Paper 166-27

Data Socket Adapters

Sigurd W. Hermansen, Westat, Rockville, MD

ABSTRACT

Inspired by the idea of "… data sockets located in practically every room…" (as advertised on the Web site of the Department Physics of the University of Glasgow), this brief peek into the future gives those who can't stand to wait for it a preview of emerging data delivery technologies. In the first scene an Oracle equivalent of the Cable Guy asks, "Where's your premises distribution system outlet?" I shrug helplessly, so he pushes my desk away from the wall and exposes what appears to be an ordinary data socket. "That's the baby," he says. "Now what kind of adapter do you need for that employee dimension of your data warehouse?" I have no idea what he is talking about, but I don't want to spoil the plot for you. You'll have to show up and spend fifty minutes away from the Wild Toad Ride to hear the rest. To appreciate it you will have to know enough SAS to know that this isn't it. Yet.

OPENING

Ziggy Z. Laptop types laboriously at keyboard, then stares at screen. He shakes his head, grabs a cell phone, and with his thumb quickly punches in a telephone number.

"Hi.It's me again. For the Bids and Proposals component of the data warehouse, we need technical employee ID's, skill classifications, levels, and hourly wage rates. I'll need to speak with the Director of Human Resources Information Systems? She can talk with me now? Great."

(Aside to audience: By the way, you know this is fiction because in reality you never reach someone directly. You leave a message asking for a return call, and she'll call back during the 15 minute interval that you're away from your desk, saying that she is rushing off to a meeting, but you can reach her this afternoon between 3:30 and 3:45, right in the middle of your meeting with the Director of Financial Information Systems and …)

"Oh, Yes, Ms. Powers. As I was telling Bob, for the Bids and Proposals component of the enterprise data warehouse, we need technical employee ID's, skill classifications, levels, and hourly wage rates. Can you extract those data items from the HRIS database? We don't need personal identifiers, just a unique ID per person. Data socket adapters? Nope, never heard of them. Where can I find documentation ….

" Self-documenting ….? You'll send a cable guy out to help me install a data socket adapter? How long will that take? We're in a bind here, and …."

THE CABLE GUY

" Hello, Ziggy!" he calls out. "I'm the cable guy."

"Never mind. He's here."

"I understand that you need a data socket adapter, right?"

"I guess so…."

"Where's the premises distribution system outlet?"

Ziggy shrugs helplessly, so the Cable Guy pushes his desk away from the wall and exposes what appears to be an ordinary data socket.

"That's the baby. Now what kind of adapter do you need for that employee dimension of your data warehouse?"

"I guess I need one that I can use to extract the data I need. Actually I was hoping for a simple file of some sort."

"Not around here, my friend. What happens when we update the HRIS database?"

"You could send me a new file?"

"That could be habit-forming. You'll do better with this."

The cable guy pulls a small device from his pocket and, after disconnecting the network cable, plugs the device into the wall socket and plugs the network cable into the device.

"That's all there is to it."

Ziggy stoops down to take a look at the adapter.

"So now it's installed. What am I supposed to do with it?"

"Get the data you need. Start up your Web browser and enter the service and address http://dataSockets ."

This pops up:

LAYERS	
PHYSICAL	LOGICAL
server	data source
⇕	⇕
adapters	adapters
⇕	⇕
sockets	**sockets**
⇕	⇕
network	data stream
⇕	⇕
sockets	**sockets**
⇕	⇕
adapters	adapters
⇕	⇕
client	data object

"I started out installing TV cables, so I find it easier to explain data socket adapters in terms of network connections. A server stores data. A cable connected to the network plugs into a socket on the server. That's like the hardware that connects a TV station to the cable grid. In a computer network, the server actually transfers data from data sources through a "port", a directory path associated with a socket connector. On the network side, a logical socket binds a port to a Universal Resource Link (URL) that resolves to an Internet Protocol (IP) address. That works something like a TV channel A server defines a logical socket and publishes a URL, to which clients can forward requests for data.

"Now take a closer look at the dataSockets diagram. Notice that hardware sockets on servers and clients pair up to logical sockets. A few years ago different types of data transmissions

used different hardware sockets. Printers, for example, connected to servers via parallel ports, while modems connected to serial ports. The connector "shaped" or "adapted" data transmissions to match the protocol accepted by other devices on a network. Of course one hardware socket can support more than one logical socket.

"For the most part today, data packets 'stream' between two logical sockets. Logical server and client sockets have become a class of objects with property parameters and methods for requesting and targeting data streams."

Ziggy frantically signals for a "time out".

"I'm not sure that I understand all that you are telling me about data sockets. Could we walk through what to do first?"

"Sure thing. You haven't searched the catalog of socket adapter objects yet, have you?"

"No."

"Then that's what we need to do first."

SOCKET ADAPTER METADATA

"What's the catalog of socket adapter objects?"

"That's where we store the metadata that describe logical socket adapters. You know how metadata of a database describe tables and their columns. The metadata of logical socket adapters provide the URL's related to sockets, the values of socket parameters, and, in very general terms, the method used to select and package data.

"Let's take a quick look at one of the socket adapters currently in the Enterprise data collection."

The cable guy types http://socketAdapters in the Address slot of the browser. Up pops a page:

reference	connect
enterprise	Consultoids
schema	HRIS
view	techstaff
sockets	sockets

"The enterprise name, schema, and view identify each instance of a socketAdapter; each instance has a different combination of the values of these data items. The reference links to a view of the schema. Clicking on it causes the view (assuming that you have access rights) to execute and streams an extract from data sources into the location specified in the view. The view data item links to the text of the view program and serves as a modifiable source for the view. As an additional security measure, the server automatically joins user access rights to the results and restricts the view to what you can see. This keeps the bad guys from illegally using a data socket to get to our data. A socketAdapter has no more data access privileges than its user.

"The schema data item references a Web page that launches another view, in this case, of the metadata that describe table structures and column variable types. The sockets data item links to a page that shows the methods for creating and property settings of the input and output sockets.

"So that's it. A socketAdapter's page shows you the links that you need to choose an existing socketAdapter, or to find one that you can adapt to your requirements."

Ziggy looks puzzled.

"It all sounds very convenient, cable guy… Let me ask you this, though. How do you know what you are getting? How does it work?"

"Guess you'll just have to take my word for it, won't you. You don't trust me?"

"Sure, I trust you. I trust you, but I can't trust you the way I would a better known software provider, like Microsoft, who as we all know always produces totally reliable and error-free software. I need to know something about how these socket-Adapters work! Give me a few minutes to check out the documentation."

The Cable Guy hangs his head and sulks on his way off stage.

BEHIND THE SCENES

Ziggy starts a dialogue with the audience.

"Finally, I'll have a chance to look at what's going on here. It takes some time nowadays to come to grips with the latest versions of data systems. Every vendor out these is trying to grab a piece of the market for distributed databases and Internet presence. Nothing ever connects directly to data. Everything goes through this vendor's middleware under that protocol and connects into that vendor's universal database server. It's worse than the phone system! Right?

"Also, ever notice that it always happens that the feature that you really need to connect to a database through the Web won't be available until the next version of a DBMS? I swear that these guys are stringing us along.

"Now let's take a look at the data sockets that supposedly drive this so-called dataAdapter."

Ziggy clicks on sockets. Up pops

```
*SERVER INPUT SOCKET CONNECTION;
/* Server waits for data requests here. */
   filename in_msg SOCKET ':70' SERVER;

*SERVER OUTPUT SOCKET CONNECTION;
/* Responses to data requests go thru here. */
   filename webout SOCKET 'server:80';

*CLIENT OUTPUT SOCKET CONNECTION;
/* Send requests for data thru here. */
   filename  clntOUT SOCKET 'client:70'
   recfm=V termstr=CRLF;

*CLIENT INPUT SOCKET CONNECTION;
/* Receive responses to data request thru here. */
   filename clientIN SOCKET ':80' server;
```

"They look strangely familiar, don't they? Any SAS programmer would recognize them immediately as variations of FILENAME statements. It doesn't take too great a stretch of imagination to think of sockets as input or output data streams, much like pipes of files, that a SAS program can read or write.

"Now what are we actually doing with the sockets? Let's check the view.

Ziggy clicks on techStaff and up pops

I knew that you would need my help, my friend, and you've come to the right cable guy. Here's what the server is running to listen for requests for views of data:

Server "daemon" listening for data requests:

```
<server path>\server.sas
/* Adapted from Gleason and Yu. */
%macro server(portno);
%do %while ( 1 );
filename in_msg SOCKET ":&portno" SERVER;
filename temp_pgm "
%sysfunc(pathname(work))/temp_pgm.sas" ;
data _null_;
infile in_msg;
file temp_pgm;
input ;
put _infile_;
run;
filename in_msg;
%inc temp_pgm;
run;
filename temp_pgm;
%end;
%mend server;
%server(70)
```

"What the … This seems to be some form of instant messaging spliced on a Web page …."

```
Now the server is monitoring the socket
in_msg for data requests. The clients sends
requests ….
```

Client request:

```
c:\my documents\getData.sas
/* Submit view program techStaff.sas */
filename techStaf "c:\my
documents\techStaff.sas";
filename  clntOUT SOCKET 'client:70'
recfm=V
termstr=CRLF;
data _null_;
  infile techStaf;
  input;
  file clntOUT;
  put _infile_;
run;

Now, back to your regular programming…. CG
```

"That guy seems to be reading my thoughts… Um, let's see. The initial view of the data looks simple enough."

DBMS View: techStaff:

```
<server path>\HRIS.mdb (query=techStaff)
SELECT *
FROM Employees
WHERE class="Technical";
```

"Let's take a look now at then rest of this page:"

Extraction of View:

```
c:\my documents\techStaff.sas
/* Write selection from database to XML
document prior to transport. */
PROC IMPORT OUT= WORK.techStaff
/*Import techStaff view*/
DATATABLE= "techStaff"
DBMS=ACCESS97 REPLACE;
DATABASE="<server path>\HRIS";
USERID="admin";
PASSWORD="";
RUN;
libname techStaf xml "<server
path>\techStaff.xml";
data techStaff.personTableVars;
set techStaff;
run;
libname techStaf;
%let portno= 80;
filename Webout SOCKET "server:&portno";
filename techStaf "<server
path>\techStaff.xml";
data _null_;
infile XMLfile;
file webout;
input;
put _infile_;
run;
filename XMLfile;
```

It's me again, Buddy. Here's what the adapter is really doing. The server has thrown out the data into an XML document. It can then stream the XML file thru the webout socket to the client. The client catches the file, …

```
filename clntOUT;
*CLIENT INPUT SOCKET CONNECTION;
filename clientIN SOCKET ':80' server;
flename XMLfile
"c:\windows\temp\XMLfile.xml";
data testServer;
 infile clientIN;
  input;
  file XMLfile;
  put _infile_;
run;
filename clientIN;
filename XMLfile;
```

parses the XML document, and converts it to a SAS dataset ….

```
libname techStaf "c:\windows\temp";
libname metaXML xml
"c:\windows\temp\XMLfile.xml";
data techStaf.personTableVars;
  set metaXML.personTableVars;
run;
```

Voila! CG

SCHEMA

"But what if I have to adapt a socketAdapter to extract a different view of a database? How do I find out how the server database's organized?"

"Me again, again. At this stage you are probably wondering how to modify a socketAdapter. Start with the schema link …

Ziggy obediently clicks on the schema and up pops a table showing column names and attributes by table.

The link actually starts up a query on the server and extracts the database metadata into an XML document. It streams the document as a file thru the server socket to the client socket.

/* Write selection from database to XML document prior to transport. */

```
PROC IMPORT OUT= WORK.Employees
    /*Import techStaff view*/
    DATATABLE= "Employees"
    DBMS=ACCESS97 REPLACE;
   DATABASE="c:\windows\temp\HRIS";
    USERID="admin";
    PASSWORD="";
RUN;
PROC IMPORT OUT= WORK.Assignments
    /*Import techStaff view*/
    DATATABLE= "Assignments"
    DBMS=ACCESS97 REPLACE;
   DATABASE="c:\windows\temp\HRIS";
    USERID="admin";
    PASSWORD="";
RUN;
PROC IMPORT OUT= WORK.Benefits
    /*Import techStaff view*/
    DATATABLE= "Benefits"
    DBMS=ACCESS97 REPLACE;
    DATABASE="c:\windows\temp\HRIS";
    USERID="admin";
    PASSWORD="";
RUN;
proc sql noprint;
create view EmplVW as
select memname as table,name as colname,label
as collabel,length,type,format,informat from
dictionary.columns
   where MEMTYPE="DATA" and
MEMNAME="EMPLOYEES"   ;
create view AssignVW as
select memname as table,name as colname,label
as collabel,length,type,format,informat from
dictionary.columns
   where MEMTYPE="DATA" and
MEMNAME="ASSIGNMENTS"   ;
create view BeneVW as
select memname as table,name as colname,label
as collabel,length,type,format,informat from
dictionary.columns
   where MEMTYPE="DATA" and
MEMNAME="BENEFITS"   ;
quit;
/*
data Empl;
  set EmplVW;
run;
*/
libname HRISmeta XML
"c:\windows\temp\HRISmetadata.xml";
data HRISmeta.HRISmetadata;
  set EmplVW AssignVW BeneVW;
run;
libname HRISmeta;

%let portno= 80;
filename webout SOCKET "a30360:&portno";
filename HRISfile
"c:\windows\temp\HRISmetadata.xml";
data _null_;
   infile HRISfile;
   file webout;
   input;
   put _infile_;
run;
   filename HRISfile;
```

The cool part of this is the way the SAS® System engine reads and writes an XML documents the way it would a SAS dataset, but in the next step writes or reads the XML document as a data stream to or from a socket! It's like e-mailing pizza!

You know you can send me a message by clicking on the message box at the bottom of this page, don't ya. What do you think, so far? CG

REMOTE SERVERS AND SERVICES

Ziggy clicks on the message box, types in a message, and sends it to the screen …

These socketAdapters do seem useful, I admit, but don't most people want something more substantial, something more industrial strength, more Oracle, more Microsoft … ZZ

Not a problem… Not particularly interesting, but not a problem… A lot of you corporate types have a hard time trusting cable guys with a bigtime database, but we can work with anyone that can create data sockets, handle data requests, and return "well-formed" XML as a data stream.

A MS SQLServer database can process a view and return an XML document. To do that use a template file as a package for a SQL statement:

<root smlns:sql="urn:schemas-microsoft-com:xml-sql".
>
<sql:query>
SELECT …. FROM …. FOR XML RAW
</sql:query>
</roor>

This SQL statement fits into an XML structure. Saved as <file>.xml and entered in a browser ADDRESS: as http://<path>/<file>.xml, SQLServer executes the contents of the document as a SQL query and returns the results of the query to the browser as an XML document. The "FOR SQL RAW" clause puts each row of a table into attributes of a <row> … </row> "line" of an XML document.

As before, a view that extracts metadata from a database works just like an view that extracts data. You don't have to be a Bill Gates to figure out how to adapt a socket to do what you want.

The process works much the same for an Oracle server. An Oracle XSQL datapage might look like this:

<?xml version="1.0"?>
<xsql:query connection="test" xmlns:xsql="urn:oracle-sql">
SELECT …. FROM … WHERE lastname='{@LN}'
</xsql:query>

A request for information on Adams would point to an Oracle database server and include the parameter value 'Adams':
http://<url>/<view>.xsql?LN=Adams
CG

INTRANET, INTERNET, AND BEYOND

All of this confuses me, I have to admit. These socketAdapters look a lot like SAS FILENAME and LIBNAME statements, SQL queries, and ordinary types of files. Even the SQLServer and Oracle views look pretty ordinary. Where's the emerging technology in socketAdapters? ZZ

As best I see it, ZZ, all of the major database system vendors are trying to become the universal server in the next Universal Operating System (UOS). The Web looks to me like the next UOS, so I have to like the chances of XML databases and XQUERY views. In this new technology, I see the brand name databases becoming less important and any number of data sources on the Web beginning to look the same from the point of view of the Internet client. Are you with me on this? CG

What do you have to offer in the way of data security, speed, and reliability. These homegrown systems make my clients nervous. What do you have to offer in COTS products? ZZ

If you really need a security blanket, take a look at these products:

SAS® Intrnet
Onyx®
Perl DBI
SAS®/Connect for Java
SAS®/SHARE for JDBC

You can install a reliable and secure SAS® server and still use our socketAdapters method to connect clients and servers in an Intranet or across an Internet. CG

CONCLUSION

Just one more thing, cable guy. That little connector gadget, the adapter, that you attached to my data socket. What does it actually do? ZZ

OK, you pinned me on that one! Busted! The data sockets handle the transfers of data streams. The database systems handle the parsing of XML documents and conversions of XML data and embedded metadata to database tables. The adapter doesn't do anything at all. It's just that some people like to be able to point to something and say 'that is what connects me to database servers'. It's a harmless diversion. Don't ruin it for them. CG

REFERENCES

Gleason, Chapman and Hsiwei Yu, "Using the socket access method to unite SAS with the Internet", *Proceedings of the Twenty-sixth Annual SAS Users Group International Conference*, 24, 83.

Ward, David L., "Using Sockets in SAS Software for Internet Publishing", *Proceedings of the Annual Southeast SAS Users Group Conference 2K*, Systems Architecture Section.

ACKNOWLEDGMENTS

Special thanks to Duke Owen and Mike Rhoads at Westat for reviewing the paper. David Ward of InterNext, Inc., a true socket master, must share credit with the Glasgow Physics Department for inspiring this paper. The author accepts full responsibility for errors or oversights.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sigurd W. Hermansen
Westat, An Employee-Owned Research Company
1650 Research Blvd.
Rockville, MD 20815
Work Phone: (301) 251-4268
Fax: (301) 315-5936
Email: hermans1@westat.com