CIQ Web Service
Estimates Consensus
Specification 1.0

Date Created: 11.05.2008
Last Updated: 11.20.2008

Business Owner:	Jay Zachter, Michael Yusko
Technology Owner:	William Murphy (CIQ),
Business Analysts	Shawn West
Version	1.0

Application Framework

The primary technology for this solution is XML Web Services (SOAP). Capital IQ hosts an API that responds to XML requests according to this API, and returns XML structured data in response. These XML requests are encrypted via the standard HTTPS protocol.

A secondary technology for this solution is the integration of CIQ DataFeeds on client database tier. This allows for reduced network traffic for common items that change infrequently.

Capital IQ hosts this data on Windows-based servers, powered by Microsoft SQL Server in an active-passive failover cluster configuration. Data is stored in multiple fully redundant EMC Storage Area Networks (SANs). The servers that run the platform are hosted at Quality Technology Services with a disaster recovery site at XO. At all levels, these environments are redundant, fault tolerant, and backed up to industry standards.

Web Services Description Language (WSDL) documents describe the detailed Services & Ports (Function Calls) available in this specification. See http://www.w3.org/TR/wsdl for more on WSDL.

Please note that all Web Service and WSDL URLs in this document are subject to change based on changing infrastructure requirements. CIQ will provide sufficient advanced notice to the client before changing any URL, hostname, IP address, etc. It is recommended that these URLs be configurable (via config files, etc.) on the client application so that changes can be handled with minimal user downtime. CIQ monitors activity on Production systems and may shut down improper-use processes or user accounts as required to preserve overall system health.

All Web Services requests and responses in this solution are encoded in the UTF-8 character set (http://en.wikipedia.org/wiki/UTF-8). Some string data in this solution is expected to only contain Windows-1252 characters (http://en.wikipedia.org/wiki/Windows-1252); these are labeled with "(W1252)" in this document. Other string data in this solution allows full UTF-8 characters; these are labeled with "(UTF-8)" in this document. Email addresses (labeled "(email)" in this document) and website URLs (labeled "(URL)" in this document) have more limited valid character sets. See http://en.wikipedia.org/wiki/Email_address and http://en.wikipedia.org/wiki/URL for more information.

All the web services have a WSDL definition that external developers will code against and pull in data that is served from the same Capital IQ data repository as our web platform. For a full menu of our Web Services and implementation documentation, please contact your account manager.

Web Service Versioning

Versioning Web Services: Over time, Capital IQ may need to extend the tags or datasets supported by our Web services. As a results we have created a URL based versioning solution provides a scalable framework for the future. Versioning provides a way for to accommodate these enhancements in a graceful manner.

Recommendation: Capital IQ recommends that all users upgrade to version 1.0 if they are using legacy services, to conform to the new URL formats.

How versioning works: Please note in the example below *<ServiceName.asmx>* is replaced with the name of the service and is used for illustration purposes only.

1. Web Service changes are captured as a new version of the file in a new directory.

   a. **Version 1** - https://api.capitaliq.com/ciqdotnet/api/1.0/*<ServiceName.asmx>* - Represents the first release of the service
   b. **Version 2** - https://api.capitaliq.com/ciqdotnet/api/2.0/*<ServiceName.asmx>* - Represents the second release and breaking change or significant enhancement.
   c. Clients have the ability to transition to the new version of the service or stay on the original version until they can transition older code.

2. Latest version of the Service will be located at the following URL. https://api.capitaliq.com/ciqdotnet/api/current/*<ServiceName.asmx>*. Using the example in section i above https://api.capitaliq.com/ciqdotnet/api/2.0/*<ServiceName.asmx>* would be in its own directory and referenced in the current directory.

Service Changes

Estimates

Service	Version	Comments
URL	Current	https://api.capitaliq.com/ciqdotnet/api/current/Estimates.asmx?WSDL
URL	1.0	https://api.capitaliq.com/CIQDotNet/api/1.0/Estimates.asmx?WSDL

Release	Version	Comments
11/2008	1.0	First Release of Estimates Consensus
11/2008	1.0	First Release of Estimates Detail

Scenario

Capital IQ (CIQ) provides a Web Service solution that enables a client application to retrieve Estimate Consensus and Consensus Detail. The input criteria is data based on start date, end date, period dates, data items and trading items. There are four functions available that support data retrieval base on both Relative Period and Absolute Date inputs.

UserObjectID and log-in credentials are required to use this service. The parameter is required to track usage statistics this will be provided was part of the entitlement process. It is assumed that the proper “capabilities”, or data access rights, are managed by the client application for Users. No entitlements are evaluated for consensus estimates data.

GetEstimateConsensus

GetEstimateConsensus Ports (Functions):

Relative: For retrieving period estimates relative to a current period the functions accepts StartPeriodOffset/EndPeriodOffset as inputs

```
EstimateConsensus GetEstimateConsensus(
       Integer companyId(),Integer tradingItemId(), Integer startPeriodOffset, Integer endPeriodOffset,
       Integer periodTypeId(), Integer dataItemId(),DateTime startAsOfDate, Datetime endAsofDate,
       Integer currencyID, Integer currencyConversionMethodId, Integer userObjectID);
```

Absolute: For retrieving estimates with in a specific date range the functions accepts StartPeriodDate/EndPeriodDate as inputs

```
EstimateConsensus GetEstimateConsensus(
       Integer companyId(),Integer tradingItemId(),DateTime startPeriodDate, DateTime endPeriodDate,
       Integer periodTypeId(), Integer dataItemId(),DateTime startAsOfDate, Datetime endAsofDate,
       Integer currencyID, Integer currencyConversionMethodId, Integer userObjectID);
```

Parameters

1. `Array of Integer companyId()` – Each item of the array is a single CompanyID corresponding to a company that has consensus estimates data. At least one valid companyId is required. **Input** [Required/Optional], [Multiple].

2. `Array of Integer tradingItemId()` – Each item of the array is a single TradingItemId corresponding to a security listed on a particular exchange that has consensus estimates data. Supply an empty array if estimates should be returned for all TradingItems for each Company in `CompanyId()`. **Input** [Required/Optional], [Multiple].

    a. **Client Note:** [Required/Optional] Either of these two are required for input in this service.

        i. If `tradingItemID()` is given as an empty array, this function returns estimates data for every trading item that has consensus estimates data for the given CompanyIDs.

        ii. Use the `tradingItemID()` parameter to return data for just the individual Trading Items issued by Companies.

    b. **Client Note:** Only data points with non-NULL data are returned by this function. “Empty” periods and/or DataItems are not returned. To prevent performance problems, the following formula is used to calculate the maximum amount of data that can be returned by a single call:

        i. **For Consensus**:
        (periods) **x** (trading items OR 1.5 **x** companies) **x** (days in as of date range / 10) **x** (data items)
        ii. The complete Formula break down can be found in the appendix at the end of this document.

3. `Datetime startPeriodDate` – The oldest period for which the estimate/actual date should be returned. To retrieve all historical estimate data enter the historical date of 1/1/1950. **Input** [Required], [Single].

4. `Datetime endPeriodDate` – The period for which estimates/actual data should be retuned that is furthest in the future. **Input** [Required], [Single].

    a. **Client Note:** The Absolute interface uses [`startPeriodDate/endPeriodDate`] are used when retrieving estimates based on specific time periods.

5. `Integer startPeriodOffSet` – The oldest relative period for which estimates/actual data should be returned. Supply a value of 0 to return the current period of consensus estimates data. **Input** [Required], [Single].

6. `Integer endPeriodOffSet` – The relative period for which estimates/actual data should be returned that is farthest in the future. **Input** [Required], [Single].

    a. **Client Note** The Relative interface uses [`startPeriodOffSet/endPeriodOffSet`] when retrieving estimates based on fiscal quarters relative to current quarter. i.e. 0 = Today, Negative (-1) would be one quarter going backwards in history from the current fiscal quarter, Positive (+4) goes forward four fiscal quarters into the future.

7. `Integer StartAsOfDate` – Return consensus estimates calculated as of this start date and time (EST/EDT). For “latest”, specify a future date, such as 12:00AM tomorrow or leave the input as BLANK/[NULL]. Specifying 12:00AM today will exclude consensus estimates. E.g. 1/1/2007 to 1/1/2008. This will include estimate which was given in 12/12/2006 which expired in 1/12/2007 **Input** [Optional], [Single].

8. `Integer EndAsOfDate` – Return consensus estimates calculated as of this end date and time (EST/EDT). **Input** [Optional], [Single]

    a. **Client Note** Both Start and End Date can be used to determine the range when an estimate was calculated. So any estimate calculated for a given period within this range would come.

9. `Array of Integer periodTypeID()` – Each item in the array identifies the periodicity of the estimates data to be returned. Supply an empty array if estimates should be returned for all Period Types.
Input [Optional], [Multiple] Possible values:

ID	Name
1	Fiscal Year
2	Quarterly
7	Calendar Year
8	Non-periodic
10	Semi-annual
12	NTM

10. `Array of Integer dataItemID` ()– Limits the data points that are returned to only the dataItemIDs in the array. At least one valid `dataItemID` is required. **Input** [Required], [Multiple].

11. `Integer` **`currencyID`** – The currency in which to display the financial data. Monetary data will be converted to this currency, if collected in a different currency. **Default**: [0] Reported Currency, **Input** [Single], [Optional] possible values:

ID	Name
0	Reported Currency
55	British Pound
27	Canadian Dollar
50	European Union Euro
64	Hong Kong Dollar
79	Japanese Yen
160	US Dollar

12. `Integer` **`currencyConversionMethod`** – If the currency is not the reported currency, this parameter controls how the data should be currency converted. **Default**: [0] Historical , [Single], [Optional] possible values:

ID	Name
0	Historical as of Period End Date
1	Today's Spot Rate
2	Historical as of Estimate Date

13. `Integer userObjectId` – The User for whom usage statistics are tracked. **Input** [Required], [Single].

    a. **Client Note:** Capital IQ will provide this Id for use in the client application as part of the sign on process in addition to Login/Authentication credentials An exception will be thrown if the userObjectId is not provided as input.

Returns: Returns an array of EstimateConsensus, a container object with header info for a set of detailed estimates data. Each EstimateConsensus has a companyId, tradingItemId an array of Periods that the estimates are for, and an array of EstimateConsensusDataValues.

EstimateConsensus
Elements:

a. **EstimateConsensus**
Attributes:
   i. `Integer CompanyId` – CIQ Company ID of the company the set of consensus estimates data is for.
   ii. `Integer TradingItemId` – CIQ Trading Item ID of the security listing the set of consensus estimates data is for.

iii. Periods () – List of Periods (optional, multiple)

1. `Integer PeriodTypeID` – See parameters for explanation.
2. `Integer FiscalYear` – Fiscal year that the set of estimates is for.
3. `Integer FiscalQuarter` – Fiscal quarter (or semi-annual period) that the set of estimates is for.
4. `Integer CalendarYear` – Calendar year that the set of estimates is for.
5. `Integer CalendarQuarter` – Calendar quarter (or semi-annual period) that the set of estimates is for.
6. `Integer PeriodOffset` – The offset value of the period that the estimate is for.
7. `DateTime PeriodEndDate` – The date and time at which the estimate was calculated.

iv. `Array of Integer` EstimateConsensusDataValue (optional, multiple)

1. `Integer DataItemID` – Relates to a specific Estimate Datapoint. To access a full list of estimates data items via Capital IQ Reference data web service Id #25.
2. `String DataItemValue` – (0-255) The estimate value. (W1252)
3. `Integer` **`ScaleId`** – Possible values

ID	Name
1	Actual
2	Thousands
3	Millions
4	Billions

4. `Integer` **`UnitTypeId`** – Possible values

ID	Name
1	Currency
2	Ratio
3	Percentage
4	Date
5	Text
6	Enumeration
7	Boolean
8	Other

5. `Integer` **`CurrencyID`** – The currency in which to display the financial data. Monetary data will be converted to this currency, if collected in a different currency. To access a full list of available currencies via Capital IQ Reference data web service Id #26.

ID	Name
0	Reported Currency
55	British Pound
27	Canadian Dollar
79	Japanese Yen
160	US Dollar

6. `Date` **`EstimateStartDate`** – The earliest date that the estimate value is valid.

7. `Date` **`EstimateExpireDate`** – The date on which Estimate is set to expire. For currently valid estimates, set to year 2079.

8. `DateTime` **`CurrencyConversionDate`** – The date the currency rate was sourced.

9. `Double` **`CurrencyConversionRate`** – The numeric amount of the conversion

10. `Integer` **`CurrencyConversionMethod`** – If the currency is not the reported currency, this parameter controls how the data should be currency converted.

ID	Name
0	Historical as of Period End Date
1	Today's Spot Rate
2	Historical as of Estimate Date

11. `String` **`Footnote`** – (0-5000) Reserved for future use (W1252)

Exceptions

1. An exception will be thrown if the request cannot be authenticated via a session cookie.
2. An exception will be thrown if any parameter is out of range.
3. An exception will be thrown if any required parameter is missing.
4. An exception will be thrown if too much data would be returned, as determined by the input parameter formula.
5. An exception will be thrown if `companyID()` does not contain any valid CompanyIDs.
6. An exception will be thrown if `dataItemID()` does not contain any valid DataItemIDs.
7. An exception will be thrown if client application does not have the correct Capability assigned.
8. An exception will be thrown EstimatePeriodTypeId: ___ is not a valid value
9. An exception will be thrown endPeriodDate may not be null or earlier than startPeriodDate
10. An exception will be thrown if Too much data requested
11. An exception will be thrown if userObjectID is not supplied.

Appendices

1. **Windows-1252 A character encoding of the Latin alphabet**, used by default in the legacy components of Microsoft Windows in English and some other Western languages. The encoding is a superset of ISO 8859-1, but differs from the IANA's ISO-8859-1 by using displayable characters rather than control characters in the 0x80 to 0x9F range. It is known to Windows by the code page number 1252, and by the IANA-approved name "windows-1252". This code page also contains all the printable characters that are in ISO 8859-15 (though some are mapped to different code points).

2. **Extensible Markup Language (XML)** is a general-purpose markup language. Its primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet.

3. **dateTime [Definition:]** values may be viewed as objects with integer-valued year, month, day, hour and minute properties, a decimal-valued second property, and a Boolean timezoned property. Each such object also has one decimal-valued method or computed property, timeOnTimeline, whose value is always a decimal number; the values are dimensioned in seconds, the integer 0 is 0001-01-01T00:00:00 and the value of timeOnTimeline for other dateTime values is computed using the Gregorian algorithm as modified for leap-seconds. The timeOnTimeline values form two related "timelines", one for timezoned values and one for non-timezoned values. Each timeline is a copy of the ·value space· of decimal, with integers given units of seconds.

   The ·value space· of dateTime is closely related to the dates and times described in ISO 8601. For clarity, the text above specifies a particular origin point for the timeline. It should be noted, however, that schema processors need not expose the timeOnTimeline value to schema users, and there is no requirement that a timeline-based implementation use the particular origin described here in its internal representation. Other interpretations of the ·value space· which lead to the same results (i.e., are isomorphic) are of course acceptable.

   All timezoned times are Coordinated Universal Time (UTC, sometimes called "Greenwich Mean Time"). Other timezones indicated in lexical representations are converted to UTC during conversion of literals to values. "Local" or untimezoned times are presumed to be the time in the timezone of some unspecified locality as prescribed by the appropriate legal authority; currently there are no legally prescribed timezones which are durations whose magnitude is greater than 14 hours. The value of each numeric-valued property (other than timeOnTimeline) is limited to the maximum value within the interval determined by the next-higher property. For example, the day value can never be 32, and cannot even be 29 for month 02 and year 2002 (February 2002). For more details http://www.w3.org/TR/xmlschema-2/#dateTime