ObjXp

ObexTree Tutorial

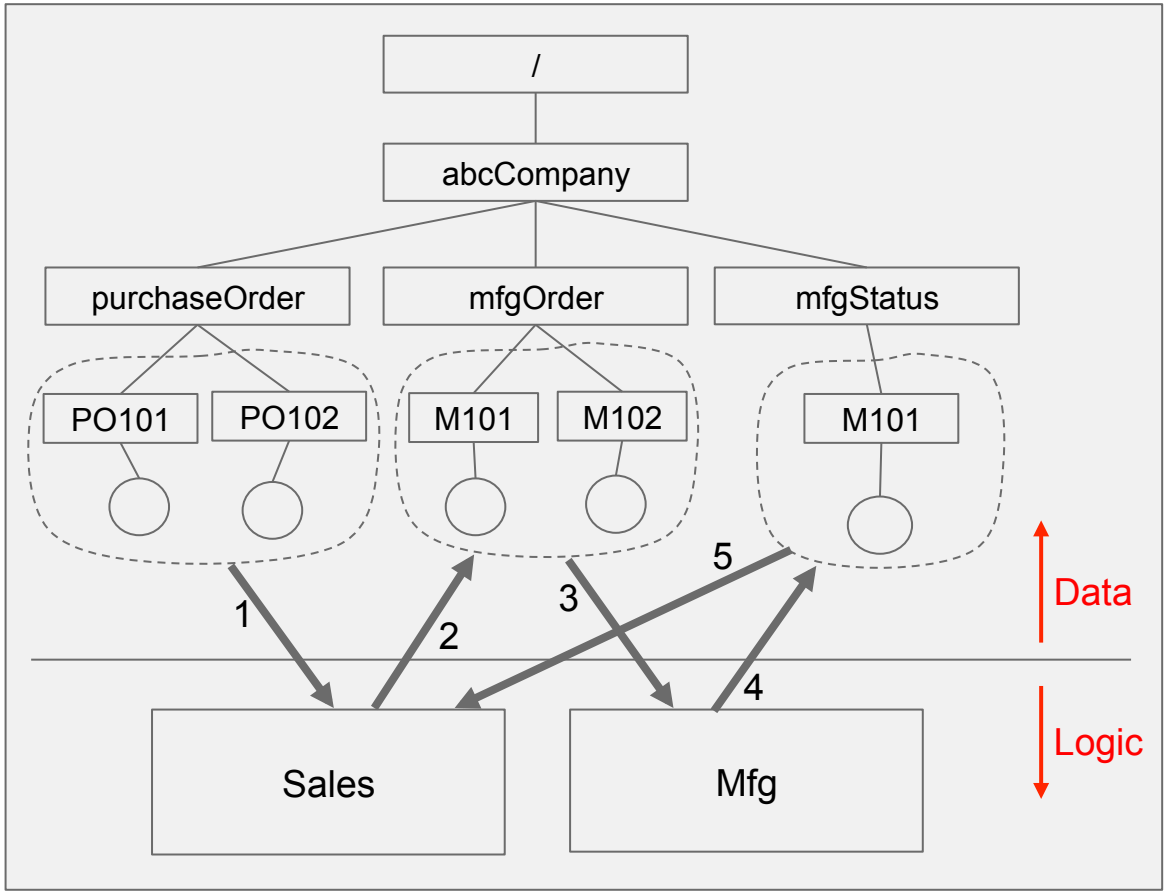
Goal

- Learn how to create ObexServer and ObexClient
- Learn how to make various connections between Obex servers and clients
- Learn how to define a custom ObexObject
- Learn how to use Yail TimerObject
- Understand the design philosophy to isolate data from business logic

How to build

```
$ BUILD_TUTORIAL=yes cmake .
$ make [ -j NUM_JOBS ] all
```

Assignment 1



ABC Company's Work Flow

1. Customer places a purchase order
2. Sales team receives the PO and converts to manufacturing order.
3. Manufacturing team receives the mfgOrder.
4. Manufacturing team updates the status following the manufacturing process like "Scheduled -> Started -> Completed". When completed, clears the corresponding mfgOrder.
5. Sales team again checks the mfgStatus and delivers the product to the customer. Then removes the item from the mfgStatus and also clears the original purchaseOrder.

Object Path Definitions

```
// Purchase order path
#define YpAbcPurchaseOrder                "/abcCompany/purchaseOrder"
#define YpAbcPurchaseOrder__PO_ID         "/abcCompany/purchaseOrder/${PO_ID}"

// Manufacturing order path
#define YpAbcMfgOrder                     "/abcCompany/mfgOrder"
#define YpAbcMfgOrder__MO_ID              "/abcCompany/mfgOrder/${MO_ID}"

// Manufacturing status path
#define YpAbcMfgStatus                    "/abcCompany/mfgStatus"
#define YpAbcMfgStatus__MO_ID             "/abcCompany/mfgStatus/${MO_ID}"
```

MfgOrder class

```
YAIL_BEGIN_CLASS(MfgOrder, EXTENDS(ObexObject))
  public:
    YAIL_OBEX_INSTANTIATOR(MfgOrder); // Required for Yail to create the instance of this class by class name

    void init(String moId, String item, int quantity) { moId_ = moId;  item_ = item;  quantity_ = quantity; }
    bool equals(ObexObject& oObj) override {  /* Criteria to judge this object is same as the given one */  }
    String toString() override {  /* return friendly string to summarize the object */  }
    String marshal() override {
      ……
      // Serialize the object
      ……
    }
    void unmarshal(String& data) override {
      ……
      // Deserialize the object
      ……
    }
    String getMoId() { return moId_; }

  Private:
    // Data members
    String moId_;
    String item_;
    int quantity_;
YAIL_END_CLASS
```

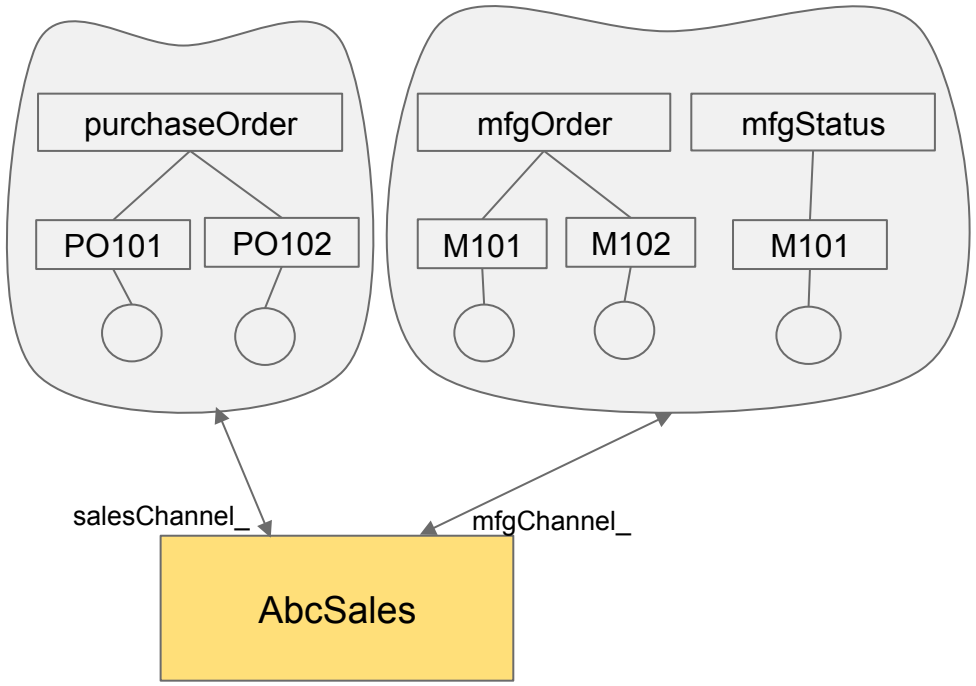
INIT_OBEX_INSTANTIATOR(MfgOrder); // Required for Yail to register class constructor by its name

Define AbcSales class

```
YAIL_BEGIN_CLASS(AbcSales, EXTENDS(YObject), IMPLEMENTS(ObexCallback))
  public:
    // Initializer
    void init(SPtr<ObexClientSession> salesChannel,
              SPtr<ObexClientSession> mfgChannel);
    ……

    // Obex callback handlers
    void onUpdated(String cbSrc, String path,
       SPtr<ObexObject> newObj,
       SPtr<ObexObject> oldObj) override;
    void onDeleted(String cbSrc, String path,
       SPtr<ObexObject> oldObj) override;
    ……

  private:
    // Client session to handle purchase orders
    SPtr<ObexClientSession> salesChannel_;
    // Client session to handle manufacturing orders
    SPtr<ObexClientSession> mfgChannel_;
    // MfgOrder to PurchaseOrder mapping
    Map<String, String> moPoMap_;
YAIL_END_CLASS
```



AbcSales Implementation

```
void AbcSales::init(SPtr<ObexClientSession> salesChannel, SPtr<ObexClientSession> mfgChannel) {
  // Keep pointers to salesChannel and mfgChannel
  salesChannel_ = salesChannel;
  mfgChannel_ = mfgChannel;

  // Subscribe all descendants of /abcCompany/purchaseOrder
  salesChannel->subscribe(YpAbcPurchaseOrder "/*");
  salesChannel->registerCallback(YpAbcPurchaseOrder "/", getThisPtr<ObexCallback>());

  // Subscribe all descendants of /abcCompany/mfgOrder
  mfgChannel->subscribe(YpAbcMfgStatus "/*");
  mfgChannel->registerCallback(YpAbcMfgStatus "/", getThisPtr<ObexCallback>());
}
```

```
void AbcSales::onUpdated(String cbSrc, String path, SPtr<ObexObject> newObj, SPtr<ObexObject> oldObj) {
   Vector<String> tokens;
   ParseYPath(tokens, path);
   if(tokens[0] != "abcCompany") return; // ignore if not an object of abcCompany

   if(tokens[1] == "purchaseOrder") { // Handle new purchase order
      // Expect newObj is an ObexStringObject of which the string format is like “CustomerName,ItemName,Quantity”
      SPtr<ObexStringObject> poDetail = DynamicPointerCast<ObexStringObject>(newObj);
      …...
      // Parse customer, item, and quantity from the poDetail
      // Generate a unique manufacturing order id(moId) and create manufacturing order object based on the information parsed above
      SPtr<MfgOrder> mfgOrder = CreateObject<MfgOrder>(moId, item, quantity);
      moPoMap_.insert(Pair<String, String>(moId, poId)); // Remember moId to poId mapping to clear the completed order later
      …...
      // Place mfg order on mfg channel
      mfgChannel_->putObject(moPath, mfgOrder);

   } else if(tokens[1] == "mfgStatus") { // Handle mfg order status change
      String moId = tokens[2];
      // Expect newObj is mfgStatus and its type is ObexStringObject
      SPtr<ObexStringObject> mfgStatus = DynamicPointerCast<ObexStringObject>(newObj);

      if(mfgStatus->toString() == "Completed") { // Handle mfgStatus “Completed”
        String moId = tokens[2];
        // String poId = get poId from moPoMap_
        moPoMap_.erase(moId); // Remove moId-to-poId mapping from moPoMap_
        …...
        String mfgStatusPath = MkYPath(YpAbcMfgStatus__MO_ID, MkYPathArg("MO_ID", moId));
        mfgChannel_->delObject(mfgStatusPath); // Clear manufacturing order
        …...
        String poPath = MkYPath(YpAbcPurchaseOrder__PO_ID, MkYPathArg("PO_ID", poId));
        salesChannel_->delObject(poPath); // Clear purchase order
      }
   } else {
      // ignore non-interested object update
   }
}
```

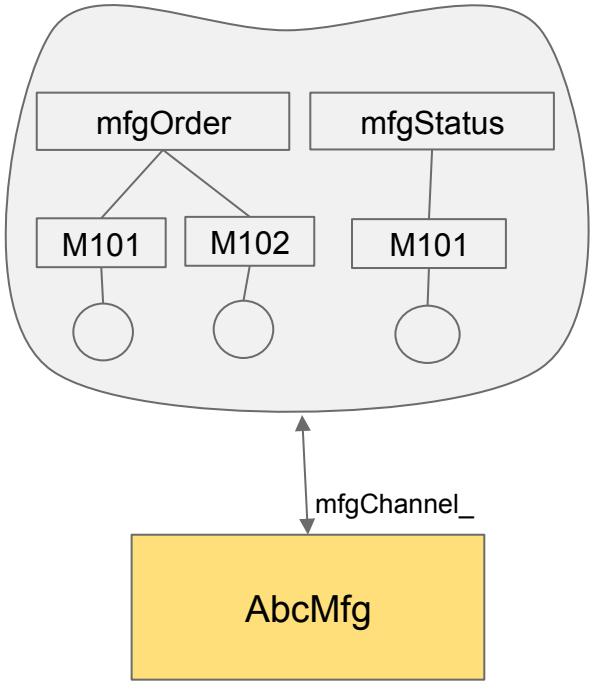
Define AbcMfg class

```
YAIL_BEGIN_CLASS(AbcMfg, EXTENDS(YObject),
                         IMPLEMENTS(ObexCallback),
                         IMPLEMENTS(TimerCallback))
  public:
    // Initializer
    void init(SPtr<ObexClientSession> mfgChannel, SPtr<IoService> ioSvc);
    ……
    // Obex callback handlers
    void onUpdated(String cbSrc, String path,
        SPtr<ObexObject> newObj, SPtr<ObexObject> oldObj) override;
    void onDeleted(String cbSrc, String path,
        SPtr<ObexObject> oldObj) override;

    // Timer callback handler
    void onTimerExpired(SPtr<TimerObject> tObj) override;

  private:
    // Client session to handle mfg order
    SPtr<ObexClientSession> mfgChannel_;

    // Timer holder.
    // We assume it takes some time to manufacture the items.
    // Simulate the situation as time delay
    Map<String, SPtr<TimerObject>> timers_;
    ……
YAIL_END_CLASS
```



AbcMfg Implementation

```
void AbcSales::init(SPtr<ObexClientSession> salesChannel, SPtr<ObexClientSession> mfgChannel) {
  // Keep pointer to mfgChannel
  mfgChannel_ = mfgChannel;
  ……

  // Subscribe all descendants of /abcCompany/mfgOrder
  mfgChannel->subscribe(YpAbcMfgOrder "/*");
  mfgChannel->registerCallback(YpAbcMfgOrder "/", getThisPtr<ObexCallback>());
}
```

```
void AbcMfg::onUpdated(String cbSrc, String path, SPtr<ObexObject> newObj, SPtr<ObexObject> oldObj) {
  Vector<String> tokens; ParseYPath(tokens, path);
  if(tokens[1] == "mfgOrder") {
    SPtr<MfgOrder> mfgOrder = DynamicPointerCast<MfgOrder>(newObj);
    String moId = mfgOrder->getMoId();
    String mfgStatusPath = MkYPath(YpAbcMfgStatus__MO_ID, MkYPathArg("MO_ID", moId));

    // Update MfgStatus to "Started"
    SPtr<ObexStringObject> mfgStatusObj = CreateObject<ObexStringObject>("Started");
    mfgChannel_->putObject(mfgStatusPath, mfgStatusObj);

    // Assume it takes some time to manufacture the items. Simulate the situation as time delay
    SPtr<TimerObject> tObj = CreateObject<TimerObject>(moId, getThisPtr<TimerCallback>());
    timers_.insert(Pair<String, SPtr<TimerObject>>(moId, tObj));
    ioSvc_->registerTimer(tObj, 3000);
  } else {
    // Ignore non-interested object update
  }
}

void AbcMfg::onTimerExpired(SPtr<TimerObject> tObj) {
  String moId = tObj->getId();
  timers_.erase(moId); // Remove the timer

  String mfgStatusPath = MkYPath(YpAbcMfgStatus__MO_ID, MkYPathArg("MO_ID", moId));

  // Update MfgStatus to "Completed"
  SPtr<ObexStringObject> mfgStatusObj = CreateObject<ObexStringObject>("Completed");
  mfgChannel_->putObject(mfgStatusPath, mfgStatusObj);

  // Clear MfgOrder
  String mfgOrderPath = MkYPath(YpAbcMfgOrder__MO_ID, MkYPathArg("MO_ID", moId));
  mfgChannel_->delObject(mfgOrderPath);
}
```

Put them all together in one process

```
YAIL_BEGIN_CLASS(AbcCompany, EXTENDS(YObject))
  public:
    void init(SPtr<IoService> ioSvc) {
      obexServer_ = CreateObject<ObexServer>("AbcCompany", "AbcCompany", 0, ioSvc); // Create ObexServer
      obexClient_ = CreateObject<ObexClient>("AbcClient", ioSvc, nullPtr(ConnectionCallback)); // Create ObexClient

      // Create a cohabiting client session connecting ObexServer and ObexClient that reside in the same process space
      obexClient_->connect("abcCompany", obexServer_);
      cohabSession_ = obexClient_->getSession("abcCompany");

      // Pass the cohabSession as the mfgChannel for the AbcMfg
      mfg_ = CreateObject<AbcMfg>(cohabSession_, ioSvc);
      // Pass the cohabSession as the salesChannel and mfgChannel for the AbcSales
      sales_ = CreateObject<AbcSales>(cohabSession_, cohabSession_);
    }

  private:
    SPtr<ObexServer> obexServer_;
    SPtr<ObexClient> obexClient_;
    SPtr<ObexClientSession> cohabSession_;
    SPtr<AbcSales> sales_;
    SPtr<AbcMfg> mfg_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SetTraceLevel(8);
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcCompany> abcCompany = CreateObject<AbcCompany>(ioSvc);
  ioSvc->run();
  return 0;
}
```

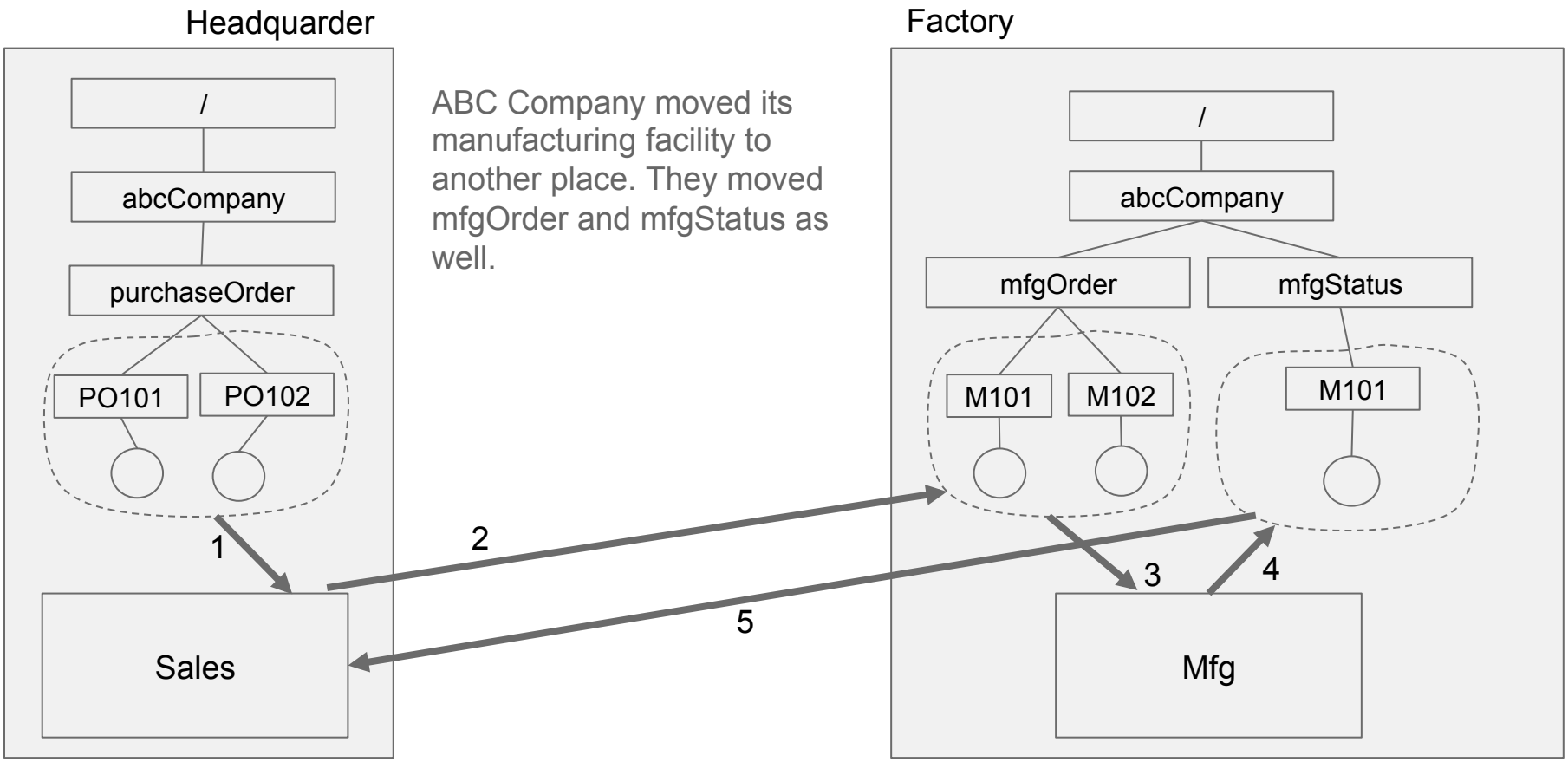
Run Assignment 1

```
yail$ ./tutorial/Obex/Assignments/Assignment1/AbcCompany
[AbcMfg::init():16] Ready
[AbcSales::init():19] Ready
[AbcSales::onUpdated():45] Purchase order received: PO[P1010, CustomerA, ProductX, 1000]
[AbcSales::onUpdated():57] Placing a manufacturing order M101
[AbcMfg::onUpdated():26] Manufacturing order received: MO[M101, ProductX, 1000]
[AbcSales::onUpdated():65] MO[M101] Status changed to Started
[AbcMfg::onUpdated():42] MO[M101] Production in progress ...

.....
```

```
# Inject a sample PO(poId:P1010, customer:CustomerA, item:ProductX, quantity:1000) to AbcCompany socket
yail$ ./tutorial/Obex/Assignments/Utils/putStringObject.py -s AbcCompany \
        /abcCompany/purchaseOrder/P1010 CustomerA,ProductX,1000
```

Assignment 2



What to do?

- Reuse MfgOrder without any change
- AbcSales and AbcMfg are data location independent ⇒ **No change required**
- Need only to modify the way to create ObexServers/ObexClients and the connections between them in separate processes.

Implement AbcHQ

```
YAIL_BEGIN_CLASS(AbcHQ, EXTENDS(YObject), IMPLEMENTS(ConnectionCallback))
  public:
    void init(SPtr<IoService> ioSvc) {
      obexServer_ = CreateObject<ObexServer>("AbcHQ", "AbcHQ", 0, ioSvc); // Create ObexServer to keep purchase orders
      obexClient_ = CreateObject<ObexClient>("AbcHQ", ioSvc, getThisPtr<ConnectionCallback>()); // Create ObexClient
      obexClient_->connect("salesChannel", obexServer_); // Connection for sales channel
      salesChannel_ = obexClient_->getSession("salesChannel");
      obexClient_->connect("mfgChannel", "AbcFactory", 1000, 0); // Connection for mfg channel
    }
    void onConnected(String sessionName, int fd) { // mfg channel connected
      mfgChannel_ = obexClient_->getSession("mfgChannel");
      sales_ = CreateObject<AbcSales>(salesChannel_, mfgChannel_); // Create AbcSales instance
    }
    void onConnectionFailed(String sessionName) { /* never enter this routine */ }
    void onDisconnected(String sessionName, int fd) { exit(0); }

  private:
    SPtr<ObexServer> obexServer_;
    SPtr<ObexClient> obexClient_;
    SPtr<ObexClientSession> salesChannel_;
    SPtr<ObexClientSession> mfgChannel_;
    SPtr<AbcSales> sales_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcHQ> abcHQ = CreateObject<AbcHQ>(ioSvc);
  ioSvc->run(); // Run AbcHQ
  return 0;
}
```

Implement AbcFactory

```
YAIL_BEGIN_CLASS(AbcFactory, EXTENDS(YObject))
  public:
    void init(SPtr<IoService> ioSvc) {
      obexServer_ = CreateObject<ObexServer>("AbcFactory", "AbcFactory", 0, ioSvc); // Create ObexServer to keep mfg data
      obexClient_ = CreateObject<ObexClient>("AbcFactory", ioSvc, nullPtr(ConnectionCallback));
      obexClient_->connect("mfgChannel", obexServer_); // Connection for mfg channel. It's a cohabiting client in this case.
      mfgChannel_ = obexClient_->getSession("mfgChannel");
      mfg_ = CreateObject<AbcMfg>(mfgChannel_, ioSvc);
    }

  private:
    SPtr<ObexServer> obexServer_;
    SPtr<ObexClient> obexClient_;
    SPtr<ObexClientSession> mfgChannel_;

    SPtr<AbcMfg> mfg_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcFactory> abcFactory = CreateObject<AbcFactory>(ioSvc);
  ioSvc->run(); // Run AbcFactory
  return 0;
}
```

Run Assignment 2

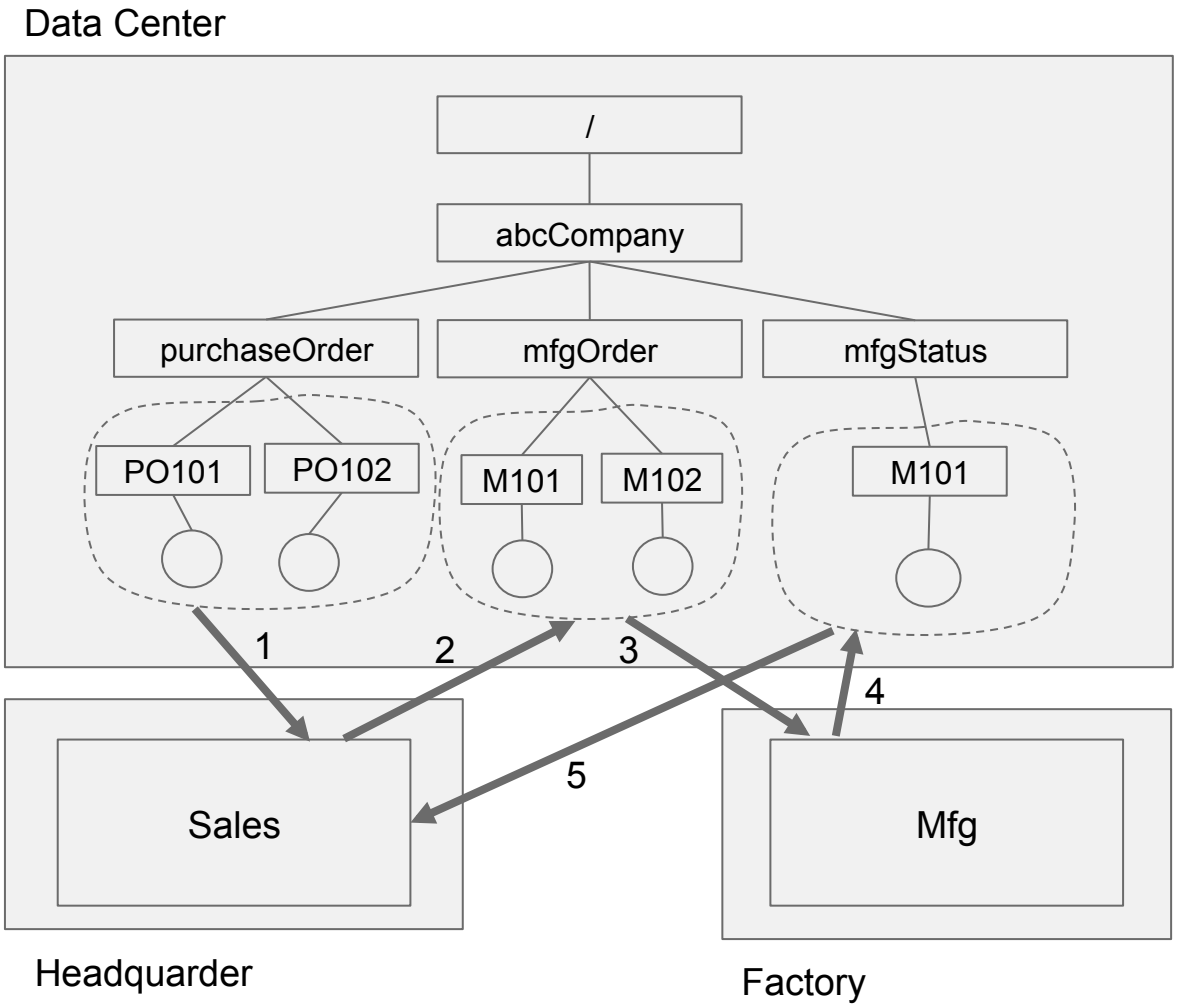
```
yail$ ./tutorial/Assignments/Assignment2/AbcFactory
[AbcMfg::init():16] Ready
[AbcMfg::onUpdated():26] Manufacturing order received: MO[M101, ProductX, 1000]
[AbcMfg::onUpdated():42] MO[M101] Production in progress ...
......
```

```
yail$ ./tutorial/Assignments/Assignment2/AbcHQ
[AbcHQ::onConnected():25] mfgChannel connected
[AbcSales::init():19] Ready
[AbcSales::onUpdated():45] Purchase order received: PO[P1010, CustomerA, ProductX, 1000]
[AbcSales::onUpdated():57] Placing a manufacturing order M101
......
```

```
yail$ ./tutorial/Obex/Assignments/Utils/putStringObject.py -s AbcHQ \
        /abcCompany/purchaseOrder/P1010 CustomerA,ProductX,1000
```

Assignment 3

ABC Company decided to move all data to central data center.



What to do?

- Again, **no change required** for MfgOrder, AbcSales and AbcMfg
- Need only to modify the way to create ObexServers/ObexClients and the connections between them in separate processes.

Implement AbcDC

```
YAIL_BEGIN_CLASS(AbcDC, EXTENDS(YObject))\
  public:
    void init(SPtr<IoService> ioSvc) {
      obexServer_ = CreateObject<ObexServer>("AbcDC", "AbcDC", 5555, ioSvc);
    }

  private:
    SPtr<ObexServer> obexServer_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcDC> abcDC = CreateObject<AbcDC>(ioSvc);
  ioSvc->run();
  return 0;
}
```

Implement AbcHQ

```
YAIL_BEGIN_CLASS(AbcHQ, EXTENDS(YObject), IMPLEMENTS(ConnectionCallback))
  public:
    void init(SPtr<IoService> ioSvc) {
      obexClient_ = CreateObject<ObexClient>("AbcHQ", ioSvc, getThisPtr<ConnectionCallback>());
      obexClient_->connect("abcDC", "AbcDC", 1000, 0); // Create connection to data center via unix socket
    }

    void onConnected(String sessionName, int fd) { // Connection to data center established
      salesChannel_ = obexClient_->getSession("abcDC");
      mfgChannel_ = salesChannel_; // salesChannel and mfgChannel are same
      sales_ = CreateObject<AbcSales>(salesChannel_, mfgChannel_); // Create AbcSales
    }
    void onConnectionFailed(String sessionName) { /* never enter this routine */ }
    void onDisconnected(String sessionName, int fd) { exit(0); }

  private:
    SPtr<ObexClient> obexClient_;
    SPtr<ObexClientSession> salesChannel_;
    SPtr<ObexClientSession> mfgChannel_;
    SPtr<AbcSales> sales_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcHQ> abcHQ = CreateObject<AbcHQ>(ioSvc);
  ioSvc->run(); // Run AbcHQ
  return 0;
}
```

Implement AbcFactory

```
YAIL_BEGIN_CLASS(AbcFactory, EXTENDS(YObject), IMPLEMENTS(ConnectionCallback))
  public:
    void init(SPtr<IoService> ioSvc) {
      ioSvc_ = ioSvc;
      obexClient_ = CreateObject<ObexClient>("AbcFactory", ioSvc, getThisPtr<ConnectionCallback>());
      obexClient_->connect("abcDC", "127.0.0.1:5555", 1000, 0); // Create connection to data center via TCP socket
    }
    void onConnected(String sessionName, int fd) { // Connection to data center established
      mfgChannel_ = obexClient_->getSession("abcDC");
      mfg_ = CreateObject<AbcMfg>(mfgChannel_, ioSvc_); // Create AbcMfg
    }
    void onConnectionFailed(String sessionName) { /* never enter this routine */ }
    void onDisconnected(String sessionName, int fd) { exit(0); }

  private:
    SPtr<IoService> ioSvc_;
    SPtr<ObexClient> obexClient_;
    SPtr<ObexClientSession> mfgChannel_;
    SPtr<AbcMfg> mfg_;
YAIL_END_CLASS

int main(int argc, char** argv) {
  SPtr<BasicIoService> ioSvc = CreateObject<BasicIoService>();
  SPtr<AbcFactory> abcFactory = CreateObject<AbcFactory>(ioSvc);
  ioSvc->run(); // Run AbcFactory
  return 0;
}
```

Run Assignment 3

```
yail$ ./tutorial/Assignments/Assignment3/AbcDC
......
```

```
yail$ ./tutorial/Assignments/Assignment3/AbcFactory
[AbcFactory::onConnected():20] abcDC connected
[AbcMfg::init():16] Ready
[AbcMfg::onUpdated():26] Manufacturing order received: MO[M101, ProductX, 1000]
......
```

```
yail$ ./tutorial/Assignments/Assignment3/AbcHQ
[AbcHQ::onConnected():20] abcDC connected
[AbcSales::init():19] Ready
[AbcSales::onUpdated():45] Purchase order received: PO[P1010, CustomerA, ProductX, 1000]
......
```

```
yail$ tutorial/Obex/Assignments/Utils/putStringObject.py -s AbcDC \
        /abcCompany/purchaseOrder/P1010 CustomerA,ProductX,1000
```