

PlexService 1.6 User's Guide

THE SOFTWARE PLATFORM PROVIDER



Table of Contents

Table of Contents	2
1. Background.....	4
2 Changing Summary	5
2.1 V1.1	5
2.2 V1.5	6
2.3 V1.6	6
3 Overview.....	8
3.1 System Architecture.....	8
3.2 Legacy Database Access Function.....	13
3.3 PlexKlip Database Access Function	13
3.4 Collaboration Database Access Function	13
3.5 Log Database Access Function	14
3.6 Web Services.....	14
4 Detail Descriptions	15
4.1 System Architecture.....	15
4.2 SOAP.....	20
4.3 XML	20
4.4 COM	21
4.5 PlexService Modules	22
4.5.1 PlexServiceCore	22
4.5.2 PlexAccessor	23
4.5.3 PlexTransformer.....	24
4.5.4 Administrator Tools	29
4.6 PlexService Functions	31
4.6.1 Legacy Database Access	31
4.6.2 PlexKlip Database Access	42
4.6.3 Collaboration Database Access.....	43
4.6.4 Log Database Access	52
4.7 Security.....	53
4.7.1 Authentication before accessing to PlexService.....	53
4.7.2 Access User Control in PlexService	55
4.7.3 Data Security.....	57
4.7.4 Access Control List	58

5	Appendix A: PlexService SOAP Request/Response PayLoad Examples	61
5.1	Query over SOAP/HTTP.....	61
5.1.1	Request.....	61
5.1.2	Response	62
6	Appendix B: PlexService Datatypes	64
6.1	SQL Server and OLE DB Datatypes.....	64
6.2	Oracle and OLE DB Datatypes.....	65
6.3	DB2 UDB and OLE DB Datatypes	65

1. Background

This Document describes the function specification of PlexService. PlexService is the Server developed in K-Plex Inc, and this will provide the function of Database integration, PlexKlip Management and Collaboration Service. PlexService is the server for PlexWare and the upper layer applications. The system will be constructed with them.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

PlexWare, PlexService, PlexKlip and Klip are trademarks under registration of K-Plex Inc.

2 Changing Summary

In this section, the differences between previous version and current version are described. This description is just summary. The more detail descriptions are in following sections.

2.1 V1.1

V1.1 is maintenance version of V1.0, mainly. But some enhancements are included, also. V1.1 unique features are as follow.

- ✓ Enhancement of accessing Legacy Database
Query argument syntax is modified for supporting more detail query function. Group element, Order element, and option element are defined. Column alias and view features are added.
And the format of config file for DB2 is changed. In DB2's config file, schema information can be described. That effect is same as one of Oracle's config file.
- ✓ Support SOAP V1.1 and WSDL (support MS SOAP Toolkit 2.0)
PlexService V1.1 will support Microsoft SOAP Toolkit V2.0. Microsoft SOAP Toolkit V2.0 is adding full support for the Web Services Description Language (WSDL) and SOAP 1.1 Specification. PlexService V1.1 will continue to support Microsoft SOAP Toolkit V1.0 (that means SDL and SOAP 1.0 specification), because PlexWare can handle only Microsoft SOAP Toolkit V1.0 Specification. MS SOAP Toolkit V1.0 and V2.0 can be co-existing.
- ✓ Some new interfaces
Additional new interfaces to access PlexKlip Database and Collaboration Database are defined and implemented. These interfaces are advanced implementation as they will be used by PlexWare V2.0.
- ✓ Changing of PlexKlip and Collaboration Database schema
Lock field was reserved in V1.0. In V1.1, datatype of this field will be changed. This modification is preparation for V2.0. The Locking function will be provided in V1.1, but PlexWare will use these functions in V2.0.
- ✓ Some bug fixes

2.2 V1.5

V1.5 is enhancement version of V1.1. The enhancement is focused on accessing functionality of Legacy Database. But other modifications exist as following.

- ✓ Enhancement of accessing Legacy Database:
Now Query function supports the following additional functions:
 1. The function to list alias, synonym and stored procedures
 2. The function to call Stored Procedures
 3. Update function to Legacy Database
 4. Implicit Transaction Control (This feature supports 2 Phase Commit.)
 5. UnifyTable can have the relationship between tables
 6. RowCounter function on request to get records
- ✓ Accessing function to Active Directory:
The function to retrieve Group and User information in previous version accessed to the machine's registry. Now same functions can access to Active Directory in order to retrieve group and user information. (If users want to use this feature, administrator must set value of registry key: [\\HKEY_LOCAL_MACHINE\\SOFTWARE\\K-Plex\\PlexService\\RootContext](#). Otherwise, PlexService will never access to Active Directory.)
- ✓ Some new interfaces:
Additional group/user retrieving functions are defined.
- ✓ Some bug fixes

2.3 V1.6

V1.6 is enhancement version of V1.5. The enhancement is focused on accessing functionality of Klip Resource Database. But other modifications exist as following.

- ✓ Enhancement of accessing Klip Resource Database:
The interface of Klip Resource Database accessing was changed from V1.5. V1.5 Klip Resource Database interface became obsolete.
- ✓ Internal logic was changed:

PlexServiceV1.6 User's Guide

- rope.dll was refined to solve memory leak problem.
- PlexServiceCore.dll and PlexAccessor was refined to tune up performance
- ✓ Some new interfaces:
Additional interface to retrieve Conference Manager URL was defined.
- ✓ Some bug fixes

3 Overview

In this chapter, the overview of PlexService functions is described. Firstly, the abstract of architecture will be described. After that, each function provided by PlexService, that is, Legacy Database Access function, PlexKlip Database Access function, Collaboration Database Access function and Log Database Access function, are described. This chapter will provide just only abstract information of PlexService. You can understand the detail description in later chapter, if you read this chapter.

PlexService provides Programming interface for the client application based on SOAP architecture. PlexService's handling data is XML. These technologies are familiar with .NET proposed from Microsoft. PlexService can be one building block of .NET architecture. Finally, relationship of PlexService and Web Services are described.

3.1 System Architecture

PlexService will accept the Web access. So, PlexService will be one of Web Services. PlexService has the external interface based on SOAP. So, PlexService has the component of SOAP Listener. The detail information of PlexService's SOAP interface is described in the document called as "PlexService SDK Specification (KP-001-02-002)". If you want to know it, see this document.

SOAP Listener will call PlexService's COM interface depending on client request content in order to execute PlexService functions. PlexService uses Microsoft SOAP SDK for implementing SOAP Listener. Currently, there are ASP version and ISAPI version. PlexService uses ROPE (Remote Object Proxy Engine), also. But client applications should not use ROPE.

If client applications can generate and send SOAP request, they can use PlexService through SOAP interface.

On the other hand, as PlexService is COM components, the modules that can use COM interface (for example, ASP) can use PlexService directly.

The following figure will show each case.

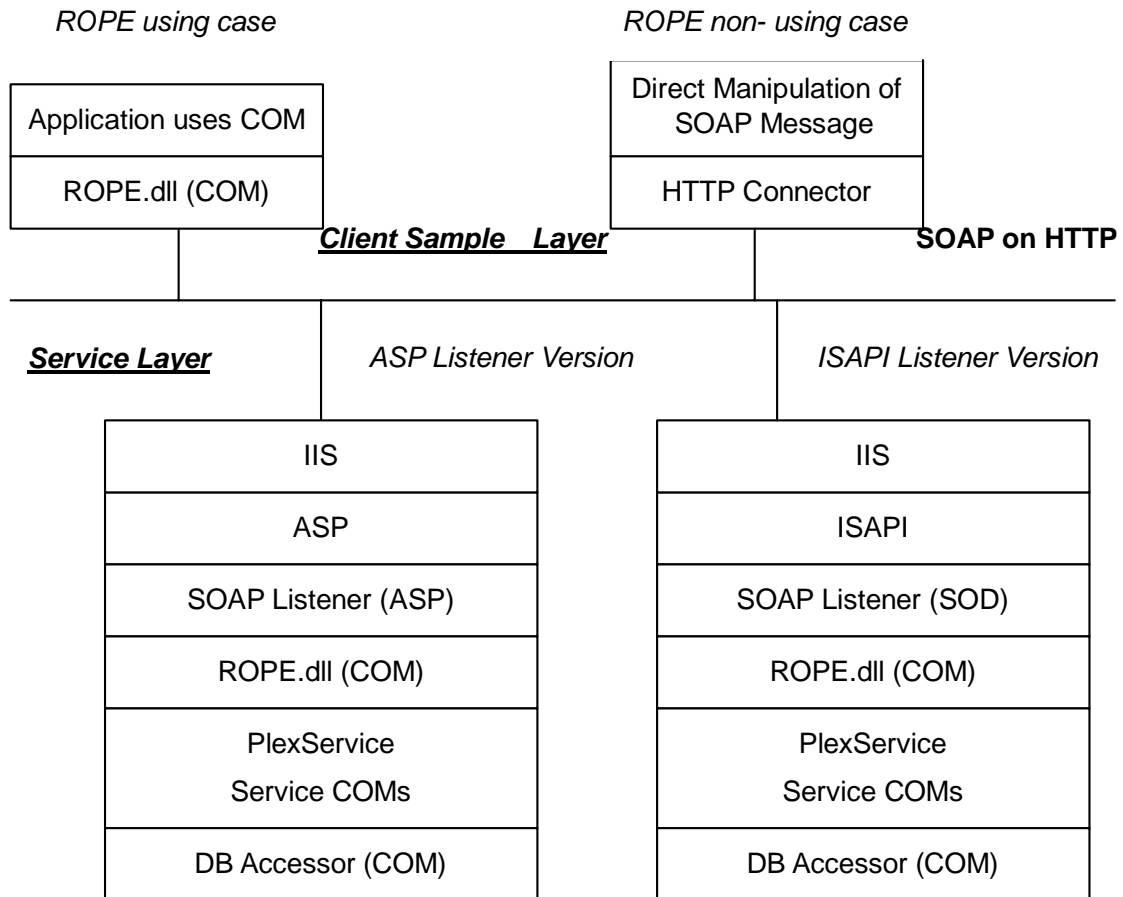


Figure 1 PlexService SOAP Architecture

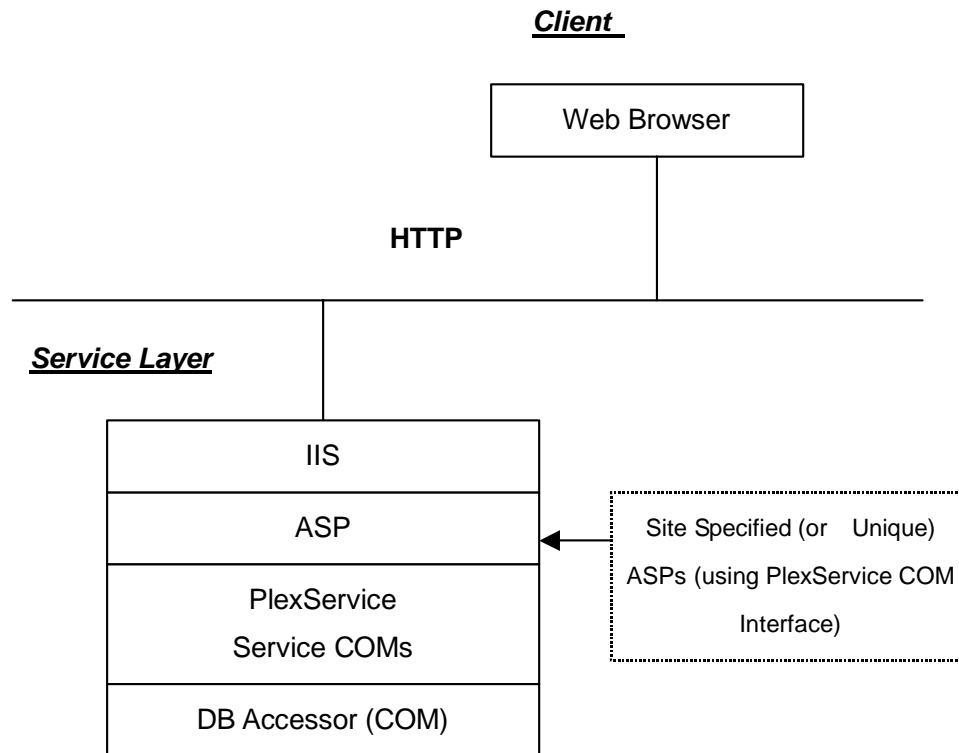


Figure 2 PlexService and ASPs

The technology related with PlexService external interface is as above.

The next figure shows the more detail structure of PlexService Service COMs.

PlexServiceCore: This module will do mapping between external COM interfaces and PlexService functions mainly. And the Unifier function that merges the searching result tables and the management function of the Active Sharing Field, that is, Collaboration function will be provided by this module.

PlexAccessor (or Master DB Accessor): This module will access to the Legacy Databases (Customer's Databases) and generate PlexService unique XML format data from the searching result. PlexService uses MSXML for generating and parsing XML. And PlexService uses OLE DB native interface for accessing databases. So, PlexService can connect to databases that provide OLE DB Provider basically. But some confirmation will be required that there is no problem in accessing databases, even if they provided OLE DB Provider.

(PlexService can't use OLE DB Provider in some cases, because OLE DB Provider does not support the PlexService's using interface. OLE DB definition interface has some levels.) Currently PlexService can connect to SQL Server 2000, Oracle 8i R8.1.6 and DB2 UDB V7.1 as the support databases. PlexAccessor will provide the access function to PlexKlip Database/Collaboration PlexKlip Database/Log Database, also. These databases will be constructed on SQL Server 2000.

PlexAccessor can access Web Pages. If user specify request-URI, PlexAccessor will retrieve (get) that Web Page.

PlexTransformer: This module is the converter from PlexService Unique XML Format Data to another XML format. Users and applications can get the required XML format data through this module.

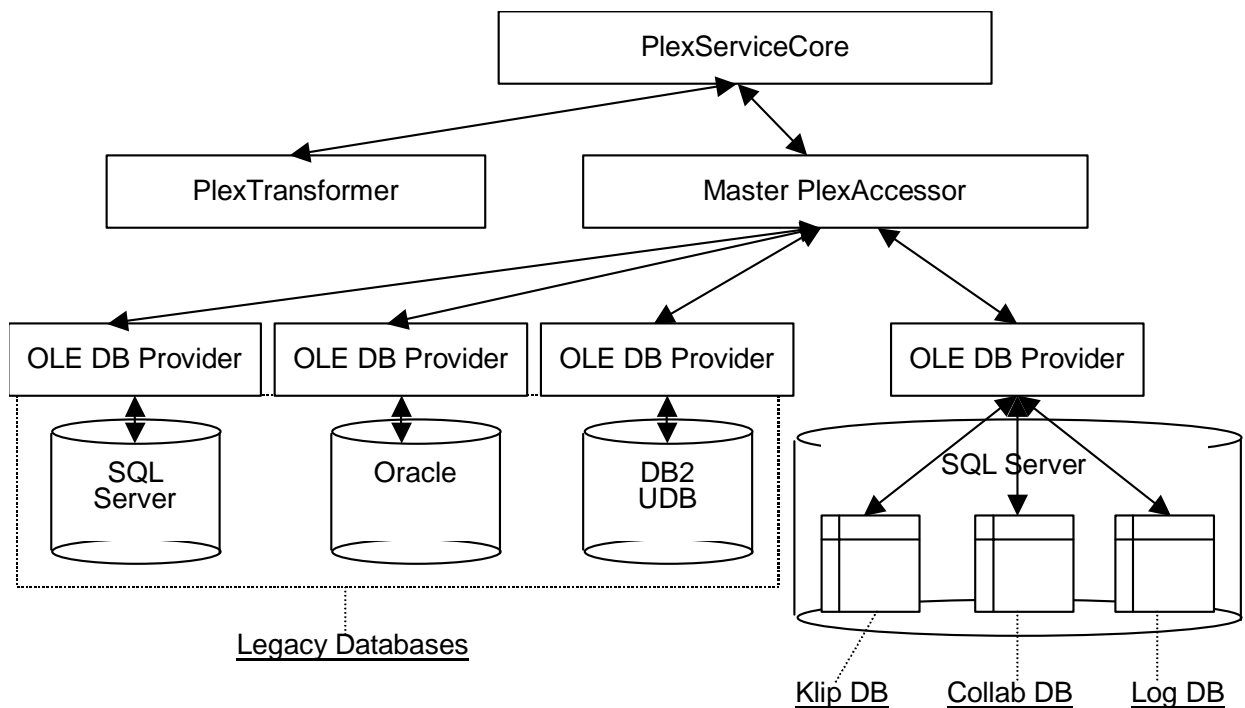


Figure 3 PlexService Function Modules

PlexAccessor and PlexTransformer are implemented with COM technology. So, they are the pluggable modules. PlexAccessor. Users can develop and add new PlexAccessor(DB Accessor) and PlexTransformer that will provide the required

functions. If you want to develop and add these modules, you must refer to the document “PlexService SDK Specification (KP-001-02-002)”. This specification has the information for developing and adding new modules.

In the following figure, the shaded boxes are new plug-in additional modules.

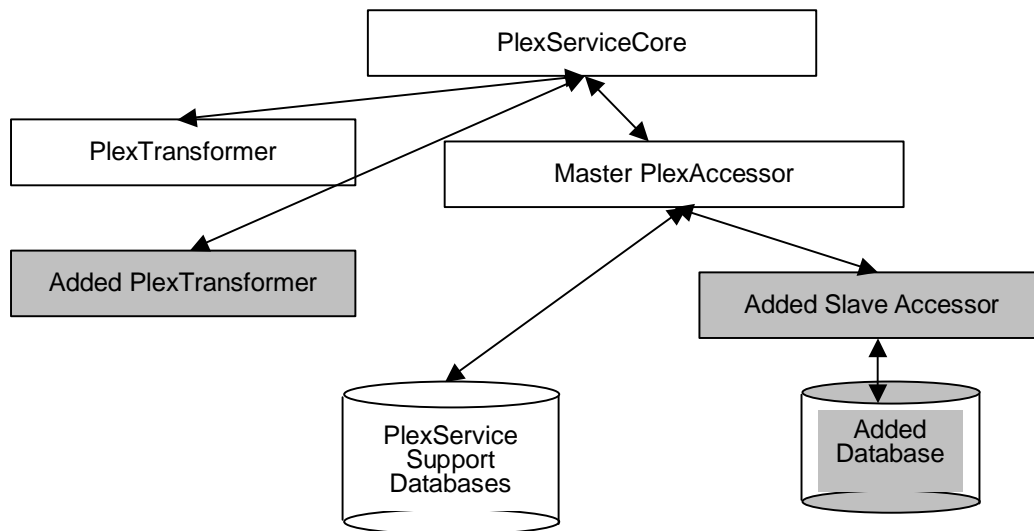


Figure 4 PlexService and Plug-in Modules

PlexAccessor will support two phase commit method as the transaction control for handling multiple databases seamlessly. So, PlexAccessor must send two phase commit control to new plug-in DB Accessors. In PlexService, they are constructed as Master/Slave configuration. Master/Slave means PlexAccessor that provides as the PlexService module is called as Master, and DB Accessor that is developed as new plug-in module is called as Slave. Such structure may allow to send the two phase commit control. But the database connected to DB Accessor must support the transaction function (commit/rollback function).

On the other hand, focusing PlexService functions, there are four major categories, as follow:

1. **Legacy Database access Function**
2. **PlexService Klip Database Access Function**

3. PlexService Collaboration Database Function

4. PlexService Log Database Access Function

In the following section, each categories function will be described.

3.2 *Legacy Database Access Function*

PlexService offers the function which accesses two or more databases which already exist in a user site unitary. It is possible to obtain a feeling of use with which the databases which existed until now is unified by one database by using this function of PlexService. At this time, correction and change are unnecessary to the data of the existing database. So, PlexService Legacy Database Access Function will provide the table unification function and the XML transformation function.

3.3 *PlexKlip Database Access Function*

PlexService offers the function to save PlexKlip, which PlexWare handles. Preservation management of the PlexKlip is carried out at SQL Server, which is a database peculiar to PlexService. This is called PlexService PlexKlip Database in these specifications. As an actual database, it consists of two kinds of databases called PlexKlip Content Database and PlexKlip Resource Database, XML data of PlexKlip is saved at PlexKlip Content Database, and the resource relation of PlexKlips, such as DLL, is saved at PlexKlip Resource Database.

3.4 *Collaboration Database Access Function*

PlexService offers Collaboration Field (referred to as Sharing Field) for PlexKlip. Two or more users refer to the same PlexKlip simultaneously by this, or a user becomes possible [sharing PlexKlip owned individually exhibiting]. Preservation management of the Sharing Field is carried out at SQL Server which is a database peculiar to PlexService. This is called PlexService Collaboration Database in these specifications.

3.5 Log Database Access Function

PlexService offers the log function to save the received request. Moreover, the function for application leaving a log clearly to service is also offered. Preservation management of these logs is carried out at SQL Server, which is a database peculiar to PlexService. This is called PlexService Log Database in these specifications.

3.6 Web Services

PlexService will provide the functions of Web Services that are not depend on intranet or internet. PlexService already uses SOAP and XML technologies that are key technologies for Web Services implementing. And as PlexService uses ROPE that is Microsoft SOAP SDK component, PlexService uses and provides SDL (Service Definition Language), also.

Currently, although it is not busy about WSDL and UDDI which are advancing standardization, it is the stage in which a certain amount of draft solidified, and the usage is due to be examined.

In Microsoft .Net project, it is the framework to integrate Web Services. So, PlexService can become one of Building Blocks that are integrated by .NET framework.

4 Detail Descriptions

In this chapter, the detail PlexService information will be described.

4.1 System Architecture

PlexService is the Server that will provide the integration of the existing databases and new services. It is one function of PlexService that defines the relationship of the individual databases and provides one new database schema that is integrated with the connected databases. It is not necessary to modify the data of the existing databases. And it is not necessary to migrate the existing database to other machines, also.

For example, if all databases, that are, SQL Server 2000, Oracle 8i R8.1.6 and DB2 UDB V7.1 are installed in individual machines, PlexService can connect them as the following figure.

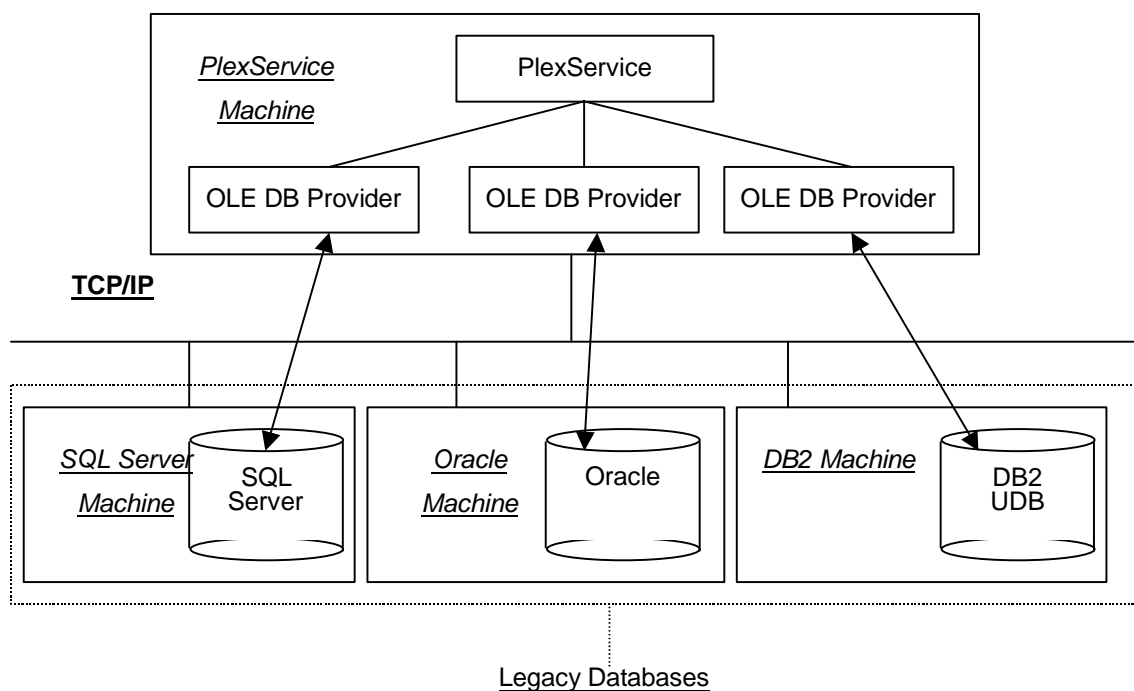


Figure 5 Isolated Databases Configuration

Moreover, when all databases are intermingled in one machine contrary to this, connecting with PlexService is possible.

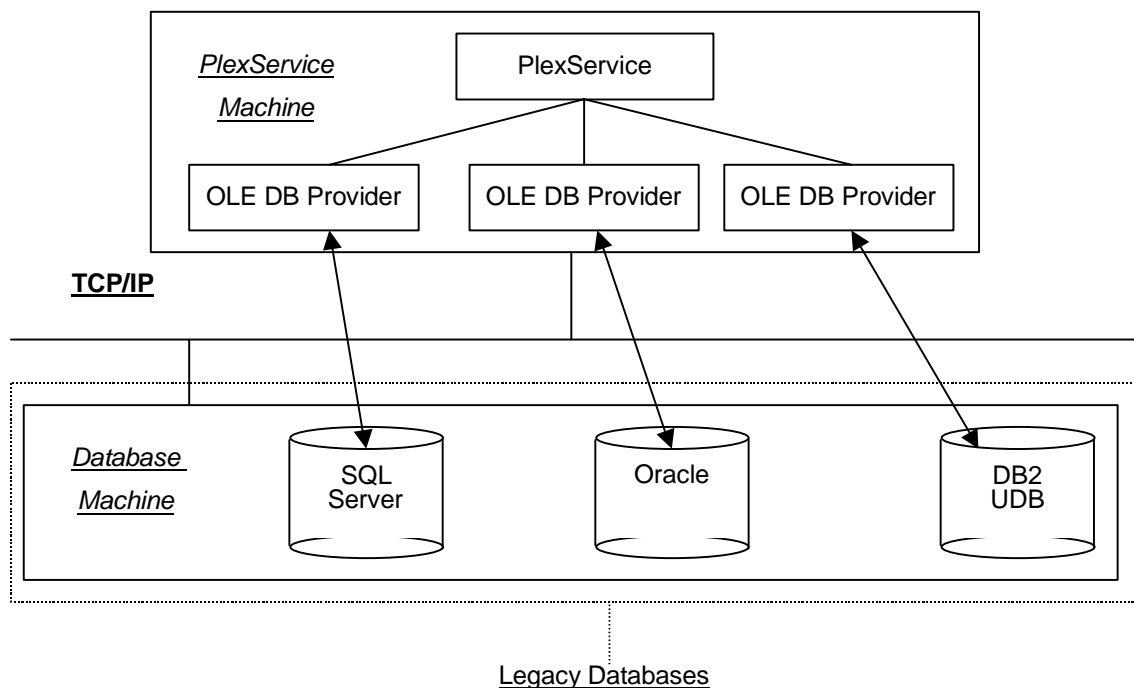


Figure 6 Some Databases on Single Machine

Thus, when installed in the machine by which PlexService is different in Legacy Database, it is necessary to install OLE DB Provider of each database in the machine by which PlexService is installed.

In the case of all modules (that is, databases and PlexService) installing in the same machine, the separate installation of each database's OLE DB Providers will not be required.

The following figure shows all modules and all files of PlexService and all relationship of them that are described in overview section.

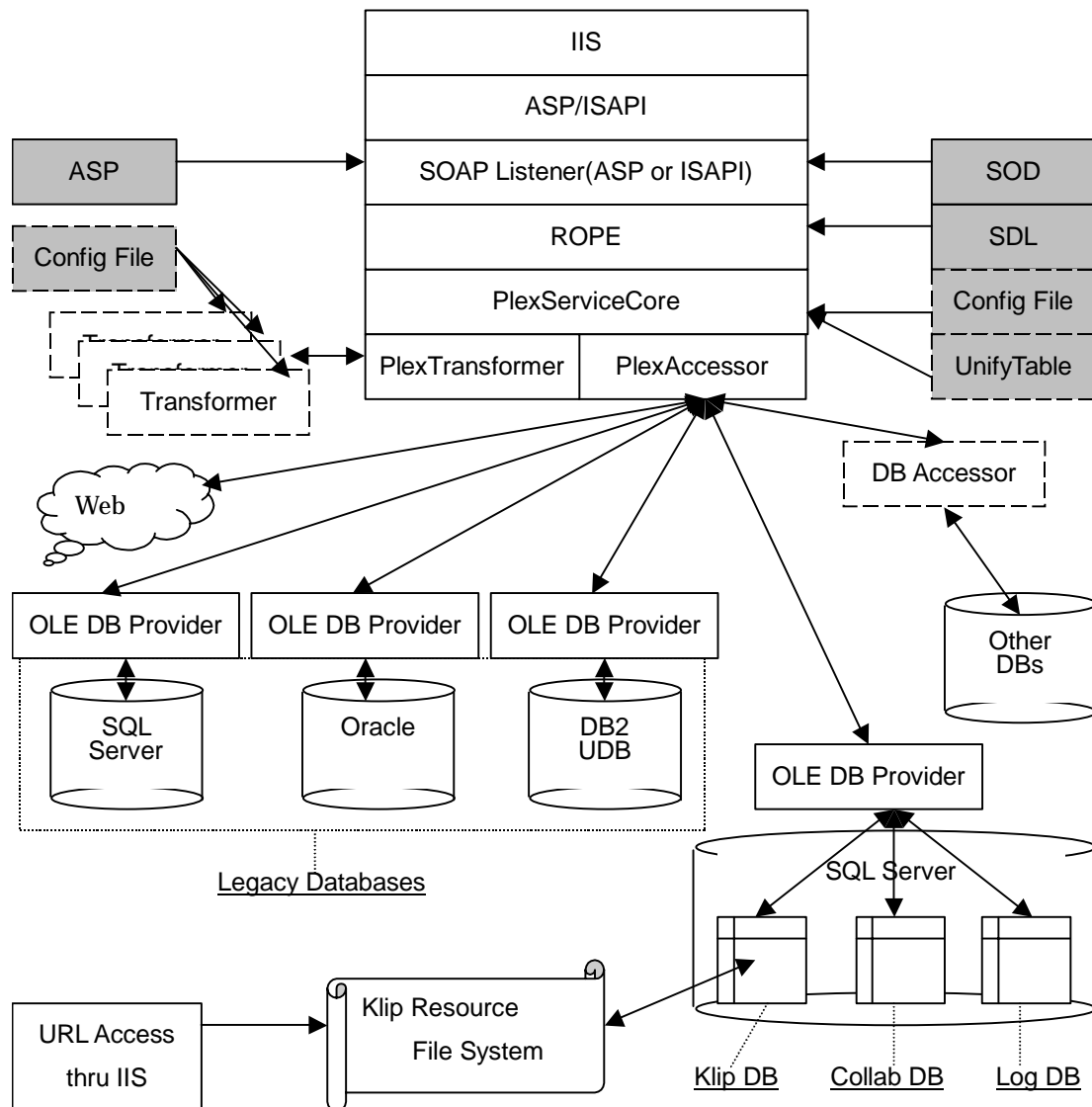


Figure 7 PlexService Overall Blocks

In the above figure, the shaded line boxes are the system files that provided with PlexService. And the shaded dash-line boxes mean the configuration files that are generated by Administrator Tool with each site configuration. The dash-line boxes mean the plug-in modules. They will be developed and added in the user sites.

Each blocks meaning is as follow:

IIS: PlexService will accept the requests through IIS that is Web Server. IIS will send the accepted request to ASP or ISAPI Module, that is, SOAP Listener.

SDL (Service Description Language) file will have the information, which is available, ASP or ISAPI module on that environment.

ASP/ISAPI: ASP or ISAPI module is for SOAP Listener. IIS will load them when required.

SOAP Listener: SOAP Listener will parse the request and the SDL file and check that the request is valid or not, using the content of SDL file. ROPE (Remote Object Proxy Engine) is used for interpreting SDL and packaging SOAP Request/Response. SOAP Listener will load ASP file or SOD file depending on its implementation. If you want to use utf-8 code set in SOAP request and response, you must use ISAPI version SOAP Listener.

PlexServiceCore: This is the core module of PlexService. This will receive the request that accepted by SOAP Listener, and distribute it to the adequate PlexService other modules. This module will provide the table unification function, also. This module is implemented by using Microsoft COM technology, so this will be invoked by calling COM Interface method.

PlexAccessor: This is the PlexService's module to access to databases. This module is implemented by using OLE DB technology. This will access to the Legacy Databases. And this will access PlexKlip Database, Collaboration PlexKlip Database and Log Database that are PlexService own databases, also. DB Accessors that will be developed for each user sites will plug-in to this, so PlexAccessor will control them. "Config File" box in the right side of the above figure means the configuration file for PlexAccessor and DB Accessor. PlexServiceCore uses this configuration file for recognizing Accessor modules and using them.

PlexTransformer: This is the PlexService's transform module. This will convert one XML to another XML. As the XML data that PlexServiceCore will generate is the unique XML format including Legacy Database's column information, the conversion methods to other XML format will be required. PlexTransformer will support this conversion in PlexService. As PlexTransformer is plug-in modules, new transformers can be developed based on the user site requests and added to PlexService flexibly. Some transformer modules can be added to PlexService. "Config File" boxes in the left side of the above figure means the configuration file for PlexTransformer. PlexServiceCore uses this configuration file for recognizing transformer

modules and using them.

OLE DB Provider: PlexAccessor will connect to databases using OLE DB Provider that will be provided by each database vendors. OLE DB Provider will be installed with the database or the database access client. If PlexService is installed to the machine that doesn't have database, OLE DB Provider must be installed to that machine to connect the database.

Legacy Databases (SQL Server/Oracle/DB2 UDB): In this version (Version 1.0), PlexService will support SQL Server 2000, Oracle 8i R8.1.6 and DB2 UDB V7.1 as the connectable databases. PlexService can integrate these databases, if PlexService can access to them by network. (PlexService can access Microsoft Access files with OLE DB provider called as "Microsoft.Jet.OLEDB.4.0". But currently there is no schedule to make it as official support Database.)

Klip DB: PlexKlip Database will consist of two Databases called as PlexKlip Content Database and PlexKlip Resource Database. PlexKlip itself will consist of XML data and DLL (in some case, it will be called as Resource). XML Data of PlexKlip will be stored to PlexKlip Content Database. Resource of PlexKlip (for example, DLL file) will be stored to PlexKlip Resource Database. PlexService will provide each access interfaces as the programming interface. So, Application can access them by these interfaces. PlexKlip Database will be constructed on PlexService own SQL Server.

Collab DB: Collaboration Database is for storing and managing the PlexKlip for Collaboration. As mentioned previously, PlexKlip will consist of XML data and DLL. The Collaboration Database will manage only XML data of PlexKlip. The Resource of PlexKlip for Collaboration will be stored to the PlexKlip Resource Database. Collaboration PlexKlip will have the status. There are 2 status called as static and active. Static status means nobody attends to collaboration. Active status means someone attend to collaboration. PlexService will manage these statuses. Collaboration database will be constructed on PlexService own SQL Server.

Log DB: PlexService will store all log information to Log Database. PlexService will provide the access interface to Log Database. So, Application can store and retrieve loges to/from Log Database. If users want to analyze log information, application can use PlexService Log Database. Log Database will

be constructed on PlexService own SQL Server.

Klip Resource File System: PlexKlip will be stored to PlexKlip Content Database and PlexKlip Resource Database. PlexKlip Resource Database will manage Resource's URN (URI) and URL information only. The resource data (for example, dll file) are stored to PlexService Local File System (in Windows 2000, NTFS). PlexService will provide the Programming Interface to get the URLs of these objects. Application can access these object using these URLs on HTTP.

4.2 SOAP

SOAP (Simple Object Access Protocol) can be caught as RPC which made XML the used language. PlexService is designed by the policy of unifying all the data with the exterior to XML, and SOAP conformed most as a communication protocol. From this reason, PlexService adopts SOAP as an external interface protocol. Furthermore, it uses in the form of SOAP on HTTP from the viewpoint of Web Services. The HTTP itself is a general-purpose protocol when passing Fire Wall, and since it conforms to PlexService, it has been adopted.

The SOAP itself has only specified the format, which rides on HTTP. For this reason, it is necessary to use a format of SOAP and to specify the original external interface of PlexService. This external interface is described in the specification "PlexService SDK Specification (KP-001-02-002)". This is the only one provided method that the external modules (for example, client applications) can use.

4.3 XML

Basically, PlexService's handling data is XML. Although Legacy Database data is table data, PlexService will generate XML based on the searching result table data. There are many varieties on XML format. PlexService will generate the original XML data. The detail description of that format is defined in the specification "PlexService SDK Specification".

PlexService will provide the plug-in mechanism, called as PlexTransformer, to

add the transformer from one XML to another XML, as some applications can not accept the PlexService original XML format. Users can use this mechanism to add the transformer from PlexService original XML to OFX, XBRL and so on.

Thus, in PlexService, all the data to generate serves as XML format, and is sent to Client Application by SOAP. Moreover, all setting files, such as Config File, which PlexService reads, also consist of an XML format.

XML processing (an interpretation and generation) of PlexService uses MSXML of Microsoft. Since it always exists in the environment where Internet Explorer of Windows is installed, it is not necessary to newly install this module.

4.4 COM

All modules of PlexService are implemented with COM technology. They are the COM components. PlexService's plug-in mechanism is used COM's late binding mechanism, so PlexService can support the plug-in in the runtime environment. And the modules that can call COM components (for example, ASP) can use PlexService, because the modules of PlexService are COM components.

The other client applications can use PlexService's Legacy Database access function with ASP, as it function is individual from the PlexKlip of PlexWare. ASP can use PlexKlip Database access function and Collaboration Database access function that are related with PlexKlips, also. But it will be not so meaning for the client applications that don't know how to handling PlexKlip

The information to develop DB Accessor (that is, the user developed modules provides the same meaning functions of PlexAccessor) and Transformer (that is, the user developed modules provides the same meaning functions of PlexTransformer) is described in the specification "PlexService SDK Specification (KP-001-02-02)". If you want to know it, please refer that specification.

4.5 PlexService Modules

In this section, each module of PlexService will be described. PlexService will consist of the following modules.

- ✓ PlexServiceCore
- ✓ PlexAccessor
- ✓ PlexTransformer

And the administrator tools to generate the system configuration files for PlexService and the UnifyTable for PlexService's table merging function will be provided, also.

4.5.1 PlexServiceCore

PlexServiceCore is the module that will accept all requests to PlexService. This module will interpret requests and distribute them to the appropriate modules. PlexServiceCore Interfaces is PlexService external interfaces. These interfaces are described in PlexService SDK Specification. PlexServiceCore will accept the requests from SOAP Listener. PlexServiceCore is implemented with COM technology, so the application that can call with COM Interface can use PlexServiceCore Interface, also. The following figure will show.

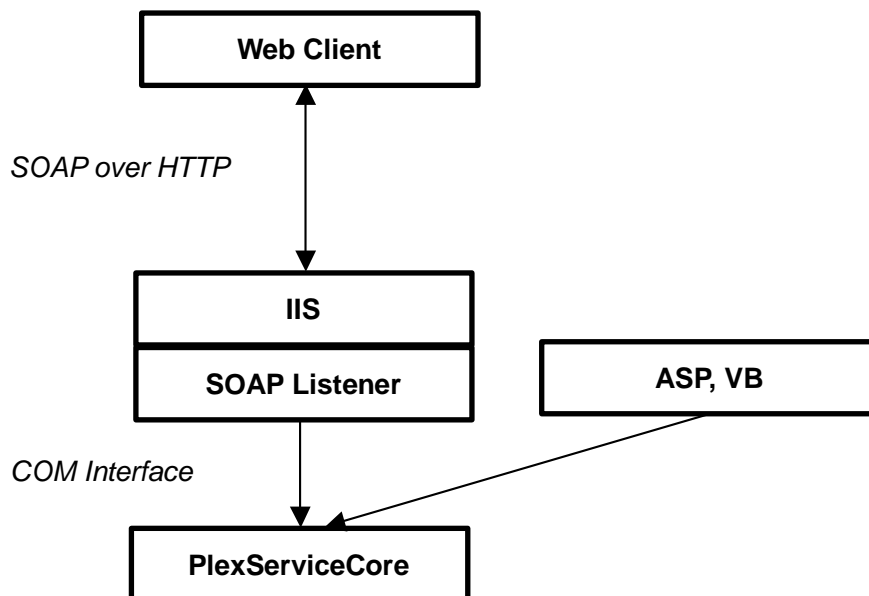


Figure 8 PlexServiceCore Interface

PlexServiceCore will unify (or merge) the each table of PlexAccessor's result according to UnifyTable file information. The unification means to generate one table from the some tables. The unification rules and unification mode are described in the UnifyTable file. The UnifyTable file can be generated for each table on the Legacy Databases. PlexServiceCore will load these files and do the specified unification. The detail description of the unification is described later.

4.5.2 PlexAccessor

PlexAccessor is the Database access module of PlexService. This module will provided as PlexService Release Set. Currently this accessor can access SQL Server 2000, Oracle8i R8.1.6 and DB2 UDB V7.1. In PlexService Specification, these databases are called as the Legacy Database. The Legacy Database means customer's database. PlexService will be installed and integrated with no modifications on the data of these databases.

PlexAccessor is implemented with using the Microsoft OLE DB Technology rather than ADO. So, PlexAccessor can access the databases supporting OLE DB Provider. But as the properties initialization of OLE DB Provider is very nervous, PlexService will define the support databases if the data handling test will have no problems. Currently PlexAccessor can access Microsoft Access File (*.mdb) without problems. If customer site will use Microsoft Access, PlexService can be integrated their data.

The following figure will show the PlexAccessor mechanism. PlexAccessor itself is implemented as COM component. So, PlexService can load depending on the necessity. PlexAccessor will use OLE DB interface when accessing OLE DB Providers. Therefore, PlexAccessor can access the databases supporting the OLE DB Provider. If users want to integrate the other Databases that don't support OLE DB Provider, users can develop new DBAccessor that will access the required Databases. PlexAccessor can load the developed DBAccessors depending on the necessity. The DBAccessor must be COM component, also. There is the guideline to develop the DBAccessor. This information is described in PlexService SDK Specification.

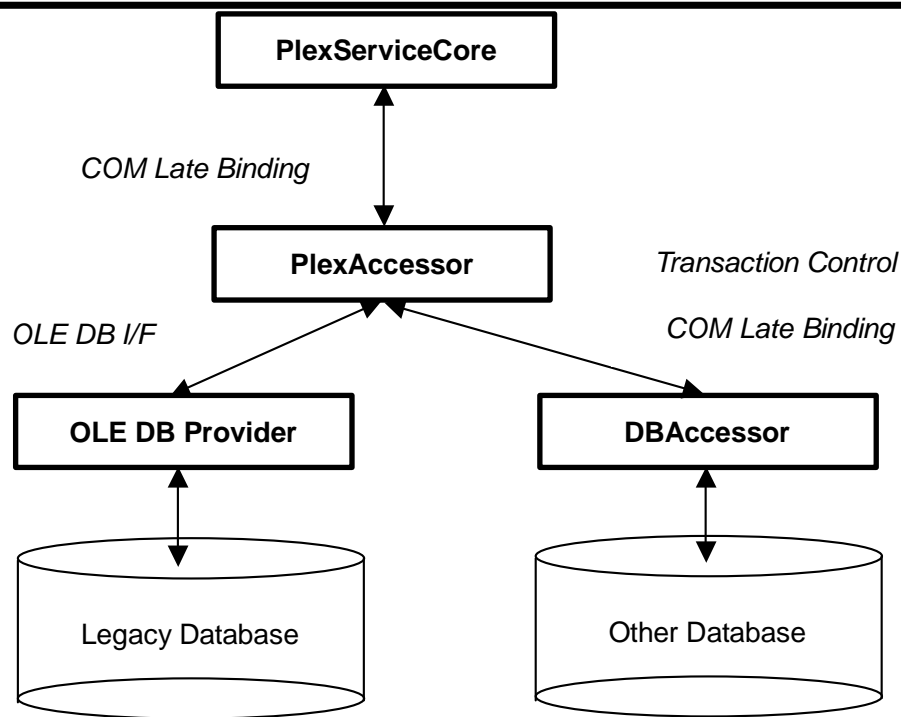


Figure 9 PlexAccessor Interface

PlexAccessor can access the Web contents, also. This feature is isolated from Database accessing functions. PlexAccessor will return XML format always. So, when the PlexAccessor will return the Web content, it will be the following format.

```

<root>
  <response>
    [Web content as text]
  </response>
</root>
  
```

[Web content as text] is response tag's one text value, as XML parser can not parse HTML variety. Users and application must take care about it.

4.5.3 PlexTransformer

PlexTransformer is the plug-in converter from the PlexService original XML format that will be generated from the searching result tables by PlexAccessor

or PlexServiceCore to other XML format. It is possible to register two or more plug-in to PlexServiceCore, and users can use two or more PlexTransformers specifying in the argument of Query function. Then the PlexServiceCore will send the one PlexTransformer's result to other PlexTransformer as its input.

Fundamentally, as shown in the following figures, PlexTransformer simple substance considers Source XML as an input, and considers Converted XML (XML which changed Source XML in accordance with a certain rule) as an output. When it is necessary to enable it to change the rule of conversion dynamically, the information input of an option called Rule is possible. An input / outputs XML and Rule are altogether treated as a character sequence.

For this reason, as for the value of Rule, it is possible that it is also the character sequence which shows URL on Web, that it is also the path name which shows a file, and for it to be also the character sequence which shows value. The method of treating the value of Rule is completely left to Transformer mounted.

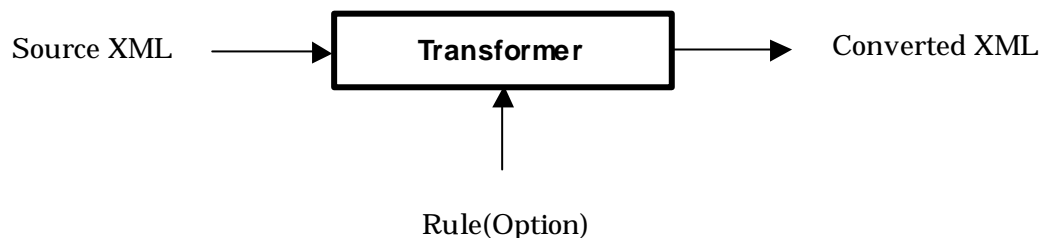


Figure 10 Transformer's Input & Output

PlexTransformer is implemented as COM component. The user developed Transformer must be COM component, also. So, PlexServiceCore will execute Transformer that specified by the argument value of Query Function using COM's late binding mechanism that enable to dynamic loading of COM DLL and executing it.

PlexTransformer consists of the following 2 files.

- ✓ Transformer COM DLL

✓ Transformer Config File

PlexServiceCore will use these files as the following figure manner in order to make the Transformer's functions available.

New developed Transformer must consist of the same meaning files.

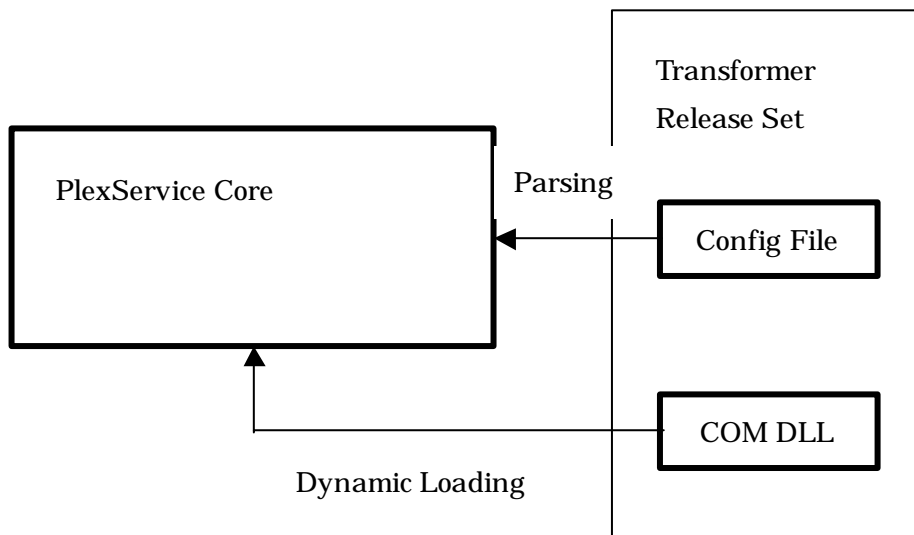


Figure 11 PlexServiceCore & PlexTransformer

The each file will be described in the following sections.

4.5.3.1 PlexTransformer COM DLL

PlexTransformer COM DLL must be implemented as the COM component that has just only one interface and one method. The method name must be "Convert", and this method is the trigger to invoke the conversion process. This method is the only one programming interface to connect PlexServiceCore and PlexTransformer. The following figure will show the relationship of PlexServiceCore and PlexTransformer. This mechanism will be same as one of the new developed Transformer.

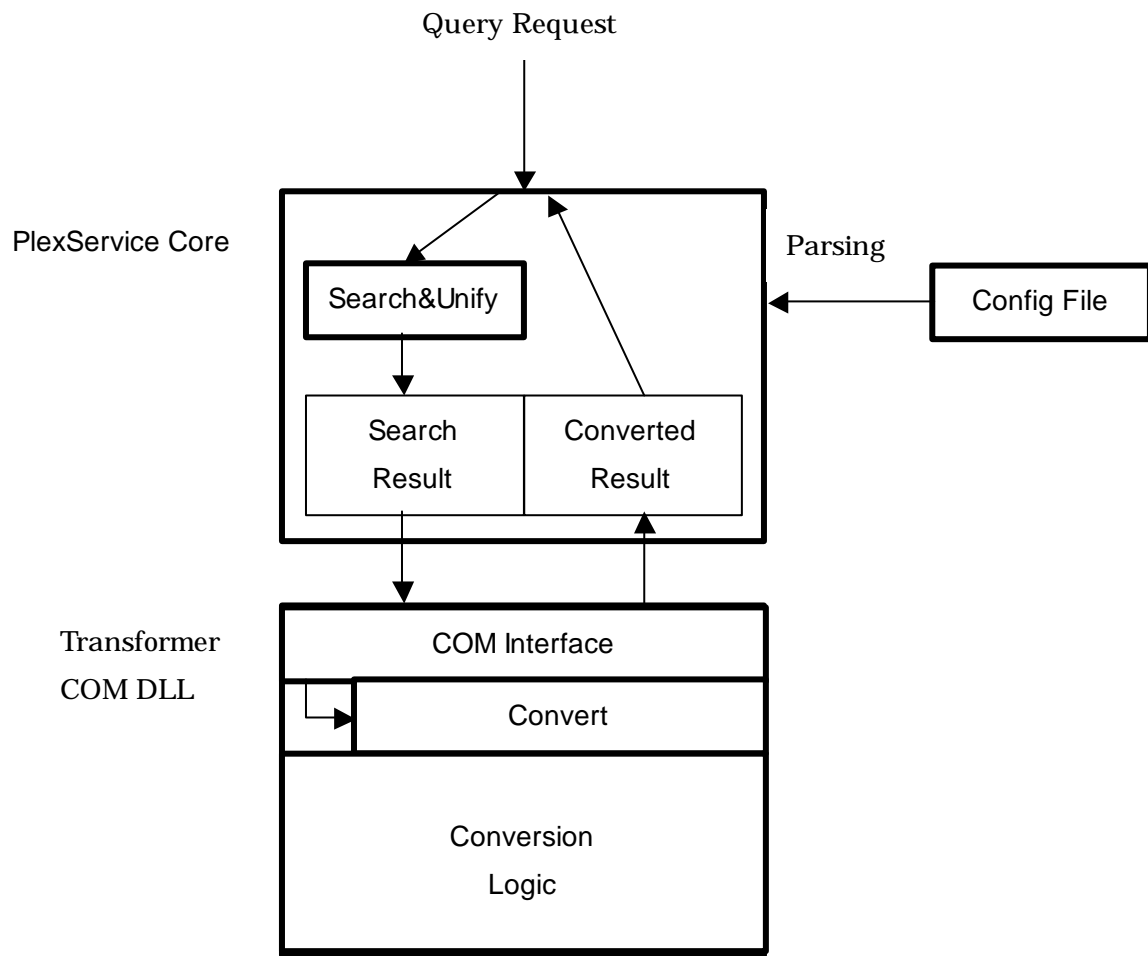


Figure 12 Transformer Block

4.5.3.2 PlexTransformer Config File

The configuration file of Transformer will have the information that will be used by PlexServiceCore to find and execute the appropriate Transformer. The developer of Transformer must have the responsibility to provide the configuration files and executable modules, as the only developer knows these information. The configuration file of PlexTransformer will be included to the system release set.

The syntax of Transformer's configuration file is as follow.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Transformer Config format XDR file -->
```

```
<Schema name="transformer.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">

<ElementType name="PlexTransform" content="eltOnly" order="seq">
  <AttributeType name="complete" dt:type="enumeration" dt:values="true
false" default="true"/>
  <attribute type="complete"/>
  <element type="transform" minOccurs="1" maxOccurs="1"/>
</ElementType>
<ElementType name="transform" content="eltOnly" order="seq">
  <AttributeType name="complete" dt:type="enumeration" dt:values="true
false" default="true"/>
  <attribute type="complete"/>
  <element type="ProgID" minOccurs="1" maxOccurs="1"/>
  <element type="Method" minOccurs="1" maxOccurs="1"/>
  <element type="Source" minOccurs="1" maxOccurs="1"/>
  <element type="Rule" minOccurs="1" maxOccurs="1"/>
  <element type="Result" minOccurs="1" maxOccurs="1"/>
  <element type="Description" minOccurs="1" maxOccurs="1"/>
</ElementType>

</ElementType name="ProgID" content=textOnly" />
</ElementType name="Method" content=textOnly" />
</ElementType name="Source" content=textOnly" />
</ElementType name="Rule" content=textOnly" />
</ElementType name="Result" content=textOnly" />
</ElementType name="Description" content=textOnly" />

</Schema>
```

The value of tag “ProgID” must be ProgID of Transform COM. The value of tag “Method” must be the method name of Transformer that has just only one method. As previous mentioned, this must be “Convert” basically. For a certain reason, although it recommends making it such, when this method name cannot be exhibited, it is possible to give another name and it must describe as

value of Tag "Method" then. Tag "Source" and Tag "Result" are reserved now. Especially the purpose of use is not specified. Since it has specified for extension of the future, it does not have value. Tag "Rule" must have regulation value as value, when Transformer with Rule Option is created. In Transformer without Rule Option, it does not have value. Tag "Description" must have a character sequence explaining the function of Transformer as value. An example is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<PlexTransform>
  <transform>
    <ProgID>TransformOFX.TransOFX</ProgID>
    <Method>Convert</Method>
    <Source></Source>
    <Rule></Rule>
    <Result></Result>
    <Description>This Transformer converts PlexServiceUnified XML to
OFX XML</Description>
  </transform>
</PlexTransform>
```

And the extension of configuration file must be “.xml”. PlexService will recognize just only files stored in appropriate path and has “.xml” extension as the configuration file.

4.5.4 Administrator Tools

In PlexService installation, the site information is required. The configuration file will have this information, and PlexService can recognize this information adequately. PlexService Administrator Tool will support the creation of Config files and environment setup.

4.5.4.1 PlexService Administrative Tool

One of AdminTool function is the generator of PlexService Configuration files. This tool will connect to the Legacy Databases and generate the adequate

configuration files. This tool will generate the UnifyTable, also. If users want to unify table with the UnifyTable, users must generate the UnifyTable that has the users' required unify rules. So, users must connect the Legacy Database using this tool and input column conversion information.

This tool will support generating PlexService Configuration files and UnifyTable files.

PlexService will check the home directory (HomeDir) setting when firstly running. Configuration files and UnifyTable files must be placed in the subdirectory of HomeDir. The configuration files must be placed in %HOMEDIR%\Config directory. The transformer configuration files must be placed in %HOMEDIR%\transform directory. The UnifyTable files must be placed in %HOMEDIR%\unifier directory. If the Home Directory setting is invalid, PlexService can not run correctly. This information will be stored in Windows Registry. AdminTool can set this information. If you don't have this tool, you can set with Windows RegEdit Tool.

And the other key values for PlexService are existing.

LogLevel will indicate PlexService's Log Content Level. Its value must be None, Normal, Detail, or Debug.

If users use KLIP Resource Database, ResourceDir and ResourceURL must be specified.

The Registry HUB is as follow.

Registry HUB name	Meaning
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\HomeDir	PlexService's Home Directory
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\LogLevel	PlexService's Log Level
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\ResourceDir	The directory to store Resource Objects
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\ResourceURL	The access URL to Resource Objects
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\AdminUsers	PlexService's Administrator Username List
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\UserGroup	PlexService's Group List (local definition)
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\CompressThreshold	Compress Threshold value on PlexService's reply.
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\RootContext	RootContext on ActiveDirectory
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\AdminGroupName	AdminGroup name on ActiveDirectory
\\HKEY_LOCAL_MACHINE\SOFTWARE\K-Plex\PlexService\ConferenceManagerURL	Conference Manager URL

4.6 PlexService Functions

4.6.1 Legacy Database Access

In this section, PlexService's Legacy Database Access Function is described. Legacy Database Access means the function to access the already existing user databases. OLE DB native functions are used for accessing to the databases. This means that PlexService can connect to all databases that support OLE DB Provider. But there are the connectable databases and the unconnectable databases depending on OLE DB Provider's support interface level even if the OLE DB Provider is provided from database vendor.

Legacy Database Access function has 3 major features.

- ✓ Ambiguous query feature
- ✓ Unify tables feature
- ✓ Result transform feature

Ambiguous query feature: PlexService will accept the ambiguous query requests. PlexService will connect to some Legacy Databases. But users can find the required tables without the specifying of databases. When PlexService receives such requests, PlexService will query to all connected databases and return searching result.

The "Query" function will provide Legacy Database access function. The varieties of access are controlled by its argument value. The definition of the "Query" function's argument is as follow.

Query.xdr:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- query format XDR file -->
<Schema name="query.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
```

```
<ElementType name="query" content="eltOnly" order="seq">
  <element type="get" minOccurs="0" maxOccurs="*" />
  <element type="insert" minOccurs="0" maxOccurs="*" />
  <element type="update" minOccurs="0" maxOccurs="*" />
  <element type="delete" minOccurs="0" maxOccurs="*" />
  <element type="exec" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="get" content="eltOnly" order="seq">
  <element type="content" minOccurs="1" maxOccurs="1" />
  <element type="target" minOccurs="1" maxOccurs="*" />
  <element type="condition" minOccurs="0" maxOccurs="*" />
  <element type="transform" minOccurs="0" maxOccurs="*" />
  <element type="store" minOccurs="0" maxOccurs="*" />
  <element type="nounify" minOccurs="0" maxOccurs="1" />
  <element type="group" minOccurs="0" maxOccurs="*" />
  <element type="order" minOccurs="0" maxOccurs="*" />
  <element type="option" minOccurs="0" maxOccurs="*" />
  <element type="rowcount" minOccurs="0" maxOccurs="1" />
</ElementType>
<ElementType name="content" content="mixed" order="seq">
  <element type="field" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="target" content="eltOnly" order="seq">
  <element type="provider" minOccurs="0" maxOccurs="1" />
  <element type="dsn" minOccurs="0" maxOccurs="1" />
  <element type="catalog" minOccurs="0" maxOccurs="1" />
  <element type="table" minOccurs="0" maxOccurs="1" />
  <element type="request-uri" minOccurs="0" maxOccurs="1" />
  <element type="procedure" minOccurs="0" maxOccurs="1" />
</ElementType>
<ElementType name="condition" content="textOnly" />
<ElementType name="transform" content="textOnly" />
<ElementType name="store" content="textOnly" />
<ElementType name="group" content="textOnly" />
<ElementType name="order" content="textOnly" />
```



```
<ElementType name="option" content="textOnly" />
<ElementType name="rowcount" content="textOnly" />
<ElementType name="provider" content="textOnly" />
<ElementType name="dsn" content="textOnly" />
<ElementType name="catalog" content="textOnly" />
<ElementType name="table" content="textOnly" />
<ElementType name="request-uri" content="textOnly" />
<ElementType name="procedure" content="textOnly" />
<ElementType name="insert" content="eltOnly" order="seq">
  <element type="target" minOccurs="1" maxOccurs="1" />
  <element type="value" minOccurs="1" maxOccurs="*" />
  <element type="option" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="value" content="textOnly" />
<ElementType name="update" content="eltOnly" order="seq">
  <element type="target" minOccurs="1" maxOccurs="1" />
  <element type="set" minOccurs="1" maxOccurs="1" />
  <element type="condition" minOccurs="1" maxOccurs="1" />
  <element type="option" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="set" content="textOnly" />
<ElementType name="delete" content="eltOnly" order="seq">
  <element type="target" minOccurs="1" maxOccurs="1" />
  <element type="condition" minOccurs="1" maxOccurs="*" />
  <element type="option" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="exec" content="eltOnly" order="seq">
  <element type="target" minOccurs="1" maxOccurs="1" />
  <element type="parameter" minOccurs="0" maxOccurs="*" />
</ElementType>
<ElementType name="parameter" content="eltOnly" order="seq">
  <element type="name" minOccurs="1" maxOccurs="1" />
  <element type="value" minOccurs="1" maxOccurs="1" />
</ElementType>
<ElementType name="name" content="textOnly" />
<ElementType name="value" content="textOnly" />
```

```
</Schema>
```

Some examples are shown below.

The next example meaning is to retrieve all table data on the connected database to PlexService.

```
<query>
  <get>
    <content>table</content>
    <target>
      <provider></provider>
      <dsn></dsn>
      <catalog></catalog>
      <table></table>
    </target>
    <condition></condition>
  </get>
</query>
```

The next example meaning is to search database called as “CRMCMaster” on SQL Server installed on DARKSTAR with filter that is “ITEMNAME=6758 AND YEAR=1998”. The searching result must be record and the PlexService’s return value must be OFX format.

```
<query>
  <get>
    <content>record</content>
    <target>
      <provider>SQLOLEDB.1</provider>
      <dsn>DARKSTAR</dsn>
      <catalog>CRMCMaster</catalog>
      <table/>
    </target>
    <condition>ITEMNAME=6758 AND YEAR=1998</condition>
```

```
<transform>TransformOFX.TransOFX</transform>

</get>

</query>
```

Unify tables feature: PlexService will provide the table merge function called as the Unify Function. This function can merge the tables in the different or same databases. So, PlexService users will don't take care about which database the user required tables exist. This function's method model is same as the relational database's one. For example, full outer join, left outer join, right outer join, inner join, and so on. The Unification Modes (the join manner) will be shown in the later table. The Legacy Databases were constructed with no care of each other databases. So PlexService will support the column mapping table called as "Unify Table". The PlexService users can generate the unify table for user's database. So, PlexService will handle the table column information through the defined Unify Table. The following figure shows the example of Unify Table function. The searching process finds Table A1 on SQL Server and Table AA1 on Oracle. The Unify Table indicates that doing FullOuterJoin and handling A column as AA column. So, the result table's column shema is "AA B C... BB CC...". The "AA" column's contents are merged Table A1's A column and Table AA1's AA column.

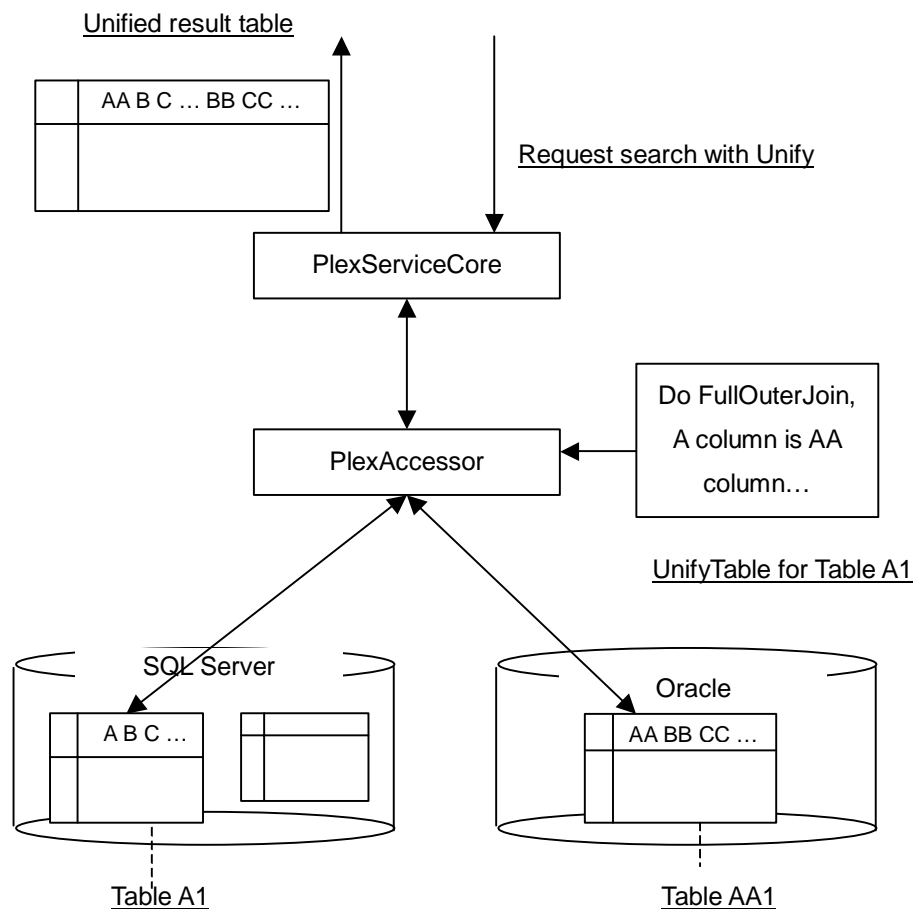


Figure 13 Example of Table Unification

The Table Unification mode will be defined in the Unify Table file. The Unify Table file will be generated with administrator tool called as "uRuleGen". The Table Unification modes are as follow:

Unification Mode	descriptions
Full Outer Join	All Records are moved to the result Table. And same Key Value Fields are merged to 1 record.
Left Outer Join	Left Table is handled as BASE Table. Same Key Value Fields are merged to 1 record.
Right Outer Join	Right Table is handled as BASE Table. Same Key Value Fields are merged to 1 record.
Equiv Join	Return the Equivalent Records only between left and right tables.
Cross Join	Cross Join to left and right tables.
Union Join	Union Join to left and right tables

Left Inner Join	Left Table is handled as BASE Table. Only records that have same key value will be merged to 1 record.
Right Inner Join	Right Table is handled as BASE table. Only records that have same key value will be merged to 1 record.
Max Inner Join	Larger Table is handled as BASE table. Only records that have same key value will be merged to 1 record.
Min Inner Join	Smaller Table is handled as BASE table. Only records that have same key value will be merged to 1 record.

Table 1 Unification Modes

These manners are based on the relational database manners.

The UnifyTable file data format is XML, also. The syntax of it is as follow:

UnifyTable.xdr:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- unify table format XDR file -->

<Schema name="unifytable.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">

<ElementType name="unifyrule" content="eltOnly" order="seq">
  <element type="Provider" minOccurs="1" maxOccurs="1"/>
  <element type="Dsn" minOccurs="1" maxOccurs="1"/>
  <element type="Catalog" minOccurs="0" maxOccurs="1"/>
  <element type="Schema" minOccurs="0" maxOccurs="1"/>
  <element type="unify" minOccurs="1" maxOccurs="1"/>
  <element type="table" minOccurs="1" maxOccurs="1"/>
  <element type="destination" minOccurs="0" maxOccurs="*" />
</ElementType>

<ElementType name="table" content="eltOnly" order="seq">
  <attribute type="name" required="yes"/>
  <element type="pattern" minOccurs="0" maxOccurs="*" />
</ElementType>

<ElementType name="pattern" content="eltOnly" order="seq">
  <element type="lid" minOccurs="1" maxOccurs="1"/>
  <element type="rid" minOccurs="1" maxOccurs="1"/>
```

```
</ElementType>
<ElementType name="lid" content="textOnly" order="seq">
  <attribute type="type" required="yes"/>
</ElementType>
<ElementType name="rid" content="textOnly" order="seq">
  <attribute type="type" required="yes"/>
</ElementType>
</ElementType name="Provider" content=textOnly" />
</ElementType name="Dsn" content=textOnly" />
</ElementType name="Catalog" content=textOnly" />
</ElementType name="Schema" content=textOnly" />
</ElementType name="unify" content=textOnly" />
<ElementType name="destination" content="eltOnly" order="seq">
  <element type="Provider" minOccurs="1" maxOccurs="1"/>
  <element type="Dsn" minOccurs="1" maxOccurs="1"/>
  <element type="Catalog" minOccurs="0" maxOccurs="1"/>
  <element type="Schema" minOccurs="0" maxOccurs="1"/>
  <element type="unify" minOccurs="1" maxOccurs="1"/>
  <element type="table" minOccurs="1" maxOccurs="1"/>
</ElementType>

</Schema>
```

The tag “unifyrule” is the root tag of the Unify Table file. It must have “Provider” tag, “Dsn” tag, “Catalog” tag, “unify” tag and “table” tag. The value of the tag “Provider” must be OLE DB Provider name string (that is ProgID of OLE DB Provider.) The value of the tag “Dsn” must be the name of Data Source Name. The value of the tag “Catalog” must be the name of Database. The value of the tag “unify” must be the unification mode. The value of the tag “table” must have the conversion rule patterns. The conversion rule pattern consists of the “lid” tag (meaning Left ID) and the “rid” tag (meaing Right ID). The value of the tag “lid” must be the name of converted column name. The value of the tag “rid” must be the name of source column name that is actual database column name.

The example of the Unify Table is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<unifyrule>
  <Provider>SQLOLEDB.1</Provider>
  <Dsn>PLEXSERVER</Dsn>
  <Catalog>CRMCMaster</Catalog>
  <unify>right outer join</unify>
  <table name="astockASCII">
    <pattern>
      <lid type="">INFODATE</lid>
      <rid type="DBTYPE_DBTIMESTAMP">IDATE</rid>
    </pattern>
    <pattern>
      <lid type="">FIRSTPRICE</lid>
      <rid type="DBTYPE_I2">OPENPRICE</rid>
    </pattern>
  </table>
</unifyrule>
```

When PlexService loads this sample UnifyTable, the table "astockASCII" will be handled as the following figure. PlexService will generate the virtual table that is converted from Original Table with UnifyTable Mapping information. And then PlexService will execute the unification to the Unify Target Table (that is, the generated virtual table).

IDATE	OPENPRICE	CODE		INFODATE	FIRSTPRICE	CODE
1999/01/01	4750	9473	⇒	1999/01/01	4750	9473
1999/02/01	4920	9473		1999/02/01	4920	9473
1999/03/01	4620	9473		1999/03/01	4620	9473

Original Table Unify Target Table

Figure 14 The UnifyTable effect

PlexService will check the tables that are the target of unification, and if the same column name in each table exists, PlexService will recognize that column as the unify key column and merge them. The example will be shown as follow.

The Unify Table is prepared for Table B only. This file indicates that CUSTOMER column should handle as UNAME column. The Unify manner is FULL OUTER JOIN. So, PlexService will generate the Unified Result Table and convert to XML and return it.

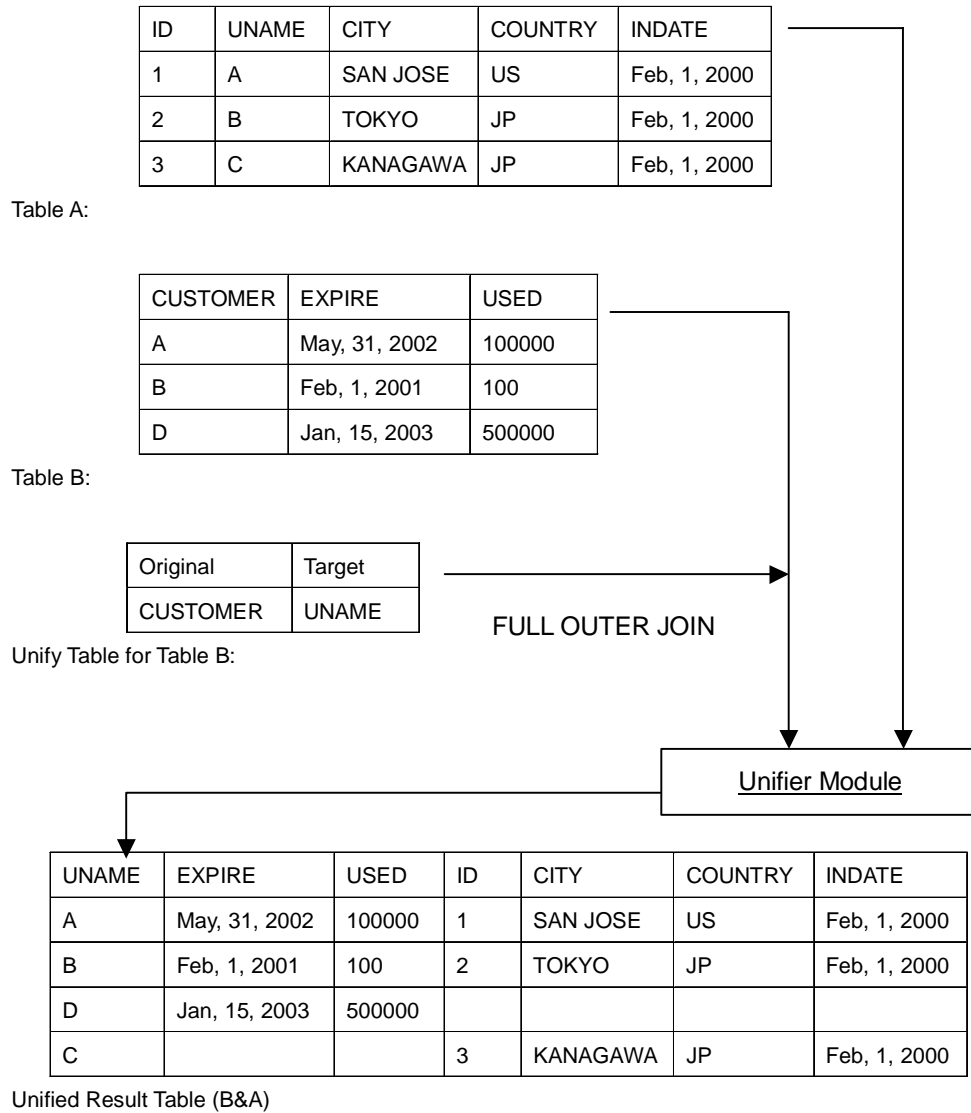


Figure 15 Unification Example

Result transform feature: PlexService will return searching result as XML original format. In some case, client application wants to use it as other dialect of XML format. PlexService will support PlexTransformer to do it.

These PlexService's Legacy Database access function will be provided as the "Query" function. The detail function's control will be specified with its argument value.

The detail of PlexTransformer is described in the section "PlexTransformer".

4.6.2 PlexKlip Database Access

In this section, the PlexKlip Database that is PlexService own Database will be described. PlexKlip consists of XML data and DLL. So, PlexService will support PlexKlip storing with PlexKlip Content Database and PlexKlip Resource Database. PlexService will provide the individual interfaces for each database. So, applications can access to these database with each interfaces.

In the following figure, the requests to PlexKlip Database are shown. PlexService will accept "KLIP Content Request" and "KLIP Resource Request" and send to each database. PlexService will receive responses from each database.

PlexKlip Content Database will manage PlexKlip Content instance. But PlexKlip Resource Database will manage just only the mapping information of KLIP Resource's URI and URL. The PlexKlip Resource Instance will be stored to PlexService's local file system. If application want to access and store this instance, applications must use HTTP put/get methods. Firstly, application uses PlexService's programming interface to PlexKlip Resource Database in order to declare creation or modification of the instance. After that, applications can use HTTP put method. But when application want to delete the instance, applications use just only PlexService's Programming Interface.

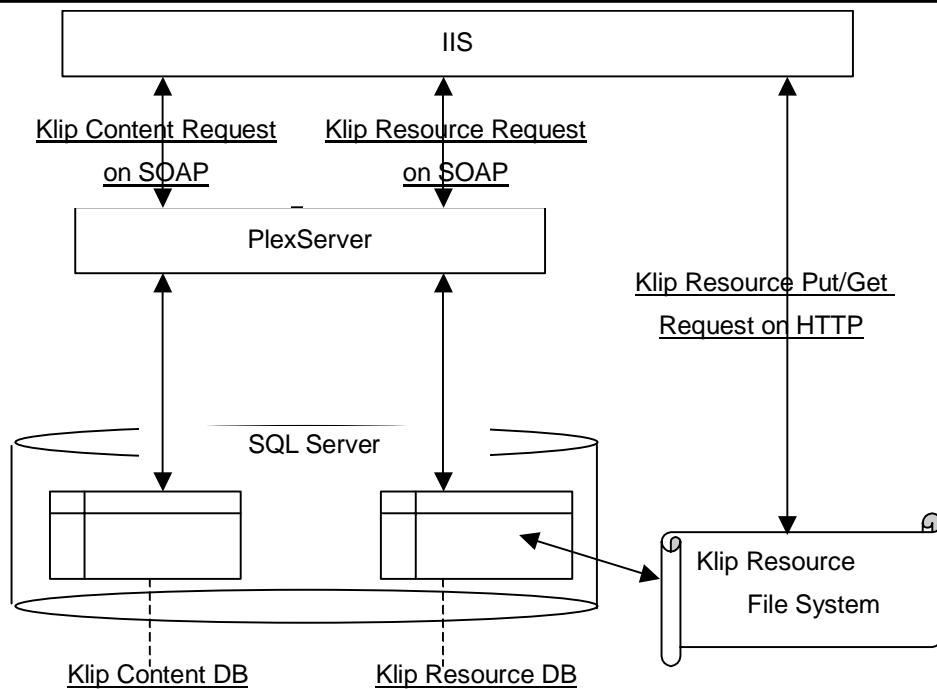


Figure 16 PlexKlip Databases

4.6.3 Collaboration Database Access

In this section, the PlexService mechanism to accessing to the Collaboration database is described. The Collaboration database will provide the share field to PlexWare.

Share Field Features are as follow:

- ✓ Share field is persistent.

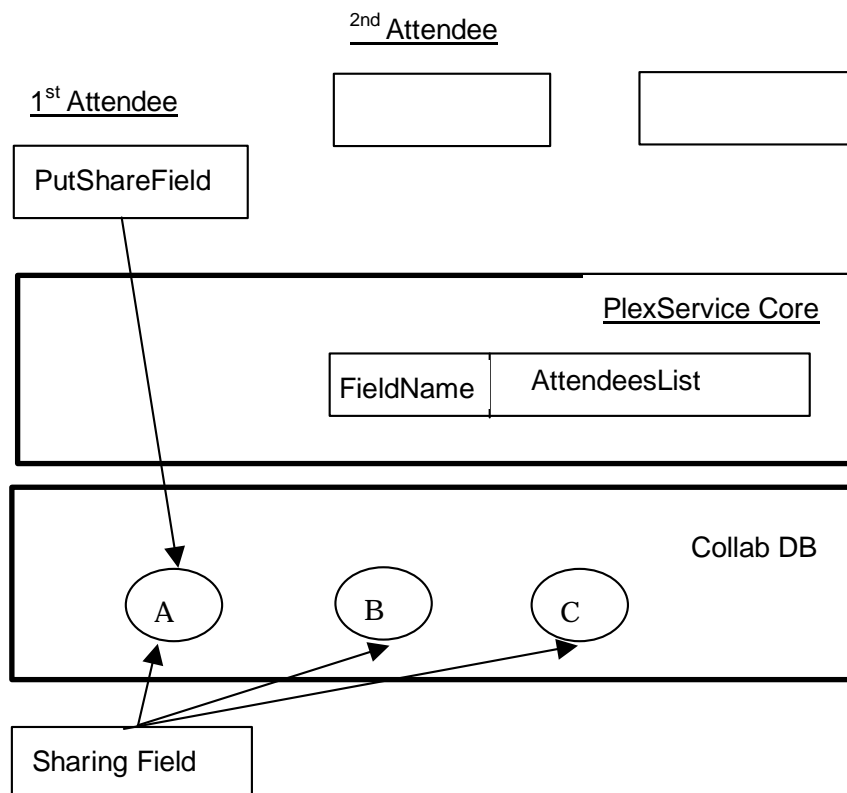
Before user starts collaboration, share field must exist in collaboration DB. And after user completes to collaboration, share field will store to the collaboration DB. So, user can re-start collaboration with the previous used share field.

- ✓ Share field is different from Desktop.

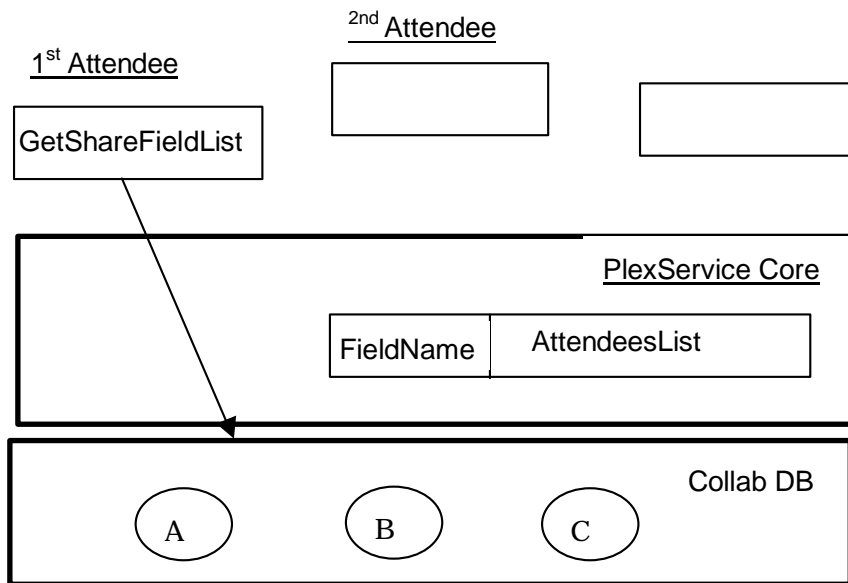
Desktop Klip is personalized object. This klip is used by desktop user only. On the other hand, share field is common object. All accessible users can use it.

PlexWare Collaboration application will load PlexService's share field firstly. Then Collaboration process will be started. Other new attendees will notify to start collaboration to PlexService. Then PlexService will reply share field owner information to them. New attendees application will communicate to owner application to get share field.

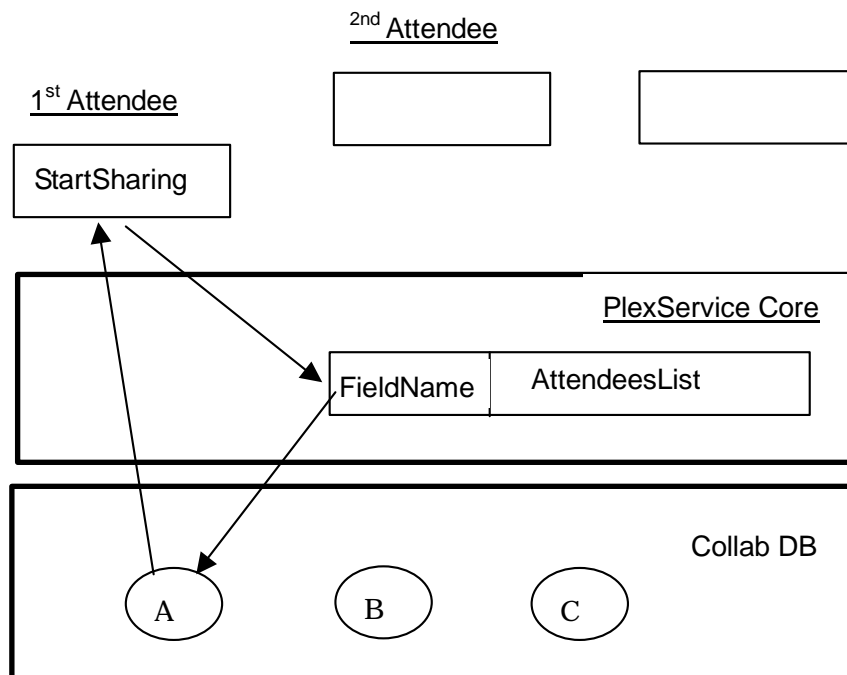
The following figure will show this architecture step by step.



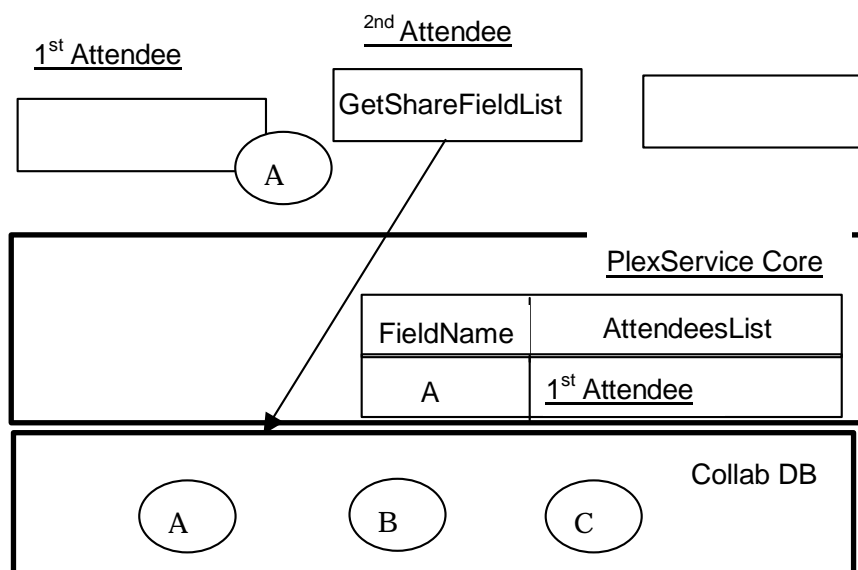
If user wants to create new share field, user must use some maintenance functions.

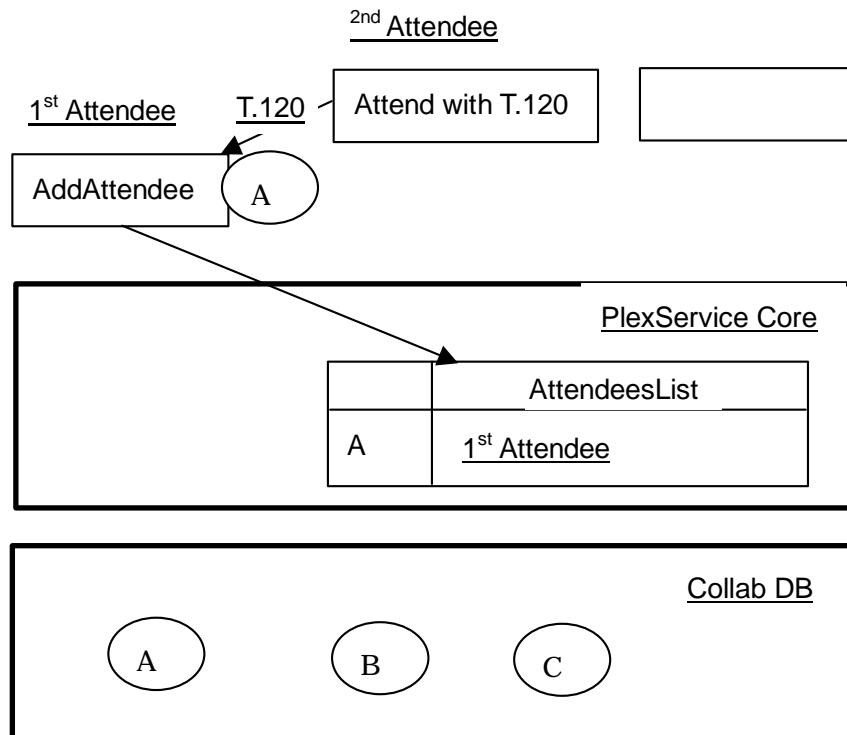


User will execute “GetShareFieldList” function in order to know accessible share fields. If user is owner of the required share field, user can execute “StartSharing” function in order to start collaboration. If user is not owner, user must execute T.120 function in order to attend the required share field. The information to attend share field is provided with the result of “GetShareFieldList” function.



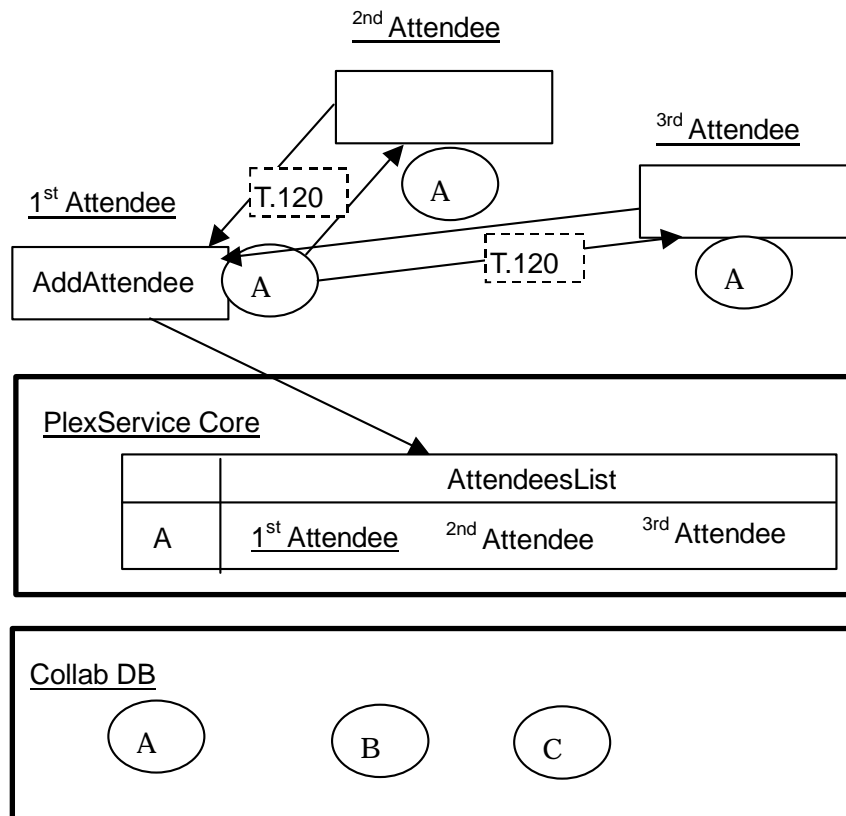
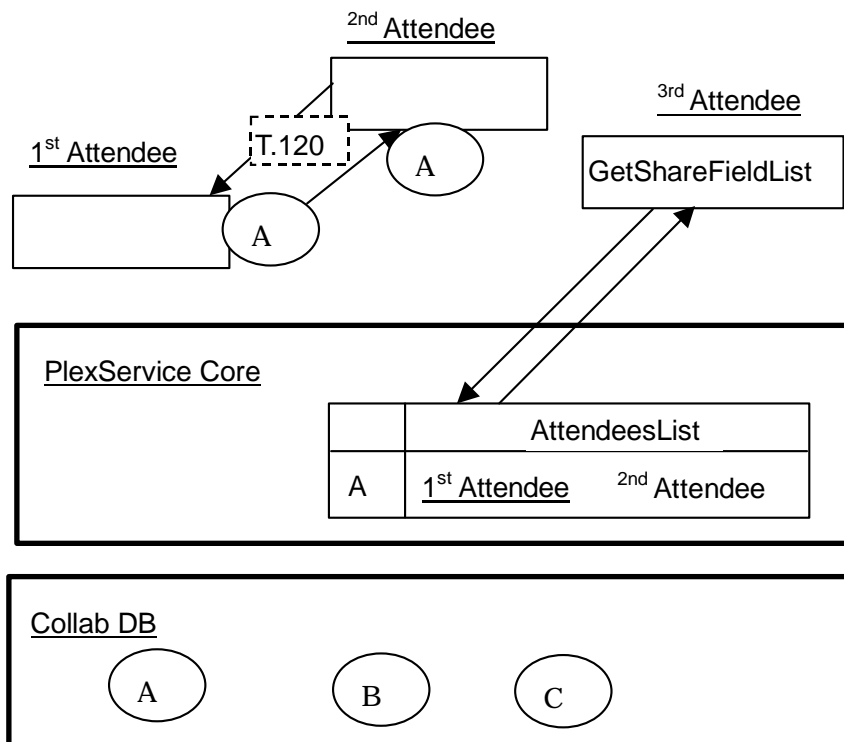
1st attendee will call “StartSharing” function to start the collaboration process. PlexService will check the Active share field List. In this case, as the active share field list doesn't have A field, PlexService will return the content of A field and add A field name and attendee information to the Active share field List.



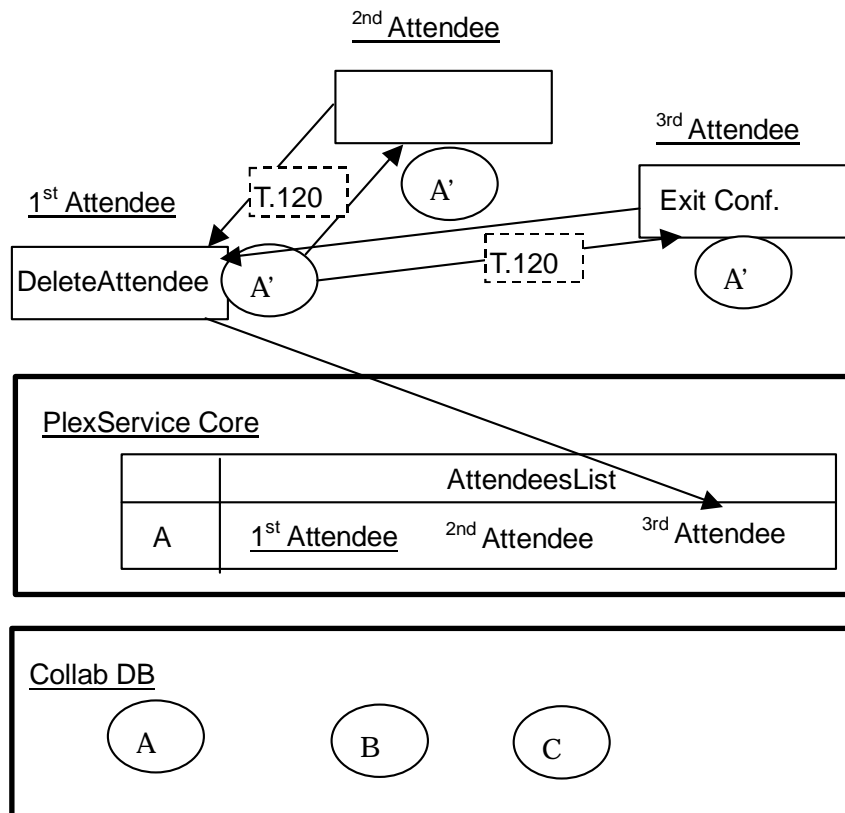


2nd attendee will call “GetShareFieldList” function to get the accessible share field and some owner information (for example, IP address that is used for connecting client applications.) 2nd attendee will try to connect owner application by T.120. Owner application will execute “AddAttendee” function in order to add 2nd attendee to PlexService’s Attendee List. The collaboration between 1st attendee and 2nd attendee is executed with T.120.

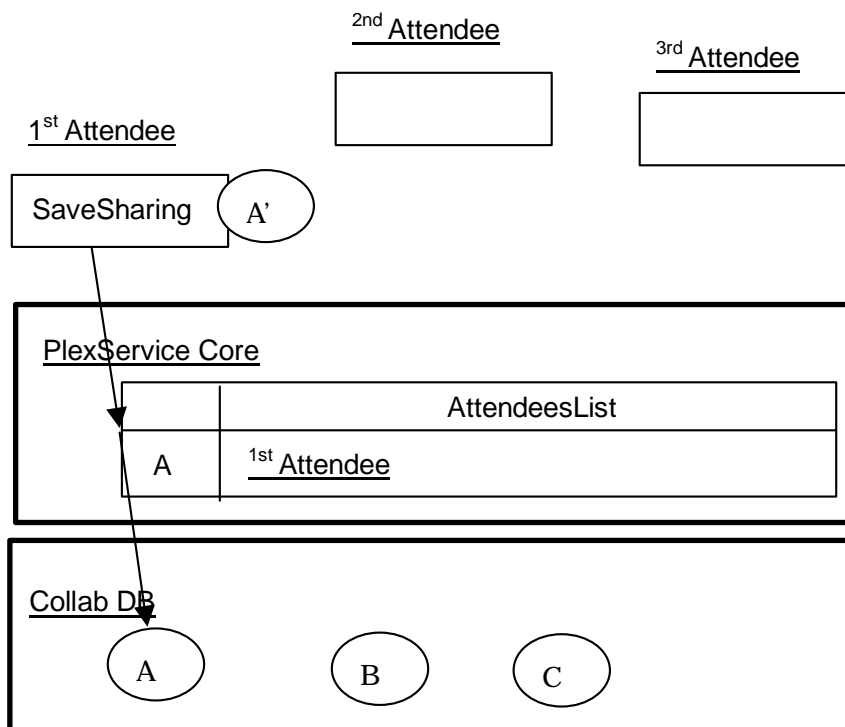
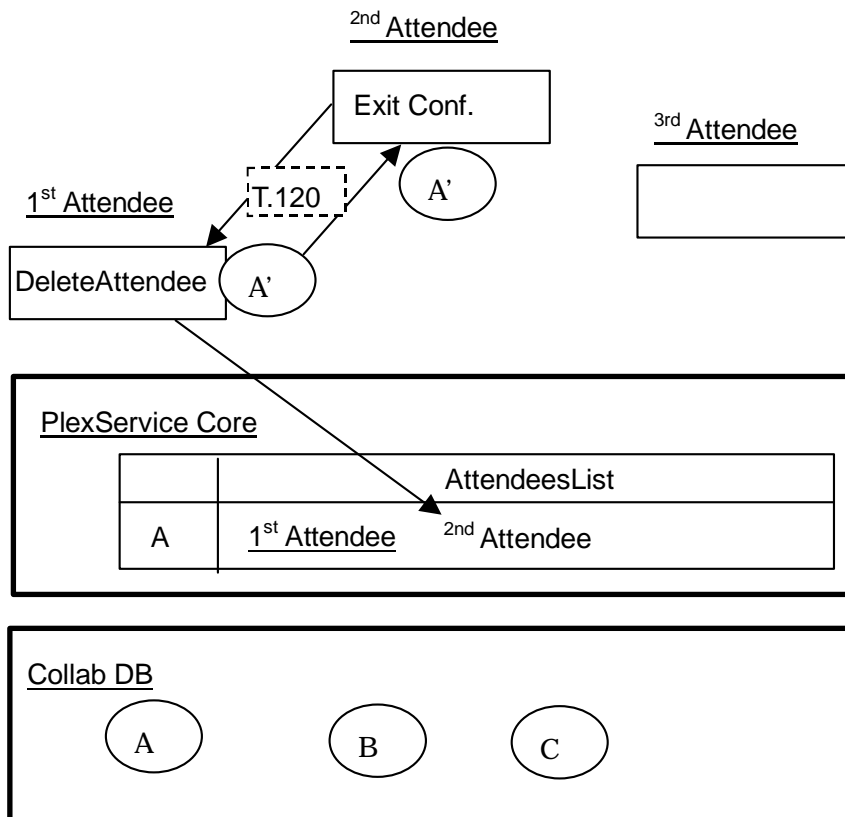
3rd attendee operations are same as the previous description. PlexService’s behavior is same, also.



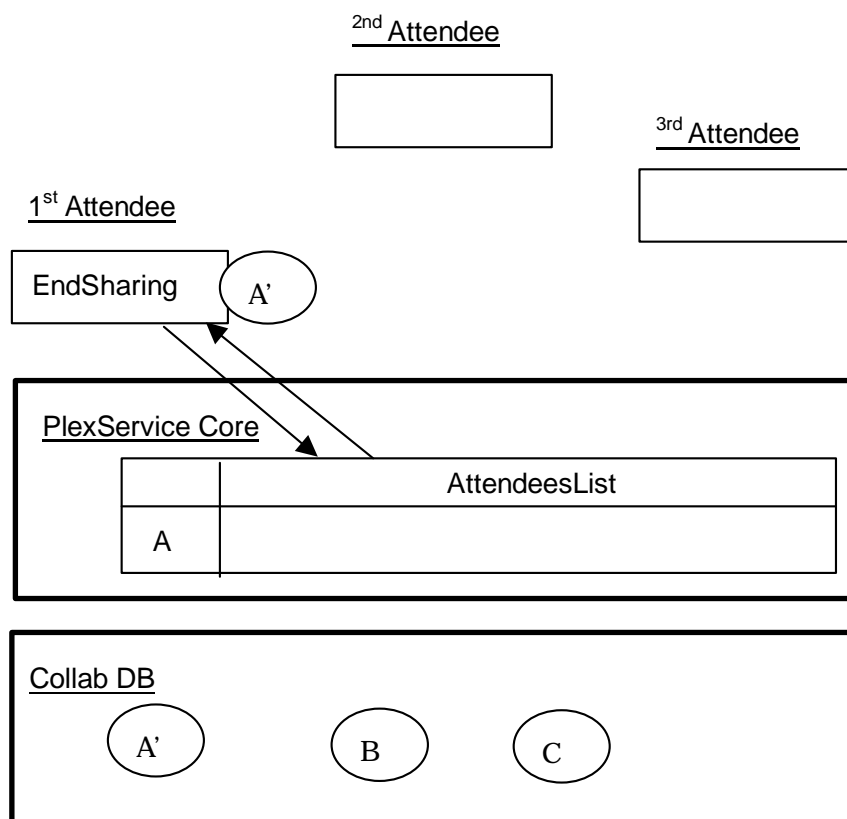
The content of share field will be modified by collaboration process. A', in the following figure, means the modified content. (But its name is A.)

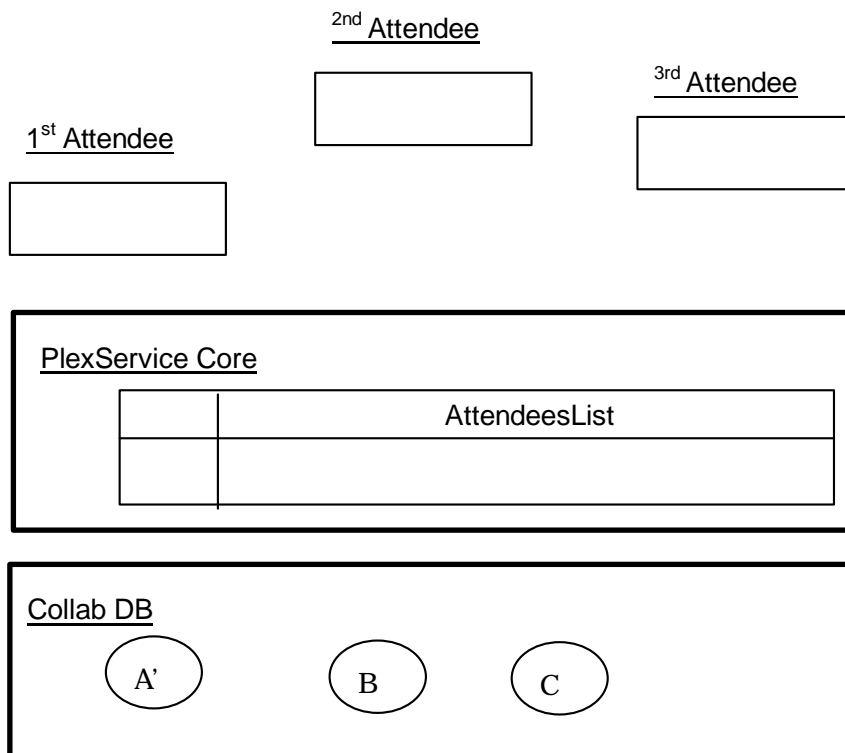


When the collaboration application (or user) wants to quit from the current sharing field, the application must negotiate with owner. Owner application will execute "DeleteAttendee" function to notice PlexService that some attendee quit from this sharing field. PlexService will delete attendee information from attendee list of the active share field.



If last attendee (that is, owner) wants to quit from current sharing field, user must execute “EndSharing” function to complete collaboration process. If user wants to keep the current share field content, user must execute “SaveSharing” function. If this function is called, PlexService will store the current share field content to Collaboration Database. When user will re-enter this share field, its content will be same as the previous session.





The Collaboration application must execute the above manner.

4.6.4 Log Database Access

Log Database is provided for the logs of PlexService itself. Administrator Tool will control the log level. The log level will specify what kinds of log will be stored. PlexService will support the following log-level.

Log Level	Description
None	No logs will be generated. If Log DB is not installed, users must specify this level.
Normal	PlexService will generate the request accept log. The logs have the request name and accept time and so on. (for detail, see Log Database Schema) In this level, the information related with result will not stored.
Detail	PlexService will generate the request accept log and its result log.
Debug	PlexService will generate the request accept log and its result log and more

4.7 Security

In this section, the security of PlexService is described. This section has the following topics.

- ✓ Authentication before accessing to PlexService
- ✓ Access User Control inside PlexService
- ✓ Data Security
- ✓ Access Control List

4.7.1 Authentication before accessing to PlexService

As PlexService uses SOAP, the user requests are accepted as anonymous user requests. (See 3.7.2 for detail) So, the system will require the authentication check mechanism before users will access to PlexService. And PlexService must check the authentication check result information indicates the valid user or not.

PlexService itself does not have account management facilities (registration, deletion, password change, etc.). This design brings many merits, such as centralization of account information on other existing systems. However this design means that user authentication schemes based on shared secrets (e.g. password) are inapplicable and it is necessary to send a password to the server. Therefore, the user authentication interface is premised on use of SSL, and are provided as a separated SOAP interface from PlexServiceCore. During use of SSL, necessity information for user authentication and message authentication will be exchanged between clients and the server in two steps as follows.

At first, a client sends a user ID and password which be authenticated. The server will return a Ticket to the client, if the ID and password is verified

successfully that is of a legitimate account by certificate authorities such as Active Directory. Since a plug-in module does the verification with a certificate authority, it is possible to adapt for arbitrary certificate authorities by adding corresponding modules. In order to use it in the next step, the ticket includes information of authenticated user and certificate authority specific, which are encrypted with the server's private secret key*.

Note: The server private secret key is unique for every server, and is generated at the time of installation

Next, the client sends some parameters along with the ticket, name of a service to access, a message authentication scheme, and an initial seed of it, to the server. As a result, the client obtains a Session Key (shared secret), an URL of the service to access and a Verifier. The verifier is encrypted like the ticket to constitutes a part of data for message authentication as called an Access Token. The remaining portion of an access token depends on the message authentication scheme. Typically, it is constructed by encrypting a timestamp, a nonce (random number), and a digest of message by the session key. Similar to user authentication, schemes of message authentication are fulfilled by a pair of plug-in modules, which are installed on both sides.

By a scheme of the message authentication that considered carefully, both sides can detect attacks (such as replay, interpolation, and fabrication). Hence, there is no need to use SSL (is significantly slower than non-SSL) for subsequent interactions as long as messages don't have confidential information (such as credit card number).

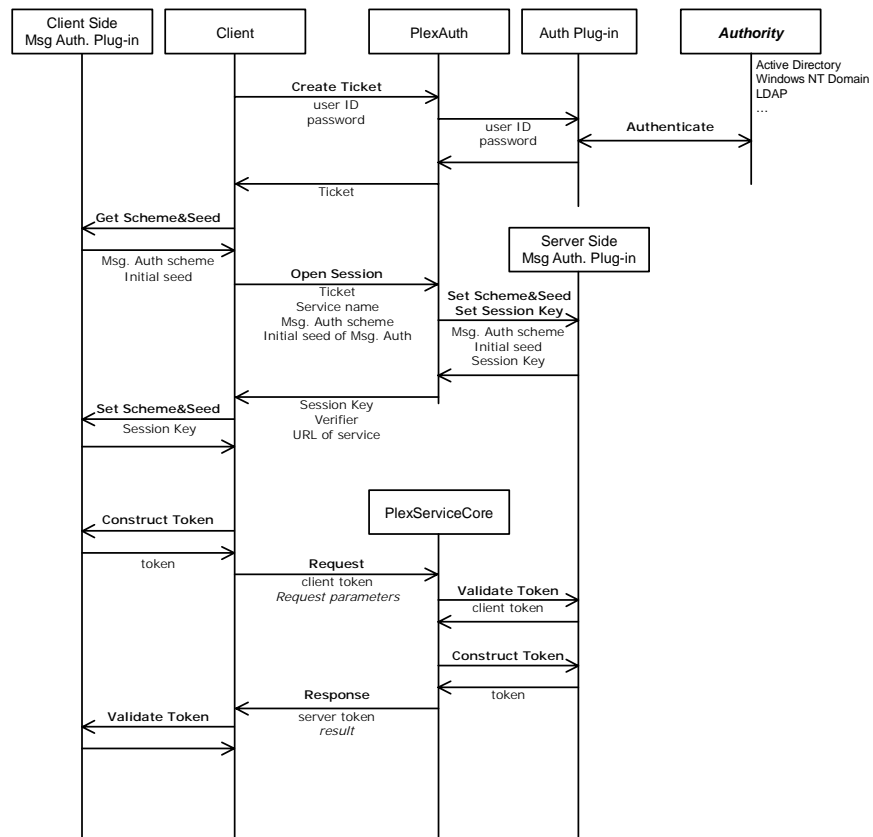


Figure 17 PlexService Authentication Sequence

4.7.2 Access User Control in PlexService

PlexService will accept the user requests through IIS. And the request/response through IIS will be just only SOAP on HTTP. On the other hand, SOAP Listener will accept just only Anonymous user access. So, IIS must be set all SOAP request convert to anonymous user access, even if that is domain user access.

PlexService will accept IIS's anonymous user (IUSER_[MachineName]) access. On the other hands, each database has the defined users having the appropriate access rights. PlexService will use user information in the configuration files that are generated in installing when PlexService will access the databases. PlexAccessor will access to the databases as the users

defined in the configuration files. So, inside PlexService, user name will be changed twice in accessing databases. The process flow of it is as following figure.

Currently, PlexService will support SQL Server 2000/ Oracle 8i R8.1.6/ DB2 UDB V7.1. In installing SQL Server 2000 for PlexService, the following cautions are required.

SQL Server 2000 provides the authentication (Login) method that integrated with Active Directory Authentication and Windows Authentication. But this authentication method can not accept the above PlexService access method, so SQL Server connected to PlexService can not use this method. It is reason that this SQL Server 2000 authentication method prepared for "Single Sign-on" and controlled process owner. If you want to use SQL Server for PlexService, you set that SQL Server must use the own authentication called as "SQL Server and Windows(Mixed Mode)".

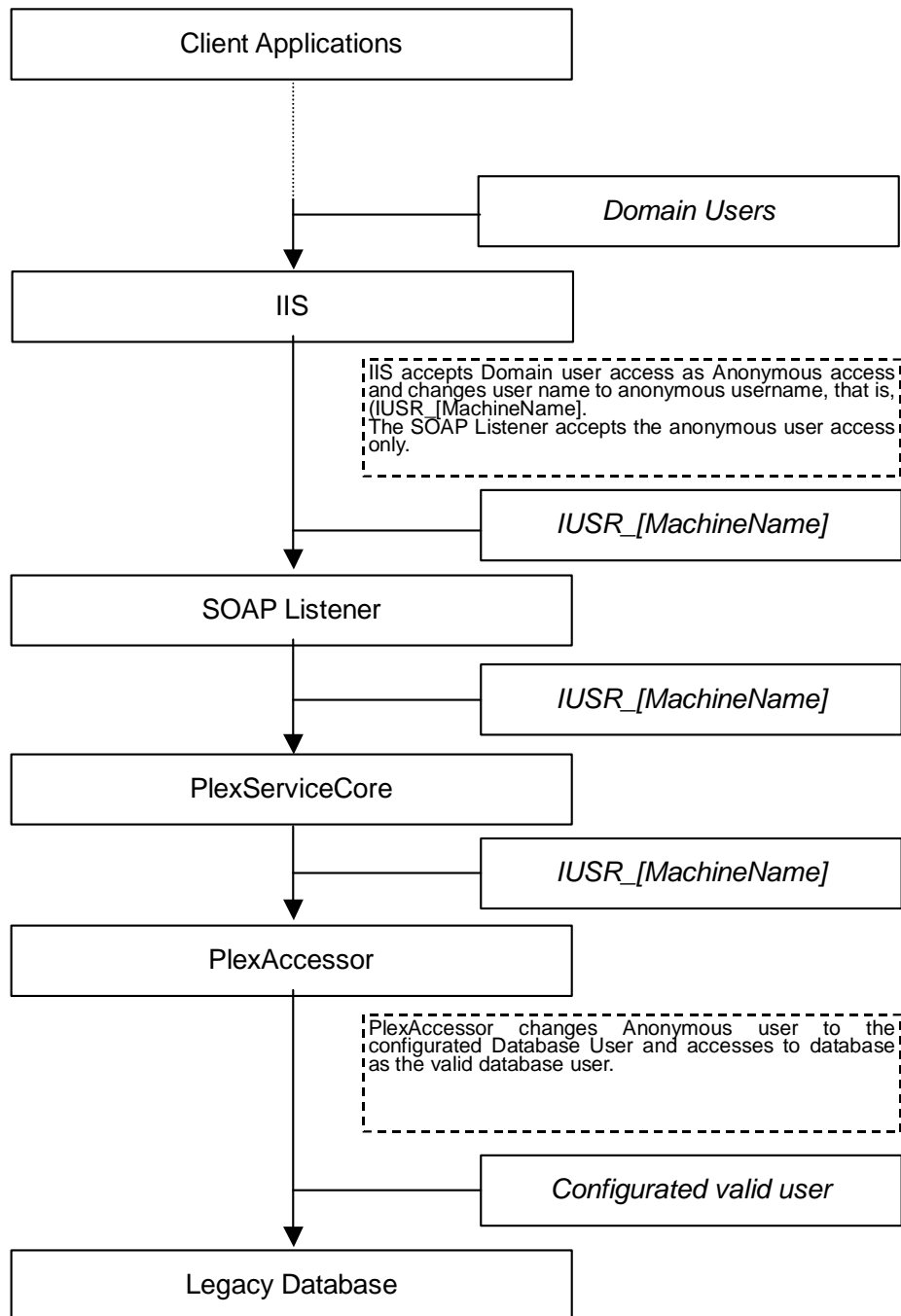


Figure 18 Authentication Control inside of PlexService

4.7.3 Data Security

PlexService will use SSL protocol for Data Security policy. SSL will encrypt the data of SOAP Request and Response on the network. So, the only valid user

can use this data, correctly.

SOAP Listener that is based on Microsoft SOAP SDK on November version will accept SSL protocol. PlexService uses this feature. If users want to use SSL protocol, the Request URI on HTTP must be "https:xxx". If applications use rope.dll, users do not take care of it as the SDL of PlexService will have adequate Request URI. In this case, the person to install PlexService must re-write the SDL content with the adequate Request URI value. ("location" tag's url attribute value must be re-written. The detail information is described in "PlexService SDK Specification".)

On the other hand, if applications will generate SOAP requests themselves, applications must take care of it. These differences are as following figure.

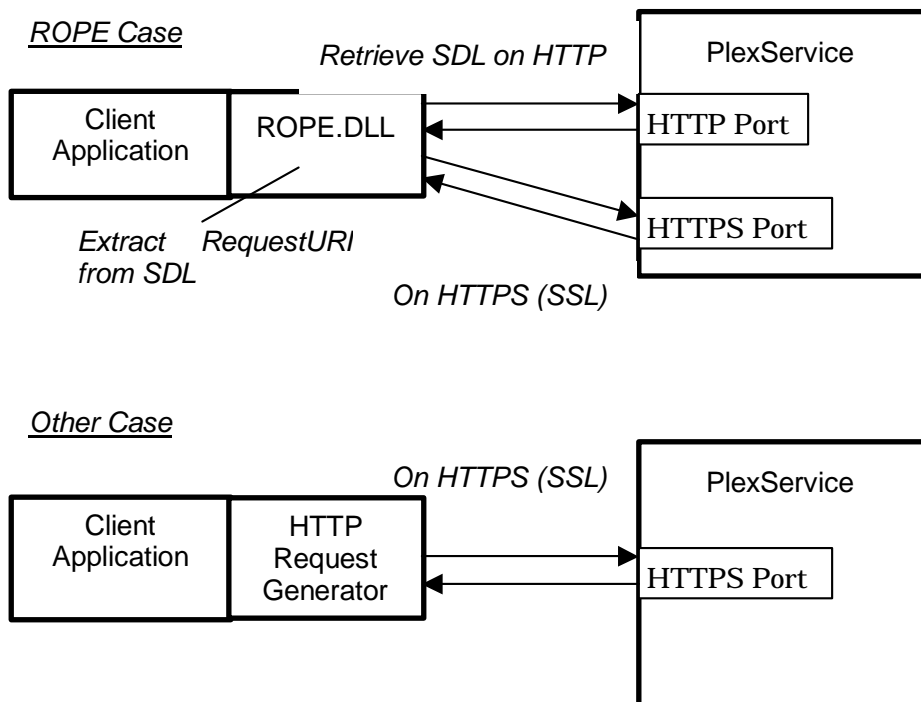


Figure 19 HTTPS (SSL) Hand shaking Example

4.7.4 Access Control List

PlexKlips stored in PlexService Database have Access Control List(ACL) information. When PlexKlip is stored to PlexService Database, PlexService add ACL that has All Access right of the execution user. After that, the execution user can add and delete other user's ACLs. The execution user can

modify self ACL, also.

The ACL format is as follow.

VOACL.xdr:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- VOACL format XDR file -->
<Schema name="VOACL.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">

  <ElementType name="VOACL" content="eltOnly" order="seq">
    <AttributeType name="complete" dt:type="enumeration" dt:values="true
false" default="true"/>
    <attribute type="complete"/>
    <element type="control" minOccurs="0" maxOccurs="*" />
  </ElementType>
  <ElementType name="control" content="eltOnly" order="seq">
    <AttributeType name="complete" dt:type="enumeration" dt:values="true
false" default="true"/>
    <attribute type="complete"/>
    <element type="user" minOccurs="0" maxOccurs="1" />
    <element type="access" minOccurs="0" maxOccurs="1" />
  </ElementType>
  <ElementType name="user" content="textOnly" />
  <ElementType name="access" content="textOnly" />
</Schema>
```

The sample ACL is as follow.

```
<VOACL>
  <control>
    <user>xxx</user>
    <access>accessRightAbbrev</access>
  </control>
</VOACL>
```

PlexServiceV1.6 User's Guide

accessRightAbbrev is string to specify the access right. The following table describes about that.

	Read Content	Write Content	Delete Content	Read Attribute	Write Attribute	Delete Attribute
r	X					
w		X				
rw	X	X				
rwd	X	X	X			
rd	X		X			
wd		X	X			
ar				X		
aw					X	
araw				X	X	
arawad				X	X	X
arad				X		X
awad				X	X	X
rwdarawad	X	X	X	X	X	X

When Visual Obejct is stored to PlexService Database, the execution user's ACL is as follow.

```
<VOACL>
  <control>
    <user>[execution user name]</user>
    <access>rwdarawad</access>
  </control>
</VOACL>
```

5 Appendix A: PlexService SOAP Request/Response PayLoad Examples

In this appendix, PlexService SOAP interface examples are shown. They are SOAP Request/Response Payload and HTTP request/Response. HTTP request/response is from the first line to previous line of "<?xml version='1.0'?>", and SOAP request/response Payload will be continued.

5.1 Query over SOAP/HTTP

5.1.1 Request

```
POST /PlexService/PlexService.asp HTTP/1.1
Host: xxxxxx
Content-Type: text/xml
Content-Length: nnn

<?xml version="1.0"?>
<SOAP:Envelope
  xmlns:SOAP="HTTP://schemas.xmlsoap.org/soap/envelope/2000-03-01"
  encodingStyle="HTTP://schemas.xmlsoap.org/soap/envelope/2000-03-01">
  <SOAP:Body>
    <Query>
      <sqlstmt>
        <query>
          <get>
            <content>record</content>
            <target>
              <provider>SQLOLEDB.1</provider>
              <dsn>DARKSTAR</dsn>
              <catalog>CRMCMaster</catalog>
              <table/>
            </target>
            <condition>ITEMNAME=6758 AND YEAR=1998</condition>
            <transform>TransformOFX.TransOFX</transform>
```

```
        </get>
      </query>
    </sqlstmt>
  </Query>
</SOAP:Body>
</SOAP:Envelope>
```

5.1.2 Response

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnn

<?xml version="1.0"?>
<SOAP:Envelope
  xmlns:SOAP="HTTP://schemas.xmlsoap.org/soap/envelope/2000-03-01"
  encodingStyle="HTTP://schemas.xmlsoap.org/soap/envelope/2000-03-01">
  <SOAP:Body>
    <QueryResponse>
      <return><root><response>
        <PRICECHANGES><ITEM>
          <PROFILE>
            <ITEMNAME type="DBTYPE_I2">6758</ITEMNAME>
            <!--Sony -->
            <UNIT type="DBTYPE_BSTR">Yen/Stock</UNIT>
            <LASTUPDATE
type="DBTYPE_DBTIMESTAMP">2000/04/01</LASTUPDATE>
          </PROFILE>
          <CHANGES>
            <DATE type="DBTYPE_DBTIMESTAMP">1998/01/09</DATE>
            <OPENPRICE type="DBTYPE_I2">12000</OPENPRICE>
            <HIGHPRICE type="DBTYPE_I2">12600</HIGHPRICE>
            <LOWPRICE type="DBTYPE_I2">11800</LOWPRICE>
            <LASTPRICE type="DBTYPE_I2">12100</LASTPRICE>
```

```
        <NETCHANGE type="DBTYPE_I2">0</NETCHANGE>
        <DEALINGS type="DBTYPE_R4">5553.1001</DEALINGS>
    </CHANGES>

    .....

</ITEM></PRICECHANGES>
</response></root></return>
</SOAP:Body>
</SOAP:Envelope>
```

6 Appendix B: PlexService Datatypes

PlexService will generate XML string with column data type information as XML attributes. PlexService uses OLE DB's data type definition as column data type, as PlexService uses OLE DB interface for database accessing.

In this appendix, the relationships between SQL Server's data type and OLE DB's data type are shown. PlexService uses OLE DB's data type in generated XML.

6.1 SQL Server and OLE DB Datatypes

Datatype Category	Datatype Name	OLE DB	Sub category	Range	Size
Character type	char(n)	DBTYPE_STR=129	Fixed size string	1-8000 characters	n
	nchar(n)	DBTYPE_WSTR=130	Fixed size string	1-4000 characters	2n
	varchar(n)	DBTYPE_STR	Fixed size string	1-8000 characters	Character count + 1
	nvarchar(n)	DBTYPE_WSTR	Fixed size string	1-4000 characters	2xcharacter count + 1
	text	DBTYPE_STR	Fixed size string	1-2GB	n+16
	ntext	DBTYPE_WSTR	Fixed size string	1-2GB	2n+16
Binary type	binary(n)	DBTYPE_BYTES=128	Fixed size binary	1-8000Byte	n
	varbinary(n)	DBTYPE_BYTES	Fixed size binary	1-8000Byte	Data size
	image	DBTYPE_BYTES	Fixed size binary	1-2GB	Data size +16
Number type	int	DBTYPE_I4=3	Integer	-2100000000 ~ 2100000000	4
	smallint	DBTYPE_I2=2	integer	-32768 ~ 32767	2
	tinyint	DBTYPE_UI1=17	Integer	0~255	1
	float	DBTYPE_R8=5(Double)	Round numbers	+/-1.79e(308)	4 or 8
	real	DBTYPE_R4=4(Float)	Round numbers	+/-3.40e(38)	4
	decimal(p,s)	DBTYPE_NUMERIC=131	True number	+/-10(38)	2~17
	numeric(p,s)	DBTYPE_NUMERIC	True number	+/-10(38)	2~17
	money	DBTYPE_CY=6	Amount of money	約+/-922京	8

	smallmoney	DBTYPE_CY	Amount of money	+/-214,748.3648	4
Date type	datetime	DBTYPE_DBTIMESTAMP=135	datetime		8
	smalldatetime	DBTYPE_DBTIMESTAMP	datetime		4
Boolean type	bit	DBTYPE_BOOL=11	Yes/No	on/off	1
SPECIAL type	timestamp	DBTYPE_BYTES	timestamp	Unique on DB	8
	uniqueidentifier	DBTYPE_GUID=72	GUID	Unique on net all	16
	cursor		cursor		

Table 2 SQL Server & OLE DB Datatypes

6.2 Oracle and OLE DB Datatypes

Oracle datatypes	Datatype Length	OLE DB datatype
BFILE	(Internal) 4GBytes	DBTYPE_BYTES
BLOB	(internal) 4GBytes	DBTYPE_BYTES
CHAR	7 Bytes	DBTYPE_STR
CLOB	(internal) 4GBytes	DBTYPE_STR
DATE	7 Bytes	DBTYPE_DBTIMESTAMP
FLOAT	DBTYPE_R8	DBTYPE_R8
LONG	2**31-1 Bytes	DBTYPE_STR
LONG RAW	(internal) 2**31-1 Bytes	DBTYPE_BYTES
NCHAR	(internal MultiByte STR) 2000Bytes	DBTYPE_STR
NCLOB	(Internal) 4GBytes	DBTYPE_STR
NUMBER	Base-100 format	DBTYPE_VARNUMERIC
NUMBER(p,s)	(internal : precision, scale)	DBTYPE_NUMERIC
NVARCHAR2	(internal) 4000Bytes	DBTYPE_STR
RAW	65535 Bytes	DBTYPE_BYTES
ROWID	System Dependent	DBTYPE_STR
VARCHAR	65535 Bytes	DBTYPE_STR

Table 3 Oracle and OLE DB Datatypes

6.3 DB2 UDB and OLE DB Datatypes

DB2 UDB datatype	OLE DB datatype
------------------	-----------------

SMALLINT	DBTYPE_I2
INTEGER	DBTYPE_I4
BIGINT	DBTYPE_I8
REAL	DBTYPE_R4
FLOAT	DBTYPE_R8
DOUBLE	DBTYPE_R8
DECIMAL	DBTYPE_NUMERIC
NUMERIC	DBTYPE_NUMERIC
DATE	DBTYPE_DBDATE
TIME	DBTYPE_DBTIME
TIMESTAMP	DBTYPE_DBTIMESTAMP
CHAR	DBTYPE_STR
VARCHAR	DBTYPE_STR
LONG VAR CHAR	DBTYPE_STR
CLOB	DBTYPE_STR DBCOLUMNFLAGS_ISLONG DBPARAMFLAGS_ISLONG
GRAPHIC	DBTYPE_WSTR
VARGRAPHIC	DBTYPE_WSTR
LONG VARGRAPHIC	DBTYPE_WSTR
DBCLOB	DBTYPE_WSTR DBCOLUMNFLAGS_ISLONG DBPARAMFLAGS_ISLONG
CHAR(n) FOR BIT DATA	DBTYPE_BYTES
VARCHAR(n) FOR BIT DATA	DBTYPE_BYTES
LONG VARCHAR(n) FOR BIT DATA	DBTYPE_BYTES
BLOB	DBTYPE_BYTES DBCOLUMNFLAGS_ISLONG DBPARAMFLAGS_ISLONG
DATALINK	DBTYPE_STR

Table 4 DB2 UDB and OLE DB Datatypes