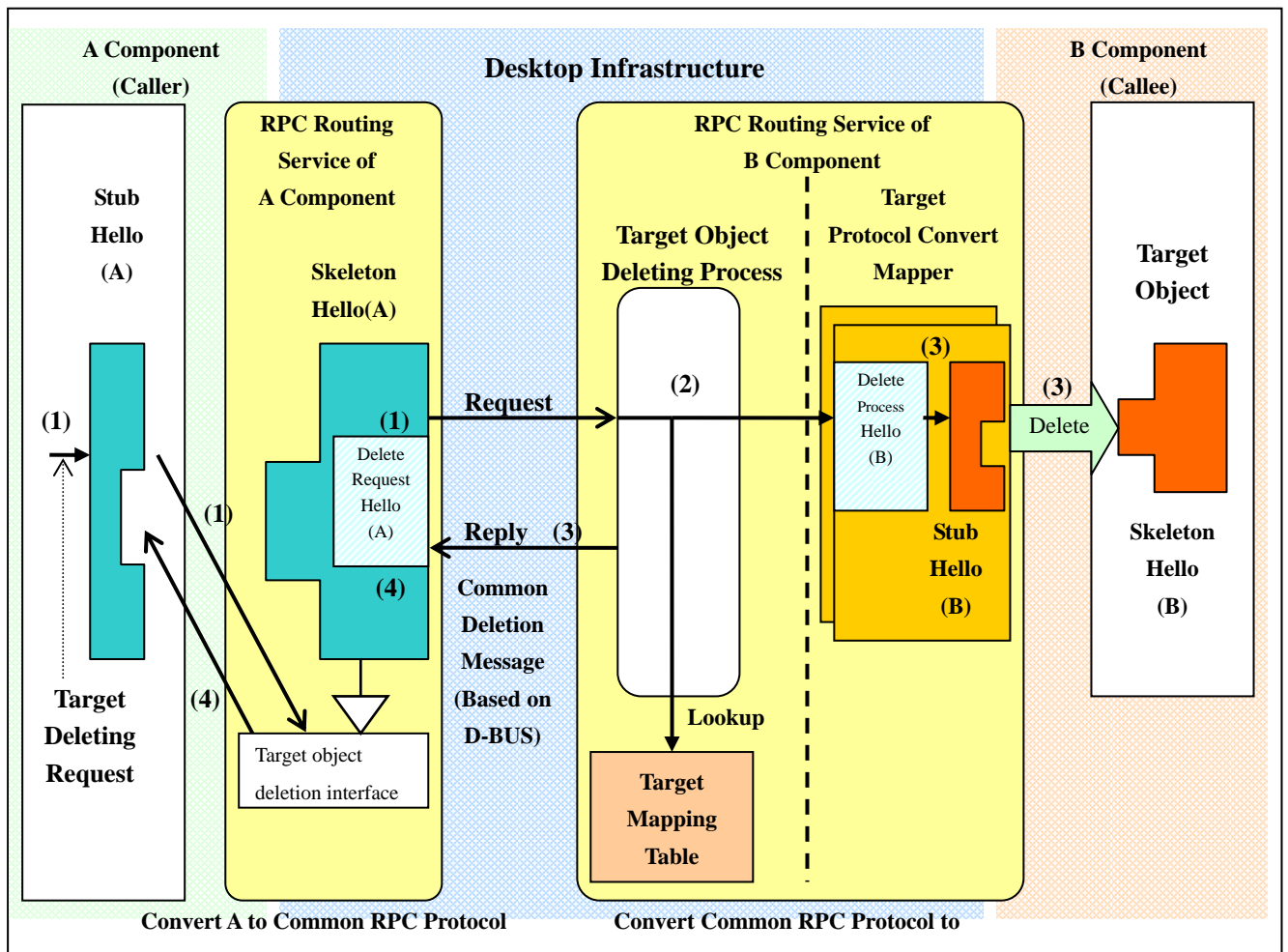
In order to delete the target object from caller, the interface which delete target object is inherited to caller skeleton class with the same method as target object class, and implementation code is embedded to it.



**Figure 8 Relation between the interfaces of deleting target object and the skeleton class**

However, when the interface which deletes the remote object (skeleton) as the function of caller component exists, use the deletion interface of the component.

The image of target object deleting process is as follows.



**Figure 9 Target object deletion process**

The flow of target object deletion process is as follows.

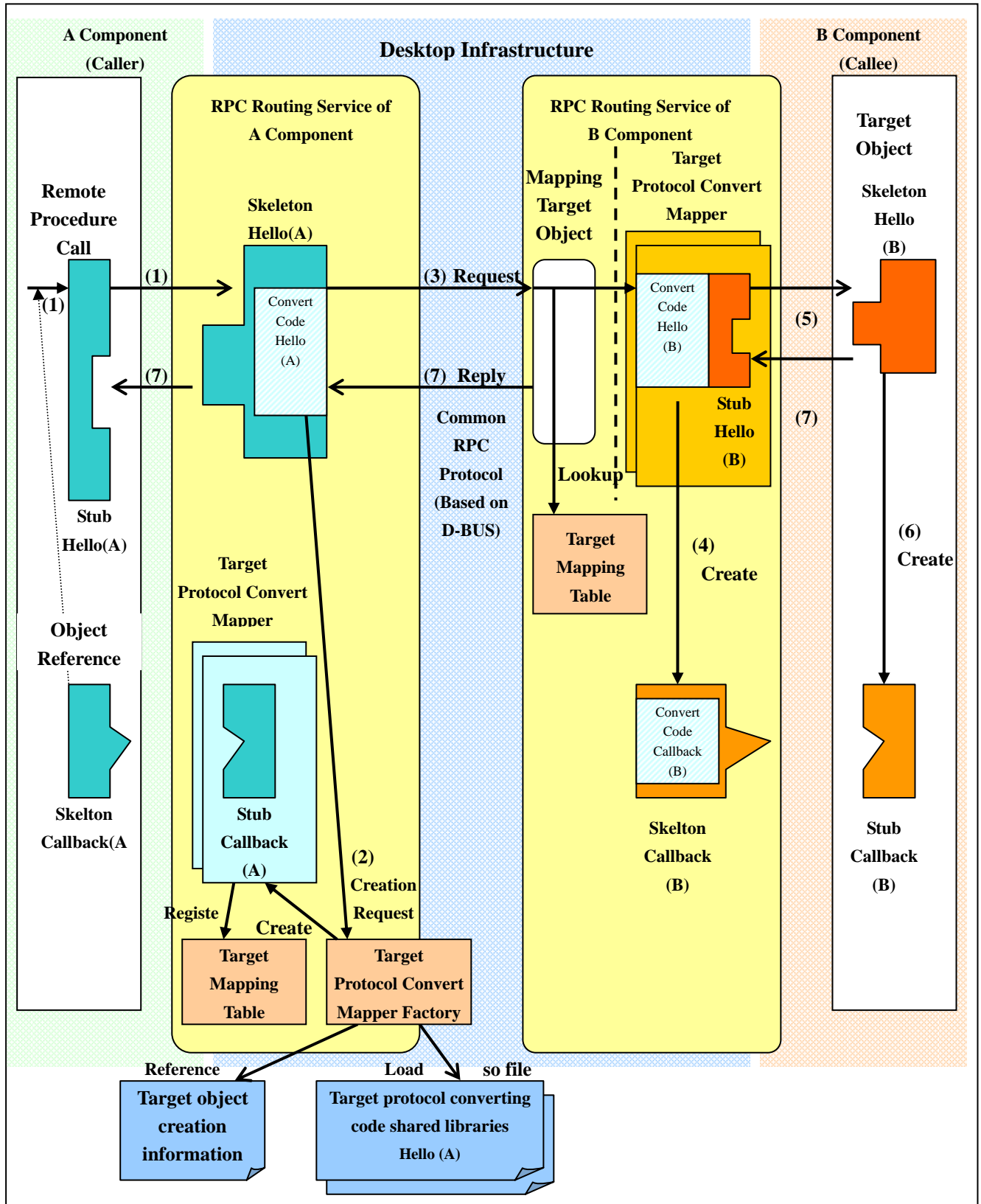
## Architecture Design Document

No	Input	Process	Output
(1)	The target deletion request function of Stub Hello (A) is called.	<ol style="list-style-type: none"> <li>The target deletion request function of Skeleton Hello (A) is called.</li> <li>The target deletion request function implemented in Skeleton Hello (A) creates deletion request message in common RPC protocol, and transmits it to RPC Routing Service of B component.</li> </ol>	The deletion request message in common RPC protocol is transmitted to RPC Routing Service of B component.
(2)	The deletion request message in common RPC protocol is received.	<ol style="list-style-type: none"> <li>The class name of the target object, component type (ex. UNO, QtDBus, etc.) and object name are acquired from the deletion request message in common RPC protocol.</li> <li>Target object looking up process searches Target Mapping Table by using the class name, component type and the object name as the key.</li> <li>The Target Protocol Convert Mapper is acquired as a searched result.</li> </ol>	Target Protocol Convert Mapper is acquired.
(3)	The Target Protocol Convert Mapper	<ol style="list-style-type: none"> <li>The skeleton Hello (B) which is target object is deleted from Stub Hello (B).</li> <li>Target Protocol Convert Mapper is freed and it is deleted from Target Mapping Table.</li> <li>The deletion reply message in common RPC protocol is created, and transmits it to RPC Routing Service of A component.</li> </ol> <p>[Notes]                      Don't delete about the target object currently referred to from the some client. The judgment about whether it is deleted or not is dependent on the specification of each component technologies.                      Moreover, about SOAP, since it is premised on using the service which has already existed, Target Protocol Convert Mapper is freed without deleting the target object.</p>	The deletion reply message is transmitted.
(4)	The reply of the deletion request is received.	<ol style="list-style-type: none"> <li>The object name held is deleted, and the state is changed into what the relation with the target object deleted.</li> <li>If release of Skeleton Hello (A) is possible, Skeleton Hello (A) will be freed. However, it is dependent on the specification of the component.</li> </ol>	The reply is transmitted to Stub Hello (A).

## 2.5 Cross reference of object references

In order to enable the cross reference of the target object realized with different component technologies, it is necessary to make the course of RPC routing corresponding to the object reference which is seted into the argument at the time of the remote function call.

The image of flow until the routing course is built is as follows.



## Architecture Design Document

The flow until the routing course is built is as follows.

No	Input	Process	Output
(1)	Remote function call	<ol style="list-style-type: none"> <li>The object reference of the remote object (skeleton Callback (A)) is acquired from the argument of the function. The object reference is referred to from the target object.</li> </ol>	Object reference of Skeleton Callback (A)
(2)	The object reference of the skeleton Callback (A)	<ol style="list-style-type: none"> <li>Stub Callback (A) is created from the object reference of Skeleton Callback (A).</li> <li>Creation of Target Protocol Convert Mapper is required from the Target Protocol Convert Mapper factory.</li> <li>Target Protocol Mapper factory searches target object creation information by using the class name of the target object and component type as the key.</li> <li>The shared library name of the target protocol conversion code which creates Target Protocol Convert Mapper is acquired as a searched result, and the shared library is loaded.</li> <li>The Target Protocol Convert Mapper creation function contained in the shared library is called, and the Target Protocol Convert Mapper is created.</li> <li>Reference of Stub Callback (A) is set on Target Protocol Convert Mapper.</li> <li>The ID of Target Protocol Convert Mapper and the reference is saved on Target Mapping Table.</li> <li>The ID of Target Protocol Convert Mapper and class name are acquired.</li> </ol>	The ID of Target Protocol Convert Mapper
(3)	The ID of Target Protocol Convert Mapper	<ol style="list-style-type: none"> <li>The arguments of the called method etc. are changed into common RPC protocol by using conversion code. In the conversion process mentioned above, the object reference of Skeleton Callback (A) is converted to the ID of Target Protocol Convert Mapper and class name.</li> <li>It is transmitted to RPC Routing Service of B Component.</li> </ol>	The method call request converted into the common RPC protocol is transmitted to RPC Routing Service of B component.
(4)	The method call request converted into the common RPC protocol is received.	<ol style="list-style-type: none"> <li>Target Protocol Mapper holding the remote object used as callee and conversion codes are specified, by searching Target Mapping Table which use Target Protocol Mapper ID which is contained in the received function call request as the key.</li> <li>The arguments are taken out from the function call request by using specified Target Protocol Conversion Mapper.</li> <li>The stub class to create is specified from the ID of Target Protocol Convert Mapper and class name (if class name exists), the Skeleton Callback (B) is created by using these.</li> <li>The ID of Target Protocol Convert Mapper corresponding to Skeleton Callback (B) is registered to it.</li> </ol>	Skeleton Callback (B)
(5)	Skeleton Callback (B)	<ol style="list-style-type: none"> <li>The method of the target object on callee is called via Stub Hello (B). In the calling process mentioned above, the ID of Target Protocol Convert Mapper in the arguments is converted to the object reference of Skeleton Callback (B).</li> </ol>	The remote function call request of Skeleton Hello (B) is transmitted.
(6)	The remote function call request of Skeleton Hello (B) is received.	<ol style="list-style-type: none"> <li>Stub Callback (B) is created from the object reference of the skeleton Callback (B) contained in the arguments.</li> </ol>	Stub Callback (B)

## Architecture Design Document

(7)	Stub Callback(B)	1. The reply is returned This reply process is the same as reply process of RPC conversion process.	
-----	------------------	---	--

## 2.6 The interface for using from the script language

The following functions are provided as the interface which uses the remote object of each component technologies by using Common Desktop Infrastructure from the script language.

1. Creating remote object
2. Looking up remote object
3. Deleting remote object
4. Cross reference of object references

Using the functions mentioned above is enabled from the script language by preparing the object which accesses the functions mentioned above at the time of initialization before script execution.

This method is the same as the thought which accesses DOM and controls the browser from JavaScript on the web browser.

## 2.7 Code generator

The code generator is the function which generates automatically the code of the parts which convert between the protocol of each component technologies and the common protocol in Common Desktop Infrastructure. The code is generated from IDL of the target object. By using this code generator, the work which is enabled connection to target object from other components is automated. Since artificial work can be eliminated as this result, the threshold of initial introduction can be lowered.

The relation between the code generator and the code generated automatically is as follows.

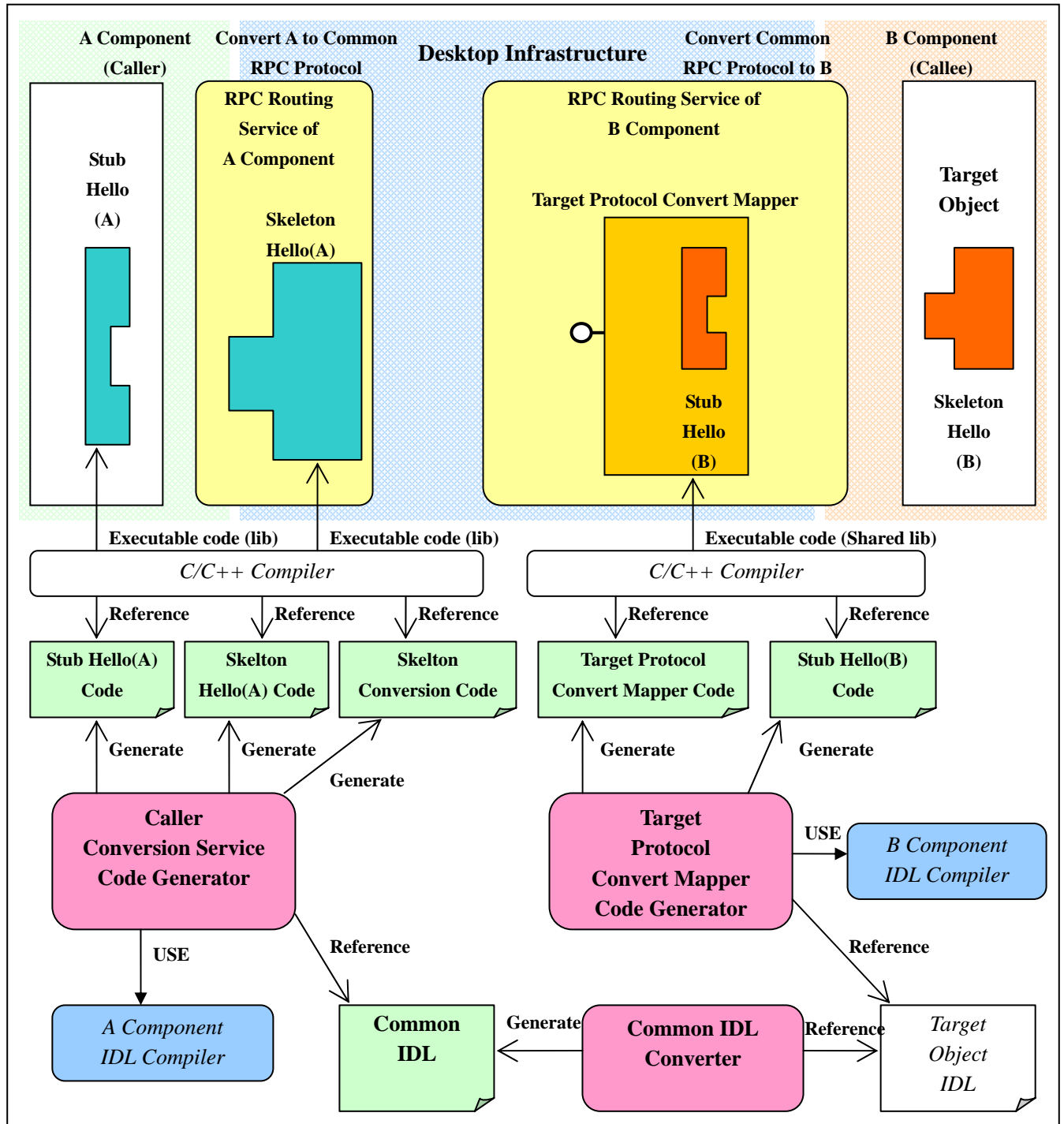


Figure 10 Structure of code generator

## Architecture Design Document

Explanation of the structure element shown by Figure 10 Structure of code generator is as follows.

No	Name of structure element	Explanation
1	Target Object IDL	This is the IDL definition file which described the interface specification of the target object (skeleton Hello (B)). The code is automatically generated with this IDL definition file as the starting point.
2	Common IDL	General IDL is used at Common Desktop Infrastructure.
3	Common IDL Converter	The program converts into the IDL of Common Desktop Infrastructure from the target object IDL. This program has to cope with the each component technologies.
4	Target Protocol Convert Mapper Code Generator	The code of Target Protocol Convert Mapper and Stub Hello (B) code used in its inside are generated based on the target object IDL. The IDL compiler of B component generates the Stub Hello (B) code. It is necessary to create this generator for each component technologies.
5	Caller Conversion Service Code Generator	The Stub Hello (A) code and the Skeleton Hello (A) code which became caller, and the Skeleton conversion code which convert request from caller into common protocol is generated based on common IDL. The IDL compiler of A component generates the Stub Hello (A) code and the Skeleton Hello (A) code. In addition, before performing the IDL compiler of A component, it is necessary to convert common IDL into IDL of A component. It is necessary to create this generator for each component technologies.
6	Stub Hello (A) Code	The stub code of the component which becomes caller.
7	Skeleton Hello (A) Code	The Skeleton code of the component which becomes caller.
8	Skeleton Conversion Code	This code inherits Skeleton Hello (A) code, and implements common protocol conversion code.
9	Target Protocol Convert Mapper Code	This receives the request converted into the common protocol, and converts it into the request of the target object. Under the process, Stub Hello (B) code is used.
10	Stub Hello (B) Code	The code of the stub corresponding to the target object.