XML 웹서비스를 이용한 전력시스템 MIB 설계

정남준○ 양일권 송재주 고종민 오도은
한전 전력연구원
njjung@kepri.re.kr, ikyang@kepri.re.kr, jjsong@kepri.re.kr, kojm@kepri.re.kr, hifive@kepri.re.kr

A Design of MIB for Power System Using XML Web Service

Nam-Joon Jung○, Il-Gwon Yang*, Jae-Ju Song*, Jong-Min Ko*, Do-Eun Oho*
Korea Electric Power Research Institute

요 약

산업계에서의 제어시스템(Control System)은 크게 SCADA(Supervisory Control And Data Acquisition) 시스템, 분산 제어시스템(Distributed Control System, DCS), PLC 시스템 (Programmable Logic Controller System)으로 분류할 수 있다. 특히 전력분야의 SCADA 시스템은 목적상 실시간의 대규모 측정정보 취득을 목적으로 운영되며, 실시간 및 이력 데이터베이스, 원격 데이터 취득, 다양한 데이터 처리, Supervisory Control 등 복잡한 기능과 다양한 전력시스템과의 정보교환이 요구된다. 다양한 전력시스템과의 정보 교환을 위한 표준화 요구에 부응하여, 향후 전력시스템은 표준화되고 공개된 정보모델을 활용한 통합시스템 개발이 요구된다. 본 논문에서는 전력 공통정보모델(CIM)과 애플리케이션 통합 정보기술을 이용한 전력시스템의 메시지 통합버스 설계에 대하여 기술하고자 한다.

1. 서 론

산업계에서의 제어시스템은 크게 SCADA 시스템, 분산 제어시스템, PLC 시스템으로 분류할 수 있다. 특히, SCADA는 일반적으로 PLC나 다른 상업용 H/W 모듈을 통해 인터페이스 하기위해 H/W 상에 위치하는 소프트웨어 패키지로서, 제철, 발전, 원자력, 배전, 화학 등으로 매우 다양한 곳에서 사용된다. 1,000 ~ 1,000,000개의 입출력 채널을 갖고 있으며, SCADA 시스템의 플랫폼으로는 UNIX, Windows NT, Linux 등 다양하다. 전력분야의 SCADA 시스템은 목적상 실시간의 대규모 측정정보 취득을 목적으로 운영되며, 실시간 및 이력 데이터베이스, 원격 데이터 취득, 다양한 데이터 처리, Supervisory Control 등 복잡한 기능과 다양한 전력시스템과의 정보교환이 요구된다. 본 논문에서는 전력 공통정보모델(CIM)과 애플리케이션 통합 정보기술을 이용한 전력시스템의 메시지 통합버스 설계에 대하여 기술한다.

2. 전력시스템의 통합 발전 형태

SCADA 등 많은 전력시스템은 전력계통의 운전정보가 직접적으로 취득하고 전달하는 시스템으로서 그 중요성이 있다. SCADA 시스템의 경우 시스템 내부적으로도 취득된 정보는 실시간 DB에 저장되고 다른 프로세스나 애플리케이션이 이 정보를 활용한다. 즉 동일한 정보를 여러 애플리케이션에서 활용하게 된다. 이렇게 취득 정보의 전달 뿐만 아니라 전력시스템의 애플리케이션간의 효율적인 정보 교환이 요구된다.

전력계통의 여러 애플리케이션간 정보교환을 위하여 가장 먼저 시도된 방식은 '전력계통 데이터'라는 공통의 데이터를 이용하여 각자의 기능을 수행하는 형태이다. 따라서 이들은 중앙의 데이터베이스를 공유하여 사용하였고, 애플리케이션 간 자료교환을 위해서는 데이터베이스를 통하여 간접적으로 전달하거나 또는 직접 point-to-point 방식으로 연결되었다. 그러나 기존의 방식으로는 애플리케이션을 추가하고 통합하는데 아래 그림에서 보는바와 같이 'Island of Automation' 형태의 문제점을 가지게 되었다.[1]

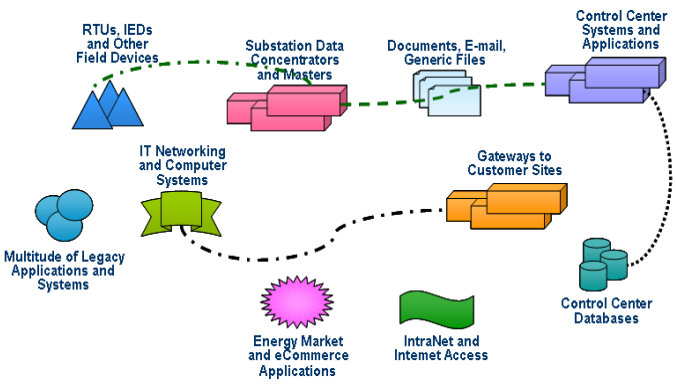

그림 1. 'Islands of Automation'방식의 통합

그 다음으로 아래 그림과 같이 데이터베이스를 이용한

애플리케이션 통합 방식이 도입되었는데 초기에는 문제가 없지만, 점차 유지보수가 힘든 단점이 있다. 만약 데이터베이스의 테이블이 하나라도 변경된다면 데이터베이스와 관련 있는 다른 애플리케이션들도 모두 수정을 해야 한다. 즉 확장하거나 업그레이드하는데 시간과 비용이 과다 발생하게 된다.

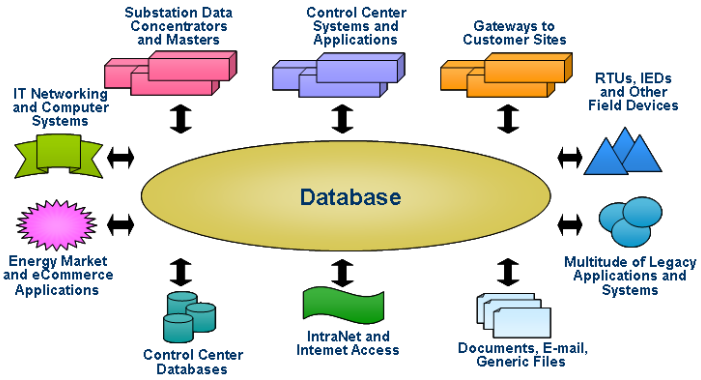

그림 2. 데이터베이스 중심의 데이터교환

따라서 최근에는 아래 그림과 같이 공통정보모델과 공통 인터페이스를 이용한 유연한 아키텍처를 제안하게 되었는데, 이러한 아키텍처는 각 애플리케이션들의 독립성을 최대한 보장하면서 애플리케이션 간에 데이터교환을 효과적으로 수행할 수 있도록 한다.

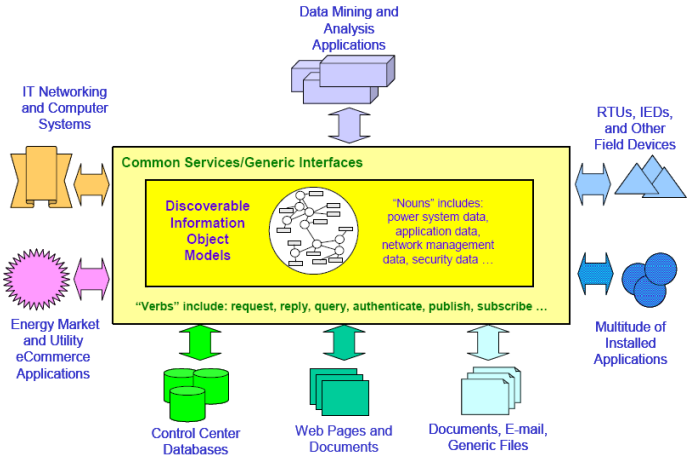

그림 3 공통정보모델과 공통인터페이스를 이용한 데이터 교환

3. 공통정보모델(CIM) 개요

CIM (Common Information Model)은 에너지 관리 및 배전시스템에서 적용되는 공통 언어이다. CIM에 대한 규격은 IEC 61970에 자세히 설명되어 있다. CIM 자체에 대한 설명은 301, 302에 기술되어 있으며, CIM 데이터 교환에 대한 설명은 501과 503에 자세히 설명되어 있다.[2][3]

IEC 61970에서는 공통정보모델(CIM)과 CIS GID를 정의하여 애플리케이션 통합 아키텍처를 구현하였다.CIM은 전력계통의 모든 구성요소를 객체지향 방법을 사용하여 모델링한 것으로서 UML(Unified Modeling Language)을 이용해서 정의한다. UML은 다른 프로그램 언어에 독립적인 모델링 언어이며, 이것을 객체지향 언어인 C++, Java, C#등을 이용하면 구조적인 변경 없이 그대로 구현할 수 있다. 시스템 통합에 활용된 통신 인터페이스는 CORBA(Common Object Request Broker Architecture)의 IDL(Interface Definition Language)을 이용하여 인터페이스를 정의하고 있고, 개발언어에 독립적인 인터페이스를 정의하고 있다. 그리고 향후에는 이것을 C++, Java, C#, Web Services방식으로 구현하기위한 명세서가 작성될 계획이다. 아래 표는 IEC 61970의 GID의 5시리즈 목록이다.

표 1. IEC 61970 GID 계획목록

Part 5XX-Y	Description
502-7	C language Profile for Common Services (-402)
503-7	C language profile for GDA (-403)
503-8	Web services profile for GDA (-403)
504-7	C Language Profile for HSDA (-404)
504-8	Web services profile for HSDA (-404)
505-7	C language profile for GES (-405)
505-8	Web services profile for GES (-405)
507-7	C language profile for TSDA (-407)
507-8	Web services profile for TSDA (-407)

4. Application 통합 기술

4.1 웹 서비스를 이용한 통합

웹서비스(Web Service)는 플랫폼과 프로그램 언어에 독립적인 특성으로 인해 최근 데이터교환 방식으로 널리 사용되는 기술로, 다음 그림은 Java 애플리케이션 간에 Web Service를 이용한 데이터 통신의 예를 보여주고 있다. Web Service는 데이터 교환 프로토콜로SOAP을 사용하고, XML메시지를 효과적으로 전달할 수 있는 방식이다. 이때 SOAP의 구성 양식은 WSDL(Web Service Description Language)를 통해서 정의된다.[4]

Web Service기술은 W3C에서 표준으로 정의되어 있으며, 차세대 분산객체 연동기술로 받아들여지고 있다.

IEC 61970은 SOAP 메시지의 구조를 WSDL을 사용하여 정의함으로써 이를 구현한 애플리케이션들을 통합할

수 있다.

4.2 엔터프라이즈 서비스 버스를 이용한 통합

인터페이스를 구현하기 위한 분산객체 기술은 크게 CORBA[1], DCOM[2], RMI[3], Web Service 방식으로 나눌 수 있는데, DCOM은 마이크로소프트 진영의 분산객체기술이고 유닉스 계열의 OS에서는 실행될 수 없다. RMI는 Java 진영의 기술로 Java VM(Virtual Machine)이 설치된 곳이면 실행될 수 있다. CORBA와 Web Service기술은 마이크로소프트 계열과 Java계열에서 모두 구현할 수 있다. SOA의 핵심 목표인 비즈니스 함수의 가상화는 서비스 정의와 사용을 서비스 구현으로부터 분리시킴으로서 실현된다. 서비스는 IBM WebSphere MQ, IBM CICS, IBM IMS, Java2 Platform, Enterprise Edition(J2EE) Enterprise JavaBeans(EJB), Java classes, IBM DB2 Queries, Java Message Services(JMS), Microsoft .NET 등 다양한 기술들을 사용하여 구현된다. ESB[4]는 참여자들 사이에서 발생하는 서비스 인터랙션의 관리와 가상화를 지원하는 아키텍쳐 패턴이다. 서비스 공급자와 요청자들 사이를 연결하고, 정확히 매치되지 않더라도 인터랙션을 가능케 한다. 이 패턴은 다양한 미들웨어 기술과 프로그래밍 모델을 사용하여 구현될 수 있다.

그림 4는 일반적인 ESB 패턴을 나타낸 것이다. 엔드포인트가 ESB와 인터랙팅하는 사이트를 서비스 인터랙션 포인트(SIP)라고 한다. SIP는 웹 서비스 엔드포인트, WebSphere MQ 큐, RMI 원격 객체용 프록시가 될 수 있다. 서비스 레지스트리는 SIP의 요구 사항과 기능, 다른 SIP와 인터랙팅을 하는 방법(동기식, 비동기식, HTTP, JMS), QoS 요구사항(보안, 신뢰성, 인터랙션), 기타 SIP 정보 등을 설명하는 메타데이터를 파악한다.[5]

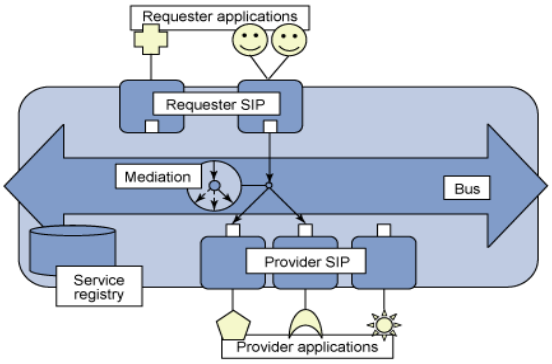

그림 4. 일반적인 ESB 패턴

1) Common Object Request Broker Architecture
2) Distributed Component Object Model
3) Remote Method Invocation
4) ESB ; Enterprise Service Bus

5. 연관된 표준화 기반 기술

IEC 61970 GID는 OMG[5]의 DAIS[6]를 상속받아 구현되었다. 즉 기본 바탕은 OMG의 DAIS를 이용하고 여기에 전력계통을 위한 요소를 추가하였다. 그런데 OMG의 DAIS도 OPC의 OPC 스펙을 참조하여 개발된 것이다. OPC는 마이크로소프트의 분산객체기술인 COM방식을 이용하여 원격데이터를 취득하기 위해 1990년대에 개발된 규격이다.

5.1 OPC

마이크로소프트에서 소프트웨어 컴포넌트 간에 통신을 위해 처음 개발된 것이 DDE(Dynamic Data Exchange)이다. 이것은 후에 OLE(Object Linking and Embedding)이라는 것으로 발전하였다. 1990년대에 PC기반의 SCADA HMI가 등장하면서 적용되었다. OPC는 각종 애플리케이션들이 서로 다른 프로세스 컨트롤 장비(DCS, PLC 등)와 데이터를 주고받을 수 할 수 있게 적용된 표준 인터페이스라고 정의 할 수 있다. 어플리케이션들은 각기 다른 여러 종류의 OPC 호환 서버(DCS, PLC 등)들로부터 데이터를 주고받는데 단지 하나의 OPC 호환 드라이버만 설치하면 된다.

5.2 OMG DAF, DAIS

OPC가 산업체에서 널리 받아들여 활용되었으나, 기본적으로 마이크로소프트 Windows를 바탕으로 하고 있기 때문에 유닉스 시스템을 주로 사용하는 전력계통에서는 활용하기가 불가능 했다. 또한 OPC는 공장자동화 같은 정도의 소형 시스템에 주로 적용되었기 때문에 좀 더 일반화될 필요가 있었다. 이에 OMG에서는 DAF(Data Access Facility Specification)와 DAIS(Data Acquisition from Industrial System)라는 스펙을 정의 하였다.

5.2.1 DAF

DAF는 SCADA 시스템과 같은 산업 시스템에서 데이터 교환을 효과적으로 하기위해 개발되었고 데이터를 읽어 들이거나 내보낼 수 있을 뿐만 아니라 데이터 모델도 전달할 수 있도록 설계되었다. 또한 데이터에 변경사항이 발생하였을 때 통지할 수 있는 기능도 고려되었다. 아래 그림 5는 DAF 모듈의 구성을 나타낸다.[6]

- DAFIdentifier는 각 데이터요소에 구분자를 제공하고 조회하기 위한 서비스이고,

5) OMG : Object Management Group
6) DAIS : Data Acquisition from Industrial System

- DAFDescription은 여러 개의 리소스를 한꺼번에 반환하지 않고 Iterater를 반환하여 조회할 수 있도록 한 서비스이다. 이 기능을 통하여 한꺼번에 대용량의 데이터를 보냄으로써 발생하는 시스템 부하를 줄일 수 있다.
- DAFQuery는 특정 조건에 맞는 데이터 리소스만을 조회하기 위한 서비스이다
- DAFEvents는 데이터에 변경사항이 발생하였을 때 통지해줄 수 있도록 이벤트를 수신 등록하고, 발생시킬 수 있는 인터페이스를 제공한다

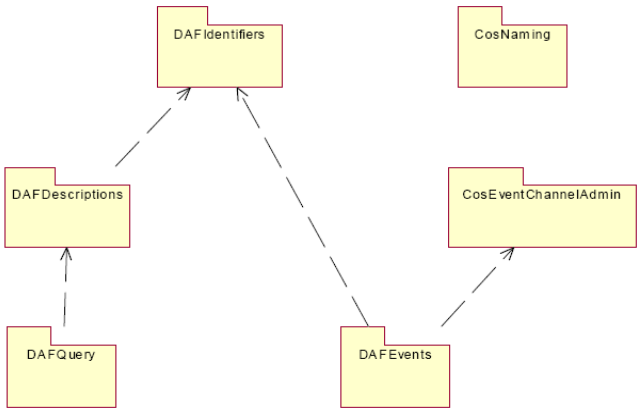

그림 5. DAF 모듈 구성

5.2.2 DAIS

DAIS(Data Acquisition from Industrial System)는 OPC를 대형 산업시스템에도 적용할 수 있도록 확장한 형태이다. 아래 그림 6에서처럼 내부적으로는 DAF 규격의 일부를 사용하고 있다.

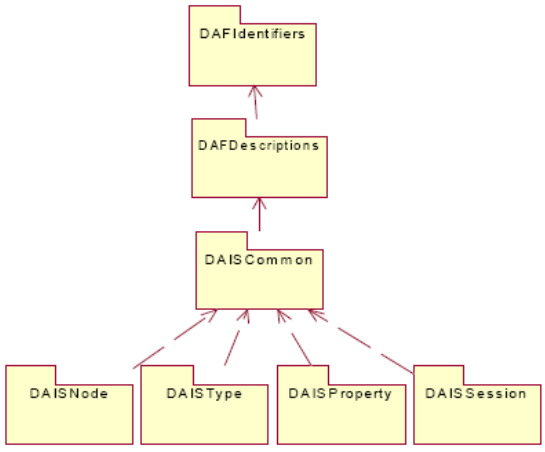

그림 6. DAIS Common 모듈 구성

5.2.3 IEC 61970 GID

GID(Generic Interface Definition)는 OMG의 DAF, DAIS를 기반으로 하고 있으며, 전력계통을 위한 인터페이스가 몇 가지 추가되어 있다. 아래 그림 7에서 볼 수 있듯이, GID의 Common Service와 GDA는 DAF를 이용하여 구현되었고, 고속데이터 전송을 위한 HSDA(High Speed Data Access) 인터페이스와 이벤트 전달을 위한 GES(Generic Eventing Subscription)는 DAIS를 이용하여 구현되었음을 알 수 있다.[7]

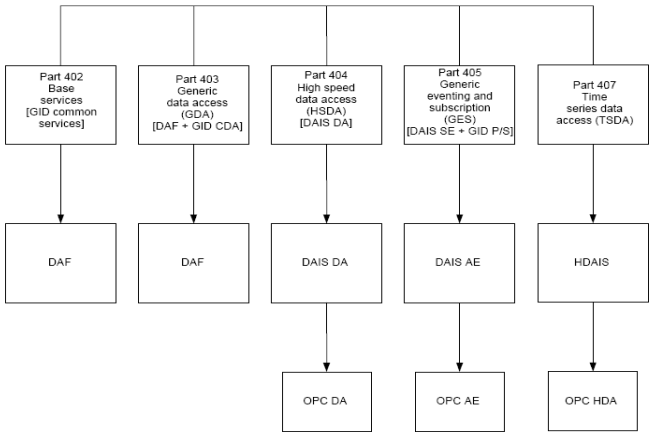

그림 7. IEC 61970의 인터페이스 규격

6. Message Integration Bus 설계

전체 시스템을 통합하는데 있어서 기존의 시스템과 통합은 필수적이다. 하지만 기존의 시스템은 IEC 61970의 표준 인터페이스를 적용하지도 않았을 뿐만 아니라 서로 다른 분산객체기술을 사용하고 있다. 앞에서 언급했듯이 서로 다른 분산객체 기술은 서로 호환되지 않는다. 다음 그림 8은 전력시스템을 위한 메시지 통합버스를 이용한 통합을 나타낸다.

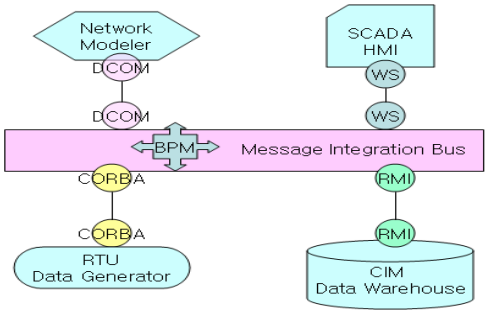

그림 8. 메시지 통합버스를 이용한 통합

메시지 통합버스는 여러 분산객체 기술을 한꺼번에 탑재할 수 있다. 따라서 앞서 언급한 분산객체기술인 DCOM, CORBA, Web Service, RMI를 모두 동시에 탑재 가능하고 이들 기술 간에 연동을 가능하게 한다. 또한 IEC 61970의 GID를 구현하지 않은 애플리케이션에 대해서도 메시지 변환을 통해서 연동이 가능하다.

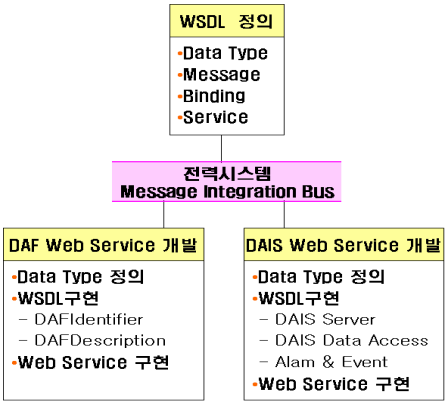

그림 9. Message Integration Bus 구현 절차

6.1 WSDL 정의

WSDL은 SOA 컴포넌트 간에 인터랙션을 위한IDL로서, 데이터 타입과 메시지, 바인딩, 서비스에 대한 정의를 한다. WSDL에서는 wsdl:types 항목에서 데이터 타입을 기입한다. 메시지는 일반 프로그램에서 함수를 호출할 때 사용되는 입출력 변수라고 할 수 있다. 입력변수의 타입을 미리 정의하고 공개함수를 정의하는portType에서 사용된다. 바인딩은 wsdl:binding 형태로 외부에 노출할 port들을 기술하고, SOAP메시지의 style과 use를 정의한다. 서비스는 최종적으로 서비스가 제공되는 부분을 기술하는 부분이다. 서비스의 최상단은 Port이며, 여기에 바인딩 방법과 서비스의 주소가 작성된다.

6.2 DAF Web Service 설계

6.2.1 Data Type 정의

기본 데이터타입으로 String으로 된 URI와 URI의 배열인 URISequence 그리고, 두 개의 long long 타입으로 이루어진 ResourceID 구조체이다. 제공되는 함수로는 ResourceIDSequence를 반환하는 get_resource_ids, URISequence를 반환하는 get_uris로 되어 있다. DAF에서 가장 핵심적인 데이터 타입이 SimpleValue이다. 이것은 모든 형태의 값을 하나의 타입으로 접근하기 위해 정의되었다. 본 연구 과제를 통해 설계된 데이터 타입의 개수는 다음과 같다.

모듈명	Simple Type	Complex Type
DAFIdentifier	1	4
DAFDescription	4	20

6.2.2 WSDL 구현

OMG의 UMS DAF에서 정의된 인터페이스를 웹서비스로 구현하기 위하여 앞서 정의된 데이터 타입을 이용하여 다음 그림 10과 같이 WSDL을 작성한다. DAF에서는 크게 DAFIdentifier와 DAFDescrition 두 가지의 모듈이 존재한다.

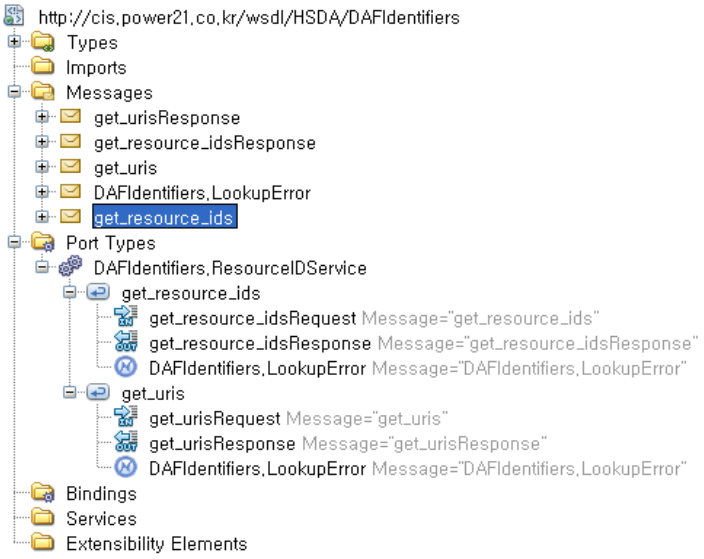

그림 10. DAFIdentifier WSDL구조도

6.2.3 Web Service 구현

PropertyValue와 ResourceDescription 두 개의 타입을 추가로 정의하였고, 이를 이용하여 Resource Description Iterator에서 max_left(), next_n(), destroy() 세 개의 함수를 제공한다. 다음은 Web Service로 구현된 구조이다.

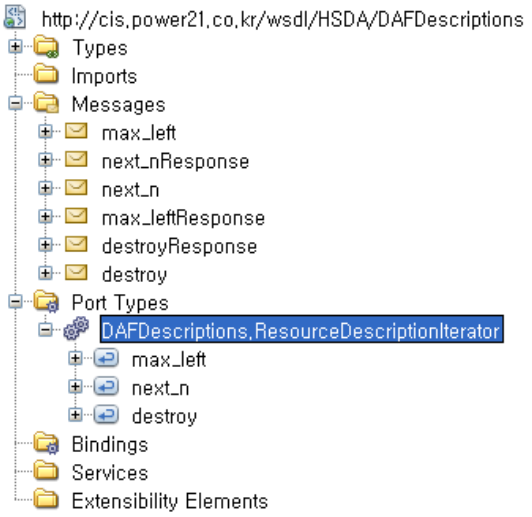

그림 11. DAFDescription WSDL 구조도

6.3 DAIS Web Service 설계

DAIS는 Server, Data Access, Alarm & Events와 같이 3파트로 구분될 수 있다. Server에서는 여러 Session을 관리하고 있는데, 한 클라이언트 당 하나의 세션이 생성된다. 클라이언트와 세션간에는 Data Access 방식과 Alarm & Event 방식으로 데이터를 교환할 수 있다.

6.3.1 Data Type 정의

클라이언트의 요청에 따라 data access session이나 alarm & event session을 생성하여 각 클라이언트 접속을 담당하는 **DAIS Server**, DAF의 ResourceID를 갖 **DAISNode**, 각 객체의 타입을 나타내며, 각 개체를 타입별로 조회하는 경우에 이용할 수 있는**DAISType**, 각 타입의 속성을 나타내는 요소인**DAISProperty**, DAIS 서비스의 한 연결을 나타내는 **DAISSession** 등이 정의되며 DAIS Common 서비스를 작성하기 위해 설계된 데이터 타입의 종류는 아래와 같다.

표 2 DAIS Common관련 클래스

인터페이스 명	Complex Type	Elements	Simple Type
DAISNode	5	24	-
DAISProperty	3	9	-
DAISServer	58	57	5
DAISSession	1	11	-
DAISItem	8	36	-
DAISGroup	5	28	-
DAISType	3	15	-

DAIS Data Access는 연속적인 데이터를 클라이언트에 보낼 때 사용된다. 주로 RTU에서 SCADA 시스템으로 취득 데이터를 전송할 때 사용될 수 있다

6.3.2 WSDL 구현

Data Access모듈은 다음 그림 12와 같이 WSDL로 작성되었으며, 웹서비스 형태로 동작한다. 그림에서 보는 바와 같이 ConnectionPoint와 비동기 입출력, 콜백, 동기 입출력 PortType을 지원한다.

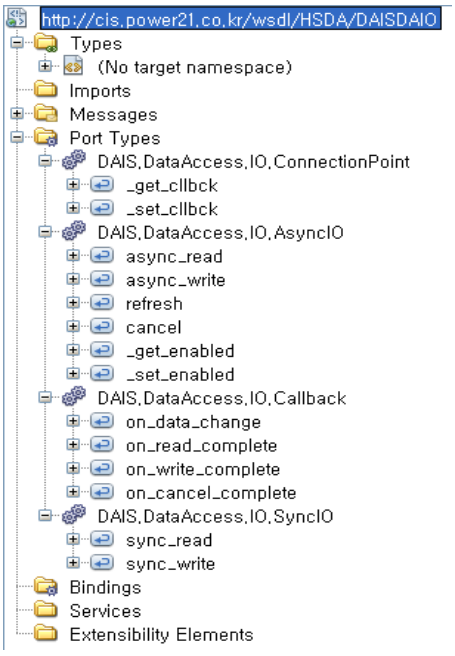

그림 12. DAIS DAIO모듈 WSDL 구조도

7. 결론

본 논문에서는 전력 공통정보모델(CIM)과 애플리케이션 통합 정보기술을 이용한 전력시스템의 메시지 통합버스 설계에 대하여 기술하였다. 메시지 통합버스는 여러 분산객체 기술을 한꺼번에 탑재할 수 있다. 따라서 다양한 분산객체기술인 DCOM, CORBA, Web Service, RMI를 모두 동시에 탑재가능하고 이들 기술 간에 연동을 가능하게 한다. 또한 IEC 61970의 GID를 구현하지 않은 애플리케이션에 대해서도 메시지 변환을 통해서 연동이 가능하다. 본 논문에서 제시한 메시지 통합버스는 향후 언어와 운영플랫폼에 독립적인 운영을 위하여 Java 플랫폼과 XML 웹서비스를 채택하여 웹서비스용 WSDL로 설계하였다. 전력분야 메시지 통합버스용 WSDL과 서비스 모듈을 개발함으로써, 앞으로 IEC 61970과 같은 국제 표준을 기반으로하는 전력시스템 개발에 쉽게 적용이 가능하리라 전망된다.

참 고 문 헌

[1] Intelligrid Architecture Volume I, EPRI, 1999

[2] IEC 61970-301, “EMS-API Part 301 : Common Information Model Base”

[3] IEC 61970: Energy Management System Application Program Interface (EMS-API) Part 402: Common Services.

[4] SOA using Java Web Services, Prentice Hall, 2007

[5] Websphere Business Integration Primer: Soa, Web Services, and Esb, Michele Chilanti, Vinod Jessanil, 2007.12

[6] Utility Management System (UMS) Data Access Facility Specification, OMG, 2005.7

[7] Data Acquisition from Industrial Systems Specification, OMG, 2005.6