

Commodity Data WebServices API

# User's Guide

---



---

<b>Commodity Data API Basics</b>	<b>4</b>
<b>Installation</b>	<b>4</b>
<b>API Authentication</b>	<b>4</b>
<b>Data Request</b>	<b>5</b>
<i>The values returned will be rounded to a maximum of three decimal places.</i>	6
<i>Query Execution</i>	6
<i>Records</i>	7
<i>Worksheets</i>	9
<i>Checking the progress of the Data Request</i>	10
<i>Response Payload for Data Request</i>	11
Response Status Codes	12
<b>Meta Data</b>	<b>12</b>
<i>Exception Handling of Meta Data</i>	17
<i>Units of Measure</i>	17
<b>Search</b>	<b>18</b>
<i>Search Response Payload</i>	18
<b>Shortcuts</b>	<b>19</b>
<i>Shortcuts Response Payload</i>	19
<b>File Service</b>	<b>20</b>
<i>Organization and Users</i>	20
<i>File Owners</i>	20
<i>File Keys</i>	21
<i>Searches</i>	21
<i>Storing Files</i>	23
<i>Getting Files</i>	24
<b>Appendix</b>	<b>27</b>
<i>Data Request XSD</i>	27
<i>Query Execution Data Request XSD</i>	27
<i>Records Data Request XSD</i>	28
<i>Worksheet Data Request XSD</i>	29
<i>Data Request Response XSD</i>	32

---

<i>Meta Data Relation Information XSD</i>	34
<i>Unit of Measure Conversion Response XSD</i>	37
<i>Search Response XSD</i>	37
<i>Shortcuts Response XSD</i>	38
<i>Search Files Response XSD</i>	39
<i>Search Files Response Content XSD</i>	40

---

## Commodity Data API Basics

- The base URI for all API services is <http://host:port/rs/api>. In this document we will refer to this path as '\$web\_context'.
- This API service uses basic access authentication and its component authorization is managed through user roles.

## Installation

To install Web Services, please consult the Web Services API Installation Guide. For the latest version please contact your Sales Director.

## API Authentication

All API requests require the use of HTTP Basic authentication to authenticate the user making the request. For a description of HTTP Basic authentication please consult [http://en.wikipedia.org/wiki/Basic\\_access\\_authentication](http://en.wikipedia.org/wiki/Basic_access_authentication)

The Morningstar Commodity Data WebService API is stateless and does not use cookies or any session mechanism. Therefore it is necessary to provide the basic authentication header in every request. If the Authentication header is not present the server will reject the request with HTTP code 401 (Unauthorized). Example exchanges between a client and server are below:

Client makes request without Authentication header:

```
> GET /rs/api/schema/relations/NG HTTP/1.1
```

Server's response:

```
< HTTP/1.1 401 Unauthorized
< Server: Apache-Coyote/1.1
< Date: Thu, 02 Sep 2010 20:10:12 GMT
```

The server has indicated that the caller must be authorized to access this resource.

Client makes request with HTTP Basic Authentication header:

```
> GET /rs/api/schema/relations/NG HTTP/1.1
> Authorization: Basic Y29saW5pOXXXXXluaQ==
```

---

Server's response:

```
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: application/xml
< Content-Length: 150
< Date: Thu, 02 Sep 2010 20:14:31 GMT
... <content follows> ...
```

The server accepted the credentials and responded with 200 (OK) and the requested resource when the client provided a basic Authorization header with valid user credentials.

Since every API resource requires authentication, there is no need to wait for the 401 response from the server. Simply add the Authorization header to every request (i.e. authenticate preemptively).

Contact the server administrator for valid username / password credentials.

## Data Request

Currently there are three methods for requesting data from the Morningstar Commodity Data Web Service

- Query Execution
- Records
- Worksheet

All of them reference the same URLs

- `$web_context/datarequests`
- `$web_context/datarequests/{id}`

Each service takes a different request payload but all return the same type of response.

There are two possible scenarios when requesting data:

1. The data request finishes, and the server returns the result to the client in the POST response payload.
2. The server returns only a request ID in the POST response payload. When this happens the client must start polling with a 'GET' request for the data request result.

---

The following sections discuss each data request service, the format of the response payload and how the response should be handled.

Each request type (query, records, and worksheet) must be wrapped within a <DataRequest> element. This element accepts a single optional attribute "scale", which can be used to instruct the server to round values to the specified scale. If the scale attribute is not specified then the server returns values "as-is" from the Commodity DataServer. For Example:

```
<DataRequest scale="3">
  <Query>
    <Text>%exec.units = 4
      Show A: Open of NG   B: Close of NG   C: Open of CL D: Close of
CL
      WHEN Date is from 04/01/2010 to 04/05/2010
    </Text>
  </Query>
</DataRequest>
```

### **The values returned will be rounded to a maximum of three decimal places. Query Execution**

The API service executes a Commodity DataServer query and returns a 'DataRequestResponse' entity.

#### *URI Formats and supported HTTP Methods*

POST: Starts the execution of the query request

`$web_context/datarequests`

#### **Request Payload**

```
<DataRequest>
  <Query>
    <Text>%exec.units = 4
      Show A: Open of NG   B: Close of NG   C: Open of CL D: Close of
CL
      WHEN Date is from 04/01/2010 to 04/05/2010
    </Text>
  </Query>
</DataRequest>
```

---

## Response Payload

See Response Payload for Data Request

## Records

The API service executes a Commodity DataServer API 'GetRecords' call and returns a 'DataRequestResponse'.

### *URI Formats and supported HTTP Methods*

POST: Starts the execution of 'GetRecords' request with the given parameters:

```
$web_context/datarequests
```

## Request Payload

```
<DataRequest>
  <GetRecsParams>
    <rels>IBM,DELL,KR (A list of comma-separated MIM relations)</rels>
    <cols>Close,Open (A list of comma-separated MIM columns)</cols>
    <nu>2 (A non-negative integer indicating number of units,
default=1)</nu>
    <unit>DAYS (The frequency to run get-records, default=4, or
DAYS)</unit>
    <fd>(From date, YYYY-MM-DD)</fd>
    <td>(To time, YYYY-MM-DD)</td>
    <ft>(From time, hh:mm:ss.sss)</ft>
    <tt>(To date, hh:mm:ss.sss)</tt>
    <mdf>FORWARD (Fill Option for missing data fill)</mdf>
    <mhf>NAN(Fill option for missing holiday fill)</mhf>
    <sn>SKIP_NONE(Skip-NaN indicator)</sn>
    <limitMode>BY_RECORDS(The limit mode to run get-
records)</limitMode>
    <limitSize>1000(size of limit: number of records or memory
usage)</limitSize>
    <ctu>APPEND_TO_DAILY (Current tick usage, default =
APPEND_TO_NONE)</ctu>
  </GetRecsParams>
</DataRequest>
```

---

XML Enumeration of units (frequency)

- DAYS (DEFAULT)
- MILLISECONDS
- SECONDS
- MINUTES
- HOURS
- WEEKS
- MONTHS
- QUARTERS
- YEARS

XML Enumeration of fill options for mdf and mhf

- NAN (Default)
- FORWARD
- BACKWARD
- INTERP\_LIN
- INTERP\_GEO
- INTERP\_LOG
- NEAREST

XML Enumeration of skip-nan indicator

- SKIP\_NONE (Default)
- SKIP\_ALL

XML Enumeration of limit mode

- BY\_RECORDS (Default)
- BY\_MEMORY

XML Enumeration of current tick usage

- APPEND\_TO\_NONE (Default)
- APPEND\_TO\_DAILY
- APPEND\_TO\_TICK
- APPEND\_TO\_ALL

**Response Payload**

See Response Payload for Data Request

---

## Worksheets

This API service executes a Commodity Charts worksheet and returns a 'DataRequestResponse'.

*URI Formats and supported HTTP Methods*

POST

`$web_context/datarequests`

### Request Payload

```
<DataRequest>
  <Worksheet version="1.1">
    <Items>
      <Item type="SERIES">
        <label>A</label>
        <symbol>NG</symbol>
        <column>Open</column>
        <description>NYMEX: Henry Hub Natural Gas Futures (USD/MMBTU):
Pit Session</description>
        <timeUnits>Daily</timeUnits>
        <units>USD/MMBTU</units>
        <factor>5.8</factor>
        <precision>4</precision>
      </Item>
      <Item type="SERIES">
        <label>B</label>
        <symbol>CL</symbol>
        <column>High</column>
        <description>NYMEX: Light, Sweet Crude Oil Futures (USD/BBL):
Pit Session</description>
        <timeUnits>DAILY</timeUnits>
        <units>USD/BBL</units>
        <factor>1</factor>
        <precision>6</precision>
      </Item>
      <Item type="FORMULA">
        <label>C</label>
        <formula>2 * Close of NG</formula>
        <description>Formula: Double NG-Close</description>
        <timeUnits>DAILY</timeUnits>
        <units>USD/MMBTU</units>
        <factor>5.8</factor>
        <precision>4</precision>
        <chartLabel>NG Close Double</chartLabel>
      </Item>
      <Item type="CURVE">
        <label>D</label>

```

```

        <symbol>NG</symbol>
        <column>Close</column>
        <description>6/09/2010 NG-Close curve</description>
        <timeUnits>DAILY</timeUnits>
        <units>USD/MMBTU</units>
        <factor>5.8</factor>
        <precision>4</precision>
        <curveDate>2010-06-09</curveDate>
    </Item>
</Items>
    <DateOptions>
        <startDate>2010-04-14</startDate>
        <startDateRelative>>false</startDateRelative>
        <endDate>2010-04-24</endDate>
        <endDateToday>>false</endDateToday>
        <numPeriod>5</numPeriod>
        <period>DAYS</period>
    </DateOptions>
    <WorksheetOptions>
        <skipAllNans>>false</skipAllNans>
        <missingDataNanFill>FILL_NAN</missingDataNanFill>
        <skipWeekends>>false</skipWeekends>
        <tableSummaryStats>>true</tableSummaryStats>
        <numExecUnits>1</numExecUnits>
        <executionUnits>DAYS</executionUnits>
        <sortAscending>>true</sortAscending>
    </WorksheetOptions>
</Worksheet>
<DataRequest

```

### Response Payload

See Response Payload for Data Request

## Checking the progress of the Data Request

GET : Checks the progress of a data request and obtains the result when finished.

\$web\_context/datarequests/{id}  
id : id of the data request (type long)

### Response Payload

See Response Payload for Data Request

---

## Response Payload for Data Request

The 'POST' and 'GET' response payloads always convey the data request ID, request status code, and start time of the request. Dates and times are formatted in XML Schema Date (xs:date) and Time (xs:time) format (Please see [http://www.w3schools.com/schema/schema\\_dtypes\\_date.asp](http://www.w3schools.com/schema/schema_dtypes_date.asp) for more information on XML Schema Date and Time formats.)

- For POST, if the processing of request finishes in 1 second without error, the response contains the request ID, status, start time, end time, and a Report element.
- For GET, if the processing has finished, the response contains the ID, status, start time, end time, and a Report element.

```
<DataRequestResponse
  id="117325"
  startTime=' '2010-04-13T14:53:07"
  endTime=' '2010-04-13T14:53:07"
  status="100"
>
  <report>(DataRequestResponse report contents)</report>
</DataRequestResponse>
```

- For POST, if the processing of request takes longer than 1 second, the response contains the ID, status, and start time only.
- For GET, if the processing has not finished, the response contains the ID, status, and start time only.

```
<DataRequestResponse
  id="117325"
  startTime=' '2010-04-13T14:53:07"
  status="200"
/>
```

- If the server encounters an unrecoverable error while processing a request it will return HTTP code 500 and a diagnostic message. Clients cannot resolve a 500 error; please provide the diagnostic message to the server's administrator for further investigation.

---

## Response Status Codes

Each DataRequestResponse element contains a numeric “status” attribute that indicates the current status of the data request, as indicated in the table.

Status Code	Meaning
100	The data request completed successfully.
120	The data request completed but there was no daily data
130	The data request completed but there was no data
200	The data request was accepted but is not yet complete
300	The data request failed

## Meta Data

The API service provides the ability to browse the hierarchy and also get information related to a Commodity DataServer relation.

*URI formats and supported HTTP Methods*

GET : Obtains Commodity DataServer relations related information to the n-th level

```
$web_context/schema/relations/{relNames}?colName={colName}&showChildren={flag}&defaultColumns={cNames}&precision={nDigits}&desc={flag}&aliasTarget={flag}&path={flag}&showColumns={flag}&cDesc={flag}&uom={flag}&dateRange={flag}&dataPreview={flag}&shorthand={flag}
```

- {relNames}
  - A comma-separated list of relation names (or paths of relations) of the nodes in the relation tree. The name strings are to be trimmed (that is, “ NG, CL, HO ” has the same effect as “NG,CL,HO”
- colName={colName}
  - The name of the target relation-column. (Default = empty string: The <Columns> tag in the response payload contains information about all relation-columns and the <DataPreview> tag contains the information about the default column. Otherwise <Columns> and <DataPreview> contain information about the specified relation-column only). NOTE: When colName is non-empty, defaultColumns, dateRange, dataPreview are

---

overridden. (Their values are then associated with colName, instead of the default column).

- showChildren={flag}
  - Whether the response provides the child relation information (false = Relinfo of the relation itself, true = Relinfos of the relation and its immediate child relations. Default: showChildren=false).
- defaultColumns={cNames}
  - A comma-separated list of default-column names (Default = empty string, in which case the server is to use a hard-coded/pre-determined list of names.) Note: When colName is non-empty, defaultColumns is overridden (The response payload contains only colName related column information.) The name strings are to be trimmed ( " Close, Mid, LmpVal " has the same effect as "Close, Mid, LmpVal".)
- precision={nDigits}
  - The number of decimal places to which the double-precision data will be rounded to (Default: precision=8)
- desc={flag}
  - Whether the response provides the description of each relation node (**true** = response has description information, **false** or missing query parameter = no description in payload.)
- aliasTarget={flag}
  - Whether the response provides the name of the target relation of an alias (**true** = the relName in the request is considered an alias and the response provides the real relation name, **false** or missing query parameter = no target relation name information in response payload.)  
Example: Assume relName = IBM and aliasTarget = true, the response provides the target relation name, TopRelation:Equities:GlobalInsight:Nyse:Tickers:i:ib:GII.IBM.NYSE.
- path={flag}
  - Whether the response provides the parent path of each relations node (**true** = response has path information, **false** or missing query parameter = no path in payload.)
- dataPreview={flag}
  - Whether the response provides preview data for the relation-column.
- shorthand={flag}
  - Whether the response **disables** field value population for child relations to accelerate the meta-data fetch process. This flag works only with showChildren=true.  
When shorthand=true, the web service does **not** populate path, alias target, columns,

---

conversions, date range, data preview for the child relations.

The web service **always** populates name, type and hasChildren for the children. The only effective **optional** flag, when shorthand=true, is desc.

- showColumns={flag}
  - Whether the response provides column and relation-column information (**true** = response has column and relation-column information, **false** or missing query parameter = no column related information. showColumns=false, overrides the parameters of cDesc, uom, dateRange.)
- cDesc={flag}
  - Whether the response provides column description information.
- uom={flag}
  - Whether the response provides units-of-measure information.
- dateRange={flag}
  - Whether the response provides data-range dates information for each Column (in <DailyDataFrom>, <DailyDataTo>, <MinDataFrom>, <MinDataTo>, <SecDataFrom>, <SecDataTo>, <MsDataFrom>, <MsDataTo> XML tags for day, minute, second, millisecond data-range dates).

#### Request Payload

None.

#### Response Payload

Based on the query parameter, depth={n}, the payload data node tree is built to the n-th level in depth.

NOTE: If the sub-tree has fewer levels than n, the payload data is built only to the deepest possible level.

NOTE: The response payload always provides the relation name and type information and the hasChildren state indicator (1=Yes, 0=No, -1=Not Sure) of each relation node.)

Each relation node contains the relation name, path, description, etc. and possibly (depending on the relation type) a list of child relations.

```
<RelInfos>
  <RelInfo desc=" futures-relation " path="TopRelation" name="Futures"
    type="CATEGORY" hasChildren="1">
```

```

<children>
  <RelInfo path="TopRelation:Futures" name="Nymex" type="CATEGORY"
    hasChildren="1">
    <children>
      <RelInfo hasChildren="-1"
        desc="NYMEX: Henry Hub Natural Gas Futures (USD/MMBTU):
          Pit Session"
        path="TopRelation:Futures:Nymex" name="NG" type="FUTURES">
        <rollDay>Expiration Day.</rollDay>
        <rollRule>front, actual prices.</rollRule>
        <rolloverDataType>BOTH</rolloverDataType>
        <contractUnits>1000.0</contractUnits>
        (<expirationDate>2900-05-17</expirationDate>)
        (<firstNoticeDay>1995-11-15</firstNoticeDay>)
        <currencyConversions>
          <Conversion from="UNS">
            <Factor value="1" to="UNS" />
          </Conversion>
          <Conversion from="USD">
            <Factor value="(10000/(Close of FXBRC))" to="BRC" />
            ...
          </Conversion>
        </currencyConversions>
        <unitConversions>
          <Conversion from="UNS">
            <Factor value="1" to="UNS" />
          </Conversion>
          <Conversion from="MMBTU">
            <Factor value="0.947817" to="GJ" />
            ...
          </Conversion>
          ...
        </unitConversions>
        <Columns>
          <Column baseUnit="UNS" baseCurrency="UNS"
            cDesc="Total Open Interest" cName="TotalOI">
            <dailyDataFrom>1990-04-03T00:00:00-
              05:00</dailyDataFrom>
            <dailyDataTo>2010-04-19T00:00:00-05:00</dailyDataTo>
          </Column>
          ...
          <Column baseUnit="MMBTU" baseCurrency="USD" cDesc="Open"
            cName="Open">
            <Conversion from="MMBTU">
              <Factor value="5.8" to="BBL" />
            </Conversion>

```

```

        <dailyDataFrom>1990-04-03T00:00:00-
05:00</dailyDataFrom>
        <dailyDataTo>2010-04-20T00:00:00-05:00</dailyDataTo>
        <minDataFrom>1990-04-04T00:00:00-05:00</minDataFrom>
        <minDataTo>2010-04-23T00:00:00-05:00</minDataTo>
    </Column>
    ...
</Columns>
<DataPreview column="Close">
    <dates>
        <date>2010-02-28</date>
        <date>2010-02-29</date>
    </dates>
    <values>
        <value>1.2345</value>
        <value>1.6789</value>
    </values>
</DataPreview>
</Relation>
</children>
</RelInfo>
</children>
</RelInfo>
<RelInfo attributes=...>
    sub-elements ...
</RelInfo>
</RelInfos>

```

(For more information, Please see Meta data Relation Information XSD in the Appendix)

■ Type of relation (com.lim.rs.api.models.RelTypeWs)

- CATEGORY
- NORMAL
- FUTURES
- FUTURES\_CONTINUOUS
- FUTURES\_CONTRACT

NOTE: The relation-column specific conversion factor lists (<Conversion> tag inside <Column> inside <RelInfo> contains only the unit factors. All currency factors are independent of relations and columns.

---

NOTE: The API Meta data sub-service will not provide artificial column (such as Bar and Mid\*) data automatically. It is up to the user code to define and synthesize these columns.

### Exception Handling of Meta Data

`$web_context/schema/relations/{relNames}? ...` allows you to specify a list of relation names to retrieve a collection of RelInfo data.

During the meta data retrieval for each relation, if the server encounters an error, it carries the HTTP status code in the error message, which in turn is embedded in the corresponding RelInfo instance.

Example: GET `http://host:port:rs/api/schema/relations/NG,CL,badrel1,badrel2? ...` generates:

```
<RelInfos>
  <RelInfo ... name="NG" type="FUTURES">...</RelInfo>
  <RelInfo ... name="CL" type="FUTURES">...</RelInfo>
  <RelInfo hasChildren="0" path="" name="badrel1" type="INVALID">
    <ErrorMsg>Status: 404. ERROR Relation does not exist:
      badrel1</ErrorMsg>
  </RelInfo>
  <RelInfo hasChildren="0" path="" name="badrel2" type="INVALID">
    <ErrorMsg>Status: 404. ERROR: Relation does not exist:
      badrel2</ErrorMsg>
  </RelInfo>
</RelInfos>
```

### Units of Measure

Units-of-measure and conversion-factor related name strings and values DO NOT need separate URIs. The unit of measure information is part of the Meta data response payload.

```
<RelInfo name="NG" type="FUTURES" ... >
  <currencyConversions>
    <Conversion ...>...</Conversion>
    ...
  </currencyConversions>
  <unitConversions>
    <Conversion from="MMBTU">
      <Factor value="0.947817" to="GJ" />
      ...
    </Conversion>
    <Conversion ...>...</Conversion>
    ...
  </unitConversions>
```

---

```
...  
</RelInfo>
```

## Search

This API service searches the Commodity DataServer database for matches to the given string

*URI formats and supported HTTP Methods*

GET : starts the search with the given query parameters

```
$web_context/search/indices/search?{query-parameters}
```

Query Parameters

Type: Enumerated MIM relation type strings, default = NOT\_CATEGORY

(Options: ALL, NOT\_CATEGORY, CATEGORY, NORMAL, FUTURES, FUTURES\_CONTINUOUS,  
FUTURES\_CONTRACT)

Search: Input search string such as search=NG, default = <empty string>

rel\_p: Flag of whether the search involves the relation name, default = true

desc\_p: Flag of whether the search involves the parent path, default = true

parent\_p: Flag of whether the search involves the parent path, default = true

Max\_results: Limit of search count, default = 1000

URI example

```
$web_context/search/indices/search?search=ng
```

```
$web_context/search/indices/search?search=IBM&max_results=100
```

## Search Response Payload

Running HTTP GET against a search service URI, you obtain the search result in the response payload.

The result is presented as a <searchList> XML tag, which contains a list of <relation> tags providing its name, description and parent path information, with the list size.

For example, running the following GET results in the below searchList.

```
$web_context/search?
```

```
search=ng*&max_results=100&rel_p=true&desc_p=true&parent_p=false&type=NOT_CATEGORY
```

```

<searchList size="100">
  <relation>
    <name>NG_F</name>
    <description>NYMEX: Henry Hub Natural Gas Futures (USD/MMBTU):
      Pit Session (January)</description>
    <parent>TopRelation:Futures:Nymex:NG</parent>
    <type>futures continuous</type>
  </relation>

  More <relation> tags...

  <totalSearchResults>100000</totalSearchResults>
</searchList>

```

## Shortcuts

This API service provides access to the user and Morningstar Commodity Data defined shortcuts.

*URI formats and supported HTTP Methods*

GET :

```

$web_context/shortcuts/public_profiles
$web_context/shortcuts/[user [/{user} [/profiles [/{profile}
[/aliases [/{alias} ]]]]]]

```

Users	: User List
{user}	: User Of Name \$user
profiles	: Profile List Under \$user
{profile}	: Profile Of Name \$profile
aliases	: Alias List Under \$profile
{alias}	: Alias Of Path \$alias, delimited with :
public_profiles	: List Of All Profiles Declared Public By Owners

## Shortcuts Response Payload

Please see Shortcuts Response XSD in the Appendix. For further details please contact client support (supportcases@lim.com) asking for information listed on the internal WIKI (website:

[http://wiki.lim.com/index.php/Mimic\\_Shortcuts\\_WS#Resource\\_Types\\_And\\_Payloads](http://wiki.lim.com/index.php/Mimic_Shortcuts_WS#Resource_Types_And_Payloads))

---

## File Service

This API service provides file storage on the server for end users. The main file service capabilities are:

- Each file has a single owner. Only the owner can modify a file or change its visibility (public or private) status.
- Each file is identified by a unique key. This key is used to store and retrieve the file.
- A file is either public or private. If public then the file can appear in other users' searches and they can read the file. If private then only the owner can read it.
- Only a file's owner can overwrite or delete it.
- File metadata includes: type, creation date, modification date, and size.
- Users can search for files owned by all users (a global search) or owned by a single user.
- File searches can be restricted by file type, key, and/or owner

## Organization and Users

Each individual who accesses the file service is a user. Users are grouped into organizations. Users only have access to files owned by other users in their organization. This allows us to segregate files per customer. For example, users from Company X could never access files from Company Y users, even if they are accessing the same Tomcat server.

We use virtual hosts to identify the organization at request time. The virtual host is the full hostname and port the caller used to reach the web service. For more information on virtual hosts please see the Apache Tomcat webpage at <http://tomcat.apache.org/tomcat-6.0-doc/virtual-hosting-howto.html>

For example: `http://www.lim.com:9999/rs/api/fs/bob/bob_file_10.txt`

The virtual host for this request is "www.lim.com:9999".

The user's username is the login they provided when authorizing with Tomcat.

For example (using curl syntax):

`http://bob:bobpassword@www.lim.com:9999/rs/api/fs/bob/bob_file_10.txt`

For this URL the username is "bob" and the organization is "www.lim.com:9999". Bob will only have access to files owned by himself and other users in his organization

## File Owners

Each file is owned by a single user. Only the file owner can change or delete it. The owner can also change the file's visibility to either public or private. Private files are only accessible by their owner and do not appear in others users' file searches.

---

## File Keys

Files are uniquely identified by keys. A key is an arbitrary string of up to 10K characters. By using a naming convention for keys it is possible to simulate a typical file system folder structure. In the following examples the keys are highlighted in red.

```
rs/api/fs/bob/bob_file_10.txt  
rs/api/fs/bob/apps/mimic/folder1/worksheets1.llc  
rs.api/fs/bob/apps|webmimic|My Folder|big_query.txt
```

In the first example the key is simply, the file name. The last two examples use a delimiter character to simulate directories. However the file service has no concept of directories, so these are just keys. Applications using the service are free to use any convention when creating keys.

## Searches

The file service provides a search feature for locating and listing files. There are two types of searches:

- Global - Searches all users' files (in a single organization)
- Per User - Searches a single user's files.

A global search scans all users' files but only returns files that are public. A per user search searches a single user's files and returns only public ones (if the user is not the owner) or both public and private (if the user is searching their own files.)

A search can be narrowed by specifying optional search criteria. The available criteria are:

- file key prefix
- owner
- file type
- modification and creation date (not yet implemented)

Any combination of these can be used to narrow the returned results.

When using a key prefix search criteria the search value is matched against the beginning of each key. The key prefix "apps/mimic/folder1" would return every file whose key begins with that prefix (in effect, providing a directory listing.)

When searching by file type the provided type must match exactly. This makes it possible to list all the Commodity Charts worksheets, etc.

Global Search Syntax

```
GET /rs/api/fs?prefix=<prefix>&typ=<mime type> HTTP/1.1
Host: colin-dt:8888
```

### Per User Search Syntax

```
GET/rs/api/fs/user_to_search?prefix=<prefix>&type=<mim type> HTTP/1.1
Host: colin-dt:8888
```

### Request Parameters

Request Parameter	Description
prefix	Optional. Narrows the search to only return files whose keys begin with a specific prefix
type	Optional. Narrows the search to only return files with the specified MIME type.

### Response

#### HTTP Response Codes

200 Ok	Request was successful.
--------	-------------------------

```
HTTP/1.1 200 Ok
Server: Apache-Coyote/1.1
Context-Type: application/xml
Context-Length: 592
Date: Mon, 17 May 2010 17:46:04 GMT
[See sample response XML below]
```

### Response XML

```
<SearchFilesResult>

  <Contents>
    <ACL>PUBLIC</ACL>
    <Created>2010-05-13T15:21:03.605-05:00</Created>
    <Key>amazon_order_for_logitech.jpg</Key>
    <MimeType>image/jpeg</MimeType>
    <LastModified>2010-05-17T11:48:37.899-05:00</LastModified>
```

```
<Size>122180</Size>
<Owner>colini</Owner>
</Contents>

<Contents>
<ACL>PRIVATE</ACL>
<Created>2010-05-13T14:50:35.574-05:00</Created>
<Key>really_long_query.xml</Key>
<MimeType>text/xml</MimeType>
<LastModified>2010-05-13T15:19:59.420-05:00</LastModified>
<Size>226</Size>
<Owner>colini</Owner>
</Contents>

</SearchFilesResult>
```

## Storing Files

A user can add a file to his store via HTTP POST. The Content-Type and Content-Length headers must be specified. The Content-Type is stored as the file's type, and the length is used to ensure the server receives the entire file. If the received bytes do not match Content-Length an error is returned and the file is not stored.

A new file will automatically overwrite an existing file if it has the same key.

Users can only put files in their own storage area. They cannot write to others.

When storing a file the user may request that the file is made public. By default files are private.

Syntax

```
POST /rs/api/fs/colini/some_image_file.jpg?acl=[0|1]&aclonly=1 HTTP/1.1
Host: colin-dt:8888
Content-Type: image/jpeg
Content-Length: 122180
[122180 bytes of data]
```

Request Parameters

Request Parameter	Description
acl	0 – Private 1 – Public  Indicates if file is public or private. Default is private
aclonly	1 – Only update ACL  Indicates that the server should only update the file's ACL instead of storing it. If this parameter is nonzero then any context in the POST is ignored. 'acl' must be specified with this parameter.

### Response

#### HTTP Response Codes

200 Ok	File was stored or ACL updated successfully
400 BAD REQUEST	Error reading data from the request or content header not set correctly
403 FORBIDDEN	Attempted to write to another user's file store.
404 NOT FOUND	Attempted to update an ACL but file was not found.

### Response XML

```
<PutFileResults/>
```

(Future versions may include more information in the XML response)

## Getting Files

A user can retrieve any of his files but can only retrieve other users' public files. The Content-Type and Content-Length headers can be used to determine the file's type and to ensure the entire file is received.

### Syntax

---

```
GET /rs/api/fs/colini/file_to_retrieve.txt HTTP/1.1
Host: colin-dt:8888
```

There are no request parameters for GET.

#### *Response*

HTTP Response Codes

200 Ok	File was retrieved and contexts returned in body of response.
403 Forbidden	Attempted to retrieve another user's private file
404 Not Found	The requested file was not found.

Response XML

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain
Content-Length: 226
Date: Mon, 17 May 2010 17:58:47 GMT
[226 bytes of data
```

### **Deleting Files**

Only the file's owner can delete the file. There is not a un-delete function.

Syntax

```
DELETE /rs/api/fs/colini/some_image.jpg HTTP/1.1
Host: aussm40-z47:9660
User-Agent: Apache-HttpClient/4.0.1 (java 1.5
```

There are no request parameters for DELETE

#### *Response*

---

### HTTP Response Codes

200 Ok	The file was deleted
403 Forbidden	Attempted to delete another user's file
404 Not Found	The file was not deleted because it was not found

### Response XML

<DeleteFileResult/>

---

## Appendix

### Data Request XSD

The content of the <DataRequest> element must be a <Query>, <GetRecsParams>, or <Worksheet> element (See Appendix Query Execution Data Request XSD, Records Data Request XSD, and Worksheet Data Request XSD).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="DataRequest" type="dataRequest"/>

  <xs:complexType name="dataRequest">
    <xs:sequence>
      <xs:any processContents="skip" namespace="##other"
minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="scale" type="xs:int"
use="required"/>
  </xs:complexType>
</xs:schema>
```

### Query Execution Data Request XSD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Query" type="query"/>

  <xs:complexType name="query">
    <xs:sequence>
      <xs:element name="Text" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

---

## Records Data Request XSD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="GetRecsParams" type="dataRecordParameterSet"/>

  <xs:complexType name="dataRecordParameterSet">
    <xs:sequence>
      <xs:element name="Rels" type="xs:string" minOccurs="0"/>
      <xs:element name="Cols" type="xs:string" minOccurs="0"/>
      <xs:element name="fd" type="xs:date" minOccurs="0"/>
      <xs:element name="td" type="xs:date" minOccurs="0"/>
      <xs:element name="ft" type="xs:time" minOccurs="0"/>
      <xs:element name="tt" type="xs:time" minOccurs="0"/>
      <xs:element name="Unit" type="Units" minOccurs="0"/>
      <xs:element name="nu" type="xs:int"/>
      <xs:element name="mdf" type="FillOption" minOccurs="0"/>
      <xs:element name="mhf" type="FillOption" minOccurs="0"/>
      <xs:element name="sn" type="SkipNaN" minOccurs="0"/>
      <xs:element name="LimitMode" type="LimitMode" minOccurs="0"/>
      <xs:element name="LimitSize" type="xs:int"/>
      <xs:element name="ctu" type="CurrentTickUsage" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="Units">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MILLISECONDS"/>
      <xs:enumeration value="SECONDS"/>
      <xs:enumeration value="MINUTES"/>
      <xs:enumeration value="HOURS"/>
      <xs:enumeration value="DAYS"/>
      <xs:enumeration value="WEEKS"/>
      <xs:enumeration value="MONTHS"/>
      <xs:enumeration value="QUARTERS"/>
      <xs:enumeration value="YEARS"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="FillOption">
    <xs:restriction base="xs:string">
      <xs:enumeration value="INVALID"/>
      <xs:enumeration value="FILL_NAN"/>
      <xs:enumeration value="FILL_FORWARD"/>
      <xs:enumeration value="FILL_BACKWARD"/>
      <xs:enumeration value="FILL_INTERP_LINEAR"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

```

    <xs:enumeration value="FILL_INTERP_GEOMETRIC"/>
    <xs:enumeration value="FILL_INTERP_LOGARITHMIC"/>
    <xs:enumeration value="FILL_NEAREST"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SkipNaN">
  <xs:restriction base="xs:string">
    <xs:enumeration value="INVALID"/>
    <xs:enumeration value="SKIP_NONE"/>
    <xs:enumeration value="SKIP_ALL_NAN"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LimitMode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="INVALID"/>
    <xs:enumeration value="BY_RECORDS"/>
    <xs:enumeration value="BY_MEMORY"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CurrentTickUsage">
  <xs:restriction base="xs:string">
    <xs:enumeration value="APPEND_TO_ALL"/>
    <xs:enumeration value="APPEND_TO_NONE"/>
    <xs:enumeration value="APPEND_TO_DAILY"/>
    <xs:enumeration value="APPEND_TO_TICK"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

## Worksheet Data Request XSD

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="DateOptions" type="dateOptions"/>

  <xs:element name="Item" type="llcItem"/>

  <xs:element name="SeasonalAnalysis" type="seasonalAnalysis"/>

  <xs:element name="Worksheet" type="worksheet"/>

```

```

<xs:element name="WorksheetOptions" type="worksheetOptions"/>

<xs:complexType name="worksheet">
  <xs:sequence>
    <xs:element ref="DateOptions" minOccurs="0"/>
    <xs:element ref="WorksheetOptions" minOccurs="0"/>
    <xs:element name="Items" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Item" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element ref="SeasonalAnalysis" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="version" type="xs:string"/>
</xs:complexType>

<xs:complexType name="dateOptions">
  <xs:sequence>
    <xs:element name="Period" type="Units" minOccurs="0"/>
    <xs:element name="NumPeriod" type="xs:int"/>
    <xs:element name="EndDateToday" type="xs:boolean"/>
    <xs:element name="EndDate" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="StartDateRelative" type="xs:boolean"/>
    <xs:element name="StartDate" type="xs:dateTime" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="worksheetOptions">
  <xs:sequence>
    <xs:element name="SortAscending" type="xs:boolean"/>
    <xs:element name="ExecutionUnits" type="Units" minOccurs="0"/>
    <xs:element name="NumExecUnits" type="xs:int"/>
    <xs:element name="TableSummaryStats" type="xs:boolean"/>
    <xs:element name="SkipWeekends" type="xs:boolean"/>
    <xs:element name="MissingDataNanFill" type="FillOption"
minOccurs="0"/>
    <xs:element name="SkipAllNans" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="llcItem">
  <xs:sequence>
    <xs:element name="ChartLabel" type="xs:string" minOccurs="0"/>

```

```

    <xs:element name="LastDataDate" type="xs:boolean"/>
    <xs:element name="CurveDate" type="xs:string" minOccurs="0"/>
    <xs:element name="Formula" type="xs:string" minOccurs="0"/>
    <xs:element name="Precision" type="xs:int"/>
    <xs:element name="Factor" type="xs:double"/>
    <xs:element name="Units" type="xs:string" minOccurs="0"/>
    <xs:element name="TimeUnits" type="DataType" minOccurs="0"/>
    <xs:element name="Description" type="xs:string" minOccurs="0"/>
    <xs:element name="Column" type="xs:string" minOccurs="0"/>
    <xs:element name="Symbol" type="xs:string" minOccurs="0"/>
    <xs:element name="Label" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="ItemType"/>
</xs:complexType>

<xs:complexType name="seasonalAnalysis">
  <xs:sequence>
    <xs:element name="seasonInfo" type="seasonsInfo" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="seasonsInfo">
  <xs:sequence>
    <xs:element name="SeasonsLocked" type="xs:boolean"/>
    <xs:element name="EndLastSeason" type="xs:dateTime"
minOccurs="0"/>
    <xs:element name="StartFirstSeason" type="xs:dateTime"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="Units">
  <xs:restriction base="xs:string">
    <xs:enumeration value="MILLISECONDS"/>
    <xs:enumeration value="SECONDS"/>
    <xs:enumeration value="MINUTES"/>
    <xs:enumeration value="HOURS"/>
    <xs:enumeration value="DAYS"/>
    <xs:enumeration value="WEEKS"/>
    <xs:enumeration value="MONTHS"/>
    <xs:enumeration value="QUARTERS"/>
    <xs:enumeration value="YEARS"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="FillOption">
  <xs:restriction base="xs:string">

```

```

    <xs:enumeration value="INVALID"/>
    <xs:enumeration value="FILL_NAN"/>
    <xs:enumeration value="FILL_FORWARD"/>
    <xs:enumeration value="FILL_BACKWARD"/>
    <xs:enumeration value="FILL_INTERP_LINEAR"/>
    <xs:enumeration value="FILL_INTERP_GEOMETRIC"/>
    <xs:enumeration value="FILL_INTERP_LOGARITHMIC"/>
    <xs:enumeration value="FILL_NEAREST"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="DataType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="TICK"/>
    <xs:enumeration value="INTRADAY"/>
    <xs:enumeration value="DAILY"/>
    <xs:enumeration value="ALL"/>
    <xs:enumeration value="MILLISECOND"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ItemType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="SERIES"/>
    <xs:enumeration value="FORMULA"/>
    <xs:enumeration value="CURVE"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

## Data Request Response XSD

```

?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="DataRequestResponse" type="dataRequestResponse"/>

  <xs:element name="Report" type="report"/>

  <xs:element name="ReportBlock" type="reportBlock"/>

  <xs:element name="ReportSummaryBlock" type="reportSummaryBlock"/>

  <xs:complexType name="dataRequestResponse">

```

```

    <xs:sequence>
      <xs:element name="Reports" type="report" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="endTime" type="xs:dateTime"/>
    <xs:attribute name="id" type="xs:long" use="required"/>
    <xs:attribute name="startTime" type="xs:dateTime"/>
    <xs:attribute name="status" type="Status" use="required"/>
    <xs:attribute name="statusMsg" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="report">
    <xs:sequence>
      <xs:element name="ReportBlocks" type="reportBlock" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="title" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="reportBlock">
    <xs:sequence>
      <xs:element name="AttributeHeadings" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="ColumnHeadingIndices" type="xs:int"
minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="ColumnHeadings" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="RowDates" type="xs:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="RowTimes" type="xs:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="SummaryReport" type="reportSummaryBlock"
minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Values" type="xs:double" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="numAttributes" type="xs:int" use="required"/>
    <xs:attribute name="numCols" type="xs:int" use="required"/>
    <xs:attribute name="numRows" type="xs:int" use="required"/>
  </xs:complexType>

  <xs:complexType name="reportSummaryBlock">
    <xs:sequence/>
    <xs:attribute name="avg" type="xs:double" use="required"/>
    <xs:attribute name="avgNegative" type="xs:double" use="required"/>
    <xs:attribute name="avgPositive" type="xs:double" use="required"/>
    <xs:attribute name="highest" type="xs:double" use="required"/>
  </xs:complexType>

```

```

<xs:attribute name="lowest" type="xs:double" use="required"/>
<xs:attribute name="pctNegative" type="xs:double" use="required"/>
<xs:attribute name="pctPositive" type="xs:double" use="required"/>
<xs:attribute name="stdDeviation" type="xs:double" use="required"/>
<xs:attribute name="sum" type="xs:double" use="required"/>
<xs:attribute name="variance" type="xs:double" use="required"/>
<xs:attribute name="zStat" type="xs:double" use="required"/>
</xs:complexType>

<xs:simpleType name="Status">
  <xs:restriction base="xs:int">
    <xs:enumeration value="100"/>
    <xs:enumeration value="110"/>
    <xs:enumeration value="120"/>
    <xs:enumeration value="200"/>
    <xs:enumeration value="300"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

## Meta Data Relation Information XSD

Contains the description, columns & data range, UOM, default column data preview information.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Conversions" type="conversionPack"/>

  <xs:element name="DataPreview" type="dataPreview"/>

  <xs:element name="RelCol" type="relCol"/>

  <xs:element name="RelInfo" type="relInfo"/>

  <xs:complexType name="relInfo">
    <xs:sequence>
      <xs:element name="RollDay" type="xs:string" minOccurs="0"/>
      <xs:element name="RollRule" type="xs:string" minOccurs="0"/>
      <xs:element name="RolloverDataType" type="RolloverDataType"
minOccurs="0"/>
      <xs:element name="ContractUnits" type="xs:float" minOccurs="0"/>
      <xs:element name="ExpirationDate" type="xs:date" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

```

```

    <xs:element name="FirstNoticeDay" type="xs:date" minOccurs="0"/>
    <xs:element name="children" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="RelInfo" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="CurrencyConversions" type="conversionPack"
minOccurs="0"/>
    <xs:element name="UnitConversions" type="conversionPack"
minOccurs="0"/>
    <xs:element name="Columns" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Column" type="relCol" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element ref="DataPreview" minOccurs="0"/>
    <xs:element name="ErrorMsg" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="RelType"/>
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="path" type="xs:string"/>
  <xs:attribute name="description" type="xs:string"/>
  <xs:attribute name="aliasTarget" type="xs:string"/>
  <xs:attribute name="defaultColumn" type="xs:string"/>
  <xs:attribute name="hasChildren" type="xs:int" use="required"/>
</xs:complexType>

<xs:complexType name="conversionPack">
  <xs:sequence>
    <xs:element name="Conversion" type="conversion" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="conversion">
  <xs:sequence>
    <xs:element name="Factor" type="conversionFactor" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="from" type="xs:string"/>
</xs:complexType>

```

```

<xs:complexType name="conversionFactor">
  <xs:sequence/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="to" type="xs:string"/>
</xs:complexType>

<xs:complexType name="relCol">
  <xs:sequence>
    <xs:element name="MsDataTo" type="xs:date" minOccurs="0"/>
    <xs:element name="MsDataFrom" type="xs:date" minOccurs="0"/>
    <xs:element name="SecDataTo" type="xs:date" minOccurs="0"/>
    <xs:element name="SecDataFrom" type="xs:date" minOccurs="0"/>
    <xs:element name="MinDataTo" type="xs:date" minOccurs="0"/>
    <xs:element name="MinDataFrom" type="xs:date" minOccurs="0"/>
    <xs:element name="DailyDataTo" type="xs:date" minOccurs="0"/>
    <xs:element name="DailyDataFrom" type="xs:date" minOccurs="0"/>
    <xs:element name="Conversion" type="conversion" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="baseUnit" type="xs:string"/>
  <xs:attribute name="baseCurrency" type="xs:string"/>
  <xs:attribute name="cDesc" type="xs:string"/>
  <xs:attribute name="cName" type="xs:string"/>
</xs:complexType>

<xs:complexType name="dataPreview">
  <xs:sequence>
    <xs:element name="Values" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Value" type="xs:double" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Dates" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Date" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="column" type="xs:string"/>
</xs:complexType>

```

```

<xs:simpleType name="RelType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="INVALID"/>
    <xs:enumeration value="CATEGORY"/>
    <xs:enumeration value="NORMAL"/>
    <xs:enumeration value="FUTURES"/>
    <xs:enumeration value="FUTURES_CONTRACT"/>
    <xs:enumeration value="FUTURES_CONTINUOUS"/>
    <xs:enumeration value="OPTIONS"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="RolloverDataType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="INVALID"/>
    <xs:enumeration value="NONE"/>
    <xs:enumeration value="DAILY"/>
    <xs:enumeration value="TICK"/>
    <xs:enumeration value="BOTH"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

## Unit of Measure Conversion Response XSD

(Unused)

## Search Response XSD

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="relation" type="relation"/>

  <xs:element name="searchList" type="searchList"/>

  <xs:complexType name="searchList">
    <xs:sequence>
      <xs:element name="relation" type="relation"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="totalSearchResults" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>

```

```

    </xs:sequence>
    <xs:attribute name="size" type="xs:int"
use="required"/>
  </xs:complexType>

  <xs:complexType name="relation">
    <xs:sequence>
      <xs:element name="description" type="xs:string"
minOccurs="0"/>
      <xs:element name="name" type="xs:string"
minOccurs="0"/>
      <xs:element name="parent" type="xs:string"
minOccurs="0"/>
      <xs:element name="type" type="xs:string"
minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

## Shortcuts Response XSD

Layered In UserGroup, User, Profile and Alias tags.

```

<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Alias" type="alias"/>

  <xs:element name="Profile" type="profile"/>

  <xs:element name="User" type="user"/>

  <xs:element name="UserGroup" type="userGroup"/>

  <xs:complexType name="userGroup">
    <xs:sequence>
      <xs:element name="User" type="user" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="pubProfilesEnabled" type="xs:boolean"
use="required"/>
  </xs:complexType>

  <xs:complexType name="user">
    <xs:sequence>

```

```

        <xs:element name="Profile" type="profile" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
</xs:complexType>

<xs:complexType name="profile">
    <xs:sequence>
        <xs:element name="Alias" type="alias" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="isPublic" type="xs:boolean" use="required"/>
    <xs:attribute name="timestamp" type="xs:string"/>
    <xs:attribute name="version" type="xs:double" use="required"/>
    <xs:attribute name="lim.xmlimhost" type="xs:string"/>
    <xs:attribute name="lim.xmlimport" type="xs:int" use="required"/>
</xs:complexType>

<xs:complexType name="alias">
    <xs:sequence>
        <xs:element ref="Alias" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="aliasPath" type="xs:string"/>
    <xs:attribute name="description" type="xs:string"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="xmimpath" type="xs:string"/>
    <xs:attribute name="xmimtype" type="xs:int" use="required"/>
</xs:complexType>
</xs:schema>

```

### Search Files Response XSD

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:element name="SearchFilesResult"
type="searchFilesResult"/>

    <xs:element name="SearchFilesResultContent"
type="searchFilesResultContent"/>

```

```

<xs:complexType name="searchFilesResult">
  <xs:sequence>
    <xs:element name="Contents"
type="searchFilesResultContent" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="searchFilesResultContent">
  <xs:sequence>
    <xs:element name="ACL" type="accessType"
minOccurs="0"/>
    <xs:element name="Created" type="xs:dateTime"
minOccurs="0"/>
    <xs:element name="Key" type="xs:string"
minOccurs="0"/>
    <xs:element name="MimeType" type="xs:string"
minOccurs="0"/>
    <xs:element name="LastModified" type="xs:dateTime"
minOccurs="0"/>
    <xs:element name="Size" type="xs:long"/>
    <xs:element name="Owner" type="xs:string"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="accessType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="PUBLIC"/>
    <xs:enumeration value="PRIVATE"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

### Search Files Response Content XSD

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="SearchFilesResultContent"
type="searchFilesResultContent"/>

```

---

```
<xs:complexType name="searchFilesResultContent">
  <xs:sequence>
    <xs:element name="ACL" type="accessType"
minOccurs="0"/>
    <xs:element name="Created" type="xs:dateTime"
minOccurs="0"/>
    <xs:element name="Key" type="xs:string"
minOccurs="0"/>
    <xs:element name="MimeType" type="xs:string"
minOccurs="0"/>
    <xs:element name="LastModified" type="xs:dateTime"
minOccurs="0"/>
    <xs:element name="Size" type="xs:long"/>
    <xs:element name="Owner" type="xs:string"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="accessType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="PUBLIC"/>
    <xs:enumeration value="PRIVATE"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>
```