



**MultiSpeak[®] Version 3.0
User's Guide**

1/1/2006

Prepared By

Gary A. McNaughton, P.E.
Warren P. McNaughton, P.E.
Cornice Engineering, Inc.
P.O. Box 2350
Pagosa Springs, Colorado 81147-2350

For

National Rural Electric Cooperative Association
4301 Wilson Boulevard
Arlington, Virginia 22203-1860

The National Rural Electric Cooperative Association (NRECA) is the national service organization of more than 900 rural electric systems. These cooperatively owned utilities own and operate about 44 percent of the miles of distribution lines in the nation to provide power to less than 10 percent of the nation's people, primarily in the sparsely populated rural areas of 46 states.

NRECA was founded in 1942 to unite rural electric systems in a way that would permit them to develop the services and support they needed to properly serve rural America. NRECA is one of the largest rural-oriented cooperative organizations in the United States.

The NRECA Cooperative Research Network, a service of NRECA that has supported this project, was created to conduct studies and carry out research of special interest to rural electric systems and their consumers.

Copyright © 2002-2006 by National Rural Electric Cooperative Association

All rights reserved. Reproduction in whole or in part is strictly prohibited without prior written approval of the National Rural Electric Cooperative Association.

Legal Notice

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, NEITHER NRECA NOR CORNICE ENGINEERING, INC., MAKES ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER NRECA NOR CORNICE ENGINEERING, INC., SHALL BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE OR USE OF THIS MATERIAL. THE INFORMATION CONTAINED IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE.

Prepared by

Cornice Engineering, Inc.
P.O. Box 2350
Pagosa Springs, Colorado 81147-2350
(970) 731-1508
(970) 731-1509 (Fax)
E-mail: gmчнаughton@frontier.net

Table of Contents

Section		Page
	Foreword	v
1	MultiSpeak® Basics	1-1
	1) <i>What is MultiSpeak®?</i>	1-1
	2) <i>How was MultiSpeak developed?</i>	1-1
	3) <i>What does MultiSpeak do?</i>	1-1
	4) <i>What is the current version of MultiSpeak?</i>	1-2
	5) <i>What software applications does MultiSpeak cover?</i>	1-4
	6) <i>What are MultiSpeak defined functions?</i>	1-5
	7) <i>How do MultiSpeak functions relate to my software products?</i>	1-5
	8) <i>Is MultiSpeak a “plug-and-play” solution? Doesn’t this take care of all my integration needs?</i>	1-7
	9) <i>Is MultiSpeak a product?</i>	1-8
	10) <i>How do I make sure that my software uses MultiSpeak?</i>	1-8
	11) <i>What does MultiSpeak cost?</i>	1-8
	12) <i>Why is MultiSpeak important?</i>	1-9
	13) <i>Does MultiSpeak send sensitive data securely?</i>	1-10
2	About MultiSpeak Version 3.0	2-1
	14) <i>How has MultiSpeak 3 changed from MultiSpeak 2.2?</i>	2-1
	15) <i>Why was the change made to web services?</i>	2-3
	16) <i>Why was compliance testing replaced with interoperability testing?</i>	2-3
	17) <i>Is MultiSpeak 3 backward-compatible with MultiSpeak 2.2?</i>	2-4
	18) <i>Is MultiSpeak 3 backward-compatible with MultiSpeak 1.1?</i>	2-4
3	About Web Services	3-1
	19) <i>What are web services?</i>	3-1
	20) <i>How do web services work?</i>	3-1
	21) <i>Why are web services important?</i>	3-1
4	Compatibility with MultiSpeak	4-1
	22) <i>What are the different types of testing that could help me determine if two software packages will interoperate?</i>	4-1
	23) <i>What testing has MultiSpeak provided in the past?</i>	4-1
	24) <i>What testing does MultiSpeak provide now?</i>	4-2
	25) <i>Why was the change in testing made?</i>	4-3
	26) <i>Why is the assertions document important to me?</i>	4-3
	27) <i>How can I use the assertions document?</i>	4-4
	28) <i>How can I check vendors’ claims of MultiSpeak compatibility?</i>	4-4
	29) <i>Will compliance (or interoperability) testing ensure that two software applications work together seamlessly at my utility?</i>	4-4

5	Using MultiSpeak	5-1
	<i>30) Is any special hardware required to implement MultiSpeak?</i>	5-1
	<i>31) Do I need a web server to use MultiSpeak web services?</i>	5-1
	<i>32) What is a MultiSpeak Participating Vendor?</i>	5-1
	<i>33) What is a MultiSpeak Software Developer?</i>	5-2
	<i>34) What is a MultiSpeak Integrator?</i>	5-2
	<i>35) The type of software product that I want to integrate is not covered by MultiSpeak; what can I do?</i>	5-3
	<i>36) How does MultiSpeak evolve?</i>	5-3
	<i>37) What else can I do to further the development of MultiSpeak?</i>	5-3
6	For More Information	6-1
	Appendixes	
A	MultiSpeak Functions and Software Applications	A-1
B	MultiSpeak Defined Interfaces	B-1
C	Choosing Between Communications Options	C-1
	C.1 Communications Transport Options	
	C.2 Message Exchange Patterns	
	C.3 File-Based versus Real-Time Communications	
D	Web Services Basics	D-1
	D.1 Introduction to Web Services	
	D.2 Creating Meaningful Web Services Messages	
	D.3 Web Services Framework	
E	How MultiSpeak Implements Web Services	E-1
	E.1 What MultiSpeak Provides	
	E.2 When You Need More than MultiSpeak Provides	
F	How to Understand a MultiSpeak Interoperability Assertion	F-1
G	Sample MultiSpeak Interoperability Assertion	G-1
H	Referenced Standards	H-1

Foreword

The MultiSpeak Initiative is a collaborative effort between the National Rural Electric Cooperative Association and software vendors serving the electric utility industry. The initiative has developed a specification package that defines standard data interfaces between software applications commonly used by small electric utilities. MultiSpeak has been steadily improving since the first release of the specification in 2000. It has evolved in Version 3.0 to (i) better reflect utility needs, (ii) increase the power and functionality of the software interfaces and (iii) make use of the most modern software tools and standards.

The MultiSpeak® specification package consists of the following documents:

1) MultiSpeak® Version 3.0 Specification. The specification more fully describes how the MultiSpeak standard should be applied. It (i) defines the underlying structure of the standard, (ii) lays out the agreements that form the basis for creating compatible software interfaces, (iii) outlines the choice of protocols that should be used to implement interfaces and (iv) establishes guidelines for how the chosen technologies should be implemented to maximize the potential for seamless integration among applications developed by different software vendors.

2) MultiSpeak.xsd, MultiSpeakBatchMsgHeader.xsd, MultiSpeakWebServicesMsgHeader.xsd and mspGeometry.xsd. These files are Extensible Markup Language (XML) schemas that document the data definitions and messaging features described in the specification document.

3) MultiSpeak V3 Schema Documentation.zip. This archive file contains a Hypertext Markup Language (HTML) file documenting the XML schemas in a user-friendly, graphic format.

4) MultiSpeakWebCopy.zip. This archive contains a complete set of Web Service Description Language (WSDL) files and supporting documentation that describe and document the MultiSpeak web services used in real-time interfaces. WSDL files are computer-readable, XML-formatted files that can be used by software development toolkits to create MultiSpeak-compatible web services.

5) MultiSpeak® Version 3.0 Security Implementation Guidelines. This document gives additional information about securely implementing MultiSpeak interfaces using Secure Sockets Layer (SSL) for transport layer security.

This user's guide has been developed for utility personnel wishing to better understand the MultiSpeak® Version 3.0 specification and its role in improving integration among their existing and planned software applications. The guide is intended to provide the information utility personnel most often need to determine how to specify MultiSpeak-compatible software applications as well as to evaluate the importance of MultiSpeak compatibility in integrating automation systems. The guide is written in a question-and-answer format.

1

MultiSpeak® Basics

1) What is MultiSpeak®?

MultiSpeak® is a specification for the automation of business processes and the exchange of data among software applications commonly applied in small electric utilities, such as electric cooperatives. The foundation of the specification is an agreement on the details of the data objects that need to be exchanged to integrate disparate software applications more fully. The specification also defines standardized interfaces among software applications that are commonly applied in small utilities. It contains guidelines on how those interfaces should be implemented to maximize the potential for compatible programs, developed by different software vendors, to interoperate. Furthermore, MultiSpeak establishes a software testing program to assure utilities that software products claiming compatibility with the specification actually include appropriate functionality.

The MultiSpeak specification is intended to assist vendors and utilities to develop interfaces so that software products from a variety of vendors can interoperate without the need for extensive custom interface development. As a result, it is expected that MultiSpeak will help make available cost-effective, integrated software applications to serve the business needs of utilities.

2) How was MultiSpeak developed?

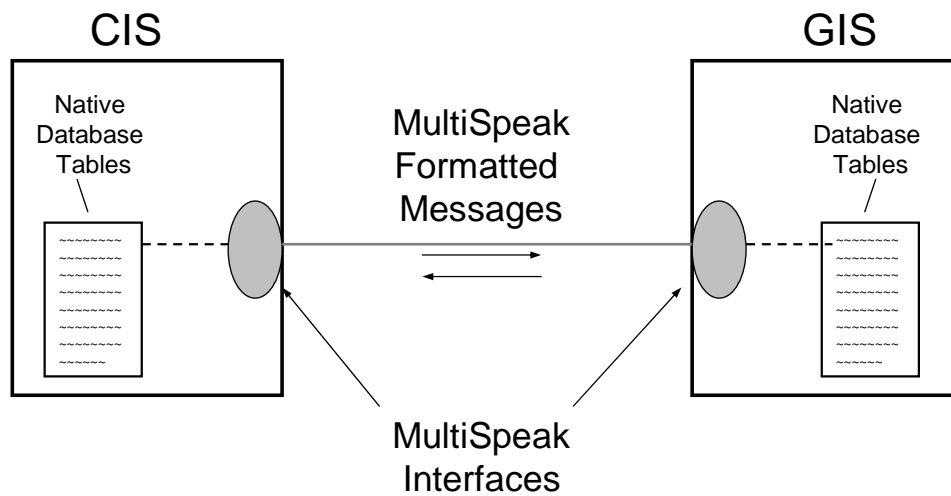
MultiSpeak was developed by the MultiSpeak Initiative, a collaborative effort between the National Rural Electric Cooperative Association and software vendors serving the electric utility industry. Vendors, along with consultants hired by NRECA, have met more than 20 times since October 1999 to design the required software interfaces. Significant funding for MultiSpeak has been provided by the Cooperative Research Network of NRECA and, more recently, by NRECA member dues. The vendor community has also provided substantial funding in the form of staff time, travel and other development costs.

3) What does MultiSpeak do?

MultiSpeak defines software interfaces, data objects and message structures. These interface, data and message definitions permit vendors to write a common interface that facilitates communication with another type of software. The MultiSpeak specification is designed so that the details of an interface should not vary substantially, regardless of the vendor of the application with which the product is to integrate. Similarly, changes in the structure of one vendor's software should not require changes in the agreed-upon interface. Since the

participating vendors support a variety of hardware and software platforms, database programs and programming languages, the MultiSpeak Initiative has developed an approach that is independent of platform or database structure--one that relies on common data definitions and agreed-upon data flows.

The approach adopted by MultiSpeak was to define interfaces rather than to define a common, comprehensive data model with which all vendors would be required to comply. This approach permits vendors to write and maintain an interface that marshals the required information from their native data structure, converts those data into Extensible Markup Language (XML) data packets and transports those packets in the form of predefined messages. The receiving application is responsible for unbundling the message, parsing the XML and taking the appropriate action with the resulting data objects. Figure 1-1 illustrates this approach schematically using a customer information system (CIS) and a geographic information system (GIS) that exchange information from their native databases. The approach adopted by MultiSpeak does limit somewhat the means that can be used to achieve integration. Any scheme that requires direct access to the native database, such as Structured Query Language (SQL) or open database connectivity (ODBC), will not achieve the goals of the group.



**Figure 1-1
MultiSpeak Approach to Common Interfacing**

4) What is the current version of MultiSpeak?

MultiSpeak is currently in its third release. Table 1-1 compares the most recent release, MultiSpeak Version 3.0 (MultiSpeak 3), with the two prior releases, Versions 1.1 and 2.2.

**Table 1-1
Characteristics of Different Versions of the MultiSpeak Specification**

Version	Release Date	Applications Covered	Number of Defined Interfaces	Number of Data Objects	Supports Batch Transfers?	Supports Real-Time Data Transfers?	Communications Options Supported	Type of Testing Provided	Comments
1.1	12/2000	<ul style="list-style-type: none"> • CIS • GIS • Engineering analysis • IVR • Automated staking 	7	45	✓		Batch file transfers	Compliance	<ul style="list-style-type: none"> • Successful proof-of-concept • Interfaces defined between applications
2.2	10/2003	All the applications covered by V1.1 plus <ul style="list-style-type: none"> • SCADA • AMR • OMS • Load management • CRM 	29	250 (approx.)	✓	✓	Batch file transfers SOAP messages TCP/IP sockets	Compliance	<ul style="list-style-type: none"> • Used then-state-of-the-art messaging framework for real-time communications. • Added incremental updates and deletions • Defined interfaces between functions
3.0	12/2005	All the applications covered by V2.2	29	275 (approx.)	✓	✓	Batch file transfers Web services	Interoperability	<ul style="list-style-type: none"> • Applies web services, current best integration communications technology • Defines more than 200 unique web service methods • Moved to interoperability testing to provide better assurance to utilities • Supports incremental updates and deletions • Defines interfaces between functions

5) What software applications does MultiSpeak cover?

The MultiSpeak Initiative initially focused on five back-office software applications (see table 1-1):

- Customer information systems (CIS)
- Geographic information systems (GIS)
- Engineering analysis (EA)
- Interactive voice response systems (IVR)
- Automated staking

Version 1.1 of the MultiSpeak specification defined seven interfaces and developed a data dictionary for the information that could meaningfully be exchanged among these applications. Figure 1-2 illustrates the applications, data flows and interfaces defined in the MultiSpeak Version 1.1 specification. Only batch file transfers were defined in Version 1.1. In addition, no provisions were made for incremental updates or for deletion of data.

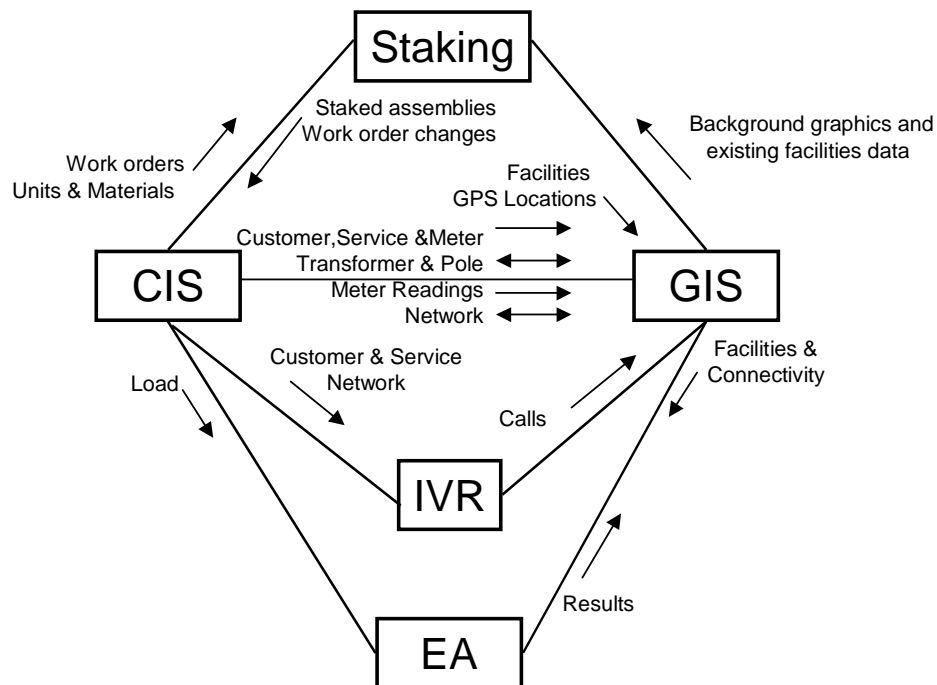


Figure 1-2
MultiSpeak Version 1.1 Process Model

MultiSpeak Version 2.2 (MultiSpeak 2.2) extended the coverage of the specification to include the following systems:

- Supervisory control and data acquisition (SCADA)
- Automated meter reading (AMR)

- Outage management (OM)
- Load management (LM)
- Customer relationship management (CRM)

MultiSpeak 3 supports all the application types covered in Version 2.2.

MultiSpeak 3 carries forward provisions for batch transfers where such transfers make sense to support utility business processes. In addition, MultiSpeak 3 includes support for real-time integration using web services. For more information about web services, see appendix D. For more information about how MultiSpeak implements web services, see appendix E. Both the batch and web services capabilities can support incremental updates and deletions with a high degree of specificity. Figure 1-3 shows the MultiSpeak defined software functions that are covered by MultiSpeak Version 3.0 and the interfaces between them. The interfaces are identified on figure 1-3 by numbers in ovals and are defined in greater detail in appendix B.

6) What are MultiSpeak defined functions?

Versions 2.2 and 3.0 of MultiSpeak are defined in terms of software *functions*. The boxes in figure 1-3 illustrate the MultiSpeak defined functions. Functions are connected using interfaces across which data are exchanged.

For a more detailed description of MultiSpeak defined functions and how they relate to vendor software products, see appendix A.

7) How do MultiSpeak functions relate to my software products?

A software product that a utility might purchase can fulfill one or more MultiSpeak functions. When purchasing software, it is important that the utility discuss with its software supplier what functions the supplier's application package provides. There might be a direct relationship between the MultiSpeak function and the application, such as for a complete engineering analysis application, which includes only the EA MultiSpeak function. In contrast, an automated meter reading application might contain only one, or as many as four, MultiSpeak functions: (i) meter reading (MR), (ii) connect/disconnect (CD), (iii) load profile (LP) and (iv) outage detection (OD).

For a more detailed description of MultiSpeak defined functions and how they relate to vendor software products, see appendix A.

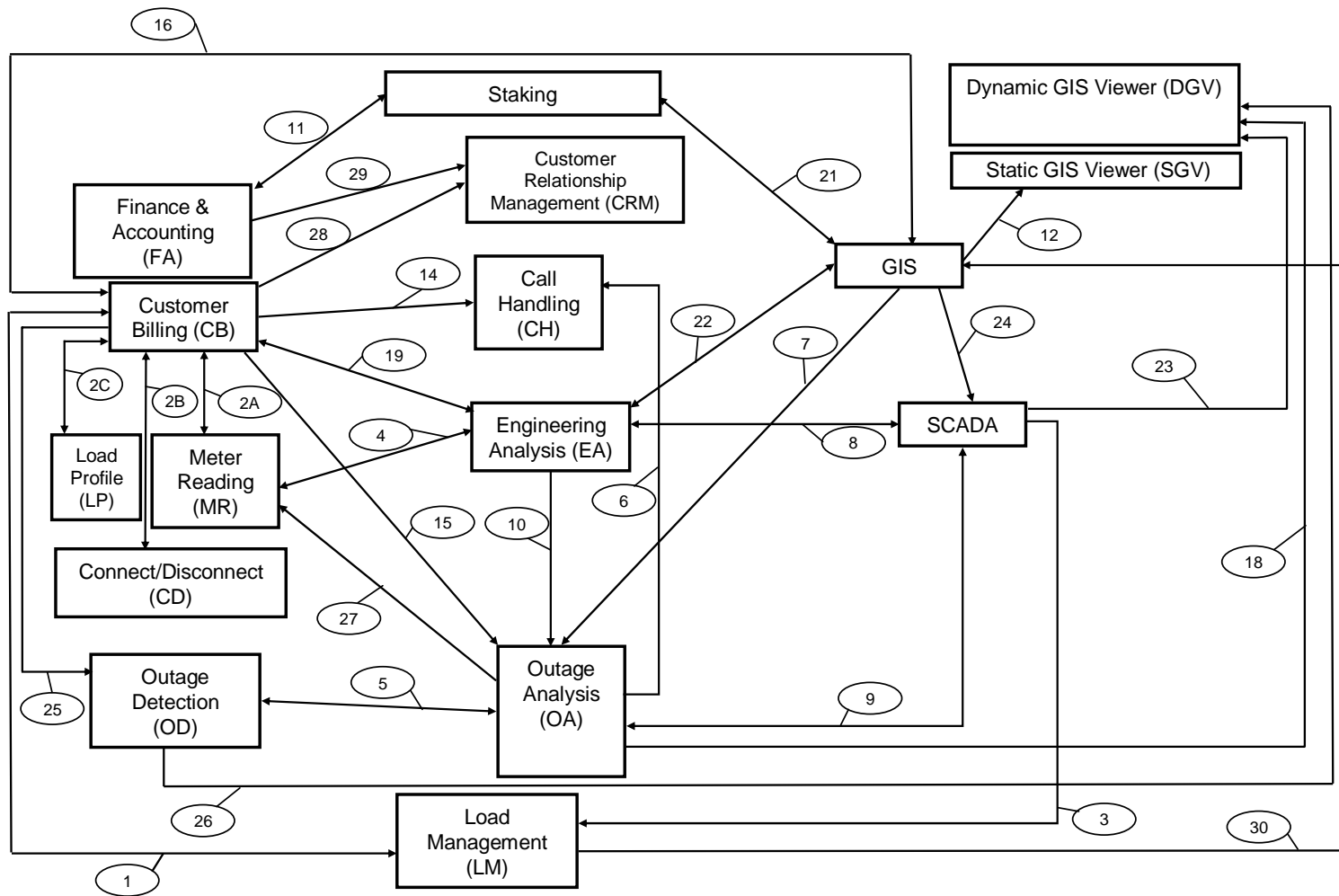


Figure 1-3
MultiSpeak Version 3.0 Process Model

8) Is MultiSpeak a “plug-and-play” solution? Doesn't this take care of all my integration needs?

Unfortunately, the requirements for integration of software applications in the utility industry are just too complicated and diverse for any single specification to provide a universal “plug-and-play” solution for every utility. **The MultiSpeak specification has been designed from the ground up to address the majority of the common integration needs of small utilities, but it is not guaranteed to solve all problems.** Your vendors may still have to create special interfaces or fine-tune their MultiSpeak interfaces to meet the specific needs of your utility. This would be especially true if you have any of the following circumstances:

- Not all your software products support MultiSpeak 3.
- Your MultiSpeak-compatible software applications do not support all the defined interfaces (see appendix B for more information on this issue). Software that needs to be integrated uses significantly different versions of MultiSpeak or sends its data using different communications transfer options (see table 1-1 or appendix C for more information on this issue).
- MultiSpeak has been crafted to include the vast majority of data items in common use in small electric utilities. However, you might have a special situation for which you want to send data items that have not been defined in MultiSpeak.

Your utility may also have made choices during implementation of your software solutions that make your particular installation nonstandard; such choices may affect the interoperability of MultiSpeak interfaces. Specific circumstances that may affect the performance of MultiSpeak interfaces at your utility include the following:

- **Uses of nonstandard data fields to store critical data.** For instance, if you store the customer identification number in a field other than the one recommended by your vendor, then without customization, your vendor's standard MultiSpeak interface might send the wrong data, or perhaps not have data to send, for this key database field. Other applications that receive customer data then would not be able to correlate that data with the correct customer or service location without a consistent database key.
- **Poor-quality data.** For instance, if you have incorrect telephone numbers in the customer database, then the MultiSpeak interface will send those incorrect numbers to other applications. An outage management or interactive voice response system that relies on such improper data could

perform incorrectly. This would not be the fault of the MultiSpeak interface; it would be a result of poor-data quality.

- **Use of unusually formatted data.** If you have chosen to format a key database field differently than is recommended by your vendor, then it is possible that the vendor's standard interface code could not correctly transmit your data without customization.

9) Is MultiSpeak a product?

MultiSpeak is a specification that defines standardized software interfaces; it is not a software product.

10) How do I make sure that my software uses MultiSpeak?

There are two ways for you to get software that supports MultiSpeak functionality: (i) purchase MultiSpeak-compatible applications directly from your current software vendors or (ii) write your own compatible interfaces.

If one of your current software vendors does not provide MultiSpeak-compatible interfaces, you may use the MultiSpeak specification to write your own compatible interface for that product or get a third party to write such an interface. The MultiSpeak specification is freely available to all interested parties from the MultiSpeak Initiative web site (www.MultiSpeak.org).

11) What does MultiSpeak cost?

There is no charge directly to utilities from NRECA or the MultiSpeak Initiative for the use of MultiSpeak. However, users should be aware that vendors have invested significant time, energy and capital in the development of MultiSpeak-compatible interfaces, and most must recover these investments in some manner. Some vendors may choose to offer MultiSpeak interfaces as part of their core software package and recover the costs from all users; others may offer MultiSpeak interfaces as add-on components and recover the costs for such add-ons from the users that request them. It should be noted, however, that as the vendors serving the electric utility industry increasingly adopt MultiSpeak-compatible interfaces, the overall cost of interoperable software should go down for all users.

12) Why is MultiSpeak important?

MultiSpeak is important to both the user and the vendor communities.

For **utilities**, MultiSpeak offers the following advantages:

- Integration streamlines business processes, improving employee efficiency and reducing costs. MultiSpeak makes it easier to achieve a high level of application integration.
- Integration, facilitated by MultiSpeak, makes it possible to provide more timely and more accurate customer service.
- MultiSpeak interfaces minimize the need for expensive and maintenance-intensive custom interfaces.
- Integration with MultiSpeak enables utilities to focus on “best-of-breed” software without the fear of lack of integration.
- MultiSpeak integration reduces the deployment time and risk in purchasing new automation applications.
- The use of common interfaces reduces the cost and hassle of changing from one application to another of the same type or upgrading to a new version of the same type of software.

For **software vendors**, MultiSpeak has the following advantages:

- Using MultiSpeak interfaces minimizes the need for developing multiple interfaces with other vendors' software, thus reducing time spent in developing and maintaining custom interfaces.
- Reducing interface development and maintenance efforts releases programmers to improve existing products or develop new offerings.
- Eliminating the need for custom interfaces reduces complexity and time to market for new products.
- Standardizing integration reduces support headaches.
- Increasing the willingness of utilities to invest in new applications helps assure them that integration problems are minimized.

13) Does MultiSpeak send sensitive data securely?

MultiSpeak does not inherently include considerations of data security. However, messages that exchange MultiSpeak-formatted data may be secured at the option of the utility if software vendors provide security support. For more information about how security can be implemented over a MultiSpeak interface, see appendix E and the separate document, *Security Considerations in Implementing MultiSpeak®-Compliant Applications*, which is available from the MultiSpeak web site (www.MultiSpeak.org).

2

About MultiSpeak Version 3.0

14) How has MultiSpeak 3 changed from MultiSpeak 2.2?

MultiSpeak 3 is significantly improved and expanded over Version 2.2 in a number of critical respects. Table 2-1 summarizes these changes.

- Version 3.0 carries forward the interfaces that were defined in Version 2.2 (see figure 1-3), but enhances some of the underlying data definitions and better supports some business processes. Interfaces that underwent substantial changes include the following:
 - **The staking-GIS interface (interface #21).** This interface received fundamental changes that made it easier to (i) facilitate the change between the GIS's features-based data model and the staking system's construction units-based view of data, (ii) lump facilities all installed at a single work order station and (iii) synchronize the staking and GIS applications in real time.
 - **The OA-OD and OA-CH interfaces (interfaces #5 and 6, respectively).** The data objects and messages were substantially changed to create a higher level of integration between the outage analysis function and the other functions.
 - **The EA-GIS interface (interface #22).** The ability was added to have changes made in either application be subsequently reflected in the other, rather than requiring one application to be the master or owner of the data. The concept of lumping changes into defined sessions was added; this permitted the EA and GIS applications to resynchronize data based on referring to the last session that was committed.
 - **All interfaces with the MR function (Interfaces #2A, 4 and 27).** The meter reading and time-of-use meter reading data objects were simplified.
- Version 2.2 supported real-time integration using a generic, standardized messaging framework and could be implemented using either (i) Simple Object Access Protocol (SOAP), a web-based means to send packets of XML data or (ii) TCP/IP sockets, a robust means for programs to talk directly with each other. In Version 3.0, the messaging framework and

**Table 2-1
Comparison of MultiSpeak Versions 2.2 and 3.0**

Version	Significantly Improved Interfaces	Communications Options Supported	Type of Testing Provided	New Functionality	Comments
2.2		Batch file transfers SOAP messages TCP/IP sockets	Compliance		<ul style="list-style-type: none"> • Used then-state-of-the-art messaging framework for real-time communications
3.0	MR-CB (#2A) MR-EA(#4) OA-OD (#5) OA-CH(#6) STK-GIS(#21) EA-GIS (#22) MR-OA(#27)	Batch file transfers Web services	Interoperability	<ul style="list-style-type: none"> • Breaks large XML files into blocks of arbitrary size • Added means to resynchronize clients and servers if communication is lost • Added means to synchronize data by sessions • Added means for applications to extend data objects at run time 	<ul style="list-style-type: none"> • Applies web services, current best integration communications technology • Defines more than 200 unique web service methods • Moved to interoperability testing to provide better assurance to utilities

multiple real-time transport options were replaced with a single real-time transport: well-defined and clearly standardized web services.

- Version 3.0 added a means to break large XML file transfers into blocks of arbitrary size to improve performance, minimize problems with parsing unmanageable files and reduce network bandwidth constraints.
- Version 3.0 added a means to resynchronize the client and server in publish/subscribe interfaces without requiring the client to request a complete dump of all information from the server.
- Version 3.0 added a means to synchronize clients and servers based on groups of updates referred to as *sessions*.
- Version 3.0 also added a new way to extend data objects at run time for real-time interfaces.

15) Why was the change made to web services?

The Version 2.2 messaging framework was very powerful and flexible and was state of the art at the time Version 2.2 was released. However, the flexibility of the messaging framework came at the price of additional complexity. Further, the flexibility increased the possibility that two vendors could implement compliant interfaces that could not interoperate. In addition, many of the MultiSpeak vendors found the complexity daunting.

The MultiSpeak Initiative had been following the web services technology for some time, but at the time that Version 2.2 was released, web services standards were not stable and there was not an industry-wide consensus that web services would achieve significant market penetration. In the time between the release of Version 2.2 and late 2004, web services evolved and matured to the point that their application to utility needs was appropriate and their adoption industry-wide was nearly ensured. Furthermore, a significant number of software development toolkits that supported the new web services standards had become available, which enabled the MultiSpeak participants to efficiently develop web services interfaces. For these reasons, the change from SOAP and TCP/IP sockets transports to web services was incorporated into MultiSpeak 3 to make a simpler, more effective and more powerful data transfer possible.

For more information about web services, see questions #19-21 and appendixes D and E.

16) Why was *compliance testing* replaced with *interoperability testing*?

Version 2.2 interfaces were certified using **compliance** testing, a means to verify that a single software application, acting in isolation, supported the requirements of the specification. Two compliant applications that shared a common interface

were more likely, but not certain, to work together. Version 3.0 interfaces are tested for **interoperability**. An interoperability test certifies that two software applications (typically provided by two different vendors) actually work together to support some utility business process.

It is still possible for a vendor to test a single product in isolation by testing the product's compatibility with a batch data transfer or using a web services testing harness. The testing harness acts as a universal client or server, providing a partner for the software under test. For more information about MultiSpeak compatibility, see questions #22-29. For more information about how to interpret interoperability test results, see appendixes F and G.

17) Is MultiSpeak 3 backward-compatible with MultiSpeak 2.2?

Batch data exchanges are quite similar between Versions 2.2 and 3.0 for some interfaces. It is possible that some Version 2.2-compliant software will interoperate with some Version 3.0 software if batch transfers are used. See figure C-1 to see which interfaces might be batch and which are defined only for real-time data exchanges. It is recommended that you discuss this issue with the vendors of your software products to see if interoperation is possible.

Although in many cases the data payloads sent in the two most recent versions will be nearly identical, the different communications transport options will prevent any chance of interoperation of two applications written to different versions of MultiSpeak without a middleware product to perform message translation. As a result, real-time Version 2.2 interfaces will not interoperate with Version 3.0 web services interfaces.

An example of this situation is the SCADA-EA interface. Two vendors developed interoperable Version 2.2 SCADA-EA interfaces using the TCP/IP sockets transport. If a vendor now develops either a SCADA or EA Version 3.0 web services interface, it will not interoperate with the TCP/IP sockets interface, even if the data packets are identical, because of the differences in transport used by the two interfaces.

The MultiSpeak Initiative has requested, but not required, that vendors provide some form of backward compatibility in their MultiSpeak 3 products where the nature of the interface permits the potential for useful interoperation. Utilities should discuss with their software vendors what backward-compatibility features are available in specific software products.

18) Is MultiSpeak 3 backward-compatible with MultiSpeak 1.1?

Unfortunately, because of the comprehensive nature of the improvements in Versions 2.2 and 3.0, it will be difficult for vendors to ensure backward compatibility with interfaces that have been tested compliant with Version 1.1. The MultiSpeak Initiative has requested, but not required, that vendors consider

providing some form of compatibility in their MultiSpeak 3 products with the earlier versions, where the nature of the changes in the interface permits. Some interfaces--for instance, staking-GIS--have changed so significantly that compatibility with Version 1.1 will not be possible. Utilities should discuss with their software vendors what backward-compatibility features are available in specific software products of interest.

3

About Web Services

19) What are web services?

A web service is a standardized means for one software application to exchange data with, or invoke actions on, another application. Each web service includes one or more *methods*. A method can be thought of as a contract between two software applications that describes the details of how a service is invoked and delivered. If you have ever used the Internet to check a stock's price or to book an airline flight, you have used a web service. In the case of booking a flight, the airline exposes a well-defined web service that a travel agency site can use to request the availability and pricing for seats and subsequently to book reservations. In this case, there are at least two methods: (i) one that describes the way that a travel agent may request availability and (ii) another to book the reservation.

20) How do web services work?

Web services typically use (i) Extensible Markup Language (XML) to define data and message elements, (ii) Simple Object Access Protocol (SOAP) to compile data into single messages and (iii) Web Services Description Language (WSDL) to link messages into business process steps. Messages are typically transported over Hypertext Transport Protocol (HTTP), using a conventional web server. Most products that implement MultiSpeak 3 web services will package a web server with the product, so that the application and its web service communications component are self-contained. For more information about web services, see Appendix D, Web Services Basics.

21) Why are web services important?

There are a number of reasons that web services facilitate a quantum leap in the capability to integrate software:

- Web services have made it possible to integrate business processes, not just data. Thus, utility employees can stay in the application software with which they are most familiar, rather than changing to another, less familiar application. This will facilitate improvements in employee efficiency.
- Web services provide well-defined and self-describing standardized interfaces. Well-defined method calls make clear to all parties exactly what is expected in order to communicate.

- Web services use the same protocols that have been standardized for use in the World Wide Web. Since the web has become nearly indispensable for modern commerce, the products, standards and protocols that support it have been subjected to intensive testing.
- Readily available software development toolkits make it much easier to develop and support interfaces for web services than for traditional interfaces.
- Web services support is provided in nearly all modern programming languages, operating systems, and server platforms. Web services have the potential to seamlessly integrate applications developed with different programming tools and hosted on radically different operating systems.
- The use of web services decouples the client and server programming by defining the nature of the *interface* between the two applications. Either party could change its programming code, or where that code is hosted, without the need for the other party to change its product. This supports a basic tenet of MultiSpeak that was established early on (see question #3).
- It is possible to add support for new methods to a web service while maintaining backward compatibility with existing methods. This will make it more likely that future enhancements to programs can be made without affecting existing functionality.

4

Compatibility with MultiSpeak

22) What are the different types of testing that could help me determine if two software packages will interoperate?

Software testing can be performed at three levels. Successive levels of testing provide increasing assurance that two applications will interoperate seamlessly.

1) Compliance testing. Testing performed to assess whether the software meets the requirements of the specification. This type of testing has been performed by the NRECA-approved MultiSpeak testing laboratory on products submitted from a single vendor for MultiSpeak Versions 1.1 and 2.2 compliance.

2) Interoperability testing. Testing performed between pairs of software applications to ensure that they work interfaces seamlessly. It is possible that two software applications could both be compliant and yet not work together seamlessly without minor modifications. This type of testing is currently being provided for Version 3.0-compatible products.

3) Site conformance testing. Testing performed at an individual utility site to ensure that pairs of products interoperate seamlessly *given the utility's specific data structures and application data*. Even if two software applications have passed both compliance and interoperability testing, it is still possible that choices of data structure made by a utility or poor-quality data stored in one of the applications may result in interface errors. Such errors may require work on the part of the software vendors or data scrubbing on the part of the utility.

23) What testing has MultiSpeak provided in the past?

For Versions 1.1 and 2.2, NRECA sponsored an independent testing organization to evaluate compatibility with the requirements of the MultiSpeak specification, and a testing protocol was developed. The testing organization certified that products submitted for testing met the requirements of the specification (**compliance testing**).

Compliance certification is for products, not vendors; certification is by interface. Thus, a particular product is deemed to be, for example, compliant with the requirements of the customer billing-geographic information system (CB-GIS) interface. If the submitted product passed the certification test for at least one

defined interface, it was declared to be compliant. **Note that the MultiSpeak-compliant designation does not ensure that the product is compliant with every applicable interface.** For instance, as shown in figure 1-3, seven possible interfaces with a GIS are defined in MultiSpeak: interfaces with staking (#21), EA (#22), OA (#7), CB (#16), SCADA (#24), LM (#30) and SGV (#12); a GIS product could be designated as MultiSpeak compliant if it had passed testing for just one of these seven interfaces. Recognizing compliance on an interface-by-interface basis permits vendors to submit interfaces for testing as they become available rather than waiting for all interfaces to be ready for testing.

Compliance testing is no longer offered for newly developed Version 1.1 interfaces; testing for Version 2.2 interfaces will continue until July 1, 2007. No compliance testing will be offered for newly developed Version 3.0 interfaces; instead, interoperability testing will be performed, as described in the next question.

24) What testing does MultiSpeak provide now?

Interoperability testing is new to the effort with Version 3.0. In interoperability testing, two products (typically from different vendors) are submitted for joint testing. The vendors develop an *assertions document* that states the joint MultiSpeak functionality that they wish to claim and how those capabilities may be used to support utility business processes. The independent testing agency then verifies that the two products integrate as described and that the integration is achieved using MultiSpeak. The verified assertions document is then posted on the MultiSpeak web site (www.MultiSpeak.org) for download by interested parties. You may use these documents as a way to see, in detail, exactly what MultiSpeak-compatible integration is included in vendor product interfaces.

It is not necessary for two products to support any minimum set of shared functionality defined by MultiSpeak to pass interoperability testing. The assertions document describes in detail what integration features the products support. It is up to the utility to determine whether the integration offered meets the needs of its business processes.

For more information about what is included in an assertions document, see appendixes F and G.

This type of testing is performed by the NRECA-approved testing agent when pairs of vendors specifically request and pay for such testing. It should be noted that, in some cases, products may be interoperable, but the vendors have not chosen to pay for testing to document this fact. For some vendors, whose products are expected to integrate with a large number of software products offered by different vendors, the cost of testing all possible combinations of software can be burdensome. Thus, they may choose not to perform all these tests.

If a vendor has not yet tested an application of interest, a utility may choose to require interoperability testing for a pair of products that have not yet been jointly tested as a software quality control step during software procurement. In such a case, the cost of the test may be borne by the utility or by the vendor(s) of the software in question.

25) Why was the change in testing made?

One of the challenges for utility staff members with Version 2.2 compliance testing was to determine (i) what level of MultiSpeak compliance was supported by vendor software, (ii) what such compliance might mean for interoperation of two software products and (iii) what business processes could be enhanced by any interoperation that was possible. Utilities were often left wondering, "Does this mean that Product X and Product Y talk to each other?" Interoperability testing and the assertions document were developed to ease this burden of interpretation and to remove the uncertainty about whether two specific products work together.

26) Why is the assertions document important to me?

MultiSpeak interoperability testing uses an assertions document to describe in detail the MultiSpeak-compatible integration features shared by two software packages. The purpose of the assertions document is to educate utility staff members on the level of MultiSpeak integration to expect between two specific products that have completed Version 3.0 interoperability testing. Assertions documents are written with the utility audience in mind. They answer the following questions:

- What can I do with the software products that would not be possible without the MultiSpeak integration?
- Why is this shared functionality important to my business processes?
- How have the vendors used MultiSpeak to accomplish the integration described?
- How much of the MultiSpeak-defined interface capability do these two products support?

Note that assertions documents do not necessarily describe *all* the capabilities of an interface between two products, only those that are accomplished using MultiSpeak. It is certainly possible for two vendors to support other integration features that are not included in MultiSpeak; for a description of such features, contact the vendors directly.

For more information about what is included in an assertions document, see appendixes F and G.

27) How can I use the assertions document?

Utilities can use assertions documents in two ways:

- 1) You have MultiSpeak-compatible software and you want to know what it does, and does not, do.
- 2) You are considering the purchase of software and want to compare the MultiSpeak integration supported by alternative vendor offerings.

28) How can I check vendors' claims of MultiSpeak compatibility?

If a product has passed either compliance testing or interoperability testing, it may be said to be MultiSpeak-compatible. The best way to determine if a product has been tested for MultiSpeak compatibility, and the extent of that compatibility, is to check the MultiSpeak Initiative web site (www.MultiSpeak.org). The web site is updated regularly with the most recent product compatibility status. Although vendors may incorporate features of MultiSpeak compatibility in their products without submitting their products for compliance or interoperability testing, you can be assured that if the product is listed as compliant or interoperable on this web site, it has passed the necessary testing.

29) Will compliance (or interoperability) testing ensure that two software applications work together seamlessly at my utility?

For a number of reasons, such as those listed in question #8, neither compliance nor interoperability testing can provide complete assurance that two software applications will work together seamlessly at your specific installation.

For example, if a utility has chosen to use nonstandard formatting for customer identifiers or has chosen to store those identifiers in a nonstandard data field, the link between customer records in a customer information system and an automated meter reading system could be jeopardized. The MultiSpeak interface might work properly and pass interoperability testing provided the customer identifier is stored as anticipated by the two vendors. However, the fact that the customary data field is missing at a specific utility could cause the link to fail only at that utility. Such a failure would not be a reflection on the generic suitability of the MultiSpeak interface. Rather, the utility's use of nonstandard data is the cause of the problem. In this case, the standard MultiSpeak interface could still be customized slightly to work for this utility, or the utility could decide to change its data structures to match the anticipated condition.

There is another situation in which two products that have passed interoperability testing might work properly at one utility and fail to work together as expected at a different utility: poor data quality. For example, if customer telephone numbers are not accurate in the customer information system, an outage management

system or an interactive voice response system that relies on those telephone numbers might give unacceptable results, even though the telephone numbers were sent accurately from the CIS to the other system.

If it is necessary (i) to document the level of site-specific integration functionality, (ii) to determine the root cause of failure of two software applications to work together as expected at a specific utility site or (iii) to determine what customization is required to achieve the expected result in nonstandard situations at a specific utility, site conformance testing is required. Utilities may request that an independent party perform site conformance testing on their behalf. In addition, utilities may find it desirable to require site conformance testing as an acceptance criterion when purchasing new software products.

5

Using MultiSpeak

30) Is any special hardware required to implement MultiSpeak?

No special hardware is required to implement MultiSpeak-compatible interfaces.

The interface specification has been designed to enable compatible applications to exchange data directly without the need for a special integration server. However, some utilities may wish to implement MultiSpeak using a MultiSpeak-compatible integration server that provides enhanced messaging and application functionality not otherwise available. Capabilities that may be available in a MultiSpeak-compatible integration server include

- Message logging, auditing and historical archiving
- Message queuing, persistence and assured delivery
- Legacy adapters (i.e., custom interfaces with noncompatible applications)
- Business process management (e.g., automatically updating a GIS map and a CIS record as a single logical transaction)

Third-party vendors may provide MultiSpeak-compatible integration servers, although neither NRECA nor MultiSpeak provides such hardware. Furthermore, neither MultiSpeak nor NRECA provides any specification for such hardware or performs any testing on such hardware.

31) Do I need a web server to use MultiSpeak web services?

Web service interfaces do require a web server in order to exchange messages. This does not mean that a dedicated piece of hardware is necessary. A web server can work on any computer that is already connected to the utility's network. Vendors that supply software using web services functionality typically package a web server software component with their system, so that utilities do not need to be concerned with installing or maintaining a separate web server.

32) What is a MultiSpeak Participating Vendor?

A MultiSpeak Participating Vendor is a company that supports the goals of the MultiSpeak Initiative and provides financial support to further those goals. A participating vendor may or may not offer MultiSpeak-compatible products.

Vendors that participate in the MultiSpeak Initiative can be identified by the following black-and-white logo:



33) What is a MultiSpeak Software Developer?

A MultiSpeak Software Developer is a company that provides software products that either have been tested (i) compliant to the MultiSpeak Version 2.2 specification or (ii) interoperable using MultiSpeak 3. A vendor of Version 2.2-compliant products or Version 3.0-interoperable products can be identified by the following color MultiSpeak Software Developer logo:



34) What is a MultiSpeak Integrator?

A MultiSpeak Integrator is a person who has received specific training on the MultiSpeak specification and passed a test on the topics covered during that training. The purpose of this training is to provide a knowledgeable cadre of consultants to help utilities implement interoperable MultiSpeak interfaces. Utility personnel may also attend Integrator training. This is an excellent way for utilities to learn more about how MultiSpeak may be used to integrate software products. The person achieving the MultiSpeak Integrator distinction, or the company that employs him or her, can be identified by the following color MultiSpeak Integrator logo:



35) The type of software product that I want to integrate is not covered by MultiSpeak; what can I do?

The MultiSpeak specification is intended to be a living document that is modified as necessary to meet the most common needs of small electric utilities. If there are common application types that are not covered by the specification-- and for which interfaces with other types of software would be beneficial--utilities or vendors should suggest to the MultiSpeak User's Group, to NRECA or to the MultiSpeak project coordinator that these applications be added to the specification in the near future. For contact information, see section 6.

36) How does MultiSpeak evolve?

The MultiSpeak specification is a collaborative effort. Representatives of NRECA and software vendors meet as required (usually about three or four times each year) to address extensions, corrections or enhancements to the specification. In addition to the scheduled topics for discussion, vendors may bring unsolicited proposals for changes to the specification to the MultiSpeak Initiative at any regular meeting. When a consensus has been reached on the topics of discussion, the required changes are reflected in the specification.

37) What else can I do to further the development of MultiSpeak?

There are three things that a utility can do to further the development of the MultiSpeak specification:

- The best way to influence the future direction of the specification is to attend the MultiSpeak User's Group meeting, which is typically held at NRECA's TechAdvantage conference. Input is solicited at such meetings from both the vendor and user communities for enhancements to the specification. Additional information about the User's Group meetings may be found on the MultiSpeak Initiative web site (www.MultiSpeak.org).
- Consider the benefits of MultiSpeak compatibility when purchasing new or upgraded application software.
- Since vendors have limited development funds, and target those funds to the goals of their users, it is important that utilities communicate with vendors the importance of MultiSpeak compatibility in vendor offerings.

6

For More Information

If the information included in this specification guide doesn't address your questions, you may obtain additional information from the following sources:

MultiSpeak web site (www.MultiSpeak.org)--The MultiSpeak Initiative web site is the best source of up-to-date information about the MultiSpeak Initiative, the MultiSpeak specification and the status of compatibility of vendor software.

MultiSpeak specification--For all the details about how MultiSpeak interfaces work and what data items and software interfaces are covered, see the detailed specification. All portions of the specification are available from the MultiSpeak web site (www.MultiSpeak.org).

MultiSpeak Integrator training--Contact the MultiSpeak project manager for more information about scheduled classes.

NRECA MultiSpeak project manager--Robert Saint, P.E. (contact information: 703.907.5863 or Robert.Saint@nreca.coop)

MultiSpeak project coordinator--Gary A. McNaughton, P.E. (contact information: 970.731.1508 or gmcnaughton@multispeak.org)

A

MultiSpeak Functions and Software Applications

The interfaces established in the MultiSpeak specification are defined on the basis of information flows between software *functions*. A commercial software product (*application*) may fulfill the role of one or more MultiSpeak functions. Definition by functions was done in order to (i) give flexibility to model a number of different product types and (ii) minimize duplication in the specification.

The first goal was to include in MultiSpeak the flexibility to model a number of different product types. An example that illustrates this need for flexibility is the automated meter reading (AMR) application. Some AMR systems have the capability to support two-way communications between the AMR server and a meter, and hence can query individual meters and implement service connect and disconnect (the MultiSpeak connect/disconnect function, CD) remotely. AMR systems using one-way communications technologies usually cannot support such functions. However, both systems support meter reading (the MultiSpeak meter reading function, MR) functionality. Figure A-1 shows examples of AMR systems that support different combinations of software functions. In that figure, AMR Product A supports only the MultiSpeak meter reading function, whereas AMR Product B supports the meter reading, load profile and remote connect/disconnect functions as defined in MultiSpeak. Thus, MultiSpeak was defined to enable vendors to choose which software functions they wish to support in their applications and to give the utility software user a clear and unbiased means to evaluate the capabilities of competing products.

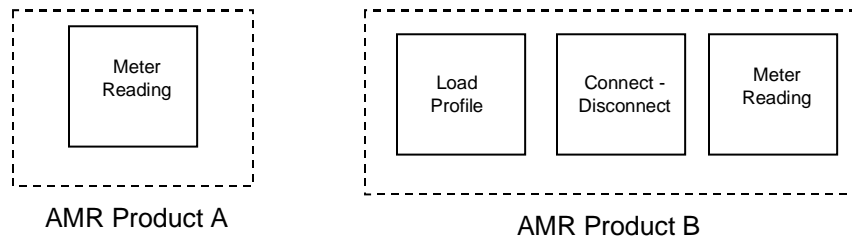


Figure A-1
Example of Applications and Supported Software Functions

A second objective was to eliminate duplication of interface definitions. An example of this situation is detection of outages. Detected outages can be logged using (i) a database application employed by customer service representatives to enter information taken from customer phone calls, (ii) an interactive voice response system that logs customer phone calls, (iii) meters reporting to an AMR

system or (iv) a system that deploys dedicated outage detection modules installed at the customer location. Regardless of the source of the outage notification, the data that need to be communicated with another software application are identical. Hence, if each of these outage detection systems had its own unique interface, redundant interface definitions would result.

Table A-1 describes the software functions currently defined by MultiSpeak. Each function is referred to by a function name; for instance, CB is the function name for the customer billing database function.

**Table A-1
MultiSpeak Functions and Function Definitions**

Function	Function Definitions
CB	Customer Billing Database. Includes a customer information database, customer billing and accounting for electrical usage.
CD	Connect/Disconnect/Power Limitation. Controls remote switches or power limiting devices installed at customer services.
CH	Call Handling. Manages incoming and outgoing calls
CRM	Customer Relationship Management. Enables tracking customer care activities for individual end-use customers.
DGV	Dynamic GIS Viewer. Accepts and displays dynamically changing data in a geographic context. It is the intention of a dynamic GIS viewer to display changes in the status of data with no intentional time delay. Examples of this function are Automatic Vehicle Location or real-time display of outages.
EA	Engineering Analysis. Is treated as a single function, although it typically includes all engineering analysis functions, such as voltage drop and fault study capabilities.
FA	Finance and Accounting. Includes the corporate accounting, accounting for time and materials, and work order accounting.
GIS	Geographic Information System. Is treated as a single function, although a GIS typically includes map editing/creation tools, printing tools and database editing/query tools. Any given vendor's GIS <i>application</i> may also include the capabilities of the dynamic GIS viewer function.
LM	Load Management. Accepts information about required power system curtailments and manages those curtailments by communicating with remote devices, such as load control switches.
LP	Load Profile. Exchanges information about how metered loads change over time by saving information about such loads periodically. The load profile function is considered to be an advanced function. Support for this function is not required for compatibility with the MultiSpeak specification. It is recommended, however, that if support for load profile is included in an AMR product, interfaces with the load profile function be compatible with the MultiSpeak definitions.
MR	Meter Reading. Collects information from remote meters, typically revenue meters, and presents it to other systems for analysis.
OA	Outage Analysis. Accepts outage information from outage detection sources. Such information is used to (i) assist a human dispatcher to determine which power system devices have likely operated to create the observed pattern of outages and (ii) facilitate outage reporting.
OD	Outage Detection. Is broadly defined for the purposes of MultiSpeak; it includes any means by which information about outages is stored on a computerized server. Information for such systems may be collected by any of the following: (i) humans taking customer calls and entering the applicable information, (ii) an IVR that collects outage information, (iii) automated outage detection devices or (iv) an automated meter reading system.
SGV	Static GIS Viewer. Accepts and displays information in a geographic context. This function is used to display information that is not rapidly changing; hence, real-time response is not necessary.
SCADA	Supervisory Control and Data Acquisition. Provides status detection, logging of analog information and control of remote power system equipment.
Staking	Automated Staking. Provides field design and cost estimation capabilities.

Table A-2 includes six common software applications and the MultiSpeak functions that might be supported by those applications.

**Table A-2
Examples of Multifunction Software Products**

Software Product	MultiSpeak Functions Potentially Incorporated in This Software Product
<i>Automated Meter Reading System (AMR)</i>	An AMR consists of at least the meter reading (MR) function, but may also include connect/disconnect/power limitation (CD), outage detection (OD) and load profile (LP) functions.
<i>Customer Information System (CIS)</i>	As applied in most cooperatives, the CIS consists of a suite of applications, typically provided and integrated by a single vendor. Usually included are the customer billing (CB) and finance and accounting (FA) functions defined above; however, a utility CIS also may include the outage detection (OD) function (via an application to enter customer calls).
<i>Geographic Information System (GIS)</i>	A GIS will include the GIS MultiSpeak function, but may also include the static GIS viewer (SGV) and/or dynamic GIS viewer (DGV) functions.
<i>GIS Viewer</i>	It is assumed that all GIS viewer applications will contain a static GIS viewer (SGV) function. They may also contain a dynamic GIS viewer (DGV) function if they are designed to reflect information that changes in response to real-time data input.
<i>Interactive Voice Response (IVR)</i>	An interactive voice response system includes, at a minimum, the call handling (CH) function, but may also include the outage detection (OD) and outage analysis (OA) functions.
<i>Outage Management System (OMS)</i>	Some outage management systems include both the outage analysis (OA) and outage detection (OD) functions, but some only supply one of these functions.

B

MultiSpeak Defined Interfaces

The MultiSpeak specification is based on a model of data ownerships and data exchanges intended to implement common business processes in small electric utilities. These relationships are outlined in table B-1. The first column of table B-1, "Data Flow Number", indicates a data flow (or interface) number. These interfaces are also illustrated in figure B-1. The second column, "Linked Functions", indicates which two functions are linked on this interface. The third column, "Data Exchanged", summarizes the types of information that are to be exchanged. The fourth column, "Direction", establishes the ownership of the data; that is, which function is the primary repository of the associated data and has the ultimate responsibility for its accuracy in the context of this defined interface. For instance, for interface 1, the owner of the customer data to be exchanged is the customer billing (CB) function and the consumer of the data is the load management (LM) function. This is implied by the designation "CB>LM" in the "Direction" column. The final column, "Communications Requirements", outlines the modes of communication that are defined for each interface. Three modes are used: batch (B), request/response (RR) and publish/subscribe (PS). These communication modes are more fully described in appendix C.

Note that each box in the process model (figure B-1) represents a software *function* (as defined in table A-1), not a software *application* or product. A given software product may encompass multiple MultiSpeak functions.

**Table B-1
Version 3.0 MultiSpeak Process Model
Data Exchanges and Communications Requirements**

Data Flow Number	Linked Functions	Data Exchanged	Direction	Communications Requirements (*1)
1	CB – LM	Customer and service information Load management devices	CB>LM LM>CB	B or (PS and RR) B or RR
2A	CB-MR	Meter readings/characteristics, service status, phasing, meter history logs Customer, service, meter and location info	MR>CB CB>MR	B or (PS and RR) B or (PS and RR)
2B	CB-CD	Connect/disconnect and power limitation commands; customer, meter, service Acknowledge connect/disconnect, power limitation commands	CB>CD CD>CB	B or (PS and RR) B or (PS and RR)
2C	LP-CB	Load profile data Meters	LP>CB CB>LP	B or (PS and RR) B or (PS and RR)
3	SCADA-LM	Load shed schedules/commands, power factor correction signal	SCADA>LM	PS
4	MR-EA	Customer metered load, measurement, meter, phase (request by EA, respond by MR) Meter connectivity and circuit elements (location of meter electrically)	MR>EA EA>MR	RR B or RR
5	OD-OA	Status change event, device information Outage status	OD>OA OA>OD	PS and RR RR
6	CH-OA	Outage information, status, restoration information, estimated time to restoration (ETOR), callbacks Is this service part of an existing outage? (CH request, OA responds) Revised callback status	OA>CH OA>CH CH>OA	PS and RR RR PS
7	GIS-OA	Connectivity file	GIS>OA	B
8	EA-SCADA	Analogs (volts, amps, MW, PF), device status (request from EA, respond by SCADA), temperature, time, SCADA point definitions Connectivity	SCADA>EA EA>SCADA	RR B or RR
9	SCADA-OA	Device status and transitions, SCADA analogs Presumed status of downstream devices	SCADA>OA OA>SCADA	PS and RR PS and RR
10	EA-OA	Connectivity file	EA>OA	B or RR
11	FA-Staking	Work orders, units and materials Revised work orders, staked assemblies, pick list (optional) and CPRs (optional)	FA>Staking Staking>FA	B or (PS and RR) B or (PS and RR)

**Table B-1
Version 3.0 MultiSpeak Process Model
Data Exchanges and Communications Requirements
(Continued)**

Data Flow Number	Linked Functions	Data Exchanged	Direction	Communications Requirements (*1)
12	GIS-SGV	Existing facilities information and background graphics, customer, service	GIS>SGV	B (required) PS (optional)
13		This interface has been intentionally omitted.		
14	CB-CH	Customer, service, network and connect/disconnect, meter	CB>CH	B or (PS and RR)
15	CB-OA	Customer connect/disconnect, power limitation command Customer meter data, customer and service information	CB>OA CB>OA	B or (PS and RR) B or (PS and RR)
16	CB-GIS	Customer, service, network, meter, transformer, pole, joint use, usage Transformer, pole, joint use, network	CB>GIS GIS>CB	B or (PS and RR) B or (PS and RR)
17		This interface has been intentionally omitted.		
18	OA-DGV	Outages to display	OA>DGV	PS and RR
19	CB-EA	Billing account load information Connectivity	CB>EA EA>CB	B or RR B or RR
20		This interface has been intentionally omitted.		
21	Staking-GIS	Existing facility information and background graphics Modified facility information and background graphics, staked work orders, units and materials	GIS>Staking Staking>GIS	B or RR B or (PS and RR)
22	EA-GIS	Connectivity and engineering database information Load flow and short circuit analysis results, connectivity and engineering database information	GIS>EA EA>GIS	B or (PS and RR) B or RR
23	SCADA-DGV	Device status, SCADA analogs and SCADA point definitions	SCADA>DGV	PS and RR
24	GIS-SCADA	Connectivity and coordinates of nodes	GIS>SCADA	B
25	CB-OD	Customer, meter and service information	CB>OD	B or (PS and RR)
26	OD-DGV	Outage detection events, outage detection devices	OD>DGV	PS and RR
27	OA-MR	Customers affected by outage, outage events and meter connectivity	OA>MR	PS and RR
28	CB-CRM	Customers, services, transformers, meters, customer calls	CB>CRM	B
29	FA-CRM	Work orders	FA>CRM	B
30	LM-GIS	Load management devices	LM>GIS	B or (PS and RR)

Note: Communications requirements defined in the specification are B = batch, PS = publish/subscribe, RR = request/response.

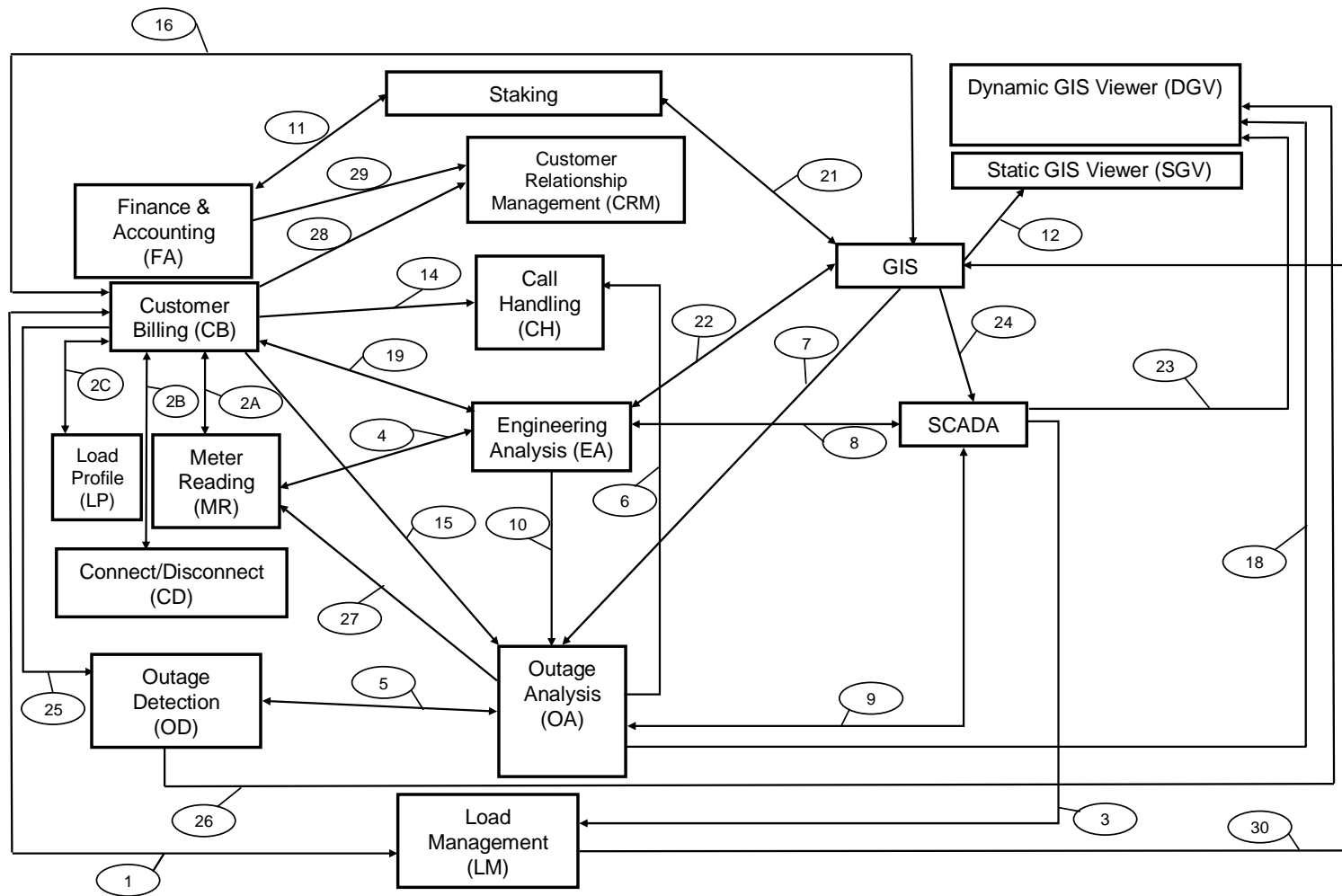


Figure B-1
MultiSpeak Version 3.0 Process Model

C

Choosing Between Communications Options

C.1 Communications Transport Options

Communications transport options are means by which information can be moved between two MultiSpeak-compatible applications. Two communications transfer options are supported in Version 3.0 of the MultiSpeak specification: (i) file-based and (ii) web services.

File-based communications. File-based transfer occurs when one software application writes data in the form of a file stored in a file space accessible to another application. The receiving software program reads the data from that file and takes the required actions based on the contents of the file. In the case of MultiSpeak messages, the file contains the data packet that needs to be sent between the two systems. File-based communications is often used to implement the *batch* message exchange pattern (see section C.2 for more information about message exchange patterns). *File-based communications is analogous to an e-mail message.* The communicating programs do not have to be talking directly at the same time, and no dedicated connection between them is required.

Web services communications. A web service is a means for a software application to exchange data with, or to access software modules on, another software product. Web services use Simple Object Access Protocol (SOAP) to create messages that carry XML-formatted data. Such messages are delivered using standard Hypertext Transfer Protocol (HTTP) over a TCP/IP network. The MultiSpeak-formatted data packet is encapsulated in a text wrapper, called a *SOAP envelope*, and sent with the same technology used to implement web servers. SOAP supports real-time data transfers. It requires both applications to be running and communicating at the same time, but it does not require a permanent connection between the two systems. *SOAP is analogous to the communication that occurs between a web browser and a web server.*

C.2 Message Exchange Patterns

MultiSpeak supports the following three different modes of communications, called *message exchange patterns*:

Batch communication implies that the function that owns the data sends information periodically in groups. In this mode, there usually is a delay between the time new information is available and the time it is sent. It should be noted that even batch messaging could appear to an end user be “real time” if the messages are kept short and are sent frequently enough. The frequency of batch messages to implement the defined interfaces is left entirely up to each vendor, but it is recommended that messages be kept small to enhance performance and not unduly burden the receiving application. An exception to this recommendation is a data *dump*, when all known information is sent to initially set up the database in the receiving application.

2) Request/response communication implies that a client (data consumer) requests that the server (data provider) take a specific action; the server responds with a message outlining the results of that action. For instance, the client could ask the server to send the current contents of a specific data item. The server would respond with the required information or an appropriate error message. Request/response messaging is often used by the client in lieu of maintaining a separate copy of the subject data.

3) Publish/subscribe communication implies that the publisher (data provider) makes available (or publishes) data stored in its system to one or more clients, which subscribe to such data. Although publish/subscribe systems are often implemented using message queue middleware that supports point-to-multipoint messaging, this is not a requirement of MultiSpeak. The cost and complexity of messaging middleware may be prohibitive in the small utility environment. Hence, MultiSpeak requires a software function that acts as a publisher to maintain a list of subscribers and send multiple point-to-point messages to implement functionality equivalent to that offered by the message queue middleware. A publishing application may accomplish the required functionality using middleware, provided the required messages are sent in accordance with the specification and the middleware is bundled with the software product when sold.

Table C-1 indicates which of the communications transport options (file-based and web services) supports the three message exchange patterns (batch, request/response and publish/subscribe).

Table C-1
Message Exchange Patterns and Communications Transport Options

Message Exchange Patterns	Communications Transfer Options	
	File-Based	Web Services
Batch	X	X
Request/Response		X
Publish/Subscribe		X

C.3 File-Based versus Real-Time Communications

In the context of the MultiSpeak specification, a data transfer is termed real time if it is completed as soon as possible after a change in data occurs; that is, with no intentional time delay. Web services are considered to be real time; hence, the request/response and publish/subscribe message exchange patterns in table C-1 can only be implemented in MultiSpeak 3 using web services. On the other hand, in file-based communications, data are typically buffered and sent in batches; in fact, they are sometimes called “batch” communications. It should be noted, however, that the same batch message exchange pattern could also be implemented using web services, where the client requests a complete set of all known information of a specific type; for instance, all the customer data. Hence, table C-1 indicates that the batch message exchange pattern can also be implemented using web services.

File-based communication is thought by some to be cumbersome or not frequent enough to meet modern data exchange requirements. However, if file-based transfers are automatically scheduled so that no human intervention is necessary, and they occur frequently enough, they are indistinguishable to the end-user from data exchanged in “real time.” File-based transfer is simple and reliable. It may serve the complete data exchange needs of some utilities, and it is likely to play a role for some interfaces in most utilities.

Having said this, there are some business processes, such as outage management, that are inconsistent with the time delays inherent in file-based communications. For MultiSpeak interfaces that support such business processes, web services are more appropriate. Figure C-1 indicates (i) which interfaces are defined in MultiSpeak 3 for both batch and real-time communications (those denoted by solid lines in figure C-1) and (ii) which will support the underlying business processes only if sent in real time (those denoted by dashed lines) and hence are defined in MultiSpeak 3 for web services transfers only.

Examples of business processes that may be adequately supported with batch communications include the following:

- Establishing a units and materials database in an automated staking system from a work order accounting system (MultiSpeak function = FA)
- Establishing an initial geographic display in a SCADA system database from a GIS
- Updating a laptop GIS viewer from a GIS server

Examples of business processes that require real-time transfers of data include the following:

- Displaying actual SCADA device status or outage status in a GIS
- Intelligent outage and outage call handling, using SCADA device status
- Maintaining up-to-date outage detection status for outage analysis
- Performing engineering studies based on actual system conditions and loads
- Querying AMR units to determine status of a customer service
- Sending real-time signals to load management switches
- Real-time crew dispatching and location

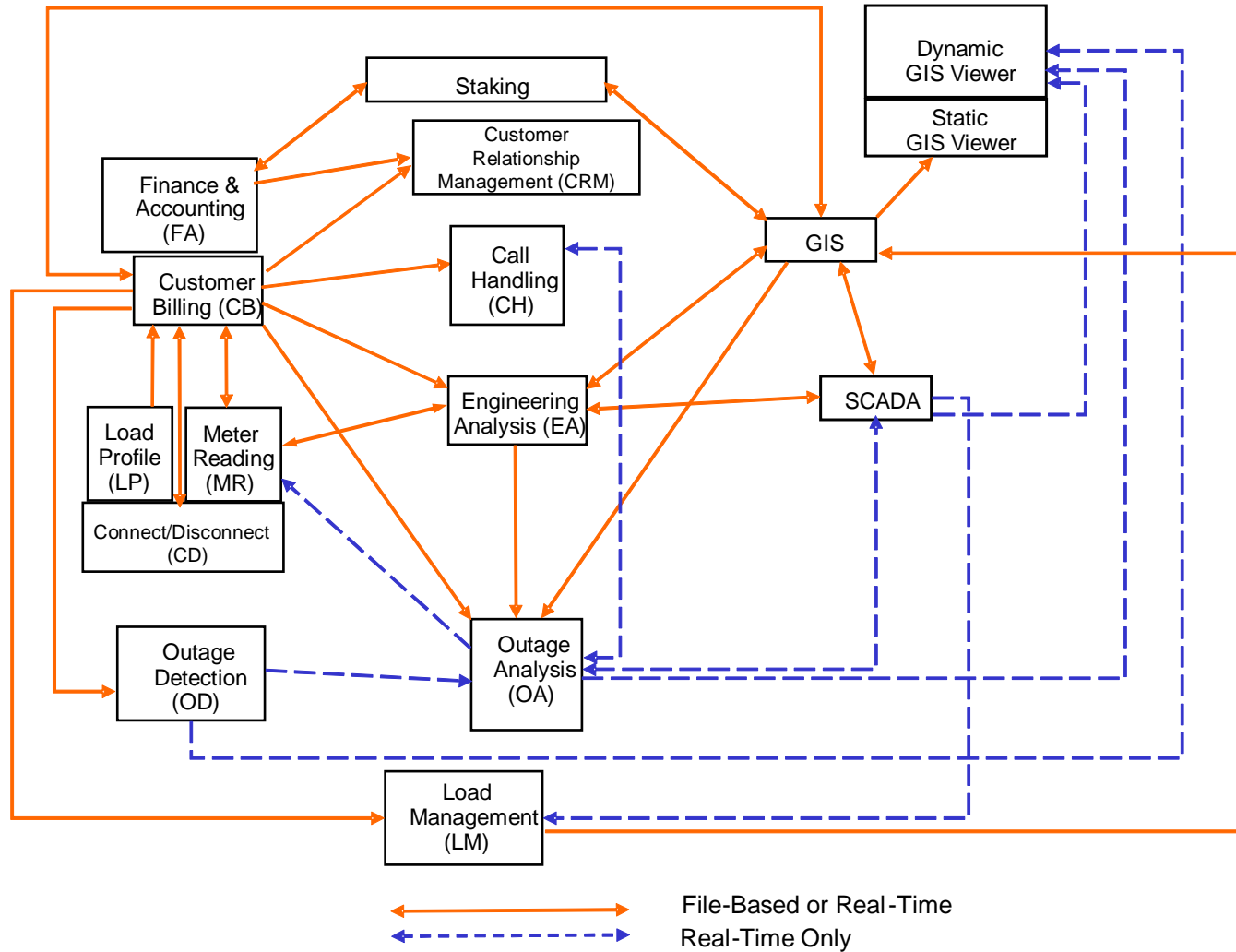


Figure C-1
File-Based versus Real-Time Communications

D

Web Services Basics

D.1 Introduction to Web Services

A web service is a standardized means for one software application to exchange data with, or invoke actions on, another application. This method of interprogram communications uses the same protocols used on the World Wide Web to access functionality (called *services*) hosted on another program; hence the name *web services*. See question #19 in section 3 of this document for an example of how a travel web site could use web services.

Figure D-1 shows the pattern of interaction in a generic web service. Application A, acting as the *service provider* (or host of a web service), first publishes to the *discovery agency* that it has a service it wishes to offer. The discovery agency (also known as the *directory*) serves the same role in web services as the yellow pages in everyday commerce. Application B (the *service requestor*) seeks out a service like the one published by Application A by querying the discovery agency, which returns information about the available service to Application B. Application B then requests that Application A provide the required service using the interface described by the discovery agent.

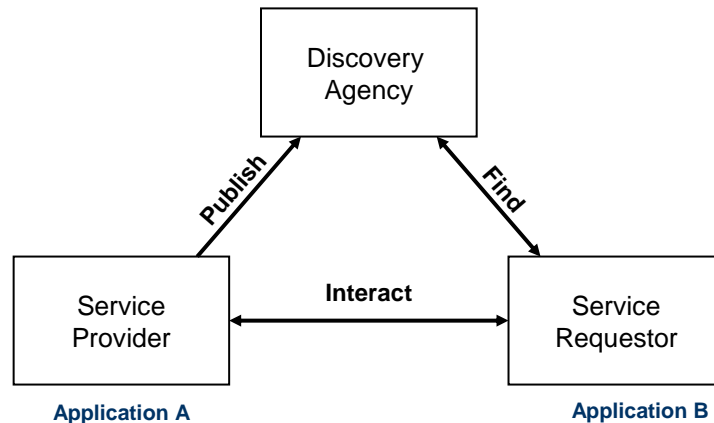


Figure D-1
Web Services Interaction Model

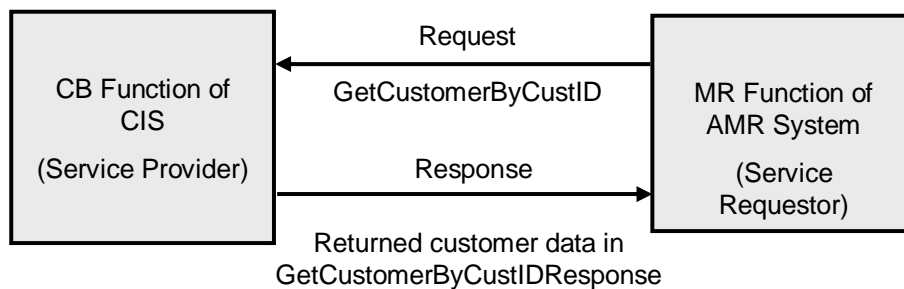
This interaction pattern is called a *dynamic* web service, since the service requestor can discover at run time the availability and nature of the desired web service. This is important if the service requestor is not familiar with the services available from the service provider, or if the nature of the interface between the two applications changes frequently.

It is also possible to implement web services using a simpler *static* interaction pattern. In this case, no discovery agency exists; hence, the publish and find messages shown in figure D-1 are not used. To implement a static web service, the business partners (Applications A and B) must have previously agreed upon the details of the web service. At present, MultiSpeak does not assume the availability of a discovery agency; therefore, it uses a static interaction pattern between the service requestor and the service provider. The examples in figures D-2 and D-3 illustrate the static interaction pattern using just a service requestor and service provider. The roles of service requestor and service provider are clarified in figures D-2 and D-3, which provide two examples of CIS-AMR integration using MultiSpeak

Example 1: CIS-AMR Data Integration Using MultiSpeak Request/Response Web Service

Business Process: The AMR system asks the CIS system for customer data about a specific customer.

In this case, the MultiSpeak meter reading (MR) function of the AMR system requests the customer billing (CB) function of the CIS to provide information about the customer. The CB function acts as the service provider and the MR function acts as the service requestor. The MR function uses the GetCustomerByCustID MultiSpeak web service, passing to the CB function the customer identifier for the customer of interest; the CB function returns the requested data. This is an example of *data integration* using web services. In this example, the AMR application can use the MultiSpeak web service to keep its copy of the customer data table in synchronism with the CIS, or could just refer to the CIS for customer information without maintaining its own copy of the data. This example, illustrated in figure D-2, uses the request/response message exchange pattern.



**Figure D-2
Example of CIS-AMR Data Integration Web Service**

web services. Note that in both examples it is assumed that the CIS and AMR have previously been configured to look for specific web services on the other software application. Hence, no discovery agency is necessary for these MultiSpeak web services.

Example 2: CIS-AMR Business Process Integration Using MultiSpeak Publish/Subscribe Web Service

Business Process: The CIS system notifies the AMR system that a specific meter is part of a planned outage.

Here the CB function of the CIS notifies (publishes to) the MR function of the AMR that a specific meter is to be part of a planned outage. The MR function returns an acknowledgment of the changed meter status. The MR function is acting as the service provider; the CB function (service requestor) calls the service on the MR function to publish the status change. In this case, the outage management business process has been modified by this web service; it is no longer necessary for a customer service representative to leave the CIS application and open the AMR application to change the meter status. This is an example of how web services can be used for *business process integration*. This web service enables employees using these two integrated products to focus on the software application that is most important to them and perform their jobs more efficiently. This example, shown in figure D-3, illustrates the publish/subscribe message exchange pattern.

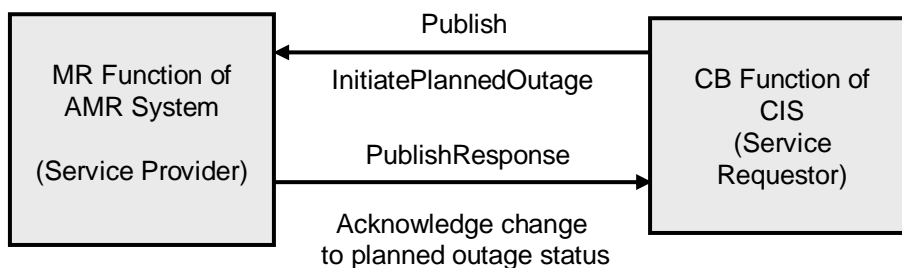


Figure D-3
Example of CIS-AMR Business Process Integration Web Service

It is not necessary for utility personnel to understand the details of how web services work if they are interested in using only web services provided by their software vendors. However, they will need to know more if they wish to write their own web services programs. You might wish to do this so that you can use integration features available from one software package but not used by another package currently in place at the utility. Another reason that you might want to

write your own web service would be to gather data from multiple web service-enabled programs and use those data to create customized management reports or management digital dashboards.

For those who wish to know more, the remainder of this appendix discusses at a high level what capabilities should ideally be available for sophisticated web services implementations. It outlines how existing standards are used to implement a theoretical web services architecture, called the web services framework, and identifies shortcomings in the state of the art of web services. Appendix E builds on that architectural discussion by identifying which standards are used to implement MultiSpeak web services. Furthermore, appendix E indicates where utilities may wish to extend MultiSpeak 3.

D.2 Creating Meaningful Web Services Messages

For two applications to communicate meaningfully using web services, they must agree upon the following:

- **Data semantics.** Semantics is a common agreement on the meaning of the data being exchanged. *In a written document, semantics may be thought of as the meaning of the words used in document.* The web services framework (discussed in section D.3) is a set of standards that defines what components are necessary to accomplish communications using web services, but nothing in the framework defines *what* is to be communicated. The *what* is semantics. MultiSpeak has established a data dictionary that defines the meaning and context of information to be exchanged in compatible interfaces. The MultiSpeak data dictionary and other agreements that are documented in the MultiSpeak specification govern the content of messages sent using compatible web services. How those messages are exchanged is the subject of the remaining items in this list. The messages may be implemented using standards that fulfill the requirements of the web services framework.
- **Data syntax.** Syntax concerns how the message content is converted into a universal, computer-readable format. *In a written document, syntax may be thought of as how the words of the message (semantics) are put into printed sentences and paragraphs.*
- **Message structure.** Once the message content is constructed using semantics and syntax, it is necessary to construct complete messages that contain information about the destination and handling instructions. *In a document to be sent by postal mail, the message content (semantics and syntax) is the letter enclosed in the envelope; the message structure is the address and mailing instructions on the envelope.*

- **Message exchange pattern.** The semantics, syntax and message structure provide the means to send a single message. How messages are built into complex information exchanges is the realm of the message exchange pattern. For instance, in a request/response message exchange pattern, two messages are linked--one requesting data and one responding with the requested information. This linked set of messages accomplishes a single step in a business process. An example of how messages are linked in a paper business process might be a utility sending out a monthly revenue bill (request) followed by the customer returning a check in payment (response). This multipart transaction accomplishes one step in the consumer billing process.

The functions described thus far in this section can be accomplished using a single simple web service. How such simple web services are linked together to create complete (multistep) business processes is the subject of *business process flow modeling*. Modeling of complex business processes can permit software applications to choose which web services to call, depending on the specific condition being handled. For instance, a customer billing application could query an AMR system to decide if energy is flowing to a service that was previously disconnected for nonpayment. If the AMR system shows that no power is flowing, no additional business process steps would be required. On the other hand, if the customer has illegally reconnected his service and power is improperly flowing to that customer, the billing system could call a web service to schedule a work order to disconnect the customer at the transformer.

D.3 Web Services Framework

This section describes the web services framework and how the concepts of the previous discussion are accomplished using that architecture. The framework contains protocols that implement data syntax, message structure and message exchange pattern, as well as some other components that are necessary to accomplish communications in a real-world information technology (IT) system. Each component of the framework will be associated with an IT standard, where such standards exist, that defines how that component may be implemented. Such standards help to ensure that software products acting as service providers and service requestors in a web service have a common means to interact.

The information technology community has adopted the web services framework (shown in figure D-4) as a means to discuss what IT services are required to accomplish a specific web service. The framework is a theoretical IT architecture, but gives a foundation for understanding how the many web services standards fit together.

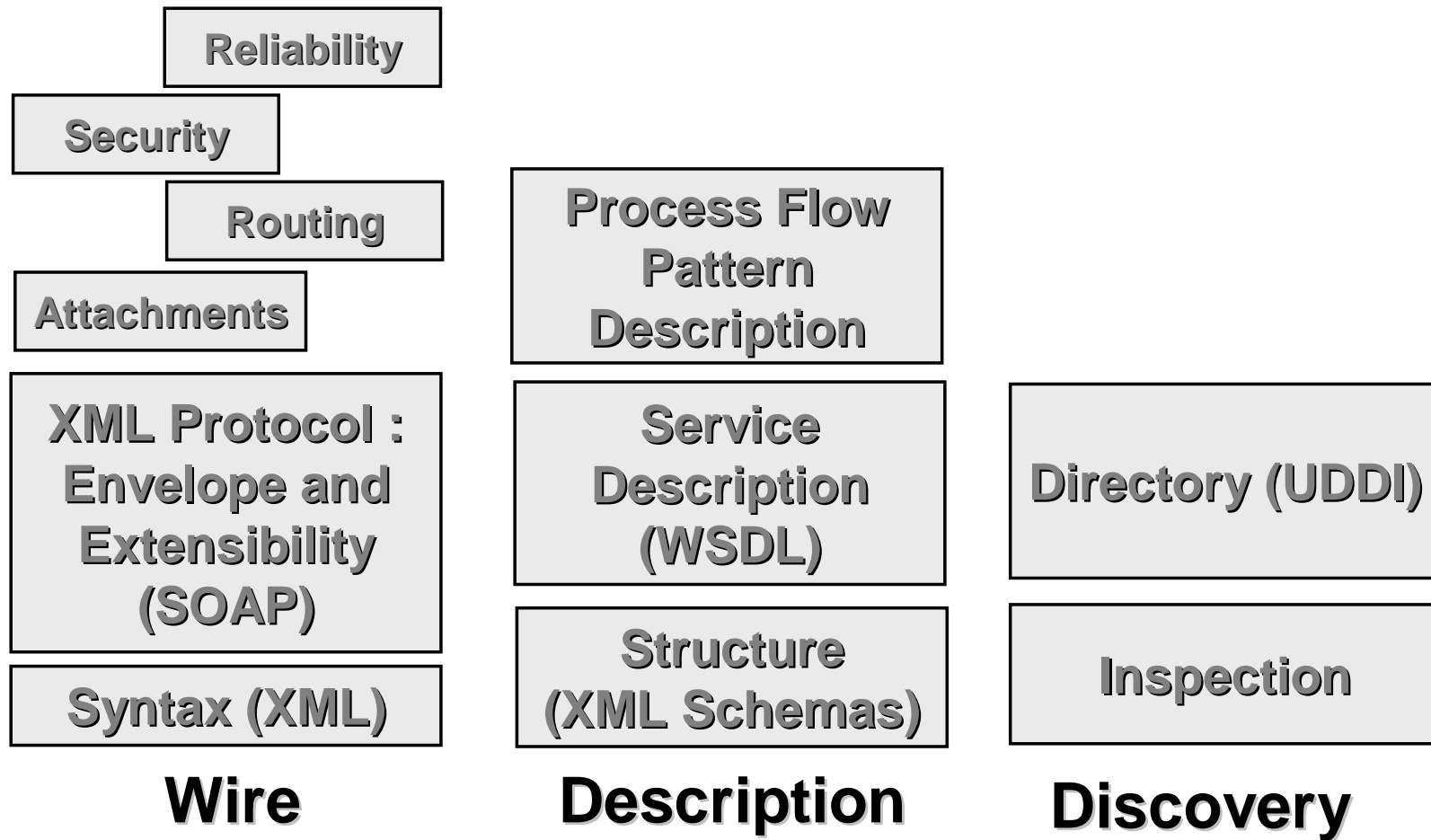


Figure D-4
Web Services Framework

Figure D-4 shows that the components in the framework are of three possible types: (i) *wire services* shown in the service stack on the left side of the figure, (ii) *description services* in the middle stack and (iii) *discovery services* in the stack on the right side of the figure. Wire services govern how data are represented and delivered across the network. Description services concern how data, web services and business processes are documented. Discovery services deal with how service requestors and providers exchange information about available web services. Remember that an agreement on data semantics is outside the scope of the web services framework, so it does not show up in either figure D-4 or table D-1.

Table D-1 describes the service components that make up the web services framework and common protocols used to implement those components. The components used to implement the basics of simple dynamic web services (i.e., XML, XML Schema, SOAP, WSDL and UDDI) are well standardized, and interoperable implementations exist for most modern programming languages, operating systems and software development toolkits.

However, it should be noted that several of the web services protocols not included in the previous list are still in draft form. In other cases, multiple candidate standards exist and the overall IT community has not reached a consensus on how to implement such components. Areas where significant uncertainty existed at the time this document was written include (i) reliability, (ii) security, (iii) routing, (iv) attachments and (v) process flow pattern description. See table D-1 for a discussion of these components.

**Table D-1
Web Services Components**

Component	Standard	Purpose
Wire Components		
Syntax	XML	Extensible Markup Language (XML). Describes how data are serialized on the wire using self-descriptive, human-readable "tags." This is the data syntax component.
XML Messaging	SOAP	Simple Object Access Protocol (SOAP). Describes how messages are constructed using XML-formatted text. Messages include addressing, header information that specifies how the message is to be used, and a data payload, called a SOAP Body. This is the message structure component.
Attachments	WS-Attachments and possibly others	Web Services-Attachments (WS-Attachments). Describes how binary objects (such as images), which cannot be directly encoded in XML, are included in SOAP messages.
Routing	WS-Routing or WS-Addressing	Web Service Routing (WS-Routing) or Web Services Addressing (WS-Addressing). Describes how to route messages using a complex, multiple-hop path on a TCP/IP network.
Security	SSL and possibly many others	Secure Sockets Layer (SSL). Describes how messages and/or message components may be encrypted and carry security information, such as authentication tokens.
Reliability	WS-Reliable Messaging and potentially others	Web Services Reliable Messaging (WS-Reliable Messaging). Concerned with ensuring that any message is delivered once and only once and that messages are delivered in the correct order so that complex transactions are committed properly. Message reliability can be provided (i) by middleware, such as a message queue or (ii) by adding protocol information in the messages themselves to ensure that message transactions do not occur until all the information is in place.
Description Components		
Structure	XML Schema	XML Schema. Standardizes how data dictionaries (message semantics) are documented.
Service Description	WSDL	Web Services Description Language (WSDL). WSDL files are XML-formatted files that build individual SOAP messages into message exchange patterns that implement a single business process step. WSDL also encapsulates and documents the XML schema and SOAP components in a computer-readable form.
Process Flow Pattern Description	WS-BPEL and possibly others	Web Services Business Process Execution Language (WS-BPEL). Dynamically links business process steps into an overall business process, based on varying needs discovered at run time.
Discovery Components		
Directory	UDDI	Universal Description, Discovery and Integration (UDDI). Designed to facilitate the discovery of web services across distributed, unrelated networks, such as the Internet.
Inspection	Included in UDDI	Facilitates discovery of information about how a web service is to be called or about specific data components exchanged using a web service.

Note: References for these standards may be found in appendix H.

E

How MultiSpeak Implements Web Services

The use of web services is a very powerful technique that has the potential to revolutionize IT systems and streamline business processes at utilities. The greater IT community is still learning how to maximize the benefits from web service implementations; doing so requires implementing complex, multiparty, dynamic web services. Fortunately, you can just dip your toe into the water and gain some of the benefits. You might take such an approach if you use MultiSpeak web services provided by your software vendors without worrying about what else might be done with the tools provided.

Wading into some of the complexity of web services will enable you to gain more benefits. This might require you to write some of your own web service applications to do things not provided by your vendors. However, you may need to take a more comprehensive and sophisticated approach if you wish to achieve the full potential of this new technology. In doing so, you may feel at times that you have gone in over your head. This appendix is provided as a guide for those who wish to understand MultiSpeak web services in enough detail to explore on their own. It discusses how much of the web services universe is supported by MultiSpeak 3 and outlines where you might need to provide components that MultiSpeak does not yet include.

E.1 What MultiSpeak Provides

One of the key challenges in implementing standardized application integration is ensuring that business partners apply the same set of standards and that they implement those standards in the same way. The Web Services Interoperability Organization (WS-I) has been formed to assist in these challenges. WS-I recommends how web services should be applied to maximize the chances that business partners can independently develop web services that will interoperate.

The MultiSpeak web services reference architecture is based on the WS-I Basic Web Services Profile 1.1 (Profile). The Profile is a set of nonproprietary web service specifications, along with clarifications, refinements, interpretations and amplifications of those standards. It is designed to promote interoperation of simple web services. More information about WS-I and the Profile is available at www.ws-i.org. Table E-1 outlines the protocols used in the Profile.

Table E-1
WS-I Basic Web Services Profile 1.1

Function	Protocol
Schema Definition	XML Schema 1.0
XML Messaging	SOAP 1.1
Service Description	WSDL 1.1
Service Publication and Discovery (1)	UDDI 2.0
Transport	HTTP 1.0 or 1.1 (HTTP 1.1 is preferred)
Security (2)	Secure Sockets Layer V3.0 (SSL) or Transport Layer Security V1.0 (TLS)

Notes:

1) MultiSpeak does not specifically assume that a discovery agency is present; hence, the use of UDDI is optional. If service publication and discovery is to be applied, it is recommended that a UDDI-compatible discovery agency be used.

2) In accordance with the tenets of the Profile, where it is desirable to secure web services transported over HTTP, either TLS or SSL should be used. The MultiSpeak Initiative has adopted the use of SSL. Security is the subject of a separate detailed utility guideline, *Security Considerations in Implementing MultiSpeak®-Compliant Applications*, which more completely addresses securing MultiSpeak messages, and is available from the MultiSpeak web site (www.MultiSpeak.org).

The WS-I Profile specifies the web services framework components (see figure D-4) that are recommended to implement basic web services, such as those implemented in MultiSpeak 3. The web services framework outlines at a high level what is required to communicate between applications, but it does not indicate *what* is to be communicated.

As described in greater detail in appendix D, four things are needed to structure a conversation: (i) *semantics*, an agreement on what the underlying words mean in the context of this communication, (ii) *syntax*, how the words are put into meaningful sentences, (iii) *message structure*, a description of how sentences are formed into single messages and (iv) *message exchange patterns*, an agreement on how messages are linked together.

A means is also needed to get the sentences between the parties to complete the conversation--in computer terms, a transport. An agreement on semantics, also called a *semantic layer*, is required. In MultiSpeak, the semantic layer is provided by the MultiSpeak data dictionary and the underlying agreements regarding utility business processes that have been incorporated into the MultiSpeak interface definitions. The syntax in web services is taken care of by XML encoding of the underlying message; message structure is provided by SOAP; message exchange patterns are described by WSDL; and network transport is handled by the use of HTTP and TCP/IP.

Figure E-1 extends the web services framework shown in figure D-4 by the addition of the MultiSpeak semantic layer and illustrates which of the framework components are applied in MultiSpeak 3. Modules shown in figure E-1 as boxes with medium grey shading and black text are web services framework components that are included in MultiSpeak 3. These modules--Syntax (XML), Structure (XML Schemas), XML Protocol (SOAP), Service Description (WSDL) and Inspection--form the core of simple web services. The modules shown as light grey shaded boxes with black text are web services framework components that are optional and/or only partially implemented in MultiSpeak--Reliability, Security and Directory (UDDI). Unshaded boxes with grey text are web services framework components that are not implemented in MultiSpeak 3 in any way--Routing, Attachments and Process Flow Pattern Description.

Table E-2 gives additional details about how the web services framework components are implemented in MultiSpeak 3 web services. Numbers and descriptions in the boxes in figure E-1 correspond to the numbers in the "Service" column and the descriptions in the "Component" column of table E-2. The text in the "Role" column describes what role this component plays in completing a web services communication. The protocols (or possible protocol choices) that are commonly used to implement this component are listed in the "Standard Protocol Used" column.

The column "Supported in MultiSpeak Version 3.0" indicates to what degree the standard is supported in MultiSpeak 3. A "Yes" in this column indicates that the standard is completely implemented and the guidelines of the Profile have been adopted. A "No" in this column means that no consideration for this functionality is included in MultiSpeak 3. Components marked as "Optional" may be included as part of a MultiSpeak-compatible installation, but are not required to make MultiSpeak web services work. The word "Partial" implies that some support for this functionality is included in the MultiSpeak message definitions (documented in the WSDL files) even if the specification mentioned is *not* included in MultiSpeak. Two components are marked "Partial"--Inspection and Reliability. In both cases, message definitions provide some functionality, but the standard protocol referred to in table E-2 is not supported in MultiSpeak. For more information about how these components are supported in MultiSpeak 3, see the MultiSpeak specification document, which is available from www.MultiSpeak.org, or contact the MultiSpeak project coordinator (see section 6 for contact information).

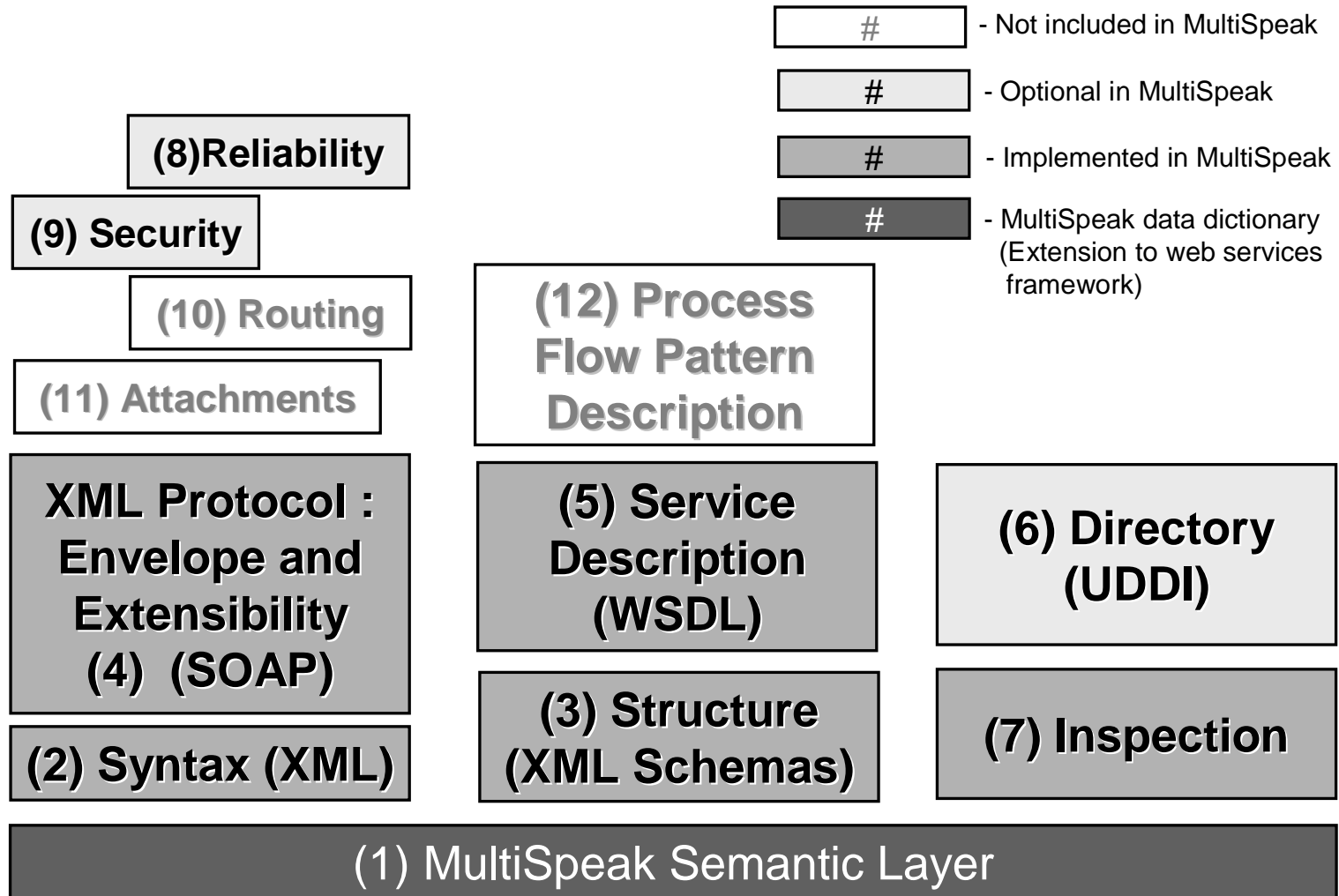


Figure E-1
MultiSpeak Implementation of the Web Services Framework

**Table E-2
Web Services Components**

Service	Component	Role	Standard Protocol Used	Supported In MultiSpeak Version 3.0	Comments
1	MultiSpeak Semantic Layer	Semantics		Yes	Provides a common understanding of the meaning and context of data objects in MultiSpeak interfaces.
2	Syntax	Syntax	XML	Yes	Provides a standard means to serialize data.
3	Structure	Documents semantics	XML Schema	Yes	Describes and documents the semantics inherent in MultiSpeak interfaces.
4	XML Messaging	Establishes message structure	SOAP	Yes	Builds messages, given the XML-formatted data payload. Includes message addressing and necessary header information.
5	Service Description	Documents web service agreements	WSDL	Yes	Compiles separate messages created by SOAP into multiple-message web service interactions that accomplish a single business process. It also describes, encapsulates and documents the XML schema and SOAP requirements.
6	Directory	Discovery agency	UDDI	Optional	Provides a standard way for service providers and service requestors to exchange information about available services.
7	Inspection	Discovery	UDDI or WSDL	Optional / Partial	Describes data and/or how services are called. Can be part of UDDI or can be included in message definition.
8	Reliability	Reliable messaging	Message queue or WS-Reliable Messaging	Optional / Partial	Ensures that messages are delivered exactly once or provides means to recover from communications failure. Currently done in messages in MultiSpeak. A common means to provide reliable messaging is the use of a message queue. In the future, this may be done with a new protocol, WS-Reliable Messaging (currently in draft form).
9	Security	Secures communications	Secure Sockets Layer (SSL) or others	Optional	Provides encryption and authentication services to secure messages. There is more development in this area than in any other. Nearly 50 candidate web services security standards are being considered with any complete solution of securing complex, multihop web services requiring a profile of multiple standards. Until a consensus is reached, MultiSpeak recommends the use of SSL for securing every network hop individually.
10	Routing	Complex routing	WS-Routing or WS-Addressing	No	Specifies routing of messages in complex, multihop web services. WS-Routing and WS-Addressing are draft standards sponsored by Microsoft, IBM and others.
11	Attachments	Attaches binary data to messages	WS-Attachments	No	Used to attach drawings or signatures to SOAP messages. Not used in Version 3.0, may be in future versions of MultiSpeak.
12	Process Flow Pattern Description	Describes complex business processes	WS-BPEL or others	No	Compiles WSDL steps into complete business processes and facilitates modification of business processes at run time. Implemented using Web Services Business Process Execution Language (WS-BPEL) (currently in draft form).

The important thing to remember about the web services framework (appendix D) and the MultiSpeak implementation of that architecture (figure E-1 and table E-2) is that MultiSpeak 3 is well positioned to handle data integration and simple business process integration for most utilities without adding the complexity of the components marked “No,” “Optional” or “Partial” in table E-2. With the exception of data security, you are unlikely to need to add components to the MultiSpeak web services architecture in your utility in the near future. The next section discusses what you can do when you *do* need more than MultiSpeak provides.

E.2 When You Need More than MultiSpeak Provides

Although MultiSpeak includes the capabilities to handle the data and business process integration needs of the typical small utility, under some unusual conditions you may wish to extend MultiSpeak. This section discusses several such situations and how you could accomplish such extensions.

One of the most likely extensions to MultiSpeak concerns provisions for network security in addition to those already included in MultiSpeak. Security of MultiSpeak-compatible web services has been specifically addressed using SSL and is described in a separate document entitled *Security Considerations in Implementing MultiSpeak®-Compliant Applications*, available from the MultiSpeak web site (www.MultiSpeak.org). SSL is a complete and adequate security solution for all simple web services that use only a single network hop (such as those implemented in MultiSpeak). The only time you might want to use a more involved approach to web services security is if you are implementing multiparty web services that involve complex routing and multiple network hops. It is highly unlikely that a utility will use such services in the near future. If you do wish to consider such web services, and security is critical to them, you will need to spend a considerable effort researching candidate security solutions. Start your research with WS-Security, a standard offered by the Organization for the Advancement of Structured Information Standards (OASIS). More information about WS-Security and OASIS is available from www.oasis-open.org.

Other than security, you need to add something to the mix only if you run up against a limitation in the MultiSpeak reference architecture, which is discussed in this appendix. Examples of when this might occur include the following:

- ***You are providing web services to be called by vendors or service providers and these parties need to dynamically discover the nature of the available services.*** In this case you might consider adding the Directory and Inspection components in table E-2 using the UDDI protocol. See appendix H for a reference to this standard.

- ***You are implementing a software application that requires guaranteed delivery of messages (message reliability), or you wish to implement an application that requires high-volume, sophisticated publish/subscribe capabilities.*** In this case, you might wish to install a message queue or investigate adding features of the WS-Reliable Messaging protocol to your existing web services. WS-Reliable Messaging is an OASIS standard; more information is available from www.oasis-open.org.
- ***You are implementing complex web services that require multihop network routing or the use of third-party intermediary web services.*** In this case, you might wish to consider implementing WS-Routing or WS-Addressing. WS-Routing is a relatively simple standard that was proposed by Microsoft; WS-Addressing is a more complex approach sponsored by Microsoft, IBM, Sun Microsystems and others. More information about either of these routing standards may be found at www.msdn.microsoft.com.
- ***You are implementing an application that needs to attach large binary files, such as digital photographs or secured images of signatures that were collected digitally.*** Large binary files are not encoded in XML; hence, they are not handled in native SOAP messages. A means must be provided to attach such files to messages sent using SOAP. Several approaches to solving this problem are possible. The most widely accepted solution at the time of this writing is WS-Attachments, an Internet Engineering Task Force draft sponsored by Microsoft and IBM. More information may be found at www.msdn.microsoft.com. Although attachments are not handled in MultiSpeak 3, the MultiSpeak Initiative is considering adding provisions for attachments in a future version of the specification.
- ***You are trying to implement a complex, dynamic business process that requires conditional branching to different business process steps based on data known only at run time.*** This is a truly advanced application of web services, and is on the leading edge of what can be accomplished with today's web services technology. Few successful implementation examples currently exist. Those considering business process flow modeling should begin their research with either Business Process Execution Language for Web Services (BPEL4WS) or Web Services Business Process Execution Language (WS-BPEL). BPEL4WS is a standards proposal made by IBM and others, which subsequently formed the basis for the WS-BPEL standard, currently in draft form at OASIS. More information is available at www.oasis-open.org.

F

How to Understand a MultiSpeak Interoperability Assertion

Starting with Version 3.0, MultiSpeak began testing two products together to document the functionality of the shared interface between the applications. This is called an interoperability test (see question #24 in section 4). The results of an interoperability test are documented in the form of an interoperability assertions document. The vendors that provide the two products jointly develop such a document, which describes the capabilities of the MultiSpeak interface(s) between their products. The claims made in the assertions document are then verified by an independent third-party testing laboratory hired by NRECA.

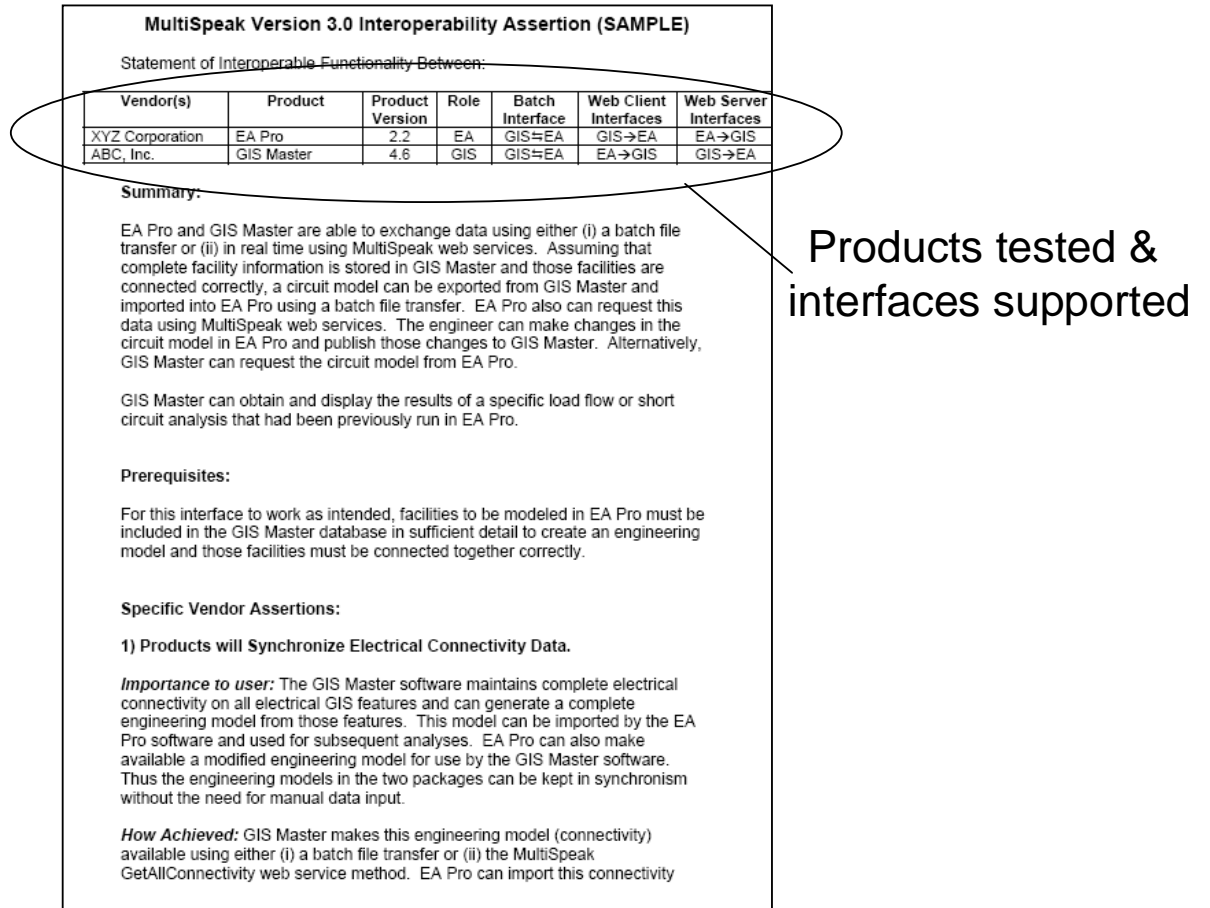
The primary purposes of an interoperability assertion are to (i) document the results of an interoperability test and (ii) educate potential software users about what functionality is supported by the joint interface. The document is designed to be understandable to users of utility software products. A sample interoperability assertion is included in appendix G. Appendix G points out features of that sample document and illustrates how a utility staff member could use the document to understand what functionality the software interface may be expected to have.

The document has the following important components:

- The products and interfaces table
- Description of interfaces tested
- The prerequisites section
- The assertions section
- Tables of data exchange specifics
- Table of batch data objects exchanged
- Tables of supported web service methods
- The certification section

Each of these components is illustrated with one or more figures that show where in the assertions document the information discussed may be found. A sample assertions document is included in appendix G.

The products and interfaces table. This table outlines (i) which products have been tested, (ii) the MultiSpeak functions that each product serves in the interface (for more information about MultiSpeak functions, see appendix A) and (iii) the batch and/or web services interfaces included in this test. Note that in the sample given, both batch and web services two-way capabilities were documented on the GIS-EA interface. This information is highlighted in figure F-1.



**Figure F-1
Products and Interfaces Table**

Description of interfaces tested. This section of the document gives an overview of the interface and describes what utility business processes can be achieved using the tested functionality. Figure F-2 shows where the summary block is on the assertions document.

MultiSpeak Version 3.0 Interoperability Assertion (SAMPLE)

Statement of Interoperable Functionality Between:

Vendor(s)	Product	Product Version	Role	Batch Interface	Web Client Interfaces	Web Server Interfaces
XYZ Corporation	EA Pro	2.2	EA	GIS→EA	GIS→EA	EA→GIS
ABC, Inc.	GIS Master	4.6	GIS	EA→GIS	EA→GIS	EA→GIS

Summary:

EA Pro and GIS Master are able to exchange data using either (i) a batch file transfer or (ii) in real time using MultiSpeak web services. Assuming that complete facility information is stored in GIS Master and those facilities are connected correctly, a circuit model can be exported from GIS Master and imported into EA Pro using a batch file transfer. EA Pro also can request this data using MultiSpeak web services. The engineer can make changes in the circuit model in EA Pro and publish those changes to GIS Master. Alternatively, GIS Master can request the circuit model from EA Pro.

GIS Master can obtain and display the results of a specific load flow or short circuit analysis that had been previously run in EA Pro.

Prerequisites:

For this interface to work as intended, facilities to be modeled in EA Pro must be included in the GIS Master database in sufficient detail to create an engineering model and those facilities must be connected together correctly.

Specific Vendor Assertions:

1) Products will Synchronize Electrical Connectivity Data.

Importance to user: The GIS Master software maintains complete electrical connectivity on all electrical GIS features and can generate a complete engineering model from those features. This model can be imported by the EA Pro software and used for subsequent analyses. EA Pro can also make available a modified engineering model for use by the GIS Master software. Thus the engineering models in the two packages can be kept in synchronism without the need for manual data input.

How Achieved: GIS Master makes this engineering model (connectivity) available using either (i) a batch file transfer or (ii) the MultiSpeak GetAllConnectivity web service method. EA Pro can import this connectivity

Description of what this product interface does

**Figure F-2
Description of Interfaces Tested**

The prerequisites section. This section outlines any prerequisites that must be in place before the tested interface will accomplish what is described in the remainder of the assertions document. The prerequisites block is highlighted in figure F-3.

MultiSpeak Version 3.0 Interoperability Assertion (SAMPLE)

Statement of Interoperable Functionality Between:

Vendor(s)	Product	Product Version	Role	Batch Interface	Web Client Interfaces	Web Server Interfaces
XYZ Corporation	EA Pro	2.2	EA	GIS=EA	GIS→EA	EA→GIS
ABC, Inc.	GIS Master	4.6	GIS	GIS=EA	EA→GIS	GIS→EA

Summary:

EA Pro and GIS Master are able to exchange data using either (i) a batch file transfer or (ii) in real time using MultiSpeak web services. Assuming that complete facility information is stored in GIS Master and those facilities are connected correctly, a circuit model can be exported from GIS Master and imported into EA Pro using a batch file transfer. EA Pro also can request this data using MultiSpeak web services. The engineer can make changes in the circuit model in EA Pro and publish those changes to GIS Master. Alternatively, GIS Master can request the circuit model from EA Pro.

GIS Master can obtain and display the results of a specific load flow or short circuit analysis that had been previously run in EA Pro.

Prerequisites:

For this interface to work as intended, facilities to be modeled in EA Pro must be included in the GIS Master database in sufficient detail to create an engineering model and those facilities must be connected together correctly.

Specific Vendor Assertions:

1) Products will Synchronize Electrical Connectivity Data.

Importance to user: The GIS Master software maintains complete electrical connectivity on all electrical GIS features and can generate a complete engineering model from those features. This model can be imported by the EA Pro software and used for subsequent analyses. EA Pro can also make available a modified engineering model for use by the GIS Master software. Thus the engineering models in the two packages can be kept in synchronism without the need for manual data input.

How Achieved: GIS Master makes this engineering model (connectivity) available using either (i) a batch file transfer or (ii) the MultiSpeak GetAllConnectivity web service method. EA Pro can import this connectivity

Prerequisites to achieve this functionality

**Figure F-3
Prerequisites Section**

The assertions section. This section lists specific claims that the vendors have made about the joint functionality supported by the MultiSpeak interface. Three things are included for each claim: (i) the claim itself, (ii) a description of why the capability is important to the utility software user and (iii) a description of how the functionality described is achieved in this interface. For instance, in the sample document the first claim is “Products Will Synchronize Electrical Connectivity Data.” Figure F-4 shows where the specific vendor assertions may be found on the assertions document.

MultiSpeak Version 3.0 Interoperability Assertion (SAMPLE)

Statement of Interoperable Functionality Between:

Vendor(s)	Product	Product Version	Role	Batch Interface	Web Client Interfaces	Web Server Interfaces
XYZ Corporation	EA Pro	2.2	EA	GIS=EA	GIS→EA	EA→GIS
ABC, Inc.	GIS Master	4.6	GIS	GIS=EA	EA→GIS	GIS→EA

Summary:

EA Pro and GIS Master are able to exchange data using either (i) a batch file transfer or (ii) in real time using MultiSpeak web services. Assuming that complete facility information is stored in GIS Master and those facilities are connected correctly, a circuit model can be exported from GIS Master and imported into EA Pro using a batch file transfer. EA Pro also can request this data using MultiSpeak web services. The engineer can make changes in the circuit model in EA Pro and publish those changes to GIS Master. Alternatively, GIS Master can request the circuit model from EA Pro.

GIS Master can obtain and display the results of a specific load flow or short circuit analysis that had been previously run in EA Pro.

Prerequisites:

For this interface to work as intended, facilities to be modeled in EA Pro must be included in the GIS Master database in sufficient detail to create an engineering model and those facilities must be connected together correctly.

Specific Vendor Assertions:

1) Products will Synchronize Electrical Connectivity Data.

Importance to user: The GIS Master software maintains complete electrical connectivity on all electrical GIS features and can generate a complete engineering model from those features. This model can be imported by the EA Pro software and used for subsequent analyses. EA Pro can also make available a modified engineering model for use by the GIS Master software. Thus the engineering models in the two packages can be kept in synchronism without the need for manual data input.

How Achieved: GIS Master makes this engineering model (connectivity) available using either (i) a batch file transfer or (ii) the MultiSpeak GetAllConnectivity web service method. EA Pro can import this connectivity

Description of
specific assertion

**Figure F-4
Assertions Section**

Tables of data exchange specifics. Following the vendor assertions is a set of tables that gives specifics about what data exchange functionality is defined by the MultiSpeak specification and which of those capabilities is supported by the tested interface. If the interface includes batch capabilities, there are tables that indicate which data objects are exchanged in the batch file transfers. If the interface includes web services capabilities, there are tables that list which of the web service methods are supported by this interface. Each of the tables (whether showing the data objects exchanged for a batch interface or the web services methods supported for a real-time interface) begins with a heading block that lists the MultiSpeak defined interface number (for example, interface #22), the MultiSpeak functions included (in this case, EA and GIS), the data flow direction (EA >GIS or GIS>EA) and whether the interface is batch or real time. The location of the header block is highlighted in figure F-5.

Interface number, data flow direction & communications option

Products: GIS Master and EA Pro

Summary of Interoperability Test Results
Interface #22
EA >GIS (Real Time)

Table 1
Recommended MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
CircuitElementChangedNotification	Used to publish changes in circuit elements to the GIS.	X	X	X
GetAllLoadFlowResults	Requests all available sets of load flow results.	X	-	-
GetAllShortCircuitAnalysisResults	Requests all available sets of short circuit analysis results.	-	-	-
GetLoadFlowResultsByConnectID	Requests a specific set of load flow results.	X	X	X
GetShortCircuitAnalysisResultsByObjectID	Requests a specific set of short circuit analysis results.	X	X	X
GetMethods	Requests a list of web service methods supported by the engineering analysis program.	X	X	X
PingURL	Queries status of the engineering analysis program.	X	X	X

Table 2
Optional MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
GetAllConnectivity	Requests a complete engineering model.	X	X	X
GetChildConnectivity	Requests all electrical features that are immediately downstream of a specified element.	X	X	X
GetDownlineConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetModifiedConnectivity	Requests all circuit elements that have changed since a specific interconnection session.	-	-	-
GetParentConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model.	X	X	X
GetUpstreamConnectivity	Requests all elements that are upstream of a specified element.	X	X	X
GetDomainMembers	Requests a list of items in a domain.	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database.	-	-	-

1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
 2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
 3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

**Figure F-5
 Header Block**

Table of batch data objects exchanged. The first type of table that might be included in the set of data exchange specifics is one showing batch data object exchanged (see figure F-6). If a batch interface was tested, then such a table will be included in the assertions document. It will (i) list MultiSpeak data objects included in the interface as defined in the specification, (ii) describe why exchanging each data object is important to the utility, (iii) list which data objects can be exported by the server, (iv) list which data objects can be imported by the client application and (v) document which data exchanges were verified during this interoperability test. Tables 1 and 4 in the sample assertions document in appendix G are tables of this type. Table 1 lists data items sent from the engineering analysis application to the GIS; table 4 lists data items sent from the GIS to engineering analysis.

Products: GIS Master and EA Pro Summary of Interoperability Test Results
Interface #22
EA→GIS (Batch)

Table 4
MultiSpeak Data Objects Exchanged (Recommended)

Object Name	Importance to User	Supported by Server (EA)	Supported by Client (GIS)	Verified Interoperable
Load flow results	The results from load flow analyses made in EA Pro can be displayed geographically in GIS Master.	X	-	-
Short circuit analysis results	The results from short circuit analyses made in EA Pro can be displayed geographically in GIS Master.	X	-	-
Connectivity, including:	The set of data objects that collectively define an engineering model for a system is called connectivity. A connectivity model includes all of the objects shown as being indented in the first column. Support for all of the objects, except fake node section is required for a complete engineering model.	X	X	X
Overhead primary line		X	X	X
Overhead secondary line		X	X	X
Underground primary line		X	X	X
Underground secondary line		X	X	X
Capacitor bank		X	X	X
Transformer bank		X	X	X
Regulator bank		X	X	X
Overcurrent device bank		X	X	X
Switch bank		X	X	X
Service location		X	X	X
Substation		X	X	X
Motor		X	X	X
Generator		X	X	X
Fake node section		-	-	-

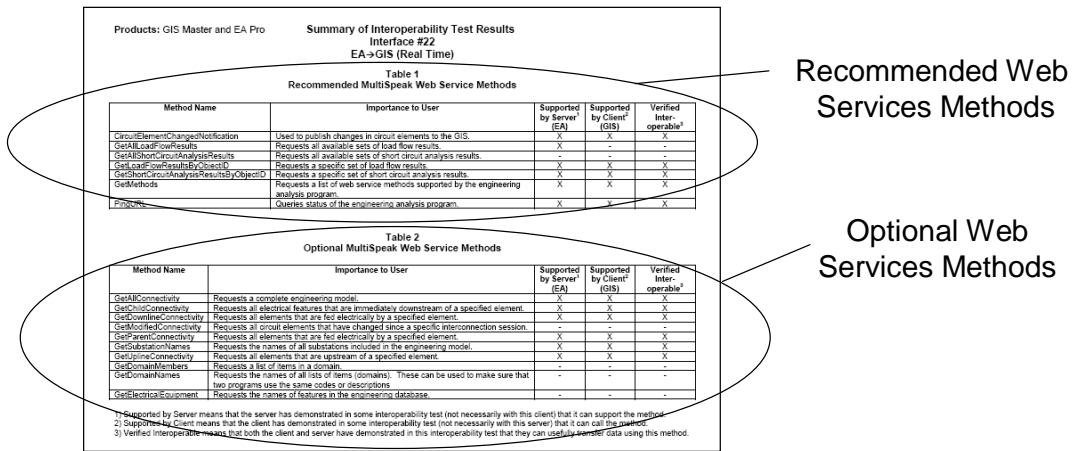
Notes:
1) EA Pro uses a line section-oriented engineering model. This interface uses line sections (rather than nodes) to exchange the engineering model.
2) EA Pro and GIS Master use a tightly coupled interface. This means that the name of a line section can be changed in either the engineering analysis or the GIS application without breaking the link between the two programs.

Batch Data Objects Exchanged

Figure F-6
Table of Batch Data Objects Exchanged

Tables of supported web service methods. If a real-time (web services) interface is included in the tested functionality, then tables will be included that describe those capabilities, similar to the batch data exchange tables described in the previous paragraph. Tables 2, 3, 5 and 6 in the sample assertions document in appendix G show (i) the web service methods included in the interface as defined in the MultiSpeak specification, (ii) a description of the importance of the method for the utility software user, (iii) which of the methods are supported by the server and client and (iv) which of the joint web services were verified as interoperable during the test.

Following the header block in the web service methods listings are two tables, one for web service methods included as recommended for this interface in the MultiSpeak 3 specification, and one for methods that are defined as being optional for this interface. Figure F-7 shows the location of these two tables.



**Figure F-7
Description Blocks on Supported Web Service Methods Table**

Tables of supported web service methods (continued). Figure F-8 focuses on the first column of the web services table, which shows where the web service methods are listed for the recommended and optional web service methods, respectively. Each web service method is described in the center section of the supported web service methods table (see figure F-9).

Web service methods defined for this interface

Products: GIS Master and EA Pro

Summary of Interoperability Test Results
Interface #22
EA→GIS (Real Time)

Table 1
Recommended MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
CircuitElementChangedNotification	Used to publish changes in circuit elements to the GIS.	X	X	X
GetAllLoadFlowResults	Requests all available sets of load flow results.	X	-	-
GetAllShortCircuitAnalysisResults	Requests all available sets of short circuit analysis results.	-	-	-
GetLoadFlowResultsByObjectID	Requests a specific set of load flow results.	X	X	X
GetShortCircuitAnalysisResultsByObjectID	Requests a specific set of short circuit analysis results.	X	X	X
GetMethods	Requests a list of web service methods supported by the engineering analysis program.	X	X	X
PingURL	Queries status of the engineering analysis program.	X	X	X

Table 2
Optional MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
GetAllConnectivity	Requests a complete engineering model.	X	X	X
GetChildConnectivity	Requests all electrical features that are immediately downstream of a specified element.	X	X	X
GetDownlineConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetIsolatedConnectivity	Requests all circuit elements that have changed since a specific interconnection session.	-	-	-
GetParentConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model.	X	X	X
GetUpstreamConnectivity	Requests all elements that are upstream of a specified element.	X	X	X
GetDomainMembers	Requests a list of items in a domain.	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database.	-	-	-

1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

Figure F-8
Methods Listings in the Supported Web Service Methods Table

Why these methods are important to the user

Products: GIS Master and EA Pro

Summary of Interoperability Test Results
Interface #22
EA→GIS (Real Time)

Table 1
Recommended MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
CircuitElementChangedNotification	Used to publish changes in circuit elements to the GIS.	X	X	X
GetAllLoadFlowResults	Requests all available sets of load flow results.	X	-	-
GetAllShortCircuitAnalysisResults	Requests all available sets of short circuit analysis results.	-	-	-
GetLoadFlowResultsByObjectID	Requests a specific set of load flow results.	X	X	X
GetShortCircuitAnalysisResultsByObjectID	Requests a specific set of short circuit analysis results.	X	X	X
GetMethods	Requests a list of web service methods supported by the engineering analysis program.	X	X	X
PingURL	Queries status of the engineering analysis program.	X	X	X

Table 2
Optional MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
GetAllConnectivity	Requests a complete engineering model.	X	X	X
GetChildConnectivity	Requests all electrical features that are immediately downstream of a specified element.	X	X	X
GetDownlineConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetIsolatedConnectivity	Requests all circuit elements that have changed since a specific interconnection session.	-	-	-
GetParentConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model.	X	X	X
GetUpstreamConnectivity	Requests all elements that are upstream of a specified element.	X	X	X
GetDomainMembers	Requests a list of items in a domain.	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database.	-	-	-

1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

Figure F-9
Description Blocks in the Supported Web Service Methods Table

Tables of supported web service methods (continued). The last component of the supported web service methods table is the right-most columns that show which of the methods are supported by the server, supported by the client and verified by the interoperability test being documented. This region is highlighted in figure F-10.

Products: GIS Master and EA Pro Summary of Interoperability Test Results
Interface #ZZ
EA→GIS (Real Time)

Table 1
Recommended MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Interoperable ³
CircuitElementChangedNotification	Used to publish changes in circuit elements to the GIS.	X	X	X
GetAllLoadFlowResults	Requests all available sets of load flow results.	X	-	-
GetAllShortCircuitAnalysisResults	Requests all available sets of short circuit analysis results.	-	-	-
GetLoadFlowResultsByObjectID	Requests a specific set of load flow results.	X	X	X
GetShortCircuitAnalysisResults	Requests a specific set of short circuit analysis results.	X	X	X
GetShortCircuitAnalysisResultsByObjectID	Requests a specific set of short circuit analysis results.	X	X	X
GetMethods	Requests a list of web service methods supported by the engineering analysis program.	X	X	X
PingURL	Queries status of the engineering analysis program.	X	X	X

Table 2
Optional MultiSpeak Web Service Methods

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Interoperable ³
GetAllConnectivity	Requests a complete engineering model.	X	X	X
GetChildConnectivity	Requests all electrical features that are immediately downstream of a specified element.	X	X	X
GetDownlineConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetModifiedConnectivity	Requests all circuit elements that have changed since a specific interconnection session.	-	-	-
GetParentConnectivity	Requests all elements that are fed electrically by a specified element.	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model.	X	X	X
GetUplineConnectivity	Requests all elements that are upstream of a specified element.	X	X	X
GetDomainMembers	Requests a list of items in a domain.	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database.	-	-	-

1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

Are these methods supported by this pair of products?

Figure F-10
Listing of Supported Methods in the Supported Web Service Methods Table

G

Sample MultiSpeak Interoperability Assertion

MultiSpeak Version 3.0 Interoperability Assertion (SAMPLE)

Statement of Interoperable Functionality Between

Vendor(s)	Product	Product Version	Role	Batch Interface	Web Client Interfaces	Web Server Interfaces
XYZ Corporation	EA Pro	2.2	EA	GIS↔EA	GIS→EA	EA→GIS
ABC, Inc.	GIS Master	4.6	GIS	GIS↔EA	EA→GIS	GIS→EA

Summary

EA Pro and GIS Master are able to exchange data using either (i) a batch file transfer (see table 1) or (ii) in real time (see table 2) using MultiSpeak web services. Assuming that complete facility information is stored in GIS Master and those facilities are connected correctly, a circuit model can be exported from GIS Master and imported into EA Pro using a batch file transfer. EA Pro also can request this data using MultiSpeak web services (see tables 3-6). The engineer can make changes in the circuit model in EA Pro and publish those changes to GIS Master. Alternatively, GIS Master can request the circuit model from EA Pro.

GIS Master can obtain and display the results of a specific load flow or short circuit analysis that had been previously run in EA Pro.

Prerequisites

For this interface to work as intended, facilities to be modeled in EA Pro must be included in the GIS Master database in sufficient detail to create an engineering model, and those facilities must be connected correctly.

Specific Vendor Assertions

1) Products Will Synchronize Electrical Connectivity Data.

Importance to user: The GIS Master software maintains complete electrical connectivity on all electrical GIS features and can generate a complete engineering model from those features. This model can be imported by the EA Pro software and used for subsequent analyses. EA Pro can also make available a modified engineering model for use by the GIS Master software. Thus, the

engineering models in the two packages can be kept in synchronism without the need for manual data input. This is important because (i) it eliminates the potential for differing models being stored in the two applications, and (ii) it eliminates the need to keep the data up to date in two locations.

How achieved: GIS Master makes this engineering model (connectivity) available using either (i) a batch file transfer or (ii) the MultiSpeak GetAllConnectivity web service method. EA Pro can import this connectivity model by (i) importing the batch file or (ii) calling the GetAllConnectivity method hosted by the GIS. In addition, EA Pro can export the engineering model (connectivity) to GIS Master by hosting the GetAllConnectivity web service method; GIS Master can import this model by calling the GetAllConnectivity method.

2) Products Will Exchange Subsets of Data.

Importance to user: It is possible for both EA Pro and GIS Master to get a subset of the engineering model so that partial updates can be exchanged. This is important because it reduces the size of the files sent and thus minimizes the time and network bandwidth necessary to keep databases in agreement.

How achieved: Either program can get the following subsets of data on the other system:

- All electrical features that are immediately downstream of a specified element using the GetChildConnectivity method.
- All elements that are fed electrically by a specified element using the GetDownlineConnectivity.
- All elements that are fed electrically by a specified element using the GetParentConnectivity method.
- All elements that are upstream of a specified element using the GetUplineConnectivity method.
- The names of all substations included in the engineering model using GetSubstatonNames.

This capability is supported only in the real-time (web services) interface; it is not supported in the batch interface.

3) Software Can Publish Changes in Electrical GIS Features.

Importance to user: This will enable changes in the GIS model to be reflected in the engineering model in real time, without the need to specifically request such changes. This is important because the respective engineering models are kept in synchronism without the need for the user to request such changes.

How achieved: The GIS Master software keeps track of changes made to its engineering model and sends those changes to EA Pro without the need for user intervention. GIS Master publishes changes in electrical GIS features to EA Pro using the CircuitElementChangedNotification web service method on EA Pro.

4) Software Can Make Engineering Analysis Results Available for Display in GIS.

Importance to user: Once engineering analyses are completed, it is sometimes desirable to display these results in the same geographic context as that presented by a geographic information system. This enables software users to use the geographic display tools with which they are most familiar.

How achieved: EA Pro can make the results of engineering analyses available for display in the GIS Master system. GIS Master obtains these results from EA Pro using the GetLoadFlowResultsByObjectID and/or GetShortCircuitAnalysisResultsByObjectID web service methods. It is also possible to send these analysis results using a batch file export and import capability, if desired.

5) Software Can Support Generic Methods.

Importance to user: Generic methods permit two applications to be more tightly integrated by exchanging information about how communications are possible between the two programs and the status of communications. This is important because it permits the applications to query each other to ensure that the most appropriate level of integration is used.

How achieved: Both GIS Master and EA Pro provide the capability to inquire of the other which web service methods it supports by supporting the GetMethods web service method. Each program can query the other regarding its status using the PingURL web service method.

Products: GIS Master and EA Pro

**Summary of Interoperability Test Results
Interface #22
EA→GIS (Batch)**

**Table 1
MultiSpeak Data Objects Exchanged (Recommended)**

Object Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
Load flow results	The results from load flow analyses made in EA Pro can be displayed geographically in GIS Master.	X	-	-
Short circuit analysis results	The results from short circuit analyses made in EA Pro can be displayed geographically in GIS Master.	X	-	-
Connectivity, including	The set of data objects that collectively define an engineering model for a system is called connectivity. A connectivity model includes all of the objects shown as being indented in the first column. Support for all the objects, except fake node section, is required for a complete engineering model.	X	X	X
Overhead primary line		X	X	X
Overhead secondary line		X	X	X
Underground primary line		X	X	X
Underground secondary line		X	X	X
Capacitor bank		X	X	X
Transformer bank		X	X	X
Regulator bank		X	X	X
Overcurrent device bank		X	X	X
Switch bank		X	X	X
Service location		X	X	X
Substation		X	X	X
Motor		X	X	X
Generator		X	X	X
Fake node section		-	-	-

Notes:

1) EA Pro uses a line section-oriented engineering model. This interface uses line sections (rather than nodes) to exchange the engineering model.

2) EA Pro and GIS Master use a tightly coupled interface. This means that the name of a line section can be changed in either the engineering analysis or the GIS application without breaking the link between the two programs.

Products: GIS Master and EA Pro

**Summary of Interoperability Test Results
Interface #22
EA→GIS (Real Time)**

**Table 2
Recommended MultiSpeak Web Service Methods**

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
CircuitElementChangedNotification	Publishes changes in circuit elements to the GIS	X	X	X
GetAllLoadFlowResults	Requests all available sets of load flow results	X	-	-
GetAllShortCircuitAnalysisResults	Requests all available sets of short circuit analysis results	-	-	-
GetLoadFlowResultsByObjectID	Requests a specific set of load flow results	X	X	X
GetShortCircuitAnalysisResultsByObjectID	Requests a specific set of short circuit analysis results	X	X	X
GetMethods	Requests a list of web service methods supported by the engineering analysis program	X	X	X
PingURL	Queries status of the engineering analysis program	X	X	X

**Table 3
Optional MultiSpeak Web Service Methods**

Method Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
GetAllConnectivity	Requests a complete engineering model	X	X	X
GetChildConnectivity	Requests all electrical features that are immediately downstream of a specified element	X	X	X
GetDownlineConnectivity	Requests all elements that are fed electrically by a specified element	X	X	X
GetModifiedConnectivity	Requests all circuit elements that have changed since a specific interconnection session	-	-	-
GetParentConnectivity	Requests all elements that are fed electrically by a specified element	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model	X	X	X
GetUplineConnectivity	Requests all elements that are upstream of a specified element	X	X	X
GetDomainMembers	Requests a list of items in a domain	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database	-	-	-

- 1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
- 2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
- 3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

Products: GIS Master and EA Pro

**Summary of Interoperability Test Results
Interface #22
GIS→EA (Batch)**

**Table 4
MultiSpeak Data Objects Exchanged (Recommended)**

Object Name	Importance to User	Supported by Server ¹ (EA)	Supported by Client ² (GIS)	Verified Inter-operable ³
Circuit elements	The circuit element data object in MultiSpeak permits two programs to change the connection of existing line sections without the need to exchange all the data about those line sections.	-	X	-
Connectivity, including the following:	The set of data objects that collectively define an engineering model for a system is called connectivity. A connectivity model includes all the objects shown as being indented in the first column. Support for all the objects, except fake node section, is required for a complete engineering model	X	X	X
Overhead primary line		X	X	X
Overhead secondary line		X	X	X
Underground primary line		X	X	X
Underground secondary line		X	X	X
Capacitor bank		X	X	X
Transformer bank		X	X	X
Regulator bank		X	X	X
Overcurrent device bank		X	X	X
Switch bank		X	X	X
Service location		X	X	X
Substation		X	X	X
Motor		X	X	X
Generator		X	X	X
Fake node section		-	-	-

Notes:

- 1) EA Pro uses a line section-oriented engineering model. This interface uses line sections (rather than nodes) to exchange the engineering model.
- 2) EA Pro and GIS Master use a tightly coupled interface. This means that the name of a line section can be changed in either the engineering analysis or the GIS application without breaking the link between the two programs.

Products: GIS Master and EA Pro

**Summary of Interoperability Test Results
Interface #22
GIS→EA (Real Time)**

**Table 5
Recommended MultiSpeak Web Service Methods**

Method Name	Importance to User	Supported by Server ¹ (GIS)	Supported by Client ² (EA)	Verified Inter-operable ³
GetAllConnectivity	Requests a complete engineering model	X	X	X
GetMethods	Requests a list of web service methods supported by the GIS program	X	X	X
PingURL	Queries status of the GIS program	X	X	X

**Table 6
Optional MultiSpeak Web Service Methods**

Method Name	Importance to User	Supported by Server ¹ (GIS)	Supported by Client ² (EA)	Verified Inter-operable ³
GetChildConnectivity	Requests a complete engineering model	X	X	X
GetDownlineConnectivity	Requests all electrical features that are immediately downstream of a specified element	X	X	X
GetModifiedConnectivity	Requests all circuit elements that have changed since a specific interconnection session	X	X	X
GetParentConnectivity	Requests all elements that are fed electrically by a specified element	X	X	X
GetSubstationNames	Requests the names of all substations included in the engineering model	X	X	X
GetUplineConnectivity	Requests all elements that are upstream of a specified element	X	X	X
GetDomainMembers	Requests a list of items in a domain	-	-	-
GetDomainNames	Requests the names of all lists of items (domains). These can be used to make sure that two programs use the same codes or descriptions.	-	-	-
GetElectricalEquipment	Requests the names of features in the engineering database	-	-	-

- 1) Supported by Server means that the server has demonstrated in some interoperability test (not necessarily with this client) that it can support the method.
- 2) Supported by Client means that the client has demonstrated in some interoperability test (not necessarily with this server) that it can call the method.
- 3) Verified Interoperable means that both the client and server have demonstrated in this interoperability test that they can usefully transfer data using this method.

H

Referenced Standards

Extensible Markup Language (XML) 1.0. W3C Recommendation 04, Third Edition, February 2004. <http://www.w3.org/TR/REC-xml>

RFC2616: Hypertext Transfer Protocol (HTTP) 1.1. Internet Engineering Task Force, June 1999. <http://www.ietf.org/rfc/rfc2616>

Secure Sockets Layer (SSL) 3.0. Internet Engineering Task Force, Draft, November 18, 1996. <http://wp.netscape.com/eng/ssl3/draft302.txt>

Simple Object Access Protocol (SOAP) 1.1. W3C Note 08, May 2000. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

Universal Description, Discovery and Integration (UDDI) 2.04. *Version 2.04 API Specification*, Organization for the Advancement of Structured Information Standards (OASIS), UDDI Committee, July 19, 2002. <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>

Web Services Description Language (WSDL) 1.1. W3C Note 15, March 2001. <http://www.w3.org/TR/wSDL.html>

WS-1 Basic Web Services Profile 1.1. Web Services Interoperability Organization, Final, August 24, 2004. <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>