

# 1010data

## Application Program Interface Reference Manual

Version 3

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 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 20070610

© Copyright 2001-2009 1010data, Inc.

# TABLE OF CONTENTS

Preface.....	4
Prerequisites.....	4
Transactions.....	4
Error Codes.....	5
<b>Transaction Details .....</b>	<b>7</b>
Start the Session.....	8
End the Session.....	9
List the Contents of a Directory.....	10
List the Contents of a Directory (Extended).....	12
Get Information About a Directory.....	10
Modify Directory Information.....	10
Create a New Directory.....	10
Get Information About a Table.....	12
Get Meta Information About a Table.....	12
Modify Table Information.....	12
Apply a Query to a Table.....	27
Get Query Results.....	29
Apply a Query to a Table and Get Results.....	33
Save Query Results as a File.....	34
Save Query Results as a Table.....	36
Upload Table.....	38
Merge Tables.....	40
Delete Table.....	42
Delete Directory.....	442

Delete Several Tables and/or Directories .....	42
Move Several Tables and/or Directories.....	42
Order the Items in a Directory .....	46
Clear the Cache .....	43
Set Session Parameters .....	48
<b>Table Tree.....</b>	<b>50</b>
Overview.....	51
Meta-Information for All Columns.....	52
Table Data.....	53
Long Description .....	54
Link Header .....	55
Download Limit.....	56
Short Description .....	57
Top-Level Wrapper.....	58
Data for One Column and Row (Cell) .....	59
Meta-Information for One Column.....	60
Table Title.....	61
Data for One Row .....	62
<b>Users Tree.....</b>	<b>63</b>
Overview.....	64
Top-Level Wrapper.....	665
User Entry .....	66

## Preface

The 1010data Application Program Interface (API) is a facility whereby a client application running on a user's machine can access and query data on the 1010data database servers. This allows for unlimited customization of the application while taking advantage of 1010data's database management services and fast database engine. The API uses HTTP and XML and is compatible with any client application written in a language that supports HTTP transactions (such as Java, Visual Basic, C++, Python and PERL.)

## Prerequisites

A working knowledge of the 1010data database service and user interface is assumed. In particular, the reader is expected to be familiar with table operations (e.g. select rows, sort, tabulate) and the 1010data macro language.

To use the API, a valid 1010data username and password is required and the username must be authorized to use the API.

## Transactions

Transactions through the API are a series of HTTP(S) POSTs. 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/gw.k?api=query&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843 http/1.0
\r\nContent-Type: text/xml\r\nContent-Length: 296\r\n\r\n<in><name>pub.demo.weather.hourly95
</name><ops><sel value="(id=94789)"/><tabu label="Average temperature and humidity in NYC" b
reaks="date "><tcoll source="temp" fun="avg" label="Average`Dry Bulb`Temp`(Celsius)"/><tcoll s
ource="hum" fun="avg" label="Average`Relative`Humidity`(%)" /></tabu></ops></in>
```

or more legibly,

```
POST /cgi-bin/gw.k?api=query&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843 http/1.0
Content-Type: text/xml
Content-Length: 296

<in>
<name>pub.demo.weather.hourly95</name>
<ops>
<sel value="(id=94789)"/>
<tabu label="Average temperature and humidity in NYC" breaks="date">
<tcoll source="temp" fun="avg" label="Average`Dry Bulb`Temp`(Celsius)"/>
<tcoll source="hum" fun="avg" label="Average`Relative`Humidity`(%)" />
</tabu>
</ops>
</in>
```

Note that the query string on the first line specifies that this is a "query" 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: Sun, 23 Dec 2001 04:27:43 GMT
Server: Apache/1.3.9 (OpenSA) (Win32) mod_ssl/2.4.2 OpenSSL/0.9.4
Content-Length: 251
Connection: close
Content-Type: text/xml
```

```

<out>
<rc>0</rc>
<msg>query successful</msg>
<nrows>365</nrows>
<table>
<cols>
<th name="date" type="i" format="type:date">
Date
</th>
<th name="t0" type="f" format="type:num;width:6;dec:2">
Average&#10;Dry Bulb&#10;Temp&#10;(Celsius)
</th>
<th name="t1" type="f" format="type:num;width:6;dec:2">
Average&#10;Relative&#10;Humidity&#10;(%)
</th>
</cols>
<data/>
</table>
</out>

```

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, 1-39 means error) </rc>
<msg> message </msg>

```

The error codes are described in detail in the next section.

## Error Codes

A nonzero value in the <rc> tag indicates there was an error in processing your transaction. The error can be related to the syntax or semantics of the XML content submitted to the server. The codes are designed to give programs the ability to gracefully deal with errors without having to interpret the contents of the <msg> tag. The universe of possible error codes is as follows –

- 0 Success (no error)
- 1 Unclassified error
- 2 XML error
- 3 Missing user identification
- 4 Invalid user identification
- 5 Already logged in
- 6 Missing transaction type
- 7 Invalid transaction type
- 8 Missing element
- 9 Invalid element value
- 10 Invalid element contents
- 11 Missing attribute
- 12 Invalid attribute name
- 13 Invalid attribute value
- 14 Invalid directory name
- 15 No such directory
- 16 Invalid table name
- 17 No such table
- 18 No such directory or table
- 19 Directory already exists
- 20 Table already exists
- 21 No query specified
- 22 Not currently implemented for Quick Queries

- 23 Too many values
- 24 Empty table
- 25 Problem saving table
- 26 Cannot save into 'uploads' directory
- 27 Problem deleting table
- 28 Problem saving file
- 29 Duplicate column names
- 30 Duplicate table names
- 31 No FTP permission
- 32 Not enough available space left in your account.
- 33 Problem moving table or directory
- 34 Problem modifying table or directory attributes
- 35 Not Logged in.
- 36 <cols> contains columns that do not appear in result
- 37 You do not own the specified group
- 38 System is busy
- 39 You are not at an authorized IP address

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 "3".
<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=3&uid=myid&pswd=mypswd&kill=yes
```

## XML Input to Server

None.

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;sid&gt;</code>	Session ID
<code>&lt;pswd&gt;</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

```
<out>
<rc>0</rc>
<sid>1748453843</sid>
<pswd>Alynnsewxhck</pswd>
<msg>Last login was: 2001-12-27 01:01:46</msg>
</out>
```

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 "3".
<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)]

The query string may also contain the following optional parameter:

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

If `kill=yes` and a query is processing, the query will be terminated and the session will logout as usual.

### Example

```
api=logout&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

None.

## XML Response from Server

A successful logout produces the following result:

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

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 "3".
<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=3&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.) To list the contents of the top level directory ("All Databases"), leave the name blank ("`<name></name>`" or "`<name/>`").

### Example

```
<in><name>pub.demo.weather</name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;table&gt;</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

```
<out>
<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.demo.weather.stations</td>
      <td>tab</td>
    </tr>
    <tr>
      <td>pub.demo.weather.pwcodes</td>
      <td>tab</td>
    </tr>
  </data>
</table>
```

```
<tr>
  <td>pub.demo.weather.hourly</td>
  <td>tab</td>
</tr>
<tr>
  <td>pub.demo.weather.hourly95</td>
  <td>tab</td>
</tr>
</data>
</table>
</out>
```

The `listdir` transaction returns a listing of the contents of a directory (folder.) and provides more information than the `dir` transaction.

## Query String

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

<code>api</code>	Must be "listdir".
<code>apiversion</code>	Must be "3".
<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=listdir&apiversion=3&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.) To list the contents of the top level directory ("All Databases"), leave the name blank ("`<name></name>`" or "`<name />`").

And the following optional element:

`<include>` An optional filter to reduce the number of attributes returned in order to reduce transmission size. Provide a list of `<name>` elements containing the attributes you want in the results. Also include a mode attribute containing 1 or 2 depending on whether operation should apply to parents as well as children, or children only respectively.

### Example

```
<in>
  <name>pub.demo.weather</name>
  <include mode=1><name>id</name>
    <name>users</name>
</include>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;dir&gt;</code>	Directory information

The `<dir>` element contains the same attributes as the result of `getdir`. Its value has `<parents>` which contains the directories parent and `<children>` which contains a listing of its immediate descendants in the same form as the result of `gettab` and `getdir`

## Example

```
<out>
<rc>0</rc>
<msg>gettab successful</msg>
```

## List the Contents of a Directory (Extended) (Continued) `listdir`

---

```
<dir id="36011" name="pub.demo.weather" title="Weather" sdesc="" ldesc="" type="DIR"
secure="0" own="0" owner="sandy2" upload="0" update="2034-12-31 19:00:00" gif="" favorite="0"
numchild="9">
  <parents>
    <dir id="0" name="" title="All Databases" sdesc="" ldesc="" type="DIR" secure="0"
own="0" owner="" upload="0" update="2034-12-31 19:00:00" gif="" favorite="0"
numchild="9"/><dir id="2" name="pub" title="Published Data" sdesc="" ldesc=""
type="DIR" secure="0" own="0" owner="sandy2" upload="0" update="2034-12-31
19:00:00" gif="" favorite="1" numchild="4"/><dir id="36009" name="pub.demo"
title="Demo" sdesc="" ldesc="" type="DIR" secure="0" own="0" owner="sandy2"
upload="0" update="2034-12-31 19:00:00" gif="" favorite="0" numchild="4"/>
  </parents>
  <children>
    <tab id="170" name="pub.demo.weather.stations" type="REAL" display="TABLE"
report="0" chart="0" title="Stations" sdesc="Station location" ldesc=""
link="stations" rows="262" bytes="14064" segs="1" tstat="0" access="1" secure="0"
maxdown="" own="0" owner="sandy2" update="2007-02-26 19:13:37" favorite="0"/><tab
id="171" name="pub.demo.weather.pwcodes" type="REAL" display="TABLE" report="0"
chart="0" title="Present Weather Codes" sdesc="" ldesc="" link="pwcodes" rows="83"
bytes="1456" segs="1" tstat="0" access="1" secure="0" maxdown="" own="0"
owner="sandy2" update="2007-02-26 19:13:37" favorite="0"/><tab id="172"
name="pub.demo.weather.hourly" type="REAL" display="TABLE" report="0" chart="0"
title="Hourly U.S. Weather" sdesc="Hourly United States weather observations
(1990-1995)" ldesc="This table contains hourly observations at 262 weather
stations over a six year period (1990-1995). Each row contains the data for a
particular station at a particular hour of a particular day. (Note that there are
262 stations and 2191 days in the period 1990-1995, so there should be 13,777,008
rows (262 stations x 2191 days x 24 hours) in this table. There are actually
somewhat fewer because the data begins with 1:00 AM on 1/1/90 and ends with 11:00
PM on 12/31/95, so there is one hour missing.) Each station is identified by an
ID that corresponds to the ID in the &quot;Stations&quot; table." link=""
rows="13776746" bytes="1558716664" segs="4" tstat="0" access="1" secure="0"
maxdown="" own="0" owner="sandy2" update="2001-01-04 21:33:38" favorite="0"/><tab
id="173" name="pub.demo.weather.hourly95" type="REAL" display="TABLE" report="0"
chart="0" title="Hourly U.S. Weather (1995)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather
stations over a one year period. Each row contains the data for a particular
station at a particular hour of a particular day. Each station is identified by
an ID that corresponds to the ID in the Stations table." link="weather"
rows="2294858" bytes="266215904" segs="4" tstat="0" access="1" secure="0"
maxdown="" own="0" owner="sandy2" update="2007-05-17 08:41:42" favorite="0"/><tab
id="73725" name="pub.demo.weather.hourly90" type="REAL" display="TABLE" report="0"
chart="0" title="Hourly U.S. Weather (1990)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather
stations over a one year period. Each row contains the data for a particular
station at a particular hour of a particular day. Each station is identified by
an ID that corresponds to the ID in the Stations table." link="weather"
rows="2295120" bytes="266246296" segs="4" tstat="0" access="1" secure="0"
maxdown="" own="1" owner="dhorowitz" update="2007-05-17 08:41:42" favorite="0"
users=""/><tab id="73726" name="pub.demo.weather.hourly91" type="REAL"
display="TABLE" report="0" chart="0" title="Hourly U.S. Weather (1991)"
sdesc="Hourly United States weather observations" ldesc="This table contains
hourly observations at 262 weather stations over a one year period. Each row
contains the data for a particular station at a particular hour of a particular
day. Each station is identified by an ID that corresponds to the ID in the
Stations table." link="weather" rows="2295120" bytes="266246296" segs="4"
tstat="0" access="1" secure="0" maxdown="" own="1" owner="dhorowitz" update="2007-
05-17 08:41:42" favorite="0" users=""/><tab id="73727"
name="pub.demo.weather.hourly92" type="REAL" display="TABLE" report="0" chart="0"
title="Hourly U.S. Weather (1992)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather
stations over a one year period. Each row contains the data for a particular
```

```

station at a particular hour of a particular day. Each station is identified by
an ID that corresponds to the ID in the Stations table." link="weather"
rows="2301408" bytes="266975704" segs="4" tstat="0" access="1" secure="0"
maxdown="" own="1" owner="dhorowitz" update="2007-05-17 08:41:42" favorite="0"
users=""/><tab id="73728" name="pub.demo.weather.hourly93" type="REAL"
display="TABLE" report="0" chart="0" title="Hourly U.S. Weather (1993)"
sdesc="Hourly United States weather observations" ldesc="This table contains
hourly observations at 262 weather stations over a one year period. Each row
contains the data for a particular station at a particular hour of a particular
day. Each station is identified by an ID that corresponds to the ID in the
Stations table." link="weather" rows="2295120" bytes="266246296" segs="4"
tstat="0" access="1" secure="0" maxdown="" own="1" owner="dhorowitz" update="2007-
05-17 08:41:42" favorite="0" users=""/><tab id="73729"
name="pub.demo.weather.hourly94" type="REAL" display="TABLE" report="0" chart="0"
title="Hourly U.S. Weather (1994)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather
stations over a one year period. Each row contains the data for a particular
station at a particular hour of a particular day. Each station is identified by
an ID that corresponds to the ID in the Stations table." link="weather"
rows="2295120" bytes="266246296" segs="4" tstat="0" access="1" secure="0"
maxdown="" own="1" owner="dhorowitz" update="2007-05-17 08:41:42" favorite="0"
users=""/>
</children>
</dir>

```

The `getdir` transaction returns meta information of a directory (folder.)

## Query String

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

<code>api</code>	Must be "getdir".
<code>apiversion</code>	Must be "3".
<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=getdir&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

<code>&lt;name&gt;</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.) To list the contents of the top level directory ("All Databases"), leave the name blank (" <code>&lt;name&gt;&lt;/name&gt;</code> " or " <code>&lt;name/&gt;</code> ").
---------------------------	--

### Example

```
<in><name>pub.demo.weather</name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;dir&gt;</code>	Directory information

The directory information is returned as attributes to the `<dir>` element. The attributes are as follows –

<code>id</code>	ID of the directory
<code>name</code>	Name, or full path of the directory
<code>title</code>	Title of the directory as displayed in the UI
<code>sdesc</code>	Short Description as displayed under the title in the UI
<code>ldesc</code>	Long Description as displayed when navigating to the folder in the UI
<code>type</code>	Type of directory. At the moment there is only one type, DIR
<code>secure</code>	Is the Directory marked as secure (1 or 0)
<code>own</code>	Is the API user the owner of the directory specified in name (1 or 0)
<code>owner</code>	Who is the owner of the directory
<code>upload</code>	Can the API user create new items in the directory specified in name (1 or 0)
<code>update</code>	When was the directory created or its attributes changed
<code>gif</code>	The GIF file associated with the directory to be displayed in the UI
<code>favorite</code>	Is this directory marked as a favorite for the API user (1 or 0)
<code>users</code>	Space separated list of users and groups who have permission to view the directory

## Get Information About a Directory (Continued) `getdir`

---

`uploaders` Space separated list of users and groups who have permission to create tables or directories inside the directory.

`numchild` How many visible items descend from the directory specified in name

### Example

```
<out>
<rc>0</rc>
<msg>getdir successful</msg>
<dir id="36011" name="pub.demo.weather" title="Weather" sdesc="" ldesc="" type="DIR"
secure="0" own="0" owner="sandy2" upload="0" update="2001-12-31 19:00:00" gif="" favorite="0"
numchild="9"/>
</out>
```

The `putdir` transaction modifies the meta information for a directory (folder.)

## Query String

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

<code>api</code>	Must be "putdir".
<code>apiversion</code>	Must be "3".
<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=putdir&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

```
<dir>
```

With the following mandatory attributes:

<code>id</code>	ID of the directory
-----------------	---------------------

With the following optional attributes:

<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 . table</i> (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 <b>Help/About this Table</b> in the 1010data user interface.
<code>title</code>	Title of the directory as displayed in the UI
<code>sdesc</code>	Short Description as displayed under the title in the UI
<code>ldesc</code>	Long Description as displayed when navigating to the folder in the UI
<code>secure</code>	Is the Directory marked as secure (1 or 0)
<code>gif</code>	The GIF file associated with the directory to be displayed in the UI
<code>users</code>	Space separated list of users and groups who have permission to view the directory
<code>uploaders</code>	Space separated list of users and groups who have permission to create tables or directories inside the directory.
<code>owner</code>	User or group who can modify, add and delete all descending items in the directory.

**Note that omitted attributes are not modified**

### Example

```
<in><dir id="36011" name="pub.demo.newweather" title="New Weather" sdesc="" ldesc="" secure="0" /></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message

<dir>            New directory information. See XML response of getdir

## Example

```
<out>
<rc>0</rc>
<msg>putdir successful</msg>
<dir id="36011" name="pub.demo.weather" title="Weather" sdesc="" ldesc="" type="DIR"
secure="0" own="0" owner="sandy2" upload="0" update="2001-12-31 19:00:00" gif="" favorite="0"
numchild="9"/>
</out>
```

The `mkdir` transaction creates a directory (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 "3".
<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=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following elements:

<code>&lt;users&gt;</code>	The access rights for the new folder (see "Users Tree".)
<code>&lt;upload&gt;</code>	The upload rights for the new folder. Exactly like the "Users Tree" except its wrapped by the <code>&lt;upload&gt;</code> element instead the <code>&lt;users&gt;</code> element
<code>&lt;title&gt;</code>	Title of the directory as displayed in the UI
<code>&lt;sdesc&gt;</code>	Short Description as displayed under the title in the UI
<code>&lt;ldesc&gt;</code>	Long Description as displayed when navigating to the folder in the UI

With the following optional elements:

<code>&lt;name&gt;</code>	The full path to the table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . dir</i> (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 <a href="#">Help/About this Table</a> in the 1010data user interface. If name is excluded you need a <code>&lt;parent&gt;</code> element and the system will auto assign the name.
---------------------------	--

OR

<code>&lt;parent&gt;</code>	The partial path to the table in 1010data database hierarchy. The parent path represents the path up to the parent directory in which the table will reside. The API will auto assign a directory name using the time at which the transaction was captured and the user id calling the <code>mkdir</code> transaction.
-----------------------------	---

### Example

```
<in>
<name>myfolder.myfolder</name>
<title>My Folder</title>
<sdesc>My Folder contains my Items</sdesc>
<ldesc>My Folder contains my Items that I created on behalf of my Company</ldesc>
<users>
  <user>john</user>
  <user>tom</user>
</users>
<upload>
  <user>john</user>
  <user>tom</user>
</upload>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<rc>	Return code
<msg>	Message
<name>	New directory path

### Example

```
<out>  
<rc>0</rc>  
<msg>mkdir successful</msg>  
<name>myfolder.myfolder</name>  
</out>
```

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 "3".
<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=3&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

```
<in><name>pub.demo.weather.hourly95</name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;table&gt;</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

```
<out>
<rc>0</rc>
<msg>tabinfo successful</msg>
<table name="pub.demo.weather.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. Each row contains the data for a particular station at a particular hour
    of a particular day. Each station is identified by an ID that corresponds to the
    ID in the Stations 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>
</out>
```

The `gettab` transaction returns meta information of a table (folder.) Note that meta information does not include information about columns or contents of the table.

## Query String

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

<code>api</code>	Must be "gettab".
<code>apiversion</code>	Must be "3".
<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=gettab&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

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

### Example

```
<in><name> pub.demo.weather.hourly95 </name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;tab&gt;</code>	Table information

The table information is returned as attributes to the `<tab>` element. The attributes are as follows –

<code>id</code>	ID of the table
<code>name</code>	Name, or full path of the table
<code>title</code>	Title of the table as displayed in the UI
<code>sdesc</code>	Short Description as displayed under the title in the UI
<code>ldesc</code>	Long Description as displayed when navigating to the table in the UI
<code>type</code>	Type of table. Can be one of – REAL, VIEW, PARAM, MERGED, UQ, TOLERANT
<code>secure</code>	Is the Directory marked as secure (1 or 0)
<code>own</code>	Is the API user the owner of the directory specified in name (1 or 0)
<code>owner</code>	Who is the owner of the directory
<code>update</code>	When was the directory created or its attributes changed
<code>favorite</code>	Is this directory marked as a favorite for the API user (1 or 0)
<code>users</code>	Space separated list of users and groups who have permission to view the directory
<code>display</code>	The output type of the table. Can be one of – TABLE, EXCEL, CHART, REPORT, TEXT, XML
<code>report</code>	Does the table or query have report specifications saved (1 or 0)
<code>chart</code>	Does the table or query have chart specifications saved (1 or 0)

<code>link</code>	A string to be spliced to column headers when linking in to other tables
<code>rows</code>	Number of rows in the table
<code>bytes</code>	Number of bytes utilized by the table
<code>segs</code>	Number of segments in the table
<code>tstat</code>	Is the table accessible for timeseries functions (1 or 0)
<code>access</code>	Is the table accessible (1 or 0)
<code>maxdown</code>	Limit to the number of rows a user can download for the table

If the table is a Query (type of VIEW, PARAM, MERGED or TOLERANT) then the `tab` element will contain a `ops` and `dependencies` element. Any dependencies on other tables, ie links, merges, are available in the `<dependencies>` element which contains a list of lists of tables references if the table is a QQ or Merged table.

### Example

```
<out>
<rc>0</rc>
<msg>gettab successful</msg>
<tab id="173" name="pub.demo.weather.hourly95" type="REAL" display="TABLE" report="0"
chart="0" title="Hourly U.S. Weather (1995)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather stations over a
one year period. Each row contains the data for a particular station at a particular hour of
a particular day. Each station is identified by an ID that corresponds to the ID in the
Stations table." link="weather" rows="2294858" bytes="266215904" segs="4" tstat="0"
access="1" secure="0" maxdown="" own="0" owner="sandy2" update="2007-05-17 08:41:42"
favorite="0"/>
</out>
```

The `puttable` transaction modifies the meta information for a table.

## Query String

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

<code>api</code>	Must be "puttab".
<code>apiversion</code>	Must be "3".
<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=puttab&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

```
<tab>
```

With the following mandatory attributes:

`id` ID of the directory

With the following optional attributes:

<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 . table</i> (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 <b>Help/About this Table</b> in the 1010data user interface.
<code>title</code>	Title of the directory as displayed in the UI
<code>sdesc</code>	Short Description as displayed under the title in the UI
<code>ldesc</code>	Long Description as displayed when navigating to the folder in the UI
<code>secure</code>	Is the Directory marked as secure (1 or 0)
<code>gif</code>	The GIF file associated with the directory to be displayed in the UI
<code>users</code>	Space separated list of users and groups who have permission to view the directory
<code>uploaders</code>	Space separated list of users and groups who have permission to create tables or directories inside the directory
<code>owner</code>	User or group who can modify or delete the table

**Note that omitted attributes are not modified**

### Example

```
<in><tab id="36015" name="pub.demo.weather.newhourly95" title="New Weather Hourly 95"
sdesc="" ldesc="" secure="0" /></in>
```

## XML Response from Server

A successful result contains the following elements:

<rc>	Return code
<msg>	Message
<tab>	New table information. See XML response of gettab.

### Example

```
<out>
<rc>0</rc>
<msg>puttab successful</msg>
<tab id="173" name="pub.demo.weather.hourly95" type="REAL" display="TABLE" report="0"
chart="0" title="Hourly U.S. Weather (1995)" sdesc="Hourly United States weather
observations" ldesc="This table contains hourly observations at 262 weather stations over a
one year period. Each row contains the data for a particular station at a particular hour of
a particular day. Each station is identified by an ID that corresponds to the ID in the
Stations table." link="weather" rows="2294858" bytes="266215904" segs="4" tstat="0"
access="1" secure="0" maxdown="" own="0" owner="sandy2" update="2007-05-17 08:41:42"
favorite="0"/>
</out>
```

The `query` transaction applies a query to a table. A query is essentially a transformation on the table that selects a subset of its rows, reorders rows based on one or more sort criteria, summarizes the data and so on. The result is conceptually another table. Once a query is set using the `query` transaction, the values in the result table may be retrieved with the `getdata` transaction.

## Query String

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

<code>api</code>	Must be "query".
<code>apiversion</code>	Must be "3".
<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=query&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

<code>&lt;name&gt;</code>	The full path to the table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . table</i> (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 <code>Help/About this Table</code> in the 1010data user interface.
<code>&lt;ops&gt;</code>	A series of query operations in the XML macro language used on the <code>Edit Actions</code> page of the 1010data user interface.

`<name>` is required; `<ops>` is optional. If `<ops>` is omitted, the result table is the same as the original table.

**CHANGE IN VERSION 3:** If `<ops>` contains one or more `<colord>` operations, the `query` transaction takes them into account, i.e. the columns reported in the result (see below) will be only those columns selected in the `<colord>` operations and they will appear in the order specified in the `<colord>` operations.

### Example

```
<in>
<name>pub.demo.weather.hourly95</name>
<ops>
  <sel value="(id=94789)"/>
  <tabu label="Average temperature and humidity in NYC" breaks="date">
    <tccl source="temp" fun="avg" label="Average`Dry Bulb`Temp`(Celsius)"/>
    <tccl source="hum" fun="avg" label="Average`Relative`Humidity`(%)" />
  </tabu>
</ops>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;nrows&gt;</code>	Number of rows in result table
<code>&lt;table&gt;</code>	Information about columns of result table

The `<table>` element (see "Table Tree") essentially describes the columns of the result. It is equivalent to the result table with zero rows of data (i.e. the `<data>` element is empty.) The table tree does not contain `<title>`, `<sdesc>`, `<ldesc>`, `<link>` or `<maxdown>`. In the event of an error only the return code and error message are returned.

**Example**

```
<out>
<rc>0</rc>
<msg>query successful</msg>
<nrows>365</nrows>
<table>
  <cols>
    <th name="date" type="i" format="type:date">
      Date</th>
    <th name="t0" type="f" format="type:num;width:6;dec:2">
      Average&#10;Dry Bulb&#10;Temp&#10;(Celsius)</th>
    <th name="t1" type="f" format="type:num;width:6;dec:2">
      Average&#10;Relative&#10;Humidity&#10;(%)</th>
  </cols>
  <data/>
</table>
</out>
```

The `getdata` transaction retrieves the results of a query. More precisely, `getdata` retrieves the values in the result table defined by a `query` transaction.

## Query String

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

<code>api</code>	Must be "getdata".
<code>apiversion</code>	Must be "3".
<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=getdata&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications are optional consist of the following elements:

`<cols>` The column(s) to be retrieved. `<cols>` must contain one or more `<col>` elements. Each identifies one column and contains the name by which the column is referenced in formulas and query operations.

`<cols>` is optional. If not specified, all columns (after taking account `<colord>` operations) are retrieved.

`<rows>` The row(s) to be retrieved. `<rows>` must include attribute `mode` which indicates the form of the row specification. `mode="1"` allows a *relative* selection (e.g. *next* 10 rows), while `mode="2"` allows an *absolute* selection (e.g. rows 10-20). For `mode="1"`, the `<rows>` element has the form,

```
<rows mode="1"><next>number of rows</next></rows>
```

while for `mode="2"`, the form is,

```
<rows mode="2"><from>row number</from><to>row number</to></rows>
```

`<next>` is required if `mode="1"`; `<from>` and `<to>` are required if `mode="2"`.

`<rows>` is optional. If not specified, all rows are retrieved.

`<format>` The data format for the result (XML or character separated.) `<format>` must include attribute `type` and may include attribute `values`. `type` indicates the format type ("xml" or "csv"). If `type="xml"`, the `<format>` element has no contents. If `type="csv"`, it may contain the following elements:

`<sep>` Character with which to separate columns

`<linesep>` Character with which to separate rows

Both `<sep>` and `<linesep>` are optional. If not specified, the column separator is comma and the line separator is newline ("&#10;").

Attribute `values` indicates whether the data values should be "raw" or "formatted". Raw values are a simple representations of the data while formatted values are the way the data appears

in a table view in the user interface. Examples of raw values are 1234567, 7.87083333333333, and 19950101, while the corresponding formatted values might be 1,234,567, 7.87, and 1/01/95. If `type="csv"` and `values="formatted"`, `values` `<format>` is optional. If not specified, XML data with raw formatting is returned.

If no elements are specified it is assumed you wish to retrieve all rows with raw formatting in XML form.

that contain column or line separators (see `<sep>` and `<linesep>` above) are enclosed in quotes. If `values` is not specified, raw values are returned.

`<rows>` is required; `<cols>` and `<format>` are optional. If `<format>` is omitted, the result is XML.

#### Example 1: Relative row specification; XML

```
<in>
<cols>
  <col>date</col>
  <col>t0</col>
</cols>
<rows mode="1">
  <next>3</next>
</rows>
<format type="xml"/>
</in>
```

#### Example 2: Absolute row specification; tab-separated; raw values

```
<in>
<cols>
  <col>date</col>
  <col>t0</col>
</cols>
<rows mode="2">
  <from>1</from>
  <to>3</to>
</rows>
<format type="csv">
  <sep>&#9;</sep>
</format>
</in>
```

#### Example 3: Absolute row specification; comma-separated; formatted values

```
<in>
<cols>
  <col>date</col>
  <col>t0</col>
</cols>
<rows mode="2">
  <from>1</from>
  <to>3</to>
</rows>
<format type="csv" values="formatted">
  <sep>,</sep>
</format>
</in>
```

## XML Response from Server

A successful result contains the elements,

```
<rc>          Return code
<msg>        Message
```

and either `<table>` or `<csv>`, depending on the desired format.

The `<table>` element (see "Table Tree") contains the data for the specified columns and rows along with information about each column. (It does not contain `<title>`, `<sdesc>`, `<ldesc>`, `<link>` or `<maxdown>`.) Note that the `cols` element in `<table>` will differ from the query response if your `<format>` is set to `formatted`. Specifically, all the columns will report that they have been converted to alphanumeric which is what happens behind the scenes to properly format the data and is also the correct input if your trying to reload the data via the

upload transaction.

The `<csv>` element contains a CDATA section that contains the data in row-major order. The first row of data contains the column headers. In the event of an error only the return code and error message are returned.

### Example 1: XML

```
<out>
<rc>0</rc>
<msg>getdata successful</msg>
<table>
  <cols>
    <th name="date" type="i" format="type:date">
      Date
    </th>
    <th name="t0" type="f" format="type:num;width:6;dec:2">
      Average&#10;Dry Bulb&#10;Temp&#10;(Celsius)
    </th>
  </cols>
  <data>
    <tr>
      <td>19950101</td>
      <td>7.87083333333333</td>
    </tr>
    <tr>
      <td>19950102</td>
      <td>2.73333333333333</td>
    </tr>
    <tr>
      <td>19950103</td>
      <td>-1.6625</td>
    </tr>
  </data>
</table>
</out>
```

### Example 2: Tab-separated; raw values

```
<out>
<rc>0</rc>
<msg>getdata successful</msg>
<csv>
<![CDATA["Date", "Average
Dry Bulb
Temp
(Celsius)"]>
19950101 → 7.87083333333333
19950102 → 2.73333333333333
19950103 → -1.6625]]>
</csv>
</out>
```

("→" indicates a tab.)

### Example 3: Comma-separated; formatted values

```
<out>
<rc>0</rc>
<msg>getdata successful</msg>
<csv>
<![CDATA["Date", "Average
Dry Bulb
Temp
(Celsius)"]>
1/01/95,7.87
1/02/95,2.73
1/03/95,-1.66]]>
</csv>
</out>
```

The `querydata` transaction applies a query to a table and gets the results as if you called a `query` and a single `getdata` transaction. It takes the union of parameters from `query` and `getdata` respectively and returns the union of results from both `query` and `getdata`. Querydata is convenient and useful when you know the resulting dataset will be smaller than your maximum allowed download from a single transaction. Otherwise, you will still have to subsequently call `getdata` to receive all of the results.

## Query String

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

<code>api</code>	Must be "querydata".
<code>apiversion</code>	Must be "3".
<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=query&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

See XML input to `query` and `getdata` as the `querydata` transaction requires the input of both `query` and `getdata`.

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message
<code>&lt;nrows&gt;</code>	Number of rows in result table
<code>&lt;table&gt;</code>	Information about columns of result table and the resulting data.

The `savefile` transaction saves the results of a query as a file in your 1010data FTP folder. The file can then be downloaded via FTP.

Note: To use this transaction, you must be authorized for FTP downloads.

## Query String

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

<code>api</code>	Must be "savefile".
<code>apiversion</code>	Must be "3".
<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=savefile&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

<code>&lt;file&gt;</code>	The name of the result file. If a file already exists with that name, it will be replaced.				
<code>&lt;content&gt;</code>	The meta information to include in the file. <code>&lt;content&gt;</code> must contain one or more <code>&lt;meta&gt;</code> elements, each identifying one type of information. Each <code>&lt;meta&gt;</code> element must include attribute <code>type</code> and <code>type</code> may have one of the following values: <table><tr><td><code>headers</code></td><td>Column headers</td></tr><tr><td><code>names</code></td><td>Column names</td></tr></table> <code>&lt;content&gt;</code> is optional. If not specified, no meta information is included in the file.	<code>headers</code>	Column headers	<code>names</code>	Column names
<code>headers</code>	Column headers				
<code>names</code>	Column names				
<code>&lt;format&gt;</code>	The data format for the result. <code>&lt;format&gt;</code> has no attributes and may contain the following elements: <table><tr><td><code>&lt;sep&gt;</code></td><td>Character with which to separate fields (columns)</td></tr><tr><td><code>&lt;linesep&gt;</code></td><td>Character with which to separate records (rows)</td></tr></table> If not specified, the column separator is comma and the line separator is newline (" <code>&amp;#10;</code> ").	<code>&lt;sep&gt;</code>	Character with which to separate fields (columns)	<code>&lt;linesep&gt;</code>	Character with which to separate records (rows)
<code>&lt;sep&gt;</code>	Character with which to separate fields (columns)				
<code>&lt;linesep&gt;</code>	Character with which to separate records (rows)				

`<file>` is required; the others are optional.

### Example

```
<in>
<file>My File</file>
<content>
  <meta type="names"/>
</content>
<format>
  <sep>,</sep>
</format>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<rc>           Return code  
<msg>          Message

### Example

```
<out>  
  <rc>0</rc>  
  <msg>Saved file My File</msg>  
</out>
```

The `savetable` transaction saves the results of a query as a new table. The table is added to the 1010data database hierarchy and may be made accessible to other users.

## Query String

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

<code>api</code>	Must be "savetable".
<code>apiversion</code>	Must be "3".
<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=savetable&apiversion=3&uid=myid&pswd=Alynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

<code>&lt;name&gt;</code>	The contents of this element is the full path for the new table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . table</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.) The parent directory must exist and the user must be authorized to add tables to it.  <name> may include optional attribute <code>mode</code> . If a table already exists by this name and <code>mode="replace"</code> , the table will be replaced. If <code>mode="noreplace"</code> , an error message will be returned. If you want to append the data to the end of an existing table, specifying <code>mode="append"</code> will insert the records from the results for those columns whose names and types are compatible with the table specified in <code>&lt;name&gt;</code> . If <code>mode</code> is omitted, the default is <code>noreplace</code> .
<code>&lt;title&gt;</code>	The title for the new table (see "Table Tree - <code>&lt;title&gt;</code> ".)
<code>&lt;sdesc&gt;</code>	The short description for the new table (see "Table Tree - <code>&lt;sdesc&gt;</code> ".)
<code>&lt;ldesc&gt;</code>	The long description for the new table (see "Table Tree - <code>&lt;ldesc&gt;</code> ".)
<code>&lt;link&gt;</code>	The link header for the new table (see "Table Tree - <code>&lt;link&gt;</code> ".)
<code>&lt;maxdown&gt;</code>	The download limit for the new table (see "Table Tree - <code>&lt;maxdown&gt;</code> ".)
<code>&lt;merge&gt;</code>	A flag indicating whether the table should be able to be merged with other tables. If you plan to merge the saved table with another table that you save or upload, specify <code>&lt;merge&gt;1&lt;/merge&gt;</code> . <i>However your queries will run faster and use less virtual memory if you specify <code>&lt;merge&gt;0&lt;/merge&gt;</code> or omit the <code>&lt;merge&gt;</code> element entirely.</i>
<code>&lt;users&gt;</code>	The access rights for the new table (see "Users Tree".)
<code>&lt;modcol&gt;</code>	Modify a column by changing its name, title and/or display format. Specify the column you wish to modify with the required <code>name</code> attribute and optionally specify <code>title</code> to change the title and <code>format</code> to change the display format (See "Table Tree - <code>&lt;th&gt;</code> ".) Also, a new name can be provided as the value of <code>&lt;modcol&gt;</code> xml element. See the example below for more clarity.

<name> is required; the others are optional.

## Example

```
<in>
  <name mode="noreplace">myfolder.mytable</name>
  <title>My Table</title>
  <users>
    <user>john</user>
    <user>tom</user>
  </users>
  <modcol name="year" title="New Title" format="type:nocommas;width:4">new_year</modcol>
</in>
  <user>tom</user>
</users>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<rc>	Return code
<msg>	Message

## Example

```
<out>
  <rc>0</rc>
  <msg>Saved table myfolder.mytable</msg>
</out>
```

The `upload` transaction uploads a table from the client machine to the 1010data database servers. The table is added to the 1010data database hierarchy and may be made accessible to other users. There is a 5mb limit on the size of the input for this transaction. For large tables, its recommended to use the PowerLoader API.

## Query String

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

<code>api</code>	Must be "upload".
<code>apiversion</code>	Must be "3".
<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=upload&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

<code>&lt;table&gt;</code>	The table information in the form of a table tree (see "Table Tree".) The name attribute of the <code>&lt;table&gt;</code> element optionally specifies the desired location of the table in the 1010data database hierarchy (see also " <code>&lt;name&gt;</code> " below.) The parent directory must exist and the user must be authorized to add tables to it. <code>&lt;title&gt;</code> , <code>&lt;sdesc&gt;</code> , <code>&lt;ldesc&gt;</code> , <code>&lt;link&gt;</code> and <code>&lt;maxdown&gt;</code> are all optional.
<code>&lt;name&gt;</code>	This element is ignored if the name attribute of the <code>&lt;table&gt;</code> element is specified. If the name attribute of the <code>&lt;table&gt;</code> element is omitted, the contents of this element specifies the full path for the new table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . table</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.) The parent directory must exist and the user must be authorized to add tables to it.  <code>&lt;name&gt;</code> may include optional attribute <code>mode</code> . If a table already exists by this name and <code>mode="replace"</code> , the table will be replaced. If <code>mode="noreplace"</code> , an error message will be returned. If you want to append the data to the end of an existing table, specifying <code>mode="append"</code> will insert the records from the uploaded table for those columns whose names and types are compatible with the table specified in <code>&lt;name&gt;</code> . If <code>mode</code> is omitted, the default is <code>noreplace</code> .
<code>&lt;merge&gt;</code>	A flag indicating whether the table should be able to be merged with other tables. If you plan to merge the uploaded table with another table that you upload or save, specify <code>&lt;merge&gt;1&lt;/merge&gt;</code> . <i>However your queries will run faster and use less virtual memory if you specify <code>&lt;merge&gt;0&lt;/merge&gt;</code> or omit the <code>&lt;merge&gt;</code> element entirely.</i>
<code>&lt;users&gt;</code>	The access rights for the new table (see "Users Tree".)

`<table>` is required; `<name>` and `<users>` are optional. If `<users>` is omitted, only the uploading user will have access to the table.

**Example**

```
<in>
<table>
  <title>My Table</title>
  <cols>
    <th name="name" type="a">Full Name</th>
    <th name="age" type="i" format="type:nocommas;width:2">Age</th>
  </cols>
  <data>
    <tr>
      <td>John Doe</td>
      <td>31</td>
    </tr>
    <tr>
      <td>Mary Doe</td>
      <td>29</td>
    </tr>
    <tr>
      <td>Robert Smith</td>
      <td>45</td>
    </tr>
  </data>
</table>
<name mode="noreplace">myfolder.mytable</name>
<users>
  <user>john</user>
  <user>tom</user>
</users>
</in>
```

**XML Response from Server**

A successful result contains the following elements:

```
<rc>          Return code
<msg>        Message
```

**Example**

```
<out>
<rc>0</rc>
<msg>Added table myfolder.mytable</msg>
</out>
```

The `merge` transaction creates a new table by combining one or more existing tables. The table is added to the 1010data database hierarchy and may be made accessible to other users.

The tables are combined by appending all their rows together, i.e. the new table has similar columns to the original tables but has as many rows as the sum of the rows of the original tables. The original tables must have some columns in common and each of those columns must have the same data type and other attributes (NA values, alphabetic case, etc.) in all the tables. If some columns do not appear in all the tables, those columns are not included in the new table.

## Query String

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

<code>api</code>	Must be "merge".
<code>apiversion</code>	Must be "3".
<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=merge&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

<code>&lt;tabs&gt;</code>	The table(s) to be combined. <code>&lt;tabs&gt;</code> must contain one or more <code>&lt;tab&gt;</code> elements. Each identifies one table and contains its full path in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . table</i> (similar to a Windows or Unix file-system path but with dots instead of slashes.) The name of any table can be determined by clicking on menu item <b>Help/About this Table</b> in the 1010data user interface.
<code>&lt;name&gt;</code>	The contents of this element is the full path for the new table in the 1010data database hierarchy. The parent directory must exist and the user must be authorized to add tables to it.  <code>&lt;name&gt;</code> may include optional attribute <code>mode</code> . If a table already exists by this name and <code>mode="replace"</code> , the table will be replaced. If <code>mode="noreplace"</code> , an error message will be returned. If <code>mode</code> is omitted, the default is <code>noreplace</code> .
<code>&lt;title&gt;</code>	The title for the new table (see "Table Tree - <code>&lt;title&gt;</code> ".)
<code>&lt;sdesc&gt;</code>	The short description for the new table (see "Table Tree - <code>&lt;sdesc&gt;</code> ".)
<code>&lt;ldesc&gt;</code>	The long description for the new table (see "Table Tree - <code>&lt;ldesc&gt;</code> ".)
<code>&lt;link&gt;</code>	The link header for the new table (see "Table Tree - <code>&lt;link&gt;</code> ".)
<code>&lt;maxdown&gt;</code>	The download limit for the new table (see "Table Tree - <code>&lt;maxdown&gt;</code> ".)
<code>&lt;users&gt;</code>	The access rights for the new table (see "Users Tree".)

`<tabs>` and `<name>` are required; the others are optional.

### Example

```
<in>
<tabs>
  <tab>annualtables.t2001</tab>
```

```
<tab>annualtables.t2002</tab>
</tabs>
<name mode="noreplace">myfolder.mytable</name>
<title>2001-2002</title>
<users>
  <user>john</user>
  <user>tom</user>
</users>
</in>
```

## XML Response from Server

A successful result contains the following elements:

```
<rc>          Return code
<msg>        Message
```

### Example

```
<out>
<rc>0</rc>
<msg>Saved table myfolder.mytable</msg>
</out>
```

The `droptable` transaction deletes a table from the 1010data database servers. *All information about the table, including all of its data, is lost.* The user must be authorized to delete the table.

## Query String

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

<code>api</code>	Must be "droptable".
<code>apiversion</code>	Must be "3".
<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=droptable&apiversion=3&uid=myid&pswd=Alynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

<code>&lt;name&gt;</code>	The full path to the table in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . table</i> (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 <code>Help/About this Table</code> in the 1010data user interface.
---------------------------	--

### Example

```
<in><name>myfolder.mytable</name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message

### Example

```
<out>
<rc>0</rc>
<msg>Table myfolder.mytable deleted</msg>
</out>
```

The `dropdir` transaction deletes a directory from the 1010data database servers. *All information about the folder, including all of its data, is lost.* The user must be authorized to delete the directory.

## Query String

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

<code>api</code>	Must be "dropdir".
<code>apiversion</code>	Must be "3".
<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=dropdir&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

<code>&lt;name&gt;</code>	The full path to the directory in the 1010data database hierarchy. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . dir</i> (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 <b>Help/About this Table</b> in the 1010data user interface.
---------------------------	--

### Example

```
<in><name>myfolder.myfolder</name></in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message

### Example

```
<out>
<rc>0</rc>
<msg>Directory myfolder.myfolder deleted</msg>
</out>
```

The drop transaction deletes any number of tables and directories from the 1010data database servers. *All information about the table, including all of its data, is lost.* The user must be authorized to delete the items.

## Query String

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

api	Must be "drop".
apiversion	Must be "3".
uid	Username
pswd	Encrypted password (from the result of the login transaction)
sid	Session ID (from the result of the login transaction)

### Example

```
api=drop&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following element:

<objects> A list of <names> to be deleted. <names> can contain paths or ids.

### Example

```
<in>
  <objects>
    <name>pub.demo.weather.hourly95</name>
    <name>pub.demo.baseball</name>
  </objects>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<rc> Return code if its nonzero it means one or more of the drops did not work  
<msg> Message  
<statuses> The result codes and messages for each of the items dropped.

Statuses contains a list of <status> elements with a name attribute and the following elements:

<rc> Return code  
<msg> Message

### Example

```
<out>
<rc>0</rc>
<msg>Drop Successful</msg>
<statuses>
  <status name="pub.demo.weather.hourly95\"><rc>0</rc><msg>Drop Successful for
pub.demo.weather.hourly95</msg></status>
  <status name="pub.demo.baseball\"><rc>0</rc><msg>Drop Successful for
pub.demo.baseball</msg></status>
</statuses>
</out>
```

The `move` transaction moves any number of tables from anywhere in the folder hierarchy to a single directory.

## Query String

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

<code>api</code>	Must be "move".
<code>apiversion</code>	Must be "3".
<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=move&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following elements:

`<source>` A list of `<names>` to be moved. `<names>` can contain paths or ids.

### Example

```
<in>
  <source>
    <name>pub.demo.weather.baseball195</name>
    <name>pub.demo.weather.baseball197</name>
  </source>
  <target>pub.demo.baseball</target>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code if its nonzero it means one or more of the moves did not work
<code>&lt;msg&gt;</code>	Message
<code>&lt;statuses&gt;</code>	The result codes and messages for each of the items moved.

`Statuses` contains a list of `<status>` elements with a `name` attribute and the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message

### Example

```
<out>
<rc>0</rc>
<msg>Move Successful</msg>
<statuses>
  <status name="pub.demo.weather.baseball195"><rc>0</rc><msg>Move Successful for
pub.demo.weather.baseball195</msg></status>
  <status name="pub.demo.weather.baseball197"><rc>0</rc><msg>Move Successful for
pub.demo.weather.baseball197</msg></status>
</statuses>
</out>
```

The `order` transaction orders the items in a directory. Note that directories and tables can not be interspersed with one another.

## Query String

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

<code>api</code>	Must be "order".
<code>apiversion</code>	Must be "3".
<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=order&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications must contain the following elements:

<code>&lt;name&gt;</code>	The full path to the directory in the 1010data database hierarchy where you want to reorder items. In general, a path has the form <i>directory1 . directory2 . . . . directoryn . dir</i> (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 <b>Help/About this Table</b> in the 1010data user interface.
<code>&lt;dirs&gt;</code>	A list of <code>&lt;name&gt;</code> elements of the directories in the folder where the order of the <code>&lt;names&gt;</code> represents the order desired in 1010data
<code>&lt;tabs&gt;</code>	A list of <code>&lt;name&gt;</code> elements of the tables in the folder where the order of the <code>&lt;names&gt;</code> represents the order desired in 1010data

### Example

```
<in>
  <name>myfolder.myfolder</name>
  <dirs> <name>myfolder.myfolder1</name>
        <name>myfolder.myfolder2</name>
  </dirs>
  <tabs> <name>myfolder.mytable1</name>
        <name>myfolder.mytable2</name>
  </tabs>
</in>
```

## XML Response from Server

A successful result contains the following elements:

<code>&lt;rc&gt;</code>	Return code
<code>&lt;msg&gt;</code>	Message

### Example

```
<out>
  <rc>0</rc>
  <msg>Order Successful</msg>
</out>
```

Once the results of a query are retrieved or saved, it often makes sense to free up virtual memory before running another query. The `clear` transaction clears some or all of the memory used by the first query.

*Be careful: Do not use the `clear` transaction until you are finished saving or retrieving the results of the first query. If you do, the system will have to recompute the results from scratch.*

## Query String

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

<code>api</code>	Must be "clear".
<code>apiversion</code>	Must be "3".
<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=clear&apiversion=3&uid=myid&pswd=Aiyynnsewxhck&sid=1748453843
```

## XML Input to Server

None.

## XML Response from Server

A successful transaction produces the following result:

```
<out>
  <rc>0</rc>
  <msg>Cache cleared</msg>
</out>
```

Under certain circumstances it may be useful to adjust how queries use system resources such as processing power and virtual memory. The `session` transaction provides a means to effect such changes.

## Query String

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

<code>api</code>	Must be "session".
<code>apiversion</code>	Must be "3".
<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=session&apiversion=3&uid=myid&pswd=Aiynnsewxhck&sid=1748453843
```

## XML Input to Server

The specifications consist of the following elements:

- `<inf2na>` Some computations, e.g. dividing by zero, can result in errors or "infinity" values (`0I`, `-0I`, `0i` or `-0i`). Setting this parameter to "on" can help reduce or eliminate this behavior. Specifically,
- Division by zero ( $X/0$ ) returns `0i` or `-0i` if the numerator is not zero. Setting this parameter to "on" causes N/A to be returned instead of `0i` and `-0i`.
  - Exponentiation ( $X^Y$ ) can generate error messages if  $X$  is negative. Setting this parameter to "on" mostly eliminates such messages and causes N/A to be returned in the event of an error.
  - The range functions, e.g. `range1`, return `0I`, `-0I`, `0i` or `-0i` for column values that lie outside the specified ranges. Setting this parameter to "on" causes N/A to be returned instead.
- Valid values are "on" and "off".
- `<stepwise>` Setting this parameter to "on" saves memory but may cause queries to run more slowly. Choose this setting only if you experience virtual memory problems when working with large tables. Valid values are "on" and "off".
- `<blocking>` Setting this parameter to lower numbers saves memory but may cause queries to run more slowly. Use lower numbers when doing row selections and tabulations on large tables if you experience virtual memory problems. Valid values are the integers 0 through 10.

All elements are optional.

### Example

```
<in>
<inf2na>on</inf2na>
<blocking>0</blocking>
</in>
```

**XML Response from Server**

A successful transaction produces the following result:

```
<out>  
<rc>0</rc>  
<msg>Session parameters set</msg>  
</out>
```

**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 "Description" is the long description.

About this Table

Title	Hourly U.S. Weather (1995)
Name	pub.demo.weather.hourly95
ID	173
Last Change	2007-05-17 13:41:42 GMT
Owner	sandy2
Type	Real Table

Description
This table contains hourly observations at 262 weather stations over a one year period. Each row contains the data for a particular station at a particular hour of a particular day. Each station is identified by an ID that corresponds to the ID in the Stations table.

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 two columns from the Stations table. The heading of those columns therefore begins with "stations", which is the link header of the Stations table.

Hourly U.S. Weather (1995)

Columns 22-27 of 32, Rows 1-27 of 2,294,858

Station ID	Date	Hour	ASOS Cloud Layer 3	Hourly Precip (.01 inch)	Hourly Precip Flag	Snow Depth (inches)	stations Station Name	stations State
3103	1/01/95	1	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	2	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	3	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	4	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	5	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	6	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	7	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	8	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	9	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	10	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	11	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	12	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	13	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	14	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	15	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	16	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	17	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	18	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	19	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	20	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	21	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	22	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	23	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/01/95	24	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/02/95	1	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/02/95	2	99,999	0	0	0	FLAGSTAFF	AZ
3103	1/02/95	3	99,999	0	0	0	FLAGSTAFF	AZ

## Attributes

None.

## Contents

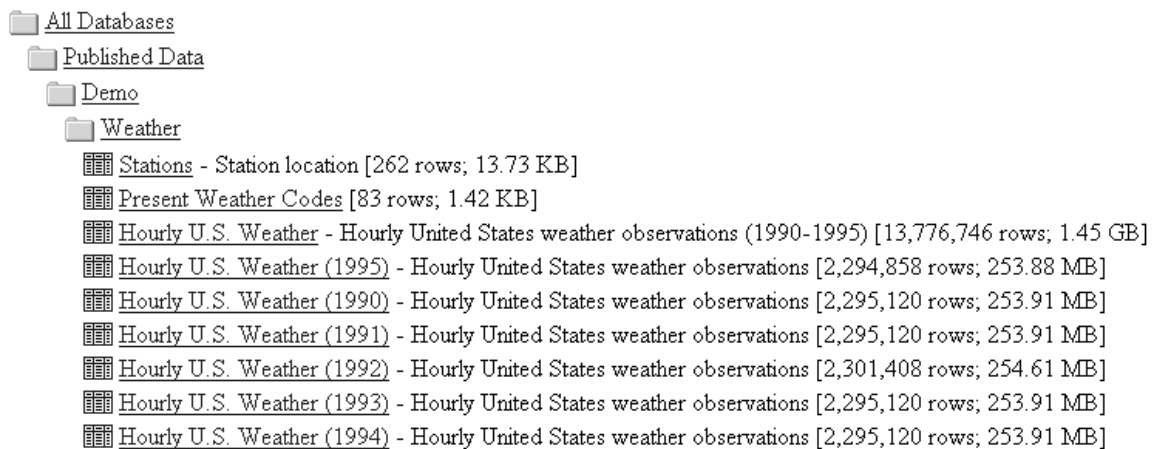
The maximum number of data items (rows  $\times$  columns) that a user of the 1010data user interface may download at one time. The download limit must be a nonnegative whole number.

## 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 location" and "Hourly United States weather observations" are short descriptions.



- 📁 All Databases
  - 📁 Published Data
    - 📁 Demo
      - 📁 Weather
        - 📄 Stations - Station location [262 rows; 13.73 KB]
        - 📄 Present Weather Codes [83 rows; 1.42 KB]
        - 📄 Hourly U.S. Weather - Hourly United States weather observations (1990-1995) [13,776,746 rows; 1.45 GB]
        - 📄 Hourly U.S. Weather (1995) - Hourly United States weather observations [2,294,858 rows; 253.88 MB]
        - 📄 Hourly U.S. Weather (1990) - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
        - 📄 Hourly U.S. Weather (1991) - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
        - 📄 Hourly U.S. Weather (1992) - Hourly United States weather observations [2,301,408 rows; 254.61 MB]
        - 📄 Hourly U.S. Weather (1993) - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
        - 📄 Hourly U.S. Weather (1994) - Hourly United States weather observations [2,295,120 rows; 253.91 MB]

<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:** `pub.demo.weather.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.

<title>	Title of table
<sdesc>	Short description of table
<ldesc>	Long description of table
<link>	Link header
<maxdown>	Download limit
<cols>	Column meta information
<data>	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 "&#10;" (the new line character.)

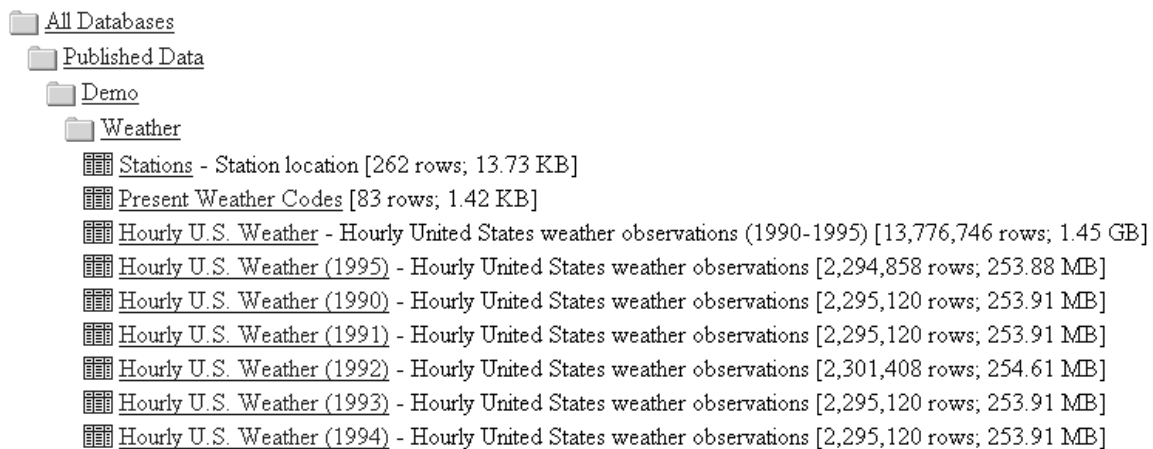
**Example:** `Street&#10;Address`

## 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.



The screenshot shows a directory tree structure. At the top is 'All Databases', which contains 'Published Data'. Under 'Published Data' is 'Demo', which contains 'Weather'. Under 'Weather' are several tables, each with a small icon and a description including row count and size.

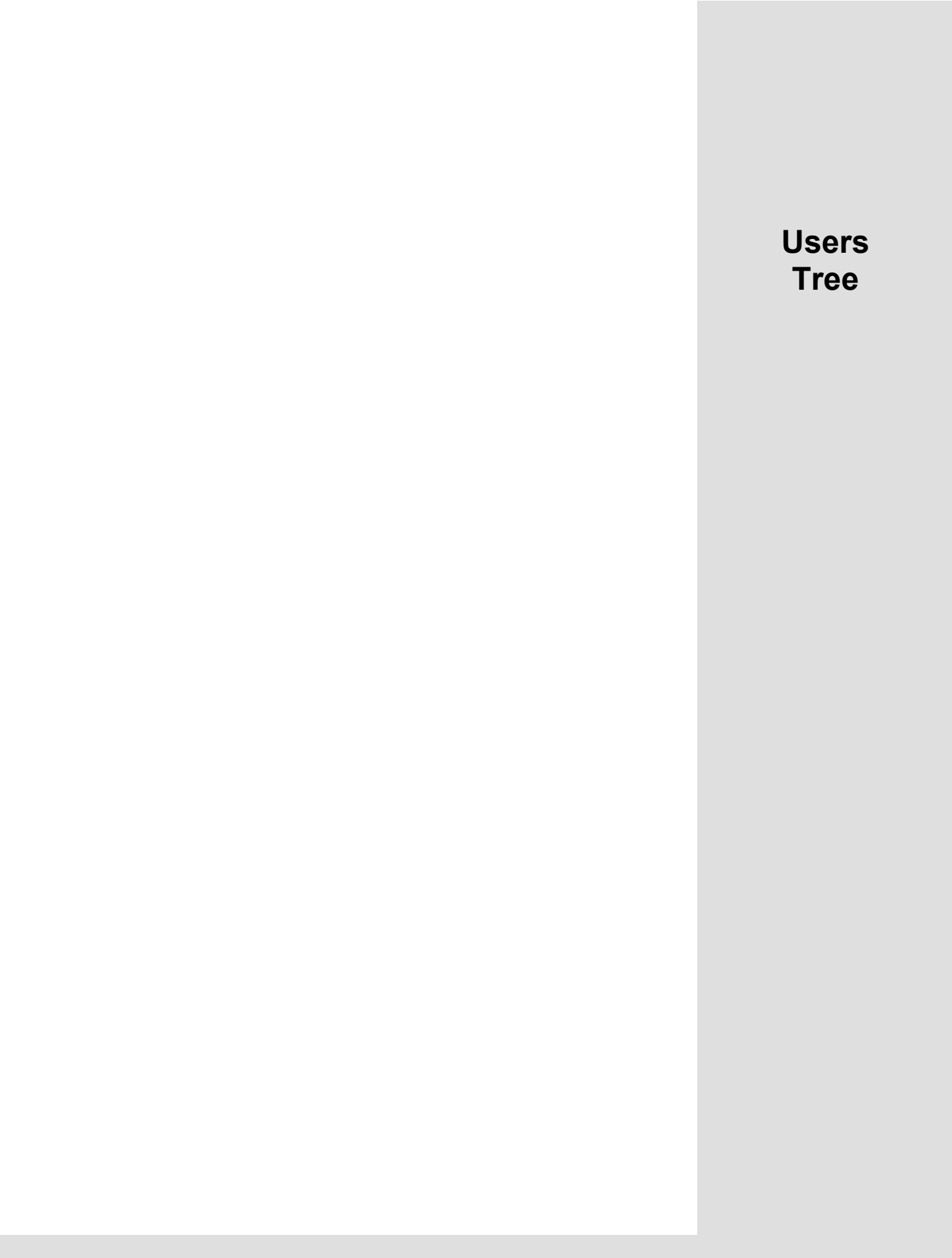
☞	<a href="#">All Databases</a>
☞	<a href="#">Published Data</a>
☞	<a href="#">Demo</a>
☞	<a href="#">Weather</a>
☞	<a href="#">Stations</a> - Station location [262 rows; 13.73 KB]
☞	<a href="#">Present Weather Codes</a> [83 rows; 1.42 KB]
☞	<a href="#">Hourly U.S. Weather</a> - Hourly United States weather observations (1990-1995) [13,776,746 rows; 1.45 GB]
☞	<a href="#">Hourly U.S. Weather (1995)</a> - Hourly United States weather observations [2,294,858 rows; 253.88 MB]
☞	<a href="#">Hourly U.S. Weather (1990)</a> - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
☞	<a href="#">Hourly U.S. Weather (1991)</a> - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
☞	<a href="#">Hourly U.S. Weather (1992)</a> - Hourly United States weather observations [2,301,408 rows; 254.61 MB]
☞	<a href="#">Hourly U.S. Weather (1993)</a> - Hourly United States weather observations [2,295,120 rows; 253.91 MB]
☞	<a href="#">Hourly U.S. Weather (1994)</a> - Hourly United States weather observations [2,295,120 rows; 253.91 MB]

### 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.