



Euronext E-TCS Access Solution

Specifications for a new Internet tool for TCS

STATUS

Version 1.2 10 March 2008

INTRODUCTION

The purpose of this document is to describe the new Internet access to Euronext's TCS system, provided by Euronext, and the way that the application messages are exchanged between members' applications and the TCS system.

The new means of access consists of two types of services:

- Access in real-time to TCS services through web services via the Internet (real-time mode)
- The enabling of the trade reporting file, based on the TCSBox located inside the member's Information System.

Both real-time and file mode use SOAP through HTTP over SSL.

Note

The Markets in Financial Instruments Directive is constantly evolving. Euronext reserves the right to update these documents at short notice as necessary, should changes become mandatory in order to meet new requirements arising from MiFID.

Website: www.euronext.com/cashmembers

CONTENTS

1	ARCHITECTURE.....	4
2	WEB SERVICES - E-TCS IN REAL-TIME MODE	5
2.1	TECHNICAL REQUIREMENTS	5
2.2	TCS WEB SERVICE	5
2.3	SOAP REQUEST EXAMPLE (SEND MESSAGES).....	8
2.4	SOAP RESPONSE EXAMPLE	9
2.5	AUTHENTICATION AND NON-REPUDIATION	10
2.6	WEBSERVICE RESPONSE	10
3	TCSBOX – E-TCS IN FILE MODE	11
3.1	INSTALLATION	11
3.2	TECHNICAL REQUIREMENTS	11
3.3	INTRODUCTION.....	12
3.4	HOW TO POST A NEW FILE.....	13
3.5	POSTED FILE CONTENT	13
3.6	MANAGE TCSBOX RESPONSES.....	13
3.7	FILE STATUS WORKFLOW	14
3.8	INPUT AND OUTPUT CSV FILE FORMATS.....	15
3.9	INPUT AND OUTPUT XML FILE FORMATS.....	19

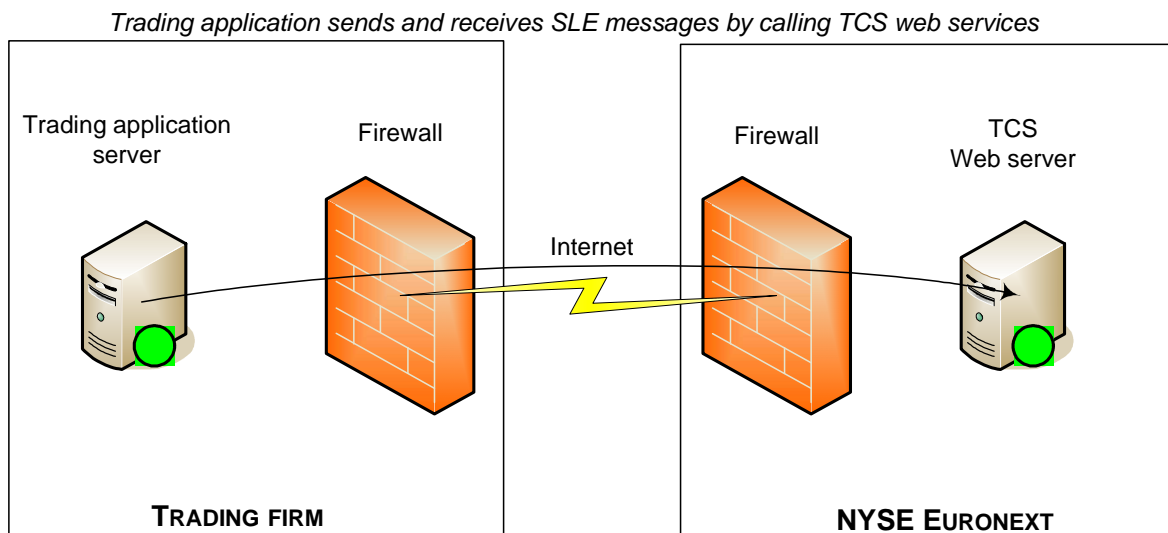
1 ARCHITECTURE

In order to use the TCS services, client applications must be capable of calling a SOAP-based (Simple Object Access Protocol) web service through HTTP over SSL, as well as being capable of generating and parsing XML documents.

The message format conforms to the SOAP 1.1 standard for exchanging XML-based messages, typically across a network using HTTP as the transport mechanism. SOAP, therefore, provides the messaging framework that web services can build upon.

The web service can only be accessed through HTTP over SSL. Encryption of this transport ensures data integrity.

The services in TCS are also protected by authentication and authorization mechanisms. To access the services, a client application is required to log on to the system with a valid subscriber ID. The log on request creates a session on the server.



For the best interoperability, TCS web services' binding is **"document"**. Input and output are encoded in **"literal"**.

2 Web Services - E-TCS in real-time mode

The new TCS web services provide a new and powerful means of access for receiving TCS messages. It is possible to send several messages at the same time and receive synchronous responses.

A full SDK (written in PHP and Java) is available to download at the following address: <https://tcs-h.euronext-net.com/register/>. It shows how to write programs capable of sending (and receiving) messages to TCS in few lines.

2.1 Technical requirements

If you use proxy to access the Internet, the web server must be started with additional system properties for proxy configuration:

- -Dhttp.proxyHost=*my.proxy.host*
- -Dhttp.proxyPort=*8090*
- -Dhttp.proxyUserName=*user*
- -Dhttp.proxyPassword=*password*

The proxyPort, proxyUsername and proxyPassword properties are optional.

➔ Your J2EE web server must be able to access TCS web services via the Internet in order to send messages: Protocol HTTPS (port 443).

2.2 TCS web service

WSDL of TCS web service is accessible at the following URL:

<https://tcs-h.euronext-net.com/ws/services/Tcs?wsdl>

2.2.1 Send messages

Method to call: `sendMessages`

Input: `MessageSet`

Output: `array of WebServiceResponse`

To send one or more messages simultaneously to TCS, you simply have to call the `sendMessages` method. The only argument consists in a `MessageSet` object. A `MessageSet` contains two attributes: an array of strings (each string must contain a well formed SLE message, base64 encoded) and a processing method parameter (not used yet).

The authorized size of the array is fixed at 100 SLE messages.

SLE messages format documentation (Messages exchanged between MMTP SLEs and the TCS application) is available online at this address:

<http://www.euronext.com/forourclient/mrdoc/general/wide/mrDoc-3485-EN.html>

Accepted messages:

- 0402 Declaration cancellation request
- 0403 Declaration refusal request issued by the counterparty
- 0404 TCS trade cancellation request
- 0425 NAV+/- entry (for Dutch funds only)
- 0428 Trade entry
- 0441 (MiFID compliance) Declaration

For each message sent, the response array will contain the corresponding SLE message response. For example, for a 0428, the response will contain a 0429 (acknowledgement) or a 0418 (rejection).

For each call, authentication is mandatory and consists of a part of the SOAP header. The full description of these methods is available in the sub-chapters below.

2.2.2 Get messages count

This method returns the number of messages in your queue (acknowledgements and unsolicited notices). It is not necessary to call `getMessages` if the response to `getMessagesCount` equals zero.

Method to call: `getMessagesCount`

Input: none

Output: A positive long (0 → n)

2.2.3 Get messages

By calling this method, you will receive your messages. Received messages are removed from the queue.

You need to call `getMessagesCount` and `getMessages` when you are waiting for unsolicited messages (like an elimination notice for example).

If you send only some 0428, the synchronous response (0429 or 0418) is totally normal and sufficient.

Method to call: `getMessages`

Input: none

Output: array of SLE messages (base64 encoded strings)

Possible type of message:

- 0411 Declaration notice
- 0412 Notification of a declaration issued by the counterparty
- 0413 Declaration cancellation notice
- 0414 Declaration refusal notice
- 0415 Matching notice
- 0416 Elimination notice for declaration
- 0429 Trade reception notice
- 0417 TCS trade cancellation notice
- 0418 Rejection notice
- 0419 Start of TCS trading session notice
- 0420 End of TCS trading session notice
- 0426 NAV+/- acknowledgement (for Dutch funds only)

2.3 SOAP request example (send messages)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:ns2093="http://testuri.com">
  <SOAP-ENV:Header>
    <AuthenticationToken>
      <Username>scott</Username>
      <Password>tiger</Password>
    </AuthenticationToken>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <sendMessages xmlns="http://tcs.ws.exchange">
      <messageSet>
        <messages>
          <string>...[base64 encoded message]...</string>
        </messages>
        <processingMethod>0</processingMethod>
      </messageSet>
    </sendMessages>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.4 SOAP response example

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <sendMessagesResponse xmlns="http://tcs.ws.exchange">
      <responses xmlns="http://tcs.ws.exchange">
        <WebServiceResponse>
          <message>...</message>
          <returnCode>0</returnCode>
        </WebServiceResponse>
      </responses>
    </sendMessagesResponse>
  </soap:Body>
</soap:Envelope>
```

2.5 Authentication and non-repudiation

For the moment, each SOAP message must contain a SOAP header. The SOAP header must contain an *Authentication Token* built as follow:

```
<AuthenticationToken>
  <Username>...</Username>
  <Password>...</Password>
</AuthenticationToken>
```

IMPORTANT: a stronger authentication coupled with non-repudiation system is currently in development. It will be documented and published in due time.

2.6 WebServiceResponse

When a set of one or more messages is processed successfully by the TCS engine, the return code is equal to zero. Otherwise, it contains an error number.

Description of the object *WebServiceResponse*:

Field name	Description	XSD Type
ReturnCode	Return code	Integer
Message	Error/rejection message (base64 encoded)	String

Possible values for return code:

Value	Signification
0	Success. Field <i>message</i> contains result notice. For example, if you send a 0441, <i>message</i> will contain one 0411 or one 0412. For a 0428, it will contain one 0429, etc.
1	Unexpected error. This kind of error occurs when the authorized length for a message is exceeded. Error message appears in <i>message</i> .
2	TCS engine rejection. Field <i>message</i> contains one 0418.

3 TCSBox – E-TCS in file mode

Using the TCSBox, members can report easily to the market.

Members have to build files with some Trade entry messages in XML or in CSV format. Definitive file format will be provided within the application.

Files posted into the TCSBox *inbox* directory are processed automatically. TCS responses are posted into the TCSBox *outbox* directory. TCS responses consist of acknowledgments and rejections.

The TCSBox is available to download at the following address:

<https://tcs-h.euronext-net.com/register/>.

The sub-chapters below describe precisely how the TCSBox can be installed and works.

3.1 Installation

Technical requirements are provided below. Please refer to your web server documentation for deploying the war packaged application.

3.2 Technical requirements

The following configuration is required to run the TCSBox:

- JDK 1.5.x or later
- 64 MB of free disk space
- Standard J2EE web server:
 - o Tomcat
 - o Weblogic
 - o Websphere
- A user with an *admin* role is required to access the TCSBox.

If you use proxy to access the Internet, the web server must be started with additional system properties for proxy configuration:

- -Dhttp.proxyHost=*my.proxy.host*
- -Dhttp.proxyPort=*8090*
- -Dhttp.proxyUserName=*user*
- -Dhttp.proxyPassword=*password*

The proxyPort, proxyUsername and proxyPassword properties are optional.

➔ Your J2EE web server must be able to access TCS web services through the Internet in order to send messages: Protocol HTTPS (port 443).

3.3 Introduction

This chapter describes how the TCSBox file mode works.

For the purposes of this example, we assume that the TCSBox is deployed under following directory:

Windows:

```
C:\tcsbox\
```

Unix/Linux:

```
/tcsbox/
```

In this chapter, this TCSBox root directory is named `${tcsbox.home}`.

3.4 How to post a new file

Directory location: `$(tcsbox.home)/WEB-INF/data/public/inbox/`

First, generate or copy your file into the **inbox** directory. When your data file is ready, rename it with the **“.in”** suffix. TCSBox will start to process all ***.in** files automatically.

→ Each posted file must have a unique name and the **“.xml”** or **“.csv”** extension

→ When a posted filename ends with **“.in”** that means that the TCSBox is allowed to process it.

For example, a file named **“trades.xml”** must be renamed **“trades.xml.in”** to be processed.

3.5 Posted file content

The TCSBox handles two file formats: XML and CSV. XML file description will be provided within the TCSBox application. All other formats are rejected and corresponding error log files are created in the **outbox** directory.

Independently of file format, all fields described in the SLE documentation will be covered.

3.6 Manage TCSBox responses

Directory location: `$(tcsbox.home)/WEB-INF/data/public/outbox/`

Use case: the following file **“trade.xml.in”** is posted into the **inbox** directory, the TCSBox processes the file and a timestamp is added into the filename in order to avoid file collision. For example **“trade.xml.in”** will be renamed **“timestamp.trade.xml.in”**.

→ Each processed file comes with its **.out** file in the **outbox** directory. Out files are normalized files, containing the following fields:

Field name
Internal member reference
Return code
Message
[...]
Trade reception notice fields
[...]

If the input file is an XML file, the output file will be an XML file.

In case of CSV, the same fields will be generated separated by semicolons.

If the return code is not zero, that means that input data are not compatible with the MMTP SLE messages. In the other case, the function code will contain the TCS engine response code.

3.7 File status workflow

Case: New and correctly formed file

Step	Explanation	Inbox content	Outbox content
1/6	File generation.	trade.xml	
2/6	File is ready and should be processed by the TCSBox.	trade.xml.in	
3/6	File is renamed with a timestamp	<i>timestamp</i> .trade.xml.in	
4/6	The TCSBox is processing the file in the inbox directory. At the same time, the corresponding output file is generated in the outbox directory.	<i>timestamp</i> .Trade.xml.lock	<i>timestamp</i> .trade.xml.lock
5/6	File has been processed. Input is renamed with ".done" suffix. Output file is renamed with ".out" suffix.	<i>timestamp</i> .trade.xml.done	<i>timestamp</i> .trade.xml.out
6/6	Processed file is moved to the inbox directory.		<i>timestamp</i> .trade.xml.out <i>timestamp</i> .trade.xml.done

Case: badly formed file (binary or corrupted content)

Step	Explanation	Inbox content	Outbox content
1/5	File generation.	trade.xml	
2/5	File is ready and should be processed by the TCSBox.	trade.xml.in	
3/5	File is renamed with a timestamp.	<i>timestamp</i> .trade.xml.in	
4/5	The TCSBox is processing the file in the inbox directory. At the same time, the corresponding output file is generated in the outbox directory.	<i>timestamp</i> .trade.xml.lock	<i>timestamp</i> .trade.xml.lock
5/5	TCSBox is not able to read file content. Input and output files are renamed with ".error" suffix. A log file will be generated. This file will contain error message(s).	<i>timestamp</i> .trade.xml.error	<i>timestamp</i> .trade.xml.error <i>timestamp</i> .trade.xml.error.log

This is the same behaviour for CSV file format, with the .csv extension file instead of the .xml extension.

3.8 INPUT AND OUTPUT CSV FILE FORMATS

3.8.1 Input format

The input CSV format must be made up of the fields of a 0428 Trade Entry, in the following order, separated by the “;” character. Each line corresponds to a Trade entry declaration.

Fields
Internal member reference
Declaring party principal code
Declaring party side
Quantity
Security code
Price
Order type
Counterparty member code
Counterparty principal code
Operation type indicator
Price notation
Other factors
Quantity notation
Venue Identification
Trading day
Trading time
Bypass controls indicator
Trade ID
Price multiplier
Deferred trade indicator
Settlement delay
Client Identification
Client Identification for a cross
Client code type Identification
Client code type Identification for a cross

3.8.2 Output format

The output CSV format is made up of the following fields, in the following order, separated by the “;” character.

Field	
Internal member reference	First part These fields indicate the result of the submission of the Trade entry.
Return code	
Message	
Internal Member Reference	Second part These fields are the Trade reception notice fields If an error code is returned (return code ≠ 0), there is no Trade reception notice data. Only separators ";" are present in this part.
Tcs Reference	
Declaring Party Principal Code	
Declaring Party Side	
Quantity	
Security Code	
Price	
Order Type	
Counterparty Member Code	
Counterparty Principal Code	
Time Trade Received	
Operation Type Indicator	
Price Notation	
Other Factors	
Quantity Notation	
Venue Identification	
Trading Day	
Trading Time	
Bypass Controls Indicator	
Trade ID	
Price Multiplier	
Deferred Trade Indicator	
Settlement delay	
Client Identification	
Client Identification for a cross	
Client code type Identification	
Client code type Identification for a cross	

Each line in the output file corresponds to one Trade entry response.
A line of Trade entry responses corresponds to the same line in the input file of Trade entry declaration.

3.9 Input and output XML file formats

3.9.1 Input format

The input XML format must be made up of the fields of a 0428 Trade Entry. The order of XML elements is not important.

Example with two Trade entry declarations:

```
<?xml version="1.0" encoding="UTF-8"?>
<RequestEntries>
  <TradeEntry>
    <InternalMemberReference>REFERENCE0000001</InternalMemberReference>
    <DeclaringPartyPrincipalCode></DeclaringPartyPrincipalCode>
    <DeclaringPartySide>A</DeclaringPartySide>
    <Quantity>12</Quantity>
    <SecurityCode>FR0000000010</SecurityCode>
    <Price>12.3456</Price>
    <OrderType>1</OrderType>
    <CounterpartyMemberCode>MEMBER1</CounterpartyMemberCode>
    <CounterpartyPrincipalCode></CounterpartyPrincipalCode>
    <OperationTypeIndicator>M</OperationTypeIndicator>
    <PriceNotation>EUR</PriceNotation>
    <OtherFactors>1</OtherFactors>
    <QuantityNotation>UNT</QuantityNotation>
    <VenueIdentification>VENUEIDENTI</VenueIdentification>
    <TradingDay>20070308</TradingDay>
    <TradingTime>120000</TradingTime>
    <BypassControlsIndicator>1</BypassControlsIndicator>
    <TradeID>TRADEID042800000000000000000001</TradeID>
    <PriceMultiplier>10000000000000000010</PriceMultiplier>
    <DeferredTradeIndicator>0</DeferredTradeIndicator>
    <SettlementDelay></SettlementDelay>
    <ClientIdentification></ClientIdentification>
    <ClientIdentificationForACross></ClientIdentificationForACross>
    <ClientCcodeTypeIdentification></ClientCcodeTypeIdentification>
    <ClientCodeTypeIdentificationForACross></ClientCodeTypeIdentificationForACross>
  </TradeEntry>

```

</RequestEntries>

3.9.2 Output format

3.9.3 The output XML format is made up of the same fields as for the CSV

Output format.

In the XML case:

- if there is no error, there is no Message XML element.
- if an error code is returned, the TradeReceptionNotice XML element is absent.

Each TradeResponse XML element in the output file corresponds to the response to the Trade entry declaration in the input file at the same index.

Example with two Trade entry responses:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseEntries>
  <TradeResponse>
    <InternalMemberReference>REFERENCE0000001</InternalMemberReference>
    <ReturnCode>0</ReturnCode>
    <TradeReceptionNotice>
      <InternalMemberReference>REFERENCE0000001</InternalMemberReference>
      <TcsReference>0000000010001786</TcsReference>
      <DeclaringPartyPrincipalCode></DeclaringPartyPrincipalCode>
      <DeclaringPartySide>A</DeclaringPartySide>
      <Quantity>12</Quantity>
      <SecurityCode>FR0000000010</SecurityCode>
      <Price>12.3456</Price>
      <OrderType>1</OrderType>
      <CounterpartyMemberCode>MEMBER1</CounterpartyMemberCode>
      <CounterpartyPrincipalCode></CounterpartyPrincipalCode>
      <TimeTradeReceived>175356</TimeTradeReceived>
      <OperationTypeIndicator>M</OperationTypeIndicator>
      <PriceNotation>EUR</PriceNotation>
      <OtherFactors>1</OtherFactors>
      <QuantityNotation>UNT</QuantityNotation>
      <VenueIdentification>VENUEIDENTI</VenueIdentification>
      <TradingDay>20070308</TradingDay>
      <TradingTime>120000</TradingTime>
      <BypassControlsIndicator>1</BypassControlsIndicator>
      <TradeID>TRADEID042800000000000000000001</TradeID>
      <PriceMultiplier>1000000000000000010</PriceMultiplier>
      <DeferredTradeIndicator>0</DeferredTradeIndicator>
    </TradeReceptionNotice>
  </TradeResponse>
</ResponseEntries>
```

```
<SettlementDelay></SettlementDelay>
<ClientIdentification></ClientIdentification>
<ClientIdentificationForACross></ClientIdentificationForACross>
<ClientCcodeTypeIdentification></ClientCcodeTypeIdentification>
</TradeReceptionNotice>
</TradeResponse>

<TradeResponse>
<InternalMemberReference>REFERENCE0000001</InternalMemberReference>
<ReturnCode>418</ReturnCode>
<Message>000120 - THE INSTRUMENT IS INVALID</Message>
</TradeResponse>
</ResponseEntries>
```

In this example, the first Trade entry response is correct (return code is 0) with Trade reception notice data (TradeReceptionNotice XML element), while the second Trade entry response has returned an error code (418) with an error message and no TradeReceptionNotice XML element.