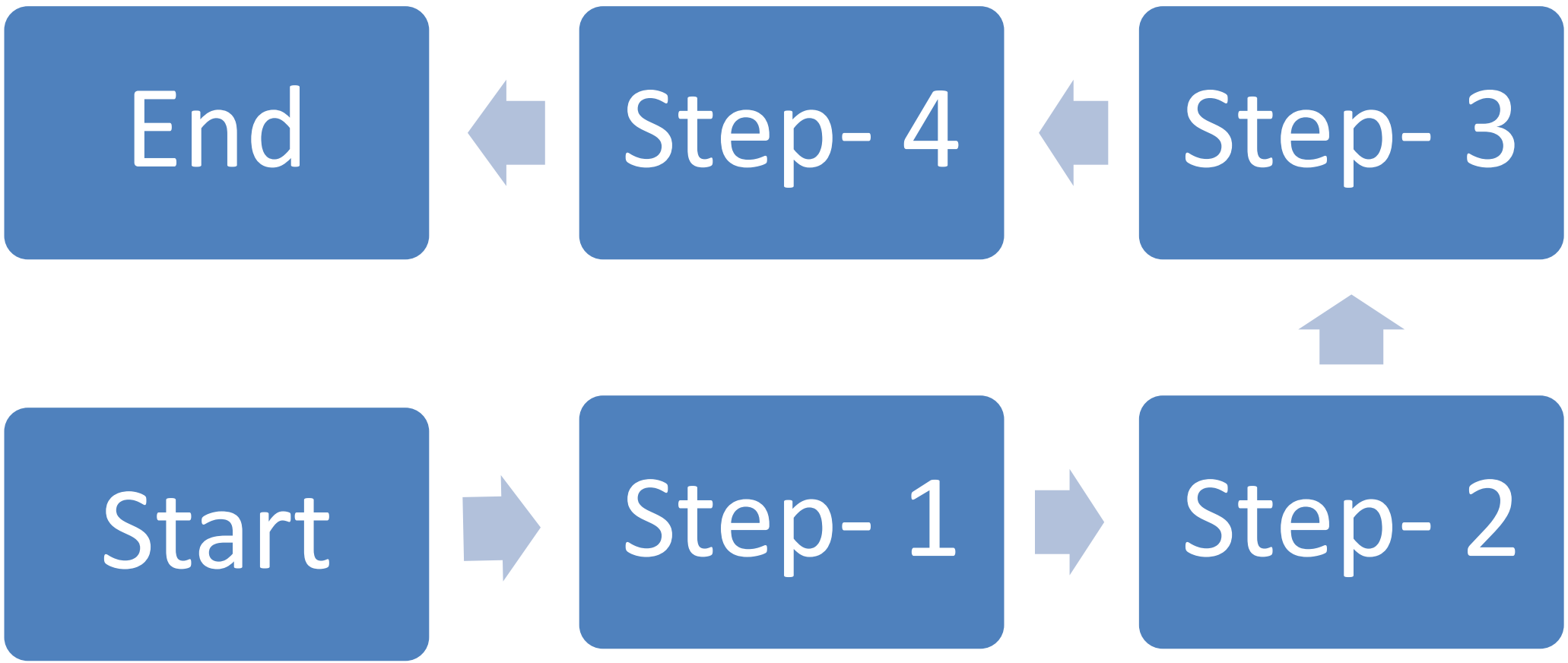
Creating new strategy using API version 2.0

Please visit our website: www.utradesolutions.com
Please follow our product demos: www.youtube.com/user/uTradeSolutions

Introduction

- **History**: Started in **2011** from India
- **Mission**: **Enabling Smarter Trading**
- **Target Clients**: Capital markets participants including brokers, HFT firms, proprietary traders, exchanges, banks and other financial institutions and **Educational Institutions.**
- **Products**: **Trading Platform, Algorithmic trading, and Portfolio Analytics products**
- **Technology**: Based on **Open Source technologies**, developed from scratch, C/C++/Python/HTML5 programming languages, and modular architecture; **6 patents filed in India, and 1 patent filed in US/UK to lead innovation**
- **Team**: **Team size of around 35 employees** with 30+ focused on product development
- **Recognition**: Shortlisted in **London Fintech Innovation Labs** program 2014 and Indian Industry organizations including **CII & Nasscom** as most innovative and top 50 emerging Indian companies for year 2013

Flow Chart- Creating a new strategy using API Version 2



Step- 1: Start

Lets get started

Step- 2: Design Frontend

Design front end using utrade design toolbox Choose and model parameters according to your needs choose label name wisely as same will be used to get values from backend.

Step- 3: Download front end script and embed in backend strategy code

use linux command xxd -i <parameters_frontfile>,
This will convert text into embeddable code

create a function

```
extern "c"  std::string getFrontEndDesign();
```

to return this design

Step- 3: Example File

```
[SYMBOL]
SYMBOL LEG1=UINT64
Order Mode 1=UCHAR
SYMBOL LEG2=UINT64
Order Mode 2=UCHAR
#SYMBOL LEG3=UINT64
#Order Mode 3=UCHAR
#SYMBOL LEG4=UINT64
#Order Mode 4=UCHAR

[STRATEGY_PARAMS]
Total Qty=UINT64:Q
Order Qty=UINT64:Q
Min Profit=FLOAT:P
Hedge Method=COMBO:Market
Order:Best Bid/Ask:Best Ask/Bid
SEPARATOR=1
Timer=TIMER
Spread
Type=RADIO:Product:Actual:Actual
STRETCH=1
Chk Spread=BOOL
STRETCH=1
Bid Diff=BOOL
[OTHER]
```

Step- 3: xxd- i params.txt

```
unsigned char param_txt[] =
{
    0x5b, 0x53, 0x59, 0x4d, 0x42, 0x4f, 0x4c, 0x5d, 0x0a, 0x53, 0x59, 0x4d,  0x42,
    0x4f, 0x4c, 0x20, 0x4c, 0x45, 0x47, 0x31, 0x4e,  0x54, 0x36, 0x34, 0x0a, 0x4f,
    0x72, 0x64, 0x65, 0x72, 0x20, 0x4d, 0x6f,  0x64, 0x65, 0x20, 0x31, 0x3d, 0x55,
    0x43,   0x0a, 0x53,   0x59, 0x4d, 0x42, 0x4f, 0x4c, 0x20, 0x4c, 0x45, 0x47,
    0x32, 0x3d, 0x55,  0x49, 0x4e, 0x54, 0x36, 0x34, 0x0a, 0x73, 0x74, 0x20, 0x42,
    0x69, 0x64, 0x2f, 0x41, 0x73, 0x6b, 0x3a, 0x42,  0x65, 0x73, 0x74, 0x20, 0x41,
    0x73, 0x6b, 0x2f, 0x0a,  0x53, 0x45, 0x50, 0x41, 0x52, 0x41, 0x54, 0x4f, 0x52,
    0x3d, 0x31, 0x0a,  0x54, 0x69, 0x6d, 0x65, 0x72, 0x3d, 0x54 0x52, 0x0a,  0x53,
    0x70, 0x72, 0x65, 0x61, 0x64, 0x20, 0x54, 0x79, 0x70, 0x65, 0x3d,  0x52, 0x41,
    0x44, 0x49, 0x4f, 0x3a, 0x64, 0x75, 0x63,  0x74, 0x3a, 0x41, 0x63, 0x74, 0x75,
    0x61, 0x6c, 0x3a, 0x41, 0x63, 0x74,  0x75, 0x61, 0x6c, 0x0a, 0x530x54, 0x43,
    0x48, 0x3d,  0x31, 0x0a, 0x43, 0x68, 0x6b, 0x20, 0x53, 0x70, 0x72, 0x65, 0x61,
    0x64,   0x3d, 0x42, 0x4f, 0x4f, 0x54, 0x52, 0x45, 0x54, 0x43,   0x48, 0x3d,
    0x31, 0x0a, 0x42, 0x69, 0x64, 0x20, 0x44, 0x69, 0x66, 0x66,  0x3d, 0x42, 0x4f,
    0x0a, 0x5b, 0x4f, 0x54, 0x48, 0x45,  0x52, 0x5d, 0x20, 0x20, 0x0a
};

unsigned int param_txt_len = 425;
```

Step- 3: Corresponding embedded code

```
extern "C"
std::string getFrontEndDesign()
{
   const char param_txt[] =
   {
        0x5b, 0x53, 0x59, 0x4d, 0x42, 0x4f, 0x4c, 0x5d, 0x0a, 0x53, 0x59, 0x4d,
        0x42, 0x4f, 0x4c, 0x20, 0x4c, 0x45, 0x47, 0x31, 0x3d, 0x55, 0x49, 0x4e,
        0x54, 0x36, 0x34, 0x0a, 0x4f, 0x72, 0x64, 0x65, 0x72, 0x20, 0x4d, 0x6f,
        0x64, 0x65, 0x20, 0x31, 0x3d, 0x55, 0x43, 0x48, 0x41, 0x52, 0x0a, 0x53,
        0x59, 0x4d, 0x42, 0x4f, 0x4c, 0x20, 0x4c, 0x45, 0x47, 0x32, 0x3d, 0x55,
        0x49, 0x4e, 0x54, 0x36, 0x34, 0x0a, 0x4f, 0x72, 0x64, 0x65, 0x72, 0x20,
        0x4d, 0x6f, 0x64, 0x65, 0x20, 0x32, 0x3d, 0x55, 0x43, 0x48, 0x41, 0x52,
        0x0a, 0x23, 0x53, 0x59, 0x4d, 0x42, 0x4f, 0x4c, 0x20, 0x4c, 0x45, 0x47,
        0x33, 0x3d, 0x550x0a
   };

   unsigned int param_txt_len = 425;
   return std::string(param_txt,param_txt_len);

}
```

Step- 4: create getDriver function

create a function

```
extern "C"
void * getDriver(void * params)
```

this function will be the entry point for your strategy.
Here you create an object of your strategy class and start your algo

Sample getDriver function

```
extern "C"
void *getDriver(void *params)
{
   API2::StrategyParameters *sgParams = API2::StrategyParameters*)params;
   SampleStrategy context(sgParams);
   return context.startAlgo(true); // Market event required
   //return context.startAlgo(false); // Market event not required
}
```

Step- 5: Creating strategy class

All strategy need to be derived from `API2::SGContext` class

This class has predefined functionality to send order and receive market data & virtual functions which user can override.

Sample Strategy Class

```
class SampleStrategy : public API2::SGContext
{
    public :
    Context(API2::StrategyParameters *params);
    //overridden void onModifyStrategy(API2::AbstractUserParams*);
    private:
    SINGNED_LONG _symbolid;
    //will be set by  front end parameters
    API2::SGCommon::InstrumentOrderId *_instrumentOrderId;
    //used to uniquely identify order
    API2::COMMON::Instrument * _instrument;
    //used in market data subscription , sending orders
    // and p&l calculation
};
```

Step- 5: Things to do in Constructor

-> setup log file using Base class contructor

-> read front end parameters

```
SampleStrategy(API2::StrategyParameters *params)):
API2::SGContext(params, "SampleStrategy") // Sample strategy log file created
{
    API2::UserParams *frontendParams  = (API2::UserParams *)params->getInfo();
    //Parameters received from frontend

    //reading parameters

    if(frontendParams->getValue("SYMBOL LEG1",_symbolid) !=
    API2::UserParamsError_OK)
    {
            std::cout<<"Error in getting SYMBOL LEG1"<<std::endl;
    }
}
```

Step- 5: Subscribing Market order and setting up instrument

```
getAddInstrument(_instrument,
//sending by reference will be set internally_symbolId,
//will be used for creating instrument true,
//need to register for feed or not false);
//feed type false = TBT feed ; true = snapshot feed
```

Receiving mkt event

if startAlgo funtion is with market event enabled then feed event will be received by overriding onDerivedMarketDataEvent

```
void SampleStrategy::onDerivedMarketDataEvent(UNSIGNED_LONG symbolid)
{
     std::cout<<"Symbol id Market Data "
     <<symbolid
     <<std::endl;
     API2::COMMON::MktData *mkData = getUpdateMktData(symbolid);
     mkData->dump();
}
```

Step- 5: Sending Orders

```
API2::SingleOrder *order;
API2::DATA_TYPES::RiskStatus riskStatus = API2::CONSTANTS::RSP_RiskStatus_MAX;
createNewOrder(order, _instrument, <QTY>, <Revealed QTY>,
API2::CONSTANTS::CMD_OrderMode_BUY,
API2::CONSTANTS::CMD_OrderType_LIMIT,
API2::CONSTANTS::CMD_OrderValidity_DAY,
API2::CONSTANTS::CMD_ProductType_DELIVERY,
<PRICE>);_instrumentOrderId = NULL:
if( ! sendNewOrder(riskStatus,_instrument,_order, _instrumentOrderId))
//_instrumentOrderId is used to uniquely identify an order
{
     //error in sending order
     //riskstatus variable will have error code store in it , describing the
problem.
}
else
{
     //order sent out successfully
}
```

Step- 5: Receiving Confirmations

SGContext class has many virtual functions such as

- onNewConfirmed
- onCanceledonReplaced
- onReplaceRejected
- onCancelRejectedonNewReject
- onFilled
- onPartialFill

they need to be overridden to capture confirmations such as

```
void SampleStrategy::onNewConfirmed(API2::OrderConfirmation  &confirmation,
API2::COMMON::InstrumentOrderId *orderId)
{
    if(orderId == _instrumentOrderId)
    {
        //confirmation received from exchange
        //do something according ti your logic
    }
}
```

Step- 6: End

Strategy is created

Disclaimer

This presentation is intended for sharing the business idea, product presentation, and exploring a partnership opportunity. No content and the ideas presented for the new venture hereby maybe re-used, re-distributed or discussed outside of the organization where it is presented (without the prior content of the author of this presentation). None of the ideas for financial trading technology presented in these slides maybe copied or shared with the operations of existing or upcoming companies operating in this market segment. The information shall not be distributed or used by any person or entity in any jurisdiction or countries were such distribution or use would be contrary to the applicable laws or Regulations. It is advised that prior to acting upon this presentation, independent consultation / advise may be obtained and necessary due diligence, investigation etc. may be done at your end. You may also contact us directly for any questions or clarifications. All statements regarding the future are subject to inherent risks and uncertainties, and many factors may lead to actual profits or losses & developments deviating substantially from what has been expressed or implied in such statements.

Reach us at:
marketing@utradesolutions.com

uTrade
Enabling Smarter Trading

uTrade Solutions Private Ltd
2nd floor, landmark plaza (f3). Quarkcity SEZ, A40A,
Industrial Area, Phase 8 extension,
Mohali, Chandigarh 160071 (India)
contact: info@utradesolutions.com