Technological Systems & Services Directorate
Trading Systems Development Department

***A**thens **E**xchange S.A.*

OASIS - IDS
Interface Technical Specification (IOCP)

Version 5.0.0

Athens, April 2017

Warranties & Disclaimers

The exchange endeavours to ensure that the data and other material that contained in this document are correct and complete, but does not accept any responsibility or liability for any error made or omission from it and/or for the accuracy and reliability for the information of this document and/or arising from access to, or the information on this document.

The exchange develops products and services continuously, thus its published information may not be up to date. The exchange makes no representation and disclaims all express and implied warranties of any kind to any third party including warranties as to accuracy, timeliness, completeness or fitness for any particular purpose.

Use of Documents

This material comprises a part of the technical documentation of the Feed Information Center (FIC) – TCP/IP Service Interface project and is disclosed by the exchange only to Data Vendors and Independent Software Vendors (ISV) who intent to exploit this new delivery service of the exchange's Markets Feed.

Revision History		
Issue	**Date**	**Description**
1.00	07/05/2004	First Version
1.002	10/08/2004	**Correction:** In the “Type” column of Table 6-1 the sizes of the *userNameHash*, *userPasswordHash* and *Total Bytes* fields were changed from 17, 17and 44 to 21, 21 and 52 respectively. (paragraph: 6.1.1.1 Client Login ‘CL’)
1.003	14/09/2004	**Correction:** The value of the *newUserNameStr* string, in the User Name Hash Computation example, becomes “**G1E7O2R.G1060.021.0301 00**” instead of “**1G7E2O.R1G60.020.1 30100000000000**”. (section: 5.2 Hash Values Computation) The indices of the “newUserNameStr” array in the User Name Hash Computation algorithm displayed on the figure, should be **[count]** and **[count+1]** instead of **[i+count]** and **[i+count+1]**, respectively. (section: 5.2 Hash Values Computation) **Addition:** The line “If maxL MOD 2 <> 0 then maxL <- maxL + 1” was added inside the User Name Hash Computation algorithm displayed on Figure. (section: 5.2 Hash Values Computation)
1.10	1/11/2005	**Correction:** The version of the Market Data Feed Specification document was changed (section: 6.2 Data Messages, subsection: 1.3.2 Data Feed Services (Functionalities and Message Specifications)) **Correction:** The contact information have changed (section: 1.6 Contact Information) **Correction:** The levels of BBO available are 1 or 5 instead of 1 or 3 (section: 2.5 Filtering) **Addition:** The “data feed source” filtering criterion was added (section: 2.5 Filtering) **Correction:** Lost data feed messages could be recovered without retransmission (section: 3.4 Losing Data Channel) **Addition:** Specifies how the IOCP handles multiple authentication trials from the same client with the same authentication tokens (section: 5.1 Authentication Steps) **Correction:** The meaning of the *lstPackSent* field values sent with a CT message(paragraph: 6.1.1.4 Client Transmission ‘CT’) **Correction:** The user name and password hashes are 21 bytes long instead of 17 (paragraph: 6.1.2.1 Server Login ‘SL’) **Addition:** The “Frequently Asked Questions” chapter was added (chapter: 10 Appendix III: Frequently Asked Questions)
1.101	21/3/2006	**Addition:** The value ‘104’ in the ‘SL’ reply message denotes a valid CL request (paragraph: 6.1.2.1 Server Login ‘SL’)
1.102	18/9/2006	**Addition:** Note on the byte ordering (Big/Little Endian) used by IOCP for multi-byte values (paragraph: 6.1Control Messages) **Addition:** A Question/Answer pair explaining the term Big/Little Endian byte ordering (chapter: 10 Appendix III: Frequently Asked Questions) **Correction:** Deletion of the invalid URL and suggesting a WEB search as an alternative. (section: 5.3 SHA1 References)

		Rephrasing: The explicit version numbers and issue dates of the Data Feed Specifications were replaced by the term “latest”. (section: 6.2 Data Messages) **Addition:** The meaning of the *lstPackSent* field for CT ‘Stop’ requests (paragraph: 6.1.1.4 Client Transmission ‘CT’ and table: Table 6-4) **Addition:** The meaning of the retransmission range fields for CR ‘Stop’ requests, and the valid security type for Index-related (‘I’) retransmissions (paragraph: 6.1.1.5 Client Retransmission ‘CR’ and table: Table 6-5)
2.000	13/12/2006	**Rewriting:** The content and wording of various paragraphs and one appendix has been changed. The business issues, however, covered in these paragraphs remain unaltered (the first paragraph of the Introduction, paragraphs 2.5 Filtering , 6.1.1.4 Client Transmission ‘CT’ and 6.1.2.4 Server Transmission ‘ST’ and Appendix III: Frequently Asked Questions) **Addition:** The new meaning of the instrument type and retransmission range fields used in retransmission requests (paragraphs: 6.1.1.5 Client Retransmission ‘CR’ and 6.1.2.5 Server Retransmission ‘SR’ and table: Table 6-5) **Addition:** The dissemination of index-baselines as a filtering criterion (paragraph : 2.5 Filtering) **Addition:** The updated list of the security-related data feed messages currently supported by the IOCP (Table 6-14)
2.001	29/5/2007	**Modification:** The type of the field in the AC message was changed from **ULONG** to **long** (paragraph 6.1.3.1 Server Authentication Challenge ‘AC’ and table:Table 6-11)
3.000	14/9/2008	**Rewriting:** The content and wording of various paragraphs has been changed mainly to support the new concept of time-sensitive and relaxed data feed Account types (paragraphs : • 2.1 Vendor-Service Communication • 2.2 Vendor-Service Interaction • 2.4 Data Feed Accounts • 2.5 Filtering of Data Feed • 3.2 Opening Data Channel • 4.3 Data Connection Phase • 4.4 Data Flow Controlling Phase • 6.1.1.4 Client Transmission ‘CT’ • 6.1.1.5 Client Retransmission ‘CR’ • 6.1.2.4 Server Transmission ‘ST’ • 6.1.2.5 Server Retransmission ‘SR’ • 6.2 Data Messages • the third and fifth Question/Answer pair in Appendix III: Frequently Asked Questions) **Addition:** Three new Control Messages ‘CS’, ‘SS’, ‘GN’ (paragraphs : 6.1.1.3 Client Channel Status ‘CS’, 6.1.2.3 Server Channel Status ‘SS’, 6.1.3.3 Server General Notification ‘GN’ and Table 9-1)

3.001	7/3/2009	**Rewriting:** The Category M message becomes Projected-Auction/Auction-Open/Projected Close Price Message instead of Projected-Auction/Auction-Open Price Message (paragraph 6.2 Data Messages)
4.000	26/4/2013	**Deletion:** The 'D' and 'E' from the valid data feed types in CT message. 'A' is the ONLY valid date feed type Rewriting: The data feed messages in Table 6-14
5.000	26/4/2017	**Modification**: The first channel now supports ONLY the Trading Platform's related data (real time data) whereas the second channel supports ONLY the OTC related data in case the Exchange will provide MiFID II APA facilities. The market reference data will be provided though another service, called RDS. Specification for this service will be provided separately

Table of Contents

List of Figures

List of Tables

1. Introduction

Via the FIC-IOCP service the exchange provides to market data vendors a technically more advanced, secure, and cost effective way to connect to a central source of information wherefrom they can receive both security-related and derivative-related data feeds. Furthermore, it allows the Vendors to interact, through their applications, directly with the service.

The FIC-IOCP service comprise both hardware and software components. However, the cornerstone of this service is a software application called Internet-Oriented Communication Portal (IOCP). Given the importance of the IOCP as part of the service, the document will henceforth refer to the FIC-IOCP service as IOCP.

1.1. Document Scope

The document's scope is twofold:

a) to provide all necessary information pertinent to the development of an efficient and IOCP-compliant Vendor application

b) and to ensure that the Vendors will exploit properly the full spectrum of the IOCP's capabilities

1.2. Document Audience

The information of this document should concern mainly the following three types of readers:

1. application designers who would like to design IOCP-compliant Vendor applications
2. application programmers who would like to implement IOCP-compliant Vendor applications
3. all Market Data Vendors and/or ISVs who would like to learn the restrictions imposed on their applications by IOCP and the exchange's infrastructure

1.3. Prerequisite Background

The following three subsections define the technical background a reader should possess before start reading this document. If any of the first two topics is unknown to the reader, we advise him\her to acquire this knowledge first and then come back to this document.

1.3.1. Server Logic and Data Structures

The reader should have a good understanding of the following theoretic topics:

- TCP-IP sessions and how they get established (both for server and client)
- Basic Data Structures (i.e. link lists, queue, circular queues etc.)
- Threads and their usage
- Input Output (I/O) types (i.e. overlapped, blocking, non-blocking etc.)

1.3.2. Data Feed Services (Functionalities and Message Specifications)

The reader should be familiar with the functional characteristics (i.e. messages, retransmission types, Vendor's filtering settings etc.) of the existing data feed dissemination services. Furthermore, we expect that the reader has read the latest version of the document "**Market Data Feed Specification**".

1.3.3. Programming Language & Libraries

All of the source code excerpts, algorithms, or pseudo-codes listed in this document are based on the C/C++ programming language, and the standard sockets API. Thus, a reader who has a working knowledge of the given language and API would be able to take the most out of this information.

1.4. Document layout

This document is divided into the following eight chapters:

- ***Chapter 1,* Introduction.** This introduction.
- ***Chapter 2,* IOCP Main Characteristics.** This chapter outlines the main functional characteristics of the IOCP service.
- ***Chapter 3,* IOCP-Client Connection Channels.** This chapter describes the proper order with which a client application should handle its communication channels with the IOCP.
- ***Chapter 4,* Interaction Protocol.** This chapter defines the IOCP's message-based, interaction protocol, and how this protocol should be utilized by the IOCP and the client applications.
- ***Chapter 5,* Authentication.** This chapter explains the authentication procedure with which a client application logs itself into the IOCP service.
- ***Chapter 6,* Messages.** This chapter lists and describes the message types (control and data) supported by the IOCP.
- ***Chapter 7,* Development Guidelines.** This chapter provides a set of general and implementation-specific guidelines pertinent to the development of an IOCP-compliant client application.
- ***Chapter 8,* Appendix I: Source Code Examples in C.** This chapter contains a set of source-code examples (in the C programming language) pertinent to socket programming and to the IOCP-Client authentication mechanism.
- ***Chapter 9,* Appendix II: Error Code List.** This chapter lists all error codes returned by the IOCP to a client application via the control reply messages or the general error message.
- ***Chapter 10*, Appendix III: Frequently Asked Questions.** This chapter answers the most frequently asked questions related to the IOCP service.

1.5. Acronyms and Abbreviations

Acronym	Explanation
ANSI	American National Standards Institute
API	Application Programming Interface
ATHEX	Athens Exchange
CSE	Cyprus Stock Exchange
FIC	Feed Information Center
IOCP	Internet Oriented Communication Portal
I/O	Input Output
ISV	Independent Software Vendor
O/S	Operating System
SDK	Software Developement Kit

Acronym	Explanation
TCP/IP	Transport Control Protocol/Internet Protocol

1.6. Contact Information

Please address your questions/recommendations pertinent to the contents of this document by mail to:

Market Data Services
Business Development – Services Division
Athens Exchange S.A.
110, Athinon Ave., GR 104 42 Athens
Tel. +30 210 336 6340,
Fax. (+30) 210 336 6296
MDS@helex.gr

1.7. Final Note

The exchange has set a number of rules which ensure the proper and rational use of its computing and network infrastructure. This service is subjected to these rules whenever it utilizes the aforementioned infrastructure.

2. IOCP Main Characteristics

The IOCP service is defined by the following characteristics:

a) utilizes TCP/IP as its communication protocol

b) implements a proprietary Vendor-Service interaction protocol

c) enforces Vendor authentication

d) supports two types of data feed accounts (i.e. time sensitive and relaxed)

e) facilitates data feed filtering per Vendor connection (see Section 2.5 for more information).

Please note that the service does NOT provide/export a library with Application Programming Interface (API) calls.

The following five sections outline the main points of these characteristics.

2.1. Vendor-Service Communication

Two TCP/IP-based channels (i.e. Control and Data) comprise and facilitate the Vendor-Service communication. The Control channel is the master channel and through it the Vendor can interact with the service. The Data channel is a one-way channel used by the service for data feed dissemination (see Chapter 3 for more information on establishing these channels).

Figure 2-1 depicts an abstraction of the Vendor-Service Communication

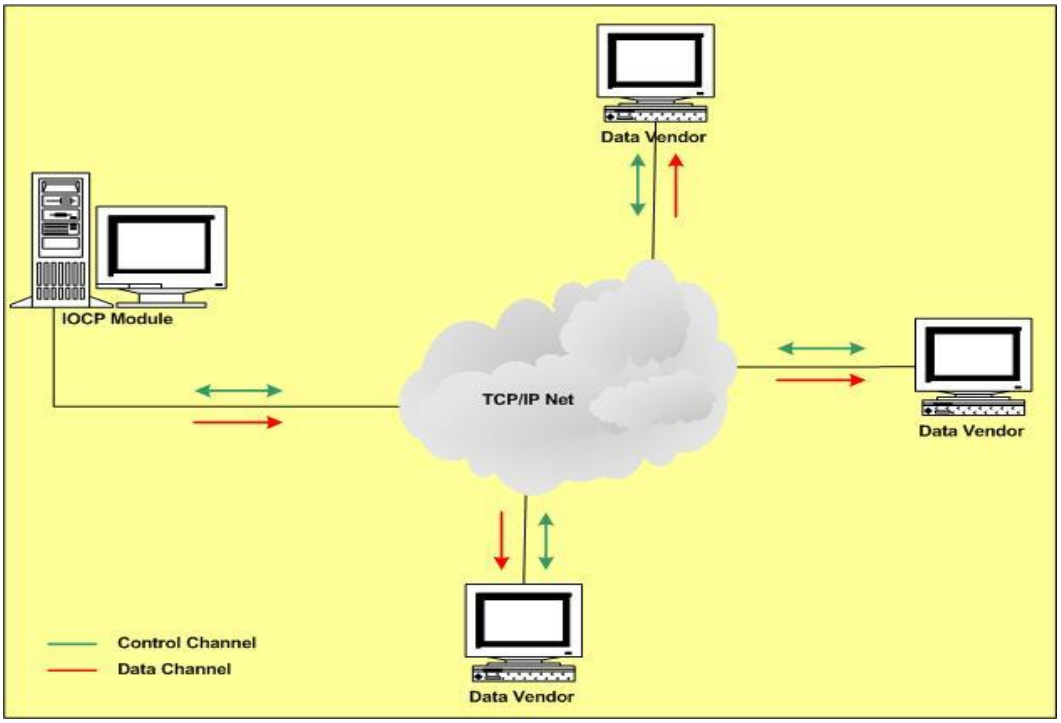

Figure 2-1, Abstraction of the Vendor-Service Communication

2.2. Vendor-Service Interaction

A set of control messages comprise the IOCP's interaction protocol. Through these messages the Vendor application can authenticate itself to the service, check the status of its control/data connections, close its control/data connections, and commence or stop the transmission/retransmission of data feed messages. Please read Chapter 4 and Chapter 6 for

more information related to the interaction protocol and for a full list of the Control message structures, respectively.

2.3. Vendor Authentication

Given the untrustworthy nature of IP-based communications, the IOCP asks each prospective client to authenticate him/herself prior to using the service. The authentication procedure utilizes a simple hash-based, challenge-response scheme where the user's name, password, IP address and a random number comprise the authentication tokens (read Chapter 5 for more information related to the authentication procedure).

2.4. Data Feed Accounts

The IOCP categorizes the data feed messages as follows:

- messages that must be disseminated in real time (**time-sensitive**)
- messages that can withstand a minor delay (**relaxed**)

The need for such distinction derives from the fact that the IOCP might have to disseminate messages up to 9 MB long each, which in their turn will introduce unacceptable transmission, lags to the remaining feed. To accommodate the above distinction the IOCP provides two types of Data channels and maps these channels to two different **Account** types. The first type is for real time transmission speeds and supports only Trading-Server-derived data feed (for either securities, derivatives or both); whereas, the second type is used for disseminating ONLY OTC Trades - Category T (read paragraph 6.2 Data Messages for more information).

2.5. Filtering of Data Feed

The IOCP enables data feed packet filtering on its outbound traffic. Namely, the service is capable to filter the outbound traffic on each Vendor connection according to the following seven main criteria:

1. the data feed Venue ID (e.g. Athens Exchange, Cyprus Exchange, Hellenic OTC etc)
2. the data feed type (e.g. Trading Server-derived data feed for securities and/or derivatives and Other Source-derived data feed for OTC Trades)
3. the dissemination or not of the full Order depth
4. the number of Best Bid Offer levels (i.e. 1 or 5) a list of specific instrument symbols

The last three filtering settings apply only to time-sensitive data feed packets.

3. IOCP-Client Connection Channels

Every Vendor application (henceforth the **Client**) must establish two socket-based connections with the IOCP (i.e. control and data). The client must establish these connections in the following order:

- first the control connection
- second the data connection

Note that the client interacts with the IOCP through its control connection and receives the data feed messages through its data one.

The first two sections of this chapter explain how a client should establish these connections. The last two sections describe how a client should recover its lost control and/or data connections.

3.1. Opening Control Channel

A client application should open its control connection with the server as follows:

- Create a STREAM socket.
- Set as remote IP and Port the values **<server's IP>** and **<server's control port>** respectively.
- Issue a connection request to that remote address

Normally, the IOCP will accept the connection request and the control channel will be established. From that point on, the client application can use the control messages to interact with the IOCP (see Chapters 4 and 6 for more information regarding the interaction protocol and the types of control messages). In addition, see Section 8.1 for a coding example.

If the IOCP does not accept the connection, then the client application must log the returned error code, check the correctness of the IP and Port values, and try again to connect. If the problem persists, the Vendor or developer must gather all these error codes and contact the exchange for assistance.

3.2. Opening Data Channel

If there is a control connection between the client and the IOCP and the client is authenticated (see Chapter 5 for more information) then and ONLY then the client can initiate the data connection sequence. This sequence is the following:

- Create a STREAM socket.
- Set as remote IP and Port the values **<server's IP>** and **<server's data port>** respectively.
- Issue a connection request to that remote address

Normally, the IOCP will accept the connection request and the data channel will be established. From that point on, the client application can instruct the IOCP to commence the dissemination of data feed messages (see Chapters 4 and 6 for more information regarding the interaction protocol and the types of data feed messages). In addition, see Section 8.2 for a coding example.

If the IOCP does not accept the connection, then the client application must log the returned error code, check the correctness of the IP and Port values, and try again to connect. If the problem persists, the Vendor or developer must gather all these error codes and contact the exchange for assistance.

NOTE: The client MUST establish the data channel that matches its Account type (i.e. time-sensitive or relaxed). Otherwise, the IOCP will reject the client's connection attempts.

3.3. Losing Control Channel

If a client loses its control connection with the IOCP, then the IOCP will log off that client, close the client's data connection (if existed) and initialize all IOCP-internal variables (i.e. serial number of last packet sent, filtering settings, etc) pertinent to that client. In that case, the client will have to establish a new control connection with the IOCP, authenticate itself to the IOCP, and establish a new data connection with the IOCP, prior to start receiving again data feed messages.

The following five reasons can cause the closure of an IOCP-Client control connection:

- the IOCP operator logged him off. In that case the IOCP will send a 'GE' message to the client
- the IOCP went down due to a system or an application crash
- the Client went down due to a system or an application crash
- a network-wide failure occurred
- the client issued a close-control-connection ('CC') request message

3.4. Losing Data Channel

If a client loses ONLY its data connection with the IOCP, then the client can re-establish it by following the steps listed in section 3.2. After establishing a new data connection with the IOCP, the client should issue a transmission start request message ('CT'). If it does so, the IOCP will start the dissemination of data feed messages from the point indicated by the CT's *lstPackSent* field (see Paragraph 6.1.1.4 for more information related to the format and use of the 'CT' control message).

On the other hand, if the client looses its data connection due to a control connection failure, then it should follow the steps mentioned in the previous section prior to establishing the data channel and issuing a transmission start request message ('CT').

The following six reasons can cause the closure of an IOCP-Client data connection:

- the IOCP operator closed the connection forcefully. In that case the IOCP will send a 'GE' message to the client
- the IOCP went down due to a system or an application crash
- the Client went down due to a system or an application crash
- the client lost its control connection with the IOCP
- a network-wide failure occurred
- the client issued a close-data-connection ('CC') request message

4. Interaction Protocol

Figure 4-1 depicts an abstraction of the interaction protocol divided into phases. The message types used by the protocol are listed in section 6.1. Note that a client could interact with the IOCP ONLY through its control channel.

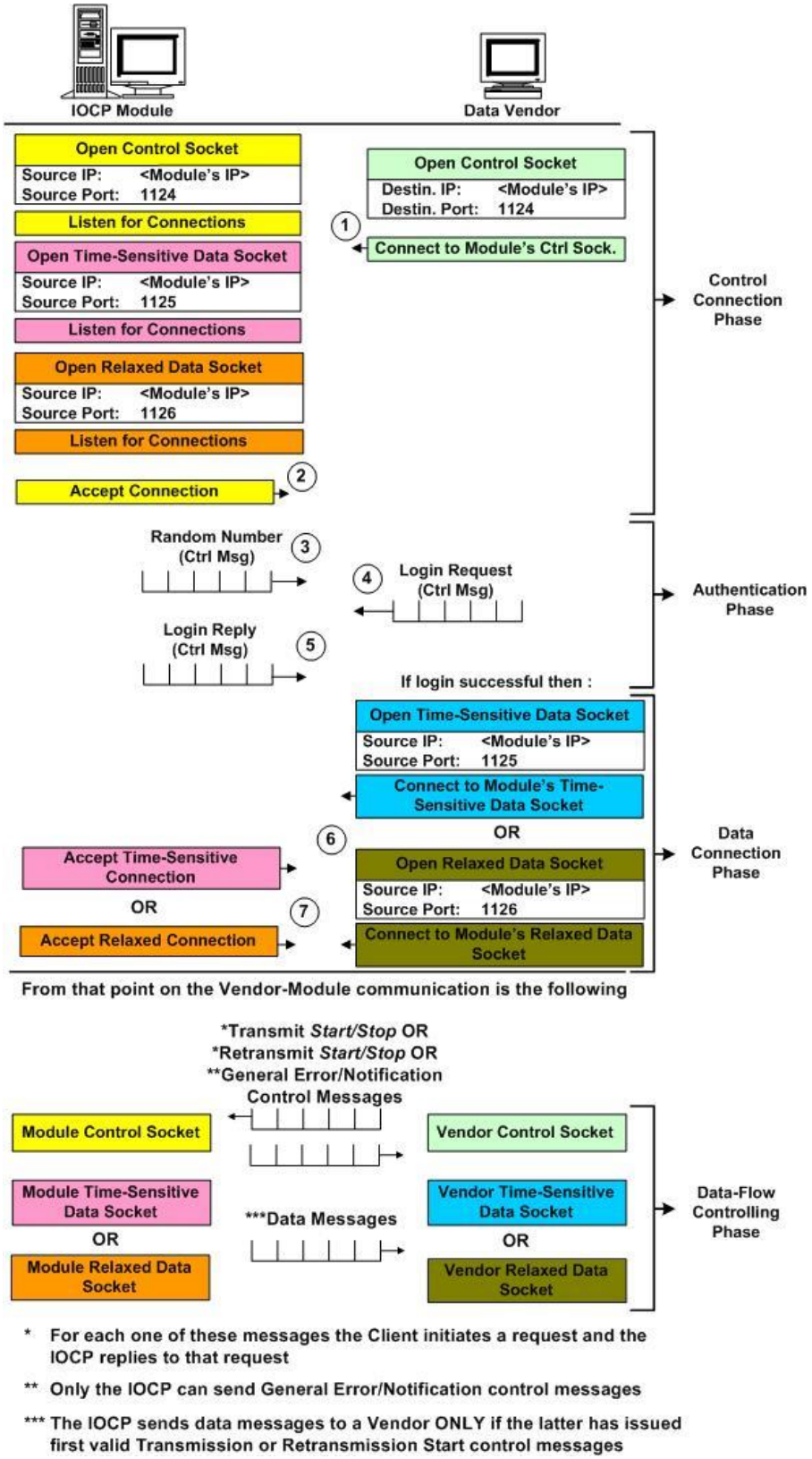

Figure 4-1, Abstraction of the IOCP-Vendor Interaction Protocol

According to **Figure 4-1** the protocol proceeds as follows:

1. the IOCP opens three listening sockets (control, time-sensitive data and relaxed data)
2. the client opens its control socket and connects to the IOCP's one
3. the IOCP accepts the connection and sends a random number through the Authentication Challenge ('AC') control message
4. the client uses this random number along with the rest of the authentication tokens and issues a Login ('CL') request control message
5. the IOCP authenticates the client and sends back a Login ('SL') reply control message
6. if the client becomes authenticated then based on its account type it can open
    a. Either a time-sensitive data socket
    b. Or a relaxed data socket

    (but NOT both) and connect it with the respective listening socket on IOCP
7. the IOCP will accept the connection and become ready to commence the dissemination of data feed messages. From this point on, the interaction between the client and the IOCP involves the following types of control messages:
    - Client's Close Channel 'CC' request and IOCP's Close Channel 'SC' reply
    - Client's Channel Status 'CS' request and IOCP's Channel Status 'SS' reply
    - Client's Transmission (Start/Stop) 'CT' request and IOCP's Transmission (Start/Stop) 'ST' reply
    - Client's Retransmission (Start/Stop) 'CR' request and IOCP's Retransmission (Start/Stop) 'SR' reply
    - IOCP's General Error ('GE') and General Notification ('GN')

Moreover, the steps of the protocol are divided into four phases. Each phase contains either TCP/IP-protocol-specific steps/messages, IOCP-specific steps/messages, or both. Table 4-1 groups these messages into their respective phase.

Phase	Specific To		# of Msgs used		IOCP Msgs types
	TCP/IP	IOCP	TCP/IP	IOCP	
Control Connection Phase	Yes	No	2	0	None
Authentication Phase	No	Yes	0	3	'AC', 'CL', 'SL'
Data Connection Phase	Yes	No	2	0	None
Data Flow Controlling Phase	No	Yes	0	10	'CC', 'SC' or 'CS', 'SS' or 'CT', 'ST' or 'CR', 'SR' or 'GE' or 'GN'

Table 4-1, Interaction Protocol Phases and Messages Used

The following four sections provide additional information related to these phases.

4.1. Control Connection Phase

During this phase the client establishes its control channel with the IOCP. It is imperative that the application completes successfully this phase prior to proceeding with the next ones. See Section 3.1 for more information on establishing a control channel with the IOCP.

4.2. Authentication Phase

During this phase the client authenticates itself to the IOCP. Unauthenticated clients will be dropped by the IOCP within 16 seconds from the time their control channels became established. If a client does not complete this step and proceeds with the next ones, the IOCP will automatically reject all client control-message requests from that point on. See Chapter 5 for more information on authenticating a client to the IOCP.

4.3. Data Connection Phase

During this phase the client establishes a data channel pertinent to its Account type with the IOCP. If a client does not complete successfully this phase then it will not be able to receive any data feed type (i.e. equities, derivatives, or Financial News). See Section 3.2 for more information on establishing a data channel with the IOCP.

4.4. Data Flow Controlling Phase

During this face the client can request from the IOCP the following:

- To close its data connection (‘CC‘ message)
- To close its control connection (‘CC‘ message)
- To check the status of its control connection (‘CS‘ message)
- To check the status of its data connection (‘CS‘ message)
- To commence transmission (‘CT‘ message)
- To stop transmission (‘CT’ message)
- To commence retransmission (‘CR’ message)
- To stop retransmission (‘CR’ message)

The IOCP will answer to those requests as follows:

- It will either close the data channel for the given client or return an error code with an ‘SC’ message
- It will either close the control channel for the given client (i.e. log off the client) or return an error code with an ‘SC’ message
- It will report back to the client with an ‘SS’ message the IOCP’s view of the client’s control channel status
- It will report back to the client with an ‘SS’ message the IOCP’s view of the client’s data channel status
- It will either commence transmission for the given client or return an error code with an ‘ST’ message
- It will either stop transmission for the given client or return an error code with an ‘ST’ message
- It will either commence retransmission for the given client or return an error code with an ‘SR’ message. If the request is valid but contains no data then the IOCP will also send a ‘GE’ message to the client stating this fact. Nonetheless, the IOCP will signal the ending of a valid retransmission with a ‘GN’ message.
- It will either stop retransmission for the given client or return an error code with an ‘SR’ message. If there is no retransmission to stop the IOCP will simply disregard the client’s requests.

See Paragraphs 6.1.1.2, 6.1.1.3, 6.1.1.4, 6.1.1.5, 6.1.2.2, 6.1.2.3, 6.1.2.4, 6.1.2.5, 6.1.3.3 and 6.1.3.2 for more information related to the control messages involved in this phase.

5. Authentication

Every client must be authenticated prior to gaining full access to the IOCP's resources. The authentication tokens are the following:

- the client's user name (issued by the IOCP)
- the client's password
- the client's IP address
- a random number (issued by the IOCP)

The client's username is issued by the exchange right after the signing of the service contract. The username cannot be modified. The password and IP are issued by the Vendor and can be modified by the Vendor non-interactively (i.e. the Vendor must contact the exchange and request the respective modification). Finally, the random number is generated by the IOCP, it is session-based, and represents the challenge part of the challenge-response authentication scheme. In addition to these tokens, the authentication procedure utilizes the Secure Hash Algorithm 1 (SHA1) to disguise the client's user name and password.

If the client becomes authenticated then it can proceed with the next steps of the interaction protocol (see Chapter 4 for more information regarding the interaction protocol). Otherwise it must try to authenticate itself again. Remember that the IOCP will close all unauthenticated client control connections within 16 seconds from the time these connections were initially established.

5.1. Authentication Steps

A client must perform the following three steps during authentication:

Compute the SHA1 of its username and IP Address

Compute the SHA1 of its Password, appended with the IOCP-provided random number, appended with its username, appended with its IP Address

Send a Login Request control message to the IOCP with these two hash values

The IOCP will reply with a Login Reply control message stating a successful login or the reason why the authentication failed. Note that it can be ONLY one authenticated client connected (per session) to the IOCP for each set of tokens (i.e. the IOCP does NOT allow two sessions of the same client to be connected to the IOCP at the same time). Nonetheless, if a client tries to establish a second session with the same authentications tokens, then the IOCP will handle it as follows:

a) it will authenticate the client

b) it will assign a new session to that client

c) it will purge the client's old session and all IOCP-internal structures related to it

d) it will start serving the new session

5.2. Hash Values Computation

Figure 5-1 and Figure 5-2 list the algorithms used in the computation of a client's SHA1 hash values.

Figure 5-1, User Name Hash Computation algorithm

Figure 5-2, User Password Hash Computation algorithm

According to the aforementioned algorithms a client with user name **GEORG0001**, password **1kodikos4**, IP address **172.16.2.31** and session random number **12340875**, will produce the following intermediate *newUserNameStr* and *newPasswordStr* strings respectively:

User Name Hash Computation

- Sets for *UserName* the string "**GEORG00010000000**"
- Sets for *IPAddress* the string "**172.16.2.3100000**"
- Interweaves the above strings into one string and sets for *newUserNameStr* the string "**G1E7O2R.G1060.021.030100**"

User Password Hash Computation

- Appends the password, random number, user name, and IP address into one string and sets for *newPasswordStr* the string "**1kodikos412340875GEORG0001172.16.2.31**"

If the implementation of the SHA1 algorithm is correct then the hash values of the above two intermediate strings will be correct and the IOCP will validate the given client (see Section 8.3 for a coding example related to the computation of these two intermediate strings).

5.3. SHA1 References

RFC3174 describes the SHA1 algorithm and provides an implementation of this algorithm in C. Different implementations of the same algorithm (in different programming languages as well) can also be found on the Web (hint: use **"Secure Hash Algorithm 1"** as a search string).

6. Messages

An IOCP client must support the following two sets of messages:

- Control
- Data

The control set contains client request messages, server reply messages and general control messages. Each one of these messages can be disseminated ONLY through the IOCP-Client control channel. The client can send from this set to the IOCP ONLY the client request control messages.

The data set contains the actual Security/Derivative or Financial News related data feed messages. ONLY the IOCP can disseminate such messages to the client and ONLY through the IOCP-Client data channel.

6.1. Control Messages

There are five client-request messages and their respective server-reply ones. Finally, there are three general purpose control messages. The next three subsections describe the structures and types of these messages.

NOTE: Multi-byte numbers (ULONG, DWORD, long, WORD, short) conform to the **Little Endian** byte-ordering scheme (read the byte-ordering question in Chapter 10 for more information).

6.1.1. Client Request Control Messages

6.1.1.1. Client Login 'CL'

A client sends this message in order to authenticate itself to the IOCP. A client must send this message immediately after the establishment of its control channel with the IOCP and the reception of the 'AC' message. The IOCP answers to the client's 'CL' request with an 'SL' message. Table 6-1 lists the structure and format of the CL message.

CL (Client's Login Request)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'C' for client
h2	BYTE	Header 2 : 'L' for login
sNum	long[1]	Packet's Serial number : **for future use**
userNameHash	char*21	User name + IP address SHA1 hash
userPasswordHash	char*21	User password + random number + user name + IP address SHA1 hash
randomNum	DWORD	The random number send by IOCP via the authentication challenge 'AC' message
Total Bytes	52	

Table 6-1, Client's Login Request, Message Structure

[1] When counting the size of the message the reader should set the long, ULONG and DWORD equal to 4 bytes each; WORD and short equal to 2 bytes each; BYTE and char equal to 1 byte each.

The IOCP will reject every ‘CL’ message sent by a client who does not have a signed service contract with the exchange.

6.1.1.2. Client Close Connection ‘CC’

With a ‘CC’ message a client can terminate its control and/or data connection(s) with the IOCP. The IOCP answers to the client’s ‘CC’ request with an ‘SC’ message. Table 6-2 lists the structure and format of the CC message.

CC (Client’s Close Connection Request)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'C' for client
h2	BYTE	Header 2 : 'C' for close
sNum	long	Packet's Serial number : **for future use**
Channel	BYTE	The channel to close 'C' : for control 'D' : for data
Total Bytes	7	

Table 6-2, Client's Close Connection Request, Message Structure

Note that the IOCP will log off every client issuing a valid close-control-connection request message.

6.1.1.3. Client Channel Status ‘CS’

With a ‘CS’ message a client can query from the IOCP status information pertinent to its control or data connection. The IOCP answers to the client’s ‘CS’ request with an ‘SS’ message. Table 6-3 lists the structure and format of the CC message.

CS (Client’s Channel Status Request)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'C' for client
h2	BYTE	Header 2 : 'S' for status
sNum	long	Packet's Serial number : **for future use**
Channel	BYTE	The channel to check for its connection status 'C' : for control 'D' : for data
Total Bytes	7	

Table 6-3, Client's Channel Status Request, Message Structure

The client is free to exploit this message as a session Keep-Alive.

6.1.1.4. Client Transmission ‘CT’

With this message the client instructs the IOCP to do the following:

- Commence the dissemination of,
  a) time-sensitive data feed (i.e. set the *State* to ‘B’ and *dType* to ‘A’)

b) OR, relaxed data feed (i.e. set the *State* to ‘B’ and *dType* to ‘O’)

- Stop the dissemination of,
  a) time-sensitive data feed (i.e. set the *State* to ‘S’ and *dType* to ‘A’)
  b) OR, relaxed data feed (i.e. set the *State* to ‘S’ and *dType* to ‘O’)

The IOCP answers to the client’s ‘CT’ request with an ‘ST’ message. Table 6-4 lists the structure and format of the CT message.

CT (Client’s Transmission Request)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'C' for client
h2	BYTE	Header 2 : 'T' for transmit
sNum	long	Packet's Serial number : **for future use**
State	BYTE	Transmission state 'B' for begin 'S' for stop
dType	BYTE	Valid data feed types for ***Time-Sensitive*** 'A' all real time data feed (security and derivatives data feed) ***Relaxed*** 'O' for Other feeds
lstPackSent	long	Transmission’s initial point. This value does not apply if State is S ' <-1' : send the next message from the one the given client had last received ' -1' : send the last message in the client’s list '>=0' : start sending from the given serial number
Total Bytes	12	

Table 6-4, Client's Transmission Request, Message Structure

When the client sends a transmission “begin” request it must specify the data feed type matching its Account type (i.e. time-sensitive or relaxed). It must also set the transmission’s initial point (i.e. the first message to be sent). The following three value ranges can be used as valid initial points:

- *Any number less than minus one (-1)*, instructs the IOCP to take the next available data feed packet in the client’s list (relative to the current position) and send that packet to the client. For example, if the client has not received any data feed then the IOCP will send to that client the message with serial number zero (0) if the client’s feed type is ‘A’ or ‘O’. On the other hand, if the serial number of the last data message received (for a given feed type) is 100 then the IOCP will send the message with serial number equal to 101. A client **SHOULD NOT** use this range as an initial point if its data connection with the

IOCP terminates abnormally (e.g. failure of the client's control connection, IOCP system or application crash, etc).

- *The number minus one (-1)*, instructs the IOCP to skip all data feed messages (of a given feed type) pending dissemination and to send the last message available. For example, if the latest data message in the IOCP, for a given feed type, is the one with serial number 3000 but for some reason the client has stayed behind and received only messages up to 500, then the IOCP will skip 2499 data messages and it will send back to the client the message with serial 3000. The client can recover the 2499 preceding data messages through retransmission.
- *Any nonnegative number*, instructs the IOCP to send the data feed message whose serial number equals to the given number. For example, if the client issues a transmission start request message, for a given feed type, with *lstPackSent* = 1000 then the IOCP will send the data message with serial number 1000. If such action introduces a gap then the client can recover from that gap through retransmission. The IOCP will reject a 'CT' message, if the number entered does not represent a valid data feed message.

Note that the transmission's initial point does not apply for Transmission 'Stop' requests. Furthermore, if the client stops the transmission of a given data feed type it will not receive any data feed for that type until it sends a CT message again with *State* equal to 'B'. Under normal circumstances the client should send one transmission "begin" message for each data feed type at the beginning of the trading day and one transmission "stop" message at the end of the trading day. A client should not send two or more successive transmission 'B' (begin) or 'S' (stop) messages, since the logical sequence is one transmission 'B' for each transmission 'S' message sent.

6.1.1.5. Client Retransmission 'CR'

With this message the client instructs the IOCP to do the following:

- Commence a retransmission of,
  a) time-sensitive data feed (i.e. set the *State* to 'B' and *iType* to 'A')
  b) OR, relaxed data feed (i.e. set the *State* to 'B' and *iType* to 'O')
- Stop and purge a given retransmission (i.e. set the *State* to 'S' and enter a valid *rID*).

The IOCP answer's to the client's 'CR' request with an 'SR' message. Table 6-5 lists the structure and format of the CR message.

CR (Client's Retransmission Request)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'C' for client
h2	BYTE	Header 2 : 'R' for retransmit
sNum	long	Packet's Serial number : **for future use**
State	BYTE	Retransmission state 'B' for begin 'S' for stop
rID	long	Retransmission ID Put the *rID* value of the retransmission to stop (i.e. the state = 'S') Put 0 otherwise
iType	BYTE	Valid instrument types for

		Time-Sensitive 'A' for all real time data feed (security and derivatives data feed) ***Relaxed*** 'O' for Other feeds
rCateg	BYTE	Valid retransmission categories for ***Time-Sensitive*** 'G' for all instrument summaries 'A' for all real time data feed ***Relaxed*** 'T' for all OTC trades
rRangeB	long	The beginning of the retransmission range. The range does not apply if State=S, or iType=A and rCateg=G '<=0' : start from the beginning ' >0' : start from the given serial number
rRangeE	long	The ending of the retransmission range. The range does not apply if State=S, or iType=A and rCateg=G '<=0' : retransmit until the IOCP reaches the end of the client's list ' >0' : stop retransmission up to that given serial number
Total Bytes	21	

Table 6-5, Client's Retransmission Request, Message Structure

Specifically, the client can set the following retransmission characteristics, part of which are confined by its Account type:

- *retransmission ID*: This field applies ONLY to retransmission STOP request messages. The client must enter in this field the ID of the retransmission to be stopped.
- *Instrument type*: The client can instruct the IOCP to retransmit data feed messages for types falling under the client's Account type. If a client asks for an instrument type that (s)he is not allowed to access/receive then the IOCP will reject such retransmission request.
- *retransmission category*: This field provides a finer grain on the set of retransmitted data. For relaxed types of retransmissions this field can be set ONLY to 'T', since that type of Account supports ONLY OTC Trades. For time-sensitive types of retransmissions, however, this field can take the values of 'G', or 'A' if *iType* equals to 'A'. Please note that for the category 'G' the IOCP will send the summaries it finds. Finally, if a client asks for a retransmission category that (s)he is not allowed to receive, according to its Account/Client profile, then the IOCP will reject such retransmission request.
- *retransmission range*: The client can define the range of retransmission. If the client sets *rRangeB* = -1 and *rRangeE* = -1 then the IOCP will retransmit to the client all available data feed messages. The range does not apply if:

1. it is a Retransmission ‘Stop’ request
2. the instrument type is set to ‘A’ and the retransmission category is set to ‘G’ (i.e. retransmit instrument summaries)

Each client can issue up to three successive retransmission-start request messages. Additional retransmission-start request messages can be issued by the same client ONLY if any one of his/her initial retransmissions gets stopped or completes itself. Finally, if a client issues a retransmission-stop request message without issuing first a retransmission-start request message then the IOCP will discard and not reply to that request.

6.1.2. IOCP Reply Control Messages

6.1.2.1. Server Login ‘SL’

The IOCP sends this message as a reply to a client’s ‘CL’ request. With this message the IOCP provides the following information:

- the client is authenticated successfully
- the client sent either an empty username hash, or a hash with size greater than 21 bytes or a hash which does not match with the hash computed by the IOCP
- the client sent either an empty password hash, or a hash with size greater than 21 bytes or a hash which does not match with the hash computed by the IOCP
- the random number contained in the client’s CL message does not match with the random number contained in the IOCP’s AC message
- the client in this session is already authenticated (a warning response)

Table 6-6 lists the structure and format of the SL message.

SL (Server’s Login Reply)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'S' for server
h2	BYTE	Header 2 : 'L' for login
Result	short	The possible results of the server’s reply: '0' : login successful '101' : invalid username '102' : invalid password '103' : invalid random number '104' : user in this session is already logged in
sNum	long	Packet's Serial number : **for future use**
Total Bytes	8	

Table 6-6, Server's Login Reply, Message Structure

All result values other than zero (0) and ‘104’ (user already logged in) denote that the IOCP has rejected the client’s CL request.

6.1.2.2. Server Close Connection ‘SC’

The IOCP sends this message as a reply to a client’s ‘CC’ request. With this message the IOCP provides the following information:

- the IOCP closed the channel successfully

- the client requested the closure of a channel other than the control or the data one

Table 6-7 lists the structure and format of the SC message.

SC (Server's Close Connection Reply)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'S' for server
h2	BYTE	Header 2 : 'C' for close
Result	short	The possible results of the server's reply: '0' : the channel was closed '201' : invalid channel type
sNum	long	Packet's Serial number : **for future use**
Channel	BYTE	The channel to close Puts the channel value of the client's CC request message if result = '201' Puts 0 otherwise
Total Bytes	9	

Table 6-7, Server's Close Connection Reply, Message Structure

All result values other than zero (0) denote that the IOCP has rejected the client's CC request.

6.1.2.3. Server Channel Status 'SS'

The IOCP sends this message as a reply to a client's 'CS' request. With this message the IOCP provides the following information:

- the IOCP registers the queried channel as Up (i.e. connected)
- the IOCP registers the data channel as Down (sent ONLY if the queried channel was the Data one)
- the IOCP registers the control channel as being authenticated (sent ONLY if the queried channel was the Control one)
- the client queried for the status of a channel other than control or data

Table 6-8 lists the structure and format of the SC message.

SS (Server's Channel Status Reply)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'S' for server
h2	BYTE	Header 2 : 'S' for status
Result	short	The possible results of the server's reply: '0' : the channel is Up '501' : the data channel connection is Down '502' : the control channel has been authenticated

		'503' : invalid channel type
sNum	long	Packet's Serial number : **for future use**
Channel	BYTE	The channel to close Puts the channel value of the client's CC request message if result = '503' Puts 0 otherwise
Total Bytes	9	

Table 6-8, Server's Channel Status Reply, Message Structure

All result values other than zero (0) and '502' (control channel has been authenticated) denote that the IOCP has identified a problem with the status of the channel under question.

6.1.2.4. Server Transmission 'ST'

The IOCP sends this message as a reply to a client's 'CT' request. With this message the IOCP provides the following information:

- the client sent an invalid transmission state (i.e. a character other than 'B' or 'S')
- there is no data connection between the client and the IOCP
- the client's transmission request was accepted and executed by the IOCP successfully
- the client tries to stop the transmission of an already stopped data feed type (i.e. time sensitive or relaxed market data feed)
- the client tries to start the transmission for a data feed type that the IOCP has already started the transmission for
- the client's initial transmission point is invalid (i.e. a positive number that does not represent a valid data feed message). In this case the IOCP will place in the message's *lstPackSent* field the initial-point value obtained by the client's CT message
- the client issued a transmission start/stop message for a data feed type other than all instrument-related 'A' data feed or Other 'O'
- the client issued a transmission start message for a data feed type not allowed to receive. For example a client with a time-sensitive Account type, asks the IOCP to start transmitting relaxed data feed messages (i.e. set dType to 'T'); or a client set to receive ONLY relaxed data feed messages asks the IOCP to start transmitting data feed messages for all instruments (i.e. set dType to 'A') instead of Other (i.e. set dType to 'T');

Table 6-9 lists the structure and format of the ST message.

ST (Server's Transmission Reply)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'S' for server
h2	BYTE	Header 2 : 'T' for transmit
Result	short	The possible results of the server's reply: '-4': invalid state '-3': no data connection present '0' : transmission started\stopped '301' : transmission never started

		'302' : transmission never stopped '303' : invalid lstPackSent number '304' : invalid transmission data feed type '305' : The data feed type is not allowed for the given client/Account type
sNum	long	Packet's Serial number : **for future use**
State	BYTE	Transmission state 'B' for begin 'S' for stop
lstPackSent	long	Transmission's initial point Puts the lstPackSent value of the client's CT request message if result = '303' Puts 0 otherwise
Total Bytes	13	

Table 6-9, Server's Transmission Reply, Message Structure

All result values other than zero (0) denote that the IOCP has rejected the client's CT request.

6.1.2.5. Server Retransmission 'SR'

The IOCP sends this message as a reply to a client's 'CR' request. With this message the IOCP provides the following information:

- the client sent an invalid retransmission state (i.e. a character other than 'B' or 'S')
- there is no data connection between the client and the IOCP
- the client's retransmission request was accepted and stored by the IOCP successfully
- the client has range-begin value greater than the range-end value, or a range that doesn't exist (e.g. the IOCP has ONLY 7000 data feed messages in its list and the client asks for range 9000-10000). In this case the IOCP will place in the message's rRangeB and rRangeE fields the range values contained in the client's CR message.
- either the ID number of a new valid retransmission, or the ID number of the retransmission to be stopped. Specifically, if the client issued a retransmission start message and the IOCP accepts the request then the IOCP will assign an ID to this request and place it inside the *rID* field of this message. The client must save this ID since it is required for stopping the given retransmission. If the client issued a retransmission stop message for an unknown retransmission ID then the IOCP will reject that request and place in the message's *rID* field the value of the retransmission ID contained initially in the client's CR message.
- the client issued a retransmission start message for an unknown retransmission category, or for a category that is not valid in conjunction with the selected instrument type. For example the client asked the IOCP to retransmit data feed messages for category 'S' which does not exist.
- the client issued a retransmission start/stop message for an instrument type other than 'A' for time-sensitive Account types and 'O' for relaxed ones
- the client issued a retransmission start message for an instrument type, or a retransmission category that it is not allowed to receive. For example a client, allowed to receive ONLY time sensitive related data feed, asks the IOCP to retransmit relaxed data feed (i.e. set iType to 'T'.

- the client has already three retransmissions active within the IOCP

Table 6-10 lists the structure and format of the SR message.

SR (Server's Retransmission Reply)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'S' for server
H2	BYTE	Header 2 : 'R' for retransmit
Result	short	The possible results of the server's reply: '-4': invalid state '-3': no data connection present '0' : retransmission started\stopped '401' : invalid retransmission range '402' : invalid retransmission ID '403' : invalid retransmission category '404' : invalid retransmission instrument type '405' : the given client/Account type is not allowed to receive data feeds for this instrument and/or category '407' : no more retransmissions allowed
sNum	long	Packet's Serial number : **for future use**
State	BYTE	retransmission state 'B' for begin 'S' for stop
rID	long	retransmission ID Puts the rID value of the client's CR request message if state = 'B' and result = 0, or state = 'S' and result = 402 Puts 0 otherwise

rRangeB	long	retransmission range begin
		if result = 401 then
		Puts Client's rRangeB value
		Else
		Puts 0

rRangeE	long	retransmission range end
		if result = 401 then
		Puts Client's rRangeE value
		Else
		Puts 0

Total Bytes	21	

Table 6-10, Server's Retransmission Reply, Message Structure

All result values other than zero (0) denote that the IOCP has rejected the client's CR request.

6.1.3. IOCP General Control Messages

6.1.3.1. Server Authentication Challenge 'AC'

This is the first message sent during the authentication phase. It contains the random number, with which the client will compute its password hash value and then complete its CL request message. Note that ONLY the IOCP can send such message. Table 6-11 lists the structure and format of the AC message.

AC (Server's Authentication Challenge)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'A' for authentication
h2	BYTE	Header 2 : 'C' for challenge
sNum	long	Packet's Serial number **for future use**
randNum	long	the random number to challenge with
Total Bytes	10	

Table 6-11, Server's Authentication Challenge Message Structure

6.1.3.2. Server General Error 'GE'

The IOCP (and ONLY the IOCP) will send such message to a client, whenever the following types of systemic or transaction errors occur:

- the IOCP lost its data connection with the client (systemic)
- the IOCP operator closed forcefully the IOCP's data connection with the given client (systemic)
- the IOCP operator closed forcefully the IOCP's data and control connections with the given client (systemic)
- the client issued a valid retransmission request which contains no data (transaction)
- the client tried to establish a data connection with the IOCP, although the latter has already registered such connection to the given client (systemic). In this case the client can send a 'CC' control message to the IOCP, requesting the closure of the existing data channel (see Paragraph 6.1.1.2 for more information regarding the structure and use of the 'CC' control message)

Table 6-12 lists the structure and format of the GE message.

GE (Server's General Error)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'G' for general
h2	BYTE	Header 2 : 'E' for error
sNum	long	Packet's Serial number **for future use**
errorType	BYTE	the error type 'S' for systemic 'T' for transaction

errorID	short	the error ID '1' : The data channel is lost '2' : Server has closed the data channel '3' : Server logged off the client '4' : Retransmission contains no data '5' : The data channel exists
Total Bytes	9	

Table 6-12, Server's General Error Message Structure

6.1.3.3. Server General Notification 'GN'

The IOCP (and ONLY the IOCP) will send such message to a client, whenever the following types of systemic or transaction notifications occur:

- the retransmission issued by that client, with ID equal to *notifID*, came to an end (transaction)

Table 6-13 lists the structure and format of the GN message.

GN (Server's General Notification)		
Field	Type	Meaning\Possible Values
h1	BYTE	Header 1 : 'G' for general
h2	BYTE	Header 2 : 'N' for notification
sNum	long	Packet's Serial number **for future use**
notifType	BYTE	the notification type 'S' for systemic 'T' for transaction
notifID	short	the notification ID '6' : The retransmission with ID equal to 'retrID' came to an end
retrID	long	the retransmission ID if 'notifID' is 6. Otherwise it is -1
Total Bytes	13	

Table 6-13, Server's General Notification Message Structure

6.2. Data Messages

The IOCP supports/disseminates the data feed message categories listed on Table 6-14 for time sensitive Account types, and the categories listed on Table 6-15 for relaxed Account types.

Message Category	Code
Control Message	K
Trade Message	A
Quote Message	B

Message Category	Code
Index Message	C
Instrument Summary Message	G
Canceled Trade Message	I
Closing/Fixing Price Message	L
Projected-Auction/Auction-Open/Projected Close Price Message	M
High/Low Limit Modification Message	N
Instrument State Message	O
Market Status Message	P
Order Message	Q
Canceled Order Message	R
Exchange Notifications Message	S

Table 6-14, All instrument-related Message Categories (time-sensitive type)

Message Category	Code
Control Message	K
OTC Trade Message	T

Table 6-15, Other source-related Message Categories (relaxed type)

More information related to these message categories and their format can be found inside the latest version of the document "**Market Data Feed Specification**".

NOTE: The IOCP starts the sequencing for the messages in Table 6-14 and Table 6-155 from **zero** (0).

7. Development Guidelines

An IOCP-compliant application must satisfy the following general functional requirements:

a) comply with the IOCP interaction protocol

b) support all control messages and the necessary data messages

c) handle and log all error codes returned by the various standard API calls

d) handle and log all error codes sent by the IOCP through the server reply control messages or the general error message

e) log all data feed messages received

This chapter provides a set of guidelines which might assist the implementer of an IOCP-compliant client application to meet the above requirements and to achieve the following:

- design and implement an application which exploits the full spectrum of the IOCP's services
- optimize the application's performance
- minimize the required overall coding and debugging time

Although it is not mandatory for the implementer to follow these guidelines, we advise him\her to take them, at least, under consideration.

7.1. General Guidelines

The implementer should perform the following tasks before (s)he starts implementing an IOCP-compliant client application:

- read this document carefully and identify the key characteristics of the IOCP service
- seek out any additional information related to the existing market data feed services and their business role
- define the business logic and the functional requirements of the client application
- identify the application's programmable functional modules and see if they satisfy the requirements and business logic set earlier
- break these modules into disjoint tasks (if possible) and assign each task to a different thread of execution (multithreading is a good designing practice, especially for applications performing a good amount of background processing like, updating data bases, updating log files, etc.)
- incorporate (if possible) non-blocking, overlapped I/O characteristics, into the application's socket reading/writing and file-data-dumping modules
- verify that your design accommodates error-handling, error-logging, and data-logging features

Finally, the implementer should promote modularity and parameterization in his/her design whenever possible.

7.2. Coding Specific Guidelines

The implementer should adopt the following coding techniques during the implementation of a client application:

- code each control message as an object or a structure, and use direct casting whenever the application reads from or writes to its control socket

- prior to reading from or writing to a socket initialize all socket-related buffers, counters and data structures
- free all heap-allocated memory when done using it
- terminate all unnecessary threads
- before copying a string to a memory buffer nullify its end (do that only if your programming language does not support boundary checking)

8. Appendix I: Source Code Examples in C

This chapter contains a set of source-code examples pertinent to socket programming and to the IOCP-Client authentication mechanism. These examples utilize both ANSI C and Microsoft SDK API calls. We believe that it should be trivial to code these examples in a different programming language.

Finally, these code examples are provided by the exchange “as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of these examples, even if advised of the possibility of such damage.

8.1. Creating Control Socket

The following example demonstrates the opening of a control connection.

```
SOCKET          m_controlSoc;
SOCKADDR_IN     m_controlAddrRemote;
int             m_controlAddrRemoteLen;
BOOL            yes = TRUE;
int             err;

// get the socket handle
m_controlSoc = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

// could not create the socket
if (m_controlSoc == INVALID_SOCKET){
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        return;  }

// make the address reusable
err = setsockopt(m_controlSoc, SOL_SOCKET, SO_REUSEADDR, (const char *) &yes, sizeof(yes));

if (err == SOCKET_ERROR) {
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_controlSoc);
        return; }

// setup the remote address
m_controlAddrRemoteLen = sizeof(SOCKADDR_IN);
memset(&m_controlAddrRemote, 0, m_controlAddrRemoteLen);
m_controlAddrRemote.sin_family = AF_INET;
m_controlAddrRemote.sin_addr.s_addr = htonl(/* put IOCP's IP address */);
m_controlAddrRemote.sin_port = htons(/* put IOCP's CONTROL port */);

// issue the actual connect call
err = connect(m_controlSoc, (SOCKADDR*)&m_controlAddrRemote, m_controlAddrRemoteLen);

// the socket was not connected
if (err == SOCKET_ERROR && WSAGetLastError() != WSAEWOULDBLOCK){
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_controlSoc);
        return; }
```

8.2. Creating Data Socket

The following example demonstrates the opening of a data connection.

```
SOCKET                  m_dataSoc;
SOCKADDR_IN             m_dataAddrRemote; // IOCP's address
int                     m_dataAddrRemoteLen, err = -1;
BOOL                    yes = TRUE;

// get the socket handle
m_dataSoc = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

// could not create the socket
if (m_dataSoc == INVALID_SOCKET){
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        return;  }

// make the address reusable
err = setsockopt(m_dataSoc, SOL_SOCKET, SO_REUSEADDR, (const char *) &yes, sizeof(yes));

if (err == SOCKET_ERROR) {
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_dataSoc);
        return; }

// setup the remote address
m_dataAddrRemoteLen = sizeof(SOCKADDR_IN);
memset(&m_dataAddrRemote, 0, m_dataAddrRemoteLen);
m_dataAddrRemote.sin_family = AF_INET;

m_dataAddrRemote.sin_addr.s_addr = htonl(/* put IOCP's IP address */);
m_dataAddrRemote.sin_port = htons(/* put IOCP's DATA port */);

// issue the actual connect call
err = connect(m_dataSoc, (SOCKADDR*)&m_dataAddrRemote, m_dataAddrRemoteLen);

// the socket was not connected
if (err == SOCKET_ERROR && WSAGetLastError() != WSAEWOULDBLOCK){
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_dataSoc);
        return; }
```

8.3. Computing the Username and Password Hashes

The following two examples demonstrate how a client application should prepare its username and password intermediate strings. Note that in this example the two intermediate strings will be stored within the buffers pointed by the *inputBuf* parameters of the ***PrepareUserNameStr*** and ***PrepareTotalStr*** functions, respectively. When these functions return, a client application should run the contents of these buffers through the SHA1 algorithm.

```
void PrepareUserNameStr(BYTE* inputBuf, int inputBufSize, LPSTR userName, LPSTR IPAddr)
{
	char	tmp1Str[16],  tmp2Str [16];
	int	count, rep, maxLoopCount, str1Size = lstrlen(IPAddr), str2Size = lstrlen(userName);

	// initialize all buffers
	memset(inputBuf, 0, inputBufSize);

	// zero pad the strings
	memset(tmp1Str, '0', sizeof(tmp1Str));
	memset(tmp2Str, '0', sizeof(tmp2Str));

	maxLoopCount = max(str1Size, str2Size);

	// this should NEVER occur
	if (maxLoopCount <= 0 || maxLoopCount >= 16)
		return;

	// set the temp strings
	memcpy(tmp1Str, IPAddr, str1Size);
	memcpy(tmp2Str, userName, str2Size);

	// make sure the "maxLoopCount" counter is divisible by 2
	if ((maxLoopCount%2) != 0)
		maxLoopCount += 1;

	// concatenate the temp strings into one
	for (rep = 0, count = 0; rep < maxLoopCount; count += 2, rep++) {
		*(inputBuf+count) = (BYTE)*(tmp2Str+rep);
		*(inputBuf+count+1) = (BYTE)*(tmp1Str+rep); }
}
```

```
void PrepareTotalStr(BYTE* inputBuf, int inputBufSize, LPSTR userName, LPSTR password, LPSTR IPAddr,
		ULONG randNum)
{
	char randNumStr[12];

	// initialize all buffers
	memset(inputBuf, 0, inputBufSize);
	memset(randNumStr, 0, sizeof(randNumStr));

	// get the string representation of the random number
	itoa(randNum, randNumStr, 10);

	// concatenate the strings
	sprintf(((char*)inputBuf), "%s%s%s%s%c", password, randNumStr, userName, IPAddr, '\0');
}
```

8.4. Reading and Writing to a Socket

The following examples list two of the API calls with which a client application can read from or write to a socket.

```
BYTE            m_msgBuf[512];
WSABUF          m_wsaBuf;
DWORD           nBytes, flags = 0;

// initialize structures
m_wsaBuf.buf = (CHAR*)m_msgBuf;
m_wsaBuf.len = sizeof(m_msgBuf);
memset(m_msgBuf, 0, m_wsaBuf.len);

// issue a read call
err = WSARecv(m_SocketHandle, &m_wsaBuf, 1, &nBytes, &flags,
                NULL, // put an overlapped structure for overlapped I/O
                NULL); // put the name of overlapped call-back handling procedure

if (err == SOCKET_ERROR && WSAGetLastError() != WSA_IO_PENDING) {
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_SocketHandle);
        return;  }
```

```
BYTE            m_msgBuf[512];
WSABUF          m_wsaBuf;
DWORD           nBytes, flags = 0;

// initialize structures
m_wsaBuf.buf = (CHAR*)m_msgBuf;
m_wsaBuf.len = sizeof(m_msgBuf);
memset(m_msgBuf, 0, m_wsaBuf.len);

// issue a write call
err = WSASend(m_SocketHandle, &m_wsaBuf, 1, &nBytes, flags,
                NULL, // put an overlapped structure for overlapped I/O
                NULL); // put the name of overlapped call-back handling procedure

if (err == SOCKET_ERROR && WSAGetLastError() != WSA_IO_PENDING) {
        // log the error code
        printf("Error Code %d", WSAGetLastError());
        // close the socket
        closesocket(m_SocketHandle);
        return;  }
```

9. Appendix II: Error Code List

This chapter lists all possible error codes returned by the IOCP to a client via the control reply messages or the general error message. Table 9-1 displays these values and outlines their respective meaning.

Error Codes	Description
Global	
-4	The client sent an invalid transmission/retransmission state
-3	The client does not have a data connection with the IOCP
0	The control message was handled by the IOCP successfully
General Error	
1	The client's data channel with IOCP is lost
2	IOCP closed forcefully its data connection with the given client
3	IOCP logged off the given client
4	The client's valid retransmission request contains no data
5	The client has already a data connection with the IOCP
General Notification	
6	The retransmission with ID equal to *retrID* came to an end
Login	
101	The client sent an invalid user name during authentication
102	The client sent an invalid user password during authentication
103	The client sent an invalid random number during authentication
104	The client in this session is already logged in (authenticated)
Close Channel	
201	The client tried to close an invalid channel type
Transmit	
301	The client tried to stop a transmission which is already stopped
302	The client tried to start a transmission which is already started
303	The client sent an invalid Last Packet serial number
304	The client issued a transmission start/stop message for an invalid data feed type (i.e. a data feed type other than security 'E' or derivative 'D' related or Other 'O')
305	The given client/Account type is not allowed to receive data feed messages for the given data feed type
Retransmit	
401	The client sent an invalid retransmission range
402	The client sent an invalid retransmission ID
403	The client sent an invalid retransmission type
404	The client sent an invalid retransmission instrument type
405	The given client/Account type is not allowed to receive data feeds for this instrument type and/or retransmission category
407	No more retransmissions allowed for the given client
Channel Status	

501	The client’s data channel connection is registered as Down
502	The client’s control channel connection has been authenticated
503	the client queried for the status of a channel other than control or data

Table 9-1, Error Code Values

10. Appendix III: Frequently Asked Questions

This chapter lists a set of questions pertinent to IOCP and provides the respective answers.

Q: Although I will only use one connection at a time to access the FIC-IOCP service, I want to be able to use a number of clients. Is it valid to use a range of IPs, ie. 212.251.62.x, instead of a specific Source IP, ie 212.251.62.1?

A: Access is valid only through one specific IP (ie. 212.251.62.1). The FIC-IOCP service does not support IP ranges (ie. 212.251.62.x). However, the IOCP will authenticate successfully any client with a valid IP address, regardless if the IP used in the authentication does or does not match the IP address on the client's interface card.

Q: How can I change the Source IP address I stated in the FIC-IOCPs form?

A: The vendor should contact the Support System and Services Department and state its request (see paragraph 1.6 for contact information). The data required for the substitution are:

1. The new Source IP
2. The corresponding login account (mandatory for the vendors having more than one login accounts)
3. The transition date. If a date is not specified the Source IP's substitution will take place the same or the next working date

Q: To how many different login accounts (i.e. user name, password, IP etc.) can a Vendor connect using the same physical IP address?

A: Currently, the Vendor can choose one of the following two combinations:

1. one login account, through which it will receive all time sensitive data feeds, and one login account, through which it will receive messages for other source-related data feeds (i.e. relaxed type of Account)

Note however that the data feeds will be transmitted to the client over the same physical line, regardless of the choice made.

Q: How many TCP/IP connections a client application can establish with the IOCP, per login account?

A: The IOCP will allow a client with a valid login account to establish ONLY one Control and Data connection with it. Thus, a client can open, per login account, the maximum of two TCP/IP channels.

Q: What bandwidth the network connections to the new TCP/IP-based feed service should support?

A: Given the current bandwidth utilization of both data feed services (i.e. securities and derivatives) and the Financial News, a connection with bandwidth of 256Kbs should suffice.

Q: What is the approximate amount of data feed messages sent per day for securities and /or derivatives?

A: Currently, the two feed services disseminate per day an average of 155000 data feed messages for securities and 100000 for derivatives.

Q: Is there any difference on the structures and/or framing characters (i.e. the SOH, STX, ETX, etc.) between the data feed messages sent over TCP/IP and the ones sent over the serial connections (currently on production)?

A: The structures and/or framing characters of the feed messages (Data Messages) sent over TCP/IP are identical to the ones sent over the serial connections. As such, a message-parser should be able to process ONLY the structures and/or framing characters described in the latest version of the document "**Market Data Feed Specification**"..

Q: Do the control messages have framing characters?

A: There aren't any framing characters for the Control messages. The structures and types of the Control messages are described in the latest version of the document "**FIC–IOCP Service Interface Technical Specification**".

Q: What is the Big/Little Endian byte-ordering scheme and how these schemes could affect the IOCP-Client control channel communication?

A: Microprocessor architectures utilize two different schemes for storing the individual bytes of multi-byte numerical data in memory. For example, a Motorola processor stores a two-byte integer with its most significant byte first, followed by its least significant byte (i.e. the "short" number **101** in base 10 is stored as **0x00**, **0x65** in hex). This scheme is called "Big-Endian" byte ordering. On the other hand, a Intel x86 processor stores a two-byte integer with the least significant byte first, followed by the most significant byte (i.e. the "short" number **101** in base 10 is stored as **0x65**, **0x00** in hex). This scheme is called "Little-Endian" byte ordering.

The IOCP's Control Messages contain multi-byte numerical data (i.e. ULONG, DWORD, long, WORD, short) which conform to the "Little-Endian" byte ordering scheme (although the norm for network-oriented applications is the "big-Endian" ordering scheme). Thus, all client applications running on Intel-based chipsets should not face any problems communicating correctly back and forth control messages with the IOCP. The same however is not true for clients running under chipsets that utilize the "Big-Endian" ordering (i.e. Motorola, PowerPc, etc). In that case, the client application must convert the multi-byte numerical values from Big to Little Endian when sending a control message to the IOCP and convert them back from Little to Big Endian when receiving a control message from the IOCP; otherwise the IOCP will deem that client's control messages as invalid.

Q: What byte-values (in decimal notation) the SHA1 algorithm will produce for the following authentication tokens?

User Name: GEORG1801

Password: gemini9

IP Address: 172.16.2.31

Random number: 13450000

A: The SHA1 message digests, relevant to the aforementioned authentication tokens, are given in Table 10-1:

Array Position	userNameHash (in decimal)	userPasswordHash (in decimal)
0	44	128
1	143	44
2	206	6
3	95	11
4	150	127
5	91	11
6	208	242
7	245	214
8	66	153
9	117	140
10	173	129
11	106	238
12	142	70
13	233	207
14	220	129
15	14	52
16	99	151
17	171	87
18	99	87
19	238	238

Table 10-1, User Name and Password SHA1 message digest values