

User Guide

**BCS**

**API**

**PROGRAMMER'S  
MANUAL**

Issue 3.7a | 31<sup>ST</sup> OCTOBER 2022



**EURONEXT CLEARING**

# TABLE OF CONTENTS

Revision History .....	5
<b>1. introduction.....</b>	<b>6</b>
<b>2. Connection to the BCS Clearing system .....</b>	<b>8</b>
<b>3. Configuration file .....</b>	<b>10</b>
<b>4. Type definitions.....</b>	<b>14</b>
4.1 GK_Reply_t.....	15
4.2 GK_MarketReply_t.....	17
4.3 GK_ClassType_t.....	18
4.4 GK_Status_t .....	18
4.5 GK_Chain_t .....	18
4.6 GK_Notification_t .....	18
4.7 GK_ApplicationData_t .....	19
4.8 GK_Callback_t .....	20
4.9 GK_Tag_t.....	20
4.10 GK_Data_t.....	20
4.11 GK_Transaction_t.....	20
4.12 GK_Subscription_t.....	20
4.13 GK_Inquire_t.....	21
4.14 GK_Context_t .....	21
4.15 GK_Connection_t.....	21
4.16 GK_Length_t .....	21
4.17 GK_Byte_t .....	21
4.18 GK_UnzipHelper_t .....	21
<b>5. Main callback functions .....</b>	<b>22</b>
5.1 GK_Initialize.....	23
5.2 GK_Terminate.....	23
5.3 GK_CreateContext .....	24
5.4 GK_Dispatch .....	25
5.5 GK_ReleaseContext .....	25
5.6 GK_Connect.....	26
5.7 GK_Disconnect .....	28
5.8 GK_CreateTransaction .....	28

5.9 GK_DestroyTransaction .....	29
5.10 GK_Submit .....	30
5.11 GK_Subscribe .....	31
5.12 GK_UnSubscribe .....	33
5.13 GK_Inquire .....	34
5.14 GK_GetVersion .....	36
5.15 GK_ConnectEx .....	36
<b>6. Introduction to Callbacks .....</b>	<b>39</b>
6.1 Connection request result.....	40
6.2 Disconnect request result.....	41
6.3 Connection monitoring .....	42
6.4 Application message submission result .....	43
6.5 Application message subscription result .....	43
6.6 Application message unsubscription result .....	44
6.7 Data inquiry request result.....	44
6.8 Data subscription notification.....	45
6.9 Data inquiry notification .....	45
<b>7. Retrieving data from callback objects.....</b>	<b>47</b>
7.1 Connection request result.....	48
7.2 GK_GetNotificationType .....	48
7.3 GK_GetConnectionStatus.....	48
7.4 GK_GetTransactionID .....	49
7.5 GK_GetMarketResponse .....	50
7.6 GK_GetSubscriptionID.....	51
7.7 GK_GetInquireID.....	51
7.8 GK_GetClassName .....	52
7.9 GK_DecodeData.....	53
7.10 GK_GetValueString .....	53
7.11 GK_GetValueLong.....	54
7.12 GK_GetValueDouble .....	55
7.13 GK_GetValueInt.....	55
7.14 GK_GetChain .....	56
7.15 GK_GetBinaryData.....	57
<b>8. Building application data messages.....</b>	<b>58</b>

8.1 GK_CreateApplicationData .....	59
8.2 GK_EncodeData .....	59
8.3 GK_SetValueString .....	60
8.4 GK_SetValueLong .....	60
8.5 GK_SetValueDouble.....	61
8.6 GK_SetValueInt .....	62
8.7 GK_DestroyApplicationData.....	62
8.8 GK_SetTransactionID .....	63
<b>9. Unzipping callback functions .....</b>	<b>64</b>
9.1 GK_CreateUnzipHelper .....	65
9.2 GK_DestroyUnzipHelper .....	66
9.3 GK_InitializeUnzipHelper .....	66
9.4 GK_ClearUnzipHelper .....	67
9.5 GK_UnzipBinaryData .....	67
<b>10. recovery .....</b>	<b>69</b>
10.1 Services.....	70
10.2 Subscribe.System.ServiceMarketStatus.....	71
10.3 Notify.System.ServiceMarketStatus.....	71
10.4 Recovery Simulation in CDS (Test) environment .....	73

## Revision History

Date	Version	Description	Author
Apr 2021	3.7	Euronext rebranding	Borsa Italiana
Oct 2022	3.7a	Rebranding	Borsa Italiana



# 1. INTRODUCTION



This document describes the main features of BCS API library (GKAPI). It is to be used in conjunction with the BCS API Data Layouts document in order to have an overview of how to interface the BCS Clearing system using the BCS API libraries.

The BCS API library provides developers with a set of callback functions which allows third party applications to correctly interface toward the BCS Clearing system, managing connections, transactions, subscriptions and notifications. It also defines operation types (Connect, Submit, Subscribe, etc.) and response types (CallBackConnect, CallBackSubscribe, CallBackData, etc..).

The BCS API library:

- is a thread-safe library;
- allows connections to the BCS Clearing System through one or more application servers;
- implements a proprietary protocol to exchange application data messages; it maintains a live connection until the client disconnection has been requested;
- manages configurable application windows;
- monitors the TCP/IP connection and alerts when connectivity problems arise;
- traces all working activities;

## 2. CONNECTION TO THE BCS CLEARING SYSTEM





In order to properly connect to the BCS Clearing System, a set of technical callback functions should be used. The following steps need to be executed before sending/receiving data:

- Initialize: this allows to initialize the BCS API library;
- Create Context: this allows to establish a physical connection to the specified application server of the BCS Clearing system; the Context Id returned by the callback should be used as an input parameter in any request sent to the system (Submit, Inquire, Subscribe, UnSubscribe, ...);
- Start a dedicated thread to manage Dispatch: this allows to handle callbacks as soon as an event raises; a thread should be created for each working context;
- Connect: this allows to start a communication session to the BCS Clearing system;
- Create Transaction: this allows to get a Transaction Id which has to be used in every Submit sent to the BCS Clearing system; if the system is still processing a submit request, it will reject any other submit request using the same Transaction Id, whereas it will accept requests with different Transaction Ids (previously received with a Create Transaction);

The following steps have to be executed in order to properly disconnect from the BCS Clearing system:

- Destroy Transaction: this allows to release all internal structures set up by the CreateTransaction function;
- Disconnect: this allows to disconnect from the BCS Clearing system;
- Release Context: this allows to release/destroy a working context;
- Terminate: this allows to release the BCS API library;

## 3. CONFIGURATION FILE



The BCS API library configuration file (GKApi.cfg) allows to define:

- the keep-alive message frequency;
- the application windows size;
- the application servers of the BCS Clearing system the BCS library should connect to;

The configuration file structure is defined as follows:

#### **[GENERIC\_SETTINGS]**

```
TRACE_FILE=.\GKApi.log      // Application messages trace output file.
TRACE_LEVEL=ERR            // ERR,WRN,INF,DBG
MESSAGES_FILE=.\GKMessages.cfg // Configuration file which contains
// debugging messages
CALLBACK_QUEUE_SIZE=1024   // Maximum number of queued call-backs
MAX_NUMBER_OF_CONTEXT=512 // Maximum number of contexts that can be
// created and used at the same time (this value
// depends on the number of available sockets)
```

#### **[GATEMARKET\_SERVERS]**

```
SERVER_LIST=METAMARKET01;METAMARKET02
// List of available application servers
```

#### **[METAMARKET01]**

```
TCP_IP= 213.92.93.177
TCP_PORT= 34900
KEEPALIVE_TIMEOUT=30      // Expressed in seconds
APPLICATION_WINDOW_SIZE=20000
// Maximum number of pending requests that can
// be managed at the same time for the current
// context.
TRACE_LEVEL=DBG          // ERR,WRN,INF,DBG
TRANSACTION_BUFFER_SIZE=20000
// Maximum number of parallel transactions to be
// preallocated and used by the GK-API.
// If exceeded, new resources will be allocated
// upon request
```

```

SUBSCRIPTION_BUFFER_SIZE=20000
// Maximum number of parallel subscriptions to
// be preallocated and used by the GK-API.
// If exceeded, new resources will be allocated
// upon request

INQUIRE_BUFFER_SIZE=20000 // Maximum number of parallel inquiries to be
// preallocated and used by the GK-API.
// If exceeded, new resources will be allocated
// upon request

TCP_BUFFER_SIZE=30000 // Maximum I/O buffer size expressed in
bytes.

[METAMARKET02]
TCP_IP=213.92.93.178
TCP_PORT=34900
KEEPALIVE_TIMEOUT=30 // Expressed in seconds
APPLICATION_WINDOW_SIZE=20000
// Maximum number of pending requests that can
// be managed at the same time for the current
// context.

TRACE_LEVEL=DBG // ERR,WRN,INF,DBG
TRANSACTION_BUFFER_SIZE=20000
// Maximum number of parallel transactions to be
// preallocated and used by the GK-API.
// If exceeded, new resources will be allocated
// upon request

SUBSCRIPTION_BUFFER_SIZE=20000
// Maximum number of parallel subscriptions to
// be preallocated and used by the GK-API.
// If exceeded, new resources will be allocated
// upon request

INQUIRE_BUFFER_SIZE=20000 // Maximum number of parallel inquiries to be
// preallocated and used by the GK-API.

```

```

// If exceeded, new resources will be allocated
// upon request
TCP_BUFFER_SIZE=30000 // Maximum I/O buffer size expressed in
bytes.
```

## 4. TYPE DEFINITIONS



The BCS API library manages the following data types:

- `GK_Reply_t`                      Reply code from each protocol session
- `GK_MarketReply_t`                Reply structure to handle returned events from previous requests
- `GK_ClassType_t`                    Application data layout type
- `GK_Status_t`                        Connection status types
- `GK_Chain_t`                        Types for controlling chains for snapshot information
- `GK_ApplicationData_t`            Type structure which contains application data to be sent
- `GK_Callback_t`                    Call-back generic structure
- `GK_Tag_t`                            User Tag returned by each call-back; (void\*)
- `GK_Data_t`                          Application data handle (long)
- `GK_Transaction_t`                Transaction identifier (long)
- `GK_Subscription_t`                Subscription identifier (long)
- `GK_Inquire_t`                      Inquire identifier (long)
- `GK_Context_t`                      Connection session identifier
- `GK_Connection_t`                Identifier of a communication channel with an application server. It is a socket corresponding to connection on a context
- `GK_Notification_t`                Call-back notification types
- `GK_Byte_t`                          Data type used for buffers containing binary data
- `GK_Length_t`                        Data buffer's size
- `GK_UnzipHelper_t`                Internal structure used to unzip binary compressed data

## 4.1 `GK_Reply_t`

Contains return code coming back from a protocol session. It is an enumerated type and may assume the following values:

- `GK_SUCCESS`                      Request successfully completed
- `GK_FAILED`                        Generic error. Usually returned by all functions that extract data from call-backs
- `GK_INVALID_CONFIG_FILE`        Configuration file not valid

- GK\_INVALID\_SERVER Application server not valid
- GK\_INVALID\_HANDLE Handle not valid
- GK\_API\_ERROR Internal API error
- GK\_API\_NOT\_INITIALIZED API not initialized
- GK\_API\_ALREADY\_INITIALIZED API already initialized
- GK\_INVALID\_CONTEXT Market context not valid
- GK\_SERVER\_UNREACHABLE Application server not reachable
- GK\_INVALID\_TRANSACTIONID Request refused. Transaction identifier not valid
- GK\_INVALID\_SUBSCRIPTIONID Request refused. Subscription identifier not valid
- GK\_COMMAND\_ON\_GOING Request refused. Request of the same type is still on going
- GK\_TYPE\_MISMATCH Attempting to read -a field using a wrong field-type.
- GK\_CONTEXT\_BUSY Context is busy whenever it is trying to connect to a context already in use
- GK\_MISSING\_CONNECTION A request has been sent before establishing a connection
- GK\_OVERLOAD The application window is full. The client application must wait for the completion of some previously issued requests before sending a new one
- GK\_INVALID\_PARAMETER Request refused. One or more supplied parameters are null or invalid.
- GK\_DATA\_ERROR Request refused. Supplied data are invalid or corrupted.
- GK\_MORE\_OUTPUT\_AVAILABLE Request successfully completed. More output space have to be provided to complete the whole operation.
- GK\_MORE\_INPUT\_NEEDED Request successfully completed. More input data are required to complete the whole operation.



## 4.2 GK\_MarketReply\_t

Contains return codes from a market gateway or clearing house system. It is an enumerated type and may assume the following values:

- GK\_REQUEST\_ACCEPTED Request accepted
- GK\_REQUEST\_REJECTED Request refused. Generic Error
- GK\_REQUEST\_WARNING Request has been accepted but a warning situation arises (e.g one of the contexts is not connected)
- GK\_ALREADY\_CONNECTED Connection already established
- GK\_INVALID\_MARKET Request refused. Market name is invalid
- GK\_INVALID\_CLASS Request refused. Class name is invalid
- GK\_NO\_MARKET\_CONTEXT Request refused. Connection has not been established
- GK\_INVALID\_FIELD Request refused. One of the class fields is invalid
- GK\_REQUEST\_ON\_GOING Request refused. A request of the same type is already pending
- GK\_LICENCE\_ERROR Maximum number of connections reached
- GK\_PROPOSAL\_ALREADY\_EXISTS A proposal on the same transaction already exists
- GK\_PROPOSAL\_NOT\_EXISTS A proposal on the transaction does not exist
- GK\_INVALID\_PROPOSAL\_KEY Invalid proposal referenced
- GK\_MISSING\_FIELD\_VALUE Mandatory field not set
- GK\_ACCESS\_DENIED User authentication completed unsuccessfully
- GK\_INSUFFICIENT\_PRIVILEGES Insufficient privileges
- GK\_WRONG\_FIELD\_VALUE A field contains a wrong value (e.g. Side field is different from Buy and Sell)
- GK\_SERVER\_NOT\_AVAILABLE Application server unreachable
- GK\_NOT\_CONNECTED Request refused. Connection not established
- GK\_WRONG\_PARAMETER Request refused. Some parameters are wrong (e.g. parameter non allocated, etc.)

- **GK\_TIMED\_OUT** Request refused. Client has been disconnected due to keep-alive timeout

### 4.3 GK\_ClassType\_t

Defines a class type and is an enumerated type and may assume the following values:

- **GK\_META\_CLASS** Meta-market application data layout, i.e. class type used for a market class that merges all differences among different market class into a single class
- **GK\_MARKET\_CLASS** Native market application data layout

### 4.4 GK\_Status\_t

Defines a market connection status. It is an enumerated type and may assume the following values:

- **GK\_CONNECTION\_UP** Connection is active
- **GK\_CONNECTION\_DOWN** Connection is broken
- **GK\_CONNECTION\_WARNING** Applicable to OnMarketStatus event only: this means that not all connections are active. Depending on the market, it means that the bandwidth is being reduced or, alternatively, that interaction with the market can be seriously damaged
- **GK\_SERVER\_DOWN** Connection lost from application server

### 4.5 GK\_Chain\_t

Defines a chain type of snapshot data coming from events. It is an enumerated type and may assume the following values:

- **GK\_CHAIN\_CONTINUE** New snapshot data can arrive
- **GK\_CHAIN\_END** Snapshot data are ended
- **GK\_CHAIN\_NO\_DATA** Snapshot data not available

### 4.6 GK\_Notification\_t

Defines notification types of call-backs. It is an enumerated type and may assume the following values:

- GK\_MARKET\_STATUS\_NOTIFICATION
- GK\_CONNECTION\_RESPONSE\_NOTIFICATION
- GK\_DISCONNECTION\_RESPONSE\_NOTIFICATION
- GK\_TRANSACTION\_STATUS\_NOTIFICATION
- GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION
- GK\_SUBMIT\_RESPONSE\_NOTIFICATION
- GK\_SUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_UNSUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_INQUIRE\_RESPONSE\_NOTIFICATION
- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION
- GK\_SET\_NOTIFICATION\_PERIOD\_NOTIFICATION
- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION

## 4.7 GK\_ApplicationData\_t

Defines the template of application messages to be sent to a market or clearing house system.

```
typedef GK_ApplicationData_t
(
    GK_ClassType_t classType,
    const char* className,
    const char* data
)
```

Fields can have the following values:

Type	Property Name	Description
<b>GK_ClassType_t</b>	ClassType	Class type or application data layout type (meta-market or market class)
<b>const char*</b>	ClassName	Class name
<b>const char*</b>	Data	Data layout in the format field=value

## 4.8 GK\_Callback\_t

Defines the template of call-backs.

```
typedef void (*GK_Callback_t)
(
    GK_Context_t context, // Context who did generate the event
    GK_Data_t    gkData,      // Data Handle
    GK_Tag_t     gkTag       // User Tag
)
```

## 4.9 GK\_Tag\_t

The user can assign a tag to each request. The call-back will return it to the caller.

```
typedef const void *    GK_Tag_t;
```

## 4.10 GK\_Data\_t

Data handle returned by the call-back. It can be used to find data coming from the call-back itself.

```
typedef long           GK_Data_t;
```

## 4.11 GK\_Transaction\_t

Transaction Id. This value has to be used in every Submit sent to the BCS Clearing system; if the system is still processing a submit request, it will reject any other submit request using the same Transaction Id, whereas it will accept requests with different Transaction Ids (previously received with a Create Transaction).

```
typedef long           GK_Transaction_t;
```

## 4.12 GK\_Subscription\_t

Subscription Id. This value identifies a Subscription sent to the BCS Clearing system.

```
typedef long           GK_Subscription_t;
```

## 4.13 GK\_Inquire\_t

Inquiry Id. This value identifies an Inquire sent to the BCS Clearing system.

```
typedef long      GK_Inquire_t;
```

## 4.14 GK\_Context\_t

Context Id. This value has to be used as an input parameter in any request sent to the system.

```
typedef long      GK_Context_t;
```

## 4.15 GK\_Connection\_t

Connection Id. This value identifies a socket connection to an application server. The client application must use it in the 'select' function to handle asynchronous events.

```
typedef int       GK_Connection_t;
```

## 4.16 GK\_Length\_t

Data buffer's size. Given a pointer to a data buffer, it defines how many elements of the buffer are significant starting from the element pointed to.

```
typedef unsigned int  GK_Length_t;
```

## 4.17 GK\_Byte\_t

Data type used for binary data buffers. It defines the data type of buffer elements used to store binary data.

```
typedef unsigned char  GK_Byte_t;
```

## 4.18 GK\_UnzipHelper\_t

Structure used to unzip binary compressed data. It is managed internally by the GK-API.

```
typedef void*         GK_UnzipHelper_t;
```

# 5. MAIN CALLBACK FUNCTIONS



The following sections describe all the BCS API callback functions.

## 5.1 GK\_Initialize

```
GK_Reply_t GK_Initialize(const ConfigFile);
                        char*
```

Parameters	ConfigFile	Pathname of the file which contains configuration parameters for the GK-API
Return values	GK_SUCCESS	Initialization has been successfully completed
	GK_INVALID_CONFIG_FILE	Initialization failure. Configuration file not found or corrupted
	GK_API_ERROR	Internal error
	GK_API_ALREADY_INITIALIZE	GK-API already initialized
	D	
	GK_INVALID_PARAMETER	<i>ConfigFile</i> is null
Description	This function must be called before any other GK-API function in order to initialize the GK-API library.	

## 5.2 GK\_Terminate

```
GK_Reply_t GK_Terminate();
```

Parameters	none	
Return values	GK_SUCCESS	Initialization has been successfully completed
	GK_API_NOT_INITIALIZED	API not initialized
Description	This function must be called in order to release the GK-API library.	

## 5.3 GK\_CreateContext

```
GK_Reply_t      GK_CreateContext (const serverName,
                                   char*
                                   GK_Context_t* pContext,
                                   GK_Connection_t* pConnection);
```

Parameters	<b>serverName</b>	Name of the application server through which connection must be set up (one from the list in SERVER_LIST in the configuration file)
	<b>pContext</b>	Working context identifier returned by the GK-API
	<b>pConnection</b>	Identifier of a socket connection to the application server. The client application must use it in 'select' function to handle asynchronous events
Return values	GK_SUCCESS	Context available, socket connection established
	GK_API_ERROR	Internal error
	GK_INVALID_SERVER	Application server name invalid (check if it is present in the configuration file)
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_INVALID_PARAMETER	At least one of <i>serverName</i> , <i>pContext</i> or <i>pConnection</i> is null
Description	This function must be called to establish a physical connection to the specified application server. A Context Id is returned. This identifier must be used in any other request sent to the BCS Clearing system (i.e. Submit, Inquire, Subscribe, UnSubscribe, ...). It is possible to create more than one context.	



## 5.4 GK\_Dispatch

```
GK_Reply_t      GK_Dispatch (context);
                (GK_Context_t
```

Parameters	<b>context</b>	Working context identifier
Return values	GK_SUCCESS	Dispatch successfully completed
	GK_API_ERROR	Internal error
	GK_INVALID_CONTEXT	Context not valid
	GK_API_NOT_INITIALIZED	API not initialized

**Description** This function is used to handle callbacks. GK\_Dispatch must be called as soon as an event raises from the working context. For this purpose, before calling GK\_Dispatch, call "select" API on the socket returned by GK\_CreateContext using a positive timeout values (i.e. not zero; usual timeout value is 5 seconds). Moreover, it is recommended to call GK\_Dispatch using a dedicated thread, one for each working context.

## 5.5 GK\_ReleaseContext

```
GK_Reply_t      GK_ReleaseContext (GK_Context_t context);
```

Parameters	<b>context</b>	Working context identifier
Return values:	GK_SUCCESS	Context successfully released.
	GK_API_ERROR	GK-API not initialized or internal error
	GK_INVALID_CONTEXT	Context not valid
	GK_API_NOT_INITIALIZED	GK-API not initialized

**Description** This function must be called to release/destroy a working context.

## 5.6 GK\_Connect

```
GK_Reply_t      GK_Connect context,
                (GK_Context_t
                 const char* userName,
                 const char* password,
                 const char* market,
                 GK_Callback_t pCallbackResponse,
                 GK_Callback_t pCallbackMarketStatus,
                 GK_Tag_t gkTag)
```

Parameters	<b>context</b>	Active context identifier through which a connection must be performed.
	<b>userName</b>	Name of the user requiring the connection
	<b>password</b>	Password of the user requiring the connection.
	<b>market</b>	Market or Clearing House name to which a connection is requested (e.g. MTA, CCG, ...)
	<b>pCallbackResponse</b>	Callback to handle a notification event for the related request.
	<b>pCallbackMarketStatus</b>	Callback to handle a notification event for the connection status
	<b>gkTag</b>	User tag returned by the callback
Return values:	GK_SUCCESS	Connection request successfully executed
	GK_API_ERROR	Internal error
	GK_INVALID_CONTEXT	Context is not valid
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_API_NOT_INITIALIZED	API not initialized
	GK_COMMAND_ON_GOING	A connection request is still on going and a notification event for the previous request must be received

GK_CONTEXT_BUSY	Context is already in use (a connection on the context is already in place)
GK_INVALID_PARAMETER	At least one of <i>userName</i> , <i>password</i> or <i>market</i> is null or too long

*from pCallbackResponse*

GK_REQUEST_ACCEPTED	Connection accepted
GK_REQUEST_REJECTED	Connection refused
GK_ALREADY_CONNECTED	Connection already in place
GK_INVALID_MARKET	MarketName is invalid
GK_ACCESS_DENIED	Unknown user
GK_LICENCE_ERROR	Maximum number of concurrent connections exceeded
GK_INSUFFICIENT_PRIVILEGES	User cannot connect to the specified market

*from pCallbackMarketStatus*

GK\_MARKET\_STATUS\_NOTIFICATION

- GK\_CONNECTION\_UP All connections are active
- GK\_CONNECTION\_WARNING At least one connection is active, while one or more other connections can be down
- GK\_CONNECTION\_DOWN No connection is active
- GK\_SERVER\_DOWN Application server not reachable

GK\_TRANSACTION\_STATUS\_NOTIFICATION

- GK\_CONNECTION\_UP Transaction is active
- GK\_CONNECTION\_DOWN Transaction is not active

GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION

- GK\_CONNECTION\_UP Subscription is active
- GK\_CONNECTION\_DOWN Subscription is not active

Description This function must be invoked to establish a connection to the BCS Clearing system.

## 5.7 GK\_Disconnect

```
GK_Reply_t      GK_Disconnect context,
                (GK_Context_t
                 GK_Callback_t pCallbackResponse,
                 GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>pCallbackResponse</b>	Call-back for request notification
	<b>gkTag</b>	User tag returned by the call-back

Return values:	GK_SUCCESS	Disconnection successfully completed
	GK_API_ERROR	Internal error
	GK_INVALID_CONTEXT	Context is not valid
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_API_NOT_INITIALIZED	API not initialized

*from pCallbackResponse*

GK_REQUEST_ACCEPTED	Connection accepted
GK_REQUEST_REJECTED	Connection refused
GK_NOT_CONNECTED	Connection not existing

Description This function must be invoked to release a connection to the BCS Clearing system.

## 5.8 GK\_CreateTransaction

```
GK_Reply_t      GK_CreateTransaction
                (GK_Context_t context,
                 GK_Transaction_t* pTransactionID);
```

Parameters:	<b>context</b>	Context identifier
	<b>pTransactionID</b>	Transaction identifier returned by the function

Return values	GK_SUCCESS	Transaction creation successfully completed
	GK_INVALID_CONTEXT	Context is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_INVALID_PARAMETER	<i>pTransactionID</i> is null

**Description:** This function must be invoked in order to create a transaction within the BCS Clearing system. A transaction is a physical connection between the client and the BCS Clearing system which allows handling fault detection and load balancing. The Transaction Id returned by this function has to be used in every Submit sent to the BCS Clearing system; if the system is still processing a submit request, it will reject any other submit request using the same Transaction Id, whereas it will accept requests with different Transaction Ids (previously received with a Create Transaction).

## 5.9 GK\_DestroyTransaction

*GK\_Reply\_t* **GK\_DestroyTransaction**  
 (*GK\_Context\_t* context,  
*GK\_Transaction\_t* transactionID);

Parameters:	<b>context</b>	Context identifier
	<b>transactionID</b>	Transaction identifier
Return values	GK_SUCCESS	Destroy transaction successfully completed
	GK_INVALID_TRANSACTIONID	Transaction identifier is not valid
	GK_INVALID_CONTEXT	Context not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	API not initialized
	GK_SERVER_UNREACHABLE	Server unreachable

Description: This function must be invoked to release all internal structures set up by the CreateTransaction function. It must be invoked before the GK\_Disconnect function.

## 5.10 GK\_Submit

```
GK_Reply_t GK_Submit (GK_Context_t context,
                      GK_Transaction_t transactionID,
                      GK_ApplicationData_t* applicationData,
                      GK_Callback_t pCallbackResponse,
                      GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>transactionID</b>	Transaction identifier
	<b>applicationData</b>	Application data layout to be executed. It can be built using proper functions (see below)
	<b>pCallbackResponse</b>	Callback to handle a notification event for that request.
	<b>gkTag</b>	User tag returned by the callback

Return values	GK_SUCCESS	Submit request successfully executed
	GK_INVALID_CONTEXT	Context not valid
	GK_API_ERROR	Internal error
	GK_INVALID_TRANSACTIONID	Transaction identifier is not valid
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_COMMAND_ON_GOING	A connection request is still on going and a notification event for the previous request must be received
	GK_OVERLOAD	Application window is exhausted. The caller must wait for completion of some previous accepted requests
	GK_INVALID_PARAMETER	<i>applicationData</i> is null



*GK\_Tag\_t gkTag,*  
*GK\_Subscription\_t\* pSubscriptionID);*

Parameters:	<b>context</b>	Context identifier
	<b>applicationData</b>	Application Data layout to be executed. It can be built using proper functions (see below)
	<b>pCallbackResponse</b>	Call-back to handle a notification event for that request.
	<b>pCallbackData</b>	Call-back to handle a notification event containing returned data.
	<b>gkTag</b>	User tag returned by the call-back
	<b>pSubscriptionID</b>	Unique identifier for the requested subscription
Return values	GK_SUCCESS	Subscription request successfully executed
	GK_INVALID_CONTEXT	Context not valid
	GK_API_ERROR	Internal error
	GK_INVALID_SUBSCRIPTIONID	Transaction identifier is not valid
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_OVERLOAD	Application window is exhausted. The caller must wait for completion of some previous accepted requests
	GK_INVALID_PARAMETER	At least one of <i>applicationData</i> or <i>pSubscriptionID</i> is null
	<i>from pCallbackResponse</i>	
	GK_REQUEST_ACCEPTED	Connection accepted
	GK_REQUEST_REJECTED	Connection refused
	GK_REQUEST_WARNING	Request accepted with some specified warnings
	GK_NO_MARKET_CONTEXT	The market or clearing house context is not available



GK_INVALID_FIELD	The specified field name is invalid
GK_MISSING_FIELD_VALUE	Mandatory field is empty
GK_INVALID_CLASS	Class not valid
GK_NOT_CONNECTED	Connection has not been set
GK_WRONG_PARAM	Wrong parameters passed

Description: This function must be invoked to send a Subscribe data structure to the BCS Clearing system.

## 5.12 GK\_UnSubscribe

```
GK_Reply_t    GK_UnSubscribe (GK_Context_t context,
                               GK_Subscription_t* pSubscriptionID,
                               GK_Callback_t pCallbackResponse,
                               GK_Tag_t gkTag);
```

Parameters: <b>context</b>	Context identifier
<b>pSubscriptionID</b>	Unique identifier for the requested subscription to be ended
<b>pCallbackResponse</b>	Call-back to handle a notification event for that request.
<b>gkTag</b>	User tag returned by the callback

Return values	
GK_SUCCESS	Unsubscribe request successfully executed
GK_INVALID_CONTEXT	Context not valid
GK_API_ERROR	Internal error
GK_INVALID_SUBSCRIPTIONID	Suscription identifier is not valid
GK_API_NOT_INITIALIZED	API not initialized
GK_SERVER_UNREACHABLE	Server unreachable
GK_COMMAND_ON_GOING	A connection request is still on going and a notification event for the previous request must be received
GK_OVERLOAD	Application window is exhausted. The caller must wait for

	completion of some previous accepted requests
<i>from pCallbackResponse</i>	
GK_REQUEST_ACCEPTED	Connection accepted
GK_REQUEST_REJECTED	Connection refused
GK_REQUEST_WARNING	Request accepted with some specified warning
GK_NO_MARKET_CONTEXT	The market or clearing house context is not available
GK_REQUEST_ONGOING	A previous submit operation on the same transaction identifier is still on going
GK_NOT_CONNECTED	Connection is not in place

**Description:** This function must be invoked to remove an active subscription. Subscriptions are not removed when a client application logs off via the GK\_Disconnect function.

## 5.13 GK\_Inquire

```

GK_Reply_t          GK_Inquire context,
                    (GK_Context_t
                    GK_ApplicationData_t* applicationData,
                    GK_Callback_t pCallbackResponse,
                    GK_Callback_t pCallbackData,
                    GK_Tag_t gkTag;
                    GK_Inquire_t* pInquiryID);

```

<b>Parameters:</b>	<b>context</b>	Context identifier
	<b>applicationData</b>	Application Data layout to be executed. It can be built using proper functions (see below)
	<b>pCallbackResponse</b>	Call-back to handle a notification event for that request.
	<b>pCallbackData</b>	Call-back to handle a notification event containing returned data.
	<b>gkTag</b>	User tag returned by the call-back

	<b>pInquiryID</b>	Unique identifier for the requested inquiry
Return values	GK_SUCCESS	Inquire request successfully executed
	GK_INVALID_CONTEXT	Context not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	API not initialized
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_OVERLOAD	Application window is exhausted. The caller must wait for completion of some previous accepted requests
	GK_INVALID_PARAMETER	At least one of <i>applicationData</i> or <i>pInquiryID</i> is null
	<i>from pCallbackResponse</i>	
	GK_REQUEST_ACCEPTED	Connection accepted
	GK_REQUEST_REJECTED	Connection refused
	GK_REQUEST_WARNING	Request accepted with some specified warnings
	GK_NO_MARKET_CONTEXT	The market or clearing house context is not available
	GK_INVALID_FIELD	The specified field name is invalid
	GK_MISSING_FIELD_VALUE	Mandatory field is empty
	GK_INVALID_CLASS	Class not valid
	GK_NOT_CONNECTED	Connection has not been set
	GK_REQUEST_ONGOING	A previous inquiry operation on the same transaction identifier is still on going
	GK_WRONG_PARAM	Wrong parameters passed
Description:	This function must be invoked to send an Inquire data structure to the BCS Clearing system.	

## 5.14 GK\_GetVersion

```
GK_Reply_t    GK_GetVersion(char** company,
                             char** version,
                             char** creationDate,
                             char** comment);
```

Parameters	<b>company</b>	Company that distributes the GK-API
	<b>version</b>	Version Identifier
	<b>creationDate</b>	Creation date
	<b>comment</b>	Any comment

Return values:	GK_SUCCESS	Request successfully executed
	GK_API_ERROR	Internal error

Description This function must be invoked in order to know the current GK-API version. The output parameters are allocated by the library and they must be released by the client application using the GK\_FreeString() function.

## 5.15 GK\_ConnectEx

```
GK_Reply_t    GK_ConnectEx (GK_Context_t context,
                             const char* userName,
                             const char* password,
                             const char* market,
                             const char* ClientIP,
                             const char* ClientData,
                             GK_Callback_t pCallbackResponse,
                             GK_Callback_t pCallbackMarketStatus,
                             GK_Tag_t gkTag)
```

Parameters	<b>context</b>	Active context identifier through which a connection must be performed.
	<b>userName</b>	Name of the user requiring the connection. Maximum allowed length: 40 characters.
	<b>password</b>	Password of the user requiring the connection. Maximum allowed length: 40 characters.
	<b>market</b>	Market or Clearing House name to which a connection is requested (e.g. MTA, CCG, ...). Maximum allowed length: 40 characters.
	<b>ClientIP</b>	IP address of the client host. It is sent to the server in order to univocally identify the client. Maximum allowed length: 15 characters.
	<b>ClientData</b>	Free text sent to the server for log purpose. Maximum allowed length: 512 characters.
	<b>pCallbackResponse</b>	Callback to handle a notification event for the related request.
	<b>pCallbackMarketStatus</b>	Callback to handle a notification event for the connection status
	<b>gkTag</b>	User tag returned by the callback
Return values:	GK_SUCCESS	Connection request successfully executed
	GK_API_ERROR	Internal error
	GK_INVALID_CONTEXT	Context is not valid
	GK_SERVER_UNREACHABLE	Server unreachable
	GK_API_NOT_INITIALIZED	API not initialized
	GK_COMMAND_ON_GOING	A connection request is still on going and a notification event for the previous request must be received
	GK_CONTEXT_BUSY	Context is already in use (a connection on the context is already in place)

GK\_INVALID\_PARAMETER      At least one of *userName*,  
*password*, *market*, *ClientIP* or  
*ClientData* is null or too long

*from pCallbackResponse*

GK\_REQUEST\_ACCEPTED      Connection accepted  
GK\_REQUEST\_REJECTED      Connection refused  
GK\_ALREADY\_CONNECTED      Connection already in place  
GK\_INVALID\_MARKET      Invalid MarketName  
GK\_ACCESS\_DENIED      Unknown user  
GK\_LICENCE\_ERROR      Maximum number of concurrent  
connections exceeded  
GK\_INSUFFICIENT\_PRIVILEGES      User cannot connect to the  
specified market

*from pCallbackMarketStatus*

GK\_MARKET\_STATUS\_NOTIFICATION

- GK\_CONNECTION\_UP      All connections are active
- GK\_CONNECTION\_WARNING      At least one connection is active,  
while one or more other  
connections can be down
- GK\_CONNECTION\_DOWN      No connection is active
- GK\_SERVER\_DOWN      Application server not reachable

GK\_TRANSACTION\_STATUS\_NOTIFICATION

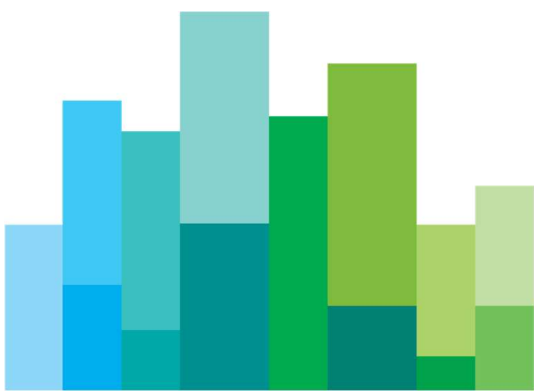
- GK\_CONNECTION\_UP      Transaction is active
- GK\_CONNECTION\_DOWN      Transaction is not active

GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION

- GK\_CONNECTION\_UP      Subscription is active
- GK\_CONNECTION\_DOWN      Subscription is not active

Description      This function must be invoked in order to establish a connection to the  
BCS Clearing system.

# 6. INTRODUCTION TO CALLBACKS



All callback functions have the following structure:

```
void Callback (GK_Context_t context,
               GK_Data_t gkData,
               GK_Tag_t gkTag);
```

The callback function is invoked by the GK-API to provide the calling application with asynchronous notifications that can contains data or connection monitoring information. The client application can define as many callbacks as required and then it can bind them to each single request by passing its pointer to the function call.

To know the notification type belonging to the callback, the client application must invoke the GK\_GetNotificationType() function in the callback itself, passing the gkData parameter.

The following notification types are available:

- GK\_MARKET\_STATUS\_NOTIFICATION
- GK\_CONNECTION\_RESPONSE\_NOTIFICATION
- GK\_DISCONNECTION\_RESPONSE\_NOTIFICATION
- GK\_TRANSACTION\_STATUS\_NOTIFICATION
- GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION
- GK\_SUBMIT\_RESPONSE\_NOTIFICATION
- GK\_SUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_UNSUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_INQUIRE\_RESPONSE\_NOTIFICATION
- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION
- GK\_SET\_NOTIFICATION\_PERIOD\_NOTIFICATION
- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION

After notification type detection, the calling application can invoke proper functions, as described below. It is possible (even if not recommended) to receive all notification events through a unique callback. It is recommended to process each received callback as soon as possible, in order to avoid disconnections due to keep-alive timeout.

## 6.1 Connection request result

```
void ConnectionResp context,
      (GK_Context_t
      GK_Data_t gkData,
```



```
GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the callback

Description The callback function pointer is passed to the connection request function. The GK-API will call the callback whenever it must notify connection result. If this callback function pointer is passed only to the connection request function, it will be possible to receive only notification of the GK\_CONNECTION\_RESPONSE\_NOTIFICATION type. In order to know the request result the GK\_GetMarketResponse() function must be invoked passing gkData.

## 6.2 Disconnect request result

```
void DisconnectionResp
      (GK_Context_t context,
       GK_Data_t gkData,
       GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the callback

Description The callback function pointer is passed to the disconnection request function. The GK-API will call the callback whenever it must notify disconnection result. If this call-back function pointer is passed only to the connection request function, it will be possible to receive only notifications of the GK\_DISCONNECTION\_RESPONSE\_NOTIFICATION type. In order to know the request result the GK\_GetMarketResponse() function must be invoked passing gkData.

## 6.3 Connection monitoring

```
void MarketStatus context,
(GK_Context_t
GK_Data_t gkData,
GK_Tag_t gkTag);
```

<b>Parameter</b>	<b>context</b>	Context identifier
<b>S:</b>		
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the callback

Description The callback function pointer is passed to the connection request function. The GK-API will call the callback whenever it must notify the market connection status. If this callback function pointer is passed only to the connection request function, it will be possible to receive notification of the following types:

- GK\_MARKET\_STATUS\_NOTIFICATION type
- GK\_TRANSACTION\_STATUS\_NOTIFICATION type
- GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION type

As regards the GK\_MARKET\_STATUS\_NOTIFICATION type, it will be possible to receive the following notifications:

- The GK\_CONNECTION\_UP status means all connections are active.
- The GK\_CONNECTION\_DOWN status means all connections are inactive.
- The GK\_CONNECTION\_WARNING status means at least one connection is active.
- The GK\_SERVER\_DOWN status means the connection to the server is lost.

In order to know the status value, the GK\_GetConnectionStatus() function must be invoked passing gkData.

As regards the GK\_TRANSACTION\_STATUS\_NOTIFICATION type it will be possible to receive the following notifications:

- The GK\_CONNECTION\_UP status means the related transaction is active.
- The GK\_CONNECTION\_DOWN status means the related transaction is inactive.

In order to know the related transaction identifier, the GK\_GetTransactionID() function must be invoked passing gkData.

As regards the GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION type it will be possible to receive the following notifications:

- The `GK_CONNECTION_UP` status means the related subscription is active.
- The `GK_CONNECTION_DOWN` status means the related subscription is inactive. In this case, the calling application should perform a new subscription from scratch.

In order to know the related subscription identifier, the `GK_GetSubscriptionID()` function must be invoked passing `gkData`.

## 6.4 Application message submission result

```
void SubmitResp (context,
                 (GK_Context_t
                  GK_Data_t gkData,
                  GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the callback

Description The callback function pointer is passed to the submit request function. The GK-API will call the callback whenever it must notify new results. If this callback function pointer is passed only to the submit request function, it will be possible to receive only notification of the `GK_SUBMIT_RESPONSE_NOTIFICATION` type. In order to know the submit result the `GK_GetMarketResponse()` function must be invoked passing `gkData`. On the other hand, to know the transaction identifier belonging to that submit the `GK_GetTransactionID()` function must be invoked passing `gkData`.

## 6.5 Application message subscription result

```
void SubscribeResp (GK_Context_t context,
                    GK_Data_t gkData,
                    GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event

**gkTag** User tag returned by the call-back

**Description** The callback function pointer is passed to the subscribe request function. The GK-API will call the callback whenever it must notify new results. If this callback function pointer is passed only to the subscribe request function, it will be possible to receive only notification of the GK\_SUBSCRIBE\_RESPONSE\_NOTIFICATION type. In order to know the subscription identifier the GK\_GetSubscriptionID() function must be invoked passing gkData. On the other hand, to know the request result the GK\_GetMarketResponse() function must be invoked passing gkData.

## 6.6 Application message unsubscription result

```
void UnSubscribeResp context,
    (GK_Context_t
     GK_Data_t gkData,
     GK_Tag_t gkTag);
```

**Parameters:**

- context** Context identifier
- gkData** Data returned from the Notification event
- gkTag** User tag returned by the call-back

**Description** The callback function pointer is passed to the unsubscribe request function. The GK-API will call the callback whenever it must notify new results. If this callback function pointer is passed only to the unsubscribe request function, it will be possible to receive only notification of the GK\_UNSUBSCRIBE\_RESPONSE\_NOTIFICATION type. In order to know the subscription identifier the GK\_GetSubscriptionID() function must be invoked passing gkData. On the other hand, to know the request result the GK\_GetMarketResponse() function must be invoked passing gkData.

## 6.7 Data inquiry request result

```
void InquireResp context,
    (GK_Context_t
     GK_Data_t gkData,
     GK_Tag_t gkTag);
```

**Parameters:** **context** Context identifier

	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the call-back
Description	The callback function pointer is passed to the snapshot subscription (inquiry) request function. The GK-API will call the callback whenever it must notify a result. If this callback function pointer is passed only to the snapshot subscription request function, it will be possible to receive only notification of the GK_INQUIRE_RESPONSE_NOTIFICATION type. In order to know the submit result the GK_GetMarketResponse() function must be invoked passing gkData. On the other hand, to know the enquiry identifier belonging to that subscription the GK_GetInquireID() function must be invoked passing gkData.	

## 6.8 Data subscription notification

```
void NotifyData (GK_Context_t context,
                 GK_Data_t gkData,
                 GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the call-back

Description	The callback function pointer is passed to the subscribe notification function. The GK-API will call the callback whenever it must notify new data. If this callback function pointer is passed only to the subscription request function, it will be possible to receive only notification of the GK_NOTIFY_DATA_NOTIFICATION type. In order to unpack incoming data the GK_GetClassName(), GK_GetClassData(), GK_GetFieldClassData() functions must be invoked passing gkData. On the other hand, to know the subscription identifier belonging to that subscription, the GK_GetSubscriptionID() function must be invoked passing gkData.	
-------------	---	--

## 6.9 Data inquiry notification

```
void NotifyData (GK_Context_t context,
                 GK_Data_t gkData,
                 GK_Tag_t gkTag);
```

Parameters:	<b>context</b>	Context identifier
	<b>gkData</b>	Data returned from the Notification event
	<b>gkTag</b>	User tag returned by the call-back
Description	<p>The callback function pointer is passed to the snapshot subscription (inquiry) notification function. The GK-API will call the callback whenever it must notify new data. If this callback function pointer is passed only to the inquiry request function, it will be possible to receive only notification of the GK_INQUIRE_DATA_NOTIFICATION and GK_BINARY_INQUIRE_DATA_NOTIFICATION types. The received notification type only depends on the class used in the inquiry request.</p> <p>In order to unpack incoming data of GK_INQUIRE_DATA_NOTIFICATION type, the GK_GetClassName(), GK_GetClassData(), GK_GetFieldClassData() functions must be invoked passing gkData. On the other hand, to know the inquiry identifier belonging to that snapshot subscription, the GK_GetInquireID() function must be invoked passing gkData. Instead, in order to manage incoming data of GK_BINARY_INQUIRE_DATA_NOTIFICATION type, the GK_GetClassName() and GK_GetBinaryData() functions must be invoked passing gkData. Data retrieved using the GK_GetBinaryData() function are binary data. If multiple binary notifications are received on an inquiry request, user have to concatenate each binary data segment in the order they are received to obtain the whole inquiry response data. Depending on the class used in the inquiry request, the received binary data can be compressed by the server. To decompress binary data, the GK_UnzipBinaryData function must be invoked (see section <b>Error! Reference source not found.</b>).</p>	

# 7. RETRIEVING DATA FROM CALLBACK OBJECTS







```
(GK_Data_t gkData,
GK_Status_t* pMarketStatus);
```

Parameters: **gkData** Handle of available data  
**pMarketStatus** Connection status

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_INVALID_HANDLE	The referred handle is not valid
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the connection status notified by a callback. The function can be used only for the following notification types:

- GK\_MARKET\_STATUS\_NOTIFICATION
- GK\_TRANSACTION\_STATUS\_NOTIFICATION
- GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION

## 7.4 GK\_GetTransactionID

```
GK_Reply_t GK_GetTransactionID
(GK_Data_t gkData,
GK_Transaction_t* pTransaction);
```

Parameters: **gkData** Handle of available data  
**pTransaction** Transaction identifier

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_INVALID_HANDLE	The referred handle is not valid
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the transaction identifier notified by a callback. The function can be used only for the following notification types:

- GK\_SUBMIT\_RESPONSE\_NOTIFICATION
- GK\_TRANSACTION\_STATUS\_NOTIFICATION

## 7.5 GK\_GetMarketResponse

```
GK_Reply_t  GK_GetMarketResponse
            (GK_Data_t  gkData,
             GK_MarketReply_t* pReply,
             char**  specification);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>pReply</b>	Reply coming from the market
	<b>specification</b>	Subscription status

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the market reply notified by a callback. The **specification** parameter is allocated by the GK-API and must be released by the calling application by using the GK\_FreeString function. The function can be used only for the following notification types:

- GK\_SUBMIT\_RESPONSE\_NOTIFICATION
- GK\_CONNECTION\_RESPONSE\_NOTIFICATION
- GK\_DISCONNECTION\_RESPONSE\_NOTIFICATION
- GK\_SUBMIT\_RESPONSE\_NOTIFICATION
- GK\_SUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_UNSUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_INQUIRE\_RESPONSE\_NOTIFICATION

## 7.6 GK\_GetSubscriptionID

```
GK_Reply_t      GK_GetSubscriptionID
                (GK_Data_t gkData,
                 GK_Subscription_t* pSubscription);
```

Parameters: **gkData** Handle of available data  
**pSubscription** Subscription identifier

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_INVALID_HANDLE	The referred handle is not valid
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the subscription identifier notified by a callback. The function can be used only for the following notification types:

- GK\_SUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_UNSUBSCRIBE\_RESPONSE\_NOTIFICATION
- GK\_SUBSCRIPTION\_STATUS\_NOTIFICATION
- GK\_NOTIFY\_DATA\_NOTIFICATION

## 7.7 GK\_GetInquireID

```
GK_Reply_t      GK_GetInquireID gkData,
                (GK_Data_t
                 GK_Inquire_t* pInquire);
```

Parameters: **gkData** Handle of available data  
**pInquire** Inquiry identifier

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_INVALID_HANDLE	The referred handle is not valid

GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the inquiry identifier notified by a callback. The function can be used only for the following notification types:

- GK\_INQUIRE\_RESPONSE\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION
- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION

## 7.8 GK\_GetClassName

```
GK_Reply_t      GK_GetClassName
                  (GK_Data_t gkData,
                  char** className,
                  GK_ClassType_t* pClassType);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>className</b>	Class name
	<b>pClassType</b>	Class type

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the class name notified by a callback. The *className* parameter is allocated by the GK-API and must be released by the calling application using the *GK\_FreeString* function. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION
- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION

## 7.9 GK\_DecodeData

```
GK_Reply_t      GK_DecodeData gkData,
                (GK_Data_t
                 char** data);
```

Parameters: **gkData** Handle of available data  
**data** Application data

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_INVALID_HANDLE	The referred handle is not valid
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the application data (string) notified by a callback. The data parameter is allocated by the GK-API and must be released by the calling application using GK\_FreeString. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION

## 7.10 GK\_GetValueString

```
GK_Reply_t      GK_GetValueString (GK_Data_t gkData,
                                     const char* Key ,
                                     char** value);
```

Parameters: **gkData** Handle of available data  
**Key** Filed name of the application data  
**Value** Returned value of requested filed

Return values:

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed

GK_INVALID_HANDLE	The referred handle is not valid
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized
GK_TYPE_MISMATCH	The requested Key does not exist

Description: This function must be invoked in order to extract the application data (string) from the message notified by a callback. The Value parameter is allocated and returned by the GK-API and must be released by the calling application using the GK\_FreeString function. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION

## 7.11 GK\_GetValueLong

```
GK_Reply_t      GK_GetValueLong gkData,
                (GK_Data_t
                 const char* key,
                 long* value);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>Key</b>	Filed name of the application data
	<b>Value</b>	Returned value of requested field

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_TYPE_MISMATCH	The requested Key does not exist

Description: This function must be invoked in order to extract the application data (long) from the message notified by a callback. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION

## 7.12 GK\_GetValueDouble

```
GK_Reply_t          GK_GetValueDouble gkData,
                    (GK_Data_t
                     const char* key ,
                     double* value);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>Key</b>	Filed name of the application data
	<b>Value</b>	Returned value of requested field

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_TYPE_MISMATCH	The requested Key does not exist

Description: This function must be invoked in order to extract the application data (double) from the message notified by a callback. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION

## 7.13 GK\_GetValueInt

```
GK_Reply_t          GK_GetValueInt gkData,
                    (GK_Data_t
                     const char* key,
                     int* value);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>Key</b>	Filed name of the application data
	<b>value</b>	Returned value of requested field

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_TYPE_MISMATCH	The requested Key does not exist

Description: This function must be invoked in order to extract the application data (integer) from message notified by a callback. The function can be used only for the following notification types:

- GK\_NOTIFY\_DATA\_NOTIFICATION
- GK\_INQUIRE\_DATA\_NOTIFICATION

## 7.14 GK\_GetChain

```
GK_Reply_t GK_GetChain (GK_Data_t gkData,
                        GK_Chain_t* pChain);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>pChain</b>	Data chain

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_TYPE_MISMATCH	The requested Key does not exist

Description: This function must be invoked in order to extract the inquiry status notified by a callback. The function can be used only for the following notification types:

- GK\_INQUIRE\_DATA\_NOTIFICATION
- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION



## 7.15 GK\_GetBinaryData

```
GK_Reply_t          GK_GetBinaryData gkData,
                    (GK_Data_t
                     GK_Byte_t** pData,
                     GK_Length_t* pDataLength);
```

Parameters:	<b>gkData</b>	Handle of available data
	<b>pData</b>	Application binary data buffer
	<b>pDataLength</b>	Returned length of binary data buffer
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_INVALID_HANDLE	The referred handle is not valid
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked in order to extract the application binary data notified by a callback. The pData parameter is allocated by the GK-API and must be released by the calling application using GK\_FreeString. The function can be used only for the following notification types:

- GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION

# 8. BUILDING APPLICATION DATA MESSAGES



## 8.1 GK\_CreateApplicationData

```
GK_Reply_t
    GK_CreateApplicationData (const char* className,
                             GK_ClassType_t classType,
                             GK_ApplicationData_t** pApplicationData);
```

Parameters:	<b>className</b>	Data class name
	<b>classType</b>	Data class type
	<b>pApplicationData</b>	Pointer to the message structure

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to create an application message pApplicationData. The pApplicationData parameter is allocated and returned by the GK-API and must be released by the calling application using the GK\_FreeApplicationData() function.

## 8.2 GK\_EncodeData

```
GK_Reply_t
    GK_EncodeData (GK_ApplicationData_t* pApplicationData,
                  const char* data);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>data</b>	Application fields (format: "field=value; field=value;..")

Return values:	GK_SUCCESS	Function successfully completed
----------------	------------	---------------------------------

GK_FAILED	Function not completed
GK_API_ERROR	Internal error
GK_API_NOT_INITIALIZED	GK-API not initialized

**Description:** This function must be invoked to insert the application message using the following format: "field=value". As opposed to the GK\_Set... functions (which set a single field value at the time), this function will set the complete message string abiding by the message layout.

## 8.3 GK\_SetValueString

```
GK_Reply_t      GK_SetValueString
                (GK_ApplicationData_t* pApplicationData,
                 const char* key,
                 const char* value);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>Key</b>	Application filed name
	<b>Value</b>	Field value to insert
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

**Description:** This function must be invoked to assign the value "value" to the field "key". If a value already exists, the new value will replace the previous one.

## 8.4 GK\_SetValueLong

```
GK_Reply_t      GK_SetValueLong
                (GK_ApplicationData_t* pApplicationData,
                 const char* key,
                 long value);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>Key</b>	Application filed name
	<b>Value</b>	Field value to insert
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to assign the value "value" to the field "key". If a value already exists, the new value will replace the previous one.

## 8.5 GK\_SetValueDouble

```
GK_Reply_t      GK_SetValueDouble
                (GK_ApplicationData_t* pApplicationData,
                 const char* key,
                 double value);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>key</b>	Application filed name
	<b>value</b>	Field value to insert
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to assign the value "value" to the field "key". If a value already exists, the new value will replace the previous one.

## 8.6 GK\_SetValueInt

```
GK_Reply_t          GK_SetValueInt
                    (GK_ApplicationData_t* pApplicationData,
                     const char* key,
                     int value);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>key</b>	Application field name
	<b>value</b>	Field value to insert

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to assign the value "value" to the field "key". If a value already exists, the new value will replace the previous one.

## 8.7 GK\_DestroyApplicationData

```
GK_Reply_t          GK_DestroyApplicationData
                    (GK_ApplicationData_t* pApplicationData);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
------------	-------------------------	---

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to release the message structure.

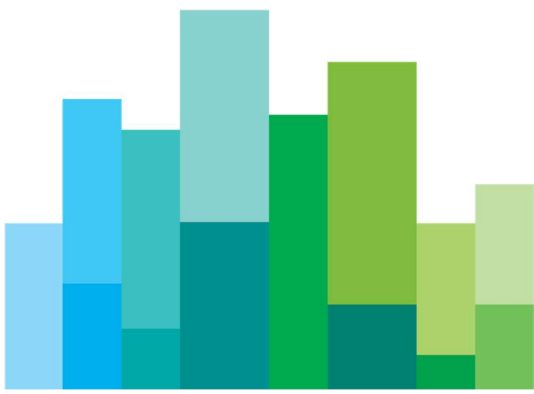
## 8.8 GK\_SetTransactionID

```
GK_Reply_t    GK_SetTransactionID
              (GK_ApplicationData_t* pApplicationData,
               GK_Transaction_t transaction);
```

Parameters	<b>pApplicationData</b>	Pointer to the message structure to be filled
	<b>transaction</b>	Transaction identifier
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

Description: This function must be invoked to insert a transaction identifier within an application message. This type of application message is needed to subscribe information related to the related transaction (e.g. status, proposal information belonging to the transaction).

# 9. UNZIPPING CALLBACK FUNCTIONS





Binary compressed data received on notification of GK\_BINARY\_INQUIRE\_DATA\_NOTIFICATION type can be decompressed using the GK\_UnzipBinaryData() function, which provides an in-memory decompression mechanism including integrity checks of the uncompressed data.

To call the GK\_UnzipBinaryData() function, user application must provide an input buffer containing the binary compressed data and an output buffer that will receive the uncompressed data. If the input buffer contains all the binary compressed data and the output buffer is large enough, decompression can be done in a single step. Otherwise, the unzip activity can be done by repeated calls of the GK\_UnzipBinaryData() function. In the latter case, the user application must provide more input and/or consume the output (providing more output space) before each call. The GK\_UnzipBinaryData() function provides each time as much output as possible, until there is no more input data or no more space in the output buffer.

In order to use the GK\_UnzipBinaryData() function, user application must also provide a parameter of GK\_UnzipHelper\_t type, which is an internal structure managed by the GK-API during the unzip process. Before starting to unzip binary data, user application has to create an instance of GK\_UnzipHelper\_t type by means of the GK\_CreateUnzipHelper() function. Then, in order to provide the input data buffer, user have to initialize the GK\_UnzipHelper\_t structure using the GK\_InitializeUnzipHelper() function; this function has to be called every time more input is needed to complete the unzip process. After that, user application have to repeatedly call the GK\_UnzipBinaryData() function until no more output is available. When the unzip process is terminated (as well as or an error has occurred), the helper structure has to be cleared using the GK\_ClearUnzipHelper() function. Finally, the helper structure has to be released using the GK\_DestroyUnzipHelper() function since it cannot be reused to start another unzip session.

## 9.1 GK\_CreateUnzipHelper

```
GK_Reply_t    GK_CreateUnzipHelper
              (GK_UnzipHelper_t* pUnzipHelper);
```

Parameters:	<b>pUnzipHelper</b>	Pointer to the returned internal helper structure
Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized

**Description:** This function must be invoked to create an internal helper structure pUnzipHelper. The pUnzipHelper parameter is allocated and returned by the GK-API and must be released by the calling application using the GK\_DestroyUnzipHelper() function.

## 9.2 GK\_DestroyUnzipHelper

```
GK_Reply_t  GK_DestroyUnzipHelper
            (GK_UnzipHelper_t  unzipHelper);
```

**Parameters:** **unzipHelper** Internal helper structure created using GK\_CreateUnzipHelper()

**Return values:**

GK_SUCCESS	Function successfully completed
GK_FAILED	Function not completed
GK_API_NOT_INITIALIZED	GK-API not initialized

**Description:** This function must be invoked to deallocate an internal helper structure allocated using the GK\_CreateUnzipHelper() function.

## 9.3 GK\_InitializeUnzipHelper

```
GK_Reply_t  GK_InitializeUnzipHelpe
            r (GK_UnzipHelper_t  unzipHelper,
              const GK_Byte_t*  Data,
              GK_Length_t  DataLength);
```

**Parameters:** **unzipHelper** Internal helper structure created using GK\_CreateUnzipHelper()

**Data** Pointer to a user buffer containing binary data to be unzipped

**DataLength** Length of the data in the user buffer

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_INVALID_PARAMETER	Value of parameter DataLength is not valid

**Description:** This function must be invoked to initialize an internal helper structure allocated using the GK\_CreateUnzipHelper() function. If binary data has to be unzipped in a single step, the Data parameter must point to a buffer containing all the binary data to be unzipped; otherwise, the Data parameter can point to a buffer containing only a part of the binary data to be unzipped.

## 9.4 GK\_ClearUnzipHelper

```
GK_Reply_t      GK_ClearUnzipHelper
                 (GK_UnzipHelper_t  unzipHelper);
```

**Parameters:** **unzipHelper** Internal helper structure created using GK\_CreateUnzipHelper()

Return values:	GK_SUCCESS	Function successfully completed
	GK_FAILED	Function not completed
	GK_API_NOT_INITIALIZED	GK-API not initialized

**Description:** This function must be invoked to clear an internal helper structure allocated using the GK\_CreateUnzipHelper() function. Internal helper structures used to unzip binary data must be cleared after each unzip session is terminated, successfully or unsuccessfully.

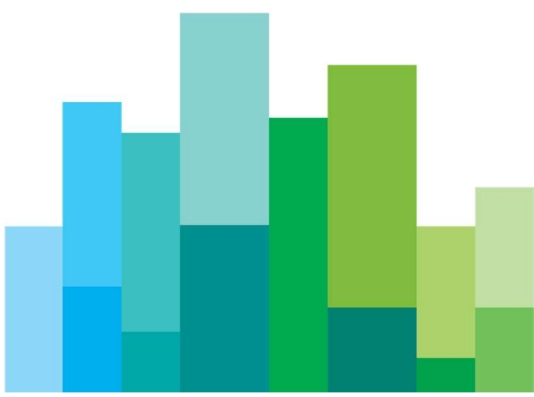
## 9.5 GK\_UnzipBinaryData

```
GK_Reply_t      GK_UnzipBinaryData
                 (GK_UnzipHelper_t  unzipHelper,
                 char*  buffer,
                 GK_Length_t  bufferLength,
                 GK_Length_t*  pDataLength);
```

Parameters:	<b>unzipHelper</b>	Internal helper structure created using GK_CreateUnzipHelper()
	<b>buffer</b>	Pointer to a user output buffer
	<b>bufferLength</b>	Length of user output buffer
	<b>pDataLength</b>	Returned length of unzipped data
Return values:	GK_SUCCESS	Function successfully completed. All the binary data have been unzipped, i.e. the end of the compressed data has been reached and all uncompressed output has been produced
	GK_MORE_OUTPUT_AVAILABLE	Function successfully completed. User buffer is full and the function must be called again because there might be more output pending
	GK_MORE_INPUT_NEEDED	Function successfully completed. All provided binary data have been unzipped and the function must be called again providing more input binary data to complete the unzip process.
	GK_FAILED	Function not completed
	GK_API_ERROR	Internal error
	GK_API_NOT_INITIALIZED	GK-API not initialized
	GK_INVALID_PARAMETER	Value of parameter bufferLength is not valid
	GK_DATA_ERROR	Supplied data are invalid or corrupted.

**Description:** This function must be invoked to unzip compressed binary data. This function decompresses as much data as possible, and stops when the input buffer becomes empty or the output buffer becomes full.

# 10. RECOVERY



This section describes the recovery process implemented by the BCS system and the actions to be taken when the system notifies the events concerning the services. In order to receive the connection status, the client application has to invoke the `Subscribe.System.ServiceMarketStatus` subscription class and it has to evaluate the data provided by the `Notify.System.ServiceMarketStatus` callback function.

Instead, events concerning the status of the connection between client and server are provided through the `MarketStatus` callback (see section 6.3).

## 10.1 Services

The BCS system is based on a set of services, each one managing a specific set of functions. It is possible to be notified about the status of a single service of the system. Possible values for service id are the following:

Service	ServiceID	Description
<b>Risk Manager</b>	2000	The service that manages all Risk Management requests
<b>Clearing Data Manager</b>	2100	The service that stores all market realtime data
<b>Report Manager</b>	2200	The service that manages all report requests
<b>Transactional Gateway</b>	2300	The gateway that connects to CC&G Clearing system and manages all transactional requests
<b>Realtime Gateway</b>	2400	The gateway that connects to CC&G Clearing system and receives real time data
<b>Sola Gateway</b>	2500	The service that manages the connection to SOLA Derivatives

Is it possible, using API, still call a `Subscribe.System.ServiceMarketStatus` that include a group of components (`ServiceID=100`). This layout is obsolete and will be dismissed soon.

Please note that in the following tables the length column stands for the maximum length of the field.

## 10.2 Subscribe.System.ServiceMarketStatus

Request the service market connection status. The status is notified by Notify.System.ServiceMarketStatus.

Field	Type	Length	Description
<b>ServiceID</b>	integer	10	The ID of the service
<b>RequestedMember</b>	string	100	Not mandatory.

## 10.3 Notify.System.ServiceMarketStatus

Notify the service connection status.

Field	Type	Length	Description
<b>Member</b>	String	100	Member name.
<b>ServiceID</b>	integer	10	The ID of the service
<b>Market</b>	string	100	Market identifier
<b>Status</b>	string	50	<p>The connection status of the service &lt;ServiceID&gt; operating on the market &lt;Market&gt; for the member &lt;Member&gt;.</p> <p>The possible values are:</p> <p>CONNECTION_UP: the service is available.</p> <p>CONNECTION_CRASH: the service is not available</p>

The following actions need to be taken when Notify.System.ServiceMarketStatus events are received:

<b>CONNECTION_UP</b>	The connection is successfully established: the user can start its activity.
----------------------	--

<b>CONNECTION_CRASH</b>	The service is no longer available: the user should wait for a CONNECTION_UP event in order to restart its activity. All the Subscribe calls executed before the CONNECTION_CRASH event should be unsubscribed and then called again by the user.
-------------------------	--

Please note that the status "CONNECTION\_DOWN" and "CONNECTION\_WARNING" has been dismissed so is not possible receive this notifies.

The below table list the link between the ServiceID and the related Subscriptions:

Service	ServiceID	Subscriptions
<b>Risk Manager</b>	2000	SubscribeStandardPortfolioParameters SubscribeCustomPortfolioParameters SubscribeTradeLimitParameters SubscribeTradeLimitAlarms SubscribePositionLimitParameters SubscribePositionLimitAlarms SubscribeMarginLimitParameters SubscribeMarginLimitAlarms
<b>Clearing Data Manager</b>	2100	SubscribeSeries SubscribePositions SubscribeRectifications SubscribePositionTransfers SubscribeContracts SubscribeContractTransfers SubscribeOpenCloseContractChanges SubscribeClientCodeContractChanges SubscribeSplitContracts SubscribeCollateralGuarantees SubscribeDepositedGuarantees SubscribeEarlyExercises SubscribeExByEx SubscribeExerciseAtExpiry



Service	ServiceID	Subscriptions
		SubscribeAssignments SubscribeAssignmentsSent SubscribeClearingMessages SubscribeIntradayMarginCalls SubscribeIntradayMarginCallsSent SubscribeSubAccountTransfers SubscribeSubAccountParameters SubscribeSubAccountClientCodeLinks SubscribeSubAccountClientCodeLinkChange SubscribeGiveOutParameters SubscribeTakeUpParameters
<b>Report Manager</b>	2200	SubscribeReport
<b>Transactional Gateway</b>	2300	-
<b>Realtime Gateway</b>	2400	-
<b>Sola Gateway</b>	2500	SubscribeFirmStatus

## 10.4 Recovery Simulation in CDS (Test) environment

It's possible to test the System.ServiceMarketStatus messages reception in the CDS (Test) environment every Tuesday. Two sessions are available, one starting at 10:00 (GMT) and one starting at 15:00 (GMT).

After the login to the system, the user should send a `Subscribe.System.ServiceMarketStatus` message for each service managed by his application, in order to receive the related status updates (as per highlighted in the table at section 10.1 ).

The crash simulation of the BCS components will be executed as follows:

GMT Time	Description
<b>10:00 / 15:00</b>	The component Report Manager crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2200 are received.
<b>10:05 / 15:05</b>	The component Report Manager is restored; one or more messages with status CONNECTION_UP and ServiceId=2200 are received.
<b>10:15 / 15:15</b>	The component Realtime Gateway crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2400 are received.
<b>10:20 / 15:20</b>	The component Realtime Gateway is restored; one or more messages with status CONNECTION_UP and ServiceId=2400 are received.
<b>10:30 / 15:30</b>	The component Transactional Gateway crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2300 are received.
<b>10:35 / 15:35</b>	The component Transactional Gateway is restored; one or more messages with status CONNECTION_UP and ServiceId=2300 are received.
<b>10:45 / 15:45</b>	The component Clearing Data Manager crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2100 are received.
<b>10:50 / 15:50</b>	The component Clearing Data Manager is restored; one or more messages with status CONNECTION_UP and ServiceId=2100 are received.
<b>11:00 / 16:00</b>	The component Risk Management crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2000 are received.
<b>11:05 / 16:05</b>	The component Risk Management is restored; one or more messages with

GMT Time	Description
	status CONNECTION_UP and ServiceId=2000 are received.
<b>11:15 / 16:15</b>	The component Sola Gateway crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2500 are received.
<b>11:20 / 16:20</b>	The component Risk Management is restored; one or more messages with status CONNECTION_UP and ServiceId=2500 are received.

After every recovery simulation session, the system becomes available as per the usual schedule.

An additional Connection Crash on the Transactional Gateway component may be received during the recovery sessions. This is caused by CCG settlement procedures.

Please note that, in case a user sends more than a `Subscribe.System.MarketStatus` for the same ServiceId, it will receive a number of `CONNECTION_CRASH` and `CONNECTION_UP` messages equal to the number of `Subscribe.System.ServiceMarketStatus` active (accepted by the system).

For instance, if a user has `3xSubscribe.System.ServiceMarketStatus` active with `ServiceId=2300`, it **will receive 3xNotify.System.ServiceMarketStatus with status CONNECTION\_CRASH and ServiceId=2300** followed by **3xNotify.System.ServiceMarketStatus with status CONNECTION\_UP and ServiceId=2300**.

# CONTACT

## Client Support

**T (toll free): 0080026772000**

**T (from mobile): +39 02 45411399**

**E: [Client-Support@borsaitaliana.it](mailto:Client-Support@borsaitaliana.it)**

## Customer Relationship Management

**T: +39 02 72426 512**

**E: [clients-services@borsaitaliana.it](mailto:clients-services@borsaitaliana.it)**

## Disclaimer

This publication is for information purposes only and is not a recommendation to engage in investment activities. This publication is provided "as is" without representation or warranty of any kind. Whilst all reasonable care has been taken to ensure the accuracy of the content, Euronext does not guarantee its accuracy or completeness. Euronext will not be held liable for any loss or damages of any nature ensuing from using, trusting or acting on information provided. No information set out or referred to in this publication shall form the basis of any contract. The creation of rights and obligations in respect of financial products that are traded on the exchanges operated by Euronext's subsidiaries shall depend solely on the applicable rules of the market operator. All proprietary rights and interest in or connected with this publication shall vest in Euronext. No part of it may be redistributed or reproduced in any form without the prior written permission of Euronext. Euronext disclaims any duty to update this information. Euronext refers to Euronext N.V. and its affiliates. Information regarding trademarks and intellectual property rights of Euronext is located at [www.euronext.com/terms-use](http://www.euronext.com/terms-use).

