TS XML DataServer^{TM}
Reference Manual

THOROUGHBRED
software international, inc

Version 8.7.0

285 Davidson Ave. • Suite 302 • Somerset, NJ 08873
Telephone: 732-560-1377 • Outside NJ 800-524-0430
Fax: 732-560-1594

Internet address: **http://www.tbred.com**

Published by:
Thoroughbred Software International, Inc.
285 Davidson Ave., Suite 302
Somerset, New Jersey 08873-4153

Document Number: TXM8.7.0M002

INTRODUCTION

By using XML (eXtensible Markup Language) and Thoroughbred Dictionary-IV definitions, TS XML DataServer extends your TS Environment to expose and share data residing on a remote server.

With SQL style syntax you can pull data from any Dictionary-IV defined database. The resulting data collection is returned to the requestor as an XML formatted string. The XML formatted data can be rendered as HTML in a browser or shared with other third party tools.

TS XML DataServer also supports data maintenance on a remote server. Writing and executing XML Methods can accomplish this using simple Script-IV like directives to add, change or delete data or.

Concepts

TS XML DataServer is Dictionary-IV centric and assumes a working knowledge of Dictionary-IV. For more information see the Dictionary-IV, and OPENworkshop online reference manuals. For more information on XSchema Dictionary definitions see the Dictionary-IV XSchema Definitions section later in this document

TS XML DataServer takes advantage of Thoroughbred Dictionary-IV using Formats, Links and XSchema definitions to map the XML data collections.

- **Format Class:** A Dictionary-IV class of objects that define a collection of data elements. A Format contains all the attributes and properties for a data element. This is similar to a table definition.

- **Link Class:** A Dictionary-IV class of objects that “link” a database source with a Format. The database source can reference a Thoroughbred data file or a supported server table. The Link XML request provides simple, quick access for flat table mapping. Only the data file or server table defined by the Link is exposed to the XML request. When supplying a Link name in a XML request, the data set is constructed based on the Link definition.

- **XSchema Class:** A Dictionary-IV class of objects used to define a complex relational data mapping across multiple Links. The XSchema identifies all the Links, data elements, the relationships, and all the necessary logic to build a data set. When supplying an XSchema name in an XML request, a data set is constructed based on the design supplied by the XSchema definition.

- **XML Method Class:** A type of Thoroughbred compiled program that runs inside of the Thoroughbred environment on the remote server. Triggered by an XML request, the XML Method receives arguments in the form of an XML string and posts back an XML string to the requestor.

The TS XML DataServer currently supports **READ** *schema*, **READ** *link*, **UPDATE** *link*, **DELETE** *link*, **INSERT** *link* and XML **Methods**.

- **READ** *schema-name* enables you to query and pull data from a remote server using a Thoroughbred Dictionary-IV XSchema definition. The XSchema allows you to define complex relational data mapping across multiple links. The data result of the XSchema is returned in XML standard format.

- **READ** *link-name* enables you to query and pull data from a remote server using standard Thoroughbred selection parameters. **READ** *link-name* will read the specified Link and return the data in XML format.

- **UPDATE** *link-name* enables you to update a record in a Link with data supplied in XML format. All data is validated against the Format definition, I/O Triggers are executed and sorts are maintained.

- **DELETE** *link-name* enables you to delete a record from a Link using a key value supplied in XML format. I/O Triggers are executed and sorts are maintained.

- **INSERT** *link-name* enables you to add a new record to a Link with data supplied in XML format. All data is validated against the Format definition, I/O Triggers are executed and sorts are maintained.

- **METHOD** *method-name* enables you to execute a method on the TS XML DataServer. The method can receive and return XML formatted strings.

The process starts with a request to the XML DataServer. The following is an example XML READ XSchema request:

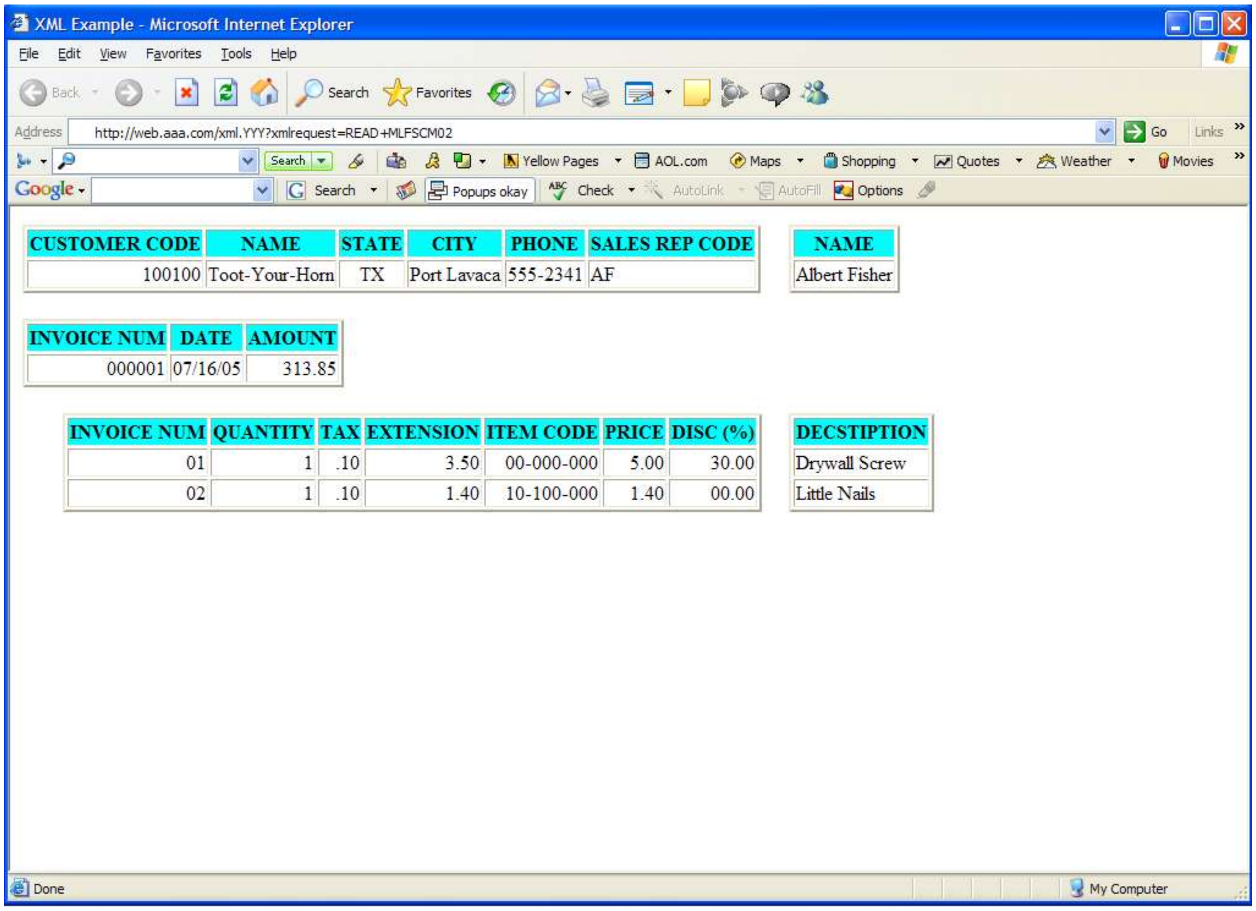
```
http://web.aaa.com/xml.tbred?xmlrequest=READ+OEFSCM02
```

The TS XML DataServer processes the XML request, accesses the data and generates a XML string and returns the string to the requestor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <XMLSOURCE_OEFSCM02>
    <OELCUST>
      <CUST_CODE>100100</CUST_CODE>
      <CUST_NAME>Toot-Your-Horn</CUST_NAME>
      <CUST_STATE>TX</CUST_STATE>
      <CUST_CITY>Port Lavaca</CUST_CITY>
      <CUST_PHONE>555-2341</CUST_PHONE>
      <SR_CODE>AF</SR_CODE>
   </OELCUST>
    <OELSLRP>
      <SR_NAME>Albert Fisher</SR_NAME>
    </OELSLRP>
    <OELINVH>
      <INV_NUM>000001</INV_NUM>
      <INV_DATE>07/16/05</INV_DATE>
      <INV_AMOUNT>313.85</INV_AMOUNT>
    </OELINVH>
    <OELINVD>
      <INV_LNUM>01</INV_LNUM>
      <ITEM_CODE>00-000-000</ITEM_CODE>
      <INV_QTY>1</INV_QTY>
      <INV_LPRICE>5.00</INV_LPRICE>
      <INV_DISC>30.00</INV_DISC>
      <INV_LEXTEN>3.50</INV_LEXTEN>
      <INV_LTAX>.10</INV_LTAX>
    </OELINVD>
    <OELINVN>
      <ITEM_DESC>Drywall Screw</ITEM_DESC>
    <OELINVD>
      <INV_LNUM>02</INV_LNUM>
      <ITEM_CODE>10-100-000</ITEM_CODE>
      <INV_QTY>1</INV_QTY>
      <INV_LPRICE>1.40</INV_LPRICE>
      <INV_DISC>00.00</INV_DISC>
      <INV_LEXTEN>1.40</INV_LEXTEN>
      <INV_LTAX>.10</INV_LTAX>
    </OELINVD>
    <OELINVN>
```

```
        <ITEM_DESC>Little nails</ITEM_DESC>
     </OELINVN>
…etc. (until the end of the detail lines)
   </XMLSOURCE_OEFSCM02>
```

The returned XML string can be rendered or processed by various XML Tools and displayed in your browser.


XML Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://web.aaa.com/xml.YYY?xmlrequest=READ+MLFSCM02
Go Links
Yellow Pages AOL.com Maps Shopping Quotes Weather Movies
Google Search Popups okay Check AutoLink AutoFill Options
Done
My Computer

CUSTOMER CODE	NAME	STATE	CITY	PHONE	SALES REP CODE
100100	Toot-Your-Horn	TX	Port Lavaca	555-2341	AF

NAME
Albert Fisher

INVOICE NUM	DATE	AMOUNT
000001	07/16/05	313.85

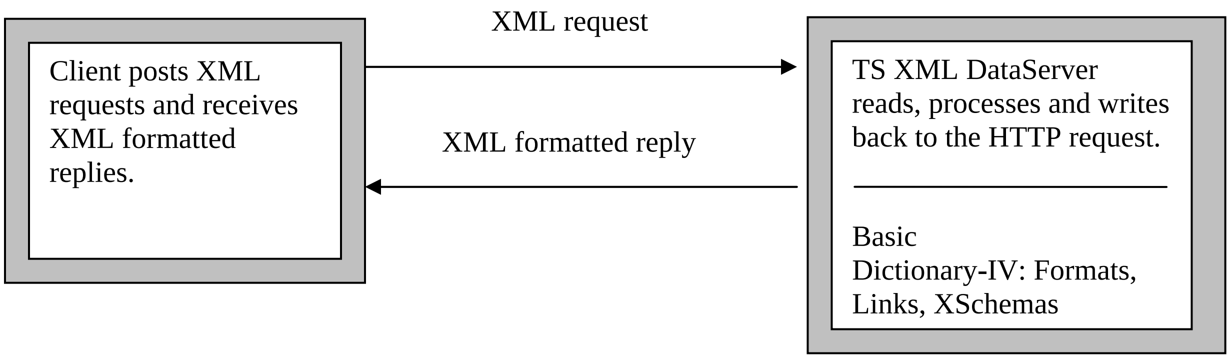
INVOICE NUM	QUANTITY	TAX	EXTENSION	ITEM CODE	PRICE	DISC (%)
01	1	.10	3.50	00-000-000	5.00	30.00
02	1	.10	1.40	10-100-000	1.40	00.00

DECSTIPTION
Drywall Screw
Little Nails

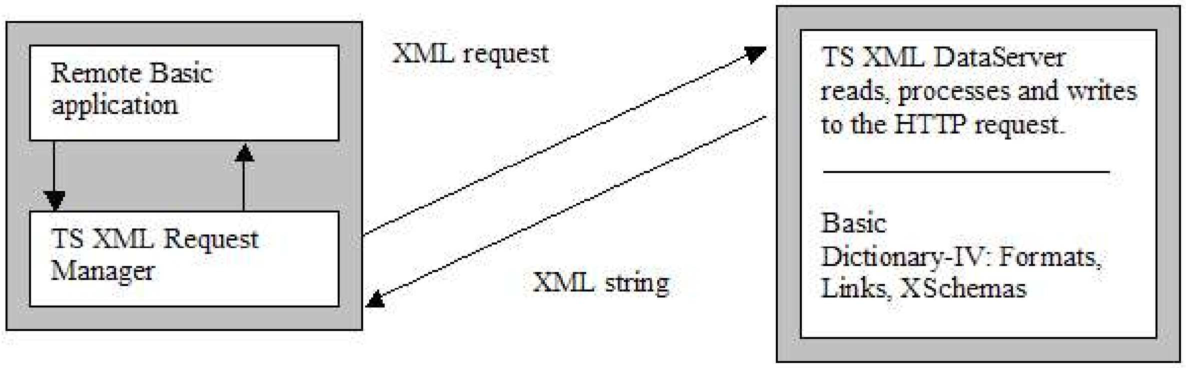
Example only

Architecture

The TS XML server process manages all communication on the server. Requests are received as URL encoded HTTP query strings. The request is passed to the TS XML DataServer engine for parsing and processing. Using the Dictionary-IV Format, Link and XSchema definitions the data collection is built and formatted as an XML string. The XML string is passed back to the server process where it is written back to the HTTP/HTTPS requestor.


XML request
Client posts XML requests and receives XML formatted replies.
XML formatted reply
TS XML DataServer reads, processes and writes back to the HTTP request.
Basic
Dictionary-IV: Formats, Links, XSchemas

The TS XML DataServer also supports XML requests from within a TS Environment by CALLing the TS XML Request Manager. The TS XML Request Manager accepts standard Thoroughbred selection parameters, constructs a properly URL encoded HTTP query string and manages all communication with the server process running on the remote server.


Remote Basic application
TS XML Request Manager
XML request
XML string
TS XML DataServer reads, processes and writes to the HTTP request.
Basic
Dictionary-IV: Formats, Links, XSchemas

Requirements:

The remote server must have the TS XML DataServer components installed and the TS XML server process must be running.

- OOXMLSRV - TS XML DataServer control program
- IPLINPUT.XML - IPLINPUT file for the TS XML DataServer server process
- xml.tbred - TS XML DataServer server control file

- TS XML DataServer server process startup script
- OOXMLSRX, optional TS XML Request Manager installed with a local Basic environment.
- Linux-based Apache Web Server (HTTP Server)

NOTE: If the data resides on the local server and no remote connection is required it is possible to call the TS XML DataServer parser and processing engine directly eliminating the need to route the request over the network through the XML server process.

Information and Organization

SYNTAX	Provides an outline of command or variable structure. Parameters and options are listed and described following the structural outline.
NOTES	Remarks and important information on command or variable syntax and the function of the command or variable.
EXAMPLES	Typical examples of command or variable usage.

Notational Symbols

Boldface	Words in boldface are required syntax. These may be upper or lowercase and should be entered exactly as shown in the syntax.
[optional]	Elements contained in [square brackets] are optional syntactic elements.
Lowercase	Elements shown in lowercase letters identify parameters that need further information or items to be supplied by the user.
. . .	An ellipsis indicates that the preceding element can be repeated.
|	The vertical bar indicates that the user has a choice between two or more elements. At least one of the entries must be chosen unless the complete set of elements is enclosed in square brackets.
"value"	A value enclosed in quotation marks indicates a string value.
Punctuation & symbols	With the exception of the above symbols, all punctuation or relational symbols shown within a command format, such as commas, parentheses, semicolons, equal signs, and so on, are part of the syntax and must be included where shown.

DICTIONARY-IV XSCHEMA DEFINITIONS

The XSchema Dictionary-IV definition class is used to define complex relational data mapping across multiple links.

SYNTAX

```
link-name USING KEY "value";| USING RANGE FROM "value" TO "value";
   [BY SORT sort-number;]
   [DATANAME LIST data-name data-name data-name...;]
                  [data-name, ...;]
   [SELECT WHEN select-clause]
   link-name USING KEY "value";| USING RANGE FROM "value" TO "value";
      [BY SORT sort number;]
      [DATANAME LIST data-name data-name data-name ...;]
      [SELECT WHEN select-clause]
```

link-name	Dictionary-IV Link definition name.
value	the key value or a key range values for reading the Link. **USING KEY** and **USING RANGE** are mutually exclusive.
sort-number	Link sort number to be used when reading with a sort other than SORT 0.
data-name	any data-name defined by the Format.
select-clause	Thoroughbred SELECT WHEN clause.

NOTES

- Spaces are required syntax.
- Each clause must be terminated with a semi-colon (;).
- Keys containing spaces and other special characters must be wrapped in quotes.
- The Links are read in the order defined by the XSchema. This order defines the relationship between data across multiple Link definitions.
- The primary (first) USING clause must specify a literal key value. At the time an XML request is posted, override values for the USING clause can be supplied. For more information on override values, see the XML XSchema Request Overview section later in this document.
- A secondary USING clause can reference #Format.data-names from previously read Links.
- The DATANAME LIST specifies one or more data-names from the Format definition associated with the Link. For example CUST-CODE CUST-NAME CUST-STATE. Only those data elements included in the DATANAME LIST are returned in XML format.

- The SELECT WHEN conditional expression may contain any relational operators supported by the Thoroughbred SELECT WHEN clause. For more information on SELECT WHEN see OPENworkshop online reference manuals.

- The SELECT clause does not support Link data-names; Format data-names must be supplied.

EXAMPLES

Schema: OEFSCM01 Sample System

```
OELCUST USING KEY "100100";
    DATANAME LIST
    CUST-CODE CUST-NAME CUST-STATE CUST-CITY CUST-PHONE SR-CODE
```

Schema: OEFSCM02 Sample System

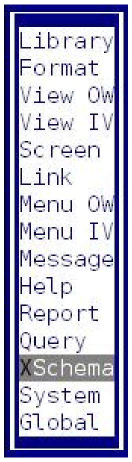
```
OELCUST USING KEY "100100";
    DATANAME LIST CUST-CODE CUST-NAME CUST-STATE CUST-CITY CUST-PHONE
                  SR-CODE

 OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-NAME

 OELINVH USING #OEFCUST.CUST-CODE;
     BY SORT 1;
     DATANAME LIST INV-NUM INV-DATE INV-PRINTED-FLAG INV-AMOUNT

  OELINVD USING #OEFINVH.INV-NUM;
      DATANAME LIST INV-LNUM INV-QTY INV-LTAX INV-LEXTEN ITEM-CODE
                    INV-LPRICE INV-DISC;
      SELECT WHEN #OEFINVD.ITEM-CODE = "10-100-000"

  OELINVN USING #OEFINVD.ITEM-CODE;
      DATANAME LIST ITEM-DESC
```

Schema: OEFSCM03 Sample System

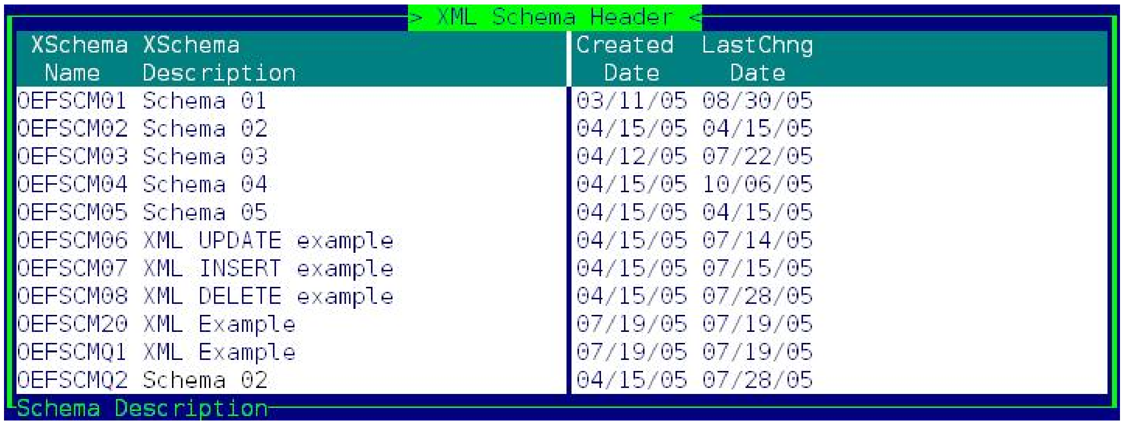
```
OELCUST USING RANGE FROM "000000" TO "999999";
    DATANAME LIST CUST-CODE SR-CODE CUST-NAME CUST-STATE
           CUST-CITY CUST-PHONE;
    SELECT WHEN #OEFCUST.SR-CODE = "HP"
```

Creating and Maintaining XSchema Definitions

From the Dictionary-IV Maintenance Menu select the XSchema option.



The XSchema Definition View will be displayed.

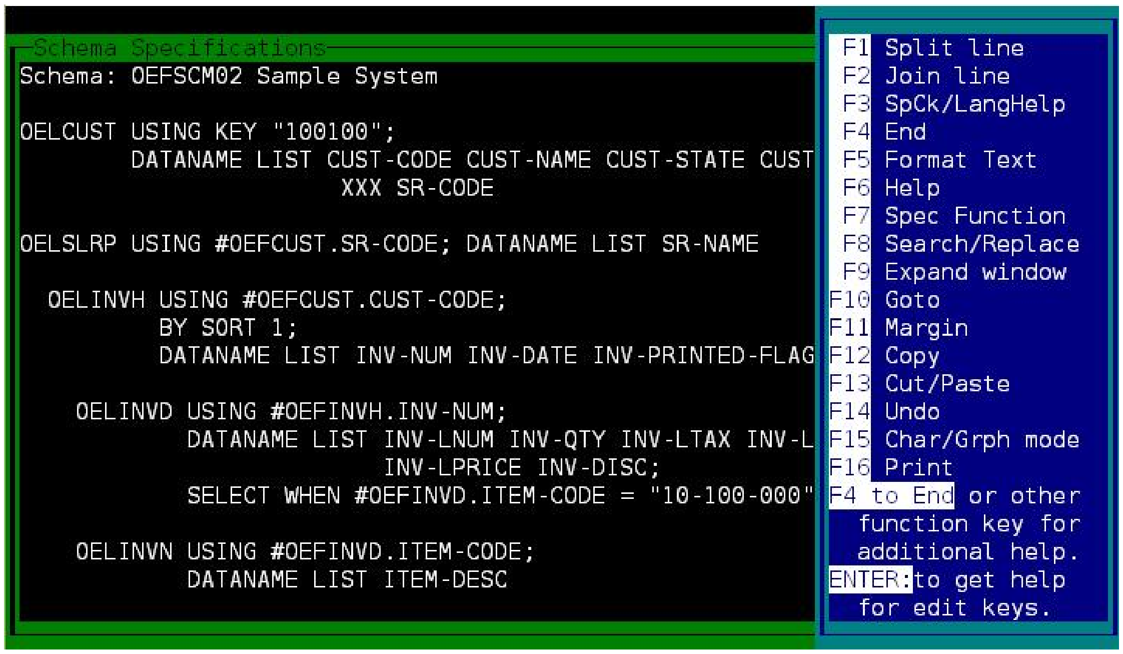


```
> XML Schema Header <
 XSchema  XSchema                           Created   LastChng
  Name    Description                        Date      Date
OEFSCM01 Schema 01                         03/11/05 08/30/05
OEFSCM02 Schema 02                         04/15/05 04/15/05
OEFSCM03 Schema 03                         04/12/05 07/22/05
OEFSCM04 Schema 04                         04/15/05 10/06/05
OEFSCM05 Schema 05                         04/15/05 04/15/05
OEFSCM06 XML UPDATE example                04/15/05 07/14/05
OEFSCM07 XML INSERT example                04/15/05 07/15/05
OEFSCM08 XML DELETE example                04/15/05 07/28/05
OEFSCM20 XML Example                       07/19/05 07/19/05
OEFSCMQ1 XML Example                       07/19/05 07/19/05
OEFSCMQ2 Schema 02                         04/15/05 07/28/05
Schema Description
```

Follow standard Dictionary-IV maintenance procedures for adding, deleting, modifying, renaming, and copying XSchema definitions.

XSchema Editor

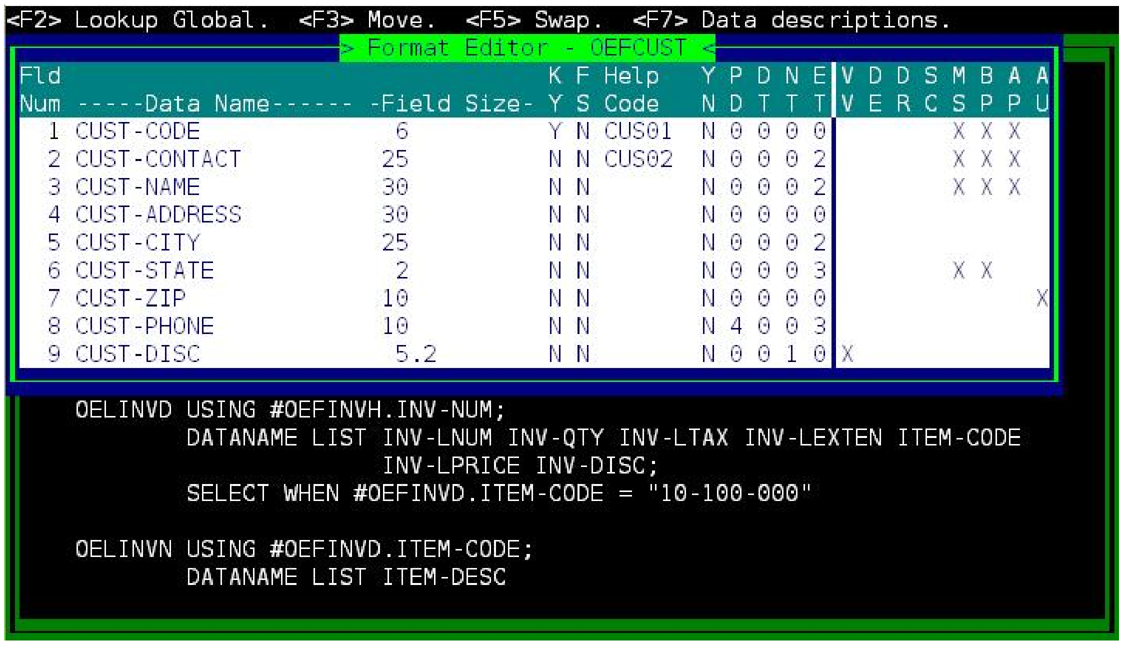
The XSchema definition is displayed in a standard Thoroughbred text editor window. Use standard editing keys to maintain the XSchema definition.



```
-Schema Specifications-
Schema: OEFSCM02 Sample System

OELCUST USING KEY "100100";
        DATANAME LIST CUST-CODE CUST-NAME CUST-STATE CUST
                       XXX SR-CODE

OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-NAME

  OELINVH USING #OEFCUST.CUST-CODE;
          BY SORT 1;
          DATANAME LIST INV-NUM INV-DATE INV-PRINTED-FLAG

    OELINVD USING #OEFINVH.INV-NUM;
            DATANAME LIST INV-LNUM INV-QTY INV-LTAX INV-L
                          INV-LPRICE INV-DISC;
            SELECT WHEN #OEFINVD.ITEM-CODE = "10-100-000"

    OELINVN USING #OEFINVD.ITEM-CODE;
            DATANAME LIST ITEM-DESC
```

```
 F1 Split line
 F2 Join line
 F3 SpCk/LangHelp
 F4 End
 F5 Format Text
 F6 Help
 F7 Spec Function
 F8 Search/Replace
 F9 Expand window
F10 Goto
F11 Margin
F12 Copy
F13 Cut/Paste
F14 Undo
F15 Char/Grph mode
F16 Print
F4 to End or other
  function key for
  additional help.
ENTER:to get help
  for edit keys.
```

F3 – To display Format definition details position the cursor over a #Format-name and press **F3**. If the cursor is not positioned over the #Format-name F3 will display standard language help.



```
<F2> Lookup Global.  <F3> Move.  <F5> Swap.  <F7> Data descriptions.
                   > Format Editor - OEFCUST <
Fld                                K F Help    Y P D N E V D D S M B A A
Num -----Data Name------ -Field Size- Y S Code    N D T T T V E R C S P P U
  1 CUST-CODE              6        Y N CUS01   N 0 0 0 0         X X X
  2 CUST-CONTACT          25        N N CUS02   N 0 0 0 2         X X X
  3 CUST-NAME             30        N N         N 0 0 0 2         X X X
  4 CUST-ADDRESS          30        N N         N 0 0 0 0
  5 CUST-CITY             25        N N         N 0 0 0 2
  6 CUST-STATE             2        N N         N 0 0 0 3         X X
  7 CUST-ZIP              10        N N         N 0 0 0 0               X
  8 CUST-PHONE            10        N N         N 4 0 0 3
  9 CUST-DISC              5.2      N N         N 0 0 1 0 X

    OELINVD USING #OEFINVH.INV-NUM;
            DATANAME LIST INV-LNUM INV-QTY INV-LTAX INV-LEXTEN ITEM-CODE
                          INV-LPRICE INV-DISC;
            SELECT WHEN #OEFINVD.ITEM-CODE = "10-100-000"

    OELINVN USING #OEFINVD.ITEM-CODE;
            DATANAME LIST ITEM-DESC
```

When the XSchema definition is saved it is compiled and is ready to be used with XML requests.

XML SYNTAX

The following is an overview of the XML syntax used by the TS XML DataServer.

XML syntax is used to describe data using simple <tag>*value*</tag> pairs. The tags identify an element of data, its corresponding value, and the relationship of xml-elements within the XML structure.

Unlike HTML syntax, XML syntax rules are strictly enforced:

- Each tag element must terminate in a closing </tag>.
- All tags must be properly nested.
- Tags and their values are case sensitive.

The following xml-element identifies a piece of data as data-name CUST-CODE with a value of 100101:

```
<CUST-CODE>100101</CUST-CODE>
```

NOTE: The "**-**" character within the XML tag is not standard XML syntax but is common to Dictionary-IV Format definitions. The TS XML DataServer will accept the "-" (dash) character in the XML tag and replace all instances with the XML standard "_" (underscore).

Example:

`<CUST-CODE>100101</CUST-CODE>` will be converted to **`<CUST_CODE>100101</CUST_CODE>`**.

The contents of an XML string passed to and returned by the TS XML DataServer will vary depending upon the XML request. For the purpose of this documentation a sample XML string returned by a **READ** *schema-name* is used as an example.

The XML string starts with the XML declaration (header) containing the XML version and encoding. This is the only XML tag that does not include a closing tag.

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

The root XML element contains the string literal "<XMLSource_" followed by the XML source name. The XML source name identifies the Dictionary-IV XSchema or Link definition used to process the XML request.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OEFSCM01>
</XMLSource_OEFSCM01>
```

Nested below the root element is the Link name used to read the data. The Link name identifies the data source and is repeated for each data set (record).

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OEFSCM01>
  <OELCUST>
    <data-name>data</data-name>
    <data-name>data</data-name>
  </OELCUST>
  <OELCUST>
    <data-name>data</data-name>
    <data-name>data</data-name>
  </OELCUST>
</XMLSource_OEFSCM01>

<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OEFSCM02>
  <OELCUST>
    <data-name>data</data-name>
  </OELCUST>
  <OELSLRP>
    <data-name>data</data-name>
  </OELSLRP>
  <OELINVH>
    <data-name>data</data-name>
  </OELINVH>
  <OELINVD>
    <data-name>data</data-name>
  </OELINVD>
  <OELINVN>
    <data-name>data</data-name>
  </OELINVN>
</XMLSource_OEFSCM02>
```

The data-names are the Format data-element names associated with the Link definition.

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OEFSCM01>
  <OELCUST>
    <CUST-CODE>data</CUST-CODE>
    <CUST-NAME>data</CUST-NAME>
  </OELCUST>
  <OELCUST>
     <CUST-CODE>data</CUST-CODE>
     <CUST-NAME>data</CUST-NAME>
  </OELCUST>
</XMLSource_OEFSCM01>
```

The following XML string is returned when a XML READ XSchema definition for OEFSCM02 is completed.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <XMLSource_OEFSCM02>
    <OELCUST>
      <CUST_CODE>100100</CUST_CODE>
      <CUST_NAME>Toot-Your-Horn</CUST_NAME>
    <CUST_STATE>TX</CUST_STATE>
    <CUST_CITY>Port Lavaca</CUST_CITY>
    <CUST_PHONE>555-2341</CUST_PHONE>
    <SR_CODE>AF</SR_CODE>
  <OELSLRP>
    <SR_NAME>Albert Fisher</SR_NAME>
  </OELSLRP>
  <OELINVH>
    <INV_NUM>000001</INV_NUM>
    <INV_DATE>07/16/05</INV_DATE>
    <INV_AMOUNT>313.85</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
    <INV_LNUM>02</INV_LNUM>
    <INV_QTY>1</INV_QTY>
    <INV_LTAX>.10</INV_LTAX>
    <INV_LEXTEN>1.40</INV_LEXTEN>
    <ITEM_CODE>10-100-000</ITEM_CODE>
    <INV_LPRICE>1.40</INV_LPRICE>
    <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
     <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
  <OELINVH>
     <INV_NUM>000016</INV_NUM>
     <INV_DATE>07/31/05</INV_DATE>
     <INV_AMOUNT>5021.29</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
    <INV_LNUM>02</INV_LNUM>
    <INV_QTY>16</INV_QTY>
    <INV_LTAX>1.57</INV_LTAX>
    <INV_LEXTEN>22.40</INV_LEXTEN>
    <ITEM_CODE>10-100-000</ITEM_CODE>
    <INV_LPRICE>1.40</INV_LPRICE>
    <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
    <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
  <OELINVH>
    <INV_NUM>000031</INV_NUM>
    <INV_DATE>08/15/05</INV_DATE>
    <INV_AMOUNT>9728.78</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
    <INV_LNUM>02</INV_LNUM>
    <INV_QTY>31</INV_QTY>
    <INV_LTAX>3.04</INV_LTAX>
    <INV_LEXTEN>43.40</INV_LEXTEN>
    <ITEM_CODE>10-100-000</ITEM_CODE>
    <INV_LPRICE>1.40</INV_LPRICE>
    <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
    <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
</XMLSource_OEFSCM02>
```

SETTINGS

There are several settings that control how TS XML DataServer will manage empty data fields and special characters embedded in data. These can be set in the format definition OOFXMLSV, the XML Sever Global. To access these settings load format OOFXMLSV in Dictionary-IV Format Maintenance then modify the preset (default) setting for the appropriate data-element. From the Dictionary-IV Format Maintenance menu select **Format**. Scroll down (or F10) to find **OOFXMLSV** (XML Server Global). Press **F1** to edit the format. Edit the "Preset Value" (**DE**fault) field in the following data elements according to the descriptions below.

XML-EMPTY-ENABLED

Use this data element to control how TS XML DataServer manages empty data fields.

> **" " or "N"** any empty data fields will not be included in the XML string. This is the default preset value.

> **"Y"** returns empty data fields in the XML string using the data element name as the XML tag. For example if CUST-NAME was empty the XML string would contain <CUST_NAME></CUST_NAME>.

XML-SPECIAL-CHAR

Use this data element to control how TS XML DataServer manages special characters.

> **" "** Does not URL encode special characters.

> **"U"** URL encodes special characters. Special character list is defined as:

Sepcial Character	Replaced with Code
- (dash)	_ (underscore)
<	&lt
>	&gt
&	&amp
'	&apos
"	&quot

> **"R"** applies the search and replace. This requires that the list of special characters be defined in XML-SPECIAL-SEARCH preset and the replacement values be defined in the XML-SPECIAL-REPLACE preset.

XML-SPECIAL-SEARCH

Use this data element to define the list of special characters that will be searched for and replaced with XML-SPECIAL-REPLACE values.

Enter a Preset Value containing from 1 to 25 characters. Do not use any separators.

See XML-SPECIAL-REPLACE for more information.

XML-SPECIAL-REPLACE

Use this data element to define the list of replacement characters for XML-SPECIAL-SEARCH.

Enter a Preset Value (DE) containing from 1 to 25 characters. Do not use any separators.

For each special character defined in XML-SPECIAL-SEARCH there must be a corresponding replacement character defined in XML-SPECIAL-REPLACE.

Examples of XML-SPECIAL-SEARCH/ XML-SPECIAL-REPLACE

Set the #OOFXMLSV.XML-SPECIAL-CHAR Preset Value to "R" to enable search and replace processing.

- To replace all instances of the & character with the _ character:

  Set XML-SPECIAL-SEARCH Preset Value to &

  Set XML-SPECIAL-REPLACE Preset Value to _

- To replace all instances of the & character with the _ character and all instances of = with the x character:

  Set XML-SPECIAL-SEARCH preset value to &=

  Set XML-SPECIAL-REPLACE preset value to _x

- To replace all instances of the &<>'="> characters with _ character:

  Set XML-SPECIAL-SEARCH preset value to &<>'="

  Set XML-SPECIAL-REPLACE preset value to ______ (6 underscore characters)

DICTIONARY-IV LINK REQUESTS – OVERVIEW

READ

The READ directive will read the specified Link and return the data in XML format.

SYNTAX:

```
READ link-name USING KEY “value”; | USING RANGE FROM “value” TO “value”;
   [BY SORT sort-number;]
   [DATANAME LIST data-name data-name data-name...;]
   [SELECT WHEN select-clause]
```

link-name	Dictionary-IV Link definition name.
value	the key value or a key range values for reading the Link. **USING KEY** and **USING RANGE** are mutually exclusive.
sort-number	the Link sort number to be used when reading with a sort other than SORT 0.
data-name	any data-name defined by the Format.
select-clause	any Thoroughbred SELECT WHEN clause.

NOTES

- The USING KEY clause and USING RANGE clause only supports literal key values, #format-name.data-name is not supported. All key values must be properly padded and wrapped in quotes.
- The DATANAME LIST specifies one or more valid data-names from the Format definition associated with the Link. For example CUST-CODE CUST-NAME CUST-STATE. Only those data elements included in the DATANAME LIST are returned in XML format. A DATANAME LIST is optional. If a DATANAME LIST is not supplied, a default DATANAME LIST will be built using all the data names defined by the Format.
- The SELECT WHEN conditional expression may contain any relational operators supported by the Thoroughbred SELECT WHEN clause. For more information on SELECT WHEN see OPENworkshop online reference manuals.
- The SELECT clause does not support Link data-names; Format data-names must be supplied.

EXAMPLES

```
READ OELCUST USING KEY "100101"
```

```
READ OELCUST USING RANGE FROM "100101" TO "100104"
```

```
READ OELCUST USING KEY "Computer Inc.           ";
BY SORT 1
```

```
OELCUST USING RANGE FROM "100100" TO "100999";
DATANAME LIST CUST-CODE CUST-NAME SR-CODE;
SELECT WHEN #OEFCUST.SR-CODE = "HP"

OELCUST USING RANGE FROM "100100" TO "100999";
DATANAME LIST CUST-CODE CUST-NAME SR-CODE;
SELECT WHEN #OEFCUST.CUST-CONTACT LIKE "*Smith*"

OELCUST USING RANGE FROM "100100" TO "100999";
DATANAME LIST CUST-CODE CUST-NAME SR-CODE;
SELECT WHEN #OEFCUST.CUST-SALES > 15000
```

UPDATE

The UPDATE directive updates existing records in the data source defined by a Link. The UPDATE clause specifies the Link name, key value, and the data.

SYNTAX

`UPDATE` *`link-name <xml-elements>`*

`link-name`	Dictionary-IV Link definition name.
`<xml-elements>`	the supplied data for the update in XML format.

NOTES

- The first set of xml-elements defines the key to the Link. The key is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.
- Before the Link is updated the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW name.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.
- If the Link contains sort definitions all sorts will be updated.

EXAMPLE

```
UPDATE OELSLRP <SR-CODE>XX</SR-CODE><SR-NAME>Justin Case</SR-NAME>
```

Changes the SR-NAME for SR-CODE "XX" in the data file for the Link (OELSLRP).

INSERT

The INSERT directive adds a new record to the data source defined by a Link. The INSERT clause specifies the Link name, key value, and the data for the new record.

SYNTAX

`INSERT` *`link-name <xml-elements>`*

`link-name`	Dictionary-IV Link definition name.
`<xml-elements>`	the supplied data for the insert in XML format.

NOTES

- The first set of xml-elements defines the key to the Link. The key is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.
- Before the new record is added to the Link the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW name.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.
- If the Link contains sort definitions all sorts will be updated.

EXAMPLE

```
INSERT OELSLRP <SR-CODE>XX</SR-CODE><SR-NAME>JustinCase</SR-NAME>
<SR-SALES>0</SR-SALES>
```

Adds a new record to the data file for the Sales Rep Link (OELSLRP).

DELETE

The DELETE directive deletes a record from the data source defined by a Link. The delete clause specifies the Link name and the key value. The supplied key must be in XML format. All xml-elements are assumed to be key fields.

SYNTAX

```
DELETE link-name <xml-elements>
```

`link-name`	Dictionary-IV Link definition name.
`<xml-elements>`	the supplied data for the insert in XML format.

NOTES

- If the Link contains an I/O Trigger, the trigger will be executed prior to the REMOVE.

EXAMPLE

```
DELETE OELSLRP <SR-CODE>XX</SR-CODE>
```

Removes Sales Rep "XX" from the data file for the Link (OELSLRP).

DICTIONARY-IV XSCHEMA REQUESTS – OVERVIEW

READ

The READ directive specifies the XSchema definition to be used to process the XML request.

SYNTAX

```
READ XSchema-name

XSchema-name      the Dictionary-IV XSchema definition name.
```

EXAMPLE

```
READ OEFSCM01
```

Returns the data collection defined by OEFSCM01 in XML format.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OEFSCM01>
  <OELCUST>
    <Customer code>100100</Customer code>
    <Customer name>Toot-Your-Horn</Customer name>
    <Customer state>TX</Customer state>
    <Customer city>Port Lavaca</Customer city>
    <Customer phone>555-2341</Customer phone>
    <Sales representative>AF</Sales representative>
  </OELCUST>
</XMLSource_OEFSCM01>
```

Override Values

When the XML XSchema request is made, the requestor can supply override values for the primary Link **USING** *clause*. This technique allows a single XSchema definition to be reused with varying key values.

For more information on USING syntax please refer to the Dictionary-IV Link Requests – Overview section.

EXAMPLE

```
OELCUST USING KEY "100100";
        DATANAME LIST CUST-CODE CUST-NAME SR-CODE

OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-NAME
```

The XSchema OEFSCM99 is defined to return a collection of data for Customer Code “100100”.

```
READ OEFSCM99 LINK=OELCUST USING KEY "100102"
```

An XML request with override values can be posted to return that same set of data for Customer Code “100102”:

The XSchema definition does not have to contain a valid key value if override values are used. The key value in the XSchema definition can simply function as a placeholder. In this case it is critical that an override value is always supplied when a XML request is posted referencing the XSchema definition.

The following is a valid XSchema definition:

```
OELCUST USING KEY "X";
        DATANAME LIST CUST-CODE CUST-NAME SR-CODE

OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-NAME
```

The above will fail to return any results unless a valid **USING** *clause* is supplied as an override value.

```
READ OEFSCM99 LINK=OELCUST USING KEY "100102"
```

XML REQUEST QUERY STRINGS

Overview

XML requests are posted as URL encoded HTTP query strings to the TS XML DataServer server process. All replies are posted back to the requestor in XML format.

SYNTAX

`http|https:``//`*`url_address/`***`xml.tbred?`**`[`**`User`**`=`*`user_id`*`][`**`&Pwd`**`=`*`password`*`]`
`&xmlrequest=`*`argument`*

url_address	is the url address of the server running the Thoroughbred XML server process.
user_id	is the optional user id for authentication.
password	is the optional password for authentication.
argument	is the XML request string. The contents vary depending upon the request type. See below for details.

NOTES

- Both http and https are supported.
- xml.tbred is the name of the server process on the TS XML DataServer.
- Multiple variables must be separated with the "&" character.
- To enable authentication the xml.tbred file must contain the following global variable definitions:
  `Authenticate="Y" Authenticate_prog=`*`program_name`*
  Authenticate must be set to **"Y"**

  Authenticate_prog specifies the name of the program to be called to authenticate the user. This program will be executed prior to executing the request supplied in the *xmlrequest* variable. If authentication fails, the *xmlrequest* will not be processed.

EXAMPLE

`http:``//tweb.tbred.com/`**`xml.tbred?xmlrequest=READ+`**`OELINVH`**`+USING+KEY+`**`%22100101%22`

Reads Link OELCUST for Customer Code 100101.

`https:``//tweb.tbred.com/`**`xml.tbred?xmlrequest=READ`**`+OELINVH+`**`USING+KEY`**`+%22100101%22`

Reads Link OELCUST for Customer Code 100101 over a SSL connection.

`http:``//tweb.tbred.com/`**`xml.tbred?`**
User=tsi**&Pwd**=tsi**&**`xmlrequest=`**`READ`**`+OELINVH+`**`USING+KEY`**`+%22100101%22`

Reads Link OELCUST for Customer Code 100101 after authenticating the user:

READ Link Query Strings

The TS XML DataServer will attempt to read the definition as an XSchema. If a matching XSchema definition is found the request will be processed as an XSchema. If a matching XSchema is not found it will be processed as a Link. If the Link is not found an empty XML string will be returned.

SYNTAX

`xmlrequest=`*`argument`*

`argument` contains the READ link-name and read request parameters.

NOTES

- The Thoroughbred key words normally separated with a space must be separated with the "+" character.

  Example:

  `READ` `OELCUST` **`USING KEY`** must be formatted as
  `READ``+OELCUST+`**`USING+KEY`**

- The query string must be URL encoded. Example:

  `READ` `OLECUST` **`USING KEY`** `"100101"` must be formatted as
  `READ``+OELCUST+`**`USING+KEY`**`+%22100101%22`

- It is required that key values be wrapped in quotes if the key contains embedded spaces or other special characters. Example:

  `key value of "123 AB"` must be encoded as
  `%22123%20%20AB%20%22`

EXAMPLES

Example 1

`http://``tweb.tbred.com`**`/xml.tbred?xmlrequest=READ`**`+OELCUST+`**`USING+KEY`**`+%22100101%22`

Posts a request to the remote sever to read Link OELCUST for Cust-Code 100101. The data will be posted back in XML format.

The requestor will receive:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100101</CUST_CODE>
    <CUST_CONTACT>David Kelly</CUST_CONTACT>
    <CUST_NAME>Fix-M-Up</CUST_NAME>
    <CUST_ADDRESS>Old Bay Shore Road</CUST_ADDRESS>
    <CUST_CITY>Seadrift</CUST_CITY>
    <CUST_STATE>TX</CUST_STATE>
    <CUST_ZIP>50250</CUST_ZIP>
    <CUST_PHONE>512 876-9070</CUST_PHONE>
    <CUST_DISC>32.00</CUST_DISC>
    <CUST_CRLMT>5010</CUST_CRLMT>
    <CUST_STC>TX</CUST_STC>
    <SR_CODE>HP</SR_CODE>
    <CUST_SALES>14530.92</CUST_SALES>
  </OELCUST>
</XMLSource_OELCUST>
```

Example 2

http://tweb.tbred.com**/xml.tbred?xmlrequest=READ**+OELINVH+**USING+KEY**+%22100101%22+;
BY+SORT+1

Posts a request to the remote sever to read Link OELINVH using Sort 1 for Cust-Code 100101.

The requestor will receive:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELINVH>
  <OELINVH>
    <INV_NUM>000002</INV_NUM>
    <INV_DATE>07/17/05</INV_DATE>
    <INV_AMOUNT>609.72</INV_AMOUNT>
    <CUST_CODE>100101</CUST_CODE>
    <SR_CODE>HP</SR_CODE>
  </OELINVH>
  <OELINVH>
    <INV_NUM>000017</INV_NUM>
    <INV_DATE>08/01/05</INV_DATE>
    <INV_AMOUNT>5182.71</INV_AMOUNT>
    <CUST_CODE>100101</CUST_CODE>
    <SR_CODE>HP</SR_CODE>
  </OELINVH>
  <OELINVH>
    <INV_NUM>000032</INV_NUM>
    <INV_DATE>08/16/05</INV_DATE>
    <INV_AMOUNT>9755.70</INV_AMOUNT>
    <CUST_CODE>100101</CUST_CODE>
    <SR_CODE>HP</SR_CODE>
  </OELINVH>
</XMLSource_OELINVH>
```

Example 3

```
http://tweb.tbred.com/xml.tbred?xmlrequest=READ+OELCUST+
USING+RANGE+FROM+100101+TO+999999;DATANAME+LIST+CUST-CODE;
SELECT+WHEN+%23OEFCUST.SR-CODE=%22HP%22
```

The HTTP request has been split for readability.

Reads Link OELCUST using a range of keys from 100101 through 999999 and return the Cust-Code data element for those records where the SR-CODE = "HP".

NOTE: The ";" is required syntax separating multiple directives. The SELECT WHEN clause requires a #format.data-name. The "#" character must be url encoded.

The requestor will receive:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100101</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100102</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100104</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100105</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>101103</CUST_CODE>
  </OELCUST>
 </XMLSource_OELCUST>
```

Example 4

```
http://tweb.tbred.com/xml.tbred?xmlrequest=READ+OELCUST+
USING+RANGE+FROM+100101+TO+999999;
SELECT+WHEN+%23OEFCUST.CUST-CONTACT+LIKE+%22%2ASmith%2A%22
```

The HTTP request has been split for readability.

Reads Link OELCUST using a range of keys from 100101 through 999999 and return records where the CUST-NAME contains the characters "Smith".

The requestor will receive:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100104</CUST_CODE>
    <CUST_CONTACT>Sarah Smith</CUST_CONTACT>
    <CUST_NAME>ACME Inc.</CUST_NAME>
    <CUST_ADDRESS>346 Elm Street</CUST_ADDRESS>
    <CUST_CITY>Dayton</CUST_CITY>
    <CUST_STATE>NJ</CUST_STATE>
    <CUST_ZIP>42523</CUST_ZIP>
    <CUST_PHONE>608 564-3200</CUST_PHONE>
    <CUST_DISC>38.00</CUST_DISC>
    <CUST_CRLMT>8010</CUST_CRLMT>
    <CUST_STC>NJ</CUST_STC>
    <SR_CODE>HP</SR_CODE>
    <CUST_SALES>14807.46</CUST_SALES>
  </OELCUST>
</XMLSource_OELCUST>
```

Example 5

```
http://tweb.tbred.com/xml.tbred?xmlrequest=READ+OELCUST+
USING+RANGE+FROM+100101+TO+999999;
DATANAME+LIST+CUST-CODE;
SELECT+WHEN+%23OEFCUST.CUST-SALES+%3E+15000
```

The HTTP request has been split for readability.

Reads Link OELCUST using a range of keys from 100101 through 999999 and return records where the CUST-SALES are > 1500. Only the CUST-CODE is returned in the XML string.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100105</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100106</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100107</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100108</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100109</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100110</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>101104</CUST_CODE>
  </OELCUST>
</XMLSource_OELCUST>
```

UPDATE Link Query Strings

SYNTAX

`UPDATE``+`*`name`*`+`*`xml-elements`*

`name`	is the Link name, XSchema names cannot be used with the UPDATE directive.
`xml-elements`	is the xml string containing the key values and the data to update. The first set of xml-elements defines the key to the Link. The key value is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.

NOTES

- Before the Link is updated the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.
- If the Link contains sort definitions all sorts will be updated.
- A successful UPDATE XML reply will contain the "." character indicating the process completed successfully:

  <standard XML header>
  <link name>
          <UPDATE>.</UPDATE>
  </link name>

  Example:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <OELCUST>
        <UPDATE>.</UPDATE>
  </OELCUST>
  ```

- If an error occurs the XML reply will contain an error message:

  <standard XML header>
  <error>*error-message*</error>

  Example:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <ERROR>1) 102   : Can't find record for specified key  2) </ERROR>
  ```

EXAMPLE

```
http://tweb.tbred.com/xml.tbred?xmlrequest=UPDATE+OELCUST+<CUST-
CODE>100104</CUST-CODE><SR-CODE>HP</SR-CODE>
```

UPDATEs Link OELCUST using a key value of 100104, changing the SR-CODE to "HP".

INSERT Link Query Strings

SYNTAX

`INSERT`+*name*+*xml-elements*

name	is the Link name.
xml-elements	is the xml string containing the key values and the data to added. The first set of xml-elements defines the key to the Link. The key is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.

NOTES

- Before the Link is updated the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.
- If the Link contains sort definitions all sorts will be updated.
- A successful INSERT XML reply will contain the "." character indicating the process completed successfully:

  ```
  <standard XML header>
  <link name>
      <INSERT>.</INSERT>
  </link name>
  ```

  Example:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <OELCUST>
     <INSERT>.</INSERT>
  </OELCUST>
  ```

- If an error occurs the XML reply will contain an error message:

  <standard XML header>
  <error>*error-message*</error>

EXAMPLE

```
http://tweb.tbred.com/xml.tbred?xmlrequest=INSERT+OELSLRP+<SR-CODE>XX</SR-CODE><SR-NAME>JustinCase</SR-NAME><SR-SALES>0</SR-SALES>
```

WRITEs a new record to Link OELSLRP using a key value of "XX" for the new SR-CODE and "Justin Case" for SR-NAME and "0" for SR-SALES.

DELETE Link Query Strings

SYNTAX

`DELETE``+`*`name`*`+`*`xml-elements`*

`name`	is the Link name.
`xml-elements`	is the xml string containing the key values of the record to be deleted.

NOTES

- All xml-elements are assumed to be key fields.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the REMOVE.
- A successful DELETE XML reply will contain the "." character indicating the process completed successfully:

  <standard XML header>
  <link name>
      <DELETE>.</DELETE>
  </link name>

  Example:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <OELCUST>
     <DELETE>.</DELETE>
  </OELCUST>
  ```

- If an error occurs the XML reply will contain an error message:

  <standard XML header>
  <error>*error-message*</error>

EXAMPLE

`http://``tweb.tbred.com`**`/xml.tbred?xmlrequest=DELETE`**`+OELCUST+<CUST-CODE>100104</CUST-CODE>`

REMOVEs a record from Link OELCUST using key 100104

Read XSchema Query Strings

The TS XML DataServer will attempt to read the definition as an XSchema. If a matching XSchema definition is found the request will be processed as an XSchema. If a matching XSchema is not found it will be processed as a Link. If the Link is not found an empty XML string will be returned.

SYNTAX

`xmlrequest=`*`argument`*

`argument`	contains the READ schema-name.

NOTES

- The Thoroughbred key words normally separated with a space must be separated with the "+" character.

  Example:

  ```
  READ OEFSCM02 must be formatted as
  READ+OEFSCM02
  ```

EXAMPLES

Schema: OEFSCM02 Sample System

```
OELCUST USING KEY "100100";
        DATANAME LIST CUST-CODE CUST-NAME CUST-STATE CUST-CITY
                      CUST-PHONE XXX SR-CODE

OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-CODE
   OELINVH USING #OEFCUST.CUST SR-CODE;
           BY SORT 1;
           DATANAME LIST INV-NUM INV-DATE INV-PRINTED-FLAG
                         INV-AMOUNT
      OELINVD USING #OEFINVH.INV-NUM
              DATANAME LIST INV-LNUM INV-QTY INV-LTAX INV-LEXTEN
                            ITEM-CODE INV-LPRICE INV-DISC;
              SELECT WHEN #OEFINVD.ITEM-CODE = "10-100-000”

      OELINVN USING #OEFINVD.ITEMCODE;
              DATANAME LIST ITEM-DESC
```

Reads Links OELCUST, OELSLRP, OELINVH, and OELINVH populating the specified data elements and returns the data structure in XML format.

```
http://tweb.tbred.com/xml.tbred?xmlrequest=READ+OEFSCM02
```

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
 <XMLSource_OEFSCM02>
  <OELCUST>
   <CUST_CODE>100100</CUST_CODE>
   <CUST_NAME>Toot-Your-Horn</CUST_NAME>
   <CUST_STATE>TX</CUST_STATE>
   <CUST_CITY>Port Lavaca</CUST_CITY>
   <CUST_PHONE>555-2341</CUST_PHONE>
   <SR_CODE>AF</SR_CODE>
  </OELCUST>
  <OELSLRP>
   <SR_NAME>Albert Fisher</SR_NAME>
  </OELSLRP>
  <OELINVH>
   <INV_NUM>000001</INV_NUM>
   <INV_DATE>07/16/05</INV_DATE>
   <INV_AMOUNT>313.85</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
   <INV_LNUM>02</INV_LNUM>
   <INV_QTY>1</INV_QTY>
   <INV_LTAX>.10</INV_LTAX>
   <INV_LEXTEN>1.40</INV_LEXTEN>
   <ITEM_CODE>10-100-000</ITEM_CODE>
   <INV_LPRICE>1.40</INV_LPRICE>
   <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
    <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
  <OELINVH>
    <INV_NUM>000016</INV_NUM>
    <INV_DATE>07/31/05</INV_DATE>
    <INV_AMOUNT>5021.29</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
   <INV_LNUM>02</INV_LNUM>
   <INV_QTY>16</INV_QTY>
   <INV_LTAX>1.57</INV_LTAX>
   <INV_LEXTEN>22.40</INV_LEXTEN>
   <ITEM_CODE>10-100-000</ITEM_CODE>
   <INV_LPRICE>1.40</INV_LPRICE>
   <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
   <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
  <OELINVH>
   <INV_NUM>000031</INV_NUM>
```

```
    <INV_DATE>08/15/05</INV_DATE>
    <INV_AMOUNT>9728.78</INV_AMOUNT>
  </OELINVH>
  <OELINVD>
    <INV_LNUM>02</INV_LNUM>
    <INV_QTY>31</INV_QTY>
    <INV_LTAX>3.04</INV_LTAX>
    <INV_LEXTEN>43.40</INV_LEXTEN>
    <ITEM_CODE>10-100-000</ITEM_CODE>
    <INV_LPRICE>1.40</INV_LPRICE>
    <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
    <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
</XMLSource_OEFSCM02>
```

Method Query Strings

SYNTAX

`http|https:``//`*`url_address`*`/`**`xml.tbred?xmlrequest=METHOD`**`+`*`name`*`+[<msg>`*`optional_method_msg`*`</msg>]`

url_address	is the url address of the server running the Thoroughbred XML server process.
name	is the name of the method to be executed on the remote server.
optional_method_msg	is an optional message that can is passed to the method. The method message must be url encoded and wrapped in xml tags.

NOTES

- The message returned by the XML Method must start with <method> and must terminate with </method>

  <method>*my-method-message-here*</method>

EXAMPLE

Example 1

`http:``//tweb.tbred.com/`**`xml.tbred?xmlrequest=METHOD+`**`WBQXML02`

Executes method WBQXML02 on the remote server. No message is supplied to the method.

Example 2

`http:``//tweb.tbred.com/`**`xml.tbred?xmlrequest=METHOD+`**`WBQXML02+%3Cmsg%3Ethis%20is%20an%20example%3C/msg%3E`

Executes method WBXML02 on the remote server passing the message "this is an example". The method WBXML02 will then parse the message and construct a reply that is posted back to the requestor.

Method WBXML02

```
ENTER MSG1$[ALL],XML$[ALL];
PROCEDURE
      MSG$=MSG1$[1];
      P=POS(">"=MSG$),
      MSG$=MSG$(P+1),
      P=POS("</"=MSG$),
      MSG$=MSG$(1,P-1),
      XML$[1]="<method>WBQXML02 received: "+MSG$+"</method>";
```

The method WBXML02 is executed. When EXIT is executed the contents of XML$[1] is posted back to the requestor.

The requestor receives the following message from the method WBXML02:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<method>WBQXML02 received: this is an example</method>
```

XML REQUEST MANAGER

Overview

The XML Request Manager provides easy to use syntax for making XML requests with a local Basic installation. The method avoids the need to URL encode the XML request and manages all communication with the TS XML server process. Requests are made by CALLing the TS XML Request Manager OOXMLSRX directly from a Basic program.

```
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL]

REQ$[1] Function:
        "e": Before posting to the remote server, url encode the query
        string. This is the default.

        note: When REQ$[1]="" the default function "e" will be applied

REQ$[2] The url address of the server running the Thoroughbred XML
        server process.

        Example: REQ$[2]="tweb.tbred.com"

REQ$[3] reserved for internal use

REQ$[4] reserved for internal use

REQ$[5] Optional time out value
        Example:  REQ$[5]="10"

REQ$[6] XMLRequest
        The XMLRequest varies depending upon the request type; READ
        (Link/XSchema), UPDATE, INSERT, DELETE, and METHOD.
        The XMLRequest strings are documented below.

REQ$[7] &User=user_id&Pwd=password
        User: Optional user id for authentication.
        Pwd: Optional password for authenticaiton.
```

NOTES

- Multiple variables must be separated with the "&" character. For example: &User=tbred&Pwd=tbred
- XML$[ALL] Contains the XML formatted response from the XML server.

- To enable authentication the xml.tbred file must contain the following global variable definitions:

  ```
  Authenticate="Y"
  Authenticate_prog=program_name
  ```

  Authenticate must be set to **`"Y"`**.
  Authenticate_prog specifies the name of the program to be called to authenticate the user. This program will be executed prior to executing the request. If authentication fails, the request will not be processed.

READ Link XML Request Manager

The TS XML DataServer will attempt to read the definition as an XSchema. If a matching XSchema definition is found the request will be processed as an XSchema. If a matching XSchema is not found it will be processed as a Link. If the Link is not found an empty XML string will be returned.

SYNTAX

`REQ$[6]=READ` *`link-name arguments`*

`link-name`	is the name of the Link definition
`arguments`	are the READ arguments: USING KEY *value*, USING RANGE FROM *value* TO *value*, BY SORT *n*, DATANAME LIST *data-names*, and the SELECT WHEN *conditional*.

NOTES

- XMLRequests do not have to be URL encoded; the XML Request Manager will construct a proper URL encoded query string.

  Examples:

  ```
  Read Link OELCUST for CUST-CODE 100101 and return all data in XML format:
  REQ$[6]="READ OELCUST USING KEY 100101"

  Read OELCUST for a range of records and return only the CUST-CODEs in XML
  format:
  REQ$[6]="READ OELCUST USING RANGE FROM 100000 TO 999999;DATANAME LIST
  CUST-CODE"

  Read OELCUST by Sort 3 for CUST-STATE NJ and return only the CUST-CODEs in
  XML format:
  REQ$[6]="READ OELCUST USING KEY NJ;BY SORT 3;DATANAME LIST CUST-CODE"

  Read OELCUST for a range of records and only return XML data where SR-CODE
  ="HP":
  REQ$[6]="READ OELCUST USING RANGE FROM 100101 TO 999999; DATANAME LIST
  CUST-CODE;SELECT WHEN #OEFCUST.SR-CODE LIKE"+QUO+"A*"+QUO
  ```

EXAMPLES

On the local server the CALL is made to OOXMLSRX. On the remote server, the Link will be read using the supplied arguments and the data returned to the calling program in XML format.

In the following example the XML string will contain all the data elements for customer code 100101.

```
DIM REQ$[6], XML$[1];
* set the server address:
REQ$[2]="tweb.tbred.com",
* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="READ OELCUST USING KEY 100101";

* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];

* if no errors, save the xml string
IF REQ$[0]="."
   XML$=XML$[1]
FI
```

In the above example XML$[1] will contain:

```
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100100</CUST_CODE>
    <CUST_NAME>Toot-Your-Horn</CUST_NAME>
    <CUST_STATE>TX</CUST_STATE>
    <CUST_CITY>Port Lavaca</CUST_CITY>
    <CUST_PHONE>555-2341</CUST_PHONE>
    <SR_CODE>AF</SR_CODE>
 </OELCUST>
</XMLSource_OELCUST>
```

In the following example the XML string will contain all the customer codes when the sales rep code contains "A*".

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2]="tweb.tbred.com",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="READ OELCUST USING RANGE FROM 1000000 TO 999999;DATANAME LIST
CUST-CODE;SELECT WHEN #OEFCUST.SR-CODE LIKE "+QUO+"A*"+QUO

* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];

* if no errors, save the xml string
IF REQ$[0]="."
   XML$=XML$[1]
FI
```

In the above example XML$[1] will contain:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XMLSource_OELCUST>
  <OELCUST>
    <CUST_CODE>100100</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>100109</CUST_CODE>
  </OELCUST>
  <OELCUST>
    <CUST_CODE>101101</CUST_CODE>
  </OELCUST>
</XMLSource_OELCUST>
```

UPDATE Link XML Request Manager

SYNTAX

`REQ$[6]=UPDATE` *`link-name xml-elements`*

`link-name`	is the Link name.
`xml-elements`	is the xml string containing the key values and the data to update. The first set of xml-elements defines the key to the Link. The key value is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.

NOTES

- Before the Link is updated the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.
- If the Link contains sort definitions all sorts will be updated.
- A successful UPDATE XML reply will contain the "." character indicating the process completed successfully:

```
<standard XML header>
<link name>
   <UPDATE>.</UPDATE>
</link name>
```

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OELCUST>
<UPDATE>.</UPDATE>
</OELCUST>
```

If an error occurs the XML reply will contain an error message:

```
<standard XML header>
<error>error-message</error>
```

Examples:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ERROR>1) 102   : Can't find record for specified key  2) </ERROR>
```

EXAMPLE

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2]="tweb.tbred.com",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="UPDATE OELCUST <CUST-CODE>100104</CUST-CODE><SR-CODE>HP</SR-
CODE>",

* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];
```

UPDATEs Link OELCUST using a key value of 100104, changing the SR-CODE to "HP".

INSERT Link XML Request Manager

SYNTAX

`REQ$[6]=INSERT` *`link-name xml-elements`*

`link-name`	is the Link name
`xml-elements`	is the xml string containing the key values and the data to added. The first set of xml-elements defines the key to the Link. The key is always required and must be the first set of xml-elements. The order of non-key xml-elements is not significant.

NOTES

- Before the Link is updated the TS XML DataServer will validate all data against the Format definition. This includes validation using LOOKUP VIEW.

- If the Link contains an I/O Trigger, the trigger will be executed prior to the WRITE.

- If the Link contains sort definitions all sorts will be updated.

- A successful INSERT XML reply will contain the "." character indicating the process completed successfully:

  ```
  <standard XML header>
  <link name>
     <INSERT>.</INSERT>
  </link name>
  ```

  Example:

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <OELCUST>
     <INSERT>.</INSERT>
  </OELCUST>
  ```

  If an error occurs the XML reply will contain an error message:

  ```
  <standard XML header>
  <error>error-message</error>
  ```

EXAMPLE

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2]="tweb.tbred.com",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="INSERT OELSLRP <SR-CODE>XX</SR-CODE><SR-NAME>JustinCase</SR-
NAME><SR-SALES>0</SR-SALES>",

* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];
```

WRITEs a new record to Link OELSLRP using a key value of "XX" for the new SR-CODE and "Justin Case" for SR-NAME and "0" for SR-SALES.

DELETE Link XML Request Manager

SYNTAX

REQ$[6]=DELETE *link-name xml elements*

link-name	is the Link name
xml-elements	is the xml string containing the key values of the record to be deleted.

NOTES

- All xml-elements are assumed to be key fields.
- If the Link contains an I/O Trigger, the trigger will be executed prior to the REMOVE.
- A successful DELETE XML reply will contain the "." character indicating the process completed successfully:

```
<standard XML header>
<link name>
   <DELETE>.</DELETE>
</link name>
```

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OELCUST>
   <DELETE>.</DELETE>
</OELCUST>
```

If an error occurs the XML reply will contain an error message:

```
<standard XML header>
<error>error-message</error>
```

EXAMPLE

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2=”tweb.tbred.com",

* set the xml server process path
REQ$[3]="/xml.tbred",

* set the tbredpost path
REQ$[4]="/opt/tbred/tbredpost/",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="DELETE OELCUST <CUST-CODE>100104</CUST-CODE>",

* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];
```

REMOVEs a record from Link OELCUST using key 100104.

READ XSchema XML Request Manager

The TS XML DataServer will attempt to read the definition as an XSchema. If a matching XSchema definition is found the request will be processed as an XSchema. If a matching XSchema is not found it will be processed as a Link. If the Link is not found an empty XML string will be returned.

SYNTAX

REQ$[6]=READ *schema-name*

schema-name	is the name of the XSchema definition

EXAMPLES

```
REQ$[6]="READ OEFSCM01"
```

The following XSchema definition will read Links OELCUST, OELSLRP, OELINVH, and OELINVH populating the specified data elements.

```
   Schema: OEFSCM02 Sample System

   OELCUST USING KEY “100100”;

           DATANAME LIST CUST-CODE CUST-NAME CUST-STATE CUST-CITY

                         CUST-PHONE XXX SR-CODE

   OELSLRP USING #OEFCUST.SR-CODE; DATANAME LIST SR-NAME

     OELINVH USING #OEFCUST.CUST-CODE;

             BY SORT 1;

             DATANAME LIST INV-NUM INV-DATE INV-PRINTED-FLAG

                           INV-AMOUNT

       OELINVD USING #OEFINVH.INV-NUM

               DATANAME LIST INV-LNUM INV-QTY INV-LTAX INV-LEXTEN

                             ITEM-CODE INV-LPRICE INV-DISC;

               SELECT WHEN #OEFINVD.ITEM-CODE = "10-100-000”

       OELINVN USING #OEFINVD.ITEM-CODE;
```

The following XSchema request will post back to the calling program the data structure defined by XSchema OEFSCM02 in XML format.

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2]="tweb.tbred.com",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="READ OEFSCM02",
* optional authenticate using the supplied user id and password

* REQ$[7]="?User=tbred&Pwd=tbred"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];

* if no errors, save the xml string
IF REQ$[0]="."
   XML$=XML$[1]
FI
```

In the above example XML$[1] will contain:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <XMLSource_OEFSCM02>
    <OELCUST>
      <CUST_CODE>100100</CUST_CODE>
      <CUST_NAME>Toot-Your-Horn</CUST_NAME>
      <CUST_STATE>TX</CUST_STATE>
      <CUST_CITY>Port Lavaca</CUST_CITY>
      <CUST_PHONE>555-2341</CUST_PHONE>
      <SR_CODE>AF</SR_CODE>
    <OELSLRP>
      <SR_NAME>Albert Fisher</SR_NAME>
    </OELSLRP>
    <OELINVH>
      <INV_NUM>000001</INV_NUM>
      <INV_DATE>07/16/05</INV_DATE>
      <INV_AMOUNT>313.85</INV_AMOUNT>
    </OELINVH>
    <OELINVD>
      <INV_LNUM>02</INV_LNUM>
      <INV_QTY>1</INV_QTY>
      <INV_LTAX>.10</INV_LTAX>
      <INV_LEXTEN>1.40</INV_LEXTEN>
      <ITEM_CODE>10-100-000</ITEM_CODE>
      <INV_LPRICE>1.40</INV_LPRICE>
      <INV_DISC>30.00</INV_DISC>
    </OELINVD>
    <OELINVN>
       <ITEM_DESC>Little nails</ITEM_DESC>
    </OELINVN>
    <OELINVH>
       <INV_NUM>000016</INV_NUM>
       <INV_DATE>07/31/05</INV_DATE>
       <INV_AMOUNT>5021.29</INV_AMOUNT>
    </OELINVH>
    <OELINVD>
      <INV_LNUM>02</INV_LNUM>
      <INV_QTY>16</INV_QTY>
      <INV_LTAX>1.57</INV_LTAX>
      <INV_LEXTEN>22.40</INV_LEXTEN>
      <ITEM_CODE>10-100-000</ITEM_CODE>
      <INV_LPRICE>1.40</INV_LPRICE>
      <INV_DISC>30.00</INV_DISC>
    </OELINVD>
    <OELINVN>
      <ITEM_DESC>Little nails</ITEM_DESC>
    </OELINVN>
    <OELINVH>
      <INV_NUM>000031</INV_NUM>
      <INV_DATE>08/15/05</INV_DATE>
      <INV_AMOUNT>9728.78</INV_AMOUNT>
    </OELINVH>
    <OELINVD>
      <INV_LNUM>02</INV_LNUM>
      <INV_QTY>31</INV_QTY>
      <INV_LTAX>3.04</INV_LTAX>
      <INV_LEXTEN>43.40</INV_LEXTEN>
      <ITEM_CODE>10-100-000</ITEM_CODE>
    <INV_LPRICE>1.40</INV_LPRICE>
    <INV_DISC>30.00</INV_DISC>
  </OELINVD>
  <OELINVN>
    <ITEM_DESC>Little nails</ITEM_DESC>
  </OELINVN>
  </OELCUST>
</XMLSource_OEFSCM02>
```

Method XML Request Manager

SYNTAX

`REQ$[6]=METHOD` *`name`* `[`**`<msg>`***`optional_method_msg`***`</msg>`**`]`

`name`	is the name of the method to be executed on the remote server.
`optional_method_msg`	is an optional message that can is passed to the method.

NOTES

- The message returned by the XML Method must start with <method> and must terminate with </method>

  <method>*my-method-message-here*</method>

EXAMPLE

The following example will execute method WBXML02 on the remote server passing the message "this is an example". The method WBXML02 will then parse the message and construct a reply that is posted back to the calling program.

```
DIM REQ$[6], XML$[1];

* set the server address:
REQ$[2]="tweb.tbred.com",

* optional time out value
REQ$[5]="10"

* set the xmlrequest query string
REQ$[6]="METHOD WBQXML02<msg>this is an example</msg>"

* call the xml request manager to process the xmlrequest query string.
CALL "OOXMLSRX", REQ$[ALL], XML$[ALL];
IF REQ$[0]="."
   XML$=XML$[1]
FI;
```

The method WBXML02 is executed. When EXIT is executed the contents of XML$[1] is posted back to the calling program.

Method WBXML02

```
ENTER MSG1$[ALL],XML$[ALL];
PROCEDURE
      MSG$=MSG1$[1];
      P=POS(">"=MSG$),
      MSG$=MSG$(P+1),
      P=POS("</"=MSG$),
      MSG$=MSG$(1,P-1),
      XML$[1]="<method>WBQXML02 received: "+MSG$+"</method>";
```

The calling program receives the following message from the method WBXML02:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<method>WBQXML02 received: this is an example</method>
```