# CIQ Technology WebServices

# Specification: Professional Detail

# Version 2.0

**Date Created: 9.12.2007**
**Last Updated: 11.20.2008**

| | |
|---|---|
| **Business Owner:** | Jay Zachter, Michael Yusko |
| **Technology Owner:** | William Murphy (CIQ), |
| **Business Analysts** | Shawn West |
| **Version** | 2.0 |

## 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

**Professional Details**

| Service | Version | Comments |
|---|---|---|
| URL | Current | https://api.capitaliq.com/ciqdotnet/api/Current/ProfessionalsDetail.asmx?WSDL |
| URL | 2.0 | https://api.capitaliq.com/ciqdotnet/api/2.0/ProfessionalsDetail.asmx?WSDL |
| URL | 1.0 | https://api.capitaliq.com/ciqdotnet/api/1.0/ProfessionalsDetail.asmx?WSDL |
| URL | Legacy | https://api.capitaliq.com/ciqdotnet/Company/ProfessionalsDetail.asmx?WSDL |

| Release | Version | Comments |
|---|---|---|
| 11/2008 | 2.0/Current | Fixed <anyType> in XML output replaced with <ArrayOfPerson> Type in XML output |
| 9/2008 | 1.0 | Updated to conform to Capital IQ new versioning criteria |

## Scenario

Capital IQ (CIQ) currently provides company information and financial data though various feeds via FTP. A number of clients are starting to utilize Web Services as a way to exchange data as an alternative to data feeds. Web Services allows our clients to access this content in a more dynamic and compact form so they can integrate it to their portals or similar internal data management systems.

This specification covers how to retrieve data using GetProfessionalDetails to retrieve granular information for a list of professionals based on personID or ProfessionalId. This function provides name, address, boards, education, and historical compensation. For GetProfessionalDetail the client application can retrieve the personID or ProfessionalId via our datafeed or by running the GetProfessional function.

## Professionals GetProfessionalsDetail

### GetProfessionalsDetails

This function allows the client application to retrieve information about professionals within a company. The function will return board memberships, affiliations, company address, year born, title and background. The data in this section is collected and owned by CIQ. Therefore no 3rd party licenses are required but access would require a separate license agreement to receive the GetProfessionals Web Service content.

### Professionals Ports (Functions):

```
Person () GetProfessionalsDetail (PersonId(), ProfessionalId());
```

**Comments:**

This function returns a list of Person that consist of ProfesionalDetails objects that contain company, board, compensation professional type, status and function. Non-NULL data is not returned by this function

**Input Parameters:**

1. `An Array of Integer` **`PersonId()`** – A unique identifier for every person in the Capital IQ data set. **Input** [Required/Optional], [Multiple].

2. `An Array of Integer` **`ProfessionalId()`** – This is a unique identifier for every person in the Capital IQ data set. **Input** [Required/Optional], [Multiple]

**Client Note**: [Required/Optional] Logic – Each item is can be Optional under in the following instance: If PersonID is supplied for input then ProfessionalID is optional. If ProfessionalId is supplied then PersonId is optional.

**Client Note**: The PersonID and ProfessionalId is an attribute of the GetProfessionalSummary output or contained in our data feed. These values are required input for this Web Service function.

**Client Note**: To prevent performance degradation only **1000 PersonId** or 1000 **ProfessionalId** may be requested via this Web Service.

**Return Output**

```
Person (ProfessionalDetail());
```

**Comments:** This function return will return an array of Person Objects each of which contains an array of ProfessionalDetail objects.

**Person** (multiple)
**Attributes:**

a. Integer **PersonId** – This is a unique identifier for every person in the Capital IQ data set.
b. String **Prefix** – (1-50) such as “Mr.”, “Ms.”, etc. (W1252)
c. String **Salutation** – For some individuals, the name or nickname they prefer to be called.
d. String **FirstName** – (1-50) First Name. (W1252)
e. String **MiddleName** – (0-50) Middle name (W1252)
f. String **LastName** – (1-50) Last name (W1252)
g. String **Suffix** – (0-50) Tag, such as “Jr.”, “III”, “Ph.D.”, etc. (W1252)
h. Date **YearBorn –** Year the professional was born
i. String **PersonBiography –** (0-8000) A paragraph describing the experience of the person
j. String **EmailAddress** – (6-100) email address for this professional (W1252)
k. Boolean **DeceasedFlag** – is the professional deceased.

l. **ProfessionalDetail (multiple)**
**Attributes**:

i. Integer **ProfessionalId** – Unique identifier for a "professional". A professional is the instance of a person being associated to a company (or educational institution, etc.)
ii. Integer **CompanyId** – The unique identifier of the Primary CIQ Company the Professional relates to.
iii. String **CompanyName** – (0-200) a name that identifies the primary Company. (W1252)
iv. Integer **ProfessionalStatusId** – Status in the organization

| ID | Name |
|---|---|
| 0 | Not A Professional |
| 1 | Current |
| 2 | Prior |

v. Integer **BoardStatusId –** Status in the Board

| ID | Name |
|---|---|
| 0 | Not A Board Member |
| 1 | Current |
| 2 | Prior |

vi. Boolean **AlumniFlag** – Is the person an Alumni True/False/[Null].
vii. String **Title –** (0-500) Professional title of the Person. Example: “Senior Director”.
viii. Integer **ProRank –** Returns an integer that provides a sort order for the professional.
ix. Integer **BoardRank** – The "rank" of this individual on the board. Basically, the type of board members are ranked (chairmen = 1) and then ordered.
x. String **AddressLine1** – (0-200) 1st line of the primary office address for this ProfessionalId. (W1252)
xi. String **AddressLine2** – (0-200) 2nd line of the primary office address for this ProfessionalId. (W1252)
xii. String **AddressLine3** – (0-200) 3rd line of the primary office address for this PersonId. (W1252)
xiii. String **AddressLine4** – (0-200) 4th line of the primary office address for this PersonId. (W1252)
xiv. String **CityName** – (0-200) String Value for the City of the primary office address for this personID.
xv. Integer **StateId** – State or province of primary office address for each ProfessionalId.

xvi. String **StateName** – (0-50) String value for the State or province of primary office address for each personID. (W1252)
xvii. String **ZipCode** – (0-50) Zip or postal code of the primary office address for this PersonId. (W1252)
xviii. Integer **CountryId** – Country of the primary office address for this PersonId.
xix. String **CountryName** – (0-50) String value for the Country of the primary office address for this personID.
xx. String **MainFaxNumber** – (0-30) Primary fax number for this PersonId (W1252)
xxi. String **MainPhoneNumber -** (0-30) Primary phone number for this PersonId. (W1252)

xxii. **ProfessionalFunction()** (optional, multiple) **–** attributes:

1. Integer **ProfunctionId -** This is the unique identifier of a given proFunction. A proFunction is a standardized way to describe titles since titles can be different across firms.
2. String **ProfunctionName –** Name that identifies ProfunctionId
3. Date **StartDate** – Start date for this professional
4. Date **EndDate** - End date for this professional

xxiii. **Compensation()** (optional, multiple) **–** Element:

1. Integer **CurrencyId -** This is the unique identifier of a given currency

| ID | Name |
|---|---|
| 0 | Reported Currency |
| 9 | Australian Dollar |
| 55 | British Pound |
| 160 | US Dollar |

2. String **CurrencyName –** (0-50) Description of the currency associated with CurrencyId (W1252)
3. Integer **CompensationTypeId -** Type of the compensation (options, cash, etc.)
4. String **CompensationTypeName –** (0-100) The description of the compensation type (W1252)
5. Date **CompensationFiscalYear** – Fiscal year for Compensation
6. DateTimeUTC **CompensationFilingDate** - Date the SEC document describing this compensation was filed
7. Decimal **CompensationValue -** Value for the compensation

### 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.

### Web Services Description Language (WSDL)

Notwithstanding anything to the contrary in this Agreement, Capital IQ reserves the right to change, expand or modify Web Services Definitions and corresponding Web Services Description Language files (WSDL) at any time. Any such modifications will be done in accordance with industry standards that support backwards compatibility with previous WSDL files. If possible, Clients will be notified in advance of any modifications.

**Login**: Using the supplied UserName and Password this is provided by Client Support or your Client Development representative.

## 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