

COMET/XML Data Exchange Utility

March 2003

Versions: XMLQUERY 1.6
XMLPOST 1.6

COMET/XML Data Exchange Utility

Table of Contents

<u>Section</u>	<u>Topic</u>	<u>Page</u>
1	Introduction.....	3
2	System Requirements.....	3
3	System Archetecture.....	3
4	XML Query Request.....	4
5	XML Query Response.....	6
6	XML FSTAT Function.....	9
7	XML Post Request.....	10
8	XML Post Response.....	12
9	Security.....	14
10	Logging.....	15
11	Sample HTML.....	16
12	Application Notes/Notices.....	17
Appendix A	Error Listing.....	18

COMET/XML Data Exchange Utility

1. Introduction

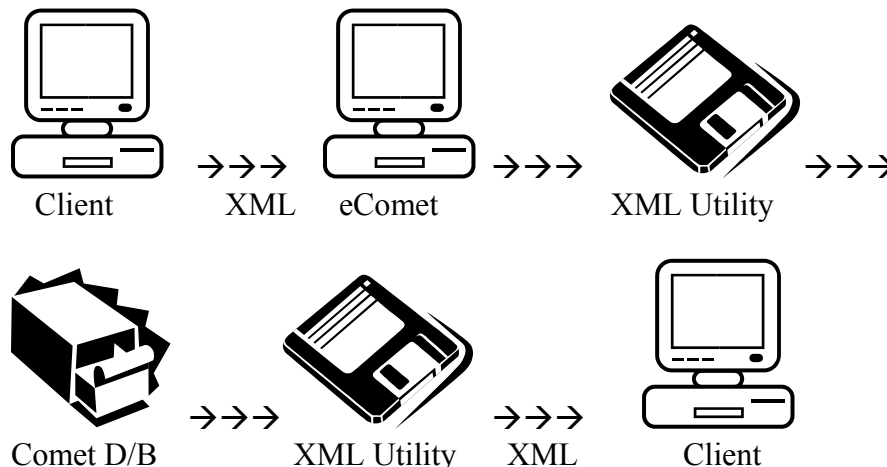
XML (Extensible Markup Language) is an emerging standard developed for the exchange of data between systems. It has a specific structure but is an open model. The following is an implementation specifically for the exchange of data from any platform supporting XML to the COMET system. The system relies on the data definitions of the COMET database to be accessed to reside in the #FILES (COMET's data dictionary).

2. System Requirements

COMET operating system provided by [Signature Systems](#)
eCOMET gateway (XAP) provided by [Signature Systems](#)
XML Utilities XMLQUERY & XMLPOST provided by Jonathan Sacks
TCP/IP access
Windows 9x or 2000

3. System Architecture

A remote system makes a request of the XML application residing on an eCOMET (XAP) server via TCP/IP to the specified Winsock port (usually 80 http). The XAP server parses the raw data in the http stream into a file that is passed to the XML application. The XML server executes the XMLQUERY or XMLPOST utility. The utility parses the data from the XAP file and processes the request back to the requesting system via the requesting port.



COMET/XML Data Exchange Utility

4. XML Query Request

4.1 Query Request: Query request are sent to the XML system to ask for data from the database. They are structured in a hierarchy of Envelops using the <request> </request> convention to wrap a segment or group of segments together. The parser understands these requests and produces the query. Requests are not case sensitive.

4.2 Query Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<queryset>
<system="system ID"/>
<query file="#FILES FILE NAME">
  <select start="[Starting Key]" end="[Ending Key]" />
  <condition field="#FILES FIELD NAME" relop="[EQ/NE/GT/GE/LT/LE]" "[VALUE NAME]" />
  <print fields="#FILES FIELD],[#FILES FIELD],....." />
</query>
</queryset>
```

4.3 Query Definitions:

<?XML version "1.0" encoding="iso-8859-1"?> – *(Required)* Identifies the XML version. *Note: Current system only supports this version.*

<QUERYSET> – *(Required)* Identifies the “Envelope” in XML containing a request for data from the remote system.

<SYSTEM> - *(Optional)* Overrides the SYSTEM ID sent in the command line (see security).

<QUERY FILE="#FILES FILE NAME"> - *(Required)* The name of the file requesting access to as defined in the #FILES definitions.

<SELECT START="[Starting Key]" END="[Ending Key]" /> - *(Optional)* Defines the starting and ending key (index) in the COMET file to be searched in the query.

<CONDITION FIELD="#FILES FIELD NAME" RELOP="[relop]" "[value]" />
- *(Optional)* Defines the condition to return data for this request (i.e. Selection). Valid relationships are EQ-Equal To, NE-Not Equal To, GT-Greater Than, LT-Less Than, GE-Greater Than or Equal To, LE-Less Than or Equal To. Value is the value of the relationship match.

COMET/XML Data Exchange Utility

<PRINT FIELDS="#FILES FIELD],#FILES FIELD],....." - *(Required)* Identifies the fields (as defined in #FILES) for the XML system to return to the user.

</QUERY> - *(Required)* Identifies the end of the QUERY envelope.

</QUERYSET> - *(Required)* Identifies the end of the QUERYSET envelope.

Note: Multiple "Queries" can be sent within a Queryset, there is currently a limitation in COMET of 999 lines total in an XML request. Additionally there is a limitation of 1024 bytes per query line.

COMET/XML Data Exchange Utility

5. XML Query Response

5.1 Query Response: The XML system sends back a structured response to the XML request in XML format.

5.2 Response Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="[#FILES FILE NAME]" system="[System ID]">
    <record keyvalue="[COMET Key Value]" nextkey="[COMET Next Sequential Key Value]">
      <field name="[#FILES FIELD NAME]" orig="[COMET Database Value]" />
    </record>
  </data>
</dataset>
```

5.3 Response Definitions:

<?XML version "1.0" encoding="iso-8859-1"?> - *(Required)* Identifies the XML version. *Note: Current system only supports this version.*

<DATASET> - *(Required)* Identifies the "Envelope" in XML containing a response of data from the remote system. *Note: Ver. Is optional and only identifies the version of the internal XML software back to the remote user.*

<DATA FILE="[#FILES FILE NAME]"> - *(Required)* The name of the file being returned as defined in the #FILES definitions.

<RECORD KEYVALUE="[COMET Key Value]" NEXTKEY="[Next Key]"> - *(Required)* Keyvalue represents the actual key of the data being returned from the COMET system. Nextkey represents the next sequential key after the current Keyvalue.

< FIELDS NAME="[#FILES FIELD] ORIG="[Comet D/B Value]" /> - *(Required)* Field name denotes the #FILES name as requested with the value returned in the field ORIG (Original Value).

</RECORD> - *(Required)* Identifies the end of the RECORD envelope.

</DATA> - *(Required)* Identifies the end of the DATA envelope.

</DATASET> - *(Required)* Identifies the end of the DATASET envelope.

Note: Multiple "Data" envelopes can be contained in a single dataset. There is currently no limitation on the number of data "records" that can be returned. Do to limitations in COMET no record can exceed 1024 bytes.

COMET/XML Data Exchange Utility

5.4 Response Error

The XML system uses a structure of failcodes in an XML response to indicate an error in processing the request. The failcode is returned at the highest envelope level where the error has occurred. Some errors are fatal and others soft. Fatal errors stop the processing of the balance of the query while soft errors allow for continued processing.

5.4.1 Query File Level Error (Fatal)

The following format will be returned for a Query File Level Error:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="[Requested File Name]" failcode="[XX]" />
</dataset>
```

Note: The system does not return a failcode name/value pair when no error exists.

5.4.1.1 Query File Failcodes:

11 – File Not Found
54 – Invalid Function for Named File

5.4.2 Condition Level Error (Fatal)

Since the dataset is asking for a subset of data from the file the existence of a bad field or condition must be returned as a fatal error.

The following format will be returned for a Condition error:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="[Requested File Name]" system="[System ID]">
    <condition field="[Requested Field Name]" failcode="[XX]" />
  </data>
</dataset>
```

5.4.2.1 Condition Failcodes:

08 – Field Not Found
09 – Invalid Relationship/Condition

COMET/XML Data Exchange Utility

5.4.3 Record Keyvalue Error (Soft)

The only error that can be returned in a failcode at the Record Keyvalue level is an end of file. The following XML is returned when an end of file has been reached.

5.4.3.1 Record Keyvalue Error Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="[#FILES File Name]" system="[System ID]">
    <record keyvalue="[Current Key]" nextkey="" failcode="02">
      <field name="[#FILES Field Name]" orig="[Value]" />
    </record>
  </data>
</dataset>
```

5.4.3.2 Record Keyvalue Failcodes:

02 – End of File

5.4.4 Print Fields Level Error (Soft)

Errors at the Print level are not fatal. A failcode is returned only in the event of an error. It is up to the programmer receiving the data to code to find the failcode in the Print Filed Envelope.

5.4.4.1 Print Filed Level Error Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="[#FILES File Name]" system="[System ID]">
    <record keyvalue="900000" nextkey="900400">
      <field name="[Requested Filed Name]" orig="" failcode="08" />
    </record>
  </data>
</dataset>
```

5.4.4.2 Print Field Failcodes:

08 – Field Not Found

COMET/XML Data Exchange Utility

6.0 FSTAT Function

The FSTAT function returns the Comet FSTAT Data in XML format. It can be combined with other requests in the same session or executed on a stand a lone basis. This function is helpful if you want to “synchronize” files between two machines by looking at the update date on the host.

6.1 FSTAT Request Example

```
</queryset>
  <fstat file="#"[#FILES FILE NAME]"/>
</queryset>
```

6.2 FSTAT Response Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.6">
  <data file="CUSTOMER    " system="TST">
    <fstat file="C1A    " directory="TST">
      <fstat name="WHOLENAME" value="C1A    "/>
      <fstat name="DIRECTORY" value="TST"/>
      <fstat name="FILETYPE" value="K"/>
      <fstat name="RECORDSIZE" value="0778"/>
      <fstat name="KEYLENGTH" value="06"/>
      <fstat name="CREATEMMDDYYYY" value="02172001"/>
      <fstat name="UPDATEMMDDYYYY" value="06192002"/>
      <fstat name="UPDATEHHMMSS" value="175842"/>
      <fstat name="OPENCOUNT" value="000"/>
      <fstat name="EXTRACTCOUNT" value="000"/>
      <fstat name="NEXTFILE" value="C1C    "/>
    </fstat>
  </data>
```

Refer to the Comet documentation at: <http://www.internetbasic.com> FSTAT function for details on the fields.

COMET/XML Data Exchange Utility

7. XML Post Request

7.1 XML Post Request: The XML system will process a request to Add, Update or Delete records in the COMET database. The current model provides no ability to “Extract” data. To insure data integrity the Posting of XML data requires that a Query be done prior to any post request to obtain an original value for a field. The post request must be made supplying “Original Value” and “New Value”. The XML post processor verifies that the Original Value as returned matches the current data in the database before posting. Should the data Original Value not match a mismatch response is returned to the sender.

7.2 Post Request Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
<system="[System ID]"/>
  <data file="#FILES FILE NAME">
    <record keyvalue="[Key]" action="[update/add/delete]" />
      <field name="#FILES FIELD" orig="[Original Value]" new="[New Value]" />
    </record>
  </data>
</dataset>
```

7.3 Post Request Definitions:

<?XML version “1.0” encoding=“iso-8859-1”?> – *(Required)* Identifies the XML version. *Note: Current system only supports this version.*

<DATASET> - *(Required)* Identifies the “Envelope” in XML containing a response of data from the remote system. *Note: Ver. Is optional and only identifies the version of the internal XML software back to the remote user.*

<SYSTEM> - (Optional) Overrides the SYSTEM ID sent in the command line (see security).

<DATA FILE=“#FILES FILE NAME”> - *(Required)* The name of the file to be modified as defined in the #FILES definitions.

<RECORD KEYVALUE=“[COMET Key Value]” ACTION=“[U/A/D]”> - *(Required)* Keyvalue represents the actual key of the data to be updated from the COMET system. Action defines the type of change to the system that is being requested. Update - Existing keys, Add – New Keys and Delete of an existing key.

COMET/XML Data Exchange Utility

< FIELDS NAME="#"[#FILES FIELD] ORIG="#"[Comet D/B Value]" NEW="#"New D/B Value]" /> - *(Required)* Field name denotes the #FILES field name to be updated. This must contain value returned in the field ORIG (Original Value) in the Query Response. NEW denotes the value to change ORIG to.

</RECORD> - *(Required)* Identifies the end of the RECORD envelope.

</DATA> - *(Required)* Identifies the end of the DATA envelope.

</DATASET> - *(Required)* Identifies the end of the DATASET envelope.

Note: Multiple "Data" envelopes can be contained in a single dataset.

COMET/XML Data Exchange Utility

8.0 XML Post Response

8.1 XML Post Response: The XML system acknowledges every request for a post. The post facility returns a “failcode” for each transaction. A failcode of “0” indicates success in the post, a failcode of 1 indicates a original value mismatch. Other fail codes are returned based on the current COMET Exception statement.

8.2 XML Post Response Example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.2">
  <data file="#FILES FILE NAME" system="[System ID]">
    <record keyvalue="[KEY VALUE]" action="[OPTION REQUESTED]" >
      <field name="#FILES FIELD NAME" failcode="[xx]" />
    </record>
  </data>
</dataset>
```

8.3 XML Post Response Definitions:

<?XML version “1.0” encoding=”iso-8859-1”?> – *(Required)* Identifies the XML version. *Note: Current system only supports this version.*

<DATASET> - *(Required)* Identifies the “Envelope” in XML containing a response of data from the remote system. *Note: Ver. Is optional and only identifies the version of the interal XML software back to the remote user.*

<DATA FILE=”[#FILES FILE NAME]”> - *(Required)* The name of the file modified as defined in the #FILES definitions.

<RECORD KEYVALUE=”[COMET Key Value]” ACTION=”[U/A/D]”> - *(Required)* Keyvalue represents the actual key of the data to be updated from the COMET system. Action defines the type of change to the system that is being requested. Update - Existing keys, Add – New Keys and Delete of an existing key.

< FIELDS NAME=”[#FILES FIELD]” ORIG=”[Comet D/B Value]” NEW=”New D/B Value]”/>- *(Required)* Field name denotes the #FILES field name processed.

</RECORD> - *(Required)* Identifies the end of the RECORD envelope.

</DATA> - *(Required)* Identifies the end of the DATA envelope.

</DATASET> - *(Required)* Identifies the end of the DATASET envelope.

COMET/XML Data Exchange Utility

Note: Multiple “Data” envelopes can be contained in a single dataset.

8.4 Post Response Failcodes:

00 – Successful

01 – Original Value to Current Value Mismatch

56 – Record Already Exists for INSERT

Note: All other failcodes are the equivalent COMET Exception.

COMET/XML Data Exchange Utility

9. Security

The system is secured by use of a command line argument SYSTEM and use of the COMET security file containing the DAB's for access. The XMLQUERY and XMLPOST expect to find a record keyed by "XML"+[password] in the security file. Each program then processes an ACCESS statement in order of the DAB's listed in the security file.

9.1 Command line argument example:

[http://\[ecomet.host.url\]/xap/xmlquery?SYSTEM=TST](http://[ecomet.host.url]/xap/xmlquery?SYSTEM=TST)

In this example the system is expecting to find a password of "XMLTST" in the system security file.

9.2 Non-Secured System:

The system does not require that the password exists or that the security file exists. In this case the system "inherits" the DABs of the current partition configuration.

9.3 User Logging:

The system allows for a command line argument USER to be logged. As of Version 1.4 this is now a MANDATORY argument. The system does not validate the USER but the argument must be present.

9.4 Command line argument example:

[http://\[ecomet.host.url\]/xap/xmlquery?SYSTEM=TST&USER=TestUser1](http://[ecomet.host.url]/xap/xmlquery?SYSTEM=TST&USER=TestUser1)

In this example the system is will return data for the TST password and log the request from USER TestUser1.

9.4.1 XML Response for Invalid Security example:

In the event that a USER is not specified the system will return the following XML response. This response is FATAL and all processing is stopped.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dataset ver="1.4">
<user id="[USER]" failcode="01"/>
</dataset>
```

COMET/XML Data Exchange Utility

10. Logging

The system creates a log for every transaction that is processed by XMLQUERY and XMLPOST. The files that contain the data are XMLLOG.HTM (Query Log) and XMLPLOG.HTM (Post Log) on the XAP directory. The file is appended to and expects an operator to erase the file periodically. It can be viewed in any standard browser.

10.1 Log Example

```
Session ID:05040114:03:02.19324
```

```
Remote Referrer: 12.3.11.102
```

```
HTTP Agent: Java1.3.0
```

```
-----Start Parser Diagnostics-----
```

```
Screen Diagnostics: Off
```

```
System Access: TST
```

```
User: user1
```

```
#FILES Name=P.O.DETAIL ,Actual File=Q1
```

```
Select Start=103845 ,Select End=103859
```

```
Field 1 -CUSTOMER.CODE/Q1CSCD$ /S/ 3/338
```

```
Field 2 -ITEM.NUMBER/QINBR$ /S/ 12/ 24
```

```
-----End Parser Diagnostics-----
```

```
Start Time:14:03:02.19
```

```
End Time: 14:03:02.22
```

```
Total Records Read: 13 , Records Returned: 13 , Fields Returned: 26
```

```
Returned Percent:100%,Total Inquiry Time:.03Seconds,Records/Sec: 433.3
```

10.2 Logging Options

There is an optional parameter to disable logging <logoff/>. This parameter will turn the logging function off for an XML Query or Post for the specified transaction. The parameter should follow the <QUERYSET> or <DATASET> commands.

10.3 Example of Log Off

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<queryset>
```

```
<logoff/>
```

```
.
```

```
.
```

```
</queryset>
```

COMET/XML Data Exchange Utility

11. Sample HTML

The XML system can be accessed via a browser or any other system that can send HTML requests via http. This sample is provided only as a guide. The system expects to receive <CR><LF> delimited data identified in a field name "XML". The XAP system parses the data in name/value pairs internally of XMLnnnn for each line received. The XML processor then reads this data sequentially from the XAP system.

11.1 HTML Query Example

The following is a sample HTML fragment that can be used to generate a request via the browser to the system. *Note: The HTML must be modified to include the URL of the host (XAP) system.*

```
<h3 align="center">Enter XML Code Here:</h3>
<form action="http://[xap.host.system]/xap/xmlquery" method="POST">
  <input type="hidden" name="SYSTEM" value="TST">
  <palign="center">
  <textarea name="XML" rows="10" cols="80" wrap="SOFT"></textarea></p>
  <input type="submit" name="TextAreaSubmit" value="XML Markup Submit"><br>
</form>
```

This example provides a "textbox" for entry of the XML request. An example of the implementation of this code is at <http://xml.ebextranet.com>.

11.2 Diagnostic option

The following HTML can be added to include a check box to turn browser diagnostics on and return data to the user in HTML format.

```
<p align="center">Diagnostics <input type="checkbox" name="XML000" value="ON"><br>
```

Note: If the value in the field XML000 is set to ON, the system will return the XML data in HTML format. This is helpful for diagnosing a problem of an unexpected response.

COMET/XML Data Exchange Utility

12. Application Notes/Notices

12.1 Known Bugs/Issues

With release of Version 1.6 there are no reported bugs or issues.

12.2 Security Concerns

An enhancement to the system should be made to force security. Currently the system will give access to all the configured DABS on the system.

The programs default to a system "XML". To limit access to the system it is recommended to include a password in the system password file "XMLXML" with access to a limited number of DABS. This will limit access by the user to only the DABS specified in this password.

12.3 Sample Applications

Sample HTML Browser applications of the Query and Post processors are available at xml.ebextranet.com. The pages may be modified and used freely as long as credit is given to the author.

12.4 Warrantees and Representation

There is no warrantee on this product expressed or implied. Use of this software is made by the user at the users risk. Jonathan Sacks, E&B Giftware, LLC, affiliates and licensees take no responsibility for data corruption caused by this system. Best efforts have been made to insure that the platform is stable but do to unforeseen issues the user understands that they will use this software at their own risk. This is a general access tool and must be installed with the knowledge that securing the system is the responsibility of the user.

12.5 Copyrights

Comet, eComet, XAP are copyrights of signature systems www.signature.net.

This document and companion programs are the property of Jonathan Sacks and may only be used under expressed written consent from the Author.

12.6 Acknowledgments

This system would not have been possible without the input and guidance by Ocean 7 Development of New York. <http://www.ocean7.com>.

