



# **FIX Gateway for TOCOM trading system Configuration and Management Guide**

Date: 05 January 2004  
Version: 1.0 (Release)

Orchid Technology K.K.  
5F. Landic II Toranomom Bldg., 3-7-8 Toranomom, Minato-ku, Tokyo 105-0001 Japan

*This document contains information proprietary to Orchid Technology K.K. and may not be reproduced, disclosed or used in whole or part without express permission of Orchid Technology K.K.*

Orchid Technology K.K., by publishing this document, does not guarantee that any information contained herein is and will remain accurate or that use of the information will ensure correct and faultless operation of the relevant service or equipment. Orchid Technology K.K., its agents and employees shall not be held liable to or through any user of this document for any loss or damage whatsoever resulting from reliance on the information contained herein.

This document refers to the products and trademarks of manufacturers. Acknowledgement is made of all trademarks, registered trademarks and trading names that are referred to in the text.

© 2004 Orchid Technology K.K. All rights reserved.

Published by  
Orchid Technology K.K.

## Table of Contents

1. Preface.....	4
2. Prerequisites .....	5
3. Introduction.....	6
3.1 Purpose .....	6
3.2 Scope .....	6
3.3 Functional overview .....	6
4. Gateway operation .....	8
4.1 Startup .....	8
4.2 End of day procedure and termination .....	8
4.3 Crontab file sample .....	8
5. Configuration management.....	9
5.1 Configuration file format overview.....	9
5.2 Active service section .....	9
5.3 Service parameters context.....	9
5.4 Exchange connectivity .....	11
5.4.1 Relay terminal description section .....	12
5.4.2 Relay Terminal settings context .....	12
6. Client side integration .....	13
6.1 Sequence number management.....	13
6.2 Implemented order types .....	13
6.3 Implemented instrument types .....	13
7. Reference documentation.....	14

## **1. Preface**

This document provides detailed information about configuration and deployment aspects of FIX gateway for TOCOM exchange. Following sides of the system are presented:

- Command-line parameters, startup/shutdown, signals
- Configuration file syntax and parameter reference
- Client-side integration (FIX protocol compliance)
- Persistent state management
- Exchange connectivity

## **2. Prerequisites**

In order to address the topics, described in this document, reader should be familiar with the following:

- FIX Protocol version 4.2
- TOCOM Orders & Execution system architecture, protocols and client side infrastructure maintenance

General knowledge of UNIX administration is highly desirable. Reference documentation is presented in reference section.

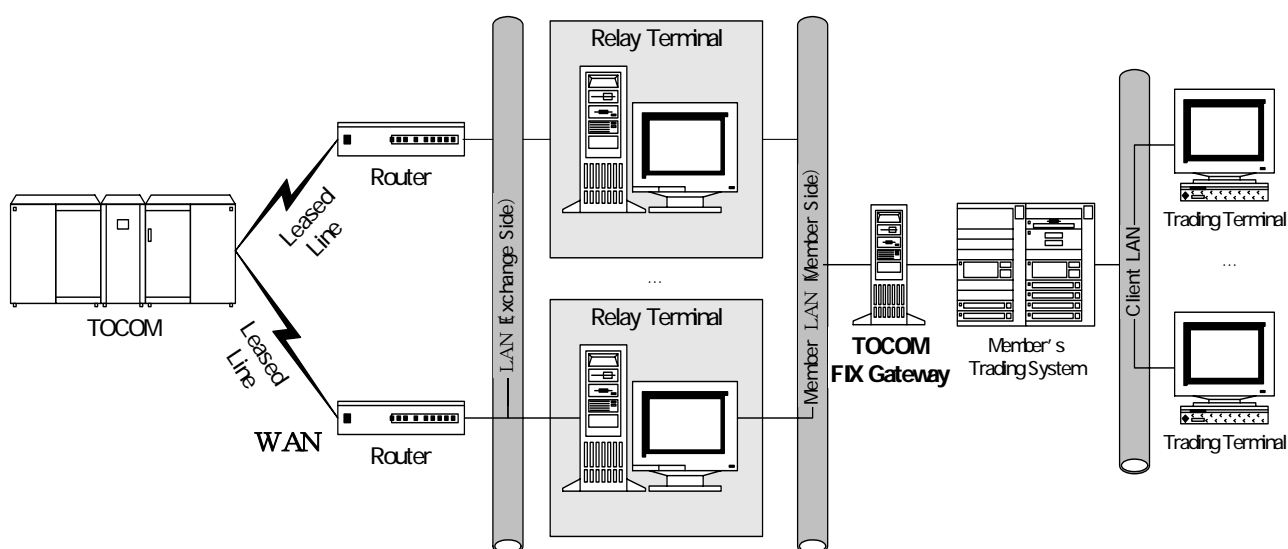
## 3. Introduction

### 3.1 Purpose

FIX Gateway for TOCOM Orders & Execution system was developed to provide standard interface to proprietary commodities crossing engine developed by NTT Data. Interface is based on industry-standard FIX protocol version 4.2 deployed via transport level of TCP/IP stack.

### 3.2 Scope

Following diagram depicts how TOCOM FIX gateway fits into overall trading infrastructure.



TOCOM trading system employs 3-tier client server model, where a number of Relay Terminals, installed at member site behave as intermediate proxies. Thus FIX gateway can interface arbitrary number of Relay Terminals. Every logical session with particular relay terminal is comprised of two transport connections: one for order input and the other for asynchronous notice reception.

### 3.3 Functional overview

FIX gateway provides trading service by means of single FIX connection for business-level interactions with Member system. Order stream is then de-multiplexed to place uniform distribution of load among all available relay terminal connections.

When transaction confirmations and other business notifications are received from TOCOM center, they are processed and forwarded to Member System. In event of temporary member's trading system disruption, all incoming events are collected in the Gateway database for subsequent delivery to client when connection resumes.

Similar approach is used for incoming orders: all are queued on gateway level whilst previously input orders are still in transit. However in case when all exchange connections go down for whatever reason, all queued orders are immediately rejected to avoid outdated orders input.

Every event is registered in external persistent database to restore proper event sequence in case of the gateway shutdown. This allows proper resumption and smooth processing of outstanding orders upon restart. When event sequence is completed (ex: Order is fully executed), it is removed from volatile memory, leaving a trace in persistent database.

Additionally, persistent database is used to recover missed data in case of exchange line failure. When failed line is eventually restored, the database is used to figure out lost notices and request retransmission.

TOCOM exchange does not explicitly notify members about the status of non-executed orders after trading day is over, assuming that member would naturally disregard outstanding orders upon trading completion. This feature is implemented on TOCOM FIX Gateway level and triggered upon reception of special signal from administrator or signal system. When this command is executed, 'Done For Day' execution report is generated for all outstanding orders, explicitly delivering the notification to member trading system to change the state of pending orders. This does not however modify gateway's database and order states for extra protection against erroneous command. Thus if the command is executed by mistake in the middle of trading day, following events from the exchange will be processed naturally and forwarded to member trading system without any regression (member trading system should be prepared for this).

## 4. Gateway operation

The gateway is normally started every morning before trading is opened and terminated after trading is over. Before terminating, optional end-of-day procedure can be triggered by sending SIGUSR1 signal to running instance. This will simulate done-for-day execution reports for all outstanding orders. Preferred way to perform these operations is via cron daemon, configured to start these task at specified times.

### 4.1 Startup

Application's configuration is loaded from external configuration file and command line parameters are kept simple:

`./TOCOM_FIX_GW_SunOS_5.8_sparc -conf <conf-file> <Active-Section>`, where

- **<conf-file>**: path to configuration file to read. Path can be either absolute or relative to the directory the application is started from.
- **<Active-Section>**: Primary function of this parameter is to identify top-level section in the configuration file that should be used for instance configuration. To simplify configuration management, the same file can hold the parameters for different instances or configurations to switch between (ex: Test vs. Live configuration). Additionally this parameter is used for naming of instance-specific files such as log, persistent and pid files.

On Startup, the gateway loads requested configuration and performs consistency checks. If an error is found in the configuration, it is reported to console/log file and application exits. Otherwise application is detached from controlling terminal and regular processing starts with listening for FIX client connections and simultaneously opening requested relay terminal connections.

### 4.2 End of day procedure and termination

End of day procedure is triggered by sending SIGUSR1 signal to running instance. Upon successful startup, pid file named **<Active-Section>.pid** is created in execution directory. It holds textual representation of instance's process identifier and can be used to communicate asynchronous interrupts to running instance. Similarly SIGINT and SIGTERM signals are used to terminate the instance. This is suggested way to stop the gateway.

### 4.3 Crontab file sample

```
45 15 * * 1-5 cd /export/home/TOCOM_FIX_GW ; ulimit -c unlimited ;
bin/TOCOMFIXGW_gcc_Linux_2_4_21_4_EL_i686_r -conf conf/member.conf TOCOM_Member
50 15 * * 1-5 cd /export/home/TOCOM_FIX_GW ; kill -USR1 `cat TOCOM_Member.pid`
55 15 * * 1-5 cd /export/home/TOCOM_FIX_GW ; kill -INT `cat TOCOM_Member.pid`
~
```

## 5. Configuration management

### 5.1 Configuration file format overview

Configuration file is comprised of a number of sections, where a section can contain other sections as well as configuration parameters. Section is represented by its name followed by figure brackets identifying section content. Parameters are always appear within a section (parameter's context) in form *Parameter\_Name=Value*, where the Value is parameter-specific and can be either identifier, string, or a list.

In order to avoid information duplication, parameters together with their context can be moved from inner level into outer level to extend their scope. For instance, port numbers for orders and notices connections are usually configured to be the same across installations: 20001 for notices and 20002 for orders. If we have been allocated several relay terminals with above mentioned port allocation scheme, we could avoid information duplication by moving corresponding parameters to upper context, visible from the levels, describing different relay terminals.

Parameter search is performed starting from inner and expanding to outer contexts. Search is stopped with successful status if all required parameters are found; otherwise missed parameter is reported to the log file accompanied with last searched context information. All parameters are considered mandatory unless stated otherwise.

### 5.2 Active service section

As mentioned earlier, the second command-line parameter for starting an instance is a name of top-level section in configuration file, defining all parameters for particular instance. Once configuration tree is loaded, top-level context is searched for the name of that section. If not found, error is reported and application terminates, otherwise configuration parsing and consistency checking is being performed. Instance-specific top-level section is referred to as *startup section*.

### 5.3 Service parameters context

This section describes global instance parameters. Startup section followed by surrounded context is searched for *Service\_Params* section to collect service-global parameters. Corresponding parameters reference is presented in following table:

<i>Parameter Name</i>	<i>Value type</i>	<i>Description</i>
Log_Path	String	Directory, where log files are to be collected. If relative path is specified, it is interpreted as starting from application startup directory.
Persistent_Directory	String	Directory, where persistent files are to be collected. If relative path is specified, it is interpreted as starting from application startup directory.

<i>Parameter Name</i>	<i>Value type</i>	<i>Description</i>	
Sync_Order_Book	Boolean	Whether to sync file operations with persistent Order Book. <b>TRUE</b> gives extra reliability at expense of performance as every order book update is to be flushed to disk before going further. <b>FALSE</b> is better in terms of performance, but Order Book might get corrupted in case of OS failure.	
Local_Comp_ID	String	FIX-specific. Local company identifier to be used in FIX traffic.	
Local_Sub_ID	String, Optional	FIX-specific	
Local_Location_ID	String, Optional	FIX-specific	
Trace_Level	List of identifiers	Parameter controls logging depth. Every specified identifier designates single category of log messages to be output. Although the amount of information logged is quite configurable, it is suggested to collect maximum information unless it has negative effects such as performance or disk space issues. Following values are supported:	
		<b>FIX_SentData</b>	All FIX messages dump, originated from the gateway.
		<b>FIX_ReceivedData</b>	All FIX messages dump, received from member trading system.
		<b>FIX_API</b>	Status information from the FIX engine.
		<b>TOCOM_Data_Sent</b>	Hexadecimal dump of the data, sent to Relay Terminal.
		<b>TOCOM_Data_Received</b>	Hexadecimal dump of the data, received from Relay Terminal.
		<b>Debug</b>	Extra information about low-level gateway operation.
		<b>Info</b>	Informative messages.

<i>Parameter Name</i>	<i>Value type</i>	<i>Description</i>	
		<b>Warning</b>	System warnings.
		<b>Fatal</b>	Abnormal events.
		<b>Config_Search_Status</b>	Only useful during configuration file tweaking. Dumps extra information about parameters being searched and corresponding contexts.
Listen_Port	Number	TCP port to be used to accept FIX connection from member's trading system.	
LIBFIX_MaxRecoverBuffers	Number	Number of messages FIX engine will store in internal queue during sequence number mismatch recovery. Stored messages will minimize number of resend requests that follows the recovery.	
LIBFIX_LogonTimeout	Number	Time interval in seconds during which normal FIX logon procedure should be completed.	
Order_Input_Method	Identifier	Relay terminal provides two types of order input:	
		<b>Quick</b>	Orders are sent in quick succession and buffered on relay terminal level. This is quickest input method as it saves on local traffic (before relay terminal), but in case of exchange line failure users will know the status of sent orders only after line restoration.
		<b>Standard</b>	Next order is only sent when previous one is acknowledged from exchange. This guarantees that the status of maximum one order will be uncertain before line is restored. All buffered orders on gateway level will be rejected immediately unless there is at least one active line capable of order input.

## 5.4 Exchange connectivity

Single gateway instance can handle arbitrary number of Relay Terminal connections. To describe the variety of available connections, following scheme has been used. Startup section is required to contain single parameter **Active\_Sessions**, which value should be a list of section identifiers, one

per available relay terminal. These identifiers are primary used to collect configuration information about particular connection, but also serve as marks in the log file to identify connection-specific information. For every section name listed, corresponding context (relay terminal description section) should be presented within startup section.

### 5.4.1 Relay terminal description section

This section is named according to user preference and listed in the **Active\_Sessions** enumeration. It's primary function is to be a placeholder for relay terminal-specific configuration information. It has to contain single section named **Relay\_Terminal\_Settings**.

### 5.4.2 Relay Terminal settings context

This section describes relay terminal-specific connection parameters. Relay terminal description section followed by surrounded contexts is searched for **Relay\_Terminal\_Settings** section to collect required parameters. Corresponding parameters reference is presented in following table:

<i>Parameter Name</i>	<i>Value type</i>	<i>Description</i>
Address	String	IP address of the target relay terminal.
Healthcheck_Interval	Number, Default value: 60	Relay Terminal health-check monitoring interval in seconds. If heartbeat message hasn't been received for specified interval, relay terminal is considered as malfunctioned and the connection is severed.
UserID	String, Optional	User ID, registered on relay terminal. If this parameter is omitted, spaces are used.
Password	String, Optional	User password, registered on relay terminal. If this parameter is omitted, spaces are used.
Reconnection_Timeout	Number	If the connection with relay terminal is severed for whatever reason, this parameter controls when to start recurring logon operation.
Port_Notices	Number	TCP port number on relay terminal side to connect to establish notice session.
Port_Orders	Number	TCP port number on relay terminal side to connect to establish order session.

## **6. Client side integration**

FIX implementation is expected to be compliant with industry-standard FIX protocol version 4.2. Implementation-specific issues are highlighted in the following paragraphs.

### **6.1 Sequence number management**

Sequence numbers are naturally reset at the beginning of new trading day and preserved during the same day. Latest sequence number pair is stored in instance-specific file in persistent day. New file is created for every business day. Similarly all outgoing FIX messages are kept to fulfill possible resend requests on FIX session.

### **6.2 Implemented order types**

Implementation supports new order messages as well as cancellations. Revisions are not implemented due to absence corresponding functionality in exchange protocol. Thus all incoming revisions are rejected with Cancellation/Revision reject message. All supported execution conditions fully correspond to those implemented on TOCOM.

### **6.3 Implemented instrument types**

Futures and options are supported. Spread contracts and evening session are not supported as being still unimplemented on exchange side.

## **7. Reference documentation**

1. FIX protocol version 4.2, available from [www.fixprotocol.org](http://www.fixprotocol.org)
2. TOCOM trading system reference documentation
3. TOCOM FIX Gateway error code description document
4. Execution ID encoding document