Tech-Web Client & Server HTTP Binding
Version 3.0
February 25, 2004

This document describes the HTTP binding specifications for OSA-EAI *Tech*-Web Client and Servers. The following terminology should be noted:

MUST	This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein, an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

The document will define the HTTP functionality which OSA-EAI *Tech*-Web clients and servers MUST support and the functionality clients and servers MAY support. The HTTP protocol utilized is a subset of the **Hypertext Transfer Protocol (HTTP)** Version 1.1 specification (RFC 2616) located at http://www.w3.org/Protocols/rfc2616/rfc2616.html .

HTTP is the network protocol used to deliver virtually all files and other data (collectively called *resources*) on the World Wide Web. In Tech-Web, HTTP must be implemented through TCP/IP sockets (though there are other possibilities).

A Tech-Web HTTP client sends an interface request to a Tech-Web HTTP server (Web server), which then sends responses back to the client. A Tech-Web Client application must be configurable to allow an end-user to specify the Internet host and port number of the destination server. If a port number if not specified by a user, the client MUST default to port 80.

HTTP *Requirements* for Tech-Web compliance

Like most network protocols, HTTP uses the client-server model. An *HTTP client* opens a connection and sends a *request message* to an *HTTP server*; the server then returns a *response message*, usually containing the resource that was requested. The HTTP *requirements* for Tech-Web compliance are described here. HTTP 1.1 has other optional features which clients and servers MAY implement.

Tech-Web Clients:

1) MAY use the HEAD request to inquire if a *Tech*-Web server supports a specific *Tech*-Web interface
2) MUST use the POST request to send all requests to a *Tech*-Web server
3) MUST recognize and support core HTTP server response status codes
4) MUST recognize and handle *chunked* transfer-encoding from a Tech-Web server

Tech-Web Servers:

1) MUST respond to a HEAD method to allow a client to inquire if a *Tech*-Web server supports a specific *Tech*-Web interface
2) MUST respond to a POST method to send the acknowledgement (results) of a request to a *Tech*-Web client
3) MUST use core HTTP server response status codes

The HEAD Method

The client MAY use the HEAD method to determine if a server supports a particular Tech-Web interface. The HEAD method request consists of:

- a required set of header lines (see description below)
- optional set of header lines which MAY exist (any additional HTTP V1.1 supported header lines may exist)
- a blank line (i.e. a <CR><LF> by itself)

Required Request Header Lines for HEAD Method
(Tokens are separated by whitespace)

Token 1	Token 2	Token 3
HEAD	Local URI path on the server of the requested resource, e.g., **/mimosaeai**	**HTTP/1.1**
Host:	*URI of the HTTP Tech-Web server, with an optional ":" and port number , e.g.,* **www.somehost.com:8080** If the port number is not specified, the default is 80. It is up to the server to translate this URI into the location of the application charged with accepting this request.	
MIMInterface:	*Extension header line which provides the name of the OSA-EAI Tech-Web interface the client is requesting, e.g.,* **mim_5001**	

NOTE: In addition to the headers above, a client **may** include a "Connection: close" header in the HTTP request if it wishes to close the current connection after receiving the response. The default in HTTP V1.1 is for the server to support persistent connections and allow several requests in series (called *pipelining).* If this header is included, the client **must not** send additional requests on this connection and the client **should** close the connection after receiving the response.

The HEAD method response consists of:

- a required set of header lines (see description below)
- optional set of header lines which MAY exist (any additional HTTP V1.1 supported header lines may exist)
- a blank line (i.e. a <CR><LF> by itself)

Required Response Header Lines for HEAD Method
(Tokens are separated by whitespace)

Token 1	Token 2	Token 3
HTTP/1.1	*response status code* that gives the result of the request, e.g., **200**	English *reason phrase* describing the status code, e.g., **OK**
Date:	*GMT of the HTTP Tech-Web server in a fixed-length subset of that defined by RFC 1123 (an update to RFC 822) in the form of Day, DD Mon YYYY HH:MM:SS GMT, e.g.,* **Sun, 06 Nov 1994 08:49:37 GMT**	

The client MUST recognize and support the following core HTTP server response status codes for the HEAD method:

Core Response Status Codes for HEAD Method
(Tokens are separated by whitespace)

Status Code	Status Reason	Meaning/Action
100	**Continue**	*During the course of an HTTP 1.1 client sending a request to a server, the server might respond with an interim "**100 Continue**" response. This means the server has received the first part of the request, and can be used to aid communication over slow links. In any case, all HTTP 1.1 clients must handle the 100 response correctly (perhaps by just ignoring it). Unlike other responses, it is always followed by another complete, final response.*
200	**OK**	*Interface is supported by the server*
400	**Bad Request**	*Client request has invalid syntax. The client SHOULD NOT repeat the request without modifications.*
404	**Not Found**	*The Web server has not found a Tech-Web server which matches the requested URI.*
500	**Internal Server Error**	*The server encountered an unexpected condition which prevented it from fulfilling the request.*
501	**Not Implemented**	*Interface is not supported by the server*
505	**HTTP Version not supported**	*A version other than 1.1 is being utilized by the client.*

If a client receives any 3xx, 4xx, or 5xx response notices, it MUST treat these responses as if a transport error has occurred.

The POST Method

The format of the POST method request consists of:

- a required set of header lines (see description below)
- optional set of header lines which MAY exist (any additional HTTP V1.1 supported header lines may exist)
- a blank line (i.e. a CRLF by itself)
- the message body consisting of an XML request which conforms to the OSA-EAI Tech-XML schema (e.g. query parameters, or row data to create).

Required Request Header Lines for POST Method
(Tokens are separated by whitespace)

Token 1	Token 2	Token 3
POST	Local URI path on the server of the requested resource, e.g., **/mimosaeai**	**HTTP/1.1**
Host:	*URI of the HTTP Tech-Web server, with an optional ":" and port number , e.g.,* **www.somehost.com:8080** If the port number is not specified, the default is 80. It is up to the server to translate this URI into the location of the application charged with accepting this request.	
Content-Type:	**text/xml;**	**charset="utf-8"**
Content-Length:	*Exact size of the attached XML message body, i.e.,* **384**	
MIMInterface:	*Extension header line which provides the name of the OSA-EAI Tech-Web interface the client is requesting, e.g.,* **mim_5001**	
MIMSchemaName:	*Extension header line which provides the name of the OSA-EAI Tech-Web schema the server should use for parsing the XML in the message body, e.g.,* **V3-0-5001-01QuerySite.xsd**	

NOTE: In addition to the headers above, a client **may** include a "Connection: close" header in the HTTP request if it wishes to close the current connection after receiving the response. If this header is included, the client **must not** send additional requests on this connection. The client **should** close the connection after receiving the response.

The POST method response consists of:

- a required set of header lines (see description below)
- optional set of header lines which MAY exist (any additional HTTP V1.1 supported header lines may exist)
- a blank line (i.e. a <CR><LF> by itself)
- the message body consisting of an XML acknowledgement which conforms to the OSA-EAI Tech-XML schema (e.g. status and query results).

Required Response Header Lines for POST Method
(Tokens are separated by whitespace)

Token 1	Token 2	Token 3
HTTP/1.1	*response status code that gives the result of the request, e.g.,* **200**	*English reason phrase describing the status code, e.g.,* **OK**
Date:	*GMT of the HTTP Tech-Web server in a fixed-length subset of that defined by RFC 1123 (an update to RFC 822) in the form of Day, DD Mon YYYY HH:MM:SS GMT, e.g.,* **Sun, 06 Nov 1994 08:49:37 GMT**	
Content-Type:	**text/xml;** *(Used when the response status code = 200 for the XML acknowledgment message body)* *OR* **text/plain;** *(Used when the response status code != 200 for explanatory text message in message body)*	**charset="utf-8"**
Content-Length:	*Exact size of the attached message body, i.e.,* **384**	

The client MUST recognize and support the following core HTTP server response status codes for the POST method:

Core Response Status Codes for POST Method

Status Code	Status Reason	Meaning/Action
100	**Continue**	*During the course of an HTTP 1.1 client sending a request to a server, the server might respond with an interim "**100 Continue**" response. This means the server has received the first part of the request, and can be used to aid communication over slow links. In any case, all HTTP 1.1 clients must handle the 100 response correctly (perhaps by just ignoring it). Unlike other responses, it is always followed by another complete, final response.* *No message body.*
200	**OK**	*Interface is supported by the server.* *Message body will contain the XML acknowledgement information. The client will still need to check inside the XML acknowledgement for the "**status"** element to see if the request was successful by reviewing the "**success"** attribute. If True (1), then the request was successful. If the "**success**" attribute is False (0), then the client should*

		refer to the message_code and message_text for a further description of the problem and take appropriate action. The V3.0 message codes & associated text are listed in a table below.
400	**Bad Request**	*Client request has invalid syntax. The client SHOULD NOT repeat the request without modifications.* *Message body may contain additional text explanation of error.*
404	**Not Found**	*The Web server has not found a Tech-Web server which matches the requested URI.* *Message body may contain additional text explanation of error.*
500	**Internal Server Error**	*The server encountered an unexpected condition which prevented it from fulfilling the request.* *Message body may contain additional text explanation of error.*
501	**Not Implemented**	*Interface is not supported by the server.* *Message body may contain additional text explanation of error.*
505	**HTTP Version not supported**	*A version other than 1.1 is being utilized by the client.* *Message body may contain additional text explanation of error.*

If a client receives any 3xx, 4xx, or 5xx response notices, it MUST treat these responses as if a transport error has occurred.

MIMOSA Tech-Web Acknowledgment Status Element Error Messages

Message Code	Message Text	Meaning/Action
0000000000000000-0001	**Invalid Connect String**	*MAY be returned only in interface mim_0003 (Schema name: V3-0-0003Connect.xsd) when a client attempts to connect to a server with a server-specified connect string.*
0000000000000000-0002	**Data Source Not Available**	*MAY be returned in any interface except mim_0004 (Schema name: V3-0-0004Disconnect.xsd). The data source(s) the server requires are not available. This could be due to database unavailability, etc.*
0000000000000000-0003	**Data Limit Exceeded**	*MAY be returned in any interface except mim_0003 and mim_0004. The resulting acknowledgement exceeds either the client-specified maximum which was established upon connection through mim_0003 interface through the **max_document_size_in_bytes** attribute on entity "param" or exceeds a server-defined limit. Client could re-try by*

		specifying additional filtering.
0000000000000000-0004	**No Privilege For Operation**	*MAY be returned in any interface except mim_0003 and mim_0004. The connected user does not have the rights to perform the request.*
0000000000000000-0005	**Language Not Supported**	*MAY be returned only in interface mim_0003 when a client attempts to change the server default language on returned text in lookup tables and other related text columns from "eng-US" through the* ***language_code*** *attribute on entity "param".*
0000000000000000-0006	**Function Not Implemented**	*MAY be returned in any interface except mim_0003 and mim_0004. Though the interface is recognized by the server, the client-requested action has not been implemented.*
0000000000000000-0007	**Cannot Create Entry - Primary Key Already Exists**	*MAY be returned in any CREATE interface when the client-requested entry to create in the server's system already exists. The Primary key is the unique set of attributes (columns) which only one row can contain.*
0000000000000000-0008	**Cannot Create Entry - Not All Required Data Specified**	*MAY be returned in any CREATE interface when the client-requested entry to create in the server's system has one or more null attributes which are defined as required attributes.*
0000000000000000-0009	**Cannot Create Entry - Invalid Data Specified**	*MAY be returned in any CREATE interface when the client-requested entry to create in the server's system has invalid data which could be attributes with invalid data types.*
0000000000000000-0010	**Cannot Create Entry - Exceeded Attachment Limit**	*MAY be returned only in interface mim_8031 (Schema name: V3-0-8031-01CreateWRBlob.xsd) when a client attempts to create more binary attachments associated with a work request than a server supports.*
0000000000000000-0011	**Cannot Create Entry - Attachment Data Type Not Supported**	*MAY be returned only in interface mim_8031 (Schema name: V3-0-8031-01CreateWRBlob.xsd) when a client attempts to create a binary attachment associated with a work request which is not supported by the server.*
0000000000000000-0012	**Session ID Not Valid - Connect Required**	*MAY be returned in any interface except mim_0003 (Schema name: V3-0-0003Connect.xsd). The client has specified in its "header" element a "session_id" attribute which is not valid on the server at this time. This may be due to a timeout condition on the server, which requires the client to re-connect before-attempting the request again.*

NOTE: User-defined Tech-Web Acknowledgment Status Element error messages MAY be sent by the server. The server MUST use a message code which contains a unique 16-character site code, followed by a dash ("-") and 4 numeric characters. If a client does

not recognize these additional status element message codes, it will only be able to log or display the associated text for human intervention.

Chunked Transfer-Encoding

If a server wants to start sending a response before knowing its total length (like with long script output), it MAY use the simple *chunked transfer-encoding*, which breaks the complete response into smaller chunks and sends them in series. A client MUST support this type of a response which contains the "**`Transfer-Encoding: chunked`**" header.

A chunked message body contains a series of *chunks*, followed by a line with "0" (zero), followed by optional footers (just like headers), and a blank line. Each chunk consists of two parts:

- a line with the size of the chunk data, in hex, possibly followed by a semicolon and extra parameters you can ignore (none are currently standard), and ending with <CR><LF>.
- the data itself, followed by <CR><LF>.

Sample HTTP Exchange

Client “connect” request:

```
POST /mimosaeai HTTP/1.1
Host: www.eaiserver.org:80
Content-Length: 420
Content-Type: text/xml; charset="utf-8"
MIMInterface: mim_0003
MIMSchemaName: V3-0-0003-01Connect.xsd
<?xml version="1.0" encoding="utf-8"?>
<mim_0003>
  <connect_req>
    <param connect_string="username=robert023 password=xfw3041"
      language_code="en-US" include_non_active_rows_def="false"
      include_all_site0_rows_def="false"
      include_row_info_columns_def="false"
      include_lc_info_columns_def="false" />
  </connect_req>
</mim_0003>
```

Server “connect” success response:

```
HTTP/1.1 200 OK
```

```
Date: Mon,02 Apr 2003 23:32:00 GMT
Content-Length: 529
Content-Type: text/xml; charset="utf-8"
<?xml version="1.0" encoding="utf-8"?>
<mim_0003 xmlns="http://www.mimosa.org/TechXMLV3-0">
  <connect_ack>
          <header include_row_info_columns="0"
          session_id="session001" include_all_site0_rows="0"
          include_non_active_rows="0"
          include_lc_info_columns="0"/>
      <server name="Indus International"/>
          <param language_code="en-US"
          include_lc_info_columns_def="0"
          include_all_site0_rows_def="0" connect_string="Indus
          Test Connect" include_non_active_rows_def="0"
          include_row_info_columns_def="0"/>
       <status success="1"/>
      <row>
          <db_mim_interface db_site="000003EA00000001"
          db_id="1" interf_type_code="9620830"/>
          <db_mim_interface db_site="000003EA00000001"
          db_id="1" interf_type_code="9620130"/>
          <site_database db_site="000003EA00000001" db_id="1"
          name="Indus InSite EAM Database"/>
      </row>
  </connect_ack>
</mim_0003>
```

HTTP Proxies

An *HTTP proxy* is a program that acts as an intermediary between a client and a server. It receives requests from clients, and forwards those requests to the intended servers. The responses pass back through it in the same way. Thus, a proxy has functions of both a client and a server.

Proxies are commonly used in firewalls, for LAN-wide caches, or in other situations. When a client uses a proxy, it typically sends all requests to that proxy, instead of to the servers in the URLs. Requests to a proxy differ from normal requests in one way: in the first line, they MUST use the complete URL of the resource being requested, instead of just the path. For example,

```
GET http://www.eaiserver.org/mimosaeai HTTP/1.1
```

That way, the proxy knows which server to forward the request to (though the proxy itself may use another proxy).
