

# STIM – StereoTrader Installation Manager

## Internal- and Product-Vendor-Documentation

### 1. STIM

STIM, written in C#, is an application for installation of expert advisors (StereoTrader) and indicators (SEA). The executable version of STIM is downloaded to the users PC and retrieves all assets and files of the product, which is to be installed, via HTTP online.

STIM is hosted by StereoTrader company only, whereby products and their contents can either be hosted by StereoTrader company or by 3rd party providers – the vendors.

The location of STIM application and its own assets is hard-wired, currently at [download.stereotrader.net/installer/\\_StereoTrader\\_Installation\\_Manager/](http://download.stereotrader.net/installer/_StereoTrader_Installation_Manager/)

With each installation of any product, also STIM is downloaded to the users PC within the MetaQuotes Terminal directory, such as

```
C:\Users\Developer\AppData\Roaming\MetaQuotes\Terminal\Common\Files\STMTX\STIM\
```

#### 1.1 Assets of STIM (*internal – non-confidential*)

- stim.exe – the executable
- config.xml – configuration data of STIM itself, including languages and version
- eula.pdf

#### 1.2 Proceeding of STIM (*internal – non-confidential*)

When STIM is started, it accesses the sts.php script of StereoTrader to retrieve the STIM vendor data in the first step. The script returns a dataset (enumerated string expression) which includes

- Numer of vendors (cnt=)
- STIM application URL (stim\_app\_url=)
- IDs (string) of all vendors which offer products which will be listed within STIM (id=)
- Name of vendor (name=)
- Homepage of vendor (homepage=)
- URL of products (stim\_url=)
- All variables as set in the custom vendor.xml

For detailed information about sts.php, please refer to the according internal documentation of the script. This information is not needed by vendors to provide a product.

Based on this information, STIM retrieves all the data of installable products by such URLs and lists the products.

#### 1.3 Vendor ID and workspace

The vendor ID is unique and is provided by StereoTrader company after applying for a vendorship. Every vendor also retrieves a password, which is used to access further services connected to product licensing and version management.

Furthermore, every vendor which provides own products, gets access to their own workspace for uploading and maintaining product data.

##### 1.3.1 Vendors workspace

This folder/workspace needs to provide two .XML files: The products.xml which contains all products and the vendor.xml file, which contains informations about the vendor. Furthermore, all the source of products including their setup-instructions are stored within subfolders of this workspace. Usually it looks like such

```

vendor.xml                > Custom vendor data
products.xml              > Products table

[sea4myproduct]           > Product folder
    setup.xml             > Setup data of product
    mysea.ex4             > Product itself (MT4 version)
    whatsnew.rtf          > Changes
    eula.rtf              > License agreement

[sea5myproduct]           > Product folder
    setup.xml             > Setup data of product
    mysea.ex5             > Product itself (MT5 version)
    whatsnew.rtf          > Changes
    eula.rtf              > License agreement

```

## 1.4 The products table (products.xml)

The file is provided and maintained by the vendors themselves and lists all available products. The structure is shown below:

Node	Type	Value
product	Element	
id	Attribute	sea5fluxx
name	Attribute	Fluxx
description	Attribute	Fluxx
localname	Attribute	
target	Attribute	MT5
type	Attribute	0
licensing	Element	
stl_id	Attribute	53
dgs24_ids	Attribute	434779
dgs24_translationurl	Attribute	
dgs24_checkaccount	Attribute	0
brokers_wol_tester	Attribute	*
brokers_wol_demo	Attribute	*
brokers_wl_tester	Attribute	*
brokers_wl_demo	Attribute	*
brokers_wl_live	Attribute	*
version	Element	
version	Attribute	2.6
build_cur	Attribute	2670
build_prm	Attribute	2670
build_min	Attribute	2670

- **id** – internal product id to identify the product in view of version management. The id is also the default name of the directory which contains the contents of the product including the setup.xml for the product to be installed (mandantory)
- **name** – the screenname of the product (mandantory)
- **description** – Short description of the product (optional)
- **url** – The URL of the product (optional)
- **url\_purchase** – The direct purchase URL (if any)
- **localname** – (optionally) directory which contains the contents of the product
- **target** – either MT4 or MT5 (mandantory)
- **type** –
  - **0** for experts or indicators (standard)
  - **1** for experts or indicators listed when STIM is executed in Beta-Mode,
  - **2** for patches of MT, listed when STIM is executed in patch mode,

- **3** for hidden products used in nested installations and not accessible as single product by users,
- **10** for experts or indicators while developing/testing, not visible to the public
- **version** section – version informations used by the script
- **licensing** section – informations for license verifications

Functionalities for version and license management are accessible for vendors by using `__STSApp.mqh` in their products. The file is provided by StereoTrader company and is not part of the standard StereoTrader API.

#### 1.4.1 Testing

To test installations (type=10), the developer must provide a config file within the directory where also `stim.exe` is located and executed, like

```
C:\Users\Developer\AppData\Roaming\MetaQuotes\Terminal\Common\Files\STMTX\STIM\
```

This name of this file must be „stim.dev“, must be of plain text and must contain the vendors id and password. The content looks like:

```
vendor_id=myid;vendor_pass=mypassword;
```

Whereby myid and mypassword are placeholders for the credentials of the corresponding vendor.

#### 1.4.2 Version management

The version section consists of

- **version** – product version displayed
- **build\_cur** – current build
- **build\_prm** – promoted build. In case the current version of the product differs from build\_prm, the user gets a recommendation to update
- **build\_min** – minimal build. In case the currently installed build is below the minimum build, the user is forced to update

#### 1.4.3 License management

The licensing section consists of

- **stl\_id** – product id within StereoLicenser, which is a separate application for manual management of licensing
- **dgs24\_ids** – DigiStore24 product ids of this product as list separated with „;“
- **dgs24\_translationurl** – (beta)
- **dgs24\_checkaccount** – 0/1 – in case the DigiStore24 license verification shall compare the name of the account with the name of the buyer. If there are no similarities, the license-check would fail
- **brokers\_wol\_tester** – List of brokers which are allowed to use the product within the strategy tester without license (wol). The names may contain wildcards with an asterisk „\*“ and are to be separated by semicolons „;“. If this field is empty, no broker is allowed. To allow any broker, a single asterisk „\*“ should be used.
- **brokers\_wol\_demo** – List of brokers which are allowed to use the product with a demo account without license (wol). Format is the same as above.
- **brokers\_wol\_live** – List of brokers which are allowed to use the product with a live account without any license (wol). Format is the same as above.
- **brokers\_wl\_tester** – List of brokers which are allowed to use the product within the strategy tester with valid license (wl).

- **brokers\_wl\_demo** – List of brokers which are allowed to use the product with demo accounts with valid license (wl).
- **brokers\_wl\_live** – List of brokers which are allowed to use the product with live accounts with valid license (wl).

The fields `brokers_*` use a masked list of names, separated by „;“. Masked means, a wildcard with an asterisk can be used. Such a list could look like

```
Banana*;FrenchFrie*;*XFX;
```

This way, only the first or the last part of the name must match the name of the broker. The verification of the names is not case sensitive.

#### 1.4.4 Usage of `_STSApp.mqh`

`_STS.mqh` / `_STSApp.mqh` are provided to vendors to handle version and license management within their software. The file contains two global objects

- **`_STS`** (class `CSTSPhp`) – native execution of script, usually not needed. This class is also available for C# (file `sts_php.cs`)
- **`_App`** (class `CSTSApp`) – used for license testing within SEAs and MQL in general, inherited by regular `CApp`

The process is self explaining and needs just a few lines of code:

##### a. Full/automated implementation within SEAs

Since the constructor of `CSTSApp` is able to read macros, they should be used to define all the parameters of the application at the beginning of its code. In this case, no additional code at all is needed to check version or license within an SEA.

The definition with macros look like:

```
#define APP_STS // Embed STS server based version and licensing
#define APP_NAME "MySEA" // Display name
#define APP_VERSION "2.7" // Display version
#define APP_BUILD 2700 // Current build
#define APP_INTERNALNAME "MYSEA" // Internal name (optional)
#define APP_EXPIRATIONDATE D'2023.06.30' // Date of general expiration (optional)
#define APP_VENDORID "st" // Vendor/manufacturer Id also used in STIM
#define APP_PRODUCTID "sea%mysea" // Product Id also used with STIM
#define APP_VERIFICATIONMODE 0xFFFF // Supported verification modes:
// 0 none / 1 license / 2 version
```

The StereoTrader API will apply the keys automatically and will do the license and version verification without any additional effort within `SEA_VERIFYPRODUCT` (`OnVerifyProduct()`)

##### b. Manual implementation

Besides the way with macros/definitions, the properties can also set using the according functions.

```
_App.Vendor_Id("vendorid");
_App.App.Product_Id("stim_productid");
_App.Product_Name("MyProduct");
_App.Product_Build("vendorid");
_App.Keys(allavailablecodes);
```

Verification would then be like:

\_\_App.VerifyBundle();

... or

\_\_App.VerifyVersion();

\_\_App.VerifyLicense();

Result evaluation:

\_\_App.IsVersionInvalid();

... true when version is obsolete and cannot be used anymore at all. In such case, the version request, if VerifyBundle() was used, returned already an invalid license additionally.

\_\_App.IsLicenseValid();

... true when license is valid/active

\_\_App.IsVersionOutdated();

... true when version is below promoted build and user should update

further functions:

\_\_App.IsVersionCurrent();

... true when version matches the most recent build (usually not needed)

\_\_App.IsVersionMatch();

... true when version matches at least the promoted build (usually not needed)

\_\_App.IsVersionUpdateNeeded();

... true when version is below promoted build or invalid

\_\_App.Content();

... the content string of the last request containing explicitly build and/or license data

## 1.5 Custom vendor data (vendor.xml)

Within this file, the vendor is described. Furthermore it defines variables for using with the sync process used for automated uploads using STIM or CodeBitch.

Node	Type	Value
▼ root	Element	
▼ stim	Element	
▼ origin	Element	
id	Attribute	mtcommon
value	Attribute	C:\Users\Developer\AppData\Roaming\MetaQuotes\Terminal\Common
► origin	Element	
► origin	Element	
► origin	Element	
▼ main	Element	
name	Attribute	Leveredge Ltd.
homepage	Attribute	https://www.stereotrader.net
▼ dgs	Element	
name	Attribute	stereotrader
apikey	Attribute	35845-jcMcBQA/hdjBIUSeaxw8YVWkqJJJ3uOxosxOm4
s_locking	Attribute	1
s_testdays_own	Attribute	14
s_testdays_3rd	Attribute	7
s_priceth_fraud	Attribute	50
s_priceth_suspect	Attribute	95

The structure:

- **stim/origin** – defines macros/vars which can be used within the field „origin“ of the setup.xml file to reduce the definitions of sources. When such a variable shall be used, add a ,%' before the name. (origin „%mtcommon/stmtx/myfile.ex4“)
- **main/name** – displayed name
- **main/homepage** – main homepage of vendor
- **dgs/name** – DigiStore24 username
- **dgs/name** – DigiStore24 API key

- **dgs/s\_locking** – Security check on/off. The following values allow for verification in dependence of purchase price and the buying-method. Especially since DigiStore24 offers test-buys for evaluation, this function can be used and controlled by the STIM security checking processes. A license/access will not be granted, in case any rule as defined is violated. Besides this, the security check results can be obtained also manually with the content which is returned by `__App.Content()` after `__App.VerifyBundle()` or `__App.VerifyLicense()` execution.
- **dgs/s\_testdays\_own** – In case of test-buy, days of usage until permission is denied when using it as vendor
- **dgs/s\_testdays\_3rd** – In case of test-buy, days of usage until permission is denied when clients use it
- **dgs/s\_priceth\_fraud** – Percentage value. Fraud is assumed, when the purchase amount is below this threshold.
- **dgs/s\_priceth\_suspect** – Percentage value. A suspicious purchase is assumed, when the purchase amount is below this threshold.

## 2. Maintaining data of installable products

### 2.1 Setup directory

The data is stored in a directory which is hosted by StereoTrader. Each vendor has access to their personal folder. The root must contain the file „products.xml“, whereby each folder should contain only the setup-data for one specific product.

The structure usually looks like the following. The example shows a directory content of a product which id is „sea5fluxx“, the field „localname“ was not used, accordingly the name of the directory is the same as the id.

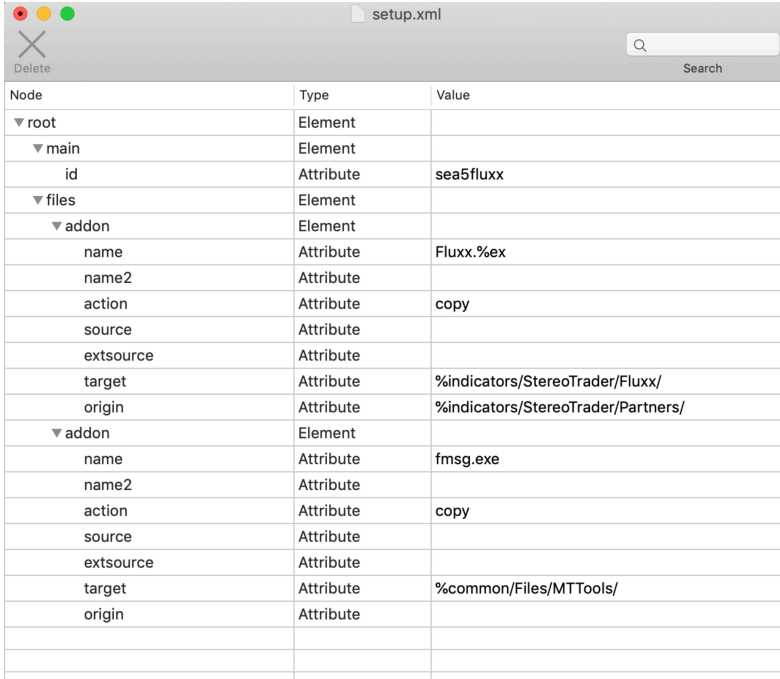


The file „eula.rtf“ contains the End-User-License-Agreement, the file „whatsnew.rtf“ contains the build-history. In case the files are not present, STIM will ignore it.

The setup.xml contains the data for a specific product, listed in the (products-).xml file. The file is located within the directory which contains all the assets for such a product. The presence of the file „setup.xml“ is mandatory.

Fluxx.ex5 in this case is the name of an indicator (SEA) and is listed within the setup.xml file.

## 2.2 Structure of XML content of setup.xml:



Node	Type	Value
▼ root	Element	
▼ main	Element	
id	Attribute	sea5fluxx
▼ files	Element	
▼ addon	Element	
name	Attribute	Fluxx.%ex
name2	Attribute	
action	Attribute	copy
source	Attribute	
extsource	Attribute	
target	Attribute	%indicators/StereoTrader/Fluxx/
origin	Attribute	%indicators/StereoTrader/Partners/
▼ addon	Element	
name	Attribute	fmsg.exe
name2	Attribute	
action	Attribute	copy
source	Attribute	
extsource	Attribute	
target	Attribute	%common/Files/MTTools/
origin	Attribute	

The setup.xml contains only the files to be copied. The XML structure is „setup/files“ or „root/files“. Each further element can be named freely and consists of these attributes:

- **name** – the source filename
- **name2** - the target filename, if empty, same as name
- **action** – the purpose
  - „copy“ – copies file from source to target
  - „copy\_absent“ - copy file when not existing in target folder
  - „delete“ – remove file or directory. In case the name field is left empty, the target folder will be deleted. Please be careful with this command.
  - „setup“ – include another setup of another product. When this is used, the field name includes the product id and source contains the vendor id.
- **source** – the source sub-url, (ends with „/“), in case the file is not located within the base directory. There are also macros available.
- **extsource** – overrides source. This field is only used, when a file shall be downloaded explicitly from another source outside the STIM sandbox.
- **target** – the target sub-directory, (ends with „/“)
- **origin** – the original path where the file is located within the developers folder structure (ends with „/“). This is used for automated/synchronized uploads by STIM or CodeBitch.

Available macros are

- %name (Name of application described at Main->Name)
- %buid
- %version
- %br (Break / new paragraph)
- %target (as specified „MetaTrader 4“ or „MetaTrader 5“)
- %targetno („4“ or „5“)
- %ex – results in .ex4 or .ex5, depending of the target platform
- %mql – mql4/mql4
- %mq – mq4/mq5

- %expert – results in MQLx/Experts
- %indicators - results in MQLx/Indicators
- %include – leads to the include folder
- %profiles – leads to the profiles folder
- %templates – leads to the templates folder
- %files –MQLx/Files folder
- %images – MQLx/Images folder
- %libraries – MQLx/Libraries folder
- %scripts – MQLx/Scripts folder
- %tester – leads to the tester folder
- %logs – guides to logs folder
- %config – to config-folder
- %history – to history data folder
- %desktop – users desktop
- %common – the Terminal/Common folder of MT4/MT5

## 2.3 Trial, Beta, Updates and Releases

Since STIM offers product type 10 (Test) for internal testing of installations, vendors should use this type for debugging and testing before a product is released to the public or even released as beta version. As long a product is tagged with type=10, such product is not visible for clients.

In case a vendor plans an update of a product which consists of a different setup-structure and/or further massive changes, the process of evaluation and updating of products should be as described below.

Lets assume, the current product has the name/id „mySEA“ and the data of the current version – which is version 1.0 and its build number is 1000 – is also located within the folder „mySEA“. Lets also assume, the new version is 2.0 and the new build number is 2000.

Recommended procedure:

### 2.3.1 Preparation:

- A – Duplicate the folder „mySEA“
- B – Rename the copy to „mySEA20“
- C – Edit all setup data within „mySEA20“
- D – Within „products.xml“, duplicate the node of „mySEA“ and rename it to „mySEA20“
- E – The tag „type“ of the new product within products.xml is set to „10“ (=testing)
- F – The values for build\_cur, build\_min and build\_prm are all set to „2000“
- G – The version tag is set to „2.0“
- H – In case licensing is used, make sure the ProductId() within the MQL code is set to „mySEA20“ accordingly

### 2.3.2 Trial period

- L – Test the installation process, licensing etc. until everything works without problems
- M – In case you want to invite some clients for a beta test, set the tag „type“ to value „1“. Then the product is visible also to clients when STIM is executed in beta-mode
- N – Increase the tags „build\_prm“, „build\_min“ and „build\_cur“ with every update during the beta-test period

### 2.3.3 Release

- U – Change the tag „type“ in products.xml to value „0“ – standard
- V – Change the tags „build\_cur“ and „build\_prm“ to the final build number, e. g. „2050“



- W – Set the tag „build\_prm“ of the old product „mySEA“ to value „2050“. All clients will be informed within 24 hours that there is a newer build/version available and that they shall update to build 2050, also the beta testers of the new product

#### **2.3.4 Force update**

In case clients shall be forced to update to version 2.0, after a while, the tag „build\_min“ of the old product „mySEA“ shall be changed to „2050“ then (same value as „build\_prm“ of the new product). In this case, all clients would be forced to install version 2.0.

#### **2.3.5 Parallel versions**

In case clients should have the ability to also stick with version 1.0, the value „build\_prm“ of version 1.0 within „products.xml“ can be set back equally to the value of „build\_cur“. In such case, clients will no longer be reminded to update to version 2.0