Efficient Programming of Large Models

Aneesha Panicker (adpanick@us.ibm.com)
Guang Feng (gfeng@us.ibm.com)
Technical Support Engineers, Level 2
10 December 2013

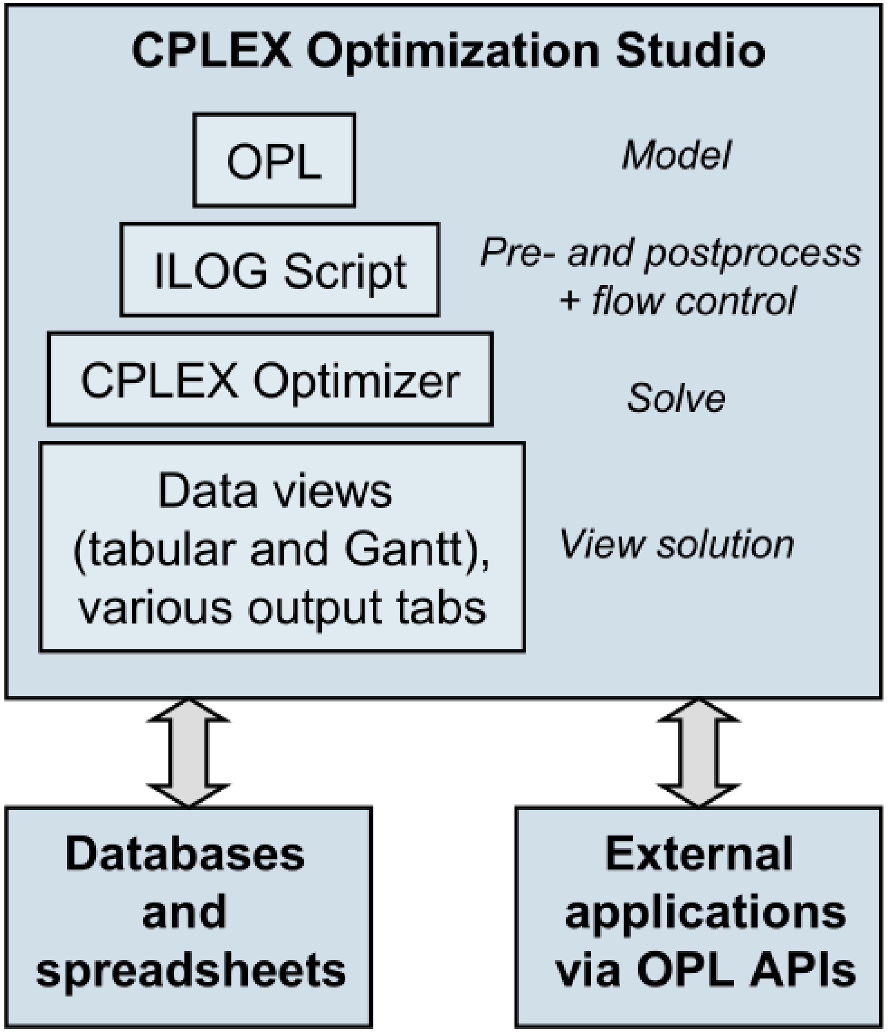
Agenda

- Introduction to Concert Modeling
- Tips on Memory Management
- Perform Similar Operations in Batch
- Suppress Excessive Notifications by Deleting CPLEX Object
- LP Matrix Representation is More Efficient than Constraint Arrays
- Different Layers of CPLEX API
- Conclusion

Introduction to Concert Modeling

IBM CPLEX Optimization Studio

- An Integrated Development Environment (IDE) for model building, debugging and tuning
- Optimization Programming Language (OPL) for modeling
- IBM ILOG Script for pre- and postprocessing, and flow control
- CPLEX Optimizer solution engine with:
  - Several MP optimizers
  - CP Optimizers for CP problems
- Database and spreadsheet connectivity
- OPL APIs for integration with external applications



CPLEX with Concert Technology

- Main features
  - Intuitive, expressive high-level modeling of problems
  - Faster development of mathematical programming applications
  - Works seamlessly with CP Optimizer
- C++ version
  - Access to all frequently used features from C++
  - Works seamlessly with IBM ILOG Solver
- JAVA™ version
  - Access to all frequently used features from JAVA
  - JAVA Native Interface (JNI) based: core unchanged - no C/C++ compiler or knowledge necessary
- C# version
  - Access to all frequently used features from C#
  - Full-featured .NET framework

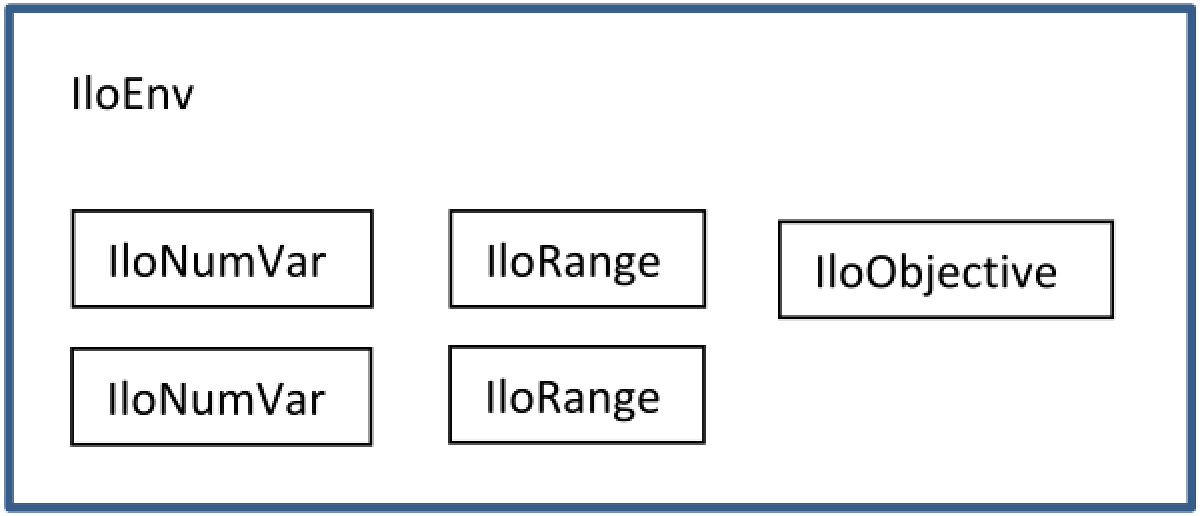
A CPLEX example using Concert API

```
IloModel model(env);

IloNumVarArray x (env);
IloRangeArray con(env);

x.add(IloNumVar(env, 0.0, 40.0));
x.add(IloNumVar(env));
x.add(IloNumVar(env));
x.add(IloNumVar(env, 2.0, 3.0, ILOINT));
model.add(IloMaximize(env, x[0] + 2 * x[1] + 3 * x[2] + x[3]));

con.add( - x[0] +     x[1] + x[2] + 10 * x[3] <= 20);
con.add(   x[0] - 3 * x[1] + x[2]             <= 30);
con.add(              x[1]        - 3.5* x[3] == 0);
model.add(con);

IloCplex cplex(model);
cplex.solve();

env.out() << "Solution status = " << cplex.getStatus() << endl;
env.out() << "Solution value  = " << cplex.getObjValue() << endl;

IloNumArray vals(env);
cplex.getValues(vals, x);

env.end();
```

Tips on Memory Management

Tips on Memory Management

- Main features of Concert
  - All CPLEX and Concert objects need to be associated to an environment object (`IloEnv` in C++ API, `IloCplex` in JAVA API and `Cplex` in .NET API).



Tips on Memory Management

- Main features of Concert
  - Most modeling objects are implemented as two classes: one handle class and one implementation class

Handle class

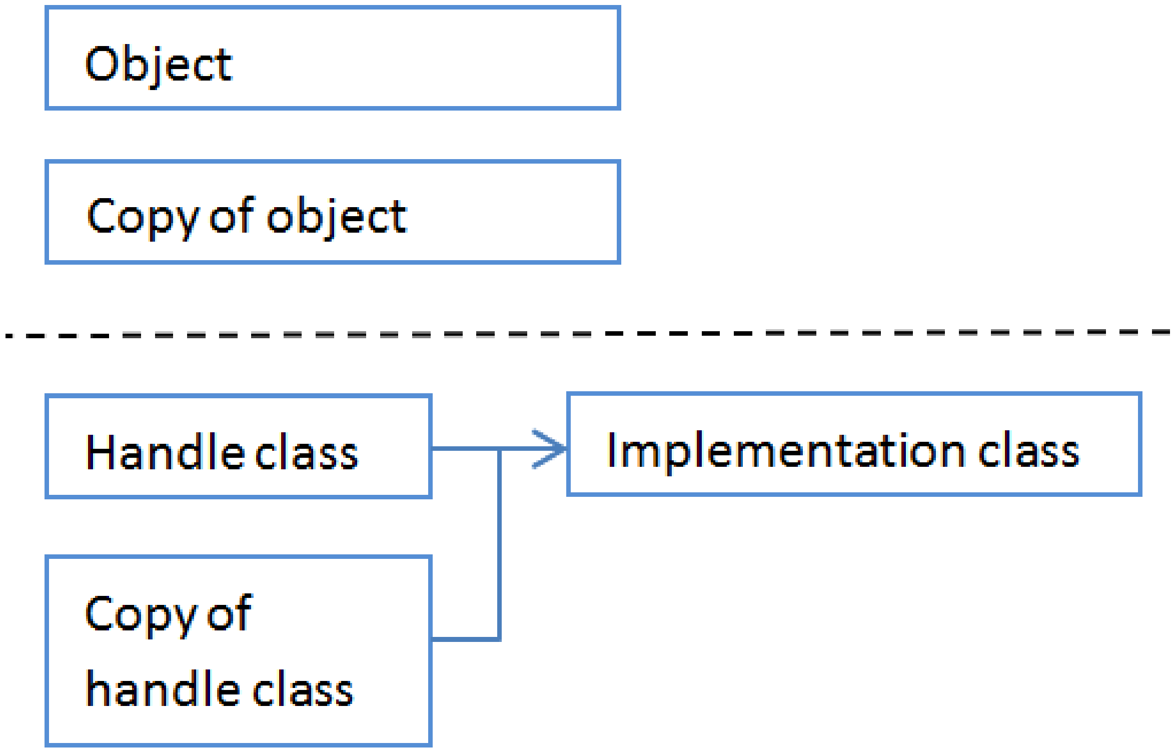
```
class Alpha {
protected:
    AlphaI* _impl;
public:
    void do();
};
```

Implementation class

```
class AlphaI {
protected:
    IlcInt x;
public:
    void do();
};
```

Tips on Memory Management

- Main features of Concert
  - Why use two classes: handle class and implementation class?



Tips on Memory Management

- Some objects do not have implementation classes, such as `IloExpr`, `IloNumColumn` `IloIntSet`
  - When using these objects to create new modeling objects, the application firstly creates a copy of the original object and then use the new copy for modeling object creation.
  - Modifications to the original objects are not applied to the new copy.
    - When you have multiple handles to such objects, some handles might be pointing to copies of the original objects.
    - We recommend keeping one single and unique handle to each Concert modeling object to avoid confusion.
  - After new modeling objects are created, the original objects can and should be deleted to save memory.

Tips on Memory Management

- Ending the environment object can effectively free all memory consumed by CPLEX and Concert objects added to the environment.
  - Useful for debugging
- In C++ API, each Concert modeling object has to be ended by invoking the end() method.
- In JAVA and .NET APIs, freeing the CPLEX object will free memory used by in unmanaged heaps as well as managed heaps.

Concert Overloaded Operators Creates Objects

Concert Overloaded Operators Create Objects

- Concert API implemented operator overloading for modeling objects
  - For example, `c.add(x - y <= 0);`
    - Concert firstly creates expression `x - y`
    - Then Concert creates an IloRange object `x - y <= 0`
    - Lastly, the above IloRange object is added to the constraint array or the model
  - Codes are more readable, and easier to write.
- The same operator can create different type of objects
  - `c.add(x <= y);`

```
IloConstraint operator<=(IloNumExprArg base, IloNumExprArg base2)
```

  - `c.add(x - y <= 0);`

```
IloRange operator<=(IloNumExprArg base, IloNum val)
```

Concert Overloaded Operators Create Objects

- Different object types can have some subtle differences.
  - For example, `IloCplex::getDuals()` accepts `IloRange`, but not `IloConstraint`.
- Issue can be avoided by rearranging the terms.

Perform Similar Operations in Batch

Perform Similar Operations in Batch

- Each Concert function call inevitably incurs some overhead
  - Data consistency check
  - Notifications of problem changes
  - Overhead is more significant when crossing the boundary of memory management, such as
    - calling CPLEX from JAVA/.NET
    - calling CPLEX from MATLAB
- Batch operation improves efficiency by reducing the aggregated overhead.

Perform Similar Operations in Batch

- CPLEX and concert provide batch functions to handle an array of objects to improve efficiency
  - Use function overloading to accept both single object and array of objects
    - `IloModel` has these two methods:

public const IloExtractableArray &	add(const IloExtractableArray & x) const
public IloExtractable	add(const IloExtractable x) const

  - Provide another function with suffix "s"
    - `IloCplex::getValue()` `IloCplex::getValues()`
  - Explicit function to handle batch modes
    - `IloExtractableArray::endElements()`
    - `IloSum` and `IloScalProd`

Perform Similar Operations in Batch

- An example (ending binary variables)

```
IloModel m(env);
IloNumVarArray x(env, n);

// Delete entries of x that are binary variables  (more efficient)
IloNumVarArray deletex(env);
for (int i=0; i < n; i++) {
  //Assemble extractables to delete
  if ( x.getType() == ILOBOOL )  deletex.add(x[i]);
}
deletex.endElements(); // Delete all at once


// Delete entries of x that are binary variables   (less efficient)
for (int i=0; i < n; i++) {
  if ( x.getType() == ILOBOOL )  x[i].end();  // Delete one at a time
}
```

- Usually the overhead to construct the temporary array to hold the reference of the objects to be worked on is negligible.

Perform Similar Operations in Batch

- Another example (adding scalar product to a model)

```
for (i = 0; i < n; i++) ex += a[i]*x[i];     // Slower
model.add(ex <= 10);
----------------------------------------------------
model.add(IloScalProd(a,x) <= 10);           // Faster
```

- The above constraint can also be modeled using the LP Matrix API

Suppress Excessive Notifications by Deleting CPLEX Object

Suppress Excessive Notifications by Deleting CPLEX Object

- The CPLEX object will be notified for model modifications.
- Previously we mentioned that you should perform similar operations in batch, which often helps suppress the amount of notifications.
- If there are a lot of modifications, after consolidation into batch operations, it might still be faster to end the CPLEX object first to suppress the notifications.
- If the model needs to be solved again, try to create a new CPLEX object.

LP Matrix Representation is More Efficient than Constraint Arrays

LP Matrix Representation is More Efficient than Constraint Arrays

- Matrix representation is very compact.

```
                 Column indices
                 0  1  2  ....  n-1
                +--+--+--+--+--+--+
r i   0         |  |  |  |  |  |  | ----->   IloRange ra;
o n             +--+--+--+--+--+--+
w d   1         |  |  |  |  |  |  | ----->   IloRange rb;
  i             +--+--+--+--+--+--+
  c   .         |  |  |  |  |  |  |   .
  e   .         +--+--+--+--+--+--+   .
  s   .         |  |  |  |  |  |  |   .
                +--+--+--+--+--+--+
      m-1       |  |  |  |  |  |  | ----->   IloRange rc;
                +--+--+--+--+--+--+
                 |  |           |
                 |  |    ...    |
                 |  |           |
                 V  V           V
IloNumVar        x, y,   ...    z;
```

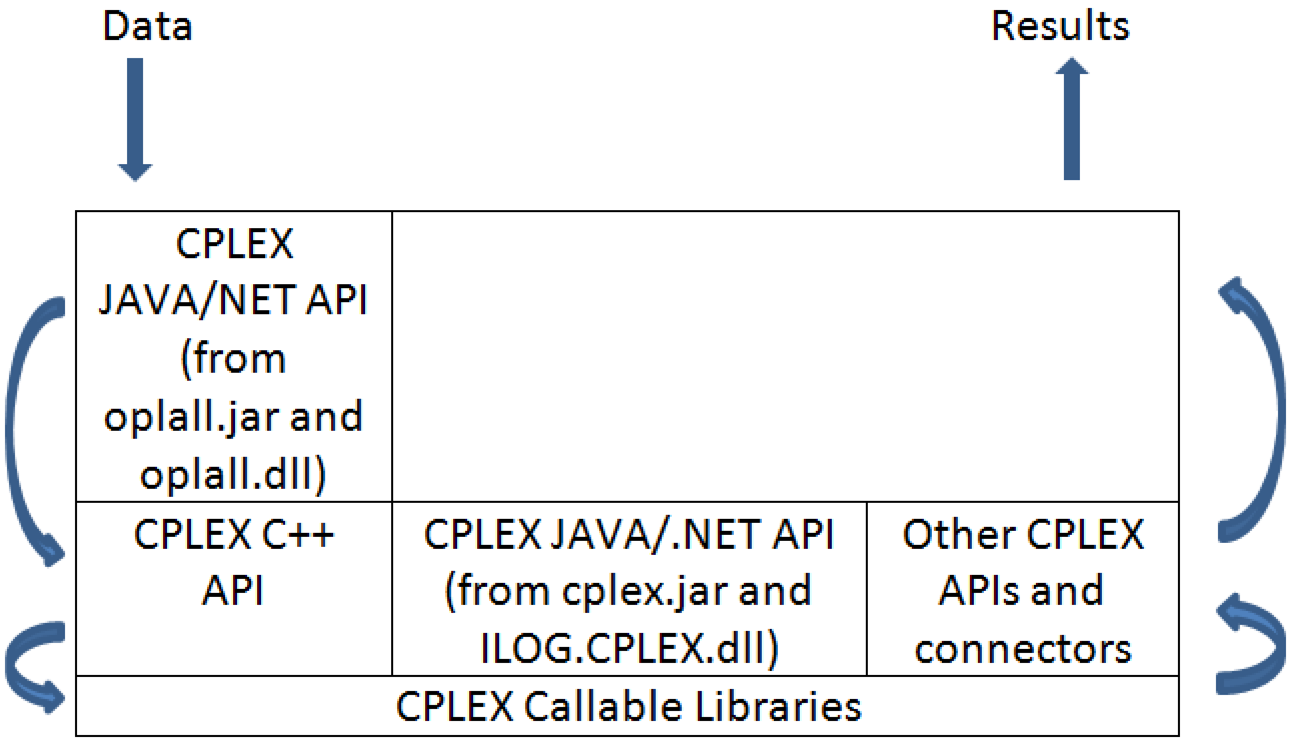
- Before solving a model, all constraints will be converted to matrix representation
- LP Matrix APIs are available in JAVA and .NET

LP Matrix Representation is More Efficient than Constraint Arrays

- The LP matrix API (`IloLPMatrix/LPMatrix`) manages constraints as a matrix
  - More memory efficient than arrays of `IloConstraint` and `IloRange`.
  - Avoid some common pitfalls in Concert modeling API
  - For very large models, LP matrix API might be the best choice.
  - It is possible to add more than one LP Matrix object to a model.
    - For best performance, use one and only one LP Matrix object for each model.

Different Layers of Concert API

Different Layers of Concert API



- Information needs to be processed at each layer before passing to the next layer
- Each layer requires some overhead. Sometimes big, sometimes small.

Different Layers of Concert API (Callable Libraries)

CPLEX JAVA/NET API (from oplall.jar and oplall.dll)		
CPLEX C++ API	CPLEX JAVA/.NET API (from cplex.jar and ILOG.CPLEX.dll)	Other CPLEX APIs and connectors
CPLEX Callable Libraries		

- The core functionalities of modeling building and optimization are implemented in callable libraries (C API).
- All CPLEX engine features are available in callable libraries.
- CPLEX Callable Libraries are most efficient, but requires more programming effort.

Different Layers of Concert API (Concert)

CPLEX JAVA/NET API (from oplall.jar and oplall.dll)		
CPLEX C++ API	CPLEX JAVA/.NET API (from cplex.jar and ILOG.CPLEX.dll)	Other CPLEX APIs and connectors
CPLEX Callable Libraries		

- The CPLEX connectors and APIs are wrappers of callable libraries with additional modeling capabilities.
  - The modeling capabilities are more for convenience
    - Piecewise linear functions and automatic linearization
  - Less efficient (both memory and speed) but easier to use
- It is possible to write a custom wrapping layer of CPLEX callable libraries.

Ease of Coding – A Sample

```
ilomipex1:
   x.add(IloNumVar(env, 0.0, 40.0));
   x.add(IloNumVar(env));
   x.add(IloNumVar(env));
   x.add(IloNumVar(env, 2.0, 3.0, ILOINT));
   c.add( - x[0] +     x[1] + x[2] + 10 * x[3] <= 20);
   c.add(   x[0] - 3 * x[1] + x[2]             <= 30);
   c.add(              x[1]        - 3.5* x[3] == 0);

mipex1:
   zmatbeg[0] = 0;    zmatbeg[1] = 2;    zmatbeg[2] = 5;   zmatbeg[3] = 7;
   zmatcnt[0] = 2;    zmatcnt[1] = 3;    zmatcnt[2] = 2;   zmatcnt[3] = 2;
   zmatind[0] = 0;    zmatind[2] = 0;    zmatind[5] = 0;   zmatind[7] = 0;
   zmatval[0] = -1.0; zmatval[2] = 1.0;  zmatval[5] = 1.0; zmatval[7] = 10.0;
   zmatind[1] = 1;    zmatind[3] = 1;    zmatind[6] = 1;
   zmatval[1] = 1.0;  zmatval[3] = -3.0; zmatval[6] = 1.0;
                      zmatind[4] = 2;                      zmatind[8] = 2;
                      zmatval[4] = 1.0;                    zmatval[8] = -3.5;
   zlb[0] = 0.0;   zlb[1] = 0.0;          zlb[2] = 0.0;          zlb[3] = 2.0;
   zub[0] = 40.0;  zub[1] = CPX_INFBOUND; zub[2] = CPX_INFBOUND; zub[3] = 3.0;
   zctype[0] = 'C';   zctype[1] = 'C';   zctype[2] = 'C';   zctype[3] = 'I';
   zsense[0] = 'L';
   zrhs[0]   = 20.0;
   zsense[1] = 'L';
   zrhs[1]   = 30.0;
   zsense[2] = 'E';
   zrhs[2]   = 0.0;
```

Different Layers of Concert API (OPL)

CPLEX JAVA/NET API (from oplall.jar and oplall.dll)		
CPLEX C++ API	CPLEX JAVA/.NET API (from cplex.jar and ILOG.CPLEX.dll)	Other CPLEX APIs and connectors
CPLEX Callable Libraries		

- The CPLEX JAVA and .NET API in OPL are slightly different from the CPLEX version due to different design.
  - Some features are not available in the OPL version
    - Most significant one is LP matrix interface
  - Classes of the same name might have different implementation.
    - Mixing files from OPL and CPLEX directories is bad

Choose an appropriate API

- Very often the choice of API is limited.
  - For example, for application server deployment, you can only use the compatible APIs.
- When there is more than one option, please consider:
  - Whether the application needs to use some advanced features
    - Export node LP: Available in Callable library only
    - Database connectivity: Available in OPL API only
  - Whether modeling generation efficiency is critical
    - Callable libraries are usually more efficient
  - Whether the ease of coding is critical
    - Callable libraries are usually more difficult to program

Choose an appropriate API

- **Model implementation and maintenance**

  Hard → Callable Library → Concert → OPL → Easy

- **Problem modifications and hot starts**

  Hard → OPL → Concert → Callable Library → Easy

- **Branch-and-bound customization**

  Hard → Callable Library → Concert → Easy

- **Matrix algorithms**

  Hard → OPL → Concert → Callable Library → Easy

- **Memory overhead**

  Higher → OPL → Concert → Callable Library → Lower

Conclusion

- We covered some details of Concert implementation.
- We discussed tips on Concert memory management.
- We also discussed some tips to improve the efficiency of concert modeling.

Technotes and Documentation Resources

- Concert Technology tutorial for C++ users
http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/topic/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/tutorials/Cplusplus/cpp_synopsis.html

- Concert Technology tutorial for JAVA users
http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/topic/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/tutorials/Java/Java_synopsis.html

- Concert Technology tutorial for .NET users
http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r5/topic/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/tutorials/Csharp/Csharp_synopsis.html

- Technote on Concert Best Practices:
http://www-01.ibm.com/support/docview.wss?uid=swg21400056

- Presentations of IBM ILOG CPLEX Optimization Studio and IBM ILOG ODM Enterprise
http://www.ibm.com/support/docview.wss?uid=swg21647915

Further ILOG Optimization Support Resources

- We are on Facebook!
  - https://www.facebook.com/ILOGOptimizationSupport
- We are on YouTube!
  - https://www.youtube.com/OptimizationSupport
- Don't forget the IBM Support Portal
  - http://ibm.com/support/

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at: http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community: http://www.webspherusergroup.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant: http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically: http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails: http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line "wste subscribe" to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com
3. **Be connected!**
   Connect with us on Facebook
   Connect with us on Twitter

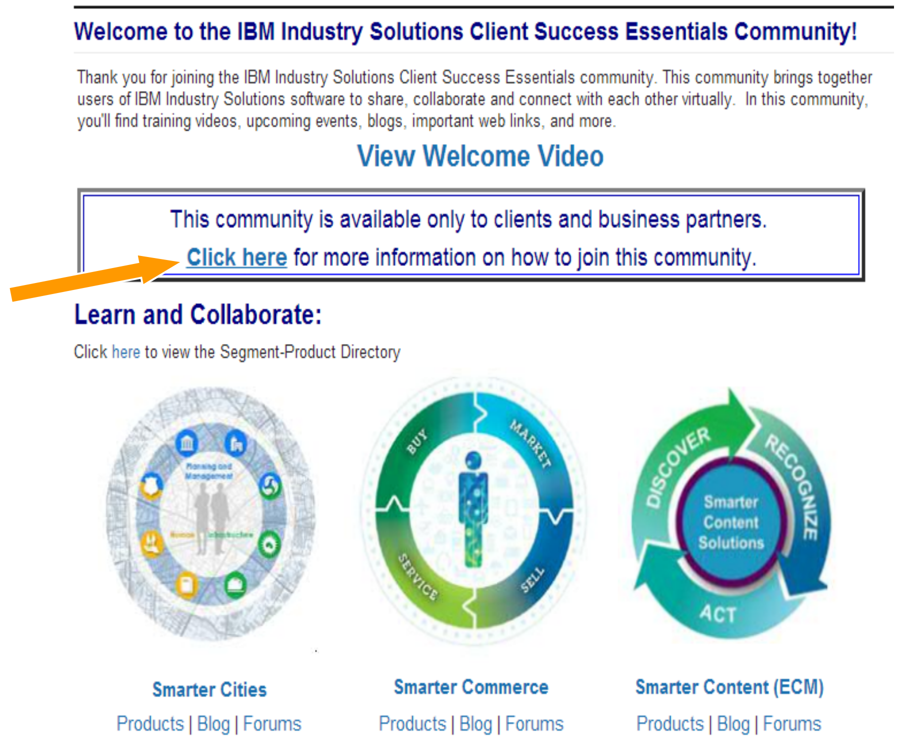
Questions and Answers

This Support Technical Exchange session will be recorded and a replay will be available on IBM.COM sites. When speaking, **do not state any confidential information, your name, company name or any information you do not want shared publicly in the replay**. By speaking in during this presentation, you assume liability for your comments.

Join Industry Solutions Client Success Essentials Community

Easily find important Support resources

- Connect with the Expert:
  - Support Technical Exchanges
  - Ask the Expert Sessions
- Product Support Newsletters
- Blog & Forums
- Training videos, IEA modules
- Event Readiness
- Proactive Services Offerings
- Essential Links to key sites:
  - IBM Support Portal
  - Client Success Portal
  - Fix Central



Join the new Industry Solutions Client Success Essentials Community   Quick start instructions