

BCS

API Programmer's Manual

Ver 3.6

December 2020



London
Stock Exchange Group

Contents

Summary

1.0	Introduction	6
2.0	Connection to the BCS Clearing system	7
3.0	Configuration file	8
4.0	Type definitions	10
4.1	GK_Reply_t	11
4.2	GK_MarketReply_t	12
4.3	GK_ClassType_t	13
4.4	GK_Status_t	13
4.5	GK_Chain_t	14
4.6	GK_Notification_t	14
4.7	GK_ApplicationData_t	15
4.8	GK_Callback_t	15
4.9	GK_Tag_t	15
4.10	GK_Data_t	15
4.11	GK_Transaction_t	16
4.12	GK_Subscription_t	16
4.13	GK_Inquire_t	16
4.14	GK_Context_t	16
4.15	GK_Connection_t	16
4.16	GK_Length_t	16
4.17	GK_Byte_t	17
4.18	GK_UnzipHelper_t	17

5.0	Main callback functions	18
5.1	GK_Initialize	18
5.2	GK_Terminate	18
5.3	GK_CreateContext	18
5.4	GK_Dispatch	19
5.5	GK_ReleaseContext	19
5.6	GK_Connect	20
5.7	GK_Disconnect	21
5.8	GK_CreateTransaction	22
5.9	GK_DestroyTransaction	22
5.10	GK_Submit	23
5.11	GK_Subscribe	24
5.12	GK_UnSubscribe	25
5.13	GK_Inquire	26
5.14	GK_GetVersion	27

Programmer's Manual

December 2020

5.15 GK_ConnectEx 28

6.0 Introduction to Callbacks 30

6.1 Connection request result 30
6.2 Disconnect request result 31
6.3 Connection monitoring 31
6.4 Application message submission result 33
6.5 Application message subscription result 33
6.6 Application message unsubscription result 34
6.7 Data inquiry request result 34
6.8 Data subscription notification 34
6.9 Data inquiry notification 35

7.0 Retrieving data from callback objects 37

7.1 Connection request result 37
7.2 GK_GetNotificationType 37
7.3 GK_GetConnectionStatus 37
7.4 GK_GetTransactionID 38
7.5 GK_GetMarketResponse 38
7.6 GK_GetSubscriptionID 39
7.7 GK_GetInquireID 40
7.8 GK_GetClassName 40
7.9 GK_DecodeData 41
7.10 GK_GetValueString 41
7.11 GK_GetValueLong 42
7.12 GK_GetValueDouble 42
7.13 GK_GetValueInt 43
7.14 GK_GetChain 44
7.15 GK_GetBinaryData 44

8.0 Building application data messages 45

8.1 GK_CreateApplicationData 45
8.2 GK_EncodeData 45
8.3 GK_SetValueString 46
8.4 GK_SetValueLong 46
8.5 GK_SetValueDouble 46
8.6 GK_SetValueInt 47
8.7 GK_DestroyApplicationData 47
8.8 GK_SetTransactionID 48

Programmer's Manual

December 2020

9.0	Unzipping callback functions	49
9.1	GK_CreateUnzipHelper	49
9.2	GK_DestroyUnzipHelper	50
9.3	GK_InitializeUnzipHelper	50
9.4	GK_ClearUnzipHelper	51
9.5	GK_UnzipBinaryData	51

10.0	Recovery	53
10.1	Services	53
10.2	Subscribe.System.ServiceMarketStatus	54
10.3	Notify.System.ServiceMarketStatus	54
10.4	Recovery Simulation in CDS (Test) environment	57

Programmer's Manual

December 2020

1.0 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;

Programmer's Manual

December 2020

2.0 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;

Programmer's Manual

December 2020

3.0 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
```

Programmer's Manual

December 2020

```

// 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.
```

Programmer's Manual

December 2020

4.0 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

Programmer's Manual

December 2020

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.

Programmer's Manual

December 2020

- 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

Programmer's Manual

December 2020

- `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

Programmer's Manual

December 2020

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

Programmer's Manual

December 2020

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
GK_ClassType_t	ClassType	Class type or application data layout type (meta-market or market class)
const char*	ClassName	Class name
const char*	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.

Programmer's Manual

December 2020

```
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;
```

Programmer's Manual

December 2020

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;
```

Programmer's Manual

December 2020

5.0 Main callback functions

The following sections describe all the BCS API callback functions.

5.1 GK_Initialize

<i>GK_Reply_t</i>	GK_Initialize (<i>const char*</i> ConfigFile);	
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_INITIALIZED GK_INVALID_PARAMETER	GK-API already initialized ConfigFile 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

<i>GK_Reply_t</i>	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

<i>GK_Reply_t</i>	GK_CreateContext (<i>const char*</i> serverName, <i>GK_Context_t*</i> pContext, <i>GK_Connection_t*</i> pConnection);	
Parameters	serverName	Name of the application server through which connection must be set up (one from the list in SERVER_LIST in the configuration file)
	pContext	Working context identifier returned by the GK-API

Programmer's Manual

December 2020

	pConnection	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 GK_API_ERROR GK_INVALID_SERVER GK_SERVER_UNREACHABLE GK_API_NOT_INITIALIZED GK_INVALID_PARAMETER	Context available, socket connection established Internal error Application server name invalid (check if it is present in the configuration file) Server unreachable GK-API not initialized At least one of <i>serverName</i> , <i>pContext</i> or <i>pConnection</i> is <i>null</i>
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

<i>GK_Reply_t</i>	<i>GK_Dispatch</i> (<i>GK_Context_t context</i>);	
Parameters	context	Working context identifier
Return values	GK_SUCCESS GK_API_ERROR GK_INVALID_CONTEXT GK_API_NOT_INITIALIZED	Dispatch successfully completed Internal error Context not valid API not initialized
Description	This function is used to handle callbacks. <i>GK_Dispatch</i> must be called as soon as an event raises from the working context. For this purpose, before calling <i>GK_Dispatch</i> , call "select" API on the socket returned by <i>GK_CreateContext</i> using a positive timeout values (i.e. not zero; usual timeout value is 5 seconds). Moreover, it is recommended to call <i>GK_Dispatch</i> using a dedicated thread, one for each working context.	

5.5 GK_ReleaseContext

<i>GK_Reply_t</i>	<i>GK_ReleaseContext</i> (<i>GK_Context_t context</i>);	
Parameters	context	Working context identifier
Return	GK_SUCCESS	Context successfully released.

Programmer's Manual

December 2020

values:	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

<i>GK_Reply_t</i>	GK_Connect (<i>GK_Context_t</i> context, const char* userName, const char* password, const char* market, <i>GK_Callback_t</i> pCallbackResponse, <i>GK_Callback_t</i> pCallbackMarketStatus, <i>GK_Tag_t</i> gkTag)	
Parameters	context	Active context identifier through which a connection must be performed.
	userName	Name of the user requiring the connection
	password	Password of the user requiring the connection.
	market	Market or Clearing House name to which a connection is requested (e.g. MTA, CCG, ...)
	pCallbackResponse	Callback to handle a notification event for the related request.
	pCallbackMarketStatus	Callback to handle a notification event for the connection status
	gkTag	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
	<i>from pCallbackResponse</i>	

Programmer's Manual

December 2020

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_WARNIN G 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** (*GK_Context_t* context,
GK_Callback_t pCallbackResponse,
GK_Tag_t gkTag);

Parameters: **context** Context identifier
 pCallbackResponse Call-back for request notification
 gkTag 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

Programmer's Manual

December 2020

	<i>from pCallbackResponse</i> GK_REQUEST_ACCEPTED GK_REQUEST_REJECTED GK_NOT_CONNECTED	Connection accepted Connection refused Connection not existing
Description	This function must be invoked to release a connection to the BCS Clearing system.	

5.8 GK_CreateTransaction

<i>GK_Reply_t</i>	<i>GK_CreateTransaction</i> (<i>GK_Context_t</i> context, <i>GK_Transaction_t*</i> pTransactionID);	
Parameters:	context pTransactionID	Context identifier Transaction identifier returned by the function
Return values	GK_SUCCESS GK_INVALID_CONTEXT GK_API_ERROR GK_API_NOT_INITIALIZED GK_INVALID_PARAMETER	Transaction creation successfully completed Context is not valid Internal error GK-API not initialized <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

<i>GK_Reply_t</i>	<i>GK_DestroyTransaction</i> (<i>GK_Context_t</i> context, <i>GK_Transaction_t</i> transactionID);	
Parameters:	context transactionID	Context identifier Transaction identifier
Return values	GK_SUCCESS GK_INVALID_TRANSACTIONID GK_INVALID_CONTEXT	Destroy transaction successfully completed Transaction identifier is not valid Context not valid

Programmer's Manual

December 2020

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:	context	Context identifier
	transactionID	Transaction identifier
	applicationData	Application data layout to be executed. It can be built using proper functions (see below)
	pCallbackResponse	Callback to handle a notification event for that request.
	gkTag	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
	<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_INVALID_FIELD	The specified field name is invalid

Programmer's Manual

December 2020

GK_REQUEST_ONGOING	A previous submit operation on the same transaction identifier is still on going
GK_PROPOSAL_ALREADY_EXISTS	A proposal belonging to the specified transaction identifier already exists
GK_PROPOSAL_NOT_EXISTS	A proposal belonging to the specified transaction identifier does not exist
GK_INVALID_PROPOSAL_KEY	Invalid proposal referenced
GK_MISSING_FIELD_VALUE	Mandatory Field is empty/missing
GK_INVALID_CLASS	Class not valid
GK_NOT_CONNECTED	Connection is not in place
GK_INVALID_TRANSACTIONID	Transaction identifier is not valid

Description: This function must be invoked to send a Submit data structure to the BCS Clearing system. If this message will be accepted, a callback will be fired. 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).

5.11 GK_Subscribe

```
GK_Reply_t  GK_Subscribe (GK_Context_t context,
                          GK_ApplicationData_t* applicationData,
                          GK_Callback_t pCallbackResponse,
                          GK_Callback_t pCallbackData,
                          GK_Tag_t gkTag,
                          GK_Subscription_t* pSubscriptionID);
```

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

Programmer's Manual

December 2020

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:	context	Context identifier
	pSubscriptionID	Unique identifier for the requested subscription to be ended
	pCallbackResponse	Call-back to handle a notification event for that request.
	gkTag	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

Programmer's Manual

December 2020

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

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

Programmer's Manual

December 2020

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

<i>GK_Reply_t</i>	GK_GetVersion (<i>char** company,</i> <i>char** version,</i> <i>char** creationDate,</i> <i>char** comment</i>);	
Parameters	company version creationDate comment	Company that distributes the GK-API Version Identifier Creation date Any comment
Return values:	GK_SUCCESS GK_API_ERROR	Request successfully executed 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.	

Programmer's Manual

December 2020

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

Programmer's Manual

December 2020

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.

Programmer's Manual

December 2020

6.0 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);
```

Programmer's Manual

December 2020

Parameters:	context	Context identifier
	gkData	Data returned from the Notification event
	gkTag	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

	<i>void</i>	DisconnectionResp (<i>GK_Context_t context,</i> <i>GK_Data_t gkData,</i> <i>GK_Tag_t gkTag</i>);
Parameters:	context	Context identifier
	gkData	Data returned from the Notification event
	gkTag	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

	<i>void</i>	MarketStatus (<i>GK_Context_t context,</i> <i>GK_Data_t gkData,</i> <i>GK_Tag_t gkTag</i>);
Parameters:	context	Context identifier
	gkData	Data returned from the Notification event
	gkTag	User tag returned by the callback

Programmer's Manual

December 2020

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.

Programmer's Manual

December 2020

6.4 Application message submission result

```
void          SubmitResp (GK_Context_t context,  
                          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 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:	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 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.

Programmer's Manual

December 2020

6.6 Application message unsubscription result

void **UnSubscribeResp** (*GK_Context_t context,*
 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** (*GK_Context_t context,*
 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 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,

Programmer's Manual

December 2020

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

Programmer's Manual

December 2020

Description 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.

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 9.0).

Programmer's Manual

December 2020

7.0 Retrieving data from callback objects

7.1 Connection request result

<i>GK_Reply_t</i>	<i>GK_FreeString</i> (<i>char* data</i>);
Parameters:	data Data to be freed
Return values:	GK_SUCCESS Function successfully completed
Description:	This function must be invoked to release all string-type and binary-type data allocated by the GK-API.

7.2 GK_GetNotificationType

<i>GK_Reply_t</i>	<i>GK_GetNotificationType</i> (<i>GK_Data_t gkData</i> , <i>GK_Notification_t* pNotificationType</i>);
Parameters:	gkData Handle of available data pNotificationType Notification 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 notification type related to a callback. The function can be used for any notification type.

7.3 GK_GetConnectionStatus

<i>GK_Reply_t</i>	<i>GK_GetConnectionStatus</i> (<i>GK_Data_t gkData</i> , <i>GK_Status_t* pMarketStatus</i>);
Parameters:	gkData Handle of available data pMarketStatus Connection status
Return values:	GK_SUCCESS Function successfully completed GK_FAILED Function not completed

Programmer's Manual

December 2020

values:

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

Programmer's Manual

December 2020

7.7 GK_GetInquireID

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

Parameters:	gkData pInquire	Handle of available data Inquiry identifier
Return values:	GK_SUCCESS GK_FAILED GK_INVALID_HANDLE GK_API_ERROR GK_API_NOT_INITIALIZED	Function successfully completed Function not completed The referred handle is not valid Internal error 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:	gkData className pClassType	Handle of available data Class name Class type
Return values:	GK_SUCCESS GK_FAILED GK_INVALID_HANDLE GK_API_ERROR GK_API_NOT_INITIALIZED	Function successfully completed Function not completed The referred handle is not valid Internal error GK-API not initialized

Programmer's Manual

December 2020

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

Parameters:	<code>gkData</code> <code>data</code>	Handle of available data Application data
Return values:	<code>GK_SUCCESS</code>	Function successfully completed
	<code>GK_FAILED</code>	Function not completed
	<code>GK_INVALID_HANDLE</code>	The referred handle is not valid
	<code>GK_API_ERROR</code> <code>GK_API_NOT_INITIALIZED</code>	Internal error 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:	<code>gkData</code> <code>Key</code> <code>Value</code>	Handle of available data Filed name of the application data Returned value of requested filed
Return values:	<code>GK_SUCCESS</code>	Function successfully completed
	<code>GK_FAILED</code>	Function not completed

Programmer's Manual

December 2020

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

Parameters:	gkData Key Value	Handle of available data Filed name of the application data Returned value of requested field
Return values:	GK_SUCCESS GK_FAILED GK_INVALID_HANDLE GK_API_ERROR GK_API_NOT_INITIALIZED GK_TYPE_MISMATCH	Function successfully completed Function not completed The referred handle is not valid Internal error GK-API not initialized 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** (*GK_Data_t* *gkData*,
 *const char** *key*,
 *double** *value*);

Parameters:	gkData	Handle of available data
-------------	---------------	--------------------------

Programmer's Manual

December 2020

	Key Value	Filed name of the application data 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:	
	<ul style="list-style-type: none">• GK_NOTIFY_DATA_NOTIFICATION• GK_INQUIRE_DATA_NOTIFICATION	

7.13 GK_GetValueInt

<i>GK_Reply_t</i>	GK_GetValueInt (<i>GK_Data_t</i> <i>gkData</i> , <i>const char*</i> <i>key</i> , <i>int*</i> <i>value</i>);	
Parameters:	gkData Key value	Handle of available data Filed name of the application data 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:	
	<ul style="list-style-type: none">• GK_NOTIFY_DATA_NOTIFICATION• GK_INQUIRE_DATA_NOTIFICATION	

Programmer's Manual

December 2020

7.14 GK_GetChain

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

Parameters:	gkData pChain	Handle of available data Data chain
Return values:	GK_SUCCESS GK_FAILED GK_INVALID_HANDLE GK_API_ERROR GK_API_NOT_INITIALIZED GK_TYPE_MISMATCH	Function successfully completed Function not completed The referred handle is not valid Internal error GK-API not initialized 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** (*GK_Data_t* gkData,
 *GK_Byte_t*** pData,
 *GK_Length_t** pDataLength);

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

Programmer's Manual

December 2020

8.0 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: **className** Data class name
 classType Data class type
 pApplicationData 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 **pApplicationData** Pointer to the message structure to be filled
 data 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.

Programmer's Manual

December 2020

8.3 GK_SetValueString

GK_Reply_t

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

Parameters	pApplicationData	Pointer to the message structure to be filled
	Key	Application filed name
	Value	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	pApplicationData	Pointer to the message structure to be filled
	Key	Application filed name
	Value	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*,

Programmer's Manual

December 2020

double value);

Parameters	pApplicationData	Pointer to the message structure to be filled
	key	Application field name
	value	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	pApplicationData	Pointer to the message structure to be filled
	key	Application field name
	value	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	pApplicationData	Pointer to the message structure to be filled
Return	GK_SUCCESS	Function successfully completed

Programmer's Manual

December 2020

values:

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 **pApplicationData** Pointer to the message structure to be filled
 transaction 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).

Programmer's Manual

December 2020

9.0 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

<i>GK_Reply_t</i>	<i>GK_CreateUnzipHelper</i> (<i>GK_UnzipHelper_t*</i> <i>pUnzipHelper</i>);	
Parameters:	pUnzipHelper	Pointer to the returned internal helper structure
Return values:	GK_SUCCESS GK_FAILED GK_API_ERROR GK_API_NOT_INITIALIZED	Function successfully completed Function not completed Internal error GK-API not initialized

Programmer's Manual

December 2020

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_InitializeUnzipHelper  
                  (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.

Programmer's Manual

December 2020

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:	unzipHelper	Internal helper structure created using GK_CreateUnzipHelper()
	buffer	Pointer to a user output buffer
	bufferLength	Length of user output buffer
	pDataLength	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.

Programmer's Manual

December 2020

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.

Programmer's Manual

December 2020

10.0 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
Risk Manager	2000	The service that manages all Risk Management requests
Clearing Data Manager	2100	The service that stores all market realtime data
Report Manager	2200	The service that manages all report requests
Transactional Gateway	2300	The gateway that connects to CC&G Clearing system and manages all transactional requests
Realtime Gateway	2400	The gateway that connects to CC&G Clearing system and receives real time data
Sola Gateway	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.

Programmer's Manual

December 2020

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
<code>ServiceID</code>	integer	10	The ID of the service
<code>RequestedMember</code>	string	100	Not mandatory.

10.3 `Notify.System.ServiceMarketStatus`

Notify the service connection status.

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

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

Programmer's Manual

December 2020

CONNECTION_UP	The connection is successfully established: the user can start its activity.
CONNECTION_CRASH	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
Risk Manager	2000	SubscribeStandardPortfolioParameters SubscribeCustomPortfolioParameters SubscribeTradeLimitParameters SubscribeTradeLimitAlarms SubscribePositionLimitParameters SubscribePositionLimitAlarms SubscribeMarginLimitParameters SubscribeMarginLimitAlarms
Clearing Data Manager	2100	SubscribeSeries SubscribePositions SubscribeRectifications SubscribePositionTransfers

Programmer's Manual

December 2020

Service	ServiceID	Subscriptions
		SubscribeContracts SubscribeContractTransfers SubscribeOpenCloseContractChanges SubscribeClientCodeContractChanges SubscribeSplitContracts SubscribeCollateralGuarantees SubscribeDepositedGuarantees SubscribeEarlyExercises SubscribeExByEx SubscribeExerciseAtExpiry SubscribeAssignments SubscribeAssignmentsSent SubscribeClearingMessages SubscribeIntradayMarginCalls SubscribeIntradayMarginCallsSent SubscribeSubAccountTransfers SubscribeSubAccountParameters SubscribeSubAccountClientCodeLinks SubscribeSubAccountClientCodeLinkChange SubscribeGiveOutParameters SubscribeTakeUpParameters
Report Manager	2200	SubscribeReport
Transactional Gateway	2300	-
Realtime Gateway	2400	-

Programmer's Manual

December 2020

Service	ServiceID	Subscriptions
Sola Gateway	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
10:00 / 15:00	The component Report Manager crashes; one or more messages with status <code>CONNECTION_CRASH</code> and <code>ServiceId=2200</code> are received.
10:05 / 15:05	The component Report Manager is restored; one or more messages with status <code>CONNECTION_UP</code> and <code>ServiceId=2200</code> are received.
10:15 / 15:15	The component Realtime Gateway crashes; one or more messages with status <code>CONNECTION_CRASH</code> and <code>ServiceId=2400</code> are received.
10:20 / 15:20	The component Realtime Gateway is restored; one or more messages with status <code>CONNECTION_UP</code> and <code>ServiceId=2400</code> are received.

Programmer's Manual

December 2020

GMT Time	Description
10:30 / 15:30	The component Transactional Gateway crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2300 are received.
10:35 / 15:35	The component Transactional Gateway is restored; one or more messages with status CONNECTION_UP and ServiceId=2300 are received.
10:45 / 15:45	The component Clearing Data Manager crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2100 are received.
10:50 / 15:50	The component Clearing Data Manager is restored; one or more messages with status CONNECTION_UP and ServiceId=2100 are received.
11:00 / 16:00	The component Risk Management crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2000 are received.
11:05 / 16:05	The component Risk Management is restored; one or more messages with status CONNECTION_UP and ServiceId=2000 are received.
11:15 / 16:15	The component Sola Gateway crashes; one or more messages with status CONNECTION_CRASH and ServiceId=2500 are received.
11:20 / 16:20	The component Risk Management is restored; one or more messages with status CONNECTION_UP and ServiceId=2500 are received.

Programmer's Manual

December 2020

GMT Time	Description
	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`.

Each and all information contained in this document are confidential, legally privileged and protected by applicable law. Any disclosure, distribution, copying or other diffusion of this communication is strictly prohibited. If you have received this document or part of it in error, are not the intended recipient, nor an employee or agent responsible for delivering this message to the intended recipient, please immediately notify Borsa Italiana S.p.A., at service-desk@borsaitaliana.it. Your co-operation is appreciated.

Contacts

Service Desk Italy, Borsa Italiana
Client Technology Services Italy, LSEG
Email service-desk@borsaitaliana.it
www.borsaitaliana.it



London
Stock Exchange Group