insightsoftware

Integration Guide

Longview

Version 26

Document Information

Notices

Copyright

Longview is a brand name of the insightsoftware.com Group. insightsoftware.com is a registered trademark of insightsoftware.com Limited. Longview is a registered trademark of insightsoftware.com International Unlimited.

Other product and company names mentioned herein may be the trademarks of their respective owners. The insightsoftware.com Group is the owner or licensee of all intellectual property rights in this document, which are protected by copyright laws around the world. All such rights are reserved.

The information contained in this document represents the current view of insightsoftware.com on the issues discussed as of the date of publication. This document is for informational purposes only. insightsoftware.com makes no representation, guarantee or warranty, expressed or implied, that the content of this document is accurate, complete or up to date.

Disclaimer

This guide is designed to help you to use the Longview applications effectively and efficiently. All data shown in graphics are provided as examples only. The example companies and calculations herein are fictitious. No association with any real company or organization is intended or should be inferred.

Contents

Document Information **2**

Notices 2

Contents **3**

Introduction to Longview **6**

About this guide 6

Warnings and notes 6

Procedures 6

Contacting Longview 7

Who Should Read This Document? 7

Longview Application Framework **8**

Consuming Other Data Sources 8

Specifying parameters for the Get DATA command 11

Format for parameters in the Get DATA command 11

Using filter in the parameters of the Get DATA command 12

Using select in the parameters of the Get DATA command 12

Supporting pagination with the Get DATA command 12

Error handling with the Get DATA command 13

Providing REST APIs 15

Create Session 21

Make a GET Request 21

Make a POST Request 21

Terminate a Session 22

REST.lvpro 24

Overview of execution 25
Set Status 27
Check Method 28
Check Accept (Use with GET / HEAD requests) 29
Check Content Type (Use with POST / PUT requests) 29
Check Required Query Parameters 29
On Error Handler 30
Write JSON Result 30
Finalize 30
Extracting Longview Data to ODS 31
Writing the calling procedure document 42
Fact Files 43
TimeStamp File 44
Symbol Properties File 44
System Properties File 45
Dimension Files 45
Field names in ODS output files 48
Rules for the ODS extraction process 49
Web Services 50
Common Web Service Elements 51
Symbol Web Services 53
Data Web Services 62
REST APIs 77

Responses 77

/solutions 78

/solutions/attributevalue 78

/solutions/attributevalue/{class} 79

/solutions/attributevalue/{class}/{name}{query} 80

GET /solutions/data 82

PUT solutions/data 87

GET /solutions/hierarchies 93

GET /solutions/hierarchies/{dimension} 94

GET /solutions/hierarchies/{dimension}/{root} 95

PUT /solutions/hierarchies/{dimension}/{root}{query} 96

Introduction to Longview

Longview provides corporate performance management (CPM) software that leading companies use to drive performance with speed, visibility, and financial integrity. Since 1994, many of the world's most respected companies have been using our technology platform to create a single repository of financial truth from which statutory consolidation, management reporting, financial planning, modeling, analysis, budgeting, forecasting, and strategic tax can be performed quickly and accurately, enterprise wide.

Longview enables enterprise clients to collect, store, analyze, and report on data in real-time by automating, centralizing, and standardizing any one or combination of the following key financial processes: Planning, Budgeting, Forecasting, Consolidation, Financial Close Management, Profitability Analytics, Statutory, XBRL Financial Reporting and Tax Provisioning. With Longview customers can reduce overreliance on spreadsheets, improve transparency, and regain control of these key finance functions.

Longview Tax calculates your company's global tax charge, effective tax rate, and deferred taxes for tax provisioning purposes. Since Longview Tax uses the same technological platform as your corporate performance management solution, the tax reporting process is directly integrated into the corporate close process. As one solution, consolidated pre-tax income can be reported by legal entity to accurately calculate consolidated income tax charges and deferred taxes.

For more information on purchasing Longview Tax, contact your Longview Account Manager. Web services are a standardized way of integrating applications over the Internet or Internet protocol-based networks. Web services rely on certain software standards including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL) and Universal Description, Discovery & Integration (UDDI).

About this guide

This guide includes basic information on how to install your Longview system. The following sections indicate conventions that are used in this guide.

Warnings and notes

This guide uses the following conventions for warnings and notes:

Caution: Warnings provide cautionary information on the possible effect of certain actions, including the unintentional deletion of data. Be sure to read and understand all warnings before performing a related procedure.

Note: Notes provide additional information to help you understand your Longview system better. They also provide important information on exceptions to general guidelines.

Procedures

There may be several ways to perform a procedure in your Longview system.

- You may be able to choose a menu command. For example, to open a file, you can choose **Open** from the File menu. In this documentation, we use: Choose **File** > **Open**.

- You may be able to use a keyboard equivalent. For example, to exit you can press the **Alt** key and then the letter **F** to open the File menu, and then press **X** for Exit.
- You may be able to click a button or icon. If a menu command has an equivalent button or icon, an illustration of the button or icon may appear in the margin.

In this documentation, we may not describe all methods to carry out a task. Use whichever method you prefer. Depending on the task you are performing, certain methods may not be available.

Contacting Longview

Questions? We are ready to help. For contact information for Longview, visit our web site at insightsoftware.com/Longview/.

Who Should Read This Document?

This documentation assumes you have a working knowledge of Microsoft Windows and the Internet, and that you understand basic terminology such as buttons, drop-down lists, defaults, and so on.

It also assumes that you are familiar with basic accounting terminology and concepts.

Longview Application Framework

Longview offers integration with other software products through Longview Application Framework or Longview Web Services.

Longview Application Framework runs as a component of the Longview Server platform. Longview Application Framework can be run directly from a command prompt in batch mode and can be scheduled via third-party software. Longview Application Framework can also be triggered by event rules.

Longview Application Framework can provide integration with third-party applications using text files. Text files remain a popular and simple method of transferring data and metadata to and from Longview and third-party applications. The general method for accomplishing integration using text files is to use Longview Application Framework Procedure commands that execute against a text file, or to use a Longview ImportSpec or Longview ExportSpec to import and export data from and to a text file. For more information, see the Longview Developer Guide.

Longview Application Framework can also provide integration with third-party applications through the following more advanced methods:

- Consuming Data Sources from third-party applications:
  - via ODBC (Open Database Connectivity) queries
  - By calling REST APIs
- Providing REST APIs that are consumable by third-party applications
- Extracting Data to ODS (Operational Data Store)

This section describes Longview Application Framework integration using these advanced methods:

- Consuming Other Data Sources
- Providing REST APIs
- Extracting Longview Data to ODS

Consuming Other Data Sources

Consuming other data sources via ODBC (Open Database Connectivity) queries

For importing data from a third-party application to a Longview DataArea or DataTable, you may use an ODBC connection. This is accomplished by using a Longview Data Import and specifying ODBC as the DataSource, along with an appropriate connection string and SQL statement to execute.

The following table lists the available Application Framework ImportSpec functions to support ODBC imports in Longview.

For more information, see ImportSpec and ExportSpec Functions in the Longview Developer Guide.

Function	Usage
DataSource ODBC, “Connection String”	Use this function with an import specification to define the source of the data to be imported.
SQLBatchSize	Specify the number of records to retrieve with each fetch against the ODBC data source
SQLStatement	Specify the SQL statement to pass along to the ODBC data source

Example of using ODBC in an ImportSpec document:

```
DataSource ODBC, "Driver={SQL Server Native Client
10.0};Server=ca127MyServer;Database=MYDB;Trusted_Connection=Yes;"

SQLBatchSize 10

SQLStatement “SELECT [Currency], [Rate] FROM FXRates”
```

Consuming other data sources by calling REST APIs

REST APIs enable organizations to communicate with each other and with clients, without deep knowledge of each other’s IT systems. The REST API provider defines a format for requests for each of its services and the response that the service generates.

Longview Application Framework can call REST APIs provided by third-party applications using various commands in a Procedure document. These commands will execute the third-party REST APIs using the provider’s defined format and retrieve the responses from the Provider in a defined format. A popular return format for REST APIs is JSON format, which is a lightweight, open-standard, object-oriented format that is easy for humans to read. When an API returns JSON format, Longview Application Framework can translate the response directly into an object. However, Longview Application Framework supports results in other formats as well. Options for handling responses include:

- using a STRING variable as the result target and processing that string in memory or exporting the result, using EXPORTVARIABLE, to a file.
- using an OBJECT variable as the result target and processing that object in memory or writing to a file via EXPORTVARIABLE.
- using a DOCUMENT as the result target and saving the resulting document to a file using SAVE. This method is useful, for example, with csv responses.

The following table lists the available Application Framework Procedure commands to support consuming REST APIs in Longview. For more information, see the Longview Developer Guide.

Command	Usage
Rest DeleteRequestHeader	Delete a header and its value that was previously set using REST SETREQUESTHEADER
Rest ExecuteDelete	Delete a resource identified by a URI

Command	Usage
Rest ExecuteGet	Retrieve information returned from a REST http GET method
Rest ExecutePost	Post to a resource using a restful http POST method
Rest ExecutePut	Put or update data to a resource using a restful http PUT method
Rest GetResponseHeader	Retrieve a specified response header for the REST EXECUTEGET, REST EXECUTEPOST, or REST EXECUTEPUT command that was previously invoked
Rest GetResponseStatus	Retrieve the http response status for the REST EXECUTEGET, REST EXECUTEPOST, or REST EXECUTEPUT command that was previously invoked
Rest SetRequestHeader	Set a header for the REST call about to be executed

Example of Calling REST APIs in a Procedure document:

```
// Example with Target as a variable
CREATE VARIABLE TestResult1 AS String
REST EXECUTEGET "http://api.fixer.io/latest" TO TestResult1
EXPORTVARIABLE TestResult1 TestResult1.json
// Example with Target as a document
CREATE DOCUMENT "TestResult2.json"
REST EXECUTEGET "http://api.fixer.io/latest" TO "TestResult2.json"
SAVE "TestResult2.json"
```

Consuming Other Data Sources Using Open Business Data Fabric

The Open Business Data Fabric allows you to retrieve data from Angles Business Views, which provides:

- shared data across data sources and applications
- pre-built business views

To utilize the Open Business Data Fabric, you require the following:

- access to the platform
- a designer license for Angles
- a service to service account set up in platform
- an access token created by a user targeting Angles PDS

To configure Longview to connect to Open Data Business Fabric:

1. Launch Application Administrator.
2. In server explorer, expand **Attributes** and click on **SYSTEM**.
3. Locate the SAccessToken attribute, right-click it, and select **Set Value**.
4. In the Enter New Value field, enter the access token generated for targeting Angles PDS.
5. Click **Assign to Value**.
6. Click **OK**.
7. Open the Actions menu and select **Maintenance** to toggle maintenance mode off.

Retrieving data using Open Business Data Fabric

You can retrieve data using the Get DATA command in application framework. This command can only be executed using the service-to-service account. The command will generate a file with the requested data in oData format. Data retrieved in oData format can be imported into a data area using the Run IMPORT command.

To use the Get DATA command, you will need the following:

- The id of the business view you will use.
- Any filter or select options you will use to restrict the data retrieved, for example to restrict the data to a single period, or limit the fields returned by the query.

The syntax of the Get DATA command is as follows:

```
Get DATA businessViewId ["parameters"] TO fileName
```

Specifying parameters for the Get DATA command

Parameters is optional in the Get DATA command but is useful for filtering the data returned in the request and limiting the properties returned for each item in the oData collection. Parameters are used to include oData system query options such as $filter and $select in the request. There are many system query options available with oData requests, but only the most commonly used options for integration with Longview will be described.

Format for parameters in the Get DATA command

Parameters take the format “system query option”=”expression”. When specifying a system query option, exclude the dollar ‘$’ character used with oData queries.

Multiple parameters are separated with the ampersand “&” character.

For example, “filter=expression&select=expression”.

The expression you use will be dependent on the business view you are querying.

Using filter in the parameters of the Get DATA command

The filter system query option can be used with a business view only if one or more Filter Values Entities have been defined in the business view. The expression for a filter is written as “field name” “operator” “value”. If the value contains spaces, it must be enclosed in single quotes.

For example, filter=accounting_period_name eq 'Feb 2019'.

Valid operators include:

- Equal (eq)
- Not equal (ne)
- Greater than (gt)
- Greater than or equal (ge)
- Less than (lt)
- Less than or equal (le)
- Has flags (has)

A filter can also include multiple conditions using logical operators “and”, “or”, and “not”.

Using select in the parameters of the Get DATA command

The select query option can be used to limit the properties returned for each item in the query result set. The expression for a select is written as a command separated list of field names. The field names must match the names defined in the business view.

For example, select=account,base_currency,ytd_consolidated_amount.

Supporting pagination with the Get DATA command

In cases where the amount of data being returned by a Get DATA request is extremely large, you can use pagination to return the results in multiple discrete requests. Open Business Data Fabric only supports pagination on the client side, so it needs to be configured in your process.

To use pagination with the Get DATA command:

1. The business view must have a unique key defined.
2. Include the top and skip system query options in the parameters.
3. Iterate through Get DATA requests until the number of records returned not equal to the value defined by top.

a. You can check the number of records by:

i. Importing the file into an object.

ii. Checking the count of the value property.

4. Import each file generated by the Get DATA command into the target data area.

Error handling with the Get DATA command

In cases where the business view ID or parameters used with the Get DATA command are invalid, the result returned will still be JSON but will not be in oData format.

You can check whether an error occurred by importing the file to object and checking for the existence of the error property. If the error property exists, the error.message property contains the message returned by the invalid request.

Example: invalid business id:

{"error":{"code":null,"message":"Cannot find EntitySet, Singleton, ActionImport or FunctionImport with name 's_66632f48a751af7e6c8c6c3z'."}}

Example: invalid field in select

{"error":{"code":null,"message":"The property 'acc0unt', used in a query expression, is not defined in type '64ccbd4795855d760edb7fb7'."}}

Caveats

1. Since multiple files are created, this technique works best when used with the ADD option for duplicate record handling in the import specification document.

2. If the import specification uses the USEFIRST or USELAST option for duplicate record handling, the first or last record in the last file processed will be imported.

3. If the import specification uses the DISALLOW option for duplicate record handling, an error will only occur if one of the files processed contains duplicate records.

Example: Retrieve oData 500,000 records at a time and import

```
Create VARIABLE top AS NUM
```

```
Create VARIABLE skip AS NUM
```

```
Create VARIABLE loop AS NUM
Create VARIABLE parameters AS STRING
Create VARIABLE oData AS OBJECT
Create DATAAREA dataAreaName USING "dataAreaSpec.lvdsp"
Set VARIABLE loop = 1
Set VARIABLE top = 500000
While $loop$
Set VARIABLE parameters = "top=$top$" + "&skip=$skip$"
Get DATA businessViewID "$parameters$" TO "oData_
$skip$.json"
Run IMPORT importSpec.lvimp TO dataAreaName
ImportObject oData "oData_$skip$.json"
Set VARIABLE loop = Count(oData.value) == $top$
If $loop$
Set VARIABLE skip = $skip$ + $top$
END If
END While
```

```
Create VARIABLE top AS NUM
Create VARIABLE skip AS NUM
```

```
Create VARIABLE loop AS NUM
Create VARIABLE parameters AS STRING
Create VARIABLE oData AS OBJECT
Create DATAAREA dataAreaName USING "dataAreaSpec.lvdsp"
Set VARIABLE loop = 1
Set VARIABLE top = 500000
While $loop$
Set VARIABLE parameters = "top=$top$" + "&skip=$skip$"
Get DATA businessViewID "$parameters$" TO "oData_$skip$.json"
Run IMPORT importSpec.lvimp TO dataAreaName
ImportObject oData "oData_$skip$.json"
Set VARIABLE loop = Count(oData.value) == $top$
If $loop$
Set VARIABLE skip = $skip$ + $top$
END If
END While
```

Related topics:

• To learn more about the insightsoftware Platform, see the Platform Experience documentation.

• To learn more about the Get DATA command, see Get Data.

• To learn more about import specs, see Creating ImportSpecs and Run (for ImportSpecs).

Providing REST APIs

In addition to calling REST APIs, Longview Application Framework also serves as the framework to provide REST APIs. Rather than providing a fixed set of APIs, Longview provides a framework for Solution Developers to design their own Solution-level APIs. This framework uses Longview Application Framework to expose Apps, that conform to a prescribed format, as REST APIs. In this manner, Longview makes it possible to invoke any Application Framework procedure from a third-party.

Examples of possibilities include:

- retrieving data from Data Tables
- synchronizing environments using Symbol Maintenance and Attribute Maintenance commands
- creating and posting a Tax Provision Calculated JE from a Longview Tax system to a Longview Close system

This section describes how to build an Application Framework Procedure that can be used to provide a REST API.

The general steps involved in Providing a REST API include:

1. A Session API is first called from the third-party application (the “REST consumer”), as an http(s) POST request to get a session.
2. Once the session is created, a Solution API request is initiated from the third-party application as an http(s) request.
3. The request travels via http(s) to Longview instance’s web server.
4. Longview launches an Application Framework session to handle the request.
    - The Application Framework procedure in the Solution API should be coded to handle the request and generate a response in a defined format.
5. Longview delivers the response back to the third-party caller application.
    - The calling application processes the results.
6. A Session API is called from the third-party application as an http DELETE request to destroy the session.

URL Syntax

REST APIs are delivered via the existing Longview Web Bridge component. Therefore, the URL to access REST APIs will begin with the standard Web Bridge URL (as defined in the server configuration parameter WEB_SERVER_BRIDGE). Appended to this URL will be the Longview system identifier, followed by the keyword “API”, then “session” (for session APIs) or “app” (for Solution APIs). Solution APIs may have additional path elements and query string parameters.

The general http(s) request format is as follows:

```
method webServerBridge/LVIdentifier/API/type[/pathToAPI][?queryString]
```

where

- **method** is one of the following: POST, PUT, GET, DELETE
- **webServerBridge** is the standard URL for the instance’s web server bridge
- **LVIdentifier** is the Longview system identifier
- **type** is one of the following: session, app
- **pathToAPI** is an optional parameter specifying the path and Longview Application Framework procedure to run. The default procedure is `rest.lvpro`if this parameter is not specified. If this parameter is specified, it specifies the path, or a specific filename.
- **queryString** is an optional parameter and specifies any additional variables to be appended to the URL. The queryString must be in the format`name=value`.

Examples of URL Syntax for Calling a Solution API:

```
GET http://LVWebserver/cgi-bin/DEMO101/lvweb.cgi/DEMO101/API/app

GET http://LVWebserver/cgi-bin/
DEMO101/lvweb.cgi/DEMO101/API/app/hello.lvpro?Var1=Value1&Var2=Value2

GET http://LVWebserver/cgi-bin/
DEMO101/lvweb.cgi/DEMO101/API/app/folder1/hello.lvpro?Var1=Value1
```

API Status Codes

The following status codes may be returned by Session or Solution API calls:

Code	Description
200 OK (ok)	OK
400	Bad request (if problem is detected with input parameters)
401	Unauthorized (authentications fails, or API request made without a valid web session ID)
404	Not found (API not found)
200 OK (error)	Server error

Session APIs

Before executing any Solution APIs, the caller must first authenticate and obtain an authentication token to be provided in subsequent calls. Callers should also terminate the session when they have completed all requests.

Creating a New Session - POST "/API/session"

You can create a session using either Longview authentication or Single Sign-On (SSO) authentication.

Note: LongviewClientID must be set to the same value as the Longview Identifier if the Longview Identify Policy (LV_IDENTIFIER_POLICY) is set to ENFORCE.

To create a session using Longview Authentication:

The message body is JSON containing the following parameters (all strings):

- LongviewUserName: name of the user
- LongviewPassword: password for the user
- LongviewGroup (optional, only required if the user belongs to more than one group)
- LongviewClientID: the Longview identifier

To create a session using ISW Platform authentication:

The first step is to obtain an access_token from the ISW platform using the interface it provides. Next, pass the access token to create a new Longview session.

The message body is JSON containing the following parameters (all strings):

- LongviewUserName: name of the user
- AADToken: the access token returned by the SSO provider
- LongviewGroup (optional, only required if the user belongs to more than one group)
- LongviewClientID: the Longview identifier

To create a session using ISW Platform Organisation key:

If your system is on ISW platform and uses ISW platform Organization API Keys for authentication, then use the following to create a Longview session. The user must be a service account with Connect to Application Framework authorization.

- ApiKey: the organization API Key that was generated in the ISW platform.
- LongviewGroup: (optional, only required if the user belongs to more than one group).

To create a session using Longview Single Sign-On:

The first step is to obtain an access_token from your SSO provider using the interface it provides. Next, pass the access token to create a new Longview session.

The header includes the authorization via Bearer Token.

The message body is JSON containing the following parameters (all strings):

- LongviewGroup (optional, only required if the user belongs to more than one group)
- LongviewClientID: the Longview identifier
- LongviewAuthMode: “3rdAuth”

To create a session using OPENID/oAuth2 without Longview Single Sign-On:

The first step is to obtain an access_token from your SSO provider using the interface it provides. Next, pass the access token to create a new Longview session.

The message body is JSON containing the following parameters (all strings):

- Id_token: The access token provided by OPENID/oAuth2
- LongviewGroup (optional, only required if the user belongs to more than one group)
- LongviewClientID: the Longview identifier

Result:

In any case, if authenticated, the response body is also JSON, containing the following properties:

- LongviewWebSID (string) to be used for subsequent requests
- LongviewErrorCode (integer)
- LongviewErrorMessage (string).

Destroying a Session – DELETE "/API/session"

Callers should also terminate the session when they have completed all requests by destroying the session. To destroy a session, you must provide the LongviewWebSID as an http header.

Creating Solution-level APIs

Solution-level APIs may be called using any of the following http methods: GET, PUT, POST, DELETE. This method is passed along as a parameter to the Application Framework procedure so that the Solution Developer can implement accordingly. The path to all Solution APIs begins with [webbridge]/API/app/… and must end with a relative path to an Application Framework procedure published to the Longview Data Servers' applications\APIs folder. The path may point directly to a document, or can specify only the path, in which case rest.lvpro is used.

Note:

- The path may also extend beyond the real folder path, allowing for the use of virtual folders to denote a noun resource upon which the http method (or "verb") is acting. An example of this might be an API to create Longview symbols. The path for such an API could be `/APIs/app/symbols/entities/newEntity`.
- The method would be POST, meaning the caller intends to create a new symbol. The actual physical path would be only `applications\APIs\symbols`. In that path, the rest.lvpro procedure would be executed.
- The remaining portion of the path, `/entities/newEntity`, would be passed along as parameters that could mean "create a symbol in the Entities dimension" and "the name of the new symbol is newEntity" respectively.

All calls to Solutions APIs must provide the LongviewWebSID as an HTTP header.

In the case of a PUT or POST method, the message body supplied can be any format. This content is written to a temporary file and passed to the Application Framework procedure for processing by the Solution API.

All methods can return content, again, in any format. This content is written to a temporary file by Application Framework and assembled into the HTTP response.

Application Framework Variables

When an Application Framework Solution API procedure is executed, the following system variables will be provided:

Variable Name	Description
LVS_APIRequestType	"POST", "PUT", "GET" or "DELETE"
LVS_APIRequestPath	path of the URL request For example, if the request is http://localhost/cgi-bin/LongviewSimple/lvweb.cgi/LongviewSimple/API/app/path1/path2/path3/path4 and rest.lvpro is found in the folder path2, then LVS_APIRequestPath contains path2/path3/path4
LVS_APIInputContentType	content type of message body (only relevant for POST or PUT)
LVS_APIInputFile	name (full path) of temp file containing message body
LVS_APIOutputFile	name (full path) of temp file to which message response should be written
LVS_APIOutputHeaderFile	output headers set by the Application Framework command RESTAPI SETREQUESTHEADER in the REST API procedure
LVS_APIRequestPathParameters	path of the URL request starting from where rest.lvpro is located. This only applies when the default rest.lvpro is used. It does not apply if a specific document has been specified. For example, if the request is http://localhost/cgi-bin/LongviewSimple/lvweb.cgi/LongviewSimple/API/app/path1/path2/path3/path4 and rest.lvpro is found in the folder path2, then LVS_APIRequestPathParameters contains path3/path4
APIRequestParameters	form data submitted with the REST request. This variable can be used when the REST API is called by a third-party application via a form submit.

Additional global variables will be added to the Application Framework session corresponding to any parameters added to the URL of the request. The example below adds two global variables, Param1 and Param2:

http://URL/API/app/pathtoAPI?Param1=Value1&Param2=Value2

Application Framework Commands and Functions

The following table lists the available Application Framework Procedure commands and functions to support providing REST APIs in Longview. For more information, see the Longview Developer Guide.

Command/Function	Usage
RESTAPI SetResponseHeader	Set customer HTTP headers for a response.
GetDocumentSize	Returns the document size in memory. The document must be loaded into memory, and saved if contents have been modified, when this function is called. The GetDocumentSize function is useful when creating Solutions-level REST APIs that may be called by third-party applications. In this case, it can be used to set the content-length header for the REST API.

The following illustrates a sample series of API calls:

Create Session

API call	Description
URL	POST http://localhost/cgi-bin/LongviewSimple/ lvweb.cgi/LongviewSimple/API/session
Message Body	{ “LongviewUserName”: “user1”, “LongviewPassword”: “password”, “LongviewGroup”: “Administrators”, “LongviewClientID”: “LongviewSimple” }
Reply Content-type	JSON
Reply Content	{ “LongviewWebSID”: “--authtoken--”, “LongviewErrorCode”: 0, “LongviewErrorMessage”: “” }

Make a GET Request

API call	Description
HTTP Headers Expected	Content-type LongviewWebSID
URL	GET http://localhost/cgi-bin/LongviewSimple/lvweb.cgi/LongviewSimple/API/app /hello.lvpro?VarTest1=Value1
Reply Content-type	The content type that hello.lvpro assigns using the RESTAPI SetResponseHeader command
Reply Content	The content that hello.lvpro writes to LVS_APIOutputFile

Make a POST Request

API call	Description
HTTP Headers Expected	Content-type LongviewWebSID
URL	POST http://localhost/cgi-bin/LongviewSimple/lvweb.cgi/LongviewSimple/API/app /hello.lvpro?VarTest1=Value1
Content-type	Text/csv
Content	Symbol,symbol,symbol,symbol,symbol,value Symbol,symbol,symbol,symbol,symbol,value etc
Reply Content-type	The content type that hello.lvpro assigns using the RESTAPI SetResponseHeader command
Reply Content	The content that hello.lvpro writes to LVS_APIOutputFile

Terminate a Session

API call	Description
HTTP Headers Expected	Content-type LongviewWebSID
URL	Delete http://localhost/cgi-bin/LongviewSimple/lvweb.cgi/LongviewSimple/API/session

Implementing custom REST APIs

The solutions framework provides utilities to support providing REST APIs with a standard implementation methodology.

API Code Library Variables and Properties

Name	Type	Use
API	OBJECT	Contains API properties
API.AcceptableContentTypes	STRING []	Used to define the acceptable content types for a GET request (CheckAccept) or a PUT / POST request (CheckContentType)
API.AllowedMethods	STRING	Comma separated list of methods Default is GET, HEAD Used to confirm the method on the target URI (CheckMethod) and returned as Allow header when 405 returned
API.BaseURI	STRING	The base URI for REST requests. Defaults to $LVS_HTTPPROTOCOL$://$LVS_WEBSERVER$ $LVS_WEBBRIDGE$/$LVS_IDENTIFIER$/api/app
API.ContentLength	NUM	Used to return the Content-Length response header
API.POSTURI	STRING	Used to store the URI for a resource created using the POST method
API.ResponseContentType	STRING	Used to return the Content-Type response header
API.SaveOutputFileOnFinalize	NUM	Indicates whether the Finalize procedure should save the output document Default is 1
API.Params	OBJECT	Used to hold the parameters defined in the request
API.Params.RequestType	STRING	The method sent in the request Default $LVS_APIRequestType$
API.Params.RequestPath	STRING	The part of the URI after /app Default $LVS_APIRequestPath$

Name	Type	Use
API.Params.Request PathParameters	STRING	The part of the URI that remains after the Rest.lvpro is found. Default $LVS_APIRequestPathParameters$
API.Params. RequestPathParameterList	STRING []	Request path parameters expressed as a list (divided by '/')
API.Params.InputContentType	STRING	The Content-Type in the request header (PUT / POST) Default: $LVS_APIInputContentType$
API.Params.InputFile	STRING	The name of the file containing the body of the request (PUT / POST) Default: $LVS_APIInputFile$
API.Params.OutputHeaderFile	STRING	The name of the file containing the response header. Set via the RESTAPI SETRESPONSEHEADER command Default: $LVS_APIOutputHeaderFile$
API.Params.OutputFile	STRING	The name of the file containing the response body. Default: $LVS_APIOutputFile$
API.Params.Request Parameters	STRING	Form data submitted with a REST request. Default $LVS_APIRequestParameters$ or $APIRequestParameters$
API.Services	OBJECT	Used to determine if a service is provided For each service provided at the location of the URI to your REST API, add a property with the name of the service with type NUM For example, if your base URI is: /api/app/custom And you provide data services at /api/app/custom/data And symbol services at /api/app/custom/symbols Add the following properties to API.Services: API.Services.data API.Services.symbols
API_Input	OBJECT	If the request body Content-Type is applications/json, used to hold the request body in an object for processing
API_Output	OBJECT	If the response body Content-Type is applications/json, used to hold the response body in an object for processing
API_ RequiredQueryParameterList	STRING []	Holds a list of required query parameters for the request handler. Used to check for a bad request due to missing query parameters (CheckRequiredQueryParameters).
API_ResponseHeaders	OBJECT []	Used to hold the list of response headers

Name	Type	Use
API_ResponseHeader	OBJECT	Used to define a response header
API_ResponseHeader.Name	STRING	The name of the response header field
API_ResponseHeader.Value	STRING	The value of the response header field

General Structure

A procedure named Rest.lvpro is provided under applications/APIs/custom. The name of the folder (custom) will represent the part of the URI path after /app.

The Rest procedure contains the logic to determine what service it is providing, with all other logic contained the REST configuration library, provided in Longview Designer. This allows you to update and create additional services provided along the /app/custom path without the need to create additional files directly on the application server.

REST.lvpro

```
////Description: Handles a REST request
//Load supporting code libraries and error handling
LoadKAR "Solutions\CL\API.kar" ""
Run PROCEDURE "API\Init.lvpro"
OnError TERMINATE "$CORE_ErrorFileName$" "API\OnError.lvpro"
//Create response body document for finalize
Create DOCUMENT "$API.Params.OutputFile$"
//Load the REST configuration library that contains the REST handler
procedure and initialize it
LoadKAR "Libraries\REST.kar" ""
Run PROCEDURE "Init.lvpro"
If "$API.Params.RequestPathParameters$" == ""
    Run PROCEDURE "custom\RestHandler.lvpro"
Else
    If PropertyExists("API_Services.$API.Params.RequestPathParameterList
[1]$")
        Run PROCEDURE "custom\$API.Params.RequestPathParameterList
[1]$\RESTHandler.lvpro"
    Else
        Run PROCEDURE "API\SetStatus_404.lvpro"
    END If
END If
//Finalize
Run PROCEDURE "API\Finalize.lvpro"
```

Overview of execution

- Initialize API code library and set up error handling
- Create a document to hold the response body
  - Depending on your process you may need to create your own document or load it if your process creates an output file directly to disk. In that case make sure it is loaded into memory before Finalize is executed.
- Load the REST configuration library and initialize it
  - The initialize at a minimum should:
    - Add properties to the API_Service object to represent the services that are available
- Determine which rest handler to execute
    - If the URI request ends with the folder that the rest procedure resides in
      - Execute the default RESTHandler custom\RESTHandler.lvpro in the REST configuration library
    - If the URI request ends in a path beyond the folder
      - Check if that service is valid by checking if a property exists in API_Services with the name of the next part of the path
      - If so:
        - Execute the RESTHandler.lvpro for that path
      - Otherwise
        - Return 404 Not Found
    - The RESTHandler at a minimum should
      - Set the list of allowed methods for the initial path
      - Set the Acceptable Content Types
      - Run the corresponding method service (ie: GET, HEAD, PUT)
- Finalize

Configuration Library Structure

This is the recommended structure for the configuration library. The Rest procedure outlined above will execute for all URIs starting from the custom folder. Any additional paths in the URI will be captured as rest path parameters.

- Create procedure RESTHandler.lvpro for each path in your URI (level 2)
- Create a procedure named after the method is handles for each path in your URI (level 2)

Example:

Assume your rest path is named custom and provides the following services:

- custom/data (GET, PUT)
- custom/symbols (GET)

On the data server applications\APIs folder there is a folder named custom with the procedure Rest.lvpro as outlined above.

Your configuration library contains the following:

- Init.lvpro
  - Adds the properties data and symbols to the API_Services object

  Example:

```
Create PROPERTY API_Services.data AS NUM
Create PROPERTY API_Services.symbols AS NUM
```

- custom\RESTHandler.lvpro
  - Sets the property API.AllowedMethods to “GET, HEAD, PUT”. This can be adjusted as required.
  - Execute API\CheckMethod.lvpro
  - Sets the valid Acceptable Content Types to */*, application/json and application/*. This can be adjusted as required.
    - If valid
    - Execute the procedure custom\$API.Params.RequestType$.lvpro
- custom\GET.lvpro: Handles a GET request for the URI /custom. By default, it will cycle through the available API_Services properties and provide the links in JSON format.
- custom\data\RESTHandler.lvpro
  - Sets the property API.AllowedMethods to “GET, PUT”
  - Executes API\CheckMethod.lvpro
  - If valid
    - Executes the procedure custom\$API.Params.RequestType$.lvpro
- custom\data\GET.lvpro: Handles a GET request for the URI / custom/data
- custom\data\PUT.lvpro: Handles a PUT request for the URI / custom/data
- custom\symbols\RESTHandler.lvpro
  - Sets the property API.AllowedMethods to “GET”
  - Execute API\CheckMethod.lvpro
  - If valid
    - Execute the procedure custom\$API.Params.RequestType$.lvpro
- custom\symbols\GET.lvpro: Handles a GET request for the URI /custom/symbols

API Code Library Utilities

The API code library contains several procedures that can be used with any rest service and support standard request checks and steps to deliver a proper response to a request.

Init

Procedure Name: API\Init.lvpro

Prerequisites:

- None

Initialize the API code library:

- Creates global variables and initializes the API object
- Writes a message to server audit trail that the REST request is being handled

Set Status

Procedure Name: API\SetStatus_<StatusCode>.lvpro

Where StatusCode is one of 200 / 201 / 400 / 404 / 405 / 406 / 415 / 500

Prerequisites:

- Execute API\Init.lvpro

A procedure for each supported status is provided. Each procedure is named SetStatus<Status Code>.lvpro.

The following status procedure are provided:

- SetStatus_200.lvpro: 200 OK
  - Used to report a successful request
  - Sets the Status response header to “200 OK”
  - Sets the LVStatus response header to “ok”
- SetStatus_201.lvpro: 201 Created
  - Used to report a successful POST request
  - Property API.POSTURI must be set before calling this procedure
  - Sets the Status response header to “201 Created”
  - Sets the Location response header to “$API.POSTURI$”
- SetStatus_400.lvpro: 400 Bad Request
  - Used to report an invalid request
  - Sets the Status response header to “400 Bad Request”
  - Sets the Content-Type response header to “text/plain”

- SetStatus_404.lvpro: 404 Not Found
  - Used to report a URI that points to an invalid resource
  - Sets the Status response header to “404 Bad Request”
  - Sets the Content-Type response header to “text/plain”

- SetStatus_405.lvpro: 405 Method Not Allowed
  - Used to report when the method in the request is not provided by the resource specified in the URI
  - Sets the Status response header to “405 Method Not Allowed”
  - Sets the Allow response header to “$API.AllowedMethods$”
  - Sets the Content-Type response header to “text/plain”
  - Normally called from API\CheckMethod.lvpro, not directly

- SetStatus_406.lvpro: 406 Not Acceptable
  - Used to report when the Accept request header does not contain a media type that is supported by the service
  - Sets the Status response header to “406 Not Acceptable”
  - Sets the Content-Type response header to “application/json”
  - Writes an object in JSON to the response body with the property “ContentTypes”, which lists the content types specified in API.AcceptableContentTypes
  - Normally called from API\CheckAccept.lvpro, not directly

- SetStatus_415.lvpro: 415 Unsupported Media Type
  - Used to report when the Content-Type in the request (PUT / POST) is not supported for the resources specified in the URI
  - Sets the Status response header to “415 Unsupported Media Type”
  - Sets the Content-Type response header to “application/json”
  - Writes an object in JSON to the response body with the property “MediaTypes”, which lists the content types specified in API.AcceptableContentTypes
  - Normally called from API\CheckContentType.lvpro, not directly

- SetStatus_500.lvpro: 200 OK
  - Used to report any other execution error
  - Sets the Status response header to “200 OK”
  - Sets the LVStatus response header to “error”
  - Sets the Content-Type response header to “text/plain”
  - Normally called from API\OnError.lvpro, which writes the contents of the error file to the response body

Check Method

Procedure Name: API\CheckMethod.lvpro

Prerequisites:

- Execute API\Init.lvpro
- Set property API.AllowedMethods (default is GET, HEAD)

Checks if the method in request (API.RequestType) is in the list of allowed methods (API.AllowedMethods). If not:

- Sets variable CORE_IsNoError = 0
- Executes API\SetStatus_405.lvpro

Check Accept (Use with GET / HEAD requests)

Procedure Name: API\CheckAccept.lvpro

Prerequisites:

- Execute API\Init.lvpro
- Set property API.AcceptableContentTypes

Checks if the Accept in request header is in the list of acceptable content types (API. AcceptableContentTypes). If not:

- Sets variable CORE_IsNoError = 0
- Executes API\SetStatus_406.lvpro

Otherwise:

- Sets property API.ResponseContentType to the first acceptable content type found

Check Content Type (Use with POST / PUT requests)

Procedure Name: API\CheckContentType.lvpro

Prerequisites:

- Execute API\Init.lvpro

Checks if the Content-Type in request header is in the list of acceptable content types API.Params.InputContentType). If not:

- Sets variable CORE_IsNoError = 0
- Executes API\SetStatus_415.lvpro

Check Required Query Parameters

Procedure Name: API\CheckRequiredQueryParameters.lvpro

Prerequisites:

- Execute API\Init.lvpro
- Set variable API_RequiredQueryParameterList

Checks if the variables specified in API_RequiredQueryParameterList exist. If not:

- Sets variable CORE_IsNoError = 0
- Executes API\SetStatus_400.lvpro
- Sets the Content-Type response header to text/plain

On Error Handler

Procedure Name: API\OnError.lvpro

Prerequisites:

- Execute API\Init.lvpro
- Execute OnError command to call this procedure when an error occurs
  - OnError TERMINATE "$CORE_ErrorFileName$" "API\OnError.lvpro"

Handles any execution errors that occur:

- Sets variable CORE_IsNoError to 0
- Writes the error file generated to the response body (API.Params.OutputFile)
- Executes API\SetStatus_500.lvpro
- Executes API\Finalize.lvpro

Write JSON Result

Procedure Name: API\WriteResultJSON.lvpro

Prerequisites:

- Execute API\Init.lvpro
- Populate the API_Output object

Writes the API_Output object to the response body and sets the Content-Type response header to application/json.

Finalize

Procedure Name: API\Finalize.lvpro

Prerequisites:

- Execute API\Init.lvpro

Finalizes the REST request by:

- Getting the size of the response body document ($API.Params.OutputFile$), which must be in memory
- Set the Content-Length response header
- If $API. SaveOutputFileOnFinalize$ == 1
  - Saves the output file
- Writes a message to the server audit trail that the REST request was handled

Extracting Longview Data to ODS

When Longview data is required by a third-party application, such as a Business Integration reporting package, it is sometimes necessary for that application to not only require the data for the requested data area but to also be able to retrieve the symbols that relate to that data area. For example, if a third-party application requests data for North American entities, it may need to know how these entities rollup to each other. Therefore, a process must extract symbol information related to the North American entities in addition to the data. The data and the associated metadata are extracted to an Operational Data Store (ODS). An ODS is simply a method to store this extracted information where it is later utilized by a third-party application.

The Longview ODS extraction process deals only with the exporting of data and the associated symbol metadata. It does not deal with how this data is used by any third-party application.

The ODS extraction process generates a series of files that correspond to:

- the data that has been requested (fact tables)
- metadata for the symbols whose data was extracted (dimension tables)

When using ODS there are two basic questions that you must answer:

- What data must be extracted?
  This is usually relatively straight forward and involves defining a data area. This is like other requests for Longview data where it is necessary to specify symbol specifications for each dimension in your system.
- How must the symbols corresponding to this data be represented?

There are several options for how the symbol metadata is formatted in the dimension tables. You must determine the format that works best for your specific use. This depends on the third-party application that utilizes these files after the ODS extraction process. For additional details, see Dimension Files.

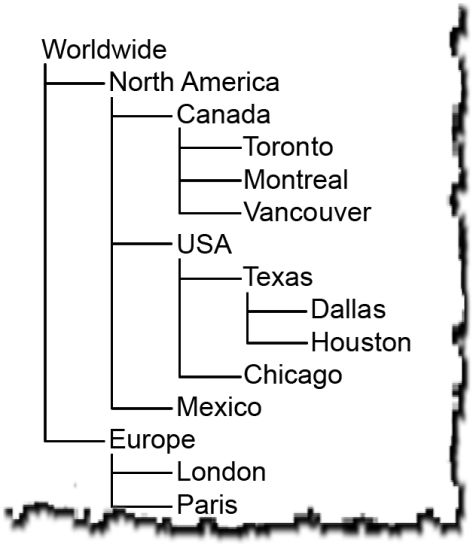
The steps involved in ODS extraction are:

1. Understanding the possible options for creating the dimension files and ensure that one of these options meets the needs of the third-party application.
2. Creating an ODS attribute and setting attribute values for selected symbols.
3. Writing the data area specification document.
4. Writing the export specification document with the required ODS options.

5. Writing the calling procedure document that runs the export specification using the data area specification.

Creating an ODS attribute and setting attribute values for selected symbols

Entities of the same type (e.g. continent, country, city) do not necessarily need to reside at the same level in the Longview hierarchy. Some branches of the hierarchy could have many levels of detail, whereas other branches of the hierarchy could have little detail. In the entity hierarchy below, note that the entity Toronto is at the third level under the entity Worldwide, whereas Dallas is at the fourth level.



Therefore, if a third-party application needs to create a report showing data for just the symbols that are cities, it cannot presume that the cities are all located at the same level in the hierarchy.

If the ODS output is to contain columns named "Continent", "Country", and "City", it is necessary to associate each entity with one of those values. This is accomplished by creating a new symbol attribute, for example "AZEntityType", and then setting that attribute's value for each entity. For example, the AZEntityType attribute values for the following entity symbols would be:

- NorthAmerica = "Continent"
- Canada = "Country"
- USA = "Country"
- Toronto = "City"

To set these attributes, the SET ATTRIBUTE command is used.

Syntax:

```
SET ATTRIBUTE SYMBOL AttrName VALUE SymName AttrValue
```

where:

- **AttrName** is the attribute name associated with a particular ODS extract.

  > **Note:** If you want to use one attribute for a particular ODS extract, you must use the ATTRIBUTEMAP command in the ODS export specification document. If you want to use multiple attributes for a particular ODS extract, you must use the ATTRIBUTEMAPFIELD command, and any combination of the ATTRIBUTEMAPDESC, ATTRIBUTEMAPFACT, ATTRIBUTEMAPID, or ATTRIBUTEMAPNAME commands in the ODS export specification document. For more information, see “Writing the export specification document”

- **SymName** is the symbol whose attribute value is being set.

- **AttrValue** is a string of the following format:

  ```
  FieldName[,NameToUse,DescToUse[,FactTableName[,IndexToUse]]]
  ```

  where:

  - **FieldName** is a string value specifying the target field in the ODS dimension table.

    If a field name is provided and that field name has been subsequently associated with a column number via an ODSMappingTable command, this symbol's information is written to that column in the appropriate dimension table. If no value has been specified for a symbol, the symbol is not written to the dimension table.
  - **NameToUse** is an optional parameter that returns the symbol's name or a hardcoded value. To use the symbol's name, specify a value of NAME. To use a hardcoded value, enclose the string in double quotation marks. The default for this parameter is NAME.
  - **DescToUse** is an optional parameter that specifies whether to use the symbol's description or a hardcoded value. To use the symbol's description, specify a value of DESCRIPTION. To use a hardcoded value, enclose the string in double quotation marks. The default for this parameter is DESCRIPTION. To illustrate when a hardcoded value may be applicable, consider the situation where the third-party application needs to know which date is associated with a time period. Returning the description for the January 2010 actuals time period may return "January 2010 Actuals". It may be more useful if this returned a hardcoded value of "01/01/2010" instead.
  - **FactTableName** is an optional parameter that specifies the name of the fact table to which the data values applicable to this symbol are written. This name is later mapped to the integer value representing a fact table (FactTableNumber). The default value for this parameter is DEFAULT, which is later mapped to fact table 1. Since multiple fact tables can exist for a single dimension only, multiple fact table names can be specified for a single dimension only. For example, if multiple fact tables are to be used to differentiate actuals data from plan data, associate all actuals time periods with a data file name such as "Actuals" and all plan time periods with a data file name such as "Plan". All symbols for dimensions other than time should not have a fact table name associated with them.
  - **IndexToUse** is an optional parameter used to indicate which integer value should be used to uniquely identify this symbol within its dimension. If IndexToUse is not specified, the Longview symbol index is used. To see how this parameter could be used, consider the case when you want to compare January data in your Actuals fact table with the

corresponding January data in your Plan fact table. Since these data values originated from different time period symbols, the symbols indexes for those two time periods are different. These two symbols could be associated to each other by using the same IndexToUse for each of them.

Note: By default, the standard Longview symbol name, symbol description, and symbol index are used. However, the attribute value can be set so that any combination of these can be overridden. For the remainder of this section, the terms UseName, UseDesc, and UseIndex refer to the name, description, and index that are used for a symbol, which could be the standard Longview value or the overridden value.

Using the example above, it is necessary to associate each symbol with its corresponding field in the ODS dimension files. If a data area fixes to a single symbol for a dimension, that symbol's field name must be set to the first field of the dimension table. Since entities vary in this example, each entity to be extracted must be mapped to its appropriate field name.

In this example, the attribute values are set using these commands:

```
Maintenance on

// Prior to running this procedure,

// it is necessary to create the user-defined attribute

// Create Attribute Symbol AZEntityType "Description" "Text" W " "

SET ATTRIBUTE Symbol AZEntityType VALUE Cash "Account"

SET ATTRIBUTE Symbol AZEntityType VALUE A1001YTD
"Period,NAME,01/01/2010,FactActual,1"

SET ATTRIBUTE Symbol AZEntityType VALUE A1002YTD
"Period,NAME,01/01/2010,FactActual,2"

SET ATTRIBUTE Symbol AZEntityType VALUE P1001YTD
"Period,NAME,01/01/2010,FactPlan,1"

SET ATTRIBUTE Symbol AZEntityType VALUE P1002YTD
"Period,NAME,01/01/2010,FactPlan,2"

SET ATTRIBUTE Symbol AZEntityType VALUE NorthAmerica "Continent"

SET ATTRIBUTE Symbol AZEntityType VALUE Canada "Country"

SET ATTRIBUTE Symbol AZEntityType VALUE USA "Country"

SET ATTRIBUTE Symbol AZEntityType VALUE Toronto "City"

// set for all relevant entities

SET ATTRIBUTE Symbol AZEntityType VALUE DIM3SET "Function"

SET ATTRIBUTE Symbol AZEntityType VALUE DIM4SET "Currency"

SET ATTRIBUTE Symbol AZEntityType VALUE DIM5SET "Segment"

SET ATTRIBUTE Symbol AZEntityType VALUE DIM6SET "Element"

SET ATTRIBUTE Symbol AZEntityType VALUE DIM7SET "Control"

Maintenance off
```

Creating ODS documents

ODS documents include data area specification documents:

1. Data area specification document — Use this document type to define the area to extract.
2. Export specification document — Use this document type to specify the target and format for the extraction.
3. Calling procedure document — Use this document type to execute the extraction.

Writing the data area specification document

A data area specification document is required to define the area that is to be extracted. A sample data area specification document is:

```
ACCOUNTS    Cash
TIMEPER     A1001YTD, A1002YTD, P1001YTD, P1002YTD
ENTITIES     NorthAmerica#99
FUNCTIONS     Dim3Set
PRODUCTS     Dim4Set
VERSIONS     Dim5Set
CURRENCY    Dim6Set
CONTROLS     Dim7Set
```

Writing the export specification document

An example export specification document includes the following lines:

```
// Indicate that:
// - this is an ODS export spec
// - all output files are prefixed with "TestODS"
DATATARGET ODS, "TestODS", EXTERNAL
// Specify an optional log file, error file and/or debug file
LOGFILE "Export.log", EXTERNAL
ERRORFILE "Export.err", EXTERNAL
ODSDEBUG "ODSDebug.out", EXTERNAL
// (If you are using one symbol attribute for the ODS extract)
// Specify the symbol attribute that is used to determine the structure
// of the dimension tables
ATTRIBUTEMAP AZEntityType
// Or
```

```
// (If you are using multiple symbol attributes for the ODS extract)
// Specify the symbol attributes that are used to determine the structure
// of the dimension tables
ATTRIBUTEMAPDESC AZEntityType1
ATTRIBUTEMAPFACT AZEntityType2
ATTRIBUTEMAPFIELD AZEntityType3
ATTRIBUTEMAPID AZEntityType4
ATTRIBUTEMAPNAME AZEntityType5
// Are values to be reversed?
REVERSESIGN ON
// Should a time stamp be created?
ODSPUBLISHINGTIME ON
// Should parent data be extracted for a dimension?
ODSINCLUDEPARENTDATA ON, ENTITIES
// Should symbol properties be included in metadata?
ODSSYMPROPERTIES AZSomeOtherAttr1 ENTITIES
ODSSYMPROPERTIES AZSomeOtherAttr2 ENTITIES
ODSSYMPROPERTIES AZSomeOtherAttr3
ODSSYMPROPERTIES DEBITCREDIT ACCOUNTS
// Should system attribute properties be included in metadata?
ODSSYSPROPERTIES SGPCurrentYear
// Specify the format of the dimension tables
//ODSSTRUCTURE RECURSIVEHIERARCHYPF, ENTITIES
//ODSSTRUCTURE RECURSIVEHIERARCHYLF, ENTITIES
//ODSSTRUCTURE STAR
ODSSTRUCTURE SNOWFLAKE, ENTITIES
// If data is to be written directly to database tables,
// Enter the connection info here
ODSDBOUTPUT ODSDataSourceABC, ODSUser, Passwordxxxx
// A symbol with an ATTRIBUTEMAP attribute value of FieldName should appear
// in column fieldNumber of the dimension tables using the command:
// ODSMappingTable ITEM, FieldName, fieldNumber
ODSMAPPINGTABLE ITEM, Continent, 1
ODSMAPPINGTABLE ITEM, Country, 2
ODSMAPPINGTABLE ITEM, City, 3
```

```
ODSMAPPINGTABLE ITEM, Account, 1
ODSMAPPINGTABLE ITEM, Period, 1
ODSMAPPINGTABLE ITEM, Function, 1
ODSMAPPINGTABLE ITEM, Currency, 1
ODSMAPPINGTABLE ITEM, Segment, 1
ODSMAPPINGTABLE ITEM, Element, 1
ODSMAPPINGTABLE ITEM, Control, 1
// A symbol that has been associated with FactTableName should appear
// in fact table number FactTableNum:
// ODSMappingTable ITEM, FactTableName, FactTableNum
ODSMAPPINGTABLE ITEM, FactActual, 1
ODSMAPPINGTABLE ITEM, FactPlan, 2
```

An ODS export specification document must contain the following commands:

Command	Description
DATATARGET ODS	The format for this command is: `DATATARGET ODS, FilePrefix, EXTERNAL` This command designates that this is an ODS export specification and that all output files are prefixed by the value as specified in the FilePrefix parameter. The FilePrefix parameter can also be used to specify a relative path to the working directory. For example, a FilePrefix value of "test\output\testods" places the output files in <current working directory>\test\output, with each file prefixed by "testods".
ODSMAPPINGTABLE	The format for this command is: `ODSMAPPINGTABLE` This command associates a field name with a column index in a dimension table, or a fact table name to a fact table number.

An ODS export specification document must contain at least one of the following commands:

Command	Description
ATTRIBUTEMAP	The format for this command is: `ATTRIBUTEMAP AttributeName` This command specifies which symbol attribute is to be used to format the dimension tables. This command overrides the ATTRIBUTEMAPFIELD command and the ODSAUTOLEVEL command.
ATTRIBUTEMAPFIELD	The format for this command is: `ATTRIBUTEMAPFIELD AttributeName` Use this command to specify the symbol attribute for the FieldName parameter of the ODS extract. This command overrides the ODSAUTOLEVEL command.
ODSAUTOLEVEL	The format for this command is: `ODSAUTOLEVEL ON|OFF` This command automatically generates the level information. The default specification for ODSAUTOLEVEL is OFF. **Note:** When using automap, the ODSMappingTable still needs to be included to map the levels to columns in the dimension tables. In this case, each field name is DimensionName_L00, DimensionName_L01, ..., DimensionName_Lxx, where xx is the level number.

Note: If the ODS export specification document contains either of the ATTRIBUTEMAP or ATTRIBUTEMAPFIELD commands and the ODSAUTOLEVEL command, and ODSAUTOLEVEL is ON, the field name stored in the attribute value overrides the auto level field name. For more information, see “Field names in ODS output files”

An ODS export specification document can contain the following optional com-mands:

Command	Description
ATTRIBUTEMAPDESC	The format for this command is: `ATTRIBUTEMAPDESC AttributeName` Use this command to specify the symbol attribute for the DescToUse parameter of the ODS extract.
ATTRIBUTEMAPFACT	The format for this command is: `ATTRIBUTEMAPFACT AttributeName` Use this command to specify the symbol attribute for the FactTableName parameter of the ODS extract.
ATTRIBUTEMAPID	The format for this command is: `ATTRIBUTEMAPID AttributeName` Use this command to specify the symbol attribute for the IndexToUse parameter of the ODS extract.
ATTRIBUTEMAPNAME	The format for this command is: `ATTRIBUTEMAPNAME AttributeName` Use this command to specify the symbol attribute for the NameToUse parameter of the ODS extract.
ERRORFILE	The format for this command is: `ERRORFILE FileName, EXTERNAL|INTERNAL` This command specifies the name of the error file to contain data on invalid records.
LOGFILE	The format for this command is: `LOGFILE FileName, EXTERNAL|INTERNAL` This command specifies the name of a log file to contain all variable values, errors, and error codes.

Command	Description
ODSAUTOLEVELDIM	The format for this command is: `ODSAUTOLEVELDIM DimensionName, ColumnName` When ODSAUTOLEVEL is ON, this command allows you to override the default auto level names and apply the desired field names based on dimension and level number. For each ODSAUTOLEVELDIM entry specified, you need to provide the appropriate mapping information through the ODSMAPPINGTABLE command, in the following format: ODSMAPPINGTABLE ITEM, ColumnName, ColumnNumber
ODSDBOUTPUT	The format for this command is: `ODSDBOUTPUT "DataSourceName", "UserName", "Password"` The output from an ODS extraction process usually creates a series of text files. However, it is also possible for this process to populate a series of database tables directly. For more information on dimension table options, see “Dimension Files”
ODSDEBUG	The format for this command is: `ODSDEBUG FileName, EXTERNAL|INTERNAL` This command specifies the name of a log file to contain debugging information.
ODSINCLUDEHIERARCHYLEVEL	The format for this command is: `ODSINCLUDEHIERARCHYLEVEL ON|OFF` This command specifies whether to include the Level column in the fact tables. The default specification for ODSINCLUDEHIERARCHYLEVEL is OFF. For more information, see “Fact Files”

Command	Description
ODSINCLUDEPARENTDATA	The format for this command is: `ODSINCLUDEPARENTDATA ON|OFF [, DimName][, DimName]...` This command specifies whether to include parent data. When determining this, you must consider whether the third-party application calculates its own totals or not. If it calculates its own totals, you likely do not need to export parent data values. The default specification for ODSINCLUDEPARENTDATA is OFF. To include parent data for all dimensions, specify: ODSINCLUDEPARENTDATA ON To include parent data for a single dimension, for example ENTITIES, specify: ODSINCLUDEPARENTDATA ON , ENTITIES To include parent data for multiple dimensions, for example ENTITIES and ACCOUNTS, specify: ODSINCLUDEPARENTDATA ON , ENTITIES, ACCOUNTS
ODSPATHFROMDB	The format for this command is: `ODSPATHFROMDB, "FolderPathName"` This command specifies a full path from the machine running the SQL Server to the folder containing the output files generated by the ODS extraction process. This command is not required with an Oracle Database Server.
ODSPUBLISHINGTIME	The format for this command is: `ODSPUBLISHINGTIME ON|OFF` This command specifies whether to log the publishing time into an output file or not.
ODSSTRUCTURE	The format for this command is: `ODSSTRUCTURE STAR|SNOWFLAKE|RECURSIVEHIERARCHYLF|RECURSIVEHIERA RCHYPF [, DimName]` This command specifies the structure of the output dimension files. The default is STAR. The optional DimName parameter specifies a dimension to which the specified structure should be applied. The output data for all other dimensions' defaults to the Star schema if the parameter is given. Otherwise, the output data for all dimensions has the specified structure.

Command	Description
ODSSYMPROPERTIES	The format for this command is: `ODSSYMPROPERTIES DEBITCREDIT|SymbolAttribute [DimName]` This command is used to include symbol properties as part of the metadata that is exported. If a DimName is specified, the DEBITCREDIT or SymbolAttribute value is included for all symbols that were extracted for that specific dimension. Otherwise, the DEBITCREDIT or SymbolAttribute value is included for all symbols that were extracted for all dimensions. **Note:** Multiple ODSSYMPROPERTIES commands can be specified in a single ODS export specification document. The properties that are exported are not used to format the fact files or the dimension files; they are written to a symbol properties file to be used by the calling third-party application.
ODSSYSPROPERTIES	The format for this command is: `ODSSYSPROPERTIES SystemAttribute` This command is used to include system attribute properties as part of the metadata that is exported. **Note:** Multiple ODSSYSPROPERTIES commands can be specified in a single ODS export specification document. The properties that are exported are not used to format the fact files or the dimension files; they are written to a system properties file to be used by the calling third-party application.

Writing the calling procedure document

The following code illustrates an example of a procedure document used to execute an ODS extraction:

```
// If not connected, connect (for more information on the CONNECT command,
see the CONNECT command in the Longview Developer's Guide)

CONNECT …

// Create the data area to be used

CREATE DATAAREA DA1 USING DataArea1.DSP
```

```
// Download data into this data area
DOWNLOAD DA1
// Run your ODS export spec using the just-read data area
RUN EXPORT TestODS.spec ON DA1
// Disconnect
DISCONNECT
// End of Procedure Document
```

Note: In the calling procedure document, the DOWNLOAD command or STREAM command can be used to download the data into the data area. The DOWNLOAD command downloads all data and builds the data tree before export. The STREAM command downloads and exports the data one block at a time. For more information, see the Download and Stream commands in the Longview Developer's Guide.

ODS Output Files

There are several types of output files that may be created:

Output file type	Status
Fact file(s)	Always created
TimeStamp file	Optionally created
Symbol Properties file	Optionally created
System Properties file	Optionally created
Dimension file(s)	Always created

The output from the ODS extraction process is always a set of text files. However, the ODSDBOUTPUT command also allows the extracted data and metadata to be loaded directly into database tables. For simplicity, these sections discuss only the text files.

Note: All ODS output files are generated in the working directory. If a relative path is used in the DATATARGET ODS command, the files are generated in the directory specified, relative to the current working directory.

Fact Files

The fact files are named

```
FileNamePrefix_FACT_FactTableName_nn.asc
```

where:

- **FileNamePrefix** is the parameter specified in the ODSTARGET line of the export specification document.
- **nn** is the FactTableNum.

An example of a fact table name is

```
ODSTEST_FACT_ACTUAL_01.ASC.
```

If this file already exists, it is overwritten.

There is one line for each data value retrieved. Each line in a fact table has the following format:

```
UseIndex1[Level1]{ UseIndex2[Level2]{…{ UseIndexN[LevelN]{Value
```

Where:

- **N** is the number of dimensions.
- **UseIndex1** is either the standard Longview symbol index for the symbol in the Xth dimension or the index specified in the IndexToUse parameter of the ODS attribute.
- **Level1** is the level of the related symbol. The Level column appears in the fact files only if Recursive Hierarchy is specified as the structure of the output dimension files.

An example of a row in a fact table is:

```
123{393{1{1{1{1{1{1{1000.00
```

TimeStamp File

If ODSPUBLISHINGTIME ON is specified, a time stamp file is created that includes the date and time of the ODS extraction. This file is named:

```
FilePrefix_ Publishing_Time.asc
```

If a file with this name already exists, it is overwritten. If the output is also being written to database tables, and the TimeStamp table is not empty, a new row is added to that table.

Symbol Properties File

If option ODSSYMPROPERTIES is specified, a symbol properties file is created that contains the DEBITCREDIT indicator, if requested, and the values for all requested symbol attributes. This file is named:

```
FilePrefix_ Symbol_Properties.asc
```

If a file with this name already exists, it is overwritten.

System Properties File

If option ODSSYSPROPERTIES is specified, a system properties file is created that contains properties and values for all requested system attributes. This file is named:

```
FilePrefix_ System_Properties.asc
```

If a file with this name already exists, it is overwritten.

Dimension Files

The format for the dimension files is typically the biggest consideration when performing an ODS extraction process. This decision is primarily influenced by how the third-party application needs to associate dimension metadata with the associated fact files.

There are several types of dimension table output formats:

- Star
- SnowFlake
- Recursive Hierarchy (Parents First or Leafs First)

The number of files for a dimension depends on which ODSSTRUCTURE setting is used for that dimension:

- Star or Recursive Hierarchy — The system creates one dimension file per dimension/hierarchy combination.

  If two ACCOUNTS hierarchies, three ENTITIES hierarchies, and a single hierarchy from each of the other dimensions are requested, the system creates two ACCOUNTS dimension files, three ENTITIES dimension files, and one file for each of the other dimensions.

- Snowflake — The system creates one dimension file per dimension/field name.

  Using the same example as above, where the entities are mapped to three fields, the system creates three ENTITIES dimension files (one per field), and one dimension file for each of the other dimensions (assuming they are all mapped to "Continent").

The name of a dimension file depends on which dimension output structure is being used:

Star or Recursive Hierarchy

The dimension files are named:

```
FilePrefix_DIM_DimName_RootSymbol_nn.asc
```

where:

- **nn** is a sequential number that refers to the fact table. If a dimension has six hierarchies extracted into six different fact tables, the dimension files for that dimension are numbered 1 to 6, each with its own unique root symbol name.

SnowFlake

The dimension files follow a generic naming format:

```
FilePrefix_DIM_DimName_Lxx_nn.asc
```

where:

- **xx** is a field number.
- **nn** is a sequential number that refers to the fact table.

If the ATTRIBUTEMAP command is used in the export specification document, Lxx is replaced by the value specified for AttrValue when the SET ATTRIBUTE command was used.

If ODSAUTOLEVEL is set to ON, and the ODSAUTOLEVELDIM command is used, Lxx is replaced by the ColumnName specified in the ODSAUTOLEVELDIM command. In addition, in the dimension tables generated, the Id column and the level names (Lxx) are replaced by ColumnNameId (the ColumnName suffixed with the string Id).

Note: For more information on field names, see “Field names in ODS output files”

The dimension format is specified by the ODSSTRUCTURE command. It has the following format:

```
ODSSTRUCTURE STAR|SNOWFLAKE|RECURSIVEHIERARCHYPF|RECURSIVEHIERARCHYLF,
["<DimName>"]
```

If a DimName is specified:

- That dimension has the selected format
- All other dimensions are in the STAR format

If a DimName is not specified:

- All dimensions are in the format specified in the command

Star format

The Star format creates one file per dimension/hierarchy combination. Within each file, there is one line for each “leaf symbol” within that dimension or hierarchy.

Note: The “leaf symbols” that correspond to the lines in the dimension files are the symbols tat the bottom of the hierarchy. They may not be “true” leaf symbols.

Each line:

- contains the full ancestry of a symbol.
- has the following format:

```
UseIndex1{UseName1{UseDesc1{UseIndex2{UseName1{UseDesc1{…
```

- has a varying length depending on how deep this symbol is under the main root symbol

Using the earlier example, if the ODS extraction requests NorthAmerica#99:

- Field1 = Continent (as specified by the ODSMAPPINGTABLE command)
- Field2 = Country (as specified by the ODSMAPPINGTABLE command)
- Field3 = City (as specified by the ODSMAPPINGTABLE command)

An example line is:

```
214{NorthAmerica{North America{Canada{Canada{Toronto{Toronto
```

SnowFlake format

The SnowFlake format creates one-dimension file per dimension/field name.

Within each file, there is one line for each extracted symbol.

Each line contains the following fields:

- UseIndex
- UseIndex (of the parent symbol)
- UseName
- UseDesc an example line is:

```
2222{123{Canada{Canada
```

Recursive Hierarchy format

The Recursive Hierarchy format creates one file per dimension/hierarchy combination.

Within each file, there is one line for each extracted symbol within that dimension or hierarchy.

Note: If Recursive Hierarchy is specified as the structure of the output dimension files, the Level field is added to each line of the fact files

Each line contains the following fields:

- UseIndex
- UseName
- UseDesc
- Level (with respect to its root symbol)
- Sequence number
- UseIndex (of the parent symbol)

The lines in the file area order either parent symbols first or leaf symbols first depending on whether RECURSIVEHIERARCHYPF (parent symbols first) or RECURSIVEHIERARCHYLF (leaf symbols first) is specified.

An example line is:

```
2222{Canada{Canada{1{2{123
```

Field names in ODS output files

Field names are used in naming the ODS output files, the output table names, and the columns of the tables for the SnowFlake and Star formats. Field names are determined by several factors.

- When ODSAUTOLEVEL is set to OFF, the field name is determined by the symbol attribute specified in the ATTRIBUTEMAP command or the ATTRIBUTEMAPFIELD command.

  Note: If both the ATTRIBUTEMAP command and the ATTRIBUTEMAPFIELD command are specified, the field name specified in the ATTRIBUTEMAP command is used

- When ODSAUTOLEVEL is set to ON, the name designated for each level is determined by matching corresponding ODSAUTOLEVELDIM entries with ODSMAPPINGTABLE entries. Any levels that are not named by matching entries are assigned a default name DimensionName_Lxx, where xx refers to the level number. These default names must be included as ODSMAPPINGTABLE entries.

  Note: The ODS export specification document can contain the ATTRIBUTEMAP command and the ATTRIBUTEMAPFIELD command when ODSAUTOLEVEL is set to ON. In this scenario, the system checks if the attribute is set for each symbol. If at least one symbol has the attribute set for a given level, the field name chosen in the attribute overrides the auto level field name. Multiple symbols at the same level should not have the attribute set to different field names.

Rules for the ODS extraction process

Prior to creating the documents, you need for performing the ODS extraction process, you need to be aware of the following rules that govern the process and guide the creation of the documents:

- The positive/negative weightings in the Longview hierarchies are ignored.
- The sign for values relating to "Credit" accounts are reversed automatically, if REVERSESIGN is set to ON.
- Only numeric data is extracted by this feature. String data is ignored.
- Multiple fact tables can be specified for a single dimension only. Therefore, fact table names that map to values other than 1 are allowed only in zero or one dimension.
- All fact table numbers must be sequential.
- Field names in ODS output tables must adhere to the following rules:
  - The first character in the field name must be an upper- or lower-case character.
  - Subsequent characters in the field name can be any alpha-numericcharacter, or the following special characters: dollar sign ($), pound (#), or underscore (_).
- The FieldName fields in the ODSMappingTable must be unique.
- Field names specified in the ODSMAPPINGTABLE and ODSAUTOLEVELDIM commands cannot exceed 64 characters.
- The FieldNum values for the symbols in any given hierarchy must be in ascending order as the hierarchy is traversed from the root symbol to the leaf symbols.
- The first FieldNum for each hierarchy should have a value of 1.
- Once a FieldNum value is encountered, it should not be repeated for any of its descendants.
- Whenever a FieldNum value of 1 is encountered, a new sub-hierarchy is started.
- All the symbols in the sub-hierarchy must belong to the same fact table.
- A symbol may exist in a dimension file without having any corresponding values in a data file. All lines in the generated data file correspond to a line in each of the dimension files.
- For the SnowFlake dimension file format, multiple hierarchies can be included, if they are independent and there are no level conflicts.
- Alternate rollups of symbols are not supported in the SnowFlake dimension file format.

Note: In Oracle only, any object (table name, index name, or column name) exceeding 30 characters is truncated and suffixed with a unique numerical identifier

Web Services

Web services allow organizations to communicate data with each other and with clients, without deep knowledge of each other's IT systems behind the firewall. The web service provider defines a format for requests for its service and the response that the service generates. A computer then makes a request for the web services across the network. The web service performs an action, and sends the response back.

Open, Extensible Markup Language (XML) standard applications and various other protocols are used to exchange data with other web-based applications. XML is used to tag the data, Simple Object Access Protocol (SOAP) is used to transfer the data, Web Service Definition Language (WSDL) is used for describing the services available, and Universal Description, Discovery and Integration (UDDI) is used for listing what services are available.

Integral to web services, XML is a web document description language. XML is used to describe content using the application of concealed tags and identifying labels. This method of classifying web data is extremely effective, making web content easy to identify, analyze, and exchange.

SOAP is a W3C standard protocol that defines the format for web service requests. SOAP messages are sent back and forth between the service provider and service user in SOAP envelopes, containing a request for an action and the result of that action. SOAP envelopes are XML formatted, and are easy to decode. WSDL is an XML-based language, used for describing web service functionality. WSDL describes the web service request, the expected parameters, and the expected response. Unlike traditional client/server models, such as a web server/web page system, web services do not provide the user with a GUI. Web services instead share business logic, data, and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the web service to a GUI (such as a web page or an executable program) to offer specific functionality to users. In Longview, Data Server web services cover the basic operations of symbols, hierarchies, and data. All Data Server web services support Longview authentication. Windows and third-party web authentication are not supported at this time.



Note: You must install the Data Server Web Service software to work with web services. For more information see the Longview Installation Guide.

The following table lists the available web services in Longview:

Category	Method	WSDL URL
Symbols/ Hierarchies	■ Assign Parent ■ Create ■ Delete ■ Remove Parent ■ Retrieve ■ Update	http://host:port/axis2/services/Symbol?wsdl where: ■ host is the name of the web services ■ Web Server.port is the web service port.
Data	■ Lock ■ Retrieve ■ Retrieve to File ■ Submit ■ Submit to File ■ Unlock	http://host:port/axis2/services/Data?wsdl where: ■ host is the name of the web services ■ Web Server.port is the web service port.

In this section, you can find information on the following topics:

- Common Web Service Elements
- Symbol Web Services
- Data Web Services

Common Web Service Elements

Web services have the following common elements:

- Connection Information
- Result Information

Connection Information

Connection information is specified in each Data Server web service request. Each of the web service functions performs its own connection to Longview. Therefore, each function contains a common section to define the parameters required to establish this connection.

The connection is specified in a similar manner to the following:

V7

```
<sym:connInfo>
    <dat:user>admin</dat:user>
    <dat:password>Admin99</dat:password>
    <dat:dataServerName>work_dataserver</dat:dataServerName>
    <dat:role>Administrator_Access</dat:role>
    <dat:group>Administrators</dat:group>
</sym:connInfo>
```

Element.Attribute	Value	Notes
User	A Longview user name	
Password	A Longview user password	If a Longview user password includes special characters, such as <, >, & and %, you must pass these characters as XML entities (<, >, and &, respectively). For example, if the password is password&, type password&.
dataServerName	A Data Server name	This is the Data Server name.
Role	A Longview user role	This is the symbol access role. If the Data Server is configured to use V3 Compatible Access, this parameter is ignored.
Group	A Longview group ID	This is required only if the Data Server is configured to use V3 Compatible Access.

Result Information

Many Data Server web service responses have the status element as part of the response. In some cases, this status structure may be the entire response. In other cases, it supplements other response fields.

The result information is specified in a similar manner to the following:

V7

```
<_:Response>
<status>
    <type>Error </type>
```

```
    <code>-10007</code>
    <message>Invalid Schedule Name (S1043)</message>
</status>
</_:Response>
```

Element.Attribute	Value	Notes
Type		This is the status type.
Code	An integer	This is the return code. 0 indicates a successful result. A negative number indicates an Error or Warning
Message	A description of the Status	This is the warning or error message.

Symbol Web Services

The following web service functions related to symbols are available in Longview:

- Assign Parent
- Create
- Delete
- Remove Parent
- Retrieve
- Update

Assign Parent

The Assign Parent function sends a request to assign existing symbols to existing parents in the Longview system. The request to assign parent-child relationships is similar to the following example:

V7 Request

```
<sym:assignParentRequest>
    <!--1 or more repetitions:-->
    <web:assignParent priority="1">
      <web:dimension>Accounts</web:dimension>
      <web:child>41010</web:child>
      <web:parent>Test</web:parent>
      <web:weight>+</web:weight>
    </web:assignParent>
```

```
    <web:assignParent priority="2">
      <web:dimension>Accounts</web:dimension>
      <web:child>41200</web:child>
      <web:parent>Test</web:parent>
      <web:weight>+</web:weight>
      </web:assignParent>
    </sym:assignParentRequest>
```

Element.Attribute	Value	Notes
Priority	An integer	
Dimension	A dimension name	
Child	A child symbol name	
Parent	A parent symbol name	
Weight	+, -, or 0	The default value is 0.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:assignParentResponse xmlns:_="http://longview.com/dataservices/
WebServices/Symbol" xmlns="http://longview.com/dataservices"
xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:assignParentResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:assignParentResult>
  </_:assignParentResponse>
```

Create

The Create function sends a request to create new symbols in the Longview system.

The request to create symbols is similar to the following example:

V7 Request

```
<sym:createRequest>
    <!--1 or more repetitions:-->
```

```
    <web:symbol>
      <web:dimension>Accounts</web:dimension>
      <web:symbolInfo virtual="false" receiveRollups="true">
        <web:name>Test</web:name>
        <!--Optional:-->
        <web:type>Standard</web:type>
        <!--Optional:-->
        <web:balanceType>Debit</web:balanceType>
        <!--Optional:-->
        <web:sortChildren>Manual</web:sortChildren>
        <!--0 to 2 repetitions:-->
        <web:description>
          <web:language>en</web:language>
          <web:description>Test Symbol</web:description>
        </web:description>
      </web:symbolInfo>
    </web:symbol>
  </sym:createRequest>
```

Element.Attribute	Value	Notes
Dimension	The name of the dimension in which the symbol is created	
Virtual	True or false	The default value is false
receiveRollUps	True or false	The default value is true.
Name	The name of the symbol	
Type	Standard, Static, or CarryForward	
balanceType	Credit, Debit, or Neither	Symbols in the Accounts dimension can have a balanceType of Credit, Debit, or Neither. Symbols in all other dimensions have a balanceType of Neither. The default value is Neither.
sortChildren	Ascending, Descending, or Manual	The default value is Manual
Description	The description of the symbol	The system allows for multiple language descriptions. Each one is specified with a two character language code, such as en.

For more information on attributes, see the *Longview Application Administrator Guide*.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:createResponse xmlns:_="http://longview.com/dataservices/WebServices/
Symbol" xmlns="http://longview.com/dataservices" xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:createResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:createResult>
</_:createResponse>
```

Delete

The Delete function sends a request to delete existing symbols in the Longview system. The request to delete symbols is similar to the following example:

V7 Request

```
<sym:deleteRequest>
    <!--1 or more repetitions:-->
    <web:symbol>
      <web:dimension>Accounts</web:dimension>
      <web:symbol>TestNew</web:symbol>
    </web:symbol>
</sym:deleteRequest>
```

Element.Attribute	Value
Dimension	A dimension name
Symbol	A symbol name

The response from the Data Server is similar to the following example:

V7 Response

```
<_:deleteResponse xmlns:_="http://longview.com/dataservices/WebServices/
Symbol" xmlns="http://longview.com/dataservices" xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
```

```
2001/XMLSchema-instance">
    <_:deleteResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:deleteResult>
</_:deleteResponse>
```

Remove Parent

The Remove Parent function sends a request to remove a symbol that is currently attached to a parent in the Longview system. The request to remove parent-child relationships is similar to the following example:

V7 Request

```
<sym:removeParentrequest>
    <!--1 or more repetitions:-->
    <web:removeParent>
      <web:dimension>Accounts</web:dimension>
      <web:child>41010</web:child>
      <web:parent>TestNew</web:parent>
    </web:removeParent>
    <web:removeParent>
      <web:dimension>Accounts</web:dimension>
      <web:child>41200</web:child>
      <web:parent>TestNew</web:parent>
    </web:removeParent>
</sym:removeParentrequest>
```

Element.Attribute	Value
Dimension	A dimension name
Child	A child symbol name
Parent	A parent symbol name

The response from the Data Server is similar to the following example:

V7 Response

```
<_:removeParentResponse xmlns:_="http://longview.com/dataservices/
```

```
WebServices/Symbol" xmlns="http://longview.com/dataservices"
xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:removeParentResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:removeParentResult>
</_:removeParentResponse>
```

Retrieve

The Retrieve function sends a request to retrieve symbols based on a symbol specification in the Longview system. There are two main types of information retrieved in this method:

- Symbol information — This includes fields such as name, index, descriptions, type, priority, etc and relates to the specific symbol.
- Parentage information — This includes fields such as the symbol indexes of the parents of each specific symbol and the weight of the symbol/parent rollups.

The request to retrieve symbols is similar to the following example:

V7 Request

```
<sym:retrieveRequest>
    <!--Optional:-->
    <web:dimension>Accounts</web:dimension>
      <!--Zero or more repetitions:-->
    <web:symbolSpec>Trial_Balance###</web:symbolSpec>
</sym:retrieveRequest>
```

Element.Attribute	Value	Notes
Dimension	A dimension name	
symbolSpec	A symbol name and notation, such as IAS_Statements#99 (99 levels of the IAS_ Statements hierarchy)	When only the symbol name is specified, the default is #0. A symbol does not need to be specified, in which case all symbols in the dimension are retrieved.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:retrieveResponse xmlns:_
="http://longview.com/dataservices/WebServices/Symbol"
xmlns="http://longview.com/dataservices"
xmlns:tns="http://longview.com/dataservices/
webServiceObject" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <_:retrieveResult>
    <tns:status>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </tns:status>
    <tns:symbol
xmlns:ns0="http://longview.com/dataservices/WebServices/Data"
    xmlns:ns1="http://longview.com/dataservices/WebServices/Symbol">
      <tns:symbolInfo virtual="false" receiveRollups="true">
        <tns:name>11101</tns:name>
        <tns:type>CarryForward</tns:type>
        <tns:balanceType>Debit</tns:balanceType>
        <tns:sortChildren>Manual</tns:sortChildren>
        <tns:description>
          <tns:language>en</tns:language>
          <tns:description>11101 - Cash</tns:description>
        </tns:description>
        <tns:description>
          <tns:language>fr</tns:language>
          <tns:description>11101 - Comptant</tns:description>
        </tns:description>
      </tns:symbolInfo>
      <tns:parent priority="50">
        <tns:name>11100</tns:name>
        <tns:weight>+</tns:weight>
      </tns:parent>
    </tns:symbol>
<tns:symbol xmlns:ns0="http://longview.com/dataservices/WebServices/Data"
xmlns:ns1="http://longview.com/dataservices/WebServices/Symbol">
     <tns:symbolInfo virtual="false" receiveRollups="true">
      <tns:name>11110</tns:name>
```

```
      <tns:type>CarryForward</tns:type>
      <tns:balanceType>Debit</tns:balanceType>
      <tns:sortChildren>Manual</tns:sortChildren>
      <tns:description>
        <tns:language>en</tns:language>
        <tns:description>11110 - Temporary cash
investments</tns:description>
      </tns:description>
      <tns:description>
        <tns:language>fr</tns:language>
        <tns:description>11110 - Invest. comptant temp.</tns:description>
      </tns:description>
    </tns:symbolInfo>
      <tns:parent priority="60">
      <tns:name>11100</tns:name>
      <tns:weight>+</tns:weight>
      </tns:parent>
    </tns:symbol>
…
</_:retrieveResult>
</_:retrieveResponse>
```

Element.Attribute	Value	Notes
Type	Error, Success, or Warning	
Virtual	True or false	
receiveRollUps	True or false	
Name	A symbol name	
Type	Standard, Static, or CarryForward	
balnceType	Credit, Debit, or Neither	Symbols in the Accounts dimension can have a balanceType of Credit, Debit, or Neither. Symbols in all other dimensions have a balanceType of Neither.
sortChildren	Ascending, Descending, or Manual	Sort children by name (ascending or descending) or manually.
Language	A two-character language code, such as en	

Element.Attribute	Value	Notes
Description		The system allows for multiple language descriptions. Each one is specified with a two character language code, such as en.
Child	A child symbol name	
Parent	A parent symbol name	
Weight	+, -, or -	
Priority	An integer	

Update

The Update web service sends a request to update properties of an existing symbol in the Longview system. The request to update symbol information is similar to the following example:

V7 Request

```
<sym:updateRequest>
    <!--Zero or more repetitions:-->
    <web:symbol>
      <web:dimension>Accounts</web:dimension>
      <!--Optional:-->
      <web:newName>TestNew</web:newName>
      <!--Optional:-->
      <web:symbolInfo virtual="true" receiveRollups="true">
        <web:name>Test</web:name>
      </web:symbolInfo>
    </web:symbol>
</sym:updateRequest>
```

Element.Attribute	Value	Notes
Dimension	A dimension name	
newName	The new name of the symbol	This is optional. A new symbol name is only required when you are renaming a symbol.
Virtual	True or false	This is optional. If virtual is not specified, the previous value is kept.
receiveRollUps	True or false	This is optional. If receiveRollUps is not specified, the previous value is kept.
Name	The name of the symbol	

Element.Attribute	Value	Notes
Type	Standard, Static, or CarryForward	This is optional. If type is not specified, the previous value is kept.
balanceType	Credit, Debit, or Neither	This is optional. If balanceType is not specified, the previous value is kept. Symbols in the Accounts dimension can have a balanceType of Credit, Debit, or Neither. Symbols in all other dimensions have a balanceType of Neither.
sortChildren	Ascending, Descending, or Manual	This is optional. If sortChildren is not specified, the previous value is kept.
Language	A two-character language code, such as en	
Description	The description of the symbol	This is optional. If description is not specified, the previous value is kept.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:updateResponse xmlns:_="http://longview.com/dataservices/WebServices/
Symbol" xmlns="http://longview.com/dataservices" xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:updateResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:updateResult>
</_:updateResponse>
```

Data Web Services

The following web service functions related to data are available in Longview:

- Lock
- Retrieve
- Retrieve to File
- Submit

- Submit from File
- Unlock

Lock

The Lock function sends a request to lock certain data areas in the Longview system. The request to create a lock on a certain data area is similar to the following example:

V7 Request

```
<data:lockRequest doJECheck="false">
    <web:comment>Dallas Trial Balance Load</web:comment>
    <!--1 or more repetitions:-->
    <web:dimensionSymbolSpec>
      <web:dimension>Accounts</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>Data_Load_Accounts###</web:symbolSpec>
    </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>TimePeriods</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>CYr_09_YTD</web:symbolSpec>
    </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>Entities</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>E11220</web:symbolSpec>
     </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>Currencies</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>USD</web:symbolSpec>
    </web:dimensionSymbolSpec>
    …
      <web:dimensionSymbolSpec>
      <web:dimension>DIMENSION15</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>DIMENSION15_Default</web:symbolSpec>
```

```
    </web:dimensionSymbolSpec>
</data:lockRequest>
```

Element. Attribute	Value	Notes
JECheck	True or false	This is optional. This indicates whether to perform a check on any unposted journal entries in the data area specified. The default value is false.
Comment	A comment	This is optional.
Schedule	A schedule name	This is optional.
symbolSpec	A symbol name and notation, such as IAS_Statements#99 (99 levels of the IAS_Statements hierarchy)	When only the symbol name is specified, the default value is #0.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:lockResponse xmlns:_="http://longview.com/dataservices/WebServices/Data"
xmlns="http://longview.com/dataservices" xmlns:tns="http://longview.com/
dataservices/webServiceObject"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
    <_:lockResult>
      <tns:status>
        <tns:type>Success</tns:type>
        <tns:code>0</tns:code>
      </tns:status>
      <tns:lockId>1</tns:lockId>
    </_:lockResult>
</_:lockResponse>
```

Retrieve

The Retrieve function sends a request to retrieve an area of data from the Longview system. This method returns an array of values found in the requested area of data. Each retrieved data value returns:

- the names of the corresponding symbols
- a data type of numeric or string indicating whether a number or string was found the data value

The request to retrieve certain base data areas is similar to the following example:

V7 Request

```
<data:retrieveRequest unadjusted="true" leafDataOnly="true" ctaData="false">
    <!--1 or more repetitions:-->
    <web:dimensionSymbolSpec>
      <web:dimension>Accounts</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>Data_Load_Accounts###</web:symbolSpec>
    </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>TimePeriods</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>CYr_09_YTD</web:symbolSpec>
    </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>Entities</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>E11220</web:symbolSpec>
    </web:dimensionSymbolSpec>
    <web:dimensionSymbolSpec>
      <web:dimension>Currencies</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>USD</web:symbolSpec>
    </web:dimensionSymbolSpec>
    …
    <web:dimensionSymbolSpec>
      <web:dimension>DIMENSION15</web:dimension>
      <!--Zero or more repetitions:-->
      <web:symbolSpec>DIMENSION15_Default</web:symbolSpec>
    </web:dimensionSymbolSpec>
</data:retrieveRequest>
```

Element.Attribute	Value	Notes
Unadjusted	True or false	The default value is true.
leafDataOnly	True or false	The default value is true.
ctaData	True or false	The default value is false.
ruleID	An integer	
Schedule	A schedule name	This is optional

Element.Attribute	Value	Notes
Dimension	A dimension name	A dimensionSymbolSpec (containing the dimension and symbolSpec attributes) is required for each dimension in the system.
symbolSpec	A symbol name and notation, such as IAS_ Statements#99 (99 levels of the IAS_Statements hierarchy)	When only the symbol name is specified, the default value is #0. A imensionSymbolSpec (containing the dimension and symbolSpec attributes) is required for each dimension in the system.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:retrieveResponse xmlns:_
="http://longview.com/dataservices/WebServices/Data"

xmlns="http://longview.com/dataservices"
xmlns:tns="http://longview.com/dataservices/

webServiceObject" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <_:retrieveResult>

      <tns:status>

        <tns:type>Success</tns:type>

        <tns:code>0</tns:code>

      </tns:status>

      <tns:dataCell
xmlns:ns0="http://longview.com/dataservices/WebServices/Data"

      xmlns:ns1="http://longview.com/dataservices/WebServices/Symbol">

        <tns:dimensionSymbol>

          <tns:dimension>Accounts</tns:dimension>

          <tns:symbol>31120</tns:symbol>

        </tns:dimensionSymbol>

        <tns:dimensionSymbol>

          <tns:dimension>TimePeriods</tns:dimension>

          <tns:symbol>CYr_09_YTD</tns:symbol>

        </tns:dimensionSymbol>

        <tns:dimensionSymbol>

          <tns:dimension>Entities</tns:dimension>

          <tns:symbol>E11220</tns:symbol>

        </tns:dimensionSymbol>

        <tns:dimensionSymbol>
```

```
          <tns:dimension>Currencies</tns:dimension>
          <tns:symbol>USD</tns:symbol>
        </tns:dimensionSymbol>
        …
        <tns:dimensionSymbol>
          <tns:dimension>DIMENSION15</tns:dimension>
          <tns:symbol>DIMENSION15_Default</tns:symbol>
        </tns:dimensionSymbol>
        <tns:type>numeric</tns:type>
        <tns:value>10000</tns:value>
      </tns:dataCell>
      <tns:dataCell
xmlns:ns0="http://longview.com/dataservices/WebServices/Data"
      xmlns:ns1="http://longview.com/dataservices/WebServices/Symbol">
        <tns:dimensionSymbol>
          <tns:dimension>Accounts</tns:dimension>
          <tns:symbol>31110</tns:symbol>
        </tns:dimensionSymbol>
        <tns:dimensionSymbol>
          <tns:dimension>TimePeriods</tns:dimension>
          <tns:symbol>CYr_09_YTD</tns:symbol>
        </tns:dimensionSymbol>
        <tns:dimensionSymbol>
          <tns:dimension>Entities</tns:dimension>
          <tns:symbol>E11220</tns:symbol>
         </tns:dimensionSymbol>
        <tns:dimensionSymbol>
          <tns:dimension>Currencies</tns:dimension>
          <tns:symbol>USD</tns:symbol>
        </tns:dimensionSymbol>
        …
        <tns:dimensionSymbol>
          <tns:dimension>DIMENSION15</tns:dimension>
          <tns:symbol>DIMENSION15_Default</tns:symbol>
        </tns:dimensionSymbol>
        <tns:type>numeric</tns:type>
```

```
        <tns:value>10000</tns:value>
      </tns:dataCell>
      …
    </_:retrieveResult>
</_:retrieveResponse>
```

Retrieve to File

The Retrieve to File function sends a request to retrieve an area of data from the Longview system to a file. The Retrieve function's result can be very large given the XML verbosity. The Retrieve to File function offers an alternative that represents the result set in a more compact manner.

The request to retrieve certain base data areas from a file is similar to the following example:

V7 Request

```
<data:retrieveToFileRequest>
    <web:dataRetrieveRequest unadjusted="true" leafDataOnly="true"
ctaData="false">
      <!--1 or more repetitions:-->
      <web:dimensionSymbolSpec>
        <web:dimension>Accounts</web:dimension>
        <!--Zero or more repetitions:-->
        <web:symbolSpec>Data_Load_Accounts###</web:symbolSpec>
      </web:dimensionSymbolSpec>
      <web:dimensionSymbolSpec>
        <web:dimension>TimePeriods</web:dimension>
        <!--Zero or more repetitions:-->
        <web:symbolSpec>CYr_09_YTD</web:symbolSpec>
      </web:dimensionSymbolSpec>
      <web:dimensionSymbolSpec>
        <web:dimension>Entities</web:dimension>
        <!--Zero or more repetitions:-->
        <web:symbolSpec>E11220</web:symbolSpec>
      </web:dimensionSymbolSpec>
      <web:dimensionSymbolSpec>
        <web:dimension>Currencies</web:dimension>
        <!--Zero or more repetitions:-->
        <web:symbolSpec>USD</web:symbolSpec>
```

```
      </web:dimensionSymbolSpec>
      …
      <web:dimensionSymbolSpec>
        <web:dimension>DIMENSION15</web:dimension>
         <!--Zero or more repetitions:-->
        <web:symbolSpec>DIMENSION15_Default</web:symbolSpec>
      </web:dimensionSymbolSpec>
    </web:dataRetrieveRequest>
    <web:uncOutputFileName>C:\Users\Administrator\Desktop\Files\Dallas_
TrialBalance_
    Retrieve.txt</web:uncOutputFileName>
</data:retrieveToFileRequest>
```

Element.Attribute	Value	Notes
Unadjusted	True or false	The default value is true.
leafDataOnly	True or false	The default value is true.
ctaData	True or false	The default values is false.
ruleID	An integer	
Schedule	A schedule name	This is optional.
Dimension	A dimension name	A dimensionSymbolSpec (containing the dimension and symbolSpec attributes) is required for each dimension in the system.
symbolSpec	A symbol name and notation, such as IAS_ Statements#99 (99 levels of the IAS_Statements hierarchy)	When only the symbol name is specified, the default value is #0. A dimensionSymbolSpec (containing the dimension an symbolSpec attributes) is required for each dimension in the system.
uncOutputFileName	A file name	

The response from the Data Server is similar to the following example:

V7 Response

```
<_:retrieveToFileResponse xmlns:_="http://longview.com/dataservices/
WebServices/Data" xmlns="http://longview.com/dataservices"
xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:retrieveToFileResult>
      <tns:type>Success</tns:type>
```

```
      <tns:code>0</tns:code>
    </_:retrieveToFileResult>
</_:retrieveToFileResponse>
```

Submit

The Submit function sends a request to submit data to the Longview system.

The request to submit certain base data areas is similar to the following example:

V7 Request

```
<data:submitRequest>
    <web:options clearDataArea="true" reverseSign="false" serverMath="true"
    submitForApproval="false" maxErrors="0" lockID="1">
      <web:comment>Load Dallas Trial Balance</web:comment>
    </web:options>
    <!--1 or more repetitions:-->
    <web:dataCell>
      <web:dimensionSymbol>
        <web:dimension>Accounts</web:dimension>
        <web:symbol>11430</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>TimePeriods</web:dimension>
        <web:symbol>CYr_09_YTD</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>Entities</web:dimension>
        <web:symbol>E11220</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>Currencies</web:dimension>
        <web:symbol>USD</web:symbol>
      </web:dimensionSymbol>
      …
      <web:dimensionSymbol>
        <web:dimension>DIMENSION15</web:dimension>
```

```
        <web:symbol>DIMENSION15_Default</web:symbol>
      </web:dimensionSymbol>
      <web:type>numeric</web:type>
      <web:value>10994.730000</web:value>
    </web:dataCell>
    <web:dataCell>
      <web:dimensionSymbol>
        <web:dimension>Accounts</web:dimension>
        <web:symbol>11440</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>TimePeriods</web:dimension>
        <web:symbol>CYr_09_YTD</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>Entities</web:dimension>
        <web:symbol>E11220</web:symbol>
      </web:dimensionSymbol>
      <web:dimensionSymbol>
        <web:dimension>Currencies</web:dimension>
        <web:symbol>USD</web:symbol>
      </web:dimensionSymbol>
      …
      <web:dimensionSymbol>
        <web:dimension>DIMENSION15</web:dimension>
         <web:symbol>DIMENSION15_Default</web:symbol>
      </web:dimensionSymbol>
       <web:type>numeric</web:type>
       <web:value>357.330000</web:value>
    </web:dataCell>
    <web:dataCell>
    …
</data:submitRequest>
```

Element.Attribute	Value	Notes
clearDataArea	True or false	This is optional. The default value is false

Element.Attribute	Value	Notes
reverseSign	True or false	This is optional. The default value is false
serverMath	True or false	This is optional. This indicates whether the server math processing logic should be turned off, or retains its current settings during the submission. The default is true.
submitForApproval	True or false	This is optional. This indicates whether the area should be submitted for approval. The default value is false.
maxErrors	An integer	This is optional. The default value is 0.
lockId	A lock ID	This is the ID of the lock. The lock must cover all of the data being submitted.
Comment	A comment	This is optional.
workflowComment	A workflow comment	This is only required if submitForApproval is true.
dataCell	A dataCell contains every dimension in the system, along with the symbol name of each dimension	More than one dataCell can be specified.
Dimension	A dimension name	The dimension, symbol, type, and value attributes make up one set per value being submitted. There is one dimension and one symbol attribute per dimension.
Symbol	A symbol name	The dimension, symbol, type, and value attributes make up one set per value being submitted. There is one dimension and one symbol attribute per dimension.
Type	Numeric or string	The dimension, symbol, type, and value attributes make up one set per value being submitted.
Value	A data value	The value of the submitted data. The dimension, symbol, type, and value attributes make up one set per value being submitted.

The response from the Data Server is similar to the following example:

V7 Response

```
<_:submitResponse xmlns:_
="http://longview.com/dataservices/WebServices/Data"

xmlns="http://longview.com/dataservices"
xmlns:tns="http://longview.com/dataservices/

webServiceObject" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <_:submitResult>

      <tns:type>Success</tns:type>

      <tns:code>0</tns:code>
```

```
      <tns:message>(Data Submission) Timer started at 2011/11/09 07:21:24</
       tns:message>
      <tns:message>SERVERMATH: DEFAULT</tns:message>
      <tns:message>MAXERROR: 0</tns:message>
      <tns:message>Records read: 435, Records rejected: 0, Records filtered:
0</
       tns:message>
      <tns:message>(Data Submission) Timer stopped at 2011/11/09 07:21:25</
       tns:message>
      <tns:message>(Data Submission) Submission assigned batch ID
15</tns:message>
      <tns:message>Data Submission completed successfully</tns:message>
    </_:submitResult>
</_:submitResponse>
```

Submit from File

The Submit from File function allows the submission of data values from a file to the Longview system. The Submit function's request can be very large, given the XML verbosity. The Submit from File function offers an alternative that represents the request set in a more compact manner.

The request to submit a certain base data area from a file is similar to the following example:

V7 Request

```
<data:submitFromFile>
    <data:connInfo>
      <dat:user>DalCont</dat:user>
      <dat:password>xxxx</dat:password>
      <dat:dataServerName>LongviewCPM</dat:dataServerName>
      <dat:role>V3_Compatible_Access</dat:role>
      <dat:group>ConU</dat:group>
    </data:connInfo>
    <data:submitFromFileRequest>
      <web:options clearDataArea="true" reverseSign="false"
serverMath="true"
       submitForApproval="false" maxErrors="0" lockID="1">
        <web:comment>Load Dallas Trial Balance</web:comment>
      </web:options>
```

```
<web:uncFileName>C:\Users\Administrator\Desktop\Files\Dallas_TrialBalance_
Load
.txt</web:uncFileName>
      </data:submitFromFileRequest>
</data:submitFromFile>
```

Element.Attribute	Value	Notes
clearDataArea	Qtrue or false	This is optional. The default value is false.
reverseSign	True or false	This is optional. The default value is false.
serverMath	true or false	This is optional. This indicates whether the math server should be turned off, or retains its current settings during the submission. The default is true.
submitForApproval	True or false	This is optional. This indicates whether the area should be submitted for approval. The default value is false.
maxErrors	An integer	This is optional. The default value is 0.
lockId	A lock ID	The is the ID of the lock.
Comment	A comment	This is optional
workflowComment	A workflow comment	This is only required if submitForApproval is true.
uncFileName	A file name	

The response from the Data Server is similar to the following example:

V7 Response

```
<_:submitFromFileResponse xmlns:_="http://longview.com/dataservices/
WebServices/Data" xmlns="http://longview.com/dataservices"
xmlns:tns="http://
longview.com/dataservices/webServiceObject" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance">
    <_:submitFromFileResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
      <tns:message>(Data Submission) Timer started at 2011/11/09 07:22:48</
       tns:message>
```

```
      <tns:message>SERVERMATH: DEFAULT</tns:message>
      <tns:message>MAXERROR: 0</tns:message>
      <tns:message>Records read: 435, Records rejected: 0, Records filtered:
0</
       tns:message>
      <tns:message>(Data Submission) Timer stopped at 2011/11/09 07:22:49</
        tns:message>
      <tns:message>(Data Submission) Submission assigned batch ID 19</
       tns:message>
      <tns:message>Data Submission completed successfully</tns:message>
    </_:submitFromFileResult>
</_:submitFromFileResponse>
```

Unlock

The Unlock function sends a request to release a previous lock on a data area in the Longview system. The request to release a previously acquired lock is similar to the following example:

V7 Request

```
<data:unlock>
    <data:connInfo>
      <dat:user>DalCont</dat:user>
      <dat:password>xxxx</dat:password>
      <dat:dataServerName>LongviewCPM</dat:dataServerName>
      <dat:role>V3_Compatible_Access</dat:role>
      <dat:group>ConU</dat:group>
    </data:connInfo>
    <data:unlockRequest>
      <web:lockId>1</web:lockId>
    </data:unlockRequest>
</data:unlock>
```

Element.Attribute	Value	Notes
lockId	A lock ID	This is the ID of the lock you want to release

The response from the Data Server is similar to the following example:

V7 Response

```
<_:unlockResponse xmlns:_
="http://longview.com/dataservices/WebServices/Data"
xmlns="http://longview.com/dataservices" xmlns:tns="http://longview.com/
dataservices/webServiceObject"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance">
    <_:unlockResult>
      <tns:type>Success</tns:type>
      <tns:code>0</tns:code>
    </_:unlockResult>
</_:unlockResponse>
```

REST APIs

The solutions framework provides standard APIs via REST. The base URI for these REST APIs is:

```
<HTTP protocol>://<web server>/<web bridge>/<Longview identifier>/api/app
```

Example:

```
http://localhost/cgi-bin/Longview/lvweb.cgi/Longview/api/app
```

The following services are provided:

- /solutions
- /solutions/attributevalue
- /solutions/attributevalue/{class}
- /solutions/attributevalue/{class}/{name}{query}
- GET /solutions/data
- PUT /solutions/data
- GET /solutions/hierarchies
- GET /solutions/hierarchies/{dimension}
- GET /solutions/hierarchies/{dimension}/{root}
- PUT /solutions/hierarchies/{dimension}/{root}{query}

To execute any solutions framework REST API, you must first create a session. For more information, see Creating a New Session - POST "/API/session" .

Remember to delete any session created after you are finished with it. For more information, see Destroying a Session – DELETE “/API/session” .

Responses

The following response headers will be returned with every request:

Header	Use
Status	Representation of the result of the request in the form: <status code> <status>
Content-Type	The type of content returned by the request.
Content-Length	The length of the content returned in the response body by the request.

The following standard response statuses may be returned in the response header for a request:

Status	Methods	Additional Response Headers
200 OK	GET / HEAD / PUT / DELETE	LVStatus
201 Created	POST	Location
400 Bad Request	All	
404 Not Found	All	
405 Method Not Allowed	PUT / POST / DELETE	Allow
406 Not Acceptable	GET / HEAD	
415 Unsupported Media Type	PUT / POST	

/solutions

This endpoint will return a list of links to the solutions API services provided in JSON format.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

URI Example:

```
http://localhost/cgi-bin/Longview/lvweb.cgi/Longview/api/app/solutions
```

Return Type: Application/json

```
{
    "links": {
        "attributevalue": "http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue",
        "data": "http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/data",
        "hierarchies": "http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies"
    }
}
```

/solutions/attributevalue

This endpoint will return a list of links to the solutions attribute value services provided in JSON format.

Authorizations Required:

To execute this API the user must have the ‘Connect to Application Framework’ authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

URI Example:

```
http://localhost/cgibin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue
```

Return Type: Application/json

```
{
    "links": {
        "symbol": "http://localhost/cgi-bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol",
        "system": "http://localhost/cgi-bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/system",
        "user": "http://localhost/cgi-bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/user"
    }
}
```

/solutions/attributevalue/{class}

This endpoint will return a list of links to specific attributes, of the class specified, available to query provided in JSON format.

Authorizations Required:

To execute this API the user must have the ‘Connect to Application Framework’ authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Request Parameters:

class (string): the attribute class (symbol, system, or user).

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol
```

Return Type: Application/json

```
{
    “links”:    {
        “ZFXSourceCurrencies”:    “http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol/ZFXS
ourceCurrencies”,
        “ZFXRateType”:    “http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol/ZFXR
ateType”,
        “ZFXTranslations”:    “http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol/ZFXT
ranslations”,
        “ZFXOverrideRates”:    “http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol/ZFXO
verrideRates”
    }
}
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain a list of attributes that are available to query. Response header LVStatus: ok
406 Not Acceptable	If the request specifies Accept in the request header that does not include application/json
404 Not Found	If the attribute class specified is not valid.
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

/solutions/attributevalue/{class}/{name}{query}

This endpoint will return the value of the attribute, provided in text/plain format.

If you return the result to an object in application framework the attribute value will be stored in the _data property of the target object.

Authorizations Required:

To execute this API the user must have the ‘Connect to Application Framework’ authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Request Parameters:

class (string): the attribute class (symbol, system, or user).

name (string): the name of the attribute.

query (string, optional): contains the query parameters for the attribute value request. Query parameters are required for symbol and user attribute value requests and take the form:

```
?object={objectName}
```

where:

- objectName is the name of a symbol when {class} is symbol.
- objectName is the name of a user when {class} is user.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/attributevalue/symbol
```

Return Type: Text/plain

```
CAD
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain the value of the requested attribute for the specified symbol object. Response header LVStatus: ok
400 Bad Request	If the OBJECT parameter is not specified
406 Not Acceptable	If the request specifies Accept in the request header that does not include text/plain, text/* or */*
404 Not Found	If either ■ the attribute specified is not valid or ■ the object (symbol) is not valid.

Status	Notes
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

GET /solutions/data

This endpoint allows you to retrieve data. The formats supported are text/csv and application/json.

GET /solutions/data{query}

Use this endpoint with a GET request to retrieve data using an ad hoc query specification.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' and 'View Data' authorizations.

Request Headers:

- LongviewWebSID (string): Specifies the Longview session identifier that grants access to the API.
- Accept (string): Specifies the acceptable response format and must be either text/csv or application/json.
- LVJSONFormat (string): Specifies the JSON format to use when Accept is application/json and must be either CONCISE or VERBOSE. If not specified data will be returned in the CONCISE format.

Request Parameters:

query (string): contains the query parameters for the data request. Query parameters are required to specify the data to retrieve and take the form:

```
{dimension}={symbol}
```

where:

- dimension is the name of a dimension in the system.
- symbol is the name of a symbol in the specified dimension.

A query parameter must be provided for each dimension in the system.

Additional query parameters may be provided as follows:

- dataOption: specifies the scope of data to retrieve, and must be one of these values:
  - STANDARDALL (default) – retrieves all parent and leaf data in the area specified.
  - STANDARDLEADONLY – retrieves leaf data in the area specified.
  - LEAFDATA – retrieve all leaf data in the area specified, excluding any calculated data.
- unadjusted (string): specifies the type of data to retrieve and must be one of these values:
  - false (default) – retrieve adjusted data.
  - true – retrieve unadjusted data.
- schedule: specifies the name of a schedule from which to retrieve data.
- JSONFormat: specifies the JSON format to return when the Accept header is application/json. This query parameter supersedes the LVJSONFormat request header and must be one of these values:
  - CONCISE (default) – retrieve data as JSON in a concise format.
  - VERBOSE – retrieve data as JSON in a verbose format.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/data?ACCOUNTS=Account_
1&TIMEPERIODS=Period_1&ENTITIES=Entity_1&dataOption=LEAFDATA
```

Return Type: Text/csv

```
Accounts,Time Periods,Entities,value
Account_1,Period_1,Entity_1, 500.000000000
```

Return Type: Application/json (VERBOSE)

The returned data includes an array of objects with one object for each data point returned:

```
[
  {
    "F1": "Account_1",
    "F2": "Period_1",
    "F3": "Entity_1",
    "F4": 500
  }
```

```
]
```

Return Type: Application/json (CONCISE)

The returned data includes an object with one metadata property describing the data and an array of data records, with each record being an array of values:

```
{
  "metadata": ["F1", "F2", "F3", "F4"],
  "data": [
    ["Account_1", "Period_1", "Entity_1", 500]
  ]
}
```

Restrictions

The following restrictions apply to this request:

- Each dimension parameter specifies a single symbol name.
- Hierarchical specifications are not supported, and all queries are executed as #99.
- When a schedule is specified all symbols in each dimension of the schedule are included in the query

GET /solutions/data/{name}{query}

Use this endpoint with a GET request to retrieve data using a pre-defined query specification.

The request requires you to provide the name of the API configuration file that contains the data spec definition and export spec definition that define how the data will be exported.

Available templates:

- Data GET – Single Value
- Data GET – Multiple Values

For more information on the Data GET templates, see the Creating data get APIs in the Longview Designer guide.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' and 'View Data' authorizations.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Accept (string): specifies the acceptable response format and must be either text/csv or application/json.

LVJSONFormat (string): specifies the JSON format to use when Accept is application/json and must be either CONCISE or VERBOSE. If not specified data will be returned in the CONCISE format.

Request Parameters:

- name (string): is the name of the API configuration file created using an API Data GET templates.

  For more information on the Data GET templates, see the Creating data get APIs in the Longview Designer Guide.

- query (string, optional): contains the query parameters for the data request. Query parameters are required to specify any variables contained in the API configuration and take the form:

  ```
  {variableName}={value}
  ```

  where:

  - variableName is the name of a variable defined in the API configuration.
  - value is the value to use for the variable.

Additional query parameters may be provided as follows:

- JSONFormat: specifies the JSON format to return when the Accept header is application/json. This query parameter supersedes the LVJSONFormat request header and must be one of these values:
  - CONCISE (default) – retrieve data as JSON in a concise format.
  - VERBOSE – retrieve data as JSON in a verbose format.

URI Example

Pre-defined query has a variable named TIMEPERIODS defined.

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/data/TrialBalance?TIMEPERI
ODS=Period_1
```

Return Type: Text/csv

The retuned data includes a header row followed by a row for each data point returned:

```
Accounts,Time Periods,Entities,value
Account_1,Period_1,Entity_1, 500.000000000
```

Return Type: Application/json (VERBOSE)

The returned data includes an array of objects with one object for each data point returned:

```
[
  {
    "Field1": "Account_1",
    "Field2": "Period_1",
    "Field3": "Entity_1",
    "Field4": 500
  }
]
```

Where the property names are the field names, as defined in the pre-defined query.

Return Type: Application/json (CONCISE)

The returned data includes an object with one metadata property describing the data and an array of data records, with each record being an array of values:

```
{
  "metadata": ["Field1", "Field2", "Field3", "Field4"],
  "data": [
    ["Account_1", "Period_1", "Entity_1", 500]
  ]
}
```

Where the metadata consists of the field names, as defined in the pre-defined query, and the data includes the related symbol name for each dimension and the corresponding value.

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain the data retrieved in media type text/csv. Response header LVStatus: ok
400 Bad Request	Caused by the following: ■ Dimension not specified in query parameters. ■ Symbol specified for a dimension is not valid. ■ Invalid optional parameter value

Status	Notes
406 Not Acceptable	If the request specifies Accept in the request header that does not include text/csv or application/json.
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

Additional Response Headers

Additional response headers are included in the response.

Header	Use
Record-Count	The number of records in the response body, including the header.

PUT solutions/data

This endpoint allows you to update data. The formats supported are text/csv and application/json.

PUT /solutions/data{query}

Use this endpoint with a PUT request to update data using an ad hoc query specification.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' and 'Modify Data' authorizations.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Content-Type (string): specifies the format of the data provided with the request and must be either text/csv or application/json.

Request Parameters:

- query (string): contains the query parameters for the data request. Query parameters are required to specify the data to retrieve and take the form:

  ```
  {dimension}={symbol}
  ```

  where:

  - dimension is the name of a dimension in the system.
  - symbol is the name of a symbol in the specified dimension.

  A query parameter must be provided for each dimension in the system.

Additional query parameters may be provided as follows:

- schedule: used to update data for a specific schedule.
- headerRecords: to indicate the number of header records in the content to be ignored. Applies only when Content-Type is text/csv.
- footerRecords: to indicate the number of footer records in the content to be ignored. Applies only when Content-Type is text/csv.

URI Example

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/data?ACCOUNTS=Account_
1&TIMEPERIODS=Period_1&ENTITIES=Entity_1
```

Content-Type: Text/csv

```
Accounts,Time Periods,Entities,value
Account_1,Period_1,Entity_1, 500.000000000
```

Content-Type: Application/json (VERBOSE)

The content must include an array of objects with one object for each data point returned:

```
[
  {
    "account": "Account_1",
    "timeperiod": "Period_1",
    "entity": "Entity_1",
    "value": 500
  }
]
```

Where each object includes these properties:

- Each dimension name in lower case, with the related symbol name as the value.
- A property named value, with the corresponding value.

Content-Type: Application/json (CONCISE)

The content must include an object with one metadata property describing the data and an array of data records, with each record being an array of values:

```
{
  “metadata”: [“account”, “timeperiod”, “entity”, “value”],
  “data”: [
    [“Account_1”, “Period_1”, “Entity_1”, 500]
  ]
}
```

Where the metadata consists of each dimension name in lower case and value and the data includes the related symbol name for each dimension and the corresponding value.

Content-Type: Application/json (oData)

```
{
  "@odata.context":"url",
  "value": [
    {
      “account”: “Account_1”,
      “timeperiod”: “Period_1”,
      “entity”: “Entity_1”,
      “value”: 500
    }
  ]
}
```

Where each object includes these properties:

- Each dimension name in lower case, with the related symbol name as the value.
- A property named value, with the corresponding value.

Restrictions

The following restrictions apply to this request:

- Each dimension parameter specifies a single symbol name.
- Hierarchical specifications are not supported, and all queries are executed as #99.
- Import is always done FULL and not INCREMENTAL.
- When a schedule is not specified, BASE data is submitted.

- If a schedule is specified, it must be the last field in the body before the data value.
- Max Errors is set to 0.

PUT /solutions/data/{name}{query}

Use this endpoint with a PUT request to update data using a pre-defined specification.

Available templates:

- Data PUT – Single Value
- Data PUT – Multiple Values

For more information on the Data PUT templates, see the Creating data put APIs in the Longview Designer guide.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' and 'Modify Data' authorizations.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Content-Type (string): specifies the format of the data provided with the request and must be either text/csv or application/json.

Request Parameters:

query (string, optional): contains the query parameters for the data request. Query parameters are required to specify any variables contained in the API configuration and take the form:

```
{variableName}={value}
```

where:

- variableName is the name of a variable defined in the API configuration.
- value is the value to use for the variable.

URI Example

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/data/LoadTrialBalance
```

Content-Type: Text/csv

```
Accounts,Time Periods,Entities,value
Account_1,Period_1,Entity_1, 500.000000000
```

Content-Type: Application/json (VERBOSE)

The returned data includes an array of objects with one object for each data point returned:

```
[
  {
    "Field1": "Account_1",
    "Field2": "Period_1",
    "Field3": "Entity_1",
    "Field4": 500
  }
]
```

Where the property names are the field names, as defined in the pre-defined specification, and the data includes the related symbol name for each dimension and the corresponding value.

Content-Type: Application/json (CONCISE)

The returned data includes an object with one metadata property describing the data and an array of data records, with each record being an array of values:

```
{
  "metadata": ["Field1", "Field2", "Field3", "Field4"],
  "data": [
    ["Account_1", "Period_1", "Entity_1", 500]
  ]
}
```

Where the metadata consists of the field names, as defined in the pre-defined query, and the data includes the related symbol name for each dimension and the corresponding value.

Content-Type: Application/json (oData)

```
{
  "@odata.context":"url",
  "value": [
    {
```

```
      "Field1": "Account_1",
      "Field2": "Period_1",
      "Field3": "Entity_1",
      "Field4": 500
    }
  ]
}
```

Where the property names are the field names, as defined in the pre-defined specification, and the data includes the related symbol name for each dimension and the corresponding value.

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain information about the import: ■ Records-Read ■ Records-Filtered ■ Records-Rejected Response header LVStatus: ok
400 Bad Request	Caused by the following: ■ Dimension not specified in query parameters. ■ Symbol specified for a dimension is not valid. ■ Invalid optional parameter value
404 Not Found	If the request contains invalid symbols.
415 Unsupported Media Type	PUT requires a text/csv or application/json, if any other file type is used this will be the error that returns.
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

Additional Response Headers

Additional response headers are included in the response.

Header	Use
Records-Read	The number of records that were read on import

Header	Use
Records-Filtered	The number of records that were filtered out from being imported
Records-Rejected	The number of invalid records that were rejected from being imported

GET /solutions/hierarchies

This endpoint will return a list of links to the hierarchies API services provided in JSON format.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies
```

Return Type: Application/json

```
{
    "links": {
        "ACCOUNTS": "http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS",
        "TIMEPERIODS": "http://localhost/cgi-bin/
Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/TIMPERIODS",
}
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain list dimensions that are available to query. Response header LVStatus: ok
406 Not Acceptable	If the request specifies Accept in the request header that does not include application/json
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

GET /solutions/hierarchies/{dimension}

This endpoint will return a list of links to the hierarchies API services for the specified dimension provided in JSON format.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Request Parameters:

dimension (string): the name of the dimension.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS
```

Return Type: Application/json

```
{
    "links": {
        " ACCOUNTS_Default ": "http://localhost/cgi-
bin/Longvoew/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS/
ACCOUNTS_Default ",
        "TRIALBAL": "http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS/TRIAL
BAL",
}
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain list of root symbols within the dimension provided that are available to query. Response header LVStatus: ok
404 Not Found	If the dimension specified is not valid.
406 Not Acceptable	If the request specifies Accept in the request header that does not include application/json

Status	Notes
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

GET /solutions/hierarchies/{dimension}/{root}

This endpoint will return the hierarchy of the specified root symbol in plain text, in the same format as the ExportHierarchy command in Application Framework.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework' authorization.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Request Parameters:

dimension (string): the name of the dimension.

root (string): the name of the root symbol in the dimension.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS/TRIAL
BAL
```

Return Type: Text/plain

```
{TRIALBAL{{Trial balance{Standard{Dynamic{Manual{Neither
TRIALBAL{BALSHEET{0{Balance sheet{CarryForward{Dynamic{Manual{Neither{1
BALSHEET {TA0000{+{Assets{CarryForward{Dynamic{Manual{ Neither {100
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will contain the exported root structure hierarchy in text plain format. Response header LVStatus: ok
400 Bad Request	If the Root symbol specified is not actually a root level in the dimension specified

Status	Notes
404 Not Found	Caused by the following: ■ Dimension is not valid. ■ Root Symbol is not a valid symbol. ■ Root symbol does not exist in the dimension specified
406 Not Acceptable	If the request specifies Accept in the request header that does not include text/plain, text/* or */*
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error

PUT /solutions/hierarchies/{dimension}/{root}{query}

Use this endpoint with a PUT request to update a specific symbol hierarchy.

Authorizations Required:

To execute this API the user must have the 'Connect to Application Framework', 'Symbols Create', 'Symbols Delete', and 'Symbols Modify' authorizations.

Request Headers:

LongviewWebSID (string): specifies the Longview session identifier that grants access to the API.

Request Parameters:

- dimension (string): the name of the dimension.
- root (string): the name of the root symbol in the dimension.
- query (string, optional): contains the query parameters for the hierarchy request. Query parameters are required to specify any variables contained in the API configuration and take the form:

  The following query parameters are supported:

  - Execute: specifies the execution mode of the PUT request and must be one of these values:
    - SIMULATE (default) – Generates a plain text output file for each type of command that would be executed if the Execute query parameter value was specified as True.
    - True - Implements the instructions and commits the designated changes to the destination hierarchy.

- Delete: indicates what action should be taken with any symbols existing in the target symbol hierarchy which are not specified in the request body.
  - (default) The symbols are not actually deleted; they are re-assigned to the {dimension}_ DELETED hierarchy structure.
  - True – Signifies that the symbol will be completely removed from the database.

URI Example:

```
http://localhost/cgi-
bin/Longview/lvweb.cgi/Longview/api/app/solutions/hierarchies/ACCOUNTS/TRIAL
BAL?Execute=True
```

Content-Type: Text/plain

```
{TRIALBAL{{Trial balance{Standard{Dynamic{Manual{Neither
TRIALBAL{BALSHEET{0{Balance sheet{CarryForward{Dynamic{Manual{Neither{1
BALSHEET {TA0000{+{Assets{CarryForward{Dynamic{Manual{ Neither {100
```

Possible Responses

Status	Notes
200 OK (OK)	The response body will be empty. Response header LVStatus: ok
400 Bad Request	If the Root symbol specified is not actually a root level in the dimension specified
404 Not Found	Caused by the following: ▪ Dimension is not valid. ▪ Root Symbol is not a valid symbol. ▪ Root symbol does not exist in the dimension specified
405 Method Not Allowed	PUT is not a supported method for the URI request.
415 Unsupported Media Type	PUT requires a text/plain, text/* or */* content type. If any other file type is used this will be the error that returns
200 OK (ERROR)	If any error occurs during the execution of the request. The response body will contain text/plain with the execution error that occurred. Response header LVStatus: error