



**GENESIS**  
SECURITIES , LLC.

**GENESIS SECURITIES, LLC**  
**LASER STANDARD API.**

**DOCUMENTATION**

**Version 1.0**

**This document is provided without any warranties explicit or implied. Use at your own risk.**

## Functions

**Function**  
**int WINAPI gtbInitialize();**

**Example**  
gtbInitialize();

**Description / Parameters**  
Initializes the API

**Return values:**  
0: success  
non-zero: fail

**Function**  
**int WINAPI gtbUninitialize();**

**Example**  
gtbUninitialize();

**Description / Parameters**  
Un-Initializes the API

**Return values:**  
0: success  
non-zero: fail

**Function**  
**HGTBSESSION WINAPI gtbCreateSession (**  
    HWND hParent,  
    const GTBSessionCallback \*IpCallBack,  
    void \*IpParam  
**);**

**Example**  
HWND hDlg;  
hDlg = ...; /\* Initialize window handle \*/  
GTBSessionCallback callback;  
/\* Initialize callback \*/  
callback.pfnOnConnect = ...  
gtbCreateSession (hDlg, &callback, 0);

**Description / Parameters**  
Creates The Session  
Parent window handle  
A pointer to a callback structure  
User parameter

**Return values:**  
NULL: fail  
Non-NULL: a handle of session

**Function**  
**int WINAPI gtbDeleteSession(**  
    HGTBSESSION hSession  
**);**

**Example**

**Description / Parameters**  
Deletes current session  
Current Session Handle

**Return values:**

```
gtbDeleteSession();
```

0: success  
non-zero: fail

---

**Function**

```
HGTBSESSION WINAPI gtbSubClassSession(  
    HWND hParent,  
    UINT nID,  
    const GTBSessionCallback *lpCallback,  
    void *lpParam  
);
```

**Description / Parameters**

Attach an API Session to an existing window dynamically. This window class name must be "GTSessionWindow".  
Parent window handle  
Child window ID  
A pointer to a callback structure  
User parameter

**Example**

```
HWND hDlg;  
hDlg = ...; /* Initialize window handle */  
GTBSessionCallback callback;  
/* Initialize callback */  
callback.pfnOnConnect = ...  
gtbSubClassSession(hDlg, IDC_SESSION, &callback, 0);
```

**Return values:**

NULL: fail  
Non-NULL: a handle of session

---

**Function**

```
int WINAPI gtbUnsubClassSession(  
    HGTBSESSION hSession  
);
```

**Description / Parameters**

Detach an API session.  
Current Session Handle

**Example**

```
gtbUnsubClassSession();
```

**Return values:**

0: success  
non-zero: fail

---

**Function**

```
BOOL gtbIsLoggedIn(  
    HGTBSESSION hSession  
);
```

**Description / Parameters**

Status of Login (Boolean).  
Current Session Handle

**Example**

```
if(gtbIsLoggedIn(hSession) == TRUE)  
{ /* it is logged in now */ }
```

**Return values:**

True: login  
False: not login

**Function**

```
int WINAPI gtbLogin (
    HGTBSESSION hSession,
    const char *pszUserID,
    const char *pszPassword
);
```

**Example**

```
gtbLogin("USERID", "passWord");
```

**Description / Parameters**

Used to log-into and establish a connection with Laser API  
 Current Session Handle  
 Your UserID (must be UPPERCASE)  
 Your Password (case sensitive)

**Return values:**

0: Login message is sent  
 non-zero: fail

**Function**

```
int WINAPI gtbLogout (
    HGTBSESSION hSession
);
```

**Example**

```
gtbLogout();
```

**Description / Parameters**

Logs out the user from Laser API. User should wait for a while until disconnect message is received.  
 Current Session Handle

**Return values:**

0: success  
 non-zero: fail

**Function**

```
int WINAPI gtbSubscribe(
    HGTBSESSION hSession,
    const char *pszStock,
    BOOL bLevel2
);
```

**Example**

```
gtbSubscribe(hSession, "INTC", TRUE);
```

**Description / Parameters**

Subscribes to receive market data for a specified symbol. A symbol must be subscribed to before orders are sent, as well.  
 Current Session Handle  
 Symbol  
 Subscribe to Level II & Level I data if True; only Level I if False

**Return values:**

0: success  
 non-zero: fail

**Function**

```
int WINAPI gtbUnsubscribe(
    HGTBSESSION hSession,
    const char *pszStock
);
```

**Example**

```
gtbUnsubscribe(hSession, "INTC");
```

**Description / Parameters**

Unsubscribes from receiving market data for specified symbol  
 Current Session Handle  
 Symbol

**Return values:**

0: success  
 non-zero: fail

```
gtbUnsubscribe(hSession, "INTC");
```

0: success  
non-zero: fail

<p><b>Function</b>  <b>int WINAPI gtbPlaceOrder</b>(  HGTBSESSION hSession,  const char *pszStock,  char chSide,  int nShares,  double dblPrice,  const char *pszMethod,  unsigned long dwOrderID,  const char *pszOrderOptions  );</p>	<p><b>Description / Parameters</b>  Places The Order  Current Session Handle  Symbol  Side: B for Buys; S for Sells and Short Sells  Number of Shares  Price of the order. Can use any resolution that ECNs support. Use 0 for market orders  Route for the order (ECN, DOT line, etc.)  User can assign an order ID to this order  Custom order options.</p>
---	---

**CUSTOM ORDER OPTIONS:**

Custom order options are entered in the following format: Setting = Value; Setting = Value..

Setting	Values	Description
Visible	'Y' or 'N'	IF N, order will go out as a hidden order. Not all ECNs support hidden orders
TIF	0 through 99999	Time in force for the order, in seconds. 0 for IOC (immediate or cancel order), 99999 for a DAY order
Reserve	number of shares	Set number of shares to be shown with the rest as reserve
AutoEX	0 = off, 1 = on	SOES AutoEx
Place	NYSE or AMEX	DOT line routes must specify a designated exchange
IncaAutoRoute	0 = off, 1 = on	If On, all INCA orders will automatically become proactive (route-out)
ArcaAutoRoute	0 = off, 1 = on	If On, all ARCA orders will automatically become proactive (route-out)

**Example**  
gtbPlaceOrder(hSession, "INTC", 'B', 100, 25.00, "ISLD", 0, "TIF=99999");

**Return values:**  
0: Order has been sent  
non-zero: fail

**Function**  
**int WINAPI gtbCancelAll**(  
HGTBSESSION hSession  
);

**Description / Parameters**  
Cancels all pending orders  
Current Session Handle

**Example**  
gtbCancelAll(hSession);

**Return values:**  
0: cancel command has been sent

non-zero: fail

**Function**  
**int WINAPI gtbCancelTicket**(  
    HGTBSESSION hSession  
    long dwTicketNo  
);

**Description / Parameters**  
Cancels order by ticket number  
Current Session Handle  
Ticket Number of the order

**Example**  
gtbCancelTicket(hSession, 100);  
*where 100 is the ticket number*

**Return values:**  
0: cancel command has been sent  
non-zero: fail

**Function**  
**int WINAPI gtbCancelStockAll**(  
    HGTBSESSION hSession,  
    const char \*pszStock  
);

**Description / Parameters**  
Cancels all pending orders on given symbol  
Current Session Handle  
Symbol

**Example**  
gtbCancelStockAll(hSession, "INTC");

**Return values:**  
0: cancel command has been sent  
non-zero: fail

**Function**  
**int WINAPI gtbSetAddress**(  
    HGTBSESSION hSession,  
    int nServerType,  
    const char \*pszIPAddress,  
    unsigned short nPort  
);

**Description / Parameters**  
Sets IP address and port number.  
Current Session Handle  
Refer to Server types  
IP address of server  
Port number of server

**Example**  
gtbSetAddress("10.1.3.100", 16805);

**Return values:**  
0: success  
non-zero: fail

**Function**  
**int WINAPI gtbTranslateToTickets**(  
    HGTBSESSION hSession,  
    unsigned long dwID,  
    int count,  
    long \*pdwTickets  
);

**Description / Parameters**  
Translates the user order ID to ticket number  
Current Session Handle  
User order ID  
Array length of pdwTickets  
Pointer to ticket number array

**Example**

```
unsigned long dwOrderID = ...;
long dwTickets[10];
gtbTranslateToTickets(hSession, dwOrderID, 10, &dwTickets);
```

**Return values:**

-1: error  
Any other value: Number of tickets

**Function**

```
unsigned long WINAPI gtbTranslateToOrderID(
    HGTBSESSION hSession,
    long dwTicketNo
);
```

**Description / Parameters**

Translates the ticket number to a user order ID  
Current Session Handle  
Ticket number

**Example**

```
gtbTranslateToOrderID(hSession, 100);
```

**Return values:**

-1: Error  
0: Not found  
Any Other Value: Ticket No

**Function**

```
void WINAPI gtbGetFirstOpenPosition(
    HGTBSESSION hSession,
    char *pszSymbol,
    int *pnShares,
    char *pchSide,
    double *pdblPrice
);
```

**Description / Parameters**

Gets the first open position  
Current Session Handle  
Symbol  
Number of Shares  
Side of position. B for Longs, T for Sells and Short Sells  
Price of the order

**Example****Return values:**

NULL: fail or not found  
Non-NULL: a handle of context

**Function**

```
void WINAPI gtbGetNextOpenPosition(
    HGTBSESSION hSession,
    void *pos,
    char *pszSymbol,
    int *pnShares,
    char *pchSide,
```

**Description / Parameters**

Gets the next open position  
Current Session Handle  
Starts from This Position  
Symbol  
Number of Shares  
Side of position. B for Longs, T for Sells and Short Sells

```

    double *pdblPrice          Price of the order
);

```

**Example**

```

void *pos;

pos = gtbGetFirstOpenPosition(hSession, szSymbol, &nShares, &chSide, &dblPrice);
while(pos != NULL)
{
    /* Save open positions here */

    pos = gtbGetFirstNextPosition(hSession, pos, szSymbol, &nShares, &chSide, &dblPrice);
}

```

**Return values:**  
 NULL: fail or no more found  
 Non-NULL: a handle of context

**Function**

```

void WINAPI gtbGetFirstFill(
    HGTBSESSION hSession,
    long *pdwTicketNo,
    char *pszSymbol,
    int *pnShares,
    char *pchSide,
    double *pdblPrice,
    char *pszMethod,
    char *pszPlace,
    int *pnTime,
    char *pszMatchNo
);

```

**Description / Parameters**

Gets the first Fill  
 Current Session Handle  
 Ticket Number of the Order  
 Symbol  
 Number of Shares  
 Side of position. B for Longs, S for Sells and Short Sells  
 Price of the order  
 Route of Order (ECN or DOT line, Etc)  
 Place of the order. Applicable to DOT orders.  
 Timestamp of the order  
 Match Number of the Fill

**Return values:**  
 NULL: fail or not found  
 Non-NULL: a handle of context

**Function**

```

void WINAPI gtbGetNextFill(
    HGTBSESSION hSession,
    void *pos,
    long *pdwTicketNo,
    char *pszSymbol,
    int *pnShares,
    char *pchSide,
    double *pdblPrice,
    char *pszMethod,
    char *pszPlace,
    int *pnTime,
);

```

**Description / Parameters**

Gets next fill  
 Current Session Handle  
 Starts from this position  
 Ticket Number of the order  
 Symbol  
 Number of Shares  
 Side of position. B for Longs, S for Sells and Short Sells  
 Price of the order  
 Route of Order (ECN or DOT line, Etc)  
 Place of the order. Applicable to DOT orders.  
 Timestamp of the order

```
);
    char *pszMatchNo
```

Match Number of the Fill

**Return values:**

NULL: fail or no more found  
 Non-NULL: a handle of context

**Example**

```
void *pos;

pos = gtbGetFirstFill(hSession, &dwTicketNo, szSymbol, &nShares, &chSide, &dblPrice, szMethod, szPlace, &nTime, szMatchNo);
while(pos != NULL)
{
    /* Save fills here */

    pos = gtbGetNextFill(hSession, pos, &dwTicketNo, szSymbol, &nShares, &chSide, &dblPrice, szMethod, szPlace, &nTime, szMatchNo);
}

```

**Function**

**void WINAPI gtbGetFirstPending(**

```
HGTBSESSION hSession,
long *pdwTicketNo,
char *pszSymbol,
int *pnShares,
char *pchSide,
double *pdblPrice,
char *pszMethod,
char *pszPlace,
int *pnTime
```

**);**

**Description / Parameters**

Gets first pending order  
 Current Session Handle  
 Ticket Number of the order  
 Symbol  
 Number of Shares  
 Side of position. B for Longs, S for Sells and Short Sells  
 Price of the order  
 Route of Order (ECN or DOT line, Etc)  
 Place of the order. Applicable to DOT orders.  
 Timestamp of the order

**Return values:**

NULL: fail or not found  
 Non-NULL: a handle of context

**Function**

**void WINAPI gtbGetNextPending(**

```
HGTBSESSION hSession,
void *pos,
long *pdwTicketNo,
char *pszSymbol,
int *pnShares,
char *pchSide,
double *pdblPrice,
char *pszMethod,
char *pszPlace,
int *pnTime
```

**);**

**Description / Parameters**

Gets next fill  
 Current Session Handle  
 Starts from this position  
 Ticket Number of the order  
 Symbol  
 Number of Shares  
 Side of position. B for Longs, S for Sells and Short Sells  
 Price of the order  
 Route of Order (ECN or DOT line, Etc)  
 Place of the order. Applicable to DOT orders.  
 Timestamp of the order

**Return values:**

NULL: fail or no more found

Non-NULL: a handle of context

**Example**

```
void *pos;

pos = gtbGetFirstPending(hSession, &dwTicketNo, szSymbol, &nShares, &chSide, &dblPrice, szMethod, szPlace, &nTime,);
while(pos != NULL)
{
    /* Save pending orders here */

    pos = gtbGetNextPending(hSession, pos, &dwTicketNo, szSymbol, &nShares, &chSide, &dblPrice, szMethod, szPlace, &nTime);
}
```

## CallBack Functions.

For callback functions, all parameters are output; return values are ignored.

### Call Back Function

```
void (WINAPI *gtbOnConnect)(  
    HGTBSESSION hSession,  
    void *lpParam,  
    int nServer,  
    int nState  
);
```

### Description / Parameters

Current Session Handle  
User parameter  
Server ID  
Server State

### Call Back Function

```
void (WINAPI *gtbOnOrderChanged)(  
    HGTBSESSION hSession,  
    void *lpParam,  
    int nOrderType,  
    unsigned long dwOrderID,  
    long dwTicketNo,  
    const char *pszSymbol,  
    int nShares,  
    char chSide,  
    double dblPrice,  
    const char *pszMethod,  
    const char *pszPlace,  
    int nTime,  
    const char *pszParam  
);
```

### Description / Parameters

Current Session Handle  
User parameter  
Order Type  
User assigned order ID  
Ticket number for the order  
Symbol  
Number of Shares  
Side of the order. B for Long, T for shorts and short-sells  
Price of the order  
Route of the order  
Place of the order  
Time of the order  
Reference Number or Match No

### Call Back Function

```
void (WINAPI *gtbOnAccountChanged)(  
    HGTBSESSION hSession,  
    void *lpParam,  
    const char *pszAccountID,
```

### Description / Parameters

Current Session Handle  
User parameter  
Account ID

```
);
    double dbIBuyingPower
    Buying Power for selected Account ID
```

#### Call Back Function

```
void (WINAPI *gtbOnErrorMessage)(
    HGTBSESSION hSession,
    void *lpParam,
    int nErrCode,
    unsigned long dwOrderID,
    const char *pszDesc
);
```

#### Description / Parameters

Current Session Handle  
 User parameter  
 Error Code  
 User Defined Order ID  
 Error Description

#### Call Back Function

```
void (WINAPI *gtbOnGotLevel1)(
    HGTBSESSION hSession,
    void *lpParam,
    const char *pszStock,
    long dwBestBidShares,
    double dbIBestBidPrice,
    char chBestBidExchangeCode,
    long dwBestAskShares,
    double dbIBestAskPrice,
    char chBestAskExchangeCode,
    char chReserved
);
```

#### Description / Parameters

Current Session Handle  
 User parameter  
 Symbol  
 Number of shares at the best bid  
 Current Best Bid  
 The Exchange of the Best Bid  
 Number of shares at the best ask  
 Current best ask  
 The Exchange of the Best Ask  
 Not used.

#### Call Back Function

```
void (WINAPI *gtbOnGotPrint)(
    HGTBSESSION hSession,
```

#### Description / Parameters

Current Session Handle

<pre> void *lpParam, const char *pszStock, long dwShares, double dblPrice, char chExchangeCode, char chSource, char chSaleCondition ); </pre>	<pre> User parameter Symbol Shares Price Exchange Code Source of the print Sale Condition </pre>
---	--



**Call Back Function**

```

void (WINAPI *gtbOnGotLevel2)(
    HGTBSESSION hSession,
    void *lpParam,
    const char *pszStock,
    long dwShares,
    char chSide,
    double dblPrice,
    const char *pszMMID,
    int nTime
);

```

**Description / Parameters**

```

Current Session Handle
User parameter
Symbol
Shares
Side of the order. B for Long, T for shorts and short-sells
Price
Market Maker ID (or ECN ID)
Time

```

## Callback Functions Structure

GTBSessionCallback is a structure, which consists of some callback function variables. Users can define their own callback functions, and give their addresses to this structure's members. When something happens in API, a corresponding function will be called.

### Sample:

#### User function:

```
void WINAPI onConnect(HGTBSESSION hSession, void *lpParam, int nServer, int nState)
{
    /* some process ... */
}
```

#### Initialize callback structure:

```
GTBSessionCallback callback =
{
    onConnect,
    /* other fuctions */
};
```

#### Pass callback structure to gtbCreateSession() like this:

```
HGTBSESSION g_hSession = NULL;
g_hSession = gtbCreateSession(hDlg, &callback, (void *)hDlg);
```

### Order Types

Below are the states in which an order can exist

### Order States

GTAPIB_ORDER_RECORDED	1	Order is recorded
GTAPIB_ORDER_PENDING	2	Order is pending
GTAPIB_ORDER_FILLED	3	Order is filled
GTAPIB_ORDER_OPEN	4	Order is open
GTAPIB_ORDER_CANCELLING	5	Order is being cancelled
GTAPIB_ORDER_CANCELLED	6	Order is cancelled
GTAPIB_ORDER_REJECTED	7	Order is Rejected
GTAPIB_ORDER_REMOVED	8	Order is Removed

## Server IDs

Server IDs are used to notify the user of the connection status to each server. Using these IDs, users are notified if the servers are up or down.

### Server IDs

---

GTAPIB_CONNECT_EXECUTOR	Connection to Laser's Execution Engine
GTAPIB_CONNECT_LEVEL1	Connection to the Level 1 Data Server
GTAPIB_CONNECT_LEVEL2	Connection to the Level 2 Server
GTAPIB_SERVER_ISB	ID of INET server
GTAPIB_SERVER_NOQS	ID of NOQS server
GTAPIB_SERVER_BRB	ID of Brut Book
GTAPIB_SERVER_ARB	ID of Arca Book
GTAPIB_SERVER_SIAC	ID of SIAC server
GTAPIB_SERVER_OPENBOOK	ID of OpenBook server
GTAPIB_SERVER_ITC	ID of INET's ITCH server
GTAPIB_SERVER_OUCH	ID of INET's OUCH server
GTAPIB_SERVER_SOES	ID of NASDAQ's SuperMontage server
GTAPIB_SERVER_BRUT	ID of Brut's server
GTAPIB_SERVER_ARCA	ID of Arca's server
GTAPIB_SERVER_INCA	ID of INCAs server
GTAPIB_SERVER_TRAC	ID of TRACK ECN's server
GTAPIB_SERVER_BELZ	ID of Belzberg's server (DOT line)
GTAPIB_SERVER_MLYN	ID of Merrill Lynch's server (DOT LINE)
GTAPIB_SERVER_ERCO	ID of Essex ERCO server
GTAPIB_SERVER_PPLN	ID of Pipeline's server
GTAPIB_SERVER_QUOTE	ID of Laser's Quote Server
GTAPIB_SERVER_TRANS	ID of Laser's Transaction Server

## Server States

Below are the possible states of the API server

### Server States

GTAPIB_STATE_DISCONNECT	specifies the current state of the server connection to the API as disconnected
GTAPIB_STATE_UNKNOWN	specifies the current state of the server connection to the API as unknown
GTAPIB_STATE_CONNECTED	specifies the current state of the server connection to the API as connected
GTAPIB_STATE_READY	specifies the current state of the server connection to the API as ready
GTAPIB_STATE_NOT_READY	specifies the current state of the server connection to the API not ready

## Session Handle

Each connection to the API needs to generate a session handle. This session handle is referenced to in most functions

```
typedef struct {void *unused;} *HGTBSESSION;
```

## Sale Conditions

*Sale conditions, under the call-back function XXX, are codes from SIAC and NQDS and are reflected below. You should be familiar with the explanations of these Codes as they come from the exchanges, not from Genesis.*

Sale Condition Code Values from SIAC		Sale Condition Code Values from NQDS	
@	Regular Sale (no condition)	@	Regular Trade
A	Cash (only) Market	A	Acquisition
B	Average Price Trade	B	Bunched Trade
C	Cash Trade (same day clearing)	C	Cash Trade
D	Next Day (only) Market	D	Distribution
E	Automatic Execution	G	Bunched Sold Trade
F	Burst Basket Execution	K	Rule 155 Trade (AMEX only)
G	Opening/Reopening Trade Detail	L	Sold Last
H	Intraday Trade Detail	N	Next Day
I	Basket Index on Close Transaction	O	Opened
J	Rule 127 Trade (NYSE)	P	Prior Reference Price
K	Rule 155 Trade (Amex)	M	Market Center Close Price
L	Sold Last (late reporting)	R	Seller (Long-Form Message Formats Only)
N	Next Day Trade (next day clearing)	S	Split Trade
O	Opened (late report of opened trade)	T	Form-T Trade
R	Seller	W	Average Price Trade
S	Reserved	Z	Sold (Out of Sequence)
T	Pre/Post Market Trade	1	Stopped Stock – Regular Trade
Z	Sold (out of sequence)	2	Stopped Stock – Sold Last
		3	Stopped Stock – Sold (Out of Sequence)

## Methods

**Laser** has many routes that you can use to send orders. Below are the current routes with their codes and a brief explanation of each. Note: there may be different fees associated with using different routes.

METHOD ID	NAME	DESCRIPTION
ISLD	INET ATS	INET ECN
INCA	INCA Routing System	If routing options for INCA are off, the order functions as a regular INET order
ARCA	Archipelago	ARCA ECN
BRUT	Brut ECN	BRUT ECN
TRAC	TRACK ECN	SuperMontageable ECN: TRACK
SOES	NASDAQ ANONYMOUS ECN	SuperMontageable ECN. Also used to send SuperMontage Orders
BELZ	Belzberg Dot Line SuperDot	Direct line to the NYSE and AMEX using a regular SuperDot order
BLZX	Belzberg Dot Line Direct+	Direct line to the NYSE using a Direct+ (NX) order
MLCO	Merrill Lynch Dot Line SuperDot	Direct line to the NYSE and AMEX using a regular SuperDot order
MLNX	Merrill Lynch Dot Line Direct+	Direct line to the NYSE using a Direct+ (NX) order
ERCO	Essex Radez Order	Order using Essex Radez Provider (contact account admin for access)
PPLN	PipeLine Order	Order using Pipeline connection (contact account admin for access)
OES	OES Dot Line SuperDot	Direct line to NYSE/AMEX using regular SuperDot Order. Able to send orders to regional exchanges. Use Place parameter for regional exchanges
OESX	OES DOT Line Direct+	Direct line to the NYSE using a Direct+ (NX) order
NYF	NYFIX	Connection to NYFIX

**NOTE: THERE MAY BE MORE AVAILABLE METHODS AS LASER IS ALWAYS INCREASING ITS CONNECTIVITY**