

1010data

PowerLoader Application Program Interface Reference Manual

Version 1

0 1 1 1 0 1 1 1 1 1 1 0 1
0 1 1 0 0 1 0 0 1 1 0 1 1
0 1 1 1 1 0 0 0 1 1 1 0 0
1 1 0 1 0 1 0 0 1 0 1 1 0
0 1 1 1 1 0 0 0 1 0 0 0 1
1 0 0 1 0 0 1 1 1 0 1 0 1
0 0 1 0 1 0 0 1 1 1 1 1 0
1 1 0 1 1 0 1 1 1 1 0 0 1
0 0 0 0 0 0 0 1 0 0 1 1 1
0 1 1 1 1 0 0 1 0 1 0 1 0
1 0 1 1 0 1 1 0 1 1 0 0 1
0 0 0 0 0 0 1 0 0 1 0 1 1
0 0 1 0 0 1 0 0 1 1 0 0 0
1 1 0 0 1 1 0 1 0 1 1 1 1
0 1 1 0 1 0 1 1 0 0 0 0 0
0 0 0 1 0 0 1 0 0 1 1 0 1
0 0 0 1 0 0 1 0 0 1 1 1 1
0 0 0 0 0 0 1 0 1 0 1 0 1
1 1 1 0 0 0 1 1 0 1 1 0 0
0 0 1 0 1 1 0 1 0 1 0 0 1
0 1 0 0 1 0 **1 0 1 0** 0 0 1
1 1 0 0 1 0 0 0 1 0 0 1 1
0 1 0 1 1 0 1 1 0 1 0 0 0
0 0 1 0 1 1 0 1 0 1 1 0 0
0 1 1 1 0 1 0 0 0 0 1 0 0
0 1 1 1 0 1 0 1 0 0 0 1 1
1 0 1 0 0 0 0 1 1 1 0 0 1
0 0 1 0 0 0 1 1 1 1 1 1 1
0 0 1 0 0 1 1 0 1 1 1 0 0
0 1 0 1 0 0 0 1 1 0 1 1 0
0 0 1 1 0 1 0 0 1 0 0 1 0
0 1 0 0 1 0 0 1 1 0 1 1 1
1 0 0 1 0 0 0 1 1 0 0 1 1
1 1 0 0 1 1 1 1 1 1 1 1 1
0 0 0 0 0 1 0 1 1 1 0 1 1
1 0 0 1 0 1 0 1 1 1 1 1 0
1 1 1 1 0 0 1 1 1 0 0 1 1
1 1 1 0 0 0 0 0 0 1 1 0 1
0 0 1 0 0 1 1 0 1 0 1 1 1
0 0 1 1 1 1 0 0 0 0 0 0 0
1 0 0 1 1 0 0 0 0 0 0 1 0
0 1 1 1 1 0 0 0 1 1 0 1 1
0 1 0 1 0 0 1 0 1 0 0 1 1
1 0 0 0 0 0 0 0 1 0 1 1 1
0 1 1 1 0 0 0 0 0 0 0 1 0
1 1 0 0 0 1 0 0 0 0 1 0 1
0 0 1 0 1 1 1 1 0 1 0 0 1 0
1 0 1 1 0 1 1 0 0 0 1 1 0
0 0 1 0 0 1 1 0 1 0 1 1 1

Release 20061110

© Copyright 2001-2006 1010data, Inc.

Preface

The 1010data PowerLoader Application Program Interface (API) is a facility whereby an application running on a user's machine or server can upload large sets of data to 1010data. The API uses XML-RPC over HTTP for transactions and FTP for file uploads. It is compatible with any client application written in a language that is network aware and supports HTTP transactions (such as Java, Visual Basic, C++, Python and PERL.)

Prerequisites

To understand and use the API, working knowledge of the 1010data database service, programmatically dealing with FTP, flat files and XML is assumed. It should also be noted that the PowerLoader API is very similar to the 1010 Interface API and all common transactions share the same inputs and outputs. The difference is that folder, permission and table changes made by other users will be visible during your API session.

To use the API to load tables, a valid 1010data username and password is required, the username must be authorized for API use and have an FTP account with enough file system space available to store the entire table.

Transactions

Transactions through the API are a series of HTTP(S) POSTs operations. The client-to-server message consists of a header followed by contents, where the two are separated by "\r\n\r\n". Among other things the header must contain a query string that specifies the type of API transaction and certain information that identifies the user. The contents, in XML format, contains the transaction details and possibly other data. (For some transactions, the contents may be empty.) An example of such a message is,

```
POST /cgi-bin/powerload?api=dir&apiversion=1&uid=dhorowitz&pswd=6hxi2xsgikgd&sid=402910118
HTTP/1.0\r\nContent-Type: text/xml\r\nContent-Length: 16\r\nHost:
test.1010data.com\r\n\r\n<name>pub</name>
```

or more legibly,

```
POST /cgi-bin/powerload?api=dir&apiversion=2&uid=dhorowitz&pswd=6hxi2xsgikgd&sid=402910118
HTTP/1.0
Content-Type: text/xml
Content-Length: 16
Host: test.1010data.com

<name>pub</name>
```

Note that the query string on the first line specifies that this is a "dir" transaction and includes the user's username (uid) and password (pswd). **The message must always contain a *lower-case* "http/1.0", "https/1.0", "http/1.1" or "https/1.1" and the Content-Length must be set to the length (number of characters) of the XML contents.**

The response from the server is also in the form of a header followed by XML contents. An example of such a message is,

```
HTTP/1.1 200 OK
Date: Thu, 09 Nov 2006 16:14:05 GMT
Server: Apache
Content-Length: 295
Connection: close
Content-Type: text/xml

<rc>0</rc>
<msg>dir successful</msg>
<table>
<cols>
<th name=\"name\" type=\"a\">name</th>
<th name=\"type\" type=\"a\">type</th>
```

```
</cols>
<data>
<tr>
<td>pub.geo</td>
<td>dir</td>
</tr>
<tr>
<td>pub.fin</td>
<td>dir</td>
</tr>
<tr>
<td>pub.demos</td>
<td>dir</td>
```

The information in the contents depends on the type of transaction, but it always includes the following two elements:

```
<rc> return code (0 means OK) </rc>
<msg> message </msg>
```

The query string and XML input and output for each transaction are described in detail in the next section.

**Transaction
Details**

The `login` transaction starts a session and must precede all other transactions.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "login".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	User password

The query string may also contain the following optional parameter:

<code>kill</code>	"yes" or "no"
-------------------	---------------

If `kill=no` and a session already exists for the specified username (i.e. the user is already logged in), an error message is returned (see "XML Response from Server" below.) If `kill=yes` or if `kill` is not specified, a new session is started and any existing session is logged out.

Example

```
api=login&apiversion=1&uid=myid&pswd=myspwd&kill=yes
```

XML Input to Server

None.

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message
<code><sid></code>	Session ID
<code><pswd></code>	Encrypted password

The session ID and encrypted password are used in subsequent transactions. If the login is not successful, only the return code and error message are returned.

Example

```
<rc>0</rc>
<sid>1748453843</sid>
<pswd>Aiyynnsewxhck</pswd>
<msg>Login Successful</msg>
```

The `logout` transaction ends the session.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "logout".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=logout&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

None.

XML Response from Server

A successful `logout` produces the following result:

```
<rc>0</rc>  
<msg>Logged out</msg>
```

The `dir` transaction returns a listing of the contents of a directory (folder.)

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "dir".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=dir&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

`<name>` The full path to the directory in the 1010data database hierarchy. In general, a path has the form *directory1 . directory2 directoryn . thisdirectory* (similar to a Windows or Unix file-system path but with dots instead of slashes.)

Example

```
<name>mycompany.folder</name>
```

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message
<code><table></code>	Directory listing

The directory listing is returned in the form of a table (see "Table Tree") with one row for each entry and two columns. The first column contains the name of the entry and the second indicates whether it is a directory (`dir`) or a table (`tab`). In the event of an error only the return code and error message are returned.

Example

```
<rc>0</rc>
<msg>dir successful</msg>
<table>
  <cols>
    <th name="name" type="a">name</th>
    <th name="type" type="a">type</th>
    <th name="title" type="a">title</th>
  </cols>
  <data>
    <tr>
      <td>mycompany.folder.table1</td>
      <td>tab</td>
      <td>table1</td>
    </tr>
    <tr>
      <td>mycompany.folder.table2</td>
      <td>tab</td>
      <td>table2</td>
    </tr>
  </data>
</table>
```

</tr>

```
<tr>
  <td>mycompany.folder.table3</td>
  <td>tab</td>
  <td>table3</td>
</tr>
<tr>
  <td>mycompany.folder.table4</td>
  <td>tab</td>
  <td>table4</td>
</tr>
</data>
</table>
```

The `tabinfo` transaction returns information about a table.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "tabinfo".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=tabinfo&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

`<name>` The full path to the table in the 1010data database hierarchy. In general, a path has the form *directory1 . directory2 directoryn . table* (similar to a Windows or Unix file-system path but with dots instead of slashes.) The name of a table can be determined by clicking on menu item Help/About this Table in the 1010data user interface.

Example

```
<name>mycompany.folder.hourly95</name>
```

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message
<code><table></code>	Information about table

The `<table>` element (see "Table Tree") contains information about the table. It is equivalent to the table with zero rows of data (i.e. the `<data>` element is empty.) In the event of an error only the return code and error message are returned.

Example

```
<rc>0</rc>
<msg>tabinfo successful</msg>
<table name="mycompany.folder.hourly95">
  <title>Hourly U.S. Weather (1995)</title>
  <sdesc>Hourly United States weather observations</sdesc>
  <ldesc>This table contains hourly observations at 262 weather stations over a one year
    period (1995). Each row contains the data for a particular station at a particular
    hour of a particular day. (Note that there are 262 stations and 365 days in 1995,
    so there should be 2,295,120 rows (262 stations x 365 days x 24 hours) in this
    table. There are actually somewhat fewer because the data begins with 1:00 AM on
    1/1/95 and ends with 11:00 PM on 12/31/95, so there is one hour missing.) Each
    station is identified by a ID that corresponds to the ID in the
    &quot;Stations&quot; table.</ldesc>
  <link></link>
  <cols>
    <th name="id" type="i" format="type:nocommas;width:5">
      Station#10;ID</th>
```

```
<th name="date" type="i" format="type:date">
  Date</th>
<th name="hour" type="i" format="type:num;width:4;dec:0">
  Hour</th>
<th name="asos" type="i" format="type:num;width:3;dec:0">
  ASOS&#10;Flag</th>
<th name="grad" type="i" format="type:num;width:6;dec:0">
  Global&#10;Radiation&#10;(0.1 w/m2)</th>
<th name="drad" type="i" format="type:num;width:6;dec:0">
  Direct&#10;Radiation&#10;(0.1 w/m2)</th>
<th name="tsky" type="i" format="type:num;width:3;dec:0">
  Total&#10;Sky&#10;Cover</th>
<th name="osky" type="i" format="type:num;width:4;dec:0">
  Opaque&#10;Sky&#10;Cover</th>
<th name="temp" type="f" format="type:num;width:8;dec:1">
  Dry Bulb&#10;Temp&#10;(Celsius)</th>
<th name="tflag" type="i" format="type:num;width:4;dec:0">
  Dry Bulb&#10;Temp&#10;Flag</th>
<th name="dew" type="f" format="type:num;width:8;dec:1">
  Dew Point&#10;Temp&#10;(Celsius)</th>
<th name="hum" type="i" format="type:num;width:5;dec:0">
  Relative&#10;Humidity&#10;(%)</th>
<th name="pres" type="i" format="type:num;width:6;dec:0">
  Station&#10;Pressure&#10;(millibars)</th>
<th name="pflag" type="i" format="type:num;width:3;dec:0">
  Station&#10;Pressure&#10;Flag</th>
<th name="wdir" type="i" format="type:num;width:6;dec:0">
  Wind&#10;Direction&#10;(degrees)</th>
<th name="wspd" type="f" format="type:num;width:4;dec:0">
  Wind&#10;Speed&#10;(m/s)</th>
<th name="vis" type="f" format="type:num;width:6;dec:1">
  Visibility&#10;&#10;(km)</th>
<th name="ceil" type="i" format="type:num;width:6;dec:0">
  Ceiling&#10;Height&#10;(meters)</th>
<th name="weath" type="i">
  Present&#10;Weather</th>
<th name="asos1" type="i" format="type:num;width:6;dec:0">
  ASOS&#10;Cloud&#10;Layer 1</th>
<th name="asos2" type="i" format="type:num;width:6;dec:0">
  ASOS&#10;Cloud&#10;Layer 2</th>
<th name="asos3" type="i" format="type:num;width:6;dec:0">
  ASOS&#10;Cloud&#10;Layer 3</th>
<th name="prec" type="i" format="type:num;width:7;dec:0">
  Hourly&#10;Precip&#10;(0.01 inch)</th>
<th name="rflag" type="a" format="type:char;width:-2">
  Hourly&#10;Precip&#10;Flag</th>
<th name="snow" type="i" format="type:num;width:4;dec:0">
  Snow&#10;Depth&#10;(inches)</th>
</cols>
<data/>
</table>
```

The `mkdir` transaction creates a directory. The API will check the permission of the user and create the directory only if the user calling the transaction is authorized to create the directory in the parent folder.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "mkdir".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=mkdir&apiversion=1&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following elements:

<code><name></code>	The full path to the directory in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 directoryn . thisdirectory</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.)
<code><users></code>	Users authorized to browse the directory. See User Tree specification for the XML schema.
<code><uploaders></code>	Users authorized to add content to the directory. See User Tree specification for the XML schema.
<code><title></code>	The title of the directory.

Example

```
<name>mycompany.folder.yearly</name>
<users type="list">
  <user>user1</user>
  <user>user2</user>
</users>
<uploaders type="list">
  <user>user2</user>
</uploaders>
<title>Yearly Data</title>
```

XML Response from Server

A successful `mkdir` produces the following result:

```
<rc>0</rc>
<msg>mkdir successful</msg>
```

The `rmdir` transaction removes a directory. The API will check the permission of the user and the directory and it will remove the directory only if the user is an owner of the parent directory. Finally, `rmdir` will only remove the directory if it is empty so be sure to remove all child tables and folders before invoking this transaction.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "rmdir".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=rmdir&apiversion=1&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

`<name>` The full path to the directory in the 1010data database hierarchy. In general, a path has the form *directory1 . directory2 directoryn . thisdirectory* (similar to a Windows or Unix file-system path but with dots instead of slashes.)

Example

```
<name>mycompany.folder.yearly</name>
```

XML Response from Server

A successful `mkdir` produces the following result:

```
<rc>0</rc>  
<msg>rmdir successful</msg>
```

The `mmdir` transaction moves a sub-directory from one folder to another. The API will check the permission of the user and the directory. It will move the directory if the user is an owner of the directory and is authorized to add contents to the destination folder.

Note: This transaction is yet not available.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "mmdir".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=mmdir&apiversion=1&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

<code><name></code>	The full path to the sub-directory in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 directoryn . thisdirectory</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.)
<code><dest></code>	The full path to the destination directory in the 1010data database hierarchy.

Example

```
<name>mycompany.folder.yearly</name>
<dest>mycompany.folder_updated.yearly</dest>
```

XML Response from Server

A successful `mmdir` produces the following result:

```
<rc>0</rc>
<msg>mmdir successful</msg>
```

The `orderdir` transaction will re-order the contents of a directory. The API will check the permission of the user and the directory. It will re-order the contents of the directory if the user is an owner of the directory

Note: This transaction is not available.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "orderdir".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=orderdir&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

XML Response from Server

A successful `orderdir` produces the following result:

```
<rc>0</rc>
<msg>orderdir successful</msg>
```

The `addtab` transaction loads a table into 1010data. The transaction expects source data in your ftp account so ensure the data is fully uploaded via FTP before invoking this transaction.

As input, `addtab` accepts an XML version of the 1010 PowerLoader V1 spec file. If you still wish to use the legacy format than use the `convert` transaction to convert your PowerLoader V1 spec file into an XML spec.

This transaction can be invoked synchronously or asynchronously or with the `<sync>` tag. In sync mode the transaction returns when the table is finished loading. In async mode, when the transaction returns a response, it does not mean the table is created, it means table creation has successfully been initiated. With async mode, you use the `status` transaction to check on the status of the table creation.

Sync mode is recommended for small tables (<1 million records) and async (> 1 million records) mode is recommended for large tables.

Note: To use this transaction, you must have a 1010data FTP account.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "addtab".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=addtab&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications consist of the following elements (mandatory elements in **bold**):

<code><sync></sync></code>	1 or 0 if you want the table to load synchronously or asynchronously.
<code><spec></code>	Outer specification element
<code><source></code>	List of source Files
<code><file></file></code>	File Element
<code><file></file></code>	
<code><file></file></code>	
<code><file></file></code>	
...	
<code></source></code>	
<code><name></name></code>	File path of table in 1010data ex. mycompany.mytable . If you reference an existing table <code>addtab</code> will replace it.
<code><title></title></code>	Title as it appears in the user interface
<code><sdesc></sdesc></code>	Short description of the table
<code><ldesc></ldesc></code>	Long description of table

<code><link></link></code>	Link header
<code><users type="list"></code>	Users provisioned for table access. See User Tree for XML schema.
<code><user></user></code>	- Excluding this element will set the table to inherit access rights.
<code><user></user></code>	
<code><user></user></code>	
...	
<code></users></code>	
<code><rectype></rectype></code>	fixed or separated
<code><sep></sep></code>	Single character delimiter. Required if rectype is separated.
<code><eor></eor></code>	End of record delimiter
<code><arch></arch></code>	byte order of data - little or big endian
<code><begbytes></begbytes></code>	Number of bytes to skip at the beginning of e/file
<code><begrecs></begrecs></code>	Number of records to skip in the beginning of e/file
<code><numrecs></numrecs></code>	Number of records to load in e/file
<code><ts></ts></code>	Time-series flag. 1 or 0 for true or false. Requires that <code><bord></code> tags are provided in at least one column.
<code><cols></code>	Column specifications – List of <code><col></code> elements, one for each column in the file.
<code><col></code>	
<code><name></name></code>	Field name – Must begin w/alpha
<code><width></width></code>	Field width in the raw file (Optional if rectype is separated)
<code><help></help></code>	Column Help (text inside ? marker above col.)
<code><sdesc></sdesc></code>	Short Description displayed in
<code><skip></skip></code>	Field is read but not displayed in table.
<code><type></type></code>	Field data type – Refer to data types section
<code><scale></scale></code>	Divisor used to divide raw value.
<code><case></case></code>	Force case, either “upper” or “lower”
<code><nowrite></nowrite></code>	Ignore the field, don’t include it in the table.
<code><order></order></code>	Column placement relative to other columns (integer)
<code><fix></fix></code>	Column is non-scrolling (pink)
<code><head></head></code>	Column header use ` to separate lines
<code><exp></exp></code>	1010 expression applied to raw data. (not nested)
<code><format></code>	Formatting Spec – Refer to Format types
	<code><type></type></code>
	<code><width></width></code>
	<code><dec></dec></code>
<code></format></code>	
<code><bord></bord></code>	TS segmentation order (integer). The number provided here is the relative segmentation order of this column in relation to other columns with the <code><bord></code>

tag. Think of bord columns as tabulation breaks. The column that denotes time should have the largest number in the sequence.

```
</col>
```

```
...
```

```
<cols>
```

```
</spec>
```

Example

```
<spec>
  <source>
    <file>mytable20060101.txt</file>
    <file>mytable20060201.txt</file>
  </source>
  <name>mycompany.mytable</name>
  <title>Mytable 2006</title>
  <sdesc>short description of the table</sdesc>
  <ldesc>long description of table</ldesc>
  <link>FOO</link>
  <users>
    <user>user1</user>
    <user>user2</user>
  </users>
  <rectype>fixed</rectype>
  <eor>crlf</eor>
  <cols>
    <col>
      <field>deal</field>
      <width>12</width>
      <head>Deal ID</head>
      <type>int</type>
      <format>
        <type>nocommas</type>
        <width>3</width>
        <dec>0</dec>
      </format>
    </col>
    <col>
      <field>date</field>
      <width>55</width>
      <head>Distributaion Date</head>
      <type>yyyymmdd</type>
      <format>
        <type>date</type>
        <width>8</width>
      </format>
    </col>
    <col>
      <field>loan</field>
      <width>21</width>
      <head>Loan`Number</head>
      <type>int</type>
      <exp>loan+2</exp>
    </col>
  </cols>
</spec>
```

XML Response from Server

A successful addtab produces the following result:

```
<rc>0</rc>
<msg>addtable successful</msg>
<name>mycompany.mytable</name>
```

The `rmtab` transaction removes a table. The API will check the permission of the user and will remove the table if the user is an owner of the table.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "rmtab".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=rmtab&apiversion=1&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

`<name>` The full path to the table in the 1010data database hierarchy. In general, a path has the form *directory1 . directory2 directoryn . thistable* (similar to a Windows or Unix file-system path but with dots instead of slashes.)

Example

```
<name>mycompany.folder.yearly.tab</name>
```

XML Response from Server

A successful `rmtab` produces the following result:

```
<rc>0</rc>  
<msg>rmtab successful</msg>
```

The `movetab` transaction moves a table from one folder to another. The API will check the permission of the user and the table. It will move the table only if the user is an owner of the table and is authorized to add contents to the destination folder.

Note: This transaction is not yet available.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "movetab".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=movetab&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications must contain the following element:

<code><name></code>	The full path to the table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 directoryn . thistable</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.)
<code><dest></code>	The full path to the directory in the 1010data database hierarchy.

Example

```
<name>mycompany.folder.yearly.tab</name>
<dest>mycompany.folder.yearly.newfolder</dest>
```

XML Response from Server

A successful `movetab` produces the following result:

```
<rc>0</rc>
<msg>movetab successful</msg>
```

The `status` transaction reports the status of an `addtab` transaction. The return code indicates the state of the load process.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "status".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=status&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

None

XML Response from Server

A successful result contains the following elements (required elements in bold):

<rc>	Return code
<msg>	Message
<code><numrecs></code>	Number of records written to the table so far. Reported if <code>addtab</code> is in the Loading state.
<code><totrecs></code>	Number of records in the source file.

Possible return Codes

- 0 - Idle – Server is idle and has not started loading anything.
- 1 - Failed – Server could not load file.
- 2 - Initializing – Server is validating load specification. Loading has not commenced.
- 3 - Loading – Server is in the process of loading a table. More tables cannot be loaded until it is complete.
- 4 - Completed – Server has completed an loading the table.
- 5 - Diagnosed – The server loaded the table but there were file format issues. The server attempted to automatically correct the issues.

Example

```
<rc>4</rc>  
<msg>Completed</msg>
```

The `validate` transaction will parse and validate the XML spec input to `adddtab`.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "validate".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=validate&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

Same as `adddtab`.

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message

Example

```
<rc>0</rc>  
<msg>validate successful</msg>
```

The `convert` transaction will parse and convert a legacy PowerLoader spec file into XML.

Note: The user must have a FTP account with a spec file in the top level directory.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "convert".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=convert&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications consist of the following elements:

<code><name></code>	The spec filename to convert located on your FTP account. See PowerLoader V1 GUI manual for file format.
---------------------------	--

XML Response from Server

A successful result contains the same XML

<code><rc></code>	Return code
<code><msg></code>	Message
<code><spec></code>	XML Spec file – See <code>addtab</code> transaction XML Input.

Example

```
<rc>0</rc>
<msg>validate successful</msg>
<spec>
  <source>
    <file>performance2001-2005.txt</file>
    <file>performance2006.txt</file>
  </source>
  <name>pub.demos.foo.performance</name>
  <title>Performance 2001-2006</title>
  <sdesc>short description of the table</sdesc>
  <ldesc>long description of table</ldesc>
  <link>FOO</link>
  <users>
    <user>user1</user>
    <user>user2</user>
  </users>
  <rectype>fixed</rectype>
  <eor>crlf</eor>
  <cols>
    <col>
      <field>deal</field>
      <width>12</width>
      <head>Deal ID</head>
      <type>int</type>
      <format>
      <type>nocommas</type>
```

```

        <width>3</width>
        <dec>0</dec>
        </format>
    </col>
    <col>
        <field>date</field>
        <width>55</width>
        <head>Distributaion Date</head>
        <type>yyyymmdd</type>
        <format>
            <type>date</type>
            <width>8</width>
            </format>
        </col>
    <col>
        <field>loan</field>
        <width>21</width>
        <head>Loan Number</head>
        <type>int</type>
        <exp>loan+2</exp>
    </col>
</cols>
</spec>

```

The `edittab` transaction changes the table Meta data (table information acquired via `tabinfo`).

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "edittab".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=edittab&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications consist of the `table` element from the XML response of `tabinfo`.

The difference is that one can opt to only transmit the top level elements being updated. See the `tabinfo` response specification for detailed info. The following element is required.

`<table name="path.to.table">` The `table` element is required and must contain at least one of the following. See `tabinfo` XML output for a detailed schema.

<code><title></code>	Table Title
<code><secure></code>	Secure SSL access
<code><owner></code>	Table owner
<code><ldesc></code>	Long description of table
<code><link></code>	Link Header
<code><cols></code>	Column information

Example

```
<table name="mycompany.mytable">  
  <title>Mytable's new Title</title>  
</table>
```

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message

Example

```
<rc>0</rc>  
<msg>edittab successful</msg>
```

The `search` transaction changes the table Meta data.

Note: This transaction is not available.

Query String

The query string in the HTTP header must contain the following parameters:

<code>api</code>	Must be "search".
<code>apiversion</code>	Must be "1".
<code>uid</code>	Username
<code>pswd</code>	Encrypted password (from the result of the <code>login</code> transaction)
<code>sid</code>	Session ID (from the result of the <code>login</code> transaction)

Example

```
api=search&apiversion=1&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

XML Input to Server

The specifications consist of the following element:

<code><name></code>	Search string
---------------------------	---------------

XML Response from Server

A successful result contains the following elements:

<code><rc></code>	Return code
<code><msg></code>	Message
<code><msg></code>	Search Results

Example

```
<rc>0</rc>  
<msg>search successful</msg>
```

Table Tree

Overview

The **table tree** is a generic way of representing a table in XML and is used in transmitting table data or other tabular information to and from the server. The overall format is,

```
<table name="table name">
  <title>table title</title>
  <sdesc>short description</sdesc>
  <ldesc>long description</ldesc>
  <link>link header</link>
  <maxdown>download limit</maxdown>
  <cols>
    <th name="name of first column"
      type="data type of first column"
      format="data format of first column">
      heading of first column
    </th>
    <th name="name of second column"
      type="data type of second column"
      format="data format of second column">
      heading of second column
    </th>
    :
    <th name="name of last column"
      type="data type of last column"
      format="data format of last column">
      heading of last column
    </th>
  </cols>
  <data>
    <tr>
      <td>value of first row of first column</td>
      <td>value of first row of second column</td>
      :
      <td>value of first row of last column</td>
    </tr>
    <tr>
      <td>value of second row of first column</td>
      <td>value of second row of second column</td>
      :
      <td>value of second row of last column</td>
    </tr>
    :
    <tr>
      <td>value of last row of first column</td>
      <td>value of last row of second column</td>
      :
      <td>value of last row of last column</td>
    </tr>
  </data>
</table>
```

The component elements are described in detail in this section. Not all components are used in all transactions. See the description of a transaction for information as to which components are relevant to that transaction.

Attributes

None.

Contents

<cols> is a wrapper for column meta-information and must contain one <th> element for each column in the table (see "<th>".)

Attributes

None.

Contents

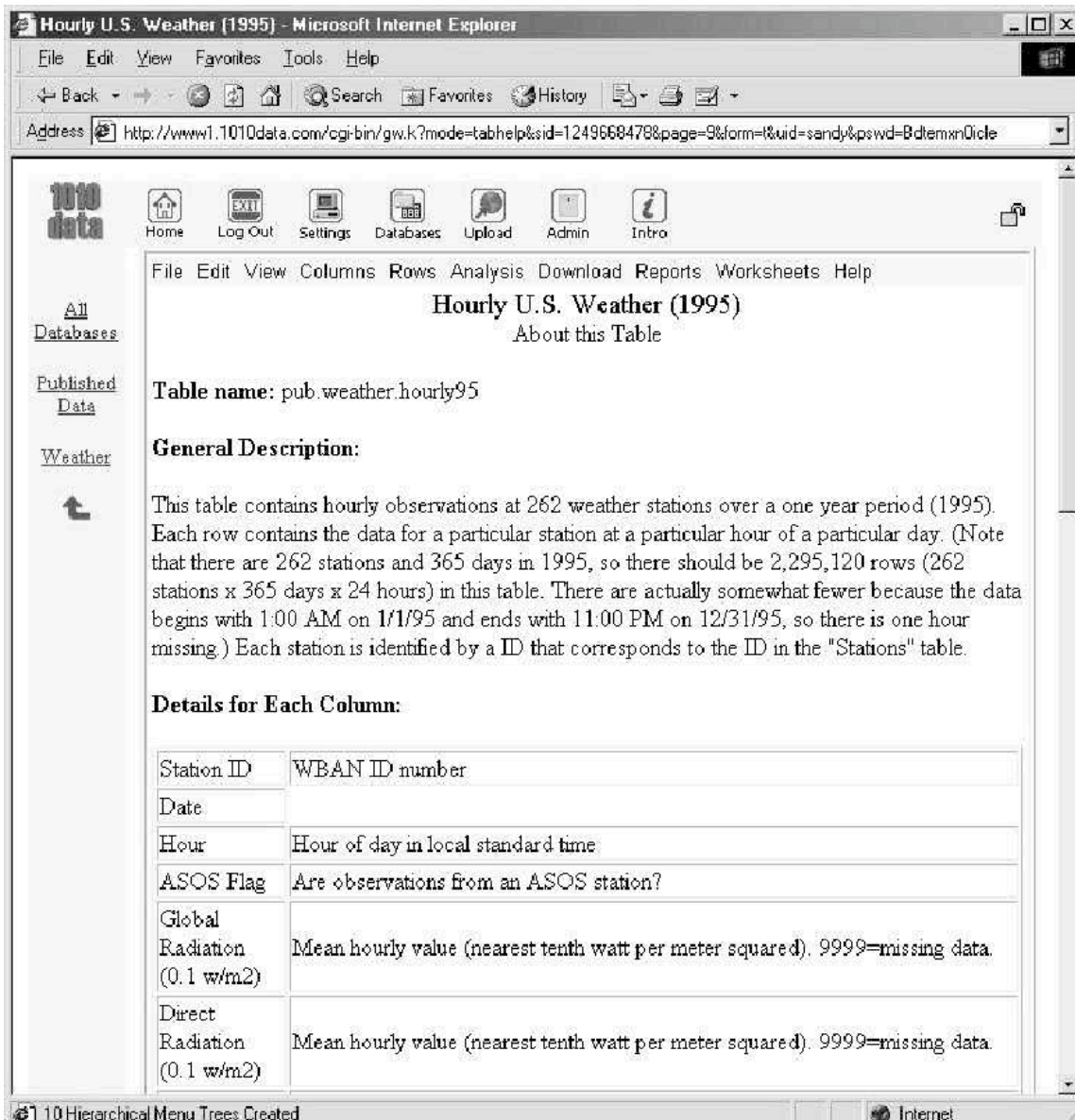
`<data>` is a wrapper for the table data and must contain one `<tr>` element for each row of the table (see "`<tr>`".)

Attributes

None.

Contents

The long description of the table, i.e. the text that appears when the user clicks on menu item Help/About this Table in the 1010data user interface. For example, in the screen shot below, the text that appears under the heading "General Description" is the long description.



Attributes

None.

Contents

A short (one or two word) title that is used when the table is linked to another table. In a link, this title is prepended to the heading of each column from this table to distinguish them from columns from the other table. For example, in the screen shot below, the Stations table was linked to the Hourly U.S. Weather table, with the last three columns from the Stations table. The heading of those columns therefore begins with "Station", which is the link header of the Stations table.



Attributes

None.

Contents

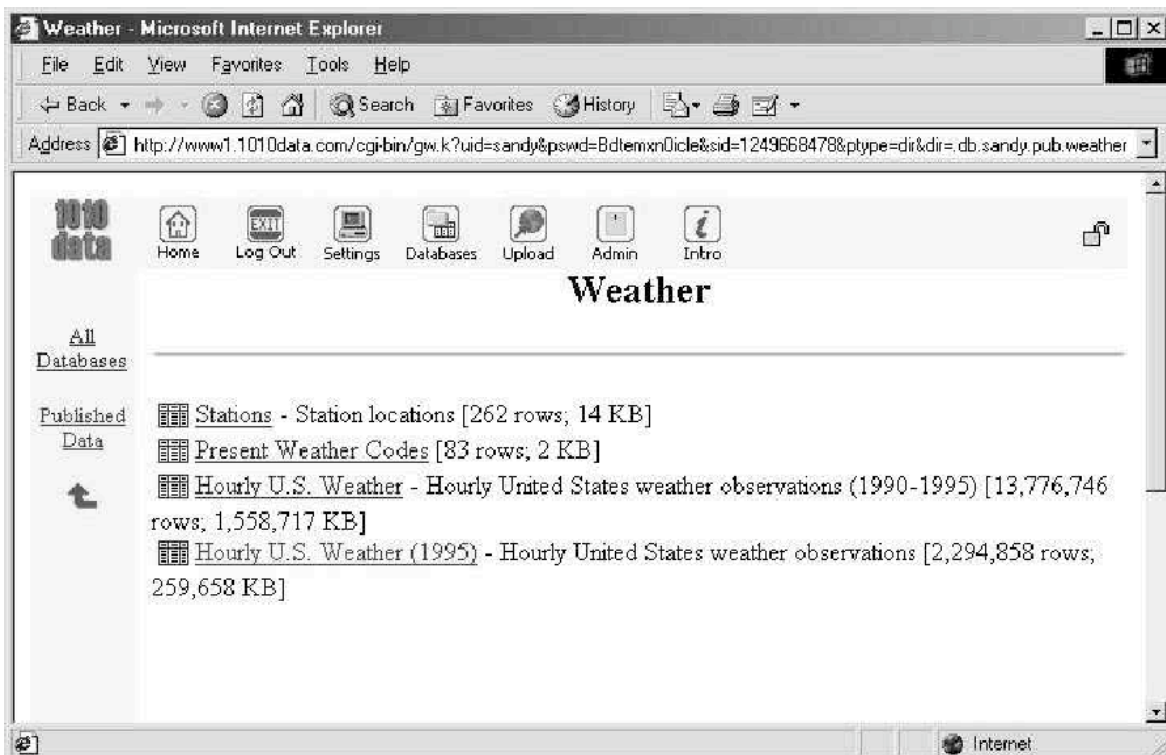
The maximum number of data items (rows \times columns) that a user of the 1010data user interface may download at one time. The download limit must be a nonnegative whole number.

Attributes

None.

Contents

The short description of the table, i.e. the text that appears next to the title in a directory listing in the 1010data user interface. For example, in the screen shot below, "Station locations" and "Hourly United States weather observations" are short descriptions.



<table> is the top-level wrapper for the table tree and is required. It may have one attribute (`name`) and must contain certain elements.

Attribute

■ `name` Name of table

If the table corresponds to a table in the 1010data system, this is the full path to the table in the 1010data database hierarchy. In general, a name has the form *directory1 . directory2 directoryn . table* (similar to a Windows or Unix file-system path but with dots instead of slashes.) The name of a table can be determined by clicking on menu item **Help/About this Table** in the 1010data user interface. Note the distinction between a table's name and its title (see "<title>".) If the table does not correspond to a table in the 1010data system, `name` has no meaning and is omitted.

Example: `mycompany.folder.hourly95`

Contents

Each of the following elements is described in detail on its own page. `<cols>` and `<data>` are always present; the others may be omitted.

<code><title></code>	Title of table
<code><sdesc></code>	Short description of table
<code><ldesc></code>	Long description of table
<code><link></code>	Link header
<code><maxdown></code>	Download limit
<code><cols></code>	Column meta information
<code><data></code>	Table data

Attributes

None.

Contents

The value of a single cell (column/row) of the table. The type of value (integer, float or alphanumeric) must conform to the column type as specified in the corresponding `<th>` entry under `<cols>`.

Attributes

■ **name** **Name of column**

The name by which the column is referenced in formulas and query operations. Note that this is generally not the same as the column's heading (see "Contents" below.)

Example: `street`

■ **type** **Data type**

The type of data in the column. `type` must be one of "i" (integer), "f" (float), or "a" (alphanumeric).

Example: `a`

■ **format** **Data format**

The display format for the data in the column. Format specifications have the same form as is used in the XML macro language used on the **Edit Actions** page of the 1010data user interface.

Example: `type:num;width:15;dec:2`

`name` and `type` must be present; `format` is optional.

Contents

The column heading that appears above the column when the table is displayed in the 1010data user interface. Lines in the heading are separated by "
" (the new line character.)

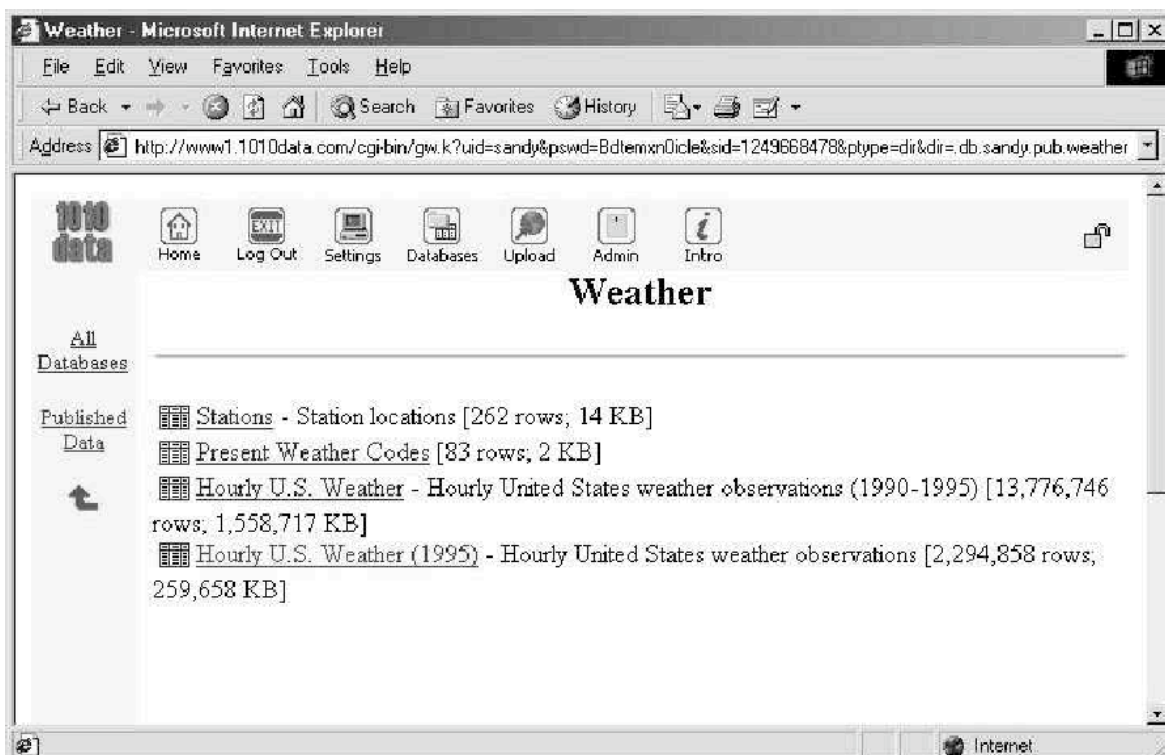
Example: `Street
Address`

Attributes

None.

Contents

The title of the table, i.e. the text that appears in both a directory listing and above the table when it is displayed in the 1010data user interface. For example, in the screen shot below, "Stations" and "Present Weather Codes" are table titles.



Attributes

None.

Contents

<tr> specifies a row of table data. It must contain one <td> element for each column of the table, each of which contains the value for that column. The first <td> specifies a value for the first column listed in the <cols> element, the second <td> specifies a value for the second column, and so on. (See "<cols>" and "<td>".)

Users Tree

Overview

The **users tree** is a way of representing access rights and is used in setting the access rights to a table that is saved or uploaded. The overall format is one of,

```
<users type="inherit"/>
```

or

```
<users type="private"/>
```

or

```
<users type="list">
  <user>username of first user</user>
  <user>username of second user</user>
  :
  <user>username of last user</user>
</users>
```

The component elements are described in detail in this section.

<users> is the top-level wrapper for the users tree. It may have one attribute (`type`) and, depending on the value of that attribute, may contain multiple instances of the <user> element.

Attribute

■ `type` Access-specification type

The type of access specification. `type` may have one of the following values:

- `private` No user, other than the table's owner, may access the table.
- `list` Only the table's owner and users listed in the users tree may access the table.
- `inherit` All users that have access to the table's parent directory may access the table.

A user with access to a table's parent directory, but not to the table itself, will not see the table in the directory's listing.

Note that a user's access to any table is subject to the user's "maximum row limit". The largest table (in terms of number of rows) that a user may access is dependent on the user's subscription level. If a user is granted access to a table via the users tree, but the table has more rows than the user's maximum row limit, the user will see the table's title in the directory listing but will be unable to access it.

`type` is optional. If omitted, the default is `list`.

Contents

If `type="private"` or `type="inherit"`, <users> must be empty. If `type="list"`, <users> must contain one or more <user> elements, each identifying a single user or group.

Attributes

None.

Contents

A username or group name. The name must refer to an existing user or group.

Field and Format Types

Field Data Types

<type>

A field's data type may be one of the following:

text	Text (Picture)
int	Integer (Picture)
float	Float (Picture)
1Btext	1-Byte Char (Binary)
1Bint	1-Byte Int (Binary)
2Bint	2-Byte Int (Binary)
4Bint	4-Byte Int (Binary)
8Bint	8-Byte Int (Binary)
4Bfloat	4-Byte Float (Binary)
8Bfloat	8-Byte Double (Binary)
dec	Decimal (BCD)
Signed	Signed (= Zoned)
yyyymmdd	Date
yymmdd	Date
mmdyy	Date
ddmmyy	Date
mmdyyyyy	Date
ddmmyyyy	Date
yyyymm	Month/Year
yymm	Month/Year
mmyy	Month/Year
mmyyyy	Month/Year

All non-numeric characters in dates are ignored, e.g. the following are equivalent:

"31505"

"3/15/05"

"3-15-05"

Input types yyyymmdd and yyyymm are tolerant of decoration characters, e.g. "2005-03-15" is the same as "20050315".

Upon load, dates are automatically converted to yyyymmdd format, which is the format 1010data uses. Similarly, month/years are converted to yyyymm.

Field Format Types

<type>

A field's display format may be one of the following:

char	Alphanumeric Text
num	Number: 1,234,567.89
nocommas	Number: 1234567.89
date	Date: 10/15/98
month	Month/Year: 10/03
quarter	Quarter/Year: 2Q03
hms24	Time: 22:45:56
datehms24	Date+Time: 10/15/98_22:45:56