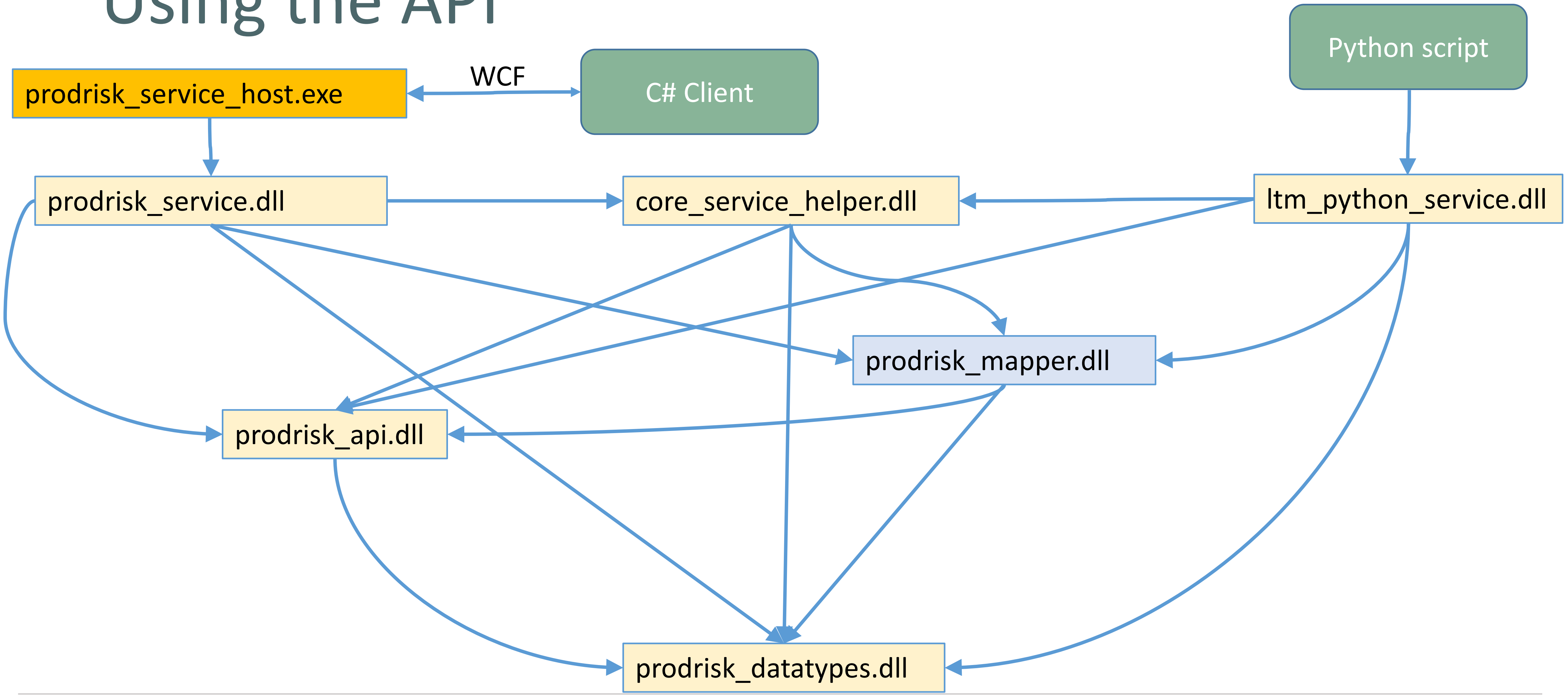
Using the API



Top level dlls

- prodrisk_service.dll
- Interface (ServiceContract):
  - ProdriskCore.IProdriskService
- Class (implementing the interface above):
  - ProdriskCore.ProdriskService
- Each method is an OperationContract

- ltm_python_service.dll
- Interfaces (ServiceContract):
  - ProdriskPythonCoreService.IProdriskPythonService
  - VansimtapPythonCoreService. IVansimtapPythonService
- Classes (implementing each interface above):
  - ProdriskPythonCoreService.ProdriskPythonService
  - VansimtapPythonCoreService.VansimtapPythonService
- Each method is an OperationContract

Calling Prodrisk from Python

```
import clr          # Common Language Runtime,  the virtual machine component of Microsoft's .NET framework
clr.AddReference(currentDir + "\\ltm_python_service")              # Load the dll
from ProdriskPythonCoreService import ProdriskPythonService as prodriskAPI     # ProdriskPythonCoreService (Namespace from the dll above), ProdriskPythonService (class)

prodriskCore = prodriskAPI("Test", False)           # Constructor, also instantiates a reference to the core (C++)
prodriskCore.SetOptimizationPeriod(startTime, endTime)           # Defines the time unit as hour and transfer the values to the core object created above

prodriskCore. SetIntValue("setting", "setting", "minIterations", 1) # Gets the index of the object and the attribute, and transfers the value via the core object
prodriskCore. SetIntValue("setting", "setting", "maxIterations", 5)

for modName, vol in initialState.startVol.items(): )                                          # Perform iterations on objects
   prodriskCore.SetDoubleValue("module",modName,"startVol",vol

objectNames = prodriskCore.GetObjectNamesInSystem()           # Retrieve all object names defined

prodriskCore.GenerateProdriskFiles()                # Adds hardcoded values, perform some sorting and other preparations before writing the ASCII and binary files
prodriskCore.RunProdrisk()        # Creates necessary commands, environment variables etc and then first executes Genpris and then Prodrisk, before retrieving the results
```

Starting a server host

```
using (var host = new ServiceHost(typeof(ProdriskService)))
{
    host.Open();
    Console.ReadLine();
    host.Close();
}
```

Some server host settings

```
<system.serviceModel>
<services>
   <service name="ProdriskCore.ProdriskService" behaviorConfiguration="MyBehavior">
    <endpoint address="" binding="netTcpBinding" bindingConfiguration="ProdriskBinding" contract="ProdriskCore.IProdriskService">
     <identity>
      <dns value="localhost"/>
     </identity>
    </endpoint>
    <endpoint address="mex" binding="mexTcpBinding" bindingConfiguration="" contract="IMetadataExchange"/>
    <host>
     <baseAddresses>
      <add baseAddress="net.tcp://localhost:9999/ProdriskServiceHost/"/>
     </baseAddresses>
    </host>
   </service>
  </services>
</system.serviceModel>
```

Calling Prodrisk from a client (C#)

```
prodriskCore = new ProdriskServiceClient();        # Reference to tool generated code
fileData = HelperFunctions.ReadAllSerializedData(filePaths.ToList());                   # In some way, the information must be read into the client
```

The following requires that the server host is running

```
prodriskCore.SetOptimizationPeriod(fileData.simulationInterval.start, fileData.simulationInterval.end);
        base.Channel.SetOptimizationPeriod(startTime, endTime);     # From the tool generated code
        Interface : [System.ServiceModel.OperationContractAttribute(Action="http://tempuri.org/IProdriskService/SetOptimizationPeriod")] # Temporary Uniform Resource Identifier
        Server side:
        public void SetOptimizationPeriod(DateTime startTime, DateTime endTime)
         {
           core.SetTimeResolution(start, end, timeUnit);
        }

int n = fileData.objectList.Count;
for (int i = 0; i < n; ++i)
{
  prodriskObject obj = fileData.objectList[i];        # Iterate on all elements
  prodriskCore.AddObject(obj.objectType, obj.objectName);      # Add the element to the prodrisk core
}
prodriskCore.Optimise();          # Perform the actual optimization
```

HydroCen