1010data API Reference Manual
Contents

1010data Application Program Interface ................................................................. 4
  Prerequisites .................................................................................................................. 4
  Transactions .................................................................................................................... 4
  Return codes .................................................................................................................. 7

Session management ..................................................................................................... 9
  login (Start the session) ................................................................................................. 9
  logout (End the session) ............................................................................................... 10
  clear (Clear the cache) ................................................................................................. 12
  session (Set session parameters) .................................................................................. 13

Directory management .................................................................................................. 16
  listdir (List the contents of a directory) ........................................................................ 16
  getdir (Get metadata about a directory) ....................................................................... 18
  putdir (Modify directory information) ........................................................................... 21
  mkdir (Create a directory) ............................................................................................ 23
  dropdir (Delete a directory) .......................................................................................... 25
  order (Order the items in a directory) .......................................................................... 27

Table management ........................................................................................................ 30
  gettab (Get information about a table) ......................................................................... 30
  tabinfo (Get information about a table) ......................................................................... 33
  savetable (Save query results as a table) ..................................................................... 35
  droptable (Delete table) ............................................................................................... 39
  upload (Upload table) .................................................................................................. 40
  puttab (Modify table metadata) .................................................................................... 43
  merge (Merge two tables) ............................................................................................. 47

Query management ....................................................................................................... 51
  query (Apply a query to a table) .................................................................................. 51
  getdata (Get the results of a query) .............................................................................. 53
  querydata (Apply a query to a table and get results) ..................................................... 59
  savefile (Save query results as a file) ........................................................................... 61

SAM pool management ................................................................................................ 64
  getuid (Retrieve a UID) ............................................................................................... 64
  reluid (Release a UID) ............................................................................................... 65
  markgid (Flush a SAM pool) ....................................................................................... 67

PowerLoader .................................................................................................................. 69
  addtab (Load a large table) .......................................................................................... 69
  status (Status of the addtab API) ................................................................................. 73
  convert (Convert a legacy spec) .................................................................................. 74
  validate (Validate a table) ........................................................................................... 77
Table Tree........................................................................................................................................85
   <table> (Top-level wrapper for table tree)..................................................................................87
   <title> (Table title)..................................................................................................................88
   <sdesc> (Short description of a table)........................................................................................89
   <ldesc> (Long description of a table).......................................................................................89
   <data> (Table data)..................................................................................................................90
   <maxdown> (Download limit).....................................................................................................90
   <segmentation> (Specify the segmentation of a table)...............................................................90
   <totals> (Table contains tabulation results)................................................................................91
   <cols> (Metadata for all columns)............................................................................................91
   <th> (Meta information for one column)....................................................................................91
   <links> (Top-level wrapper for links).......................................................................................92
   <link> (Link header)................................................................................................................93
   <data> (Table data)..................................................................................................................94
   <tr> (Data for one row)............................................................................................................94
   <td> (One cell of data in a table).............................................................................................94

Users Tree.......................................................................................................................................95
   <user> (User entry)....................................................................................................................95
   <users> (Top-level wrapper for users tree)..............................................................................95

Examples........................................................................................................................................97
   Python 2.7 example script.........................................................................................................97
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 building customized applications and interfaces, while taking advantage of 1010data's database management services and fast analytics 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).

**Session ID's and encrypted passwords**

When an API session with 1010data is established the system passes back two pieces of information that must be used in the query strings of subsequent transactions. First, once authenticated the system passes back an encrypted password. This encrypted password must be used in the password location for all other transactions. An example of what an encrypted password looks like is shown below:

```plaintext
_______04a90e8ae4c861123f1c7bb725f76e1f61305134d1fc3fca3cc993ad2b0d8fc0996a
```

**Note:** The password above is just an example and has been substantially shortened.

The encrypted password is only valid for the life of the 1010data session.

The Session ID or SID for short, is a unique identifier for your session. It is created when the session is created and ceases to exist once the session has ended. An example of a SID is shown here: 101339911.

Combined, your encrypted password and SID are a secure and reliable set of credentials for sending successive transactions in the same API session.

As a convention in this reference, the SID and encrypted password are referred to in query string examples with the variables: $SID and $PSWD respectively. The user ID for the session is represented by the $UID

**Prerequisites**

To effectively work with the 1010data XML API, you should be familiar with the concepts and topics described here.

**Prerequisites**

Working with the 1010data API is similar to working with many other service API's such as REST or SOAP. Experience building applications using such API's is recommended. The 1010data API uses the HTTPS POST method to register transactions and return results. While similar to REST API's in some ways, 1010data's stateful, session-based access model is different from traditional REST API's in several ways. Therefore, knowledge of 1010data sessions, directories, tables, and queries is recommended as well.

To use the API, a valid 1010data account with API access is required.

**Transactions**

An API session requires that a session first be established, then subsequent transactions be conducted using a unique set of identifiers. This section details how to establish a session and conduct transactions.

**Requests and inputs**

1010data API transactions are conducted using the HTTPS POST method. A transaction request consists of a POST URL and a POST body. The POST URL contains the gateway information along with the current session ID, encrypted password, API transaction name, and various other pieces of information. Most transactions require an XML input file and return an XML data structure.
The following is an example of an HTTPS query string that would be provided as the POST URL:

https://www2.1010data.com/gw.k?api=query&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**Note:** The user ID, encrypted password, and session ID are represented by the variables $UID, $PSWD and $SID respectively.

Each transaction has its own XML input specification, which is detailed in the reference documentation of the transaction. An example of an XML input for the query transaction is shown below:

```xml
POST /cgi-bin/gw.k?api=query&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
https/1.1
Content-Type: text/xml
Content-Length: 133

<in>
  <name>pub.doc.retail.salesdetail</name>
  <ops>
    <sel value="i_<10"/>
    <colord cols="trans_date,store,xsales"/>
  </ops>
</in>
```

The example above provides a full POST header containing the following pieces of information according to the HTTPS 1.1 POST protocol:

**POST**

The HTTP method being called.

**URL**

The POST URL is appended to https://www2.1010data.com and contains numerous pieces of information important to the transaction being conducted. The data required for the query string is provided in detail in the documentation for each transaction. A brief description of each is shown below:

- **cgi-bin/gw.k**
  
  This piece of the query string specifies the gateway used for the transaction. This information is required for each transaction, but is also the same for each transaction.

- **?**
  
  Indicates a query consisting of key/value pairs will follow.

- **api=$TRANSACTION_NAME**
  
  Specifies the name of the transaction to be conducted. The transaction named specifies the input required for the transaction and the information that will be returned by the system.

- **uid=$UID**
  
  Specifies the 1010data user ID under which transactions will be conducted.

- **pswd=$PSWD**
  
  The password or encrypted password used to authenticate a session and conduct transactions. The login transaction requires the user's unencrypted password. All other transactions, which can only be conducted during an active session, require the encrypted password returned by the login transaction.

- **sid=$SID**
  
  The session ID for the current session. The session ID is returned by the login transaction and is required for conducting subsequent transaction. Once a session ends the SID is no longer valid.
https 1.1

Specifies the version of the http/https protocol being used for the transaction. The 1010data XML API is compliant with http 1.0, http 1.1, https 1.0, and https 1.1.

Content-Type: text/xml

Specifies the type of data being sent in the POST body. The 1010data XML API is always of Content-Type text/xml.

Content-Length

Specifies the number of bytes in the POST body.

XML Input

The XML input is dependent on the transaction being called and must be placed in the POST body. If no input is required for the transaction the body may be left empty.

Response from 1010data

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

```
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: 612
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 Dry Bulb Temp (Celsius)</th>
      <th name="t1" type="f" format="type:num;width:6;dec:2">Average Relative Humidity (%)</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.
Return codes

Every transaction sent to 1010data returns an XML response containing at least a return code and its corresponding message. The error codes and messages are described here.

Return 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. Possible return codes are as follows:

Table 1: Return codes and messages

<table>
<thead>
<tr>
<th>Error code &lt;rc&gt;</th>
<th>Message &lt;msg&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td>Your transaction completed successfully</td>
</tr>
<tr>
<td>1</td>
<td>Unclassified error</td>
</tr>
<tr>
<td>2</td>
<td>XML error</td>
</tr>
<tr>
<td>3</td>
<td>Missing user identification</td>
</tr>
<tr>
<td></td>
<td>You did not provide a UID in the POST request.</td>
</tr>
<tr>
<td>4</td>
<td>Invalid user identification</td>
</tr>
<tr>
<td></td>
<td>The user ID you provided was invalid.</td>
</tr>
<tr>
<td>5</td>
<td>Already logged in</td>
</tr>
<tr>
<td></td>
<td>The user ID specified already has an established 1010data session.</td>
</tr>
<tr>
<td>6</td>
<td>Missing transaction type</td>
</tr>
<tr>
<td>7</td>
<td>Invalid transaction type</td>
</tr>
<tr>
<td>8</td>
<td>Missing element</td>
</tr>
<tr>
<td>9</td>
<td>Invalid element value</td>
</tr>
<tr>
<td>10</td>
<td>Invalid element contents</td>
</tr>
<tr>
<td>11</td>
<td>Missing attribute</td>
</tr>
<tr>
<td>12</td>
<td>Invalid attribute name</td>
</tr>
<tr>
<td>13</td>
<td>Invalid attribute value</td>
</tr>
<tr>
<td>14</td>
<td>Invalid directory name</td>
</tr>
<tr>
<td>15</td>
<td>No such directory</td>
</tr>
<tr>
<td>16</td>
<td>Invalid table name</td>
</tr>
<tr>
<td>17</td>
<td>No such table</td>
</tr>
<tr>
<td>18</td>
<td>No such directory or table</td>
</tr>
<tr>
<td>19</td>
<td>Directory already exists</td>
</tr>
<tr>
<td>20</td>
<td>Table already exists</td>
</tr>
<tr>
<td>Error code &lt;rc&gt;</td>
<td>Message &lt;msg&gt;</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>21</td>
<td>No query specified</td>
</tr>
<tr>
<td>22</td>
<td>Not currently implemented for Quick Queries</td>
</tr>
<tr>
<td>23</td>
<td>Too many values</td>
</tr>
<tr>
<td>24</td>
<td>Empty table</td>
</tr>
<tr>
<td>25</td>
<td>Problem saving table</td>
</tr>
<tr>
<td>26</td>
<td>Cannot save into 'uploads' directory</td>
</tr>
<tr>
<td>27</td>
<td>Problem deleting table</td>
</tr>
<tr>
<td>28</td>
<td>Problem saving file</td>
</tr>
<tr>
<td>29</td>
<td>Duplicate column names</td>
</tr>
<tr>
<td>30</td>
<td>Duplicate table names</td>
</tr>
<tr>
<td>31</td>
<td>No FTP permission</td>
</tr>
<tr>
<td>32</td>
<td>Not enough available space left in your account.</td>
</tr>
<tr>
<td>33</td>
<td>Problem moving table or directory</td>
</tr>
<tr>
<td>34</td>
<td>Problem modifying table or directory attributes</td>
</tr>
<tr>
<td>35</td>
<td>Not Logged in</td>
</tr>
<tr>
<td>36</td>
<td>&lt;cols&gt; contains columns that do not appear in result</td>
</tr>
<tr>
<td>37</td>
<td>You do not own the specified group</td>
</tr>
<tr>
<td>38</td>
<td>System is busy</td>
</tr>
<tr>
<td>39</td>
<td>You are not at an authorized IP address</td>
</tr>
<tr>
<td>41</td>
<td>1010data is unavailable</td>
</tr>
</tbody>
</table>

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.
Session management

This group of transactions provides basic functionality for 1010data session control.

**login (Start the session)**

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:

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid unencrypted 1010data password for the 1010data user name provided to the `uid` parameter.
  - The response from the login transaction will provide an encrypted password, which must be supplied as the value for the `pswd` attribute for all the other transactions.

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
    - **no**
      - Do not terminate the existing session.
      - If no session exists, a new one will be created. If a session does exist, an error is returned.
      - **Note**: If `kill=no` and a session is active, the system returns an error.
  - **possess**
    - Log in and possess the existing session.
  - **auth**
    - Authenticate with the system but do not kill, possess, or start any session.
    - The default is `yes`.

**XML input to server**

No XML input is required. All data for the transaction is specified in the connection string.
**XML response from server**

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. A successful result contains the following elements:

```xml
<rc>
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

<sid>
  The session ID assigned to the current session.
</sid>

<pswd>
  The encrypted password for the current session.
</pswd>

<msg>
  The message returned by the system. Specific messages correspond to specific return codes.
</msg>

<version>
  Contains the version of 1010data that the session is logged in to.
</version>
```

**Query string example**

```
https://www2.1010data.com/gw.k?api=login&apiversion=3&uid=$UID&pswd=$PSWD
```

**XML response example**

```xml
<out>
  <rc>0</rc>
  <sid>[SESSION_ID]</sid>
  <pswd>[ENCRYPTED_PSWD]</pswd>
  <msg>Last login was: [DATETIME]</msg>
  <version>[1010data_VERSION]</version>
</out>
```

**logout (End the session)**

The logout transaction ends an active session.

**Query string**

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

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
Specify a valid password for the 1010data user name provided to the uid parameter.

**Note:** The pswd value should be the encrypted password returned from the login transaction.

**sid**

The session ID for the current API session.

This value is returned by the login transaction and must be provided to every transaction (except login).

**kill (optional)**

Valid values for kill are as follows:

- **yes**
  
  Terminate the existing session and start a new one.

- **no**
  
  Do not terminate the existing session.

  If no session exists, a new one will be created. If a session does exist, an error is returned.

  **Note:** If kill=no and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is yes.

If kill=yes and a query is processing, the query will be terminated and the session will end.

**XML input to server**

No XML input required.

**XML response from server**

The logout transaction returns the following information:

- **<rc>**
  
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.

- **<msg>**
  
  The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**

https://www2.1010data.com/gw.k?api=logout&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML response example**

A successful logout produces the following result:
clear (Clear the cache)

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 removes completed operations from the session workspace to free the memory.

Query string

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

api
Specify the name of the API transaction.

apiversion
Specify the version of the API that should handle the requested transaction.

For the most up-to-date version, use apiversion=3.

uid
Specify a valid 1010data user name.

pswd
Specify a valid password for the 1010data user name provided to the uid parameter.

Note: The pswd value should be the encrypted password returned from the login transaction.

sid
The session ID for the current API session.

This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)

Valid values for kill are as follows:

yes
Terminate the existing session and start a new one.

no
Do not terminate the existing session.

If no session exists, a new one will be created. If a session does exist, an error is returned.

Note: If kill=no and a session is active, the system returns an error.

possess
Log in and possess the existing session.

auth
Authenticate with the system but do not kill, possess, or start any session.

The default is yes.
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.

**XML input to server**

No input is necessary for the clear transaction.

**XML response from server**

The clear transaction returns XML containing the following elements:

- `<rc>`
  - The return code generated by the transaction. For a list of return codes, see [Return codes](#) on page 7.

- `<msg>`
  - The message returned by the system. Specific messages correspond to specific return codes.

- `<memory>`
  - The amount of memory, in bytes, that represents the current workspace available in the session after the cache is cleared.

**Query string example**

```
https://www2.1010data.com/gw.k?api=clear&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML response example**

```
<out>
  <rc>0</rc>
  <msg>Cache cleared</msg>
  <memory>174369360</memory>
</out>
```

**session (Set session parameters)**

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 specify such settings.

**Query string**

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

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.

For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.
pswd

Specify a valid password for the 1010data user name provided to the uid parameter.

Note: The pswd value should be the encrypted password returned from the login transaction.

sid

The session ID for the current API session.

This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)

Valid values for kill are as follows:

yes

Terminate the existing session and start a new one.

no

Do not terminate the existing session.

If no session exists, a new one will be created. If a session does exist, an error is returned.

Note: If kill=no and a session is active, the system returns an error.

possess

Log in and possess the existing session.

auth

Authenticate with the system but do not kill, possess, or start any session.

The default is yes.

XML input to server

The specifications must contain the following elements:

<inf2na>

Accepts the values on to activate and off to disable. If active, this feature reduces or eliminates the number of errors and/or infinity values returned from computations resulting in such values. In cases when infinity values are generated, they will be turned into N/A values instead. Here are some things to keep in mind when deciding whether to use <inf2na>:

• Division by zero (\(X/0\)) returns \(0i\) or \(-0i\) if the numerator is not zero.
  
  Checking this box causes N/A to be returned instead of \(0i\) and \(-0i\).

• Exponentiation (\(X^Y\)) can generate error messages if \(X\) is negative.
  
  Checking this box mostly eliminates such messages and causes N/A to be returned in the event of an error.

• The range functions (range1, range1f, etc.) return \(0I\), \(-0I\), \(0i\), or \(-0i\) for column values that lie outside the specified ranges.

<stepwise> (optional)

Accepts the values on to activate and off to deactivate. Stepwise aggregation can be used to save memory for the session. While memory is saved, queries may run more slowly. Should only be used when experiencing issues with virtual memory when working with large tables.

<blocking>(optional)
Accepts an integer value between 0 and 10. Specifies whether to break selections and tabulations into sections to reduce memory usage when working with large tables. Should be set lower when virtual memory issues are experienced.

**XML response from server**

A successful session contains the following elements:

- `<rc>`
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
- `<msg>`
  The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**

https://www2.1010data.com/gw.k?api=session&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

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

**XML response example**

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

This section provides details on transactions used to change or retrieve information about directories.

**listdir (List the contents of a directory)**

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

**Query string**

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

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  
  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  The session ID for the current API session.
  
  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  Valid values for `kill` are as follows:

  - **yes**
    
    Terminate the existing session and start a new one.

  - **no**
    
    Do not terminate the existing session.
    
    If no session exists, a new one will be created. If a session does exist, an error is returned.
    
    **Note:** If `kill=no` and a session is active, the system returns an error.

  - **possess**
    
    Log in and possess the existing session.

  - **auth**
    
    Authenticate with the system but do not kill, possess, or start any session.

  The default is **yes**.
XML input to server

The specifications must contain the following elements:

<name>

The full name of a directory. Directory names are specified as a full path to the location of the directory, e.g., directory1.directory2.directory[N].thisdirectory.

#include

An optional filter to reduce the number of attributes returned in order to reduce transmission size. Contained inside the <include> element is a list of <name> elements that specify the directory information that should be returned by the transaction.

- <include mode="1"> specifies that data is only returned for parent objects.
- <include mode="2"> specifies that data is returned for parent and child objects in the directory.

See the XML input example for more information.

XML response from server

A successful response from the server contains the following elements:

<rc>

The return code generated by the transaction. For a list of return codes, see Return codes on page 7.

<msg>

The message returned by the system. Specific messages correspond to specific return codes.

<dir>

Contains data about parent and/or child objects in a directory.

Query string example

https://www2.1010data.com/gw.k?api=listdir&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

XML input example

<in>

  <name>pub.doc</name>
  <include mode="1">
    <name>ldesc</name>
    <name>sdesc</name>
    <name>title</name>
  </include>
</in>

XML response example

A successful listdir produces the following result:

$out>

  <rc>0</rc>
  <msg>listdir successful</msg>
  <dir title="Documentation" sdesc="" ldesc="">
    <parents>
      <dir title="All Databases" sdesc="" ldesc="" />
    </parents>
  </dir>
</out>
**getdir (Get metadata about a directory)**

The `getdir` transaction returns meta information of a directory.

**Query string**

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

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid password for the 1010data user name provided to the `uid` parameter.
    - **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  - The session ID for the current API session.
  - This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
    - **no**
      - Do not terminate the existing session.
      - If no session exists, a new one will be created. If a session does exist, an error is returned.
        - **Note:** If `kill=no` and a session is active, the system returns an error.
    - **possess**
      - Log in and possess the existing session.
Authenticate with the system but do not kill, possess, or start any session.

The default is yes.

**XML Input to Server**

The input specification for the transaction must contain the following elements:

<name>

The full name of the directory the transaction will request information about. See *Folders and Tables Browser* for additional information about directory and table names.

<include> (optional)

This optional element provides facility for specifying which specific metadata elements about the directory should be returned. Provide a list of <name> elements containing the names of the metadata elements that should be returned (e.g., <name>ldesc</name>). See the XML Response section of this topic for a complete list of metadata elements that may be returned by the transaction.

<include> also contains the mode attribute.

Use mode="1" to return metadata for parent and child directories.

Use mode="2" to only return information about child directories.

**XML Response from Server**

A successful response from the server contains the following elements:

<rc>

The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

<msg>

The message returned by the system. Specific messages correspond to specific return codes.

<dir>

Contains data about parent and/or child objects in a directory.

The directory metadata is returned as a series of attributes contained within the <dir> element. The attributes are as follows:

id

The unique identification number for the directory, table, or query. The ID is an integer.

name

The full path to the table, directory, or query to which the metadata belongs.

title

The title, or label, of the directory, table, or query.

sdesc

The short description of the directory, table, or query.

ldesc

The long description of the directory, table, or query.

type

The type of the directory. Currently, all directories are of type dir.
secure

1 if the directory, table, or query is marked as "secure" and 0 if it isn't.

Note: This metadata element is still returned but has been deprecated.

own

Boolean flag specifying whether the UID used for the API transaction is the owner of the directory.

owner

The owner of the directory, table, or query.

upload

Returns a 1 if the UID used for the API transaction has permission to add new objects to the directory. Returns a 0 if the UID does not.

update

The date and time the directory's metadata was last modified. If it was created and never modified, returns the date and time the directory was created.

gif

Returns the location of the gif file to be used to display the directory, table, or query in the UI.

users

A space-separated list of users and groups whom are allowed to access this directory, table, or query.

uploaders

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

numchild

The number of child elements that descend from the directory specified.

Query string example

https://www2.1010data.com/gw.k?api=getdir&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

XML Input Example

```
<in>
  <name>pub.demo.weather</name>
  <include mode="1">
    <name>ldesc</name>
    <name>sdesc</name>
    <name>type</name>
    <name>secure</name>
  </include>
</in>
```

XML response example

A successful getdir produces the following result:

```
<out>
```
**putdir (Modify directory information)**

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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  The session ID for the current API session.
  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  Valid values for `kill` are as follows:

  - **yes**
    Terminate the existing session and start a new one.

  - **no**
    Do not terminate the existing session.
    If no session exists, a new one will be created. If a session does exist, an error is returned.
    **Note:** If `kill=no` and a session is active, the system returns an error.

  - **possess**
    Log in and possess the existing session.

  - **auth**
    Authenticate with the system but do not kill, possess, or start any session.

The default is **yes**.
XML Input to Server

The input specification must contain the following elements:

<dir>
  Contains data about parent and/or child objects in a directory.

Include the following attributes with the <dir> element:

- **Note:** Any attributes that are omitted will not be modified.
- **Note:** You must specify both the directory name and the directory ID to modify directory metadata. If either piece of information is omitted an error is returned.

  **id**
  The unique identification number for the directory, table, or query. The ID is an integer.

  **name**
  The full path to the table, directory, or query to which the metadata belongs.

  **title**
  The title, or label, of the directory, table, or query.

  **sdesc**
  The short description of the directory, table, or query.

  **ldesc**
  The long description of the directory, table, or query.

  **secure**
  1 if the directory, table, or query is marked as "secure" and 0 if it isn't.

  **users**
  A space-separated list of users and groups whom are allowed to access this directory, table, or query.

  **owner**
  The owner of the directory, table, or query.

  **gif**
  Returns the location of the gif file to be used to display the directory, table, or query in the UI.

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

XML response from server

A successful response from the server contains the following elements:

<rc>
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.

<msg>
  The message returned by the system. Specific messages correspond to specific return codes.
<dir>
  Contains data about parent and/or child objects in a directory.

The directory metadata is returned as a series of attributes contained within the <dir> element. The attributes are as follows:

**Query string example**

```
https://www2.1010data.com/gw.k?api=putdir&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML input example**

```
<in>
  <dir id="3353015" name="certification" sdesc="Data for the 1010data certification program"/>
</in>
```

**XML response example**

A successful putdir produces the following result, based on the input example:

```
<out>
  <rc>0</rc>
  <msg>putdir successful</msg>
  <dir id="3353015" name="certification" title="Certification" sdesc="Data for the 1010data certification program" ldesc="" type="DIR" secure="0" own="1"
    owner="certification_admin" upload="1" update="2015-09-29 10:15:17"
    gif=""
    users="certification_admin rd_kdasika cp_training_users cp_training_trainers interns_2015 brausj abogdan rd_jsigler retaildemo_analian" uploaders="certification_admin"
    numchild="10"/>
</out>
```

**mkdir (Create a directory)**

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 a directory in the parent folder.

**Query string**

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

- **api**: Specify the name of the API transaction.
- **apiversion**: Specify the version of the API that should handle the requested transaction.

For the most up-to-date version, use `apiversion=3`. 
uid
   Specify a valid 1010data user name.

pswd
   Specify a valid password for the 1010data user name provided to the uid parameter.
   **Note:** The pswd value should be the encrypted password returned from the login transaction.

sid
   The session ID for the current API session.
   This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)
   Valid values for kill are as follows:
   yes
      Terminate the existing session and start a new one.
   no
      Do not terminate the existing session.
      If no session exists, a new one will be created. If a session does exist, an error is returned.
      **Note:** If kill=no and a session is active, the system returns an error.
   possess
      Log in and possess the existing session.
   auth
      Authenticate with the system but do not kill, possess, or start any session.
   The default is yes.

**XML input to server**

The specifications must contain the following elements:

<name>
   The full name of a directory. Directory names are specified as a full path to the location of the directory, e.g., directory1.directory2.directory[N].thisdirectory.

<users>
   Contains <user> elements, each containing a UID that is authorized to view the directory or table.
   See *Users Tree* on page 95 for the XML schema.

<uploaders>
   Contains <uploader> elements, each containing a UID that is authorized to modify the directory or table.
   See *Users Tree* on page 95 for the XML schema.

<title>
   The title of the directory or table.
**XML response from server**

A successful response from the server contains the following elements:

- `<rc>`
  - The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

- `<msg>`
  - The message returned by the system. Specific messages correspond to specific return codes.

- `<name>`
  - The full name of a directory. Directory names are specified as a full path to the location of the directory, e.g., `directory1.directory2.directory[N].thisdirectory`.

The directory metadata is returned as a series of attributes contained within the `<dir>` element. The attributes are as follows:

**Query string example**

```plaintext
https://www2.1010data.com/gw.k?api=mkdir&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML input example**

```xml
<in>
  <name>certification.apitests</name>
  <users type="list">
    <user>cp_training1</user>
    <user>cp_training2</user>
  </users>
  <upload type="list">
    <user>cp_trainer1</user>
  </upload>
  <title>API Training</title>
</in>
```

**XML response example**

A successful `mkdir` produces the following result:

```xml
<out>
  <rc>0</rc>
  <name>certification.apitests</name>
  <msg>Mkdir Successful</msg>
</out>
```

**dropdir (Delete a directory)**

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:

`api`
- Specify the name of the API transaction.

`apiversion`
- Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

`uid`
- Specify a valid 1010data user name.

`pswd`
- Specify a valid password for the 1010data user name provided to the `uid` parameter.
  - **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

`sid`
- The session ID for the current API session.
  - This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

`kill (optional)`
- Valid values for `kill` are as follows:
  - `yes`
    - Terminate the existing session and start a new one.
  - `no`
    - Do not terminate the existing session.
    - If no session exists, a new one will be created. If a session does exist, an error is returned.
      - **Note:** If `kill=no` and a session is active, the system returns an error.

`possess`
- Log in and possess the existing session.

`auth`
- Authenticate with the system but do not kill, possess, or start any session.
  - The default is `yes`.

XML input to server

The specifications must contain the following elements:

`<name>`
- The full name of a directory. Directory names are specified as a full path to the location of the directory, e.g., `directory1.directory2.directory[N].thisdirectory`.

XML response from server

A successful `dropdir` contains the following elements:
<rc>
The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

$msg>
The message returned by the system. Specific messages correspond to specific return codes.
</msg>

---

**Query string example**

| https://www2.1010data.com/gw.k?api=dropdir&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID |

**XML input example**

```xml
<in>
  <name>certification.apitests</name>
</in>
```

**XML response example**

```xml
<out>
  <rc>0</rc>
  <msg>Rmdir Successful</msg>
</out>
```

---

**order (Order the items in a directory)**

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

**Query string**

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

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  **Note:** The `pswd` value should be the encrypted password returned from the login transaction.

- **sid**
The session ID for the current API session.
This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

**kill (optional)**

Valid values for `kill` are as follows:

- **yes**
  
  Terminate the existing session and start a new one.

- **no**
  
  Do not terminate the existing session.

  If no session exists, a new one will be created. If a session does exist, an error is returned.

  **Note:** If `kill=no` and a session is active, the system returns an error.

- **possess**
  
  Log in and possess the existing session.

- **auth**
  
  Authenticate with the system but do not kill, possess, or start any session.

  The default is **yes**.

**XML input to server**

The specifications must contain the following elements:

- **<name>**
  
  The full name of a directory. Directory names are specified as a full path to the location of the directory, e.g., `directory1.directory2.directory[N].thisdirectory`.

- **<dirs>**
  
  Contains `<name>` elements, each containing the full name of a directory. For the order transaction, the order in which the `<name>` elements are provided represents the order in which the directories will appear in the parent directory.

- **<tabs>**
  
  Contains `<name>` elements, each containing the full name of a table. For the order transaction, the order in which the `<name>` elements are provided represents the order in which the tables will appear in the directory.

**XML response from server**

A successful order contains the following elements:

- **<rc>**
  
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.

- **<msg>**
  
  The message returned by the system. Specific messages correspond to specific return codes.
Query string example

https://www2.1010data.com/gw.k?api=order&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

XML input example

```xml
<in>
  <name>certification.upload_training.mark</name>
  <dirs>
    <name>certification.upload_training.mark.production</name>
    <name>certification.upload_training.mark.staging</name>
  </dirs>
  <tabs>
    <name>certification.upload_training.mark.rental_test</name>
    <name>certification.upload_training.mark.my_quick_app</name>
    <name>certification.upload_training.mark.csbcounty</name>
  </tabs>
</in>
```

XML response example

```xml
<out>
  <rc>0</rc>
  <msg>Order Successful</msg>
</out>
```
Table management

This section provides details on transactions used to retrieve information about or modify a table.

**gettab (Get information about a table)**

The `gettab` transaction returns metadata about a table. Note that this metadata 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:

- **api**
  
  Specify the name of the API transaction.

- **apiversion**

  Specify the version of the API that should handle the requested transaction.

  For the most up-to-date version, use `apiversion=3`.

- **uid**

  Specify a valid 1010data user name.

- **pswd**

  Specify a valid password for the 1010data user name provided to the `uid` parameter.

  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**

  The session ID for the current API session.

  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**

  Valid values for `kill` are as follows:

  - **yes**
    
    Terminate the existing session and start a new one.

  - **no**
    
    Do not terminate the existing session.

    If no session exists, a new one will be created. If a session does exist, an error is returned.

    **Note:** If `kill=no` and a session is active, the system returns an error.

    - **possess**
      
      Log in and possess the existing session.

    - **auth**
      
      Authenticate with the system but do not kill, possess, or start any session.

  The default is **yes**.
**XML input to server**

The specifications must contain the following elements:

```xml
<name>
    The full name of a table. Table names are specified as a full path to the location of the table, e.g., directory1.directory2.directory\[N]\table.
</name>

<include> (optional)
    This optional element provides facility for specifying which specific metadata elements about the directory should be returned. Provide a list of <name> elements containing the names of the metadata elements that should be returned (e.g., <name>ldesc</name>). See the XML Response section of this topic for a complete list of metadata elements that may be returned by the transaction.
    <include> also contains the mode attribute.
    Use mode="1" to return metadata for parent and child directories.
    Use mode="2" to only return information about child directories.
</include>
```

**XML response from server**

A successful response from the server contains the following elements:

```xml
<rc>
    The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

<msg>
    The message returned by the system. Specific messages correspond to specific return codes.
</msg>

<tab>
    Contains information about a table. The table’s metadata is contained within attributes of the <tab> element.

    The table metadata is returned as a series of attributes contained within the <tab> element. The attributes are as follows:

    **id**
    The id number of the table. Table id is represented as an integer value.

    **name**
    The full name of the table, represented as a path to the table.

    **type**
    Specifies the type of table. Possible return values are as follows:
    - REAL
    - VIEW
    - PARAM
    - MERGED
    - UQ
    - TOLERANT

    **display**
    The output type of the table. Possible return values are as follows:
    - TABLE
report
A boolean flag indicating whether the table or query has report specifications saved. Returns a 0 if no specifications are saved and a 1 if there are saved specifications.

chart
A boolean flag indicating whether the table or query has chart specifications saved. Returns a 0 if no specifications are saved and a 1 if there are saved specifications.

title
The title of the table.

sdesc
The short description of the table.

ldesc
The long description of the table.

link
A string that will be prepended to table headers during a link.

rows
The number of rows in the table.

bytes
The number of bytes the table consumes on disk.

segs
The number of segments contained in the table.

tstat
Specifies whether the table is can be accessed by time-series functions. If so, returns 1. If not, returns 1.

access
Specifies whether the table is accessible. If so, a 1 is returned. If not, a 0 is returned.

secure
1 if the directory, table, or query is marked as "secure" and 0 if it isn’t.

Note: This metadata element is still returned but has been deprecated.

own
Boolean flag specifying whether the UID used for the API transaction is the owner of the directory.

update
The date and time the directory’s metadata was last modified. If it was created and never modified, returns the date and time the directory was created.

users
A space-separated list of users and groups whom are allowed to access this directory, table, or query.
**maxdown**

The maximum number of rows that can be downloaded from the table at a time.

**favorite**

Specifies whether the table is flagged as a favorite for the user ID making the request. Returns 1 if the table is a favorite and a 0 if it isn't.

If the table is a query (i.e., type is VIEW, PARAM, MERGED or TOLERANT) then the tab element will contain ops and dependencies elements. Any dependencies on other tables (i.e., links, merges) are available in the <dependencies> element, which contains a list of table references if the table is a Quick Query or a merged table.

**Query string example**

https://www2.1010data.com/gw.k?api=gettab&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

```xml
<in>
  <name>pub.doc.retail.product</name>
</in>
```

**XML response example**

```xml
<out>
  <rc>0</rc>
  <msg>gettab successful</msg>
  <tab id="4857539" name="pub.doc.retail.product" type="REAL"
    display="TABLE"
    report="0" chart="0" title="Product Master" sdesc="" ldesc="&lt;note
      type=&amp;quot;base&amp;quot;&amp;gt;Applied to table:
      retaildemo.products&amp;lt;/note&amp;gt;"
    link="" rows="34573" bytes="1388328" segs="1" tstat="0" access="1"
    secure="0" maxdown="" own="0"
    owner="rd_ctaormina" mode="data" update="2015-06-29 13:50:55"/>
</out>
```

**tabinfo (Get information about a table)**

The tabinfo transaction returns information about a table. Unlike the gettab transaction, the metadata returned by a tabinfo transaction is provided in a table tree.

**Query string**

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

api

Specify the name of the API transaction.

apiversion

Specify the version of the API that should handle the requested transaction.
For the most up-to-date version, use `apiversion=3`.

**uid**
Specify a valid 1010data user name.

**pswd**
Specify a valid password for the 1010data user name provided to the `uid` parameter.

   **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

**sid**
The session ID for the current API session.

This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

**kill (optional)**
Valid values for `kill` are as follows:

   **yes**
   Terminate the existing session and start a new one.

   **no**
   Do not terminate the existing session.
   
   If no session exists, a new one will be created. If a session does exist, an error is returned.
   
   **Note:** If `kill=no` and a session is active, the system returns an error.

**possess**
Log in and possess the existing session.

**auth**
Authenticate with the system but do not kill, possess, or start any session.

The default is `yes`.

**XML input to Server**
The specifications must contain the following element:

```xml
<name>
   The full name of a table. Table names are specified as a full path to the location of the table, e.g.,
   directory1.directory2.directory[N].table.
</name>
```

**XML Response from Server**
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. A successful result contains the following elements:

```xml
<rc>
   The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

<msg>
   The message returned by the system. Specific messages correspond to specific return codes.
</msg>

<table>
A 1010data XML table tree. To see the full XML specification for the table tree, see Table Tree on page 85.

The <table> element (see Table Tree on page 85) 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.

**Query string example**

```
https://www2.1010data.com/gw.k?api=tabinfo&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML input example**

```
<in>
  <name>pub.demo.retail.prod</name>
</in>
```

**XML response example**

```
<out>
  <rc>0</rc>
  <msg>tabinfo successful</msg>
  <table name="pub.demo.retail.prod" segby="">
    <title>Product Master</title>
    <sdesc/>
    <ldesc/>
    <link/>
    <cols>
      <th name="sku" type="a" fixed="0">SKU</th>
      <th name="itemdesc" type="a" fixed="0">Item Description</th>
      <th name="class" type="i" fixed="0">Class</th>
      <th name="classdesc" type="a" fixed="0">Class Description</th>
      <th name="dept" type="i" fixed="0">Department</th>
      <th name="deptdesc" type="a" fixed="0">Department Description</th>
      <th name="div" type="i" fixed="0">Division</th>
      <th name="divdesc" type="a" fixed="0">Division Description</th>
    </cols>
    <data/>
  </table>
</out>
```

**savetable (Save query results as a table)**

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:

- **api**
  
  Specify the name of the API transaction.
**apiversion**

Specify the version of the API that should handle the requested transaction.

For the most up-to-date version, use `apiversion=3`.

**uid**

Specify a valid 1010data user name.

**pswd**

Specify a valid password for the 1010data user name provided to the `uid` parameter.

**Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

**sid**

The session ID for the current API session.

This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

**kill (optional)**

Valid values for `kill` are as follows:

- **yes**
  
  Terminate the existing session and start a new one.

- **no**
  
  Do not terminate the existing session.
  
  If no session exists, a new one will be created. If a session does exist, an error is returned.

  **Note:** If `kill=no` and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is `yes`.

**XML input to server**

The `savetable` transaction must be provided with the following information:

```xml
<name>

The content of this element is the full path for the table in the 1010data database hierarchy.

<name> may also contain any of the following optional attributes and corresponding valid values:

**mode**

- **mode** specifies how the system will respond when a table is sent that has the same name as an existing table.
  
  - **mode="replace"** will replace the existing table with the new one.
  
  - **mode="noreplace"** will return an error and will not save the new table.
  
  - **mode="append"** will append the new data to an existing table with the name specified.

If `mode` is omitted, the default is `noreplace`.```
**materialize**

The `materialize` attribute is used to activate some of the more powerful table-creation features in 1010data, such as segmentation.

To activate these features, use `materialize="1"`.

**<title>**

Specify the name of the new table.

See `<title>` (Table title) on page 88.

**<sdesc>**

The short description for the new table (optional).

See `<sdesc>` (Short description of a table) on page 89.

**<ldesc>**

The long description for the new table.

See `<ldesc>` (Long description of a table) on page 89.

**<link>**

The link header for the new table (optional).

See `<link>` (Link header) on page 93.

**<maxdown>**

The download limit for the new table (optional).

See `<maxdown>` (Download limit) on page 90.

**<merge>**

A flag indicating whether the table should be able to be merged with other tables (optional).

If you plan to merge the saved table with another table that you save or upload, specify `<merge>1</merge>`. However, your queries will run faster and use less virtual memory if you specify `<merge>0</merge>` or omit the `<merge>` element entirely.

**Note:** It is currently not possible to append a tabulated table to an existing table with the `materialize` attribute invoked.

**<users>**

The access rights for the new table (optional).

See Users Tree on page 95.

**<modcol>**

Modify a column by changing its name, title, and/or display format (optional).

Specify the column you wish to modify with the required name attribute and optionally specify title to change the title and format to change the display format. (See `<th>` (Meta information for one column) on page 91.) Also, a new name can be provided as the value of `<modcol>` XML element. See the example below for more clarity.

**Note:** `<modcol>` cannot be used with `materialize=1`.

**<segmentation>**

Specifies which column or columns by which to segment a table. This option requires `materialize=1`. 
The required `cols` attribute of `<segmentation>` specifies the columns to segment along. If the optional `sort` attribute is set to 1, the segments will also be sorted along the same columns. Specify segmentation as follows:

```
<segmentation cols="[COL_1],[COL_2],...[COL_N]" sort="1"/>
```

`<links>`

`links` can contain an arbitrary number of `<link>` elements, which act as a wrapper used by the `prelink` and `addtab` transactions to create a persistent link between two tables along a set of columns or to bring some subset of columns from a second table into the base table. It’s basic syntax is given as follows:

```
<links>
  <link table2="foreign.table.name" col="col1,col2" col2="fcol1,fcol2"
       denormalize="1" keepcols="" suffix=""/>
</links>
```

This syntax specifies that the table referenced by the `table2` attribute within the `<link>` element should be joined along the columns indicated in the `col` and `col2` attributes. If the `denormalize` attribute is 1, columns from the foreign table are denormalized into the base table, otherwise a `prelink` between the two tables is generated.

`<sort>`

The `<sort>` element is used to sort individual segments along the columns enumerated in the required `cols` attribute. This is in contrast to specifying a segmentation with `sort=1` in the `<segmentation>` element, where the sort is applied across all segmented columns. Basic syntax for `<sort>` is as follows:

```
<sort cols="[COL_1], [COL_2],...COL_N"/>
```

**Note:** `name` is the only required element of the list of above. The others are optional.

**XML response from server**

A `savetable` transaction successfully received by the API will return the following information:

`<rc>`

The return code from system.

`<msg>`

The message the system returns.

**Query string example**

```
https://www2.1010data.com/gw.k?api=savetable&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML input example**

```
<in>
  <name mode="replace" materialize="1">demos.apistock</name>
  <title>Materialized Title</title>
  <users type="list">
    <user>sandy</user>
  </users>
  <segmentation cols="symbol" sort="1" />
  <links>
    <link table2="demos.apicompanynames" col="symbol" col2="symbol"
         denormalize="1"
```


XML response example

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

droptable (Delete table)

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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  The session ID for the current API session.
  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill** *(optional)*
  Valid values for `kill` are as follows:

  - **yes**
    Terminate the existing session and start a new one.

  - **no**
    Do not terminate the existing session.

    If no session exists, a new one will be created. If a session does exist, an error is returned.

    **Note:** If `kill=no` and a session is active, the system returns an error.
possess
Log in and possess the existing session.

auth
Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

**XML input to server**
The specifications must contain the following elements:

```xml
<name>
The full name of a table. Table names are specified as a full path to the location of the table, e.g., directory1.directory2.directory[N].table.
```

**XML response from server**
A successful `droptable` contains the following elements:

```xml
<rc>
The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

<msg>
The message returned by the system. Specific messages correspond to specific return codes.
```

**Query string example**

```
https://www2.1010data.com/gw.k?api=droptable&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML input example**

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

**XML response example**

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

**upload (Upload table)**
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 5 MB limit on the size of the input for this transaction. For large tables, it is recommended to use the PowerLoader API.
Query string

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

api
Specify the name of the API transaction.

apiversion
Specify the version of the API that should handle the requested transaction.
For the most up-to-date version, use apiversion=3.

uid
Specify a valid 1010data user name.

pswd
Specify a valid password for the 1010data user name provided to the uid parameter.
Note: The pswd value should be the encrypted password returned from the login transaction.

sid
The session ID for the current API session.
This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)
Valid values for kill are as follows:

yes
Terminate the existing session and start a new one.

no
Do not terminate the existing session.
If no session exists, a new one will be created. If a session does exist, an error is returned.
Note: If kill=no and a session is active, the system returns an error.

possess
Log in and possess the existing session.

auth
Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

XML input to server

The specifications must contain the following elements:

<table>
A 1010data XML table tree. To see the full XML specification for the table tree, see Table Tree on page 85.

<name>
The content of this element is the full path for the table in the 1010data database hierarchy.
<name> may also contain any of the following optional attributes and corresponding valid values:
**mode**

*mode* specifies how the system will respond when a table is sent that has the same name as an existing table.

- *mode*="replace" will replace the existing table with the new one.
- *mode*="noreplace" will return an error and will not save the new table.
- *mode*="append" will append the new data to an existing table with the name specified.

If *mode* is omitted, the default is *noreplace*.

**materialize**

The *materialize* attribute is used to activate some of the more powerful table-creation features in 1010data, such as segmentation.

To activate these features, use *materialize*="1".

**<users>**

Contains *<user>* elements, each containing a UID that is authorized to view the directory or table.

See *Users Tree* on page 95 for the XML schema.

---

**XML response from server**

A successful upload contains the following elements:

**<rc>**

The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

**<msg>**

The message returned by the system. Specific messages correspond to specific return codes.

---

**Query string example**

https://www2.1010data.com/gw.k?api=upload&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

---

**XML input example**

```
<in>
  <table>
    <title>Store Master</title>
    <cols>
      <th name="store_id" type="i">Store ID</th>
      <th name="store" type="i">Store Number</th>
      <th name="addr" type="a">Address</th>
      <th name="city" type="a">City</th>
      <th name="state" type="a">State</th>
      <th name="zip" type="a">Zip Code</th>
      <th name="sdiv" type="i">Division (store)</th>
      <th name="sqft" type="i">Selling Area Sq Ft.</th>
      <th name="type" type="a">Location Type</th>
    </cols>
  </table>
</in>
```
XML response example

```xml
<out>
  <rc>0</rc>
  <msg>Added table certification.test_upload_table_api</msg>
</out>
```

**puttab (Modify table metadata)**

The **puttab** transaction modifies the meta information for a table.

**Query string**

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

Specify the name of the API transaction.

**apiversion**

Specify the version of the API that should handle the requested transaction.

For the most up-to-date version, use `apiversion=3`.

**uid**

Specify a valid 1010data user name.

**pswd**

Specify a valid password for the 1010data user name provided to the `uid` parameter.

Note: The `pswd` value should be the encrypted password returned from the `login` transaction.

**sid**

The session ID for the current API session.

This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

**kill (optional)**

Valid values for `kill` are as follows:

- **yes**
  
  Terminate the existing session and start a new one.

- **no**
  
  Do not terminate the existing session.
  
  If no session exists, a new one will be created. If a session does exist, an error is returned.

Note: If `kill=no` and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is `yes`.

**XML input to server**

The input specification must contain the following elements:

- **<tab>**
  
  Contains information about a table. The table's metadata is contained within attributes of the `<tab>` element.

Include the following attributes with the `<tab>` element:

Note: Any attributes that are omitted will not be modified.

Note: You must specify both the table name and the table ID to modify directory metadata. If either piece of information is omitted an error is returned.
id
   The unique identification number for the directory, table, or query. The ID is an integer.

name
   The full path to the table, directory, or query to which the metadata belongs.

title
   The title, or label, of the directory, table, or query.

sdesc
   The short description of the directory, table, or query.

ldesc
   The long description of the directory, table, or query.

secure
   1 if the directory, table, or query is marked as "secure" and 0 if it isn't.

   Note: This metadata element is still returned but has been deprecated.

owner
   The owner of the directory, table, or query.

gif
   Returns the location of the gif file to be used to display the directory, table, or query in the UI.

users
   A space-separated list of users and groups whom are allowed to access this directory, table, or query.

XML response from server

A successful response from the server contains the following elements:

The directory metadata is returned as a series of attributes contained within the <tab> element. The attributes are as follows:

id
   The id number of the table. Table id is represented as an integer value.

name
   The full name of the table, represented as a path to the table.

type
   Specifies the type of table. Possible return values are as follows:
   
   • REAL
   • VIEW
   • PARAM
   • MERGED
   • UQ
   • TOLERANT

display
   The output type of the table. Possible return values are as follows:
   
   • TABLE
- EXCEL
- CHART
- REPORT
- TEXT
- XML

**report**
A boolean flag indicating whether the table or query has report specifications saved. Returns a 0 if no specifications are saved and a 1 if there are saved specifications.

**chart**
A boolean flag indicating whether the table or query has chart specifications saved. Returns a 0 if no specifications are saved and a 1 if there are saved specifications.

**title**
The title of the table.

**sdesc**
The short description of the table.

**ldesc**
The long description of the table.

**link**
A string that will be prepended to table headers during a link.

**rows**
The number of rows in the table.

**bytes**
The number of bytes the table consumes on disk.

**segby**
The column on which the table is segmented.

**segs**
The number of segments contained in the table.

**tstat**
Specifies whether the table is can be accessed by time-series functions. If so, returns 1. If not, returns 1.

**access**
Specifies whether the table is accessible. If so, a 1 is returned. If not, a 0 is returned.

**secure**
1 if the directory, table, or query is marked as "secure" and 0 if it isn't.

**Note:** This metadata element is still returned but has been deprecated.

**own**
Boolean flag specifying whether the UID used for the API transaction is the owner of the directory.

**owner**
The owner of the directory, table, or query.
**update**

The date and time the directory’s metadata was last modified. If it was created and never modified, returns the date and time the directory was created.

**users**

A space-separated list of users and groups whom are allowed to access this directory, table, or query.

**maxdown**

The maximum number of rows that can be downloaded from the table at a time.

**favorite**

Specifies whether the table is flagged as a favorite for the user ID making the request. Returns 1 if the table is a favorite and a 0 if it isn’t.

**Query string example**

https://www2.1010data.com/gw.k?api=puttab&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

```xml
<in>
    <tab id="3427628" name="certification.finance.equity" title="Yahoo Finance Equities"/>
</in>
```

**XML response example**

A successful puttab produces the following result, based on the input example:

```xml
<out>
    <rc>0</rc>
    <msg>puttab successful</msg>
    <tab id="3427628" name="certification.finance.equity" type="REAL" display="TABLE" report="0"
        chart="0" title="Yahoo Finance Equities" sdesc="" ldesc="" link=""
        rows="28261234" bytes="1582632792" segs="4" tstat="0"
        access="1" secure="0" maxdown="" own="1" owner="chen2" mode="data"
        segby="ticker" update="2016-02-24 07:36:06" users=""/>
</out>
```

**merge (Merge two tables)**

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:

api
Specify the name of the API transaction.

apiversion
Specify the version of the API that should handle the requested transaction.
For the most up-to-date version, use apiversion=3.

uid
Specify a valid 1010data user name.

pswd
Specify a valid password for the 1010data user name provided to the uid parameter.
  Note: The pswd value should be the encrypted password returned from the login transaction.

sid
The session ID for the current API session.
This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)
Valid values for kill are as follows:
  yes
  Terminate the existing session and start a new one.
  no
  Do not terminate the existing session.
  If no session exists, a new one will be created. If a session does exist, an error is returned.
  Note: If kill=no and a session is active, the system returns an error.

possess
Log in and possess the existing session.

auth
Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

XML input to server

The specifications must contain the following elements:

<tabs>
Used to specify the table(s) to be merged. The <tabs> element must contain one or more <tab> elements, which specify the full location name of each table to be merged.
  Note: A <tabs> element containing at least one <tab> element is required by this transaction.
The full name of the table that will result from the <merge> transaction.

<number> may include optional attribute mode. If a table already exists by this name and
mode="replace", the table will be replaced. If mode="noreplace", an error message will be
returned. If mode is omitted, the default is noreplace.

Note: A <name> element containing a valid 1010data path to the new table is required by the
<merge> transaction.

<title>
The title of the directory or table.

<sdesc>
The short description of the table resulting from the <merge> transaction.

<link>
The link header for the new table resulting from the <merge> transaction. (see <link> (Link header)
on page 93)

<maxdown>
The download limit for the new table (see <maxdown> (Download limit) on page 90).

<users>
The access rights for the new table (see Users Tree on page 95).
</tabs> and <name> are required; the others are optional.

**XML response from server**
A successful merge contains the following elements:

<rc>
The return code generated by the transaction. For a list of return codes, see Return codes on page 7.

<msg>
The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**
https://www2.1010data.com/gw.k?api=merge&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

```xml
<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 Example

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

This group of transactions provides facilities for submitting queries to the system and retrieving the results.

query (Apply a query to a table)

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:

api
Specify the name of the API transaction.

apiversion
Specify the version of the API that should handle the requested transaction.
For the most up-to-date version, use apiversion=3.

uid
Specify a valid 1010data user name.

pswd
Specify a valid password for the 1010data user name provided to the uid parameter.

Note: The pswd value should be the encrypted password returned from the login transaction.

sid
The session ID for the current API session.
This value is returned by the login transaction and must be provided to every transaction (except login).

kill (optional)

Valid values for kill are as follows:

yes
Terminate the existing session and start a new one.

no
Do not terminate the existing session.
If no session exists, a new one will be created. If a session does exist, an error is returned.

Note: If kill=no and a session is active, the system returns an error.

possess
Log in and possess the existing session.

auth
Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

**XML input to server**

Input is provided to the API transaction via an XML specification. The outer-most element of the input specification is the `<in>` element. All input elements and data must be contained by `<in>`.

**<name>**

The full name of a table. Table names are specified as a full path to the location of the table, e.g., `directory1.directory2.directory[N].table`.

**<ops>**

A list of operations for the system to conduct on a table. This is effectively a 1010data macro.

**Note:** The `<ops>` element is not optional. If you want to return a complete table, you must provide an empty `<ops>` element that does not contain a 1010data macro.

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

**XML response from server**

A response from the server contains the following elements:

**<rc>**

The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

**<msg>**

The message returned by the system. Specific messages correspond to specific return codes.

**<nrows>**

The number of rows in the table returned.

**<table>**

A 1010data XML table tree. To see the full XML specification for the table tree, see *Table Tree* on page 85.

The `<table>` element (see *Table Tree* on page 85) 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.

**Query string example**

https://www2.1010data.com/gw.k?api=query&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

```
<in>
    <name>pub.doc.retail.salesdetail</name>
    <ops>
        <link table2="pub.doc.retail.product" col="sku" col2="sku" suffix="_sm"/>
        <sel value="transid=1400190251"/>
    </ops>
</in>
```
XML response example

```xml
<out>
  <rc>0</rc>
  <msg>query successful</msg>
  <nrows>9</nrows>
  <table>
    <cols>
      <th name="trans_date" type="i" fixed="0">Date</th>
      <th name="store" type="i" fixed="0">Store</th>
      <th name="xsales" type="f" fixed="0">Extended Sales</th>
    </cols>
    <data/>
  </table>
</out>
```

**getdata (Get the results of a query)**

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:

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid password for the 1010data user name provided to the `uid` parameter.
    - **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  - The session ID for the current API session.
  - This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
no

Do not terminate the existing session.
If no session exists, a new one will be created. If a session does exist, an error is returned.

**Note:** If `kill=no` and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is `yes`.

**XML input to server**

The specifications must contain the following elements:

**<cols>**

The column name(s) of the column(s) that should be retrieved for the query results. Column names should be contained in `<col>` elements, as shown below:

```
<cols>
  <col>column1</col>
  <col>column2</col>
</cols>
```

**<rows> (optional)**

The row(s) to be retrieved

* `<rows>` must include the attribute `mode`, which indicates the way in which rows to be returned are selected:
  * `mode="1"` allows a relative selection (e.g., next 10 rows)
  * `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>
```

**Note:** `<next>` is required if `mode="1"`.

For `mode="2"`, the form is:

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

**Note:** `<from>` and `<to>` are required if `mode="2"`.

If not specified, all rows are retrieved.

**format**

Specifies the format in which data will be returned. Data may be returned as XML or as a delimited data.

* `<format>` must include the `type` attribute and may include the `values` attribute.

The `type` attribute 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 *(optional)*
- `<linesep>` - Character with which to separate rows *(optional)*

If not specified, the column separator is a comma, and the line separator is a newline (`"\n"`).

The `values` attribute 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 01/01/95.

If `type="csv"` and `values="formatted"`, `<format>` is optional. If not specified, XML data with raw formatting is returned.

**XML response from server**

A successful response from the server contains the following elements:

- `<rc>`
  The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.
- `<msg>`
  The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**

https://www2.1010data.com/gw.k?api=getdata&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML examples of input and response for the getdata transaction**

The examples in this section assume a query transaction has already been successfully executed. The input for the query transaction assumed in these examples is as follows:

```
<in>
  <name>pub.doc.retail.salesdetail</name>
  <ops>
    <link table2="pub.doc.retail.product" col="sku" col2="sku" suffix="_sm"/>
    <sel value="transid=1400190251"/>
  </ops>
</in>
```

**XML Input example 1: relative row specification with data returned in XML**

This example assumes the query transaction has already run. The input for

```
<in>
  <cols>
    <col>transid</col>
    <col>trans_date</col>
    <col>sku</col>
    <col>description_sm</col>
    <col>qty</col>
    <col>xsales</col>
    <col>dept</col>
  </cols>
</in>
```
The input above provides the following response from the server:

```
<out>
  <rc>0</rc>
  <msg>getdata successful</msg>
  <table>
    <cols>
      <th name="transid" type="i" format="type:nocommas;width:6;dec:0" fixed="1">Trans ID</th>
      <th name="trans_date" type="i" fixed="0">Date</th>
      <th name="sku" type="i" format="type:nocommas" fixed="0">SKU</th>
      <th name="description_sm" type="a" fixed="0">Description</th>
      <th name="qty" type="f" format="type:num;width:5;dec:2" fixed="0">Qty/Wgt</th>
      <th name="xsales" type="f" fixed="0">Extended Sales</th>
      <th name="dept" type="i" fixed="0">Department</th>
      <th name="deptdesc_sm" type="a" fixed="0">Dept Desc</th>
    </cols>
    <data>
      <tr>
        <td>1400190251</td>
        <td>20140101</td>
        <td>242846</td>
        <td>PRIVATE LABEL</td>
        <td>1</td>
        <td>2.31</td>
        <td>20</td>
        <td>DAIRY DELI</td>
      </tr>
      <tr>
        <td>1400190251</td>
        <td>20140101</td>
        <td>105755</td>
        <td>CHAMP SLICED HAM 4X6</td>
        <td>1</td>
        <td>3.75</td>
        <td>47</td>
        <td>MEAT DELI</td>
      </tr>
      <tr>
        <td>1400190251</td>
        <td>20140101</td>
        <td>449508</td>
        <td>PRIVATE LABEL</td>
        <td>1</td>
        <td>1.37</td>
        <td>13</td>
        <td>BAKERY</td>
      </tr>
      <tr>
        <td>1400190251</td>
        <td>20140101</td>
    ```
XML input example 2: absolute row specification; tab-separated; raw values

This example differs in two ways from the first. The input for the transaction specifies the same columns to be returned, but also uses mode="2" and provides an absolute range for which rows should be returned. This example also stipulates that the data be returned in a CSV file.

```
<in>
  <cols>
    <col>transid</col>
    <col>trans_date</col>
    <col>sku</col>
    <col>description_sm</col>
    <col>qty</col>
    <col>xsales</col>
    <col>dept</col>
    <col>deptdesc_sm</col>
  </cols>
  <rows mode="2">
    <from>1</from>
    <to>10</to>
  </rows>
  <format type="csv" values="formatted">
    <sep>,</sep>
  </format>
</in>
```

The output produced from the above input is as follows.

```
<out>
  <rc>0</rc>
```
<msg>getdata successful</msg>
<csv>
<! [CDATA["Trans ID","Date","SKU","Description","Qty/Wgt","Extended Sales","Department","Dept Desc"
1400190251,20140101,242846,"PRIVATE LABEL",1,2.31,20,"DAIRY DELI"
1400190251,20140101,105755,"CHAMP SLICED HAM 4x6",1,3.75,47,"MEAT DELI"
1400190251,20140101,449508,"PRIVATE LABEL",1,1.37,13,"BAKERY"
1400190251,20140101,247560,"BANANAS - WIC",2.2,1.34,36,"PRODUCE"
1400190251,20140101,118708,"FSHR IMITATION CHEESE SINGLE 1",1,1.13,20,"DAIRY DELI"
1400190251,20140101,405883,"PRIVATE LABEL",2,3.15,55,"DAIRY"]]>
</csv>
</out>

XML input example 3: absolute row specification; comma-separated; formatted values

<in>
<cols>
<col>transid</col>
<col>trans_date</col>
<col>sku</col>
<col>description_sm</col>
<col>qty</col>
<col>xsales</col>
<col>dept</col>
<col>deptdesc_sm</col>
</cols>
<rows mode="2">
<from>1</from>
<to>10</to>
</rows>
<format type="csv" values="formatted">
<sep>,</sep>
</format>
</in>

The input above specifies that formatted values be returned. The response from the server is as follows:

<out>
<rc>0</rc>
msgidata successful</msg>
<csv>
<! [CDATA["Trans ID","Date","SKU","Description","Qty/Wgt","Extended Sales","Department","Dept Desc"
1400190251,01/01/14,242846,"PRIVATE LABEL",1.00,2.31,20,"DAIRY DELI"
1400190251,01/01/14,105755,"CHAMP SLICED HAM 4x6",1.00,3.75,47,"MEAT DELI"
1400190251,01/01/14,449508,"PRIVATE LABEL",1.00,1.37,13,"BAKERY"
1400190251,01/01/14,247560,"BANANAS - WIC",2.20,1.34,36,"PRODUCE"
1400190251,01/01/14,118708,"FSHR IMITATION CHEESE SINGLE 1",1.00,1.13,20,"DAIRY DELI"
1400190251,01/01/14,405883,"PRIVATE LABEL",2.00,3.15,55,"DAIRY"]]>
</csv>
querydata (Apply a query to a table and get results)

The `querydata` transaction applies a query to a table and gets the results as if you called a query and a single `getdata` transaction.

`querydata` 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:

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid password for the 1010data user name provided to the `uid` parameter.
    - **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  - The session ID for the current API session.
  - This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
    - **no**
      - Do not terminate the existing session.
      - If no session exists, a new one will be created. If a session does exist, an error is returned.
        - **Note:** If `kill=no` and a session is active, the system returns an error.
    - **possess**
      - Log in and possess the existing session.
    - **auth**
      - Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

**XML Input to Server**

The specifications must contain the following elements:

See XML input to *query* and *getdata*, as the *querydata* transaction requires the input of both *query* and *getdata* transactions.

**XML Response from Server**

A successful *querydata* contains the following elements:

- `<rc>`
  - The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

- `<msg>`
  - The message returned by the system. Specific messages correspond to specific return codes.

- `<nrows>`
  - The number of rows in the table returned.

- `<table>`
  - A 1010data XML table tree. To see the full XML specification for the table tree, see *Table Tree* on page 85.

**Query string example**

https://www2.1010data.com/gw.k?api=querydata&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input example**

```
<in>
  <name>pub.doc.retail.salesdetail</name>
  <ops>
    <sel value="i <10"/>
    <colord cols="trans_date,store,xsales"/>
  </ops>
  <cols>
    <col>trans_date</col>
    <col>store</col>
    <col>xsales</col>
  </cols>
  <rows mode="1">
    <next>3</next>
  </rows>
  <format type="xml"/>
</in>
```

**XML response example**

```
<out>
  <rc>0</rc>
  <msg>querydata successful</msg>
  <table>
```

savefile (Save query results as a file)

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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  **Note:** The `pswd` value should be the encrypted password returned from the login transaction.

- **sid**
  The session ID for the current API session.
This value is returned by the login transaction and must be provided to every transaction (except login).

**kill (optional)**

Valid values for kill are as follows:

- **yes**
  - Terminate the existing session and start a new one.
- **no**
  - Do not terminate the existing session.
  - If no session exists, a new one will be created. If a session does exist, an error is returned.

  **Note:** If kill=no and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is yes.

**XML Input to Server**

The specifications must contain the following elements:

- `<file>`
  - The name of the result file
  - If a file already exists with that name, it will be replaced.

- `<content>`
  - The meta information to include in the file *(optional)*
  - `<content>` must contain one or more `<meta>` elements, each identifying one type of information.
  - Each `<meta>` element must include attribute `type` and `type` may have one of the following values:
    - `headers` - Column headers
    - `names` - Column names
  - If not specified, no meta information is included in the file.

- `<format>`
  - The data format for the result *(optional)*
  - `<format>` has no attributes and may contain the following elements:
    - `<sep>` - Character with which to separate fields (columns)
    - `<linesep>` - Character with which to separate records (rows)
  - If not specified, the column separator is a comma, and the line separator is newline (\n).

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

**Query string example**

https://www2.1010data.com/gw.k?api=savefile&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
**XML input example**

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

**XML response from server**

A successful `savefile` contains the following elements:

```xml
<rc>
  The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

<msg>
  The message returned by the system. Specific messages correspond to specific return codes.
</msg>
```

**XML response example**

```xml
<out>
  <rc>0</rc>
  <msg>Saved file My File</msg>
</out>
```
SAM pool management

Provides functionality for retrieving and releasing UIDs from a SAM pool.

1010data SAM pools provide a way to share a limited number of user accounts with a larger group of people and processes that need them. SAM pools can be used to provide 1010data to a group of users on demand or to provide concurrent processes in an application with access to multiple 1010data sessions.

SAM pools retrieve user IDs from 1010data user groups. To make SAM pools easier to work with and to avoid errors and mistakes in your applications, there are some rules that should be followed when configuring a group for use with SAM pools.

- Use the same password for all the IDs in the SAM pool. While not required, it makes your code more concise and easier to manage.
- Groups should not be members of your SAM pool group. If a group is a member of the SAM pool, it will be returned as a UID. A login transaction conducted with a group ID will fail.
- Make sure all the IDs in your group have the same permissions and object access. This will ensure that the same data and features are available to anyone who retrieves an ID from the SAM Pool.
- The ID’s in your SAM Pool should be long to that group and that group only.

**getuid (Retrieve a UID)**

Retrieves an available UID from a SAM pool.

**Query string**

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

- `api`
  Specify the name of the API transaction.

- `apiversion`
  Specify the version of the API that should handle the requested transaction.

  For the most up-to-date version, use `apiversion=3`.

- `uid`
  Specify a valid 1010data user name.

- `gid`
  The group identifier for the SAM pool. In 1010data, a SAM pool is a group containing some number of UID's. A valid `gid` follows the same rules as a valid `uid`.

**kill (optional)**

Valid values for `kill` are as follows:

- `yes`
  Terminate the existing session and start a new one.

- `no`
  Do not terminate the existing session.

  If no session exists, a new one will be created. If a session does exist, an error is returned.

  **Note:** If `kill=no` and a session is active, the system returns an error.
Log in and possess the existing session.

```
auth
```

Authenticate with the system but do not kill, possess, or start any session.

The default is yes.

**Additional info**

When using the `getuid` transaction, the UID provided must be the owner of the group specified in the `gid` parameter of the query string, and the password provided must be the unencrypted password for that ID.

**XML input to server**

No XML input is required. All data for the transaction is specified in the connection string.

**XML response from server**

A successful result contains the following elements:

<rc>

The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

<msg>

The message returned by the system. Specific messages correspond to specific return codes.

<uid>

The name of the UID that is part of the specified group and currently not in use by another SAM pool process. This response is a string.

**Query string example**

```
https://www2.1010data.com/gw.k?api=getuid&apiversion=3&uid=$UID&pswd=$PSWD&gid=$GID
```

**XML response example**

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

**reluid (Release a UID)**

Releases a UID and make it available for future allocation.

**Query string**

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

```
api
```

Specify the name of the API transaction.

```
apiversion
```

Specify the version of the API that should handle the requested transaction.
For the most up-to-date version, use `apiversion=3`.

**uid**

Specify a valid 1010data user name.

**pswd**

The encrypted password for the session. If no active session exists, use the unencrypted password for the UID. See the section titled **Additional info** for more information.

**sid**

The session ID for the current API session.

This value is returned by the `login` transaction and must be provided for most subsequent transactions. Provide this value unless the session has ended or you no longer have access to the session information. See the section titled **Additional info** for more information.

**gid**

The group identifier for the SAM pool. The SAM pool is a group containing some number of UID's. A valid `gid` follows the same rules as a valid `uid`.

**Additional info**

When using the `reluid` transaction, the UID provided must be the 1010data UID that should be released back to the SAM pool for future allocation, and the password provided must be the password for that ID.

As a best practice, you should provide the session ID (`sid`) and encrypted password (`pswd`) if the UID being released has an active session. If the UID being released has been logged out, or if you no longer have access to the session credentials, you can release the UID by providing the unencrypted password for that UID. In this case, the session ID (`sid`) is not required.

**XML input to server**

No XML input is required. All data for the transaction is specified in the connection string.

**XML response from server**

A successful result contains the following elements:

- `<rc>`
  - The return code generated by the transaction. For a list of return codes, see `Return codes` on page 7.

- `<msg>`
  - The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**

```
https://www2.1010data.com/gw.k?api=reluid&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID&gid=$GID
```

**XML response example**

```
<out>
  <rc>0</rc>
  <msg>uid released</msg>
</out>
```
**markgid (Flush a SAM pool)**

Flush a SAM pool so that logins with UIDs from that pool are guaranteed to have sessions no older than the `markgid` transaction response.

**Query string**

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

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
    - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid unencrypted 1010data password for the 1010data user name provided to the `uid` parameter.
    - The response from the `login` transaction will provide an encrypted password, which must be supplied as the value for the `pswd` attribute for all the other transactions.

- **gid**
  - The group identifier for the SAM pool. In the 1010data Insights Platform, a SAM pool is a group containing some number of UIDs. A valid `gid` follows the same rules as a valid `uid`.

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
    - **no**
      - Do not terminate the existing session.
      - If no session exists, a new one will be created. If a session does exist, an error is returned.
        - **Note:** If `kill=no` and a session is active, the system returns an error.
    - **possess**
      - Log in and possess the existing session.
    - **auth**
      - Authenticate with the system but do not kill, possess, or start any session.
  - The default is `yes`.

**Additional info**

When using the `markgid` transaction, the UID provided must be the owner of the group specified in the `gid` parameter of the query string, and the password provided must be the unencrypted password for that UID.
**XML input to server**

No XML input is required. All data for the transaction is specified in the connection string.

**XML response from server**

A successful result contains the following elements:

- `<rc>`
  - The return code generated by the transaction. For a list of return codes, see *Return codes* on page 7.

- `<msg>`
  - The message returned by the system. Specific messages correspond to specific return codes.

**Query string example**

```
https://www2.1010data.com/gw.k?api=markgid&apiversion=3&uid=$UID&pswd=$PSWD&gid=$GID
```

**XML response example**

```
<out>
  <rc>0</rc>
  <msg>OK</msg>
</out>
```
PowerLoader

PowerLoader provides several transactions for uploading and managing large tables. For tables larger than tens of thousands of rows, use the addtab transaction in favor of the upload transaction.

**addtab (Load a large table)**

The addtab transaction loads a large table into the 1010data system.

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, 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 mode is recommended for large tables (>1 million records).

**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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.

  For the most up-to-date version, use apiversion=3.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the uid parameter.

  **Note:** The pswd value should be the encrypted password returned from the login transaction.

- **sid**
  The session ID for the current API session.

  This value is returned by the login transaction and must be provided to every transaction (except login).

- **kill (optional)**
  Valid values for kill are as follows:

  - **yes**
    Terminate the existing session and start a new one.
Do not terminate the existing session.

If no session exists, a new one will be created. If a session does exist, an error is returned.

**Note:** If `kill=no` and a session is active, the system returns an error.

**possess**

Log in and possess the existing session.

**auth**

Authenticate with the system but do not kill, possess, or start any session.

The default is `yes`.

**Query String Example**

https://www2.1010data.com/gw.k?api=addtab&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML Input to Server**

The specifications must contain the following elements:

<table>
<thead>
<tr>
<th>XML Element To Send</th>
<th>Description of Sent Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;sync&gt;</code></td>
<td>Specifies whether you want to load synchronously or not&lt;br&gt;Possible values are:&lt;br&gt;• 1 if you want to load synchronously&lt;br&gt;• 0 if you want the table to load asynchronously</td>
</tr>
<tr>
<td><code>&lt;spec&gt;</code></td>
<td>Outer specification element <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;source&gt;</code></td>
<td>List of source files <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;file&gt;</code></td>
<td>File element</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>File path of table in 1010data <em>(e.g., mycompany.mytable)</em>. If you reference an existing table, <code>addtab</code> will replace it</td>
</tr>
<tr>
<td><code>&lt;title&gt;</code></td>
<td>Title as it appears in the user interface</td>
</tr>
<tr>
<td><code>&lt;sdesc&gt;</code></td>
<td>Short description of the table</td>
</tr>
<tr>
<td><code>&lt;ldesc&gt;</code></td>
<td>Long description of table</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Link header</td>
</tr>
<tr>
<td><code>&lt;users type=&quot;[TYPE]&quot;&gt;&lt;/users&gt;</code></td>
<td>Specifies users provisioned for table access.&lt;br&gt;See <code>Users Tree</code> on page 95 for XML schema and <code>&lt;users&gt;</code> <em>(Top-level wrapper for users tree)</em> on page 95 for usage details.</td>
</tr>
<tr>
<td><code>&lt;user&gt;</code></td>
<td>Specifies each user that may access the table.&lt;br&gt;Used in conjunction with the <code>&lt;users&gt;</code> element when <code>type</code> is <code>list</code> (or if <code>type</code> is omitted).</td>
</tr>
<tr>
<td>XML Element To Send</td>
<td>Description of Sent Element Content</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td><code>&lt;rectype&gt;</code></td>
<td>Each <code>&lt;user&gt;</code> element is nested inside the <code>&lt;users&gt;</code> top-level wrapper.</td>
</tr>
<tr>
<td><code>&lt;sep&gt;</code></td>
<td><code>&lt;rectype&gt;</code> fixed or separated <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;sep&gt;</code></td>
<td>Single character delimiter. Required if <code>&lt;rectype&gt;</code> is separated.</td>
</tr>
<tr>
<td><code>&lt;eor&gt;</code></td>
<td>End of record delimiter</td>
</tr>
<tr>
<td><code>&lt;arch&gt;</code></td>
<td>Byte order of data - little or big endian</td>
</tr>
<tr>
<td><code>&lt;begbytes&gt;</code></td>
<td>Number of bytes to skip at the beginning of e/file</td>
</tr>
<tr>
<td><code>&lt;begrecs&gt;</code></td>
<td>Number of records to skip in the beginning of e/file</td>
</tr>
<tr>
<td><code>&lt;numrecs&gt;</code></td>
<td>Number of records to load in e/file</td>
</tr>
<tr>
<td><code>&lt;ts&gt;</code></td>
<td>Time-series flag. 1 for true, or 0 for false. Requires that <code>&lt;bord&gt;</code> tags are provided in at least one column.</td>
</tr>
<tr>
<td><code>&lt;cols&gt;</code></td>
<td>Column specifications <em>(required)</em> List of <code>&lt;col&gt;</code> elements, one for each column in the file.</td>
</tr>
<tr>
<td><code>&lt;col&gt;</code></td>
<td>A column element <em>(required)</em> There must be one of these for each column in the table.</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>Field name <em>(required)</em> Must begin with an alphabetical character.</td>
</tr>
<tr>
<td><code>&lt;width&gt;</code></td>
<td>Field width in the raw file <em>(required)</em> (Optional if <code>&lt;rectype&gt;</code> is separated.)</td>
</tr>
<tr>
<td><code>&lt;help&gt;</code></td>
<td>Column help (text inside ? marker above column)</td>
</tr>
<tr>
<td><code>&lt;skip&gt;</code></td>
<td>Field is read but not displayed in table</td>
</tr>
<tr>
<td><code>&lt;type&gt;</code></td>
<td>Field data type <em>(required)</em> See Field Data Types on page 82.</td>
</tr>
<tr>
<td><code>&lt;scale&gt;</code></td>
<td>Divisor used to divide raw value.</td>
</tr>
<tr>
<td><code>&lt;case&gt;</code></td>
<td>Force case, either upper or lower</td>
</tr>
<tr>
<td><code>&lt;nowrite&gt;</code></td>
<td>Ignore the field; don’t include it in the table.</td>
</tr>
<tr>
<td><code>&lt;order&gt;</code></td>
<td>Column placement relative to other columns (integer)</td>
</tr>
<tr>
<td><code>&lt;fix&gt;</code></td>
<td>Column is non-scrolling (pink)</td>
</tr>
<tr>
<td><code>&lt;head&gt;</code></td>
<td>Column header Use <code>\</code> to separate lines.</td>
</tr>
<tr>
<td><code>&lt;exp&gt;</code></td>
<td>1010 expression applied to raw data (not nested)</td>
</tr>
<tr>
<td><code>&lt;format&gt;</code></td>
<td>Formatting Spec Must specify <code>&lt;type&gt;</code>, <code>&lt;width&gt;</code>, and <code>&lt;dec&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>XML Element To Send</td>
<td>Description of Sent Element Content</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>&lt;bord&gt;&lt;/bord&gt;</td>
<td>TS segmentation order (integer). The number provided here is the relative segmentation order of this column in relation to other columns with the <code>&lt;bord&gt;</code> tag. Think of <code>&lt;bord&gt;</code> columns as tabulation breaks. The column that denotes time should have the largest number in the sequence.</td>
</tr>
</tbody>
</table>

**XML Input Example**

```xml
<in>
<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>Distribution Date</head>
      <type>yyyyymmdd</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>
</in>
```
XML Response from Server

A successful addtab produces the following result:

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

**status (Status of the addtab API)**

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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use *apiversion=3*.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the *uid* parameter.
  
  **Note:** The *pswd* value should be the encrypted password returned from the login transaction.

- **sid**
  The session ID for the current API session.
  This value is returned by the login transaction and must be provided to every transaction (except login).

- **kill (optional)**
  Valid values for *kill* are as follows:

  - **yes**
    Terminate the existing session and start a new one.

  - **no**
    Do not terminate the existing session.
    If no session exists, a new one will be created. If a session does exist, an error is returned.
    
    **Note:** If *kill=no* and a session is active, the system returns an error.

  - **possess**
    Log in and possess the existing session.

  - **auth**
    Authenticate with the system but do not kill, possess, or start any session.
The default is yes.

**Query String Example**

https://www2.1010data.com/gw.k?api=status&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID

**XML input to server**

None.

**XML response from server**

A successful status produces the following result:

```xml
<rc>
    The return code generated by the transaction. For a list of return codes, see Return codes on page 7.
</rc>

<msg>
    The message returned by the system. Specific messages correspond to specific return codes.
</msg>

<numrecs>
    The number of records written to the table so far. This information is returned if the addtab transaction is in the loading state.
</numrecs>
```

<table>
<thead>
<tr>
<th>XML Element Returned</th>
<th>Description of Returned Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rc&gt;</td>
<td>Return code</td>
</tr>
<tr>
<td>&lt;msg&gt;</td>
<td>Message from the server</td>
</tr>
<tr>
<td>&lt;numrecs&gt;</td>
<td>Number of records written to the table so far</td>
</tr>
<tr>
<td></td>
<td>Reported if addtab is in the Loading state.</td>
</tr>
<tr>
<td>&lt;totrecs&gt;</td>
<td>Estimate of the number of records in the source file</td>
</tr>
</tbody>
</table>

**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.

**XML Response Example**

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

**convert (Convert a legacy spec)**

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:

- **api**
  Specify the name of the API transaction.

- **apiversion**
  Specify the version of the API that should handle the requested transaction.
  For the most up-to-date version, use `apiversion=3`.

- **uid**
  Specify a valid 1010data user name.

- **pswd**
  Specify a valid password for the 1010data user name provided to the `uid` parameter.
  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  The session ID for the current API session.
  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  Valid values for `kill` are as follows:
  - **yes**
    Terminate the existing session and start a new one.
  - **no**
    Do not terminate the existing session.
    If no session exists, a new one will be created. If a session does exist, an error is returned.
    **Note:** If `kill=no` and a session is active, the system returns an error.

- **possess**
  Log in and possess the existing session.

- **auth**
  Authenticate with the system but do not kill, possess, or start any session.
  The default is `yes`.

**Query String Example**

```
https://www2.1010data.com/gw.k?api=convert&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML Input to Server**

The specifications must contain the following elements:
<table>
<thead>
<tr>
<th>XML Element To Send</th>
<th>Description of Sent Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;&lt;/name&gt;</td>
<td>The spec filename to convert located on your FTP account. See PowerLoader User's Guide for file format.</td>
</tr>
</tbody>
</table>

XML Response from Server

A successful `convert` produces the following result:

<table>
<thead>
<tr>
<th>XML Element Returned</th>
<th>Description of Returned Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rc&gt;</td>
<td>Return code</td>
</tr>
<tr>
<td>&lt;msg&gt;</td>
<td>Message from the system</td>
</tr>
<tr>
<td>&lt;spec&gt;</td>
<td>XML Spec file – See <code>addtab (Load a large table)</code> on page 69 for transaction XML Input.</td>
</tr>
</tbody>
</table>

XML Response Example

```xml
<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>Distribution Date</head>
      <type>yyyyMMdd</type>
      <format>
        <type>date</type>
        <width>8</width>
      </format>
    </col>
  </cols>
</spec>
```
**validate (Validate a table)**

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

### Query string

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

- **api**
  - Specify the name of the API transaction.

- **apiversion**
  - Specify the version of the API that should handle the requested transaction.
  - For the most up-to-date version, use `apiversion=3`.

- **uid**
  - Specify a valid 1010data user name.

- **pswd**
  - Specify a valid password for the 1010data user name provided to the `uid` parameter.
    - **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- **sid**
  - The session ID for the current API session.
  - This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- **kill (optional)**
  - Valid values for `kill` are as follows:
    - **yes**
      - Terminate the existing session and start a new one.
    - **no**
      - Do not terminate the existing session.
      - If no session exists, a new one will be created. If a session does exist, an error is returned.
        - **Note:** If `kill=no` and a session is active, the system returns an error.
    - **possess**
      - Log in and possess the existing session.
    - **auth**
      - Authenticate with the system but do not kill, possess, or start any session.
The default is **yes**.

**Query String Example**

```
https://www2.1010data.com/gw.k?api=validate&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
```

**XML Input to Server**

The specifications must contain the following elements:

<table>
<thead>
<tr>
<th>XML Element To Send</th>
<th>Description of Sent Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;sync&gt;</code></td>
<td>Specifies whether you want to load synchronously or not</td>
</tr>
<tr>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• 1 if you want to load synchronously</td>
</tr>
<tr>
<td></td>
<td>• 0 if you want the table to load asynchronously</td>
</tr>
<tr>
<td><code>&lt;spec&gt;</code></td>
<td>Outer specification element <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;source&gt;</code></td>
<td>List of source files <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;file&gt;</code></td>
<td>File element</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>File path of table in 1010data (e.g., mycompany.mytable). If you reference an existing table, adtab will replace it.</td>
</tr>
<tr>
<td><code>&lt;title&gt;</code></td>
<td>Title as it appears in the user interface</td>
</tr>
<tr>
<td><code>&lt;sdesc&gt;</code></td>
<td>Short description of the table</td>
</tr>
<tr>
<td><code>&lt;ldesc&gt;</code></td>
<td>Long description of table</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Link header</td>
</tr>
<tr>
<td><code>&lt;users type=&quot;[TYPE]&quot;&gt;</code></td>
<td>Specifies users provisioned for table access. If the <code>&lt;users&gt;</code> element is excluded, only the table owner may access the table. See <code>Users Tree</code> on page 95 for XML schema and <code>&lt;users&gt;</code> <em>(Top-level wrapper for users tree)</em> on page 95 for usage details.</td>
</tr>
<tr>
<td><code>&lt;user&gt;</code></td>
<td>Specifies each user that may access the table. Used in conjunction with the <code>&lt;users&gt;</code> element when type is list (or if type is omitted). Each <code>&lt;user&gt;</code> element is nested inside the <code>&lt;users&gt;</code> top-level wrapper.</td>
</tr>
<tr>
<td><code>&lt;rectype&gt;</code></td>
<td>fixed or separated <em>(required)</em></td>
</tr>
<tr>
<td><code>&lt;sep&gt;</code></td>
<td>Single character delimiter. Required if rectype is separated.</td>
</tr>
<tr>
<td><code>&lt;eor&gt;</code></td>
<td>End of record delimiter</td>
</tr>
<tr>
<td><code>&lt;arch&gt;</code></td>
<td>Byte order of data - little or big endian</td>
</tr>
<tr>
<td><code>&lt;begbytes&gt;</code></td>
<td>Number of bytes to skip at the beginning of e/file</td>
</tr>
<tr>
<td><code>&lt;begrecs&gt;</code></td>
<td>Number of records to skip in the beginning of e/file</td>
</tr>
</tbody>
</table>
### XML Element To Send | Description of Sent Element Content
--- | ---
<numrecs></numrecs> | Number of records to load in e/file
<ts></ts> | Time-series flag. 1 for true, or 0 for false. Requires that <bord> tags are provided in at least one column.
<cols> | Column specifications *(required)* List of <col> elements, one for each column in the file.
<col> | A column element *(required)* There must be one of these for each column in the table.
<name></name> | Field name *(required)* Must begin with an alphabetical character.
<width></width> | Field width in the raw file *(required)* (Optional if rectype is separated.)
<help></help> | Column help (text inside ? marker above column)
<skip></skip> | Field is read but not displayed in table
<type></type> | Field data type *(required)* See *Field Data Types* on page 82.
<scale></scale> | Divisor used to divide raw value.
<case></case> | Force case, either upper or lower
<nowrite></nowrite> | Ignore the field; don’t include it in the table.
<order></order> | Column placement relative to other columns (integer)
<fix></fix> | Column is non-scrolling (pink)
<head></head> | Column header Use ` to separate lines.
<exp></exp> | 1010 expression applied to raw data (not nested)
=format></format> | Formatting Spec Must specify <type></type>, <width></width>, and <dec></dec> See *Field Format Types* on page 83.
<bord></bord> | TS segmentation order (integer). The number provided here is the relative segmentation order of this column in relation to other columns with the <bord> tag. Think of <bord> columns as tabulation breaks. The column that denotes time should have the largest number in the sequence.

### XML Input Example

```xml
<in>
```
<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>yyyymmd</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 validate produces the following result:

<table>
<thead>
<tr>
<th>XML Element Returned</th>
<th>Description of Returned Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rc&gt;</td>
<td>Return code</td>
</tr>
<tr>
<td>&lt;msg&gt;</td>
<td>Message from the system</td>
</tr>
</tbody>
</table>

XML Response Example

<rc>0</rc>
**edittab (Edit a table)**

The edittab transaction changes the table metadata (table information acquired via tabinfo).

**Query string**

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

- `api`  
  Specify the name of the API transaction.

- `apiversion`  
  Specify the version of the API that should handle the requested transaction.  
  For the most up-to-date version, use `apiversion=3`.

- `uid`  
  Specify a valid 1010data user name.

- `pswd`  
  Specify a valid password for the 1010data user name provided to the `uid` parameter.  
  **Note:** The `pswd` value should be the encrypted password returned from the `login` transaction.

- `sid`  
  The session ID for the current API session.  
  This value is returned by the `login` transaction and must be provided to every transaction (except `login`).

- `kill` (optional)  
  Valid values for `kill` are as follows:
  
  - `yes`  
    Terminate the existing session and start a new one.
  
  - `no`  
    Do not terminate the existing session.  
    If no session exists, a new one will be created. If a session does exist, an error is returned.  
    **Note:** If `kill=no` and a session is active, the system returns an error.

  - `possess`  
    Log in and possess the existing session.

  - `auth`  
    Authenticate with the system but do not kill, possess, or start any session.  
    The default is `yes`.

**Query String Example**

https://www2.1010data.com/gw.k?api=edittab&apiversion=3&uid=$UID&pswd=$PSWD&sid=$SID
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 `tabinfo (Get information about a table)` on page 33 for detailed info about the response specification.

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.

<table>
<thead>
<tr>
<th>XML Element To Send</th>
<th>Description of Sent Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;title&gt;</code></td>
<td>Table title</td>
</tr>
<tr>
<td><code>&lt;secure&gt;</code></td>
<td>Secure SSL access</td>
</tr>
<tr>
<td><code>&lt;owner&gt;</code></td>
<td>Table owner</td>
</tr>
<tr>
<td><code>&lt;ldesc&gt;</code></td>
<td>Long description of the table</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Link header</td>
</tr>
<tr>
<td><code>&lt;cols&gt;</code></td>
<td>Column information</td>
</tr>
</tbody>
</table>

**XML Input Example**

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

**XML Response from Server**

A successful result contains the following elements:

<table>
<thead>
<tr>
<th>XML Element Returned</th>
<th>Description of Returned Element Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;rc&gt;</code></td>
<td>Return code</td>
</tr>
<tr>
<td><code>&lt;msg&gt;</code></td>
<td>Message</td>
</tr>
</tbody>
</table>

**XML Response Example**

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

**Field Data Types**

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>text</code></td>
<td>Text (Picture)</td>
</tr>
<tr>
<td><code>int</code></td>
<td>Integer (Picture)</td>
</tr>
<tr>
<td><code>J</code></td>
<td>Big integer (Available as of version 11.25)</td>
</tr>
</tbody>
</table>
### Field Format Types

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

<table>
<thead>
<tr>
<th>char</th>
<th>Alphanumeric Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>Number: 1,234,567.89</td>
</tr>
</tbody>
</table>

---

### Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>Float (Picture)</td>
</tr>
<tr>
<td>1Btext</td>
<td>1-Byte Char (Binary)</td>
</tr>
<tr>
<td>1Bint</td>
<td>1-Byte Int (Binary)</td>
</tr>
<tr>
<td>2Bint</td>
<td>2-Byte Int (Binary)</td>
</tr>
<tr>
<td>4Bint</td>
<td>4-Byte Int (Binary)</td>
</tr>
<tr>
<td>8Bint</td>
<td>8-Byte Int (Binary)</td>
</tr>
<tr>
<td>4Bfloat</td>
<td>4-Byte Float (Binary)</td>
</tr>
<tr>
<td>8Bfloat</td>
<td>8-Byte Double (Binary)</td>
</tr>
<tr>
<td>dec</td>
<td>Decimal (BCD)</td>
</tr>
<tr>
<td>Signed</td>
<td>Signed (= Zoned)</td>
</tr>
</tbody>
</table>

### Date Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDD</td>
<td>Date</td>
</tr>
<tr>
<td>YYMMDD</td>
<td>Date</td>
</tr>
<tr>
<td>MMDDYY</td>
<td>Date</td>
</tr>
<tr>
<td>DDMMYY</td>
<td>Date</td>
</tr>
<tr>
<td>MMDDYYYY</td>
<td>Date</td>
</tr>
<tr>
<td>DDMMYYYY</td>
<td>Date</td>
</tr>
<tr>
<td>YYYYMM</td>
<td>Month/Year</td>
</tr>
<tr>
<td>YYMM</td>
<td>Month/Year</td>
</tr>
<tr>
<td>MMYY</td>
<td>Month/Year</td>
</tr>
<tr>
<td>MMYYYY</td>
<td>Month/Year</td>
</tr>
<tr>
<td>HHMM</td>
<td>Time</td>
</tr>
</tbody>
</table>

All non-numeric characters in dates are ignored. For example, 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.
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nocommas</td>
<td>Number: 1234567.89</td>
</tr>
<tr>
<td>date</td>
<td>Date: 10/15/98</td>
</tr>
<tr>
<td>month</td>
<td>Month/Year: 10/03</td>
</tr>
<tr>
<td>quarter</td>
<td>Quarter/Year: 2Q03</td>
</tr>
<tr>
<td>hms24</td>
<td>Time: 22:45:56</td>
</tr>
<tr>
<td>datehms24</td>
<td>Date+Time: 10/15/98_22:45:56</td>
</tr>
</tbody>
</table>
The Table Tree is an input specification for uploading tables to 1010data.

The overall format is as follows:

```xml
<table name="table name">
  <title>table title</title>
  <totals>for tabulations</totals>
  <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>
```

For example, consider the following table:
This table can be represented by the following table tree:

```html
<table>
  <title>Table Tree Example</title>
  <sdesc>Example of a table tree</sdesc>
  <cols>
    <th name="transid" type="i">Transaction ID</th>
    <th name="store" type="i">Store</th>
    <th name="sumsales" type="f" format="type:currency">Sum of Sales</th>
    <th name="totunits" type="f" format="type:num;width:4;dec:0">Total Units</th>
  </cols>
  <data>
    <tr>
      <td>531</td>
      <td>1</td>
      <td>$-5.00</td>
      <td>-1</td>
    </tr>
    <tr>
      <td>532</td>
      <td>1</td>
      <td>$1.60</td>
      <td>2</td>
    </tr>
    <tr>
      <td>534</td>
      <td>1</td>
      <td>$8.25</td>
      <td>3</td>
    </tr>
    <tr>
      <td>535</td>
      <td>2</td>
      <td>$3.85</td>
      <td>3</td>
    </tr>
    <tr>
      <td>536</td>
      <td>2</td>
      <td>$6.02</td>
      <td>4</td>
    </tr>
    <tr>
      <td>537</td>
      <td>3</td>
      <td>$10.35</td>
      <td>6</td>
    </tr>
    <tr>
      <td>538</td>
      <td>1</td>
      <td>$6.95</td>
      <td>8</td>
    </tr>
    <tr>
      <td>539</td>
      <td>2</td>
      <td>$5.00</td>
      <td>1</td>
    </tr>
    <tr>
      <td>540</td>
      <td>2</td>
      <td>$1.44</td>
      <td>2</td>
    </tr>
    <tr>
      <td>541</td>
      <td>3</td>
      <td>$10.00</td>
      <td>7</td>
    </tr>
    <tr>
      <td>543</td>
      <td>1</td>
      <td>$11.39</td>
      <td>9</td>
    </tr>
  </data>
</table>
```
The component elements are described in detail in the following sections. 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.

<table> (Top-level wrapper for table tree)

The table tree is required and may have one attribute (name) and must contain certain elements.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of table</td>
</tr>
</tbody>
</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 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 Info > About this Table in the 1010data web interface.

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.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;title&gt;</code></td>
<td>Title of table</td>
</tr>
<tr>
<td><code>&lt;sdesc&gt;</code></td>
<td>Short description of table</td>
</tr>
<tr>
<td><code>&lt;ldesc&gt;</code></td>
<td>Long description of table</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Link header</td>
</tr>
<tr>
<td><code>&lt;maxdown&gt;</code></td>
<td>Download limit</td>
</tr>
<tr>
<td><code>&lt;cols&gt;</code></td>
<td>Column meta information</td>
</tr>
<tr>
<td><code>&lt;data&gt;</code></td>
<td>Table data</td>
</tr>
</tbody>
</table>

#### `<title>` (Table title)

Returns the title of a table. The table title appears above the table when it is displayed in the 1010data web interface.

**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 web interface). For example, in the screenshot below, **Sales Item Detail** is the table title.
### \(<sdesc>\) (Short description of a table)

Returns the short description of a table from the table metadata.

**Attributes**

None

**Contents**

The short description of the table (i.e., the text that appears in the Short Description field in the 1010data web interface).

---

### \(<ldesc>\) (Long description of a table)

Returns the long description of a table from the table metadata.

**Attributes**

None

**Contents**

The long description of the table (i.e., the text that appears when the user clicks on the table in the 1010data file viewer and then clicks View Info to see information about the table). For example, in the screenshot below, the text that appears under the heading Long Description.
<data> (Table data)

Returns the data in a table.

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> (Data for one row) on page 94).

<maxdown> (Download limit)

The maximum amount of data that can be downloaded at one time.

Attributes

None

Contents

The maximum number of data items (rows × columns) that a user of the 1010data web interface may download at one time. The download limit must be a non-negative whole number.

<segmentation> (Specify the segmentation of a table)

Specifies which columns should define the segmentation of a table.
Attributes
None

Contents
The `<segmentation>` container is used by the `addtab` transaction to segment tables along the specified columns.

Example

```
<segmentation cols="col1,col2" [advise="0"]>
  <advise [sort="1"] cols="col1,col3"/>
</segmentation>
```

The `cols` attribute of `<segmentation>` specifies the columns to segment along. The `<segmentation>` block can contain an arbitrary number of `<advise>` tags. Each advise tag indicates that a particular sequence of columns is also sortseg or segby, given the segmentation enforced in the `<segmentation>` tag's `cols` attribute. For example, if you have your table segby "day", then it is probably also segby "month". If for some reason you want to enforce segmentation by a particular column (or columns), but not to include it in the list of advised columns the "advise=0" attribute may be included in the `<segmentation>` tag.

`<totals>` *(Table contains tabulation results)*

Indicates whether table contains results from a tabulation.

Attributes
None

Contents
When the `<totals>` element contains the value 1, it indicates the table contains columns that are a result of a tabulation.

`<cols>` *(Metadata for all columns)*

Returns the metadata for all columns in a table.

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>` *(Meta information for one column)* on page 91).

`<th>` *(Meta information for one column)*

The metadata for a single column in a table.
## Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name by which the column is referenced in formulas and query operations. <strong>Note:</strong> This is generally not the same as the column's heading (see &quot;Contents&quot; below). E.g., street.</td>
</tr>
</tbody>
</table>
| type      | The type of data in the column. type must be one of:  
- i (integer)  
- j (big integer) (Available as of version 11.25)  
- f (float)  
- a (alphanumeric) |
| format    | The display format for the data in the column. Format specifications have the same form as is used in the XML Macro Language in the Edit Actions dialog of the 1010data web interface (e.g., type:num;width:15;dec:2) |
| fixed     | Indicates whether this is a fixed column or not. When fixed is 1, this indicates that the column is a fixed column. **Note:** Fixed columns, when they exist in a table, are always visible and are separated from the other columns by a vertical orange line. |
| total     | Indicates whether this is the result of a tabulation or not. When total is 1, this indicates that the values in the column are from the result of a tabulation. |

(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 web interface. Lines in the heading are separated by "
" (the newline character).

### Example

```xml
<th name="date" type="i" format="type:date;width:8;dec:0" fixed="1">Date</th>
<th name="t0" type="f" format="type:num;width:6;dec:2" fixed="0" total="1">Average Dry Bulb Temp (Celsius)</th>
```

### <links> (Top-level wrapper for links)

<links> can contain an arbitrary number of <link> tags, which act a wrapper used by the addtab transaction.

## Attributes

None
Contents

<links> can contain an arbitrary number of <link> tags, which act a wrapper used by the addtab transaction to create a persistent link between two tables along a set of columns or to bring some subset of columns from a second table into the base table. Its basic syntax is given as follows.

Example

```xml
<link table2="foreign.table.name" col="col1,col2"
      col2="fcol1,fcol2" [denormalize="1"] />
</links>
```

This syntax specifies that the table referenced by the table2 attribute within the <link> tag should be joined along the columns indicated in the col and col2 attributes. If the denormalize attribute is present, columns from the foreign table are denormalized into the base table, otherwise a prelink between the two tables is generated. In the general case, any valid <link> operation that could be applied in the 1010data Macro Language can be included in the <links> block.

<link> (Link header)

Returns the link header of a table. The link header is used to indicate which columns have come from a foreign table after it has been linked to another 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 screenshot below, the Product Master table was linked to the Sales Item Detail table. The headings of the columns from the Product Master table therefore begin with "Link Header".

<p>| Sales Item Detail |
|-------------------|----------------|</p>
<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Account</th>
<th>Store</th>
<th>Date</th>
<th>Item SKU</th>
<th>Units</th>
<th>Sales</th>
<th>Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>531</td>
<td>957</td>
<td>1</td>
<td>05/15/12</td>
<td>366</td>
<td>-1</td>
<td>-5</td>
<td>-1.84</td>
<td>LA MICRO TERRY CLOG LARGE</td>
</tr>
<tr>
<td>532</td>
<td>478</td>
<td>1</td>
<td>05/15/12</td>
<td>96A</td>
<td>1</td>
<td>0.5</td>
<td>0.25</td>
<td>FRUIT SNACK MIXED BERRIES</td>
</tr>
<tr>
<td>533</td>
<td>478</td>
<td>1</td>
<td>05/15/12</td>
<td>387</td>
<td>1</td>
<td>1.1</td>
<td>0.56</td>
<td>PEPSI 20 OZ</td>
</tr>
<tr>
<td>534</td>
<td>738</td>
<td>1</td>
<td>05/16/12</td>
<td>A96</td>
<td>2</td>
<td>6</td>
<td>2.9</td>
<td>DUCT TAPE</td>
</tr>
<tr>
<td>535</td>
<td>738</td>
<td>1</td>
<td>05/16/12</td>
<td>65B</td>
<td>1</td>
<td>2.25</td>
<td>1.35</td>
<td>RED DEVIL SILICONE SEBALAN</td>
</tr>
<tr>
<td>536</td>
<td>738</td>
<td>1</td>
<td>05/15/12</td>
<td>CB7</td>
<td>1</td>
<td>1.65</td>
<td>1.1</td>
<td>DGV DECOR TOWEL 3 ROLL</td>
</tr>
<tr>
<td>537</td>
<td>709</td>
<td>2</td>
<td>05/15/12</td>
<td>96A</td>
<td>1</td>
<td>1.1</td>
<td>1</td>
<td>MT DEN 2 LITER</td>
</tr>
<tr>
<td>538</td>
<td>709</td>
<td>2</td>
<td>05/15/12</td>
<td>969</td>
<td>1</td>
<td>1.1</td>
<td>1</td>
<td>PEPSI 2 LITER</td>
</tr>
<tr>
<td>539</td>
<td>748</td>
<td>2</td>
<td>05/17/12</td>
<td>A44</td>
<td>3</td>
<td>1.02</td>
<td>0.39</td>
<td>MEMO BOOK 3XS</td>
</tr>
<tr>
<td>540</td>
<td>748</td>
<td>2</td>
<td>05/17/12</td>
<td>366</td>
<td>1</td>
<td>5</td>
<td>1.8</td>
<td>LA MICRO TERRY CLOG LARGE</td>
</tr>
<tr>
<td>541</td>
<td>523</td>
<td>3</td>
<td>05/15/12</td>
<td>CB7</td>
<td>1</td>
<td>1.65</td>
<td>1.1</td>
<td>DGV DECOR TOWEL 3 ROLL</td>
</tr>
<tr>
<td>542</td>
<td>523</td>
<td>3</td>
<td>05/15/12</td>
<td>A96</td>
<td>2</td>
<td>6</td>
<td>2.9</td>
<td>DUCT TAPE</td>
</tr>
</tbody>
</table>

The table is linked.
### `<data>` (Table data)

Returns the data in a table.

**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>` *(Data for one row)* on page 94).

### `<tr>` (Data for one row)

The data for a single row in a table.

**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>` *(Metadata for all columns)* on page 91 and `<td>` *(One cell of data in a table)* on page 94.)

### `<td>` (One cell of data in a table)

Returns the value of a single cell (column/row) of the table.

**Attributes**

None

**Contents**

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

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

or

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

or

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

### `<user>` (User entry)

The username or group name. The name provided must be a user or group that already exists in 1010data.

**Attributes**

None.

**Contents**

The username or group name. The name provided must be a user or group that already exists in 1010data.

### `<users>` (Top-level wrapper for users tree)

The `users` tree may have one attribute (`type`) and, depending on the value of that attribute, may contain multiple instances of the `<user>` element.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
</table>
| type      | The type of access specification *(optional)*  
            | `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.  
            | If omitted, the default is `private`. |

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.

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.
Examples

Python 2.7 example script

A basic script depicting how to create an API session using the `login` transaction and then use the information returned by the API to send subsequent transactions.

Basic API script for submitting transactions

```python
#!/usr/bin/python
import os
# import urllib2 library for making url requests. Note this library must be installed before use.
import urllib2
# import etree library for parsing xml responses. Note this library must be installed before use.
from lxml import etree
#NOTE: Written for Python 2.7

"""Creates list of transaction values which will be used as keys in
dictionaries
This design stipulates that the transaction must be part of this list in order to have
an output file location created when the script is run.
"""
transactions = [
    "login",
    "logout",
    "clear",
    "session",
    "listdir",
    "getdir",
    "putdir",
    "mkdir",
    "gettab",
    "tabinfo",
    "puttab",
    "query",
    "getdata",
    "querydata",
    "savefile",
    "savetable",
    "upload",
    "merge",
    "droptable",
    "dropdir",
    "drop",
    "move",
    "order"]

startinputloc = "$inputFiles/"
endinputloc = "$In.xml"
startoutputloc = "$outputFiles/"
endoutputloc = "$Out.xml"
inputfiledict = {}
outputfiledict = {}

#populate dictionaries with input and output file paths. Note: input files must be created before used
```
for transaction in transactions:
    inputfiledict[transaction] = startinputloc + transaction + endinputloc
outputfiledict[transaction] = startoutputloc + transaction + endoutputloc

def post(url=None, body=None):
    return urllib2.urlopen(urllib2.Request(url, body, headers={'Content-Type': 'text/xml'})).read().decode('utf-8')

# create a 1010data session, capture the encrypted password and SID, then return a session URL that can be used for subsequent transactions
def createsession(username, password):
    url = "https://www2.1010data.com/cgi-bin/prod-stable/gw.k?protocol=xml-rpc&apiversion=3&uid=" + username
    response = post(url + "&pswd=" + password + "&api=login&kill=possess")
    tree = etree.fromstring(response)
    session = {}  
    for child in tree:
        session[child.tag] = child.text
    sessionurl = url + "&pswd=" + session['pswd'] + "&sid=" + session['sid'] + "&api="
    return sessionurl

# Generalized function for calling any transaction (except login, because it's used in createsession function)
def calltransaction(transName, sessionurl, inputfile=None):
    # some transactions don't need/want input files, so ignore this parameter if no input file is passed to function call
    if inputfile is None:
        transactionResponse = post(sessionurl + transName)
    else:
        input = open(inputfiledict[transName], "r").read()
        transactionResponse = post(sessionurl + transName, input)
    return transactionResponse

""
In __main__, a session URL is created using the createsession() function. Valid credentials must be passed to this function for the session to be created. The calltransaction function is called twice, once to apply a query to a table using the query transaction, then again to retrieve the results of the query using the getdata transaction. Results of the second call are written to an output file.
""
if __name__ == '__main__':
    sessionurl = createsession()  # enter valid user credentials as arguments (strings)
    calltransaction("query", sessionurl, inputfiledict['query'])
    response = calltransaction("getdata", sessionurl, inputfiledict['getdata'])
    fp = open(outputfiledict['getdata'], 'w')  # note the use of the transaction name as the key to the output file dictionary.
    fp.write(response)
    fp.close()
    calltransaction("logout", sessionurl)

Note: This script relies on the urllib2 and lxml libraries to handle URL requests and XML parsing respectively. You should ensure these libraries are installed with your Python 2.7 environment for this script to run.