UCS 用户手册 V1.2.0

1 简介

1.1 项目背景

作为 CORBA 用户, 我们在开发和测试过程中用过许多工具，但到目前为止还没有哪个能完全满足我们的需求。因此我们决定开发一款通用 CORBA 测试工具，这就是 **U**ltra **C**ORBA **S**imulator.

UCS 具有超强的性能和许多实用的功能。下面简单介绍一下 UCS 的基本功能. 。

- 强大的 IDL 解析器
  UCS 实现了自己的 IDL 解析器，而不依赖其他厂商的接口仓库服务。所以导入 IDL 的速度非常快，不象其他 CORBA 工具需要很长时间把 IDL 导入到接口仓库。

- 友好的用户界面
  UCS 把所有 IDL 操作和参数表示在树形结构上，使得用户很容易填写参数，甚至不用特殊培训。

- 可重用的测试用例
  带有填充参数的 IDL 操作可以被保存为 xml 格式的文件。并可以被分发给其他测试人员重用。

- 自动化的操作流
  为了支持自动化测试，UCS 提供了创建操作流功能。一个操作流包含多个操作。操作之间可以建立依赖关系。另外支持结果自动匹配功能，用户可以只关注最终结果，只有最终结果没通过时才需要检查具体失败的操作。操作流在图形界面和命令行模式下都可以执行。

- 简单的测试脚本
  为了扩展服务端模拟功能，如果用户在界面上填写的参数不能满足需求，可以写 java 脚本，比如读取文件返回结果，或者根据客户端不同输入参数给出不同应答。

- 方便的工具
  为了支持开发和测试阶段的故障诊断，UCS 提供了诸如通知服务，命名服务管理， GIOP 过滤，IOR 解析等工具。

具体功能参考第三章。

1.2 目标读者

主要面向了解 CORBA 基本知识的软件开发和测试人员

.

1.3 联系方式

1.3.1 官方 web 站点:

http://www.corbatool.com

1.3.2 邮件地址:

admin@corbatool.com

1.3.3 SourceForge 链接

http://sourceforge.net/projects/ucs/

1.4 授权方式

个人用户终身免费，企业用户需要购买 license，可直接发邮件到 admin@corbatool.com 咨询相关信息。

2 软件安装

2.1 安装

安装非常简单，只要解压 zip 文件（比如 UCS_V1.2.3.zip）到某个目录即可。本软件依赖 1.4.0.2 或更新的 java 版本。如果只有 jre 环境，某些功能，比如服务扩展，插件开发等将无法执行。因为这些功能需要使用 java 编译器。

2.2 目录结构

```
UCSV1.2.3\
+---bin
+---demo
+---doc
+---etc
+---lib
+---oemlib
+---plugin
+---projects
|   +---CallBackDemo
|   |   +---classes
|   |   |   +---Generated Source
|   |   |   +---Money
|   |   |   \---package cache
```

```
|   |   +---dsi
|   |   |   \---src
|   |   +---etc
|   |   +---idl
|   |   +---log
|   |   +---scripts
|   |   \---src
|   |       \---Money
|   +---FactoryDemo
|   |   +---classes
|   |   |   +---Money
|   |   |   \---package cache
|   |   +---dsi
|   |   |   \---src
|   |   +---etc
|   |   +---idl
|   |   +---log
|   |   +---scripts
|   |   \---src
|   |       \---Money
|   +---NotificationDemo
|   |   +---dsi
|   |   |   \---src
|   |   +---etc
|   |   +---idl
|   |   |   \---include
|   |   +---ior
|   |   |   \---ucs
|   |   +---log
|   |   \---scripts
|   |       +---AttachSequencePushConsumer
|   |       \---SendOutOneNotification
|   +---test
|   |   +---dsi
|   |   |   \---src
|   |   +---etc
|   |   +---idl
|   |   +---ior
|   |   |   \---ucs
|   |   +---log
|   |   \---scripts
|   \---TMF
|       +---dsi
|       |   \---src
|       +---etc
|       +---idl
```

```
|       |   +---omgidl
|       |   \---tnmsidl
|       +---ior
|       |   \---ucs
|       +---log
|       \---scripts
+---src
+---template
\---trace
```

2.3 配置

UCS 属性文件 “CorbaMNQ.properties “位于 etc 子目录，每个项目的属性文件位于每个项目的 etc 子目录。

如果打开项目时，项目属性文件不存在，则使用 UCS 的属性文件。

属性含义如下：

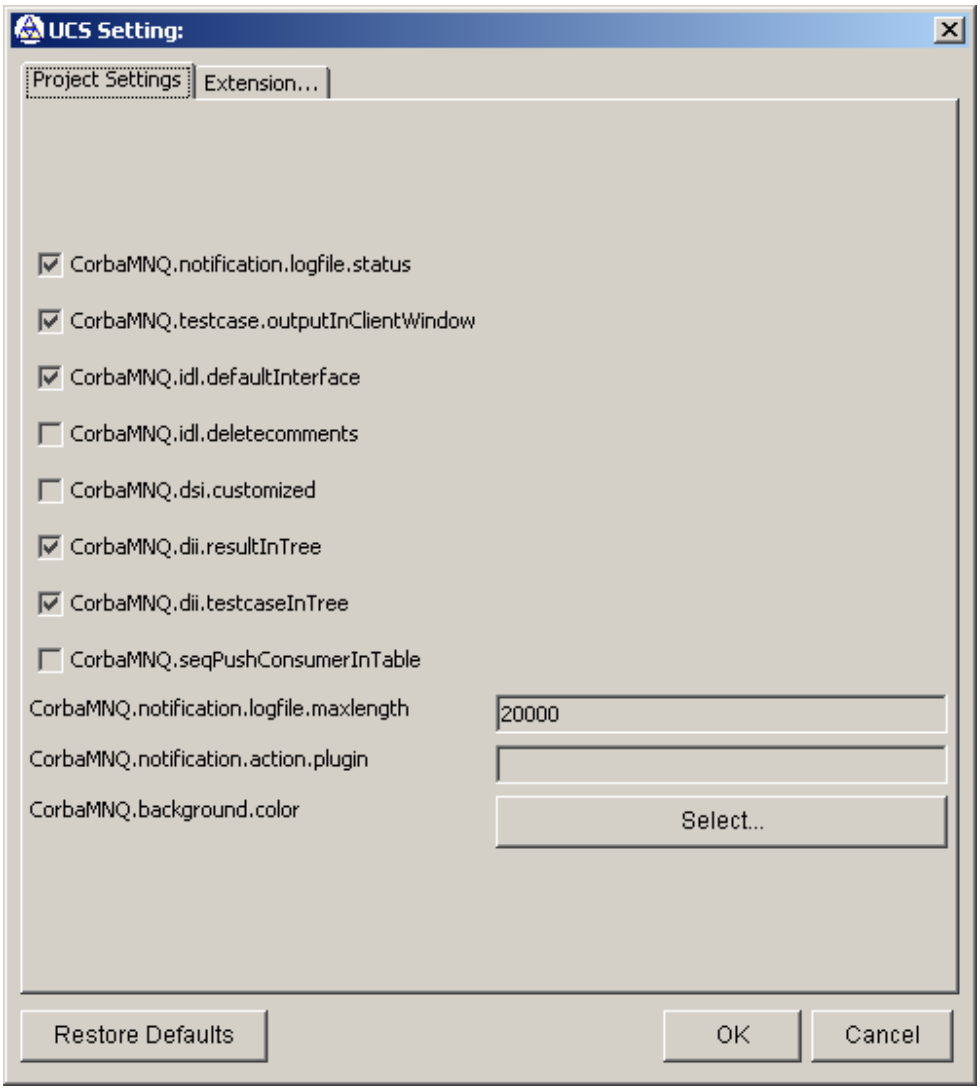
属性名	含义
CorbaMNQ.notification.logfile.maxlength	通知的日志文件最大长度（单位 Kbyte），超过此长度，将产生新的文件。默认 20M，最小 1M
CorbaMNQ.background.color	UCS 窗口的背景颜色
CorbaMNQ.dii.resultInTree	调用操作后的结果是否显示在 OperationEntry 窗口，设置此项，可以比较直观的查看操作结果，但是会比较耗资源，尤其是对有大数据量返回的操作，不建议使用。布尔值
CorbaMNQ.idl.defaultInterface	对于操作中 interface 类型的参数，是否自动创建一个有效的对象。布尔值
CorbaMNQ.testcase.outputInClientWindow	是否在 ClientOutput 窗口显示操作结果。布尔值
CorbaMNQ.notification.logfile.status	是否记录通知日志到文件。（1 是，0 否）
CorbaMNQ.notification.action.plugin	指定收到通知后调用的插件类名，用户可以实现特殊的插件来实现比如格式化通知输出等功能。
CorbaMNQ.idl.deletecomments	是否去掉 IDL 文件中的注释。布尔值
CorbaMNQ.seqPushConsumerInTable	是否以表格形式显示 PushConsumer 收到的通知。布尔值
CorbaMNQ.dii.testcaseInTree	调用操作后的结果是否显示在 OperationFlow 窗口，操作流中操作之

	间可能有依赖关系，所以建议打开此参数。对于返回大数据量的操作流，建议在命令行模式下执行，以节省资源。布尔值
CorbaMNQ.dsi.customized	是否载入用户扩展的 servant 类。布尔值
CorbaMNQ.gif.showtime	启动时显示画面的秒数，为了提高速度，不建议显示启动画面
CorbaMNQ.notification.logfile.statusy	???

下面是默认情况下生成的配置文件内容：

```
CorbaMNQ.notification.logfile.maxlength=20000
CorbaMNQ.background.color=-788885
CorbaMNQ.dii.resultInTree=true
CorbaMNQ.idl.defaultInterface=true
CorbaMNQ.testcase.outputInClientWindow=true
CorbaMNQ.notification.logfile.status=0
CorbaMNQ.notification.action.plugin=
CorbaMNQ.gif.showtime=-1
CorbaMNQ.idl.deletecomments=false
CorbaMNQ.seqPushConsumerInTable=false
CorbaMNQ.dii.testcaseInTree=true
CorbaMNQ.dsi.customized=false
CorbaMNQ.notification.logfile.statusy=1
```

上述属性可以通过任何文本编辑器修改，或者通过 UCS 菜单 Setting→Config setting…来设置。



2.4 运行

安装之后，需要修改启动脚本(Windows -- ucs.bat, Unix -- ucs.sh)中 JAVA_HOME 值，使其指向正确的路径，如果不指定，启动脚本会自动搜索一个。

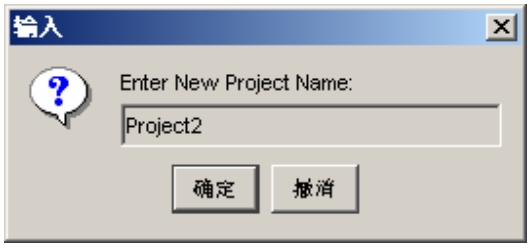
注意：在 window 2000 系统中，如果 JAVA_HOME 路径中有空格，请用“~1”来代替。比如 C:\Program Files\Java\j2re1.4.2_18 可该写成
set JAVA_HOME=C:\Progra~1\Java\j2re1.4.2_18

执行 ucs.bat 将显示如下界面：

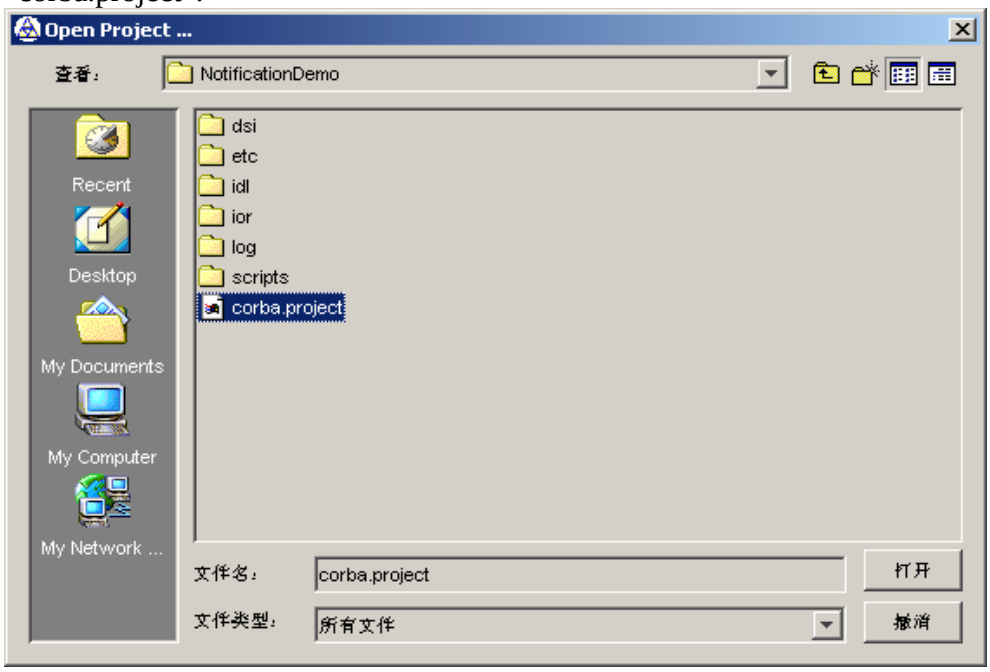
这里可以打开已经存在的项目或者创建新项目。新建的项目保存在“projects”子目录中。

注意：：

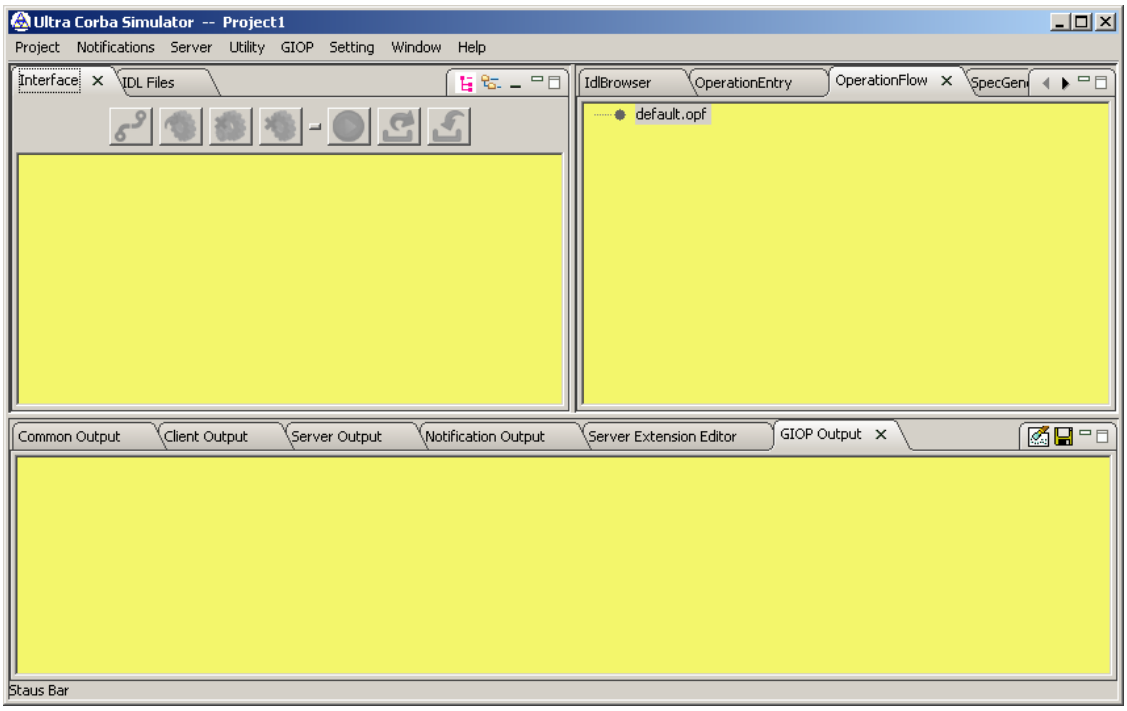
- 创建新项目时项目名不要跟其他已经存在的项目重名。



- 当打开一个已经存在的项目时，需要选择项目文件名“corba.project”.



最后将看到下面的界面：



3 使用

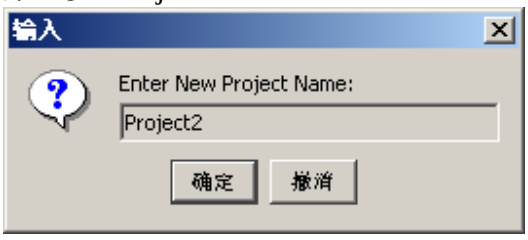
3.1 项目管理

选择 Project 菜单中 create, open 或 reload 来创建，打开或重新载入一个项目。.

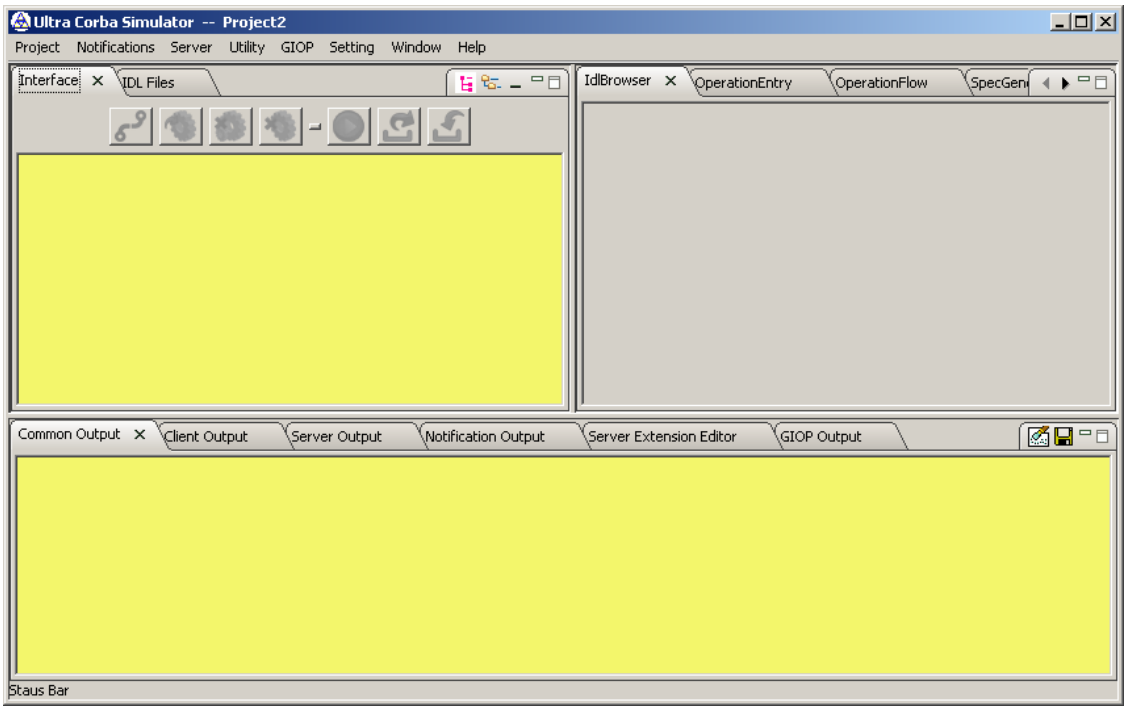
3.1.1 创建一个项目

创建了一个新项目之后，UCS 会为该项目创建所有相关子目录和文件。 (项目位于 'projects' 子目录). 比如创建了一个项目名为" ProjectName"的项目，那么就会在 UCS_HOME/projects 下生成 ProjectName 子目录。然后拷贝所有需要的 IDL 文件到 "./projects/ProjectName/idl" 目录. (以.idl 结尾的文件都会被 UCS 解析，包括子目录中 idl 文件) 最后 reload 项目就会在 UCS 中看到刷新后的数据。

第一步: Project→New …

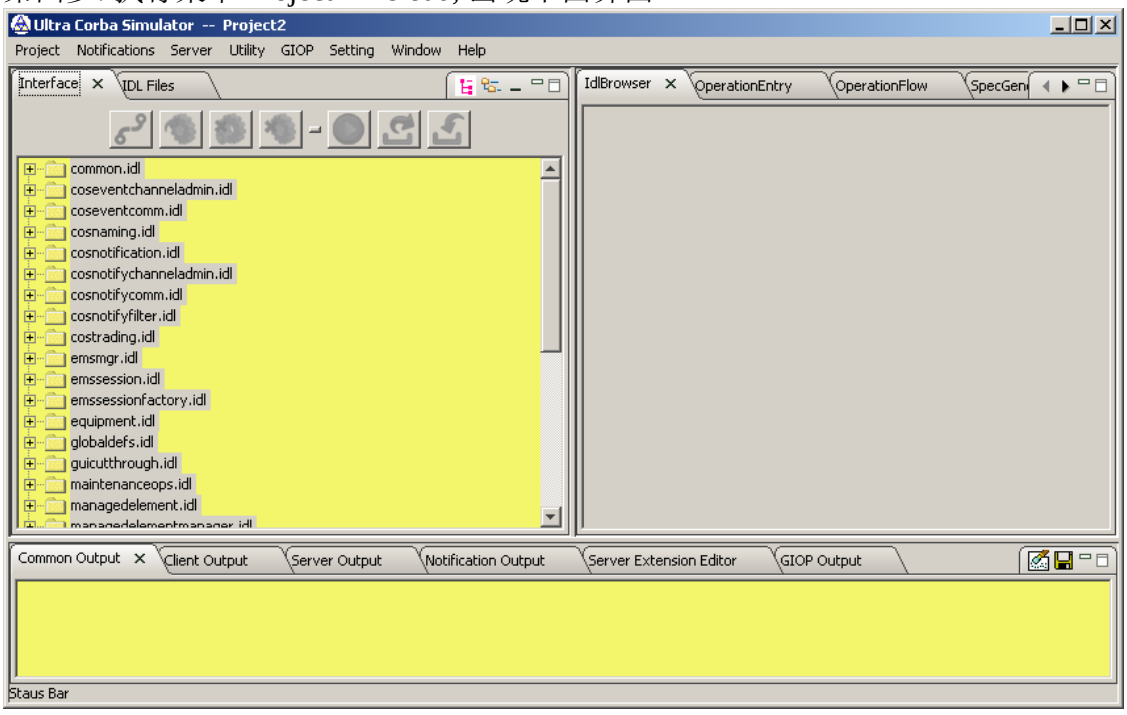


第二步: 创建项目后，你会看到：



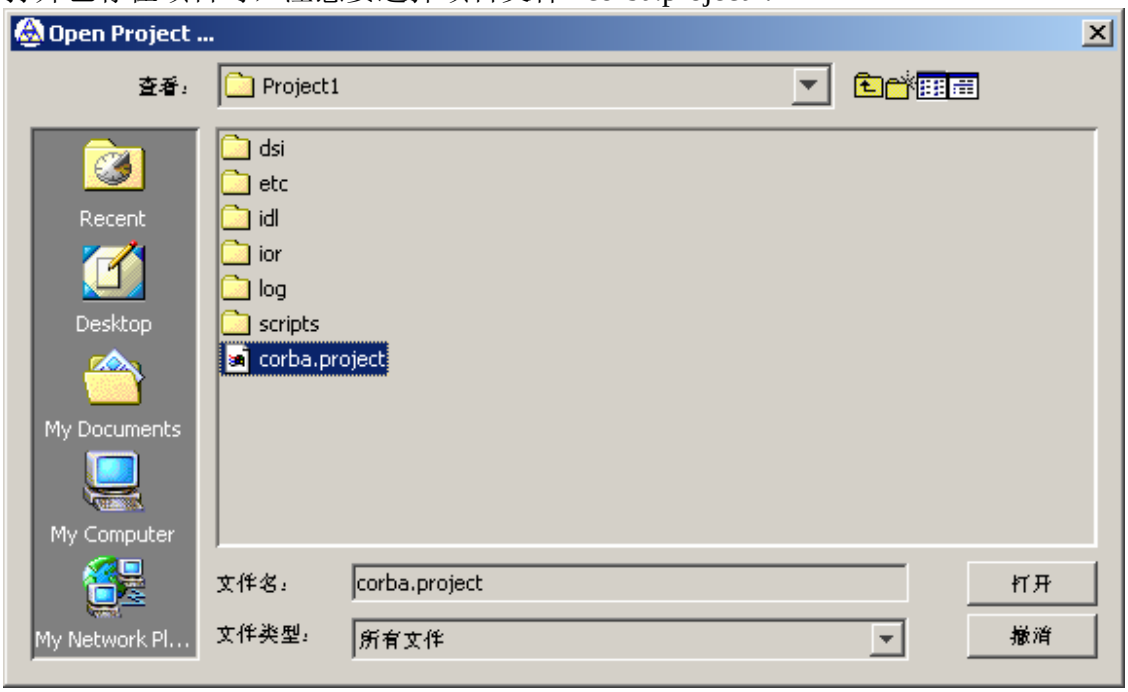
第三步: 拷贝所有 IDL 文件到 项目的 idl 子目录，这里我们拷贝一个 TMF 网管接口标准中定义的一套 idl 文件。

第四步: 执行菜单 Project→Reload, 出现下面界面



3.1.2 打开或重新载入已经存在的项目

打开已存在项目时，注意要选择项目文件 “corba.project”：



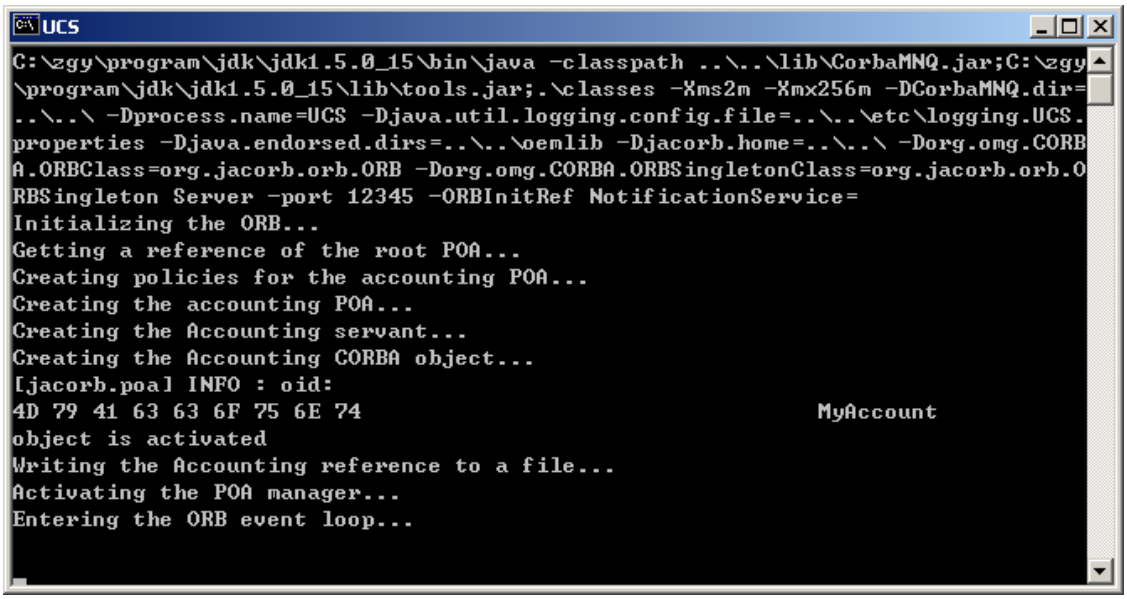
IDL 文件发生变化（比如增加，删除，修改等.)时需要重新载入项目使其生效 (Project→Reload)

3.2 CORBA 客户端模拟

即在无需编码的情况下模拟客户端调用，来测试服务端软件。当然，首先要创建一个项目，并拷贝了相关的 idl 文件。 比如我们打开 factory demo 项目。

3.2.1 启动 factory demo 的服务端

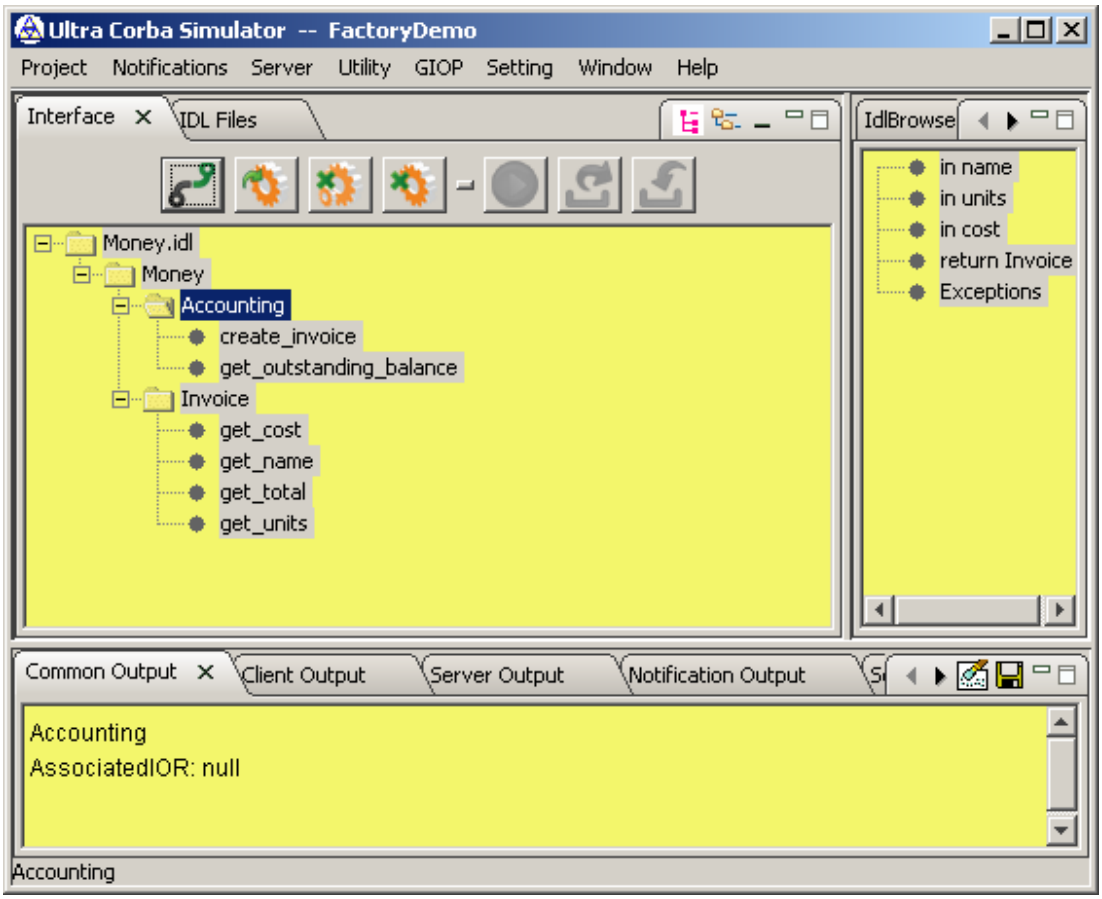
首先设置 JAVA_HOME 指向正确的路径，然后执行 FactoryDemo 目录下的 “Server.bat”.



3.2.2 关联对象 IOR

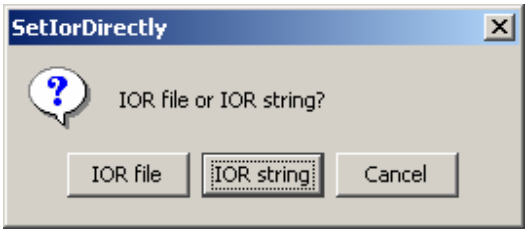
选择 Invoice 接口，然后点击 “associate an IOR to selected interface” 按纽，或者右键 setIorDirectly。

如果鼠标没有点到接口上，这个按钮是不可用的。.

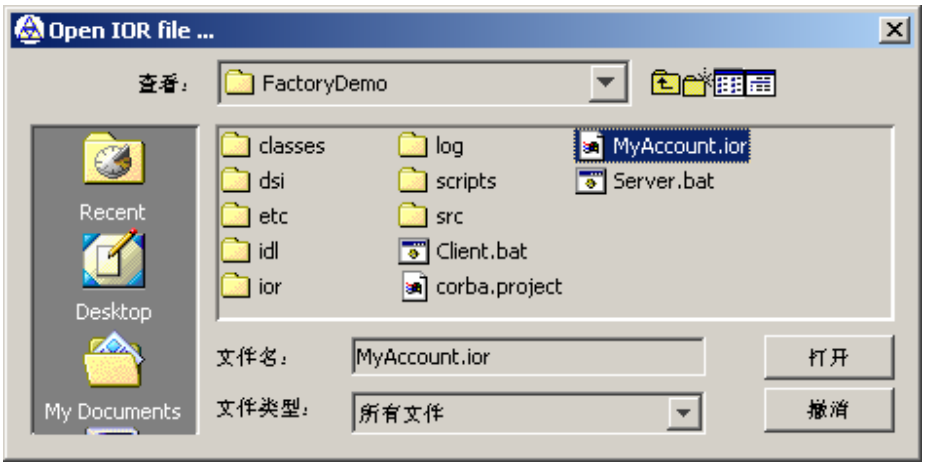


.

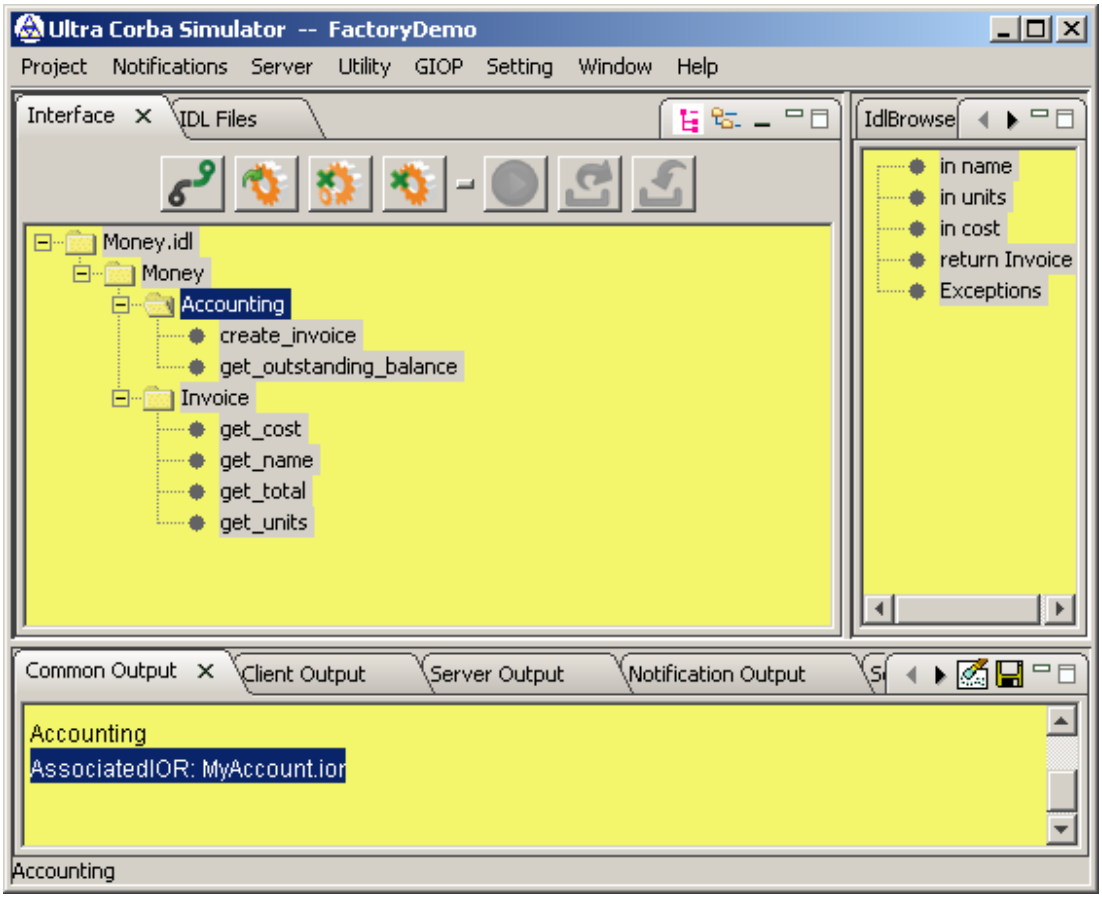
下面会问从文件导入 ior 还是直接输入字符串。



选择从文件导入 “MyAccounet.ior”

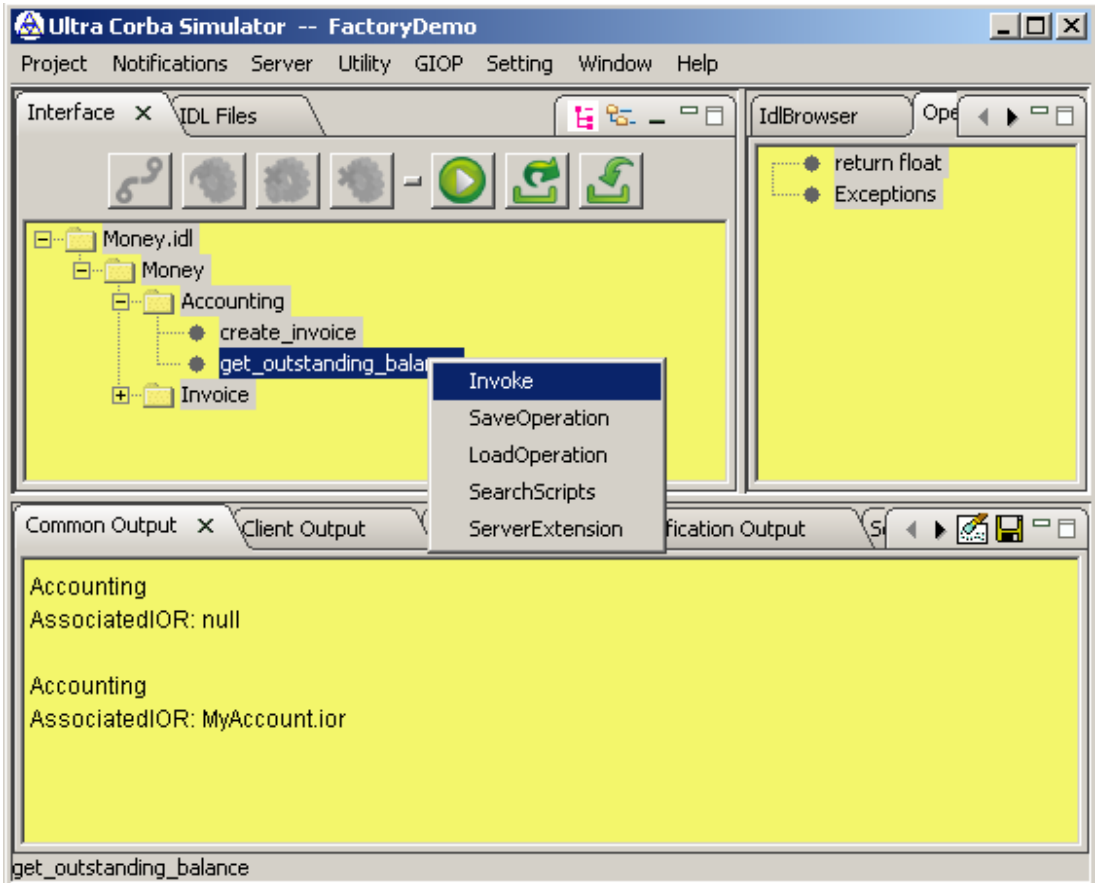


如果鼠标移开到别的对象上，再次点击 Accounting 接口时，输出窗口中会显示关联的 ior。

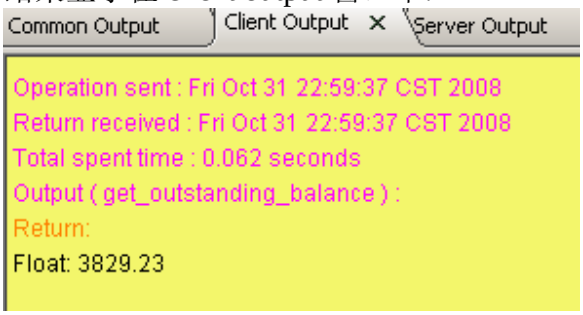


3.2.3 调用操作

在操作 get_outstanding_balance 上右键 invoke

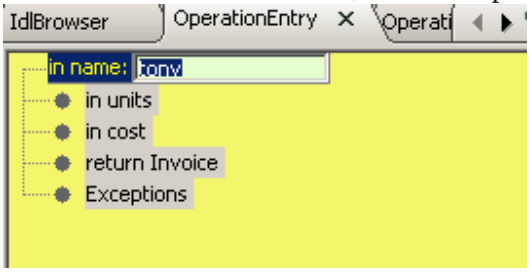


结果显示在 client output 窗口中.

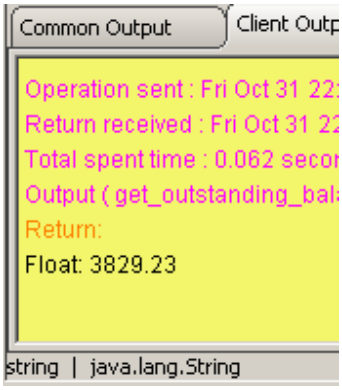


3.2.4 填写请求参数

点中 “create_invoice” 操作, 在右侧 OperationEntry 窗口中将显示参数：

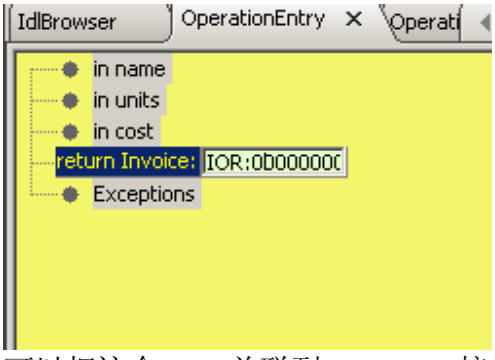


这里可以输入相关参数，如果不清楚参数类型，可以看窗口下面的状态栏提示

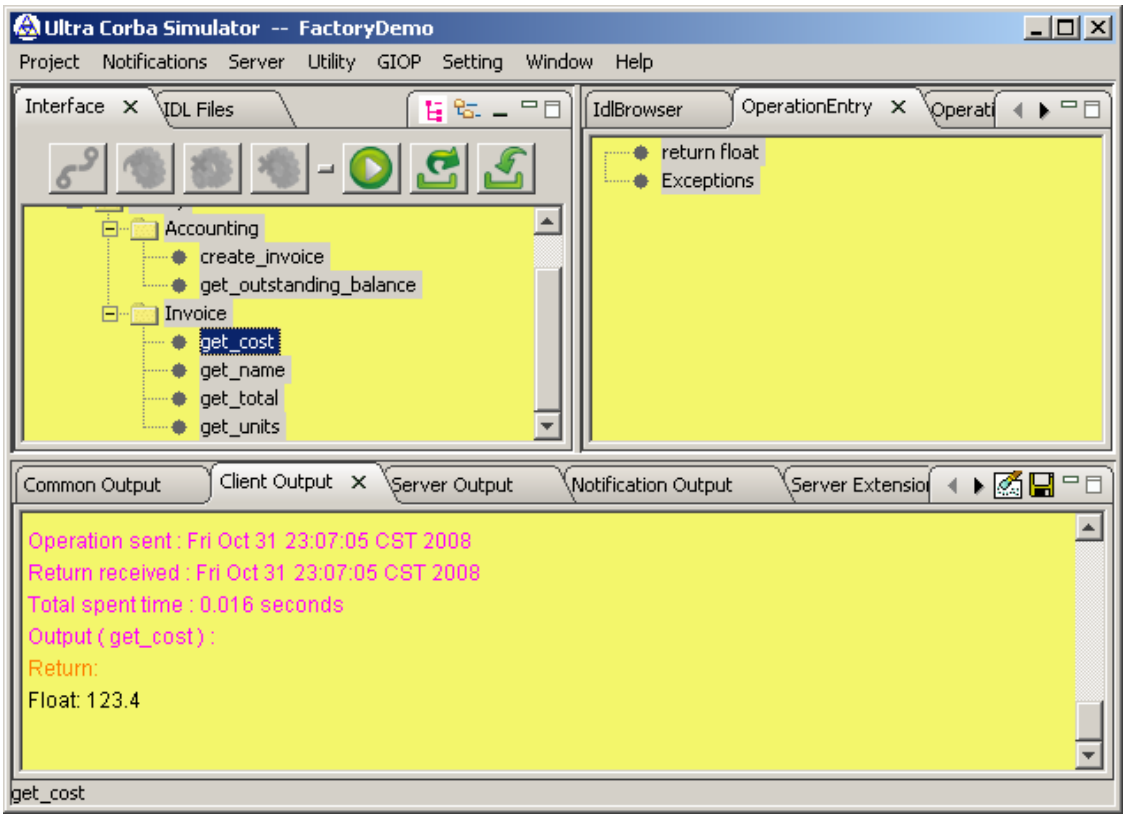


3.2.5 返回值

调用 “create invoice” 之后, 将返回一个 interface 值，实际上是一个对象 IOR。双击返回值可以进行拷贝。

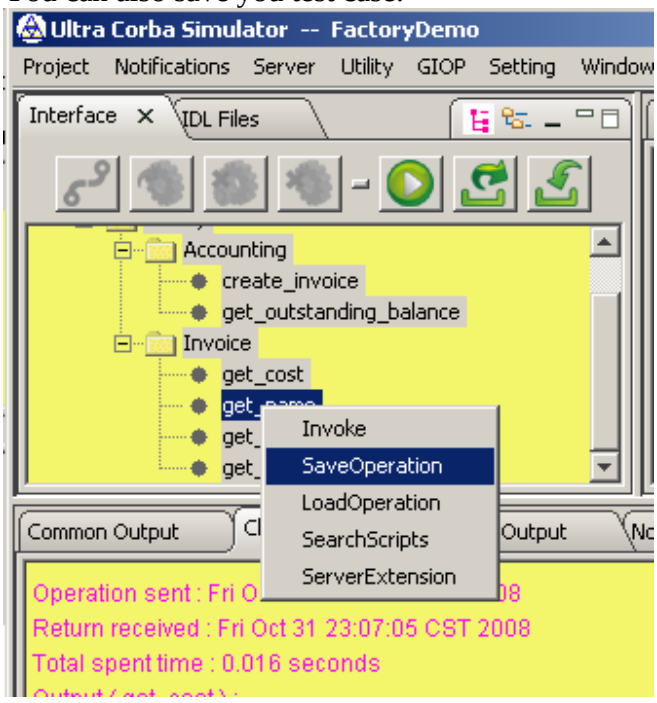


可以把这个 IOR 关联到 “Invoice” 接口上, 然后调用它的操作。



3.2.6 保存操作

You can also save you test case.



操作以 xml 格式被保存为 .op 文件，可以用任何文本编辑器打开

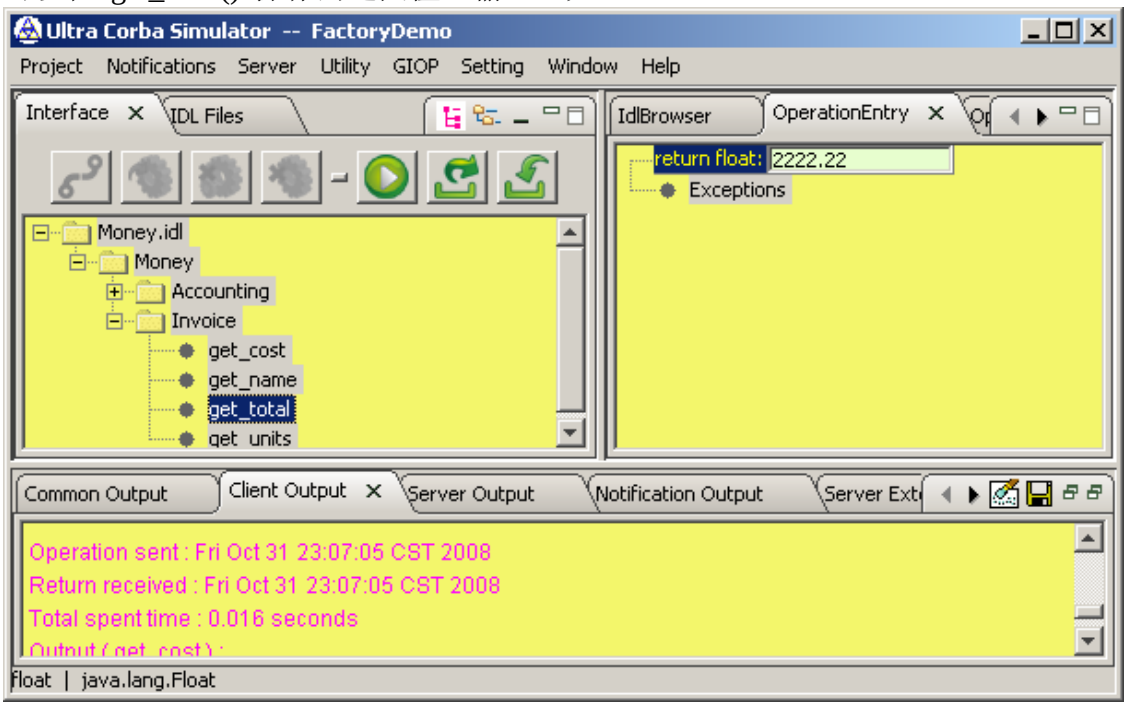
<Operation opname="get_name" cname="::Money::Invoice::get_name"
ior="IOR:000000000000001649444C3A4D6F6E65792F496E766F6963653A312E3000000000000000100000000000000068000102000000000A3132372E302E302E3100303900000015313238323138323636362F001541171E4B42412A4200000000000000200000000000000080000000004A4143000000000100000001C000000000001000100000001050100010001010900000000105010001" >
<Parameter id="0" value="" ></Parameter>
<Parameter id="1" value="None" ></Parameter>
</Operation >

3.3 CORBA 服务端模拟

这里我们用 UCS 来模拟 Factroydemo 的服务端，首先打开 Factroydemo 项目。

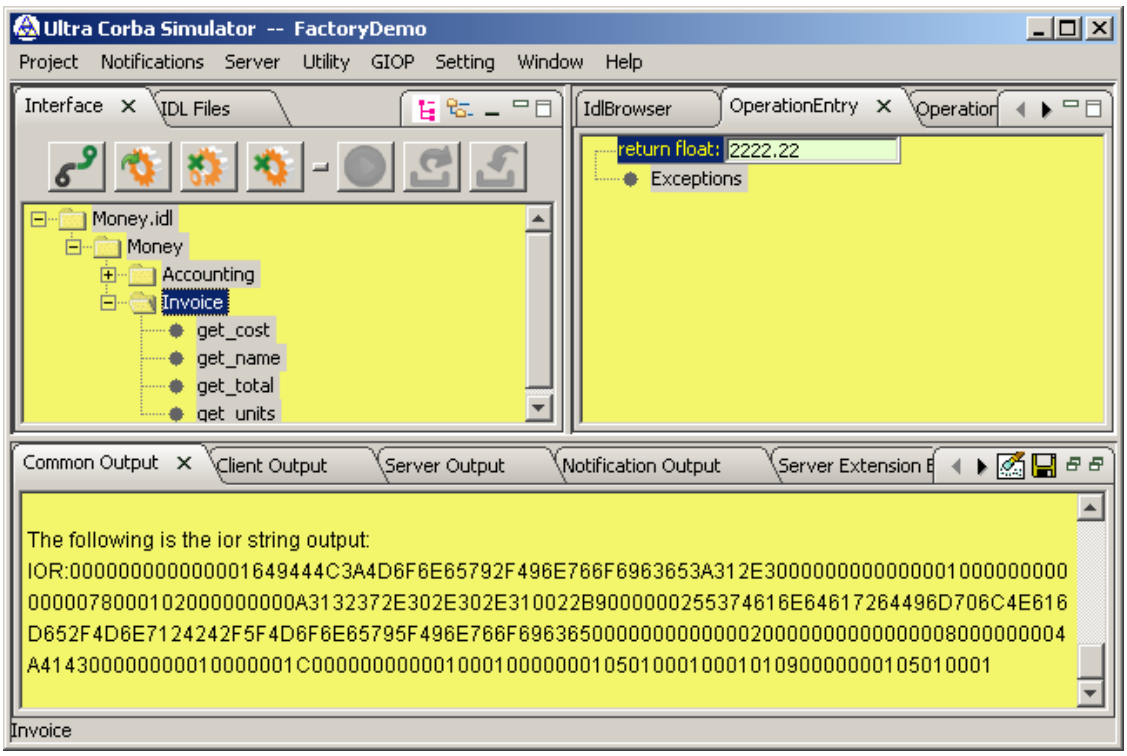
3.3.1 填写应答参数

可以在“get_total()”操作的返回值上输入比如 2222.22

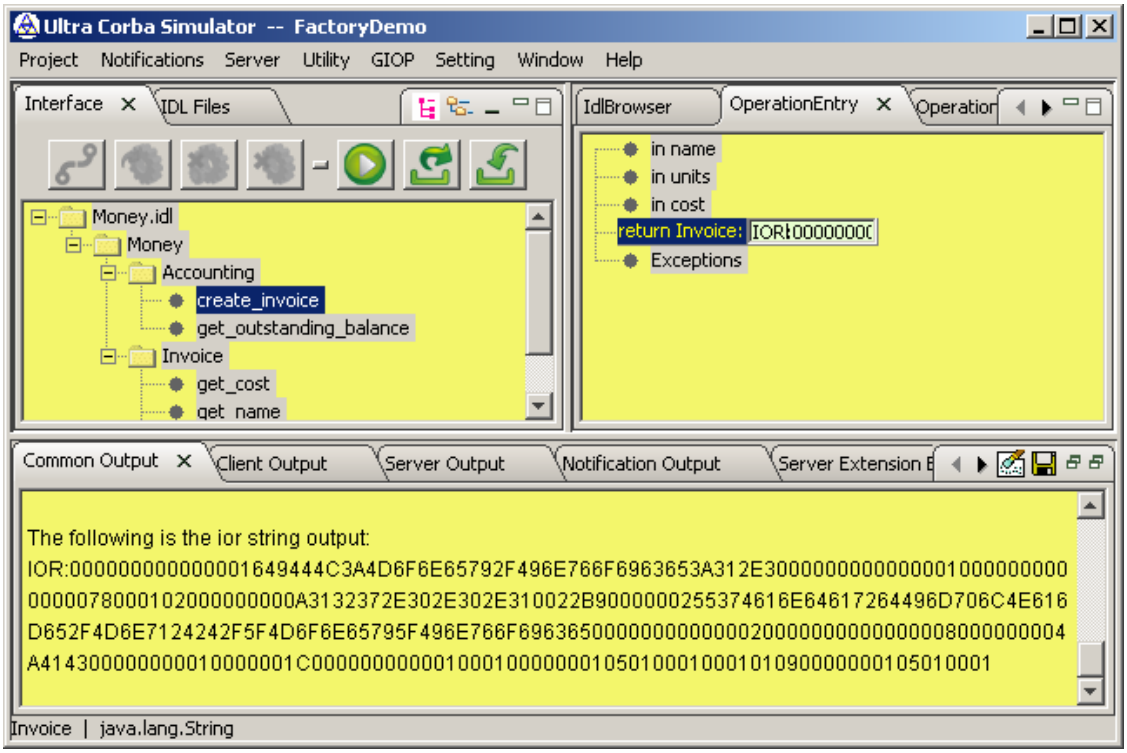


3.3.2 创建 servant

在 Invoice 接口上右键出现 createServant 菜单, 点击之后 servant 实例会被创建，IOR 会显示在 common output 窗口中。

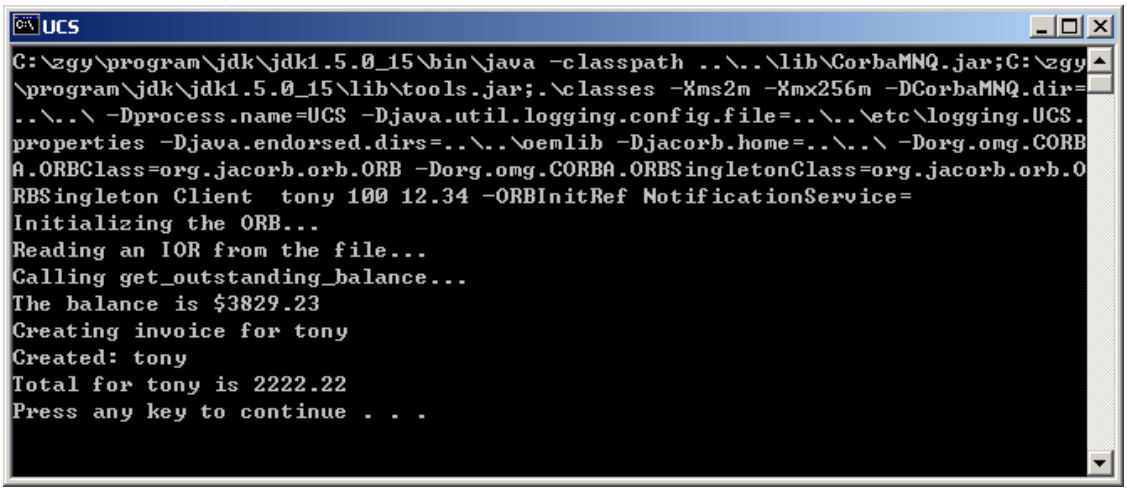


拷贝这个 IOR，把它贴到“create_invoice” 操作的返回值中。



右键 “Accounting” 接口, 创建一个 servant。

拷贝生成的 IOR 到一个文件 “MyAccount.ior”，把这个文件保存在 FactoryDemo 项目的根目录下，然后运行 “client.bat”，显示如下结果：



3.4 CORBA 服务端扩展

3.4.1 前提

- 检查 sun 的 tools.jar 文件是否位于%JAVA_HOME%\lib。(由于动态编译需要该文件)
- 使能下列熟悉 CorbaMNQ.properties (位于项目的 etc 目录)

  # Whether to load the customized servant method
  # True or False
  # Default value: False
  CorbaMNQ.dsi.customized=True

  可以通过图形界面修改或直接用文本编辑器修改。

3.4.2 步骤

1. 在 interface 窗口中，选择一个需要扩展的操作节点，
   比如 ::emsSession::EmsSession_I::getManager
   右键弹出菜单中选择 "ServerExtension"
2. 缺省的 java 代码将显示在"Server Extension Editor"窗口中
   如果以前没有对该节点写过扩展代码，那么 UCS 会创建一个缺省 java 文件；如果以前有过扩展，UCS 会载入相应的 java 文件，比如 Ucs__emsSession_EmsSession_I_getManager.java"
3. 在下面两块代码之间插入或修改你的扩展代码 (使用 java 语法)

   // TODO Below is the customized coding for servant extension
   // Begin of customization of servant extension
   // ----------Begin----------
   ...

```
                    // ------------------End----------------------
                    // End of customization of servant extension
```

4. 点击保存按钮，这个文件将被保存和编译。编译生成的 class 文件位于"project_folder/dsi"子目录；源代码位于"project_folder/dsi/src"子目录。这里的名子是 UCS_+_emsSession_EmsSession_I_getManager+.java = Ucs__emsSession_EmsSession_I_getManager.java
5. 点击 ok 后，下次启动该 servant 时，扩展的 class 会被自动调用。

3.4.3 问&答

Q1: 如何重新载入 class?

有时在 UCS 运行时，修改了服务端扩展代码很多次，为了让其生效，不必重启动 UCS，可以用菜单 "Server-->reLoadClass" 去载入最新编译的 class。

Q2: 如何重新编译所有服务端扩展的代码?

选择菜单 "Server-->reCompileAll"

Q3: 如何写 java 代码来获得输入值和设置返回值?

首先，所有输入值 (corba type)都会映射到 java 对象上。 (详细映射关系，见后面的映射表)

例 1:

```
void getManager( in string managerName, out common::Common_I managerInterface)
raises(globaldefs::ProcessingFailureException);
```

这里, managerName 映射到 java.lang.String, managerInterface 映射到 java.lang.String , ProcessingFailureException 映射到 java.lang.List

那么编码应该象下面这样：

```
if(managerName.equals("Unknown")) {
        // send the ProcessingFailureException exception
        except = new java.util.Vector();
        // insert the first member of exception output, it will be always java.lang.Integer
type.
        // Its value is the position of this exception in the "raises" sentense.
        // ProcessingFailureException is the first one, therefore, the position value is 0
        except.add(new Integer(0));
        // ProcessingFailureException is one corba exception(CTExcept), it will contain
the following
        // two member:
        // exception  ProcessingFailureException
        //        {
        //                ExceptionType_T exceptionType;
        //                string errorReason;
```

```
        //      };

        // one is corba enum (CTEnum), will be mapped to java.lang.Integer
        // value is the position value for enum value.
        // here we select the EXCPT_INVALID_INPUT value, (the third one, position
value is 2)
        except.add(new Integer(2));
        // the other is corba stirng, will be mapped to java.lang.String
        except.add(new String("haha"));
} else if(managerName.equals("EMS")) {
        // it will output EMS ior string
        managerInterface="ior string for ems";
} else if(managerName.equals("ME")) {
        // it will output ME ior string
        managerInterface="ior string for me";
}
```

注释:

对于代码里没处理的参数，UCS 会使用界面上的值填充。因此如果不想关心某些返回值，可以保持其为 null，这样 UCS 根据界面填写参数值。

3.4.4 附录: 映射表

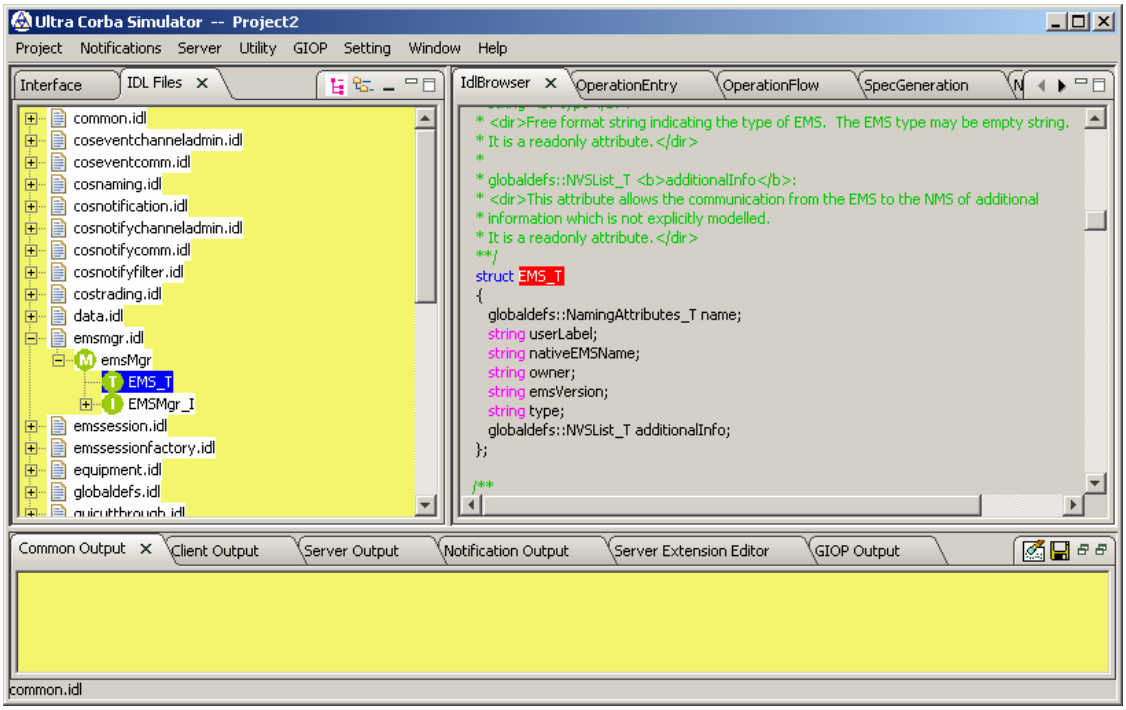
1:	CTBasic	
	"float"	java.lang.Float
	"double"	java.lang.Double
	"long double"	java.lang.Double
	"short"	java.lang.Short
	"long"	java.lang.Integer
	"long long"	java.lang.Long
	"unsigned short"	java.lang.Short
	"unsigned long"	java.lang.Integer
	"unsigned long long"	java.lang.Long
	"char"	java.lang.Character
	"wchar"	java.lang.Character
	"boolean"	java.lang.Boolean
	"octet"	java.lang.Byte
	"string"	java.lang.String
	"wstring"	java.lang.String
2:	CTArray	
	"array"	java.lang.Object[]
3:	CTDeclaration	
4:	CTEnum	
	"enum"	java.lang.Integer
5:	CTStruct	
	"struct"	java.util.Vector(List) -- list of member object value

6: CTUnion
"union" java.util.Vector(List)
-- two member,
-- first one -------- identifier object value
-- second one ------- switch body object value
7: CTSequence
"sequence" java.lang.Object[]
8: CTInterface
"interface" java.lang.String -- IOR String
9: CTExcept
"exception" java.util.Vector(List) -- list of member object value

3.5 IDL 浏览器

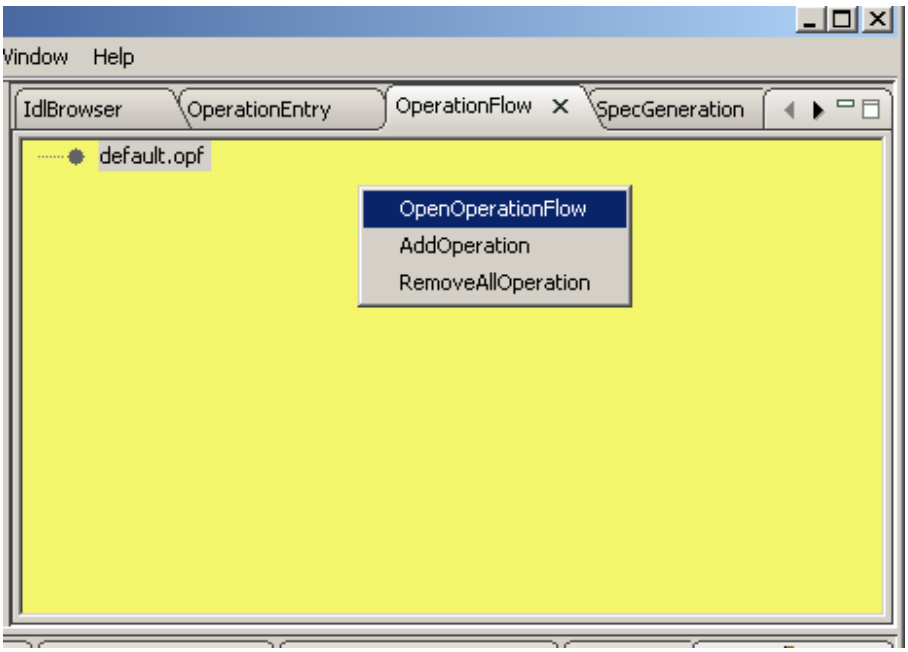
UCS 可以方便地浏览 IDL 文件。IDL 文件中的元素以树状结构显示在左侧 “IDL Files”窗口。IDL 文件的原始内容显示在右侧“IdlBrowser”窗口，并具有语法着色功能。

点击左侧窗口中 IDL 元素，UCS 的 IDL Browser 会自动导航到 IDL 文件中的相关位置，并高亮显示。见下图：



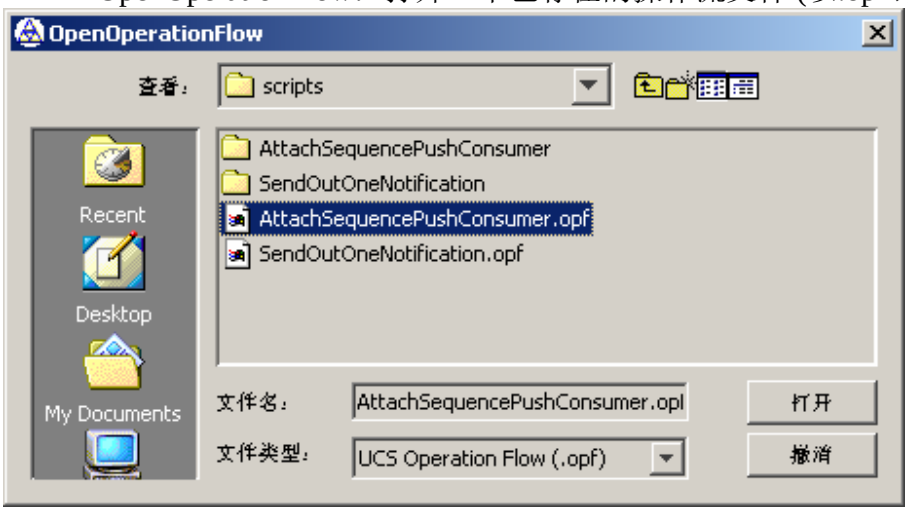
3.6 CORBA 客户端操作流

在右侧操作流 “OperationFlow” 窗口中，右键单击会弹出如下菜单：



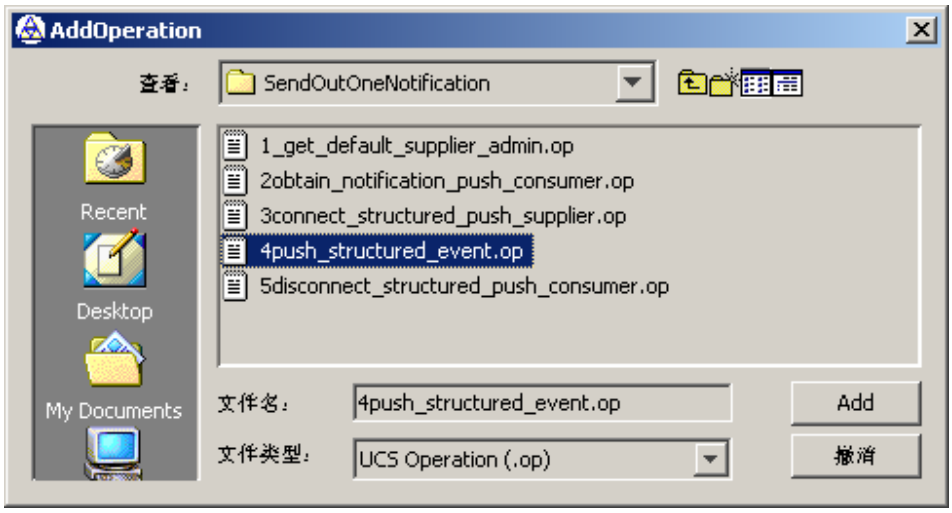
3.6.1 打开操作流菜单

OpenOperationFlow：打开一个已存在的操作流文件 (以.opf 结尾的文件).



3.6.2 添加操作菜单

AddOperation：向操作流中添加一个操作 (以 .op 结尾的文件)



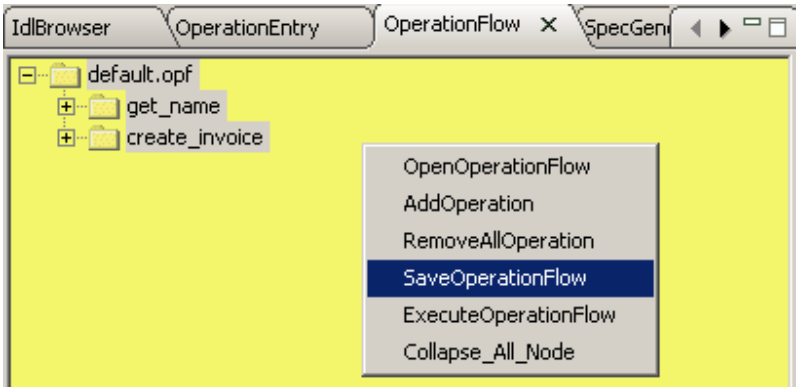
3.6.3 删除所有操作

RemoveAllOperation：清空当前操作流中所有操作

3.6.4 保存和执行操作流

在 OperationFlow 窗口中打开操作流或者增加了操作之后，右键点击操作流窗口会出现保存操作流（**SaveOperationFlow**）和执行操作流（**ExecuteOperationFlow**）的菜单。

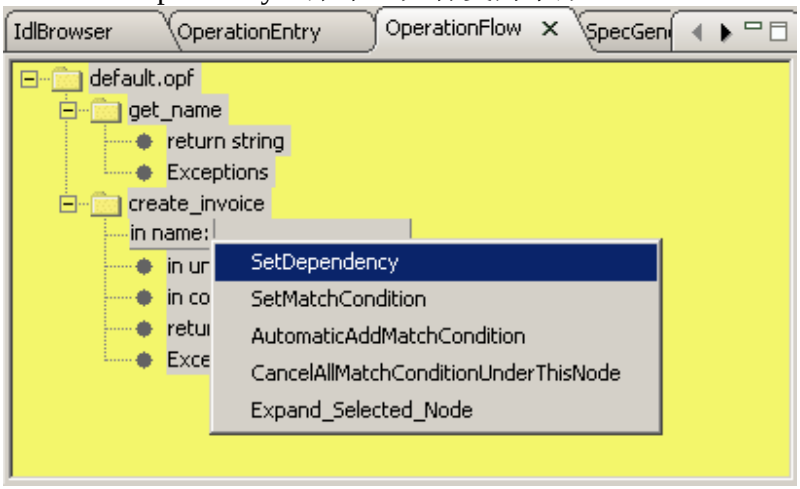
- **SaveOperationFlow**: 保存当前操作流到指定文件，同时该操作流中每个操作会被保存到与操作流同名的子目录中。
- **ExecuteOperationFlow**: 执行操作流，统计结果显示在 “Common Output”窗口中，每个操作的执行结果显示在“Client Output”窗口中。



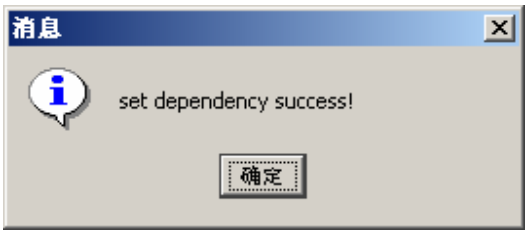
3.6.5 设置参数依赖关系

操作流中排在后面的操作的入口参数可以依赖前面操作的返回结果或者 out、inout 类型参数，运行时，设置了依赖的参数会被前面操作的结果自动替换。

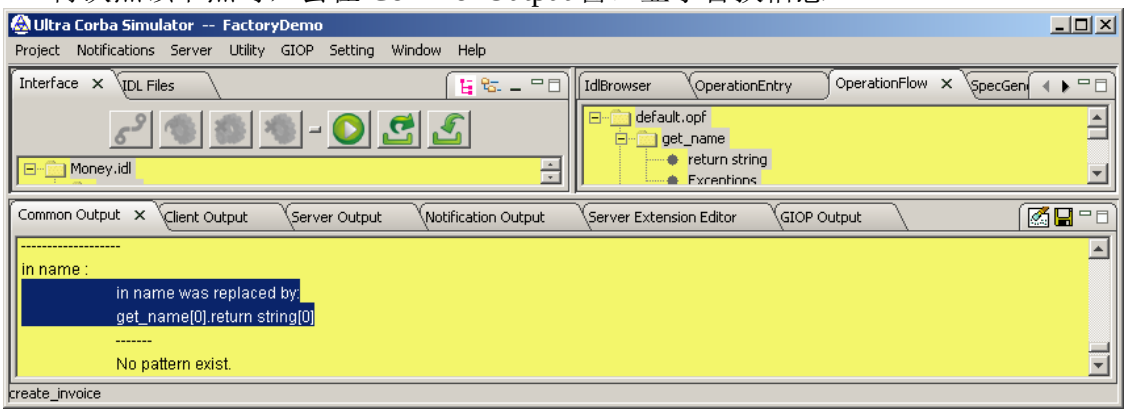
- 在操作流窗口中右键点击需要设置依赖关系的参数，选择设置依赖（SetDependency）菜单，光标变为手形。



- 用手形光标点开依赖的参数，然后左键双击，UCS 提示设置依赖成功。



再次点该节点时，会在 CommonOutput 窗口显示替换信息



3.6.6 设置匹配条件

操作流中的操作中的 out/inout 参数，返回值以及返回的异常设置匹配条件。执行操作流时，UCS 自动对操作的执行结果进行匹配检查，结果不匹配的操作在统计时被计为失败。

- 右键点击需要设置匹配条件的参数节点，选择 “SetMatchCondition”菜单。

- 输入匹配条件

对于非数字的字段，支持正则表达式语法。下面举例说明。

匹配 hello 字符串：hello

匹配以 hello 开始的字符串：hello.*

匹配以 hello 结尾的字符串：.*hello

匹配包含 hello 的字符串：.*hello.*

对于数字字段，支持大于>，小于<，等于=，不等于!。各个条件之间用括号扩起来，括号间可以是与&，或者是或|的关系。下面举例说明。

匹配大于 100.98 的数字：(>100.98)

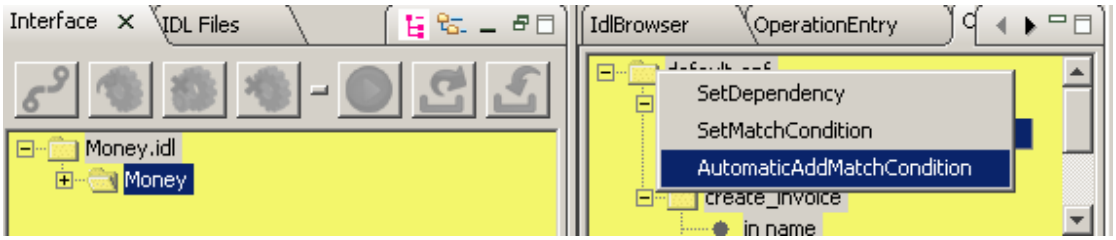
匹配大于 100 且小于 199 的数字：(>100)&(<199)

匹配大于 100 且小于 199 的且不等于 150 的，或者是值为 888 的数字：((>100)&(<199)&(!150)) | (=888)

- 最后执行操作流，结果显示在 “Common Output”窗口中。

3.6.7 自动添加匹配条件

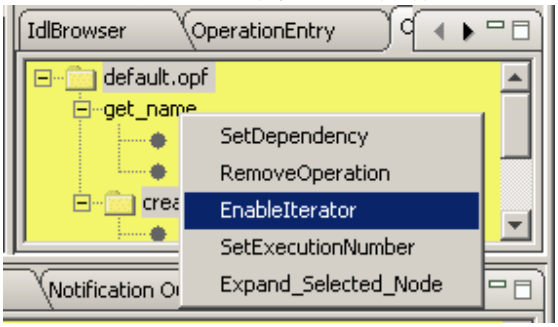
有时逐个手动设置每个参数的匹配条件需要很多时间，UCS 提供 AutomaticAddMatchCondition 菜单来自动添加匹配条件。其匹配条件为当前操作树上的结果。比如对于给定的输入参数，被测系统总是返回同样结果，如果不同则视为测试结果为失败。在这种情况下，可以在操作流里先执行一次，检测返回结果都正确，然后右键点击 AutomaticAddMatchCondition 菜单，自动添加匹配条件。



3.6.8 条件叠代功能

对于某些操作，比如 get_next 操作，可能需要对该操作叠代调用多次来遍历某些数据。直到 get_next 返回某个值，如 false 为止。这种情况可以使用条件叠代功能。

- 选中需要叠代调用的操作节点，右键菜单选择“EnableIterator”



- 然后输入叠代条件，比如，只要结果返回值为 true，就进行循环调用那么可以在叠代条件中输入“true”

取消叠代，右键点击 “CancelIterator”菜单。

3.6.9 无条件叠代功能

与“条件叠代功能”类似，只是不需要指定条件，直接使用右键菜单 SetExecutionNumber 设置某个操作节点执行的次数。

3.6.10 其他菜单

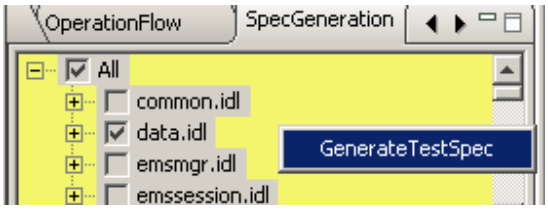
操作流中其他菜单，比较简单。

- 删除操作节点：RemoveOperation
- 取消参数依赖：CancelDependency
- 取消匹配条件：CancelMatchCondition

依赖和匹配条件信息会显示在 CommonOutput 窗口。

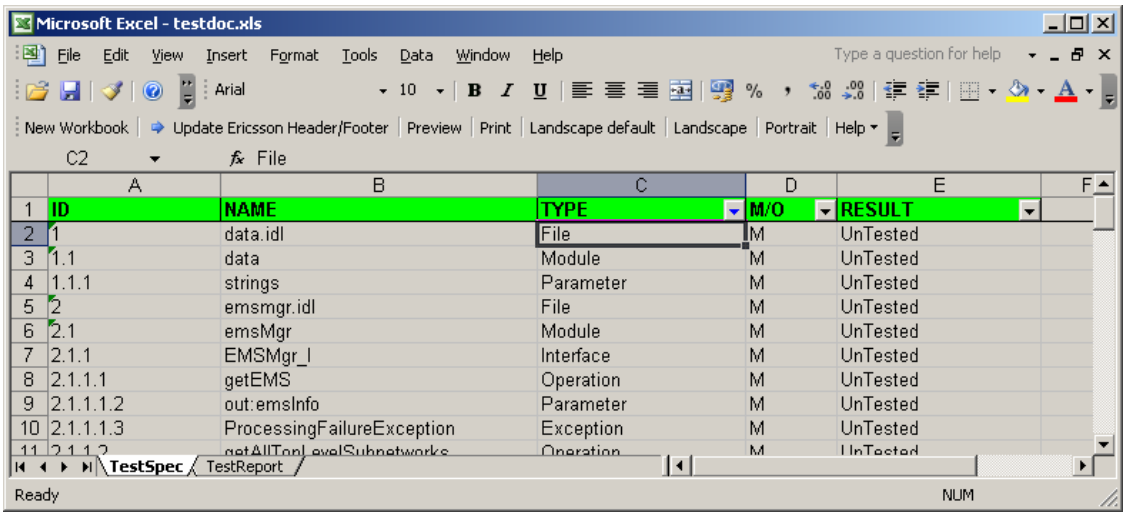
3.7 产生测试规范文档

在 SpecGenration 窗口中，选择需要生成测试规范文档的 IDL 文件或者下面的元素。右键点击菜单 GenerateTestSpec, 产生一个 MS Excel 格式的测试规范文档。



3.7.1 用作测试用例文档

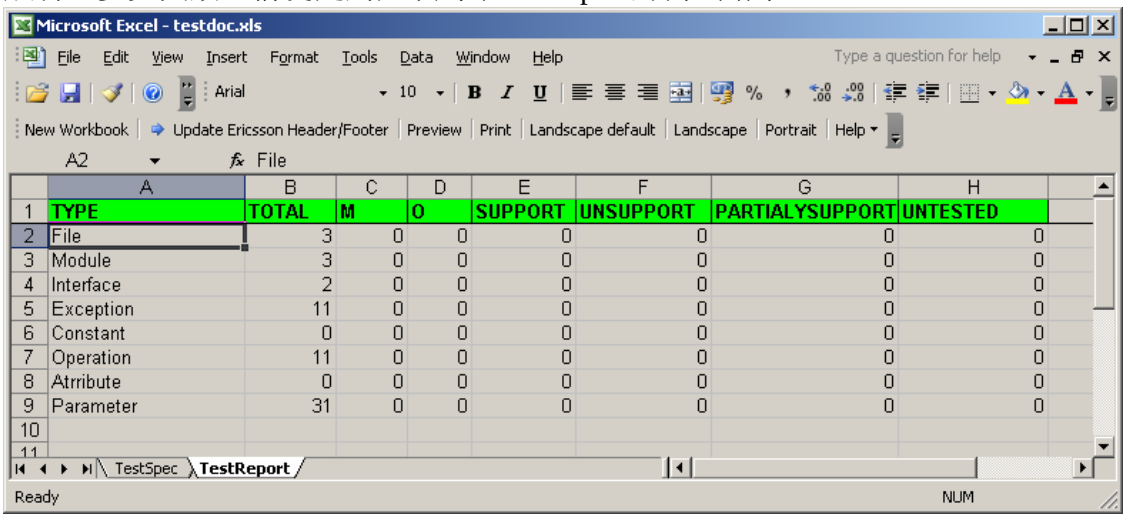
测试规范中 TestSpec 页面列出了所有 IDL 元素，比如接口、操作、参数等。测试完成后可以把结果填入 RESULT 字段。



	A	B	C	D	E	F
1	ID	NAME	TYPE	M/O	RESULT	
2	1	data.idl	File	M	UnTested	
3	1.1	data	Module	M	UnTested	
4	1.1.1	strings	Parameter	M	UnTested	
5	2	emsmgr.idl	File	M	UnTested	
6	2.1	emsMgr	Module	M	UnTested	
7	2.1.1	EMSMgr_I	Interface	M	UnTested	
8	2.1.1.1	getEMS	Operation	M	UnTested	
9	2.1.1.1.2	out:emsInfo	Parameter	M	UnTested	
10	2.1.1.1.3	ProcessingFailureException	Exception	M	UnTested	
11	2.1.1.2	getAllTopLevelSubnetworks	Operation	M	UnTested	

3.7.2 用作测试报告文档

测试规范中 TestReport 页面用来自动统计测试结果。比如测试了多少操作，有多少成功，多少失败。前提是用户填写了 TestSpec 的测试结果。

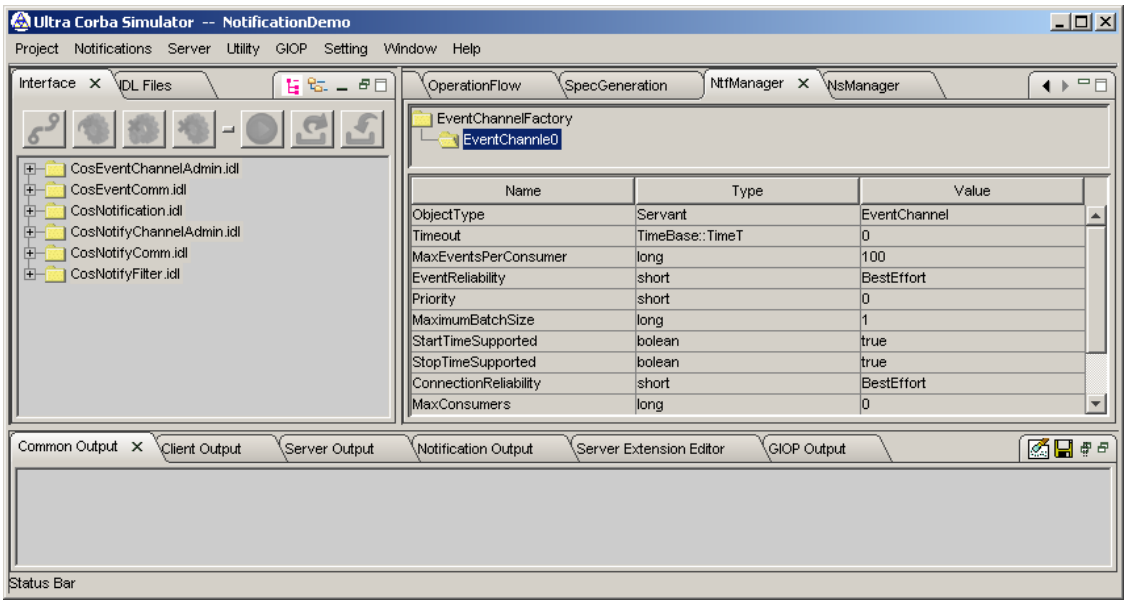


	A	B	C	D	E	F	G	H
1	TYPE	TOTAL	M	O	SUPPORT	UNSUPPORT	PARTIALYSUPPORT	UNTESTED
2	File	3	0	0	0	0	0	0
3	Module	3	0	0	0	0	0	0
4	Interface	2	0	0	0	0	0	0
5	Exception	11	0	0	0	0	0	0
6	Constant	0	0	0	0	0	0	0
7	Operation	11	0	0	0	0	0	0
8	Atrribute	0	0	0	0	0	0	0
9	Parameter	31	0	0	0	0	0	0
10								
11								

3.8 通知服务

UCS 提供了界面来管理通知服务以及发送接受通知，但不提供通知服务本身。用户可以使用任何的第三方 CORBA 厂商的通知服务。

3.8.1 通知服务管理



首先启动通知服务，并把 Notification Factory IOR 输出到某个文件。

在 NtfManager 窗口中，右键点击菜单 ConnectNotificationFactory 连接到通知服务。 NtfManager 支持下列操作：

对象	右键菜单	动作
NotificationChannelFactory	CreateChannel	创建事件通道
NotificationChannel	Destroy	删除指定的对象
SupplierAdmin	Destroy	删除指定的对象
ConsumerAdmin	Destory	删除指定的对象
ProxyConsumer	DisconnectProxy	删除指定的对象
ProxySupplier	DisconnectProxy	删除指定的对象
刷新 Refresh 和保存 SaveIOR 是每个对象都上右键菜单都支持的		

用户可以通过点击通知管理窗口中的对象来查看对象属性。

3.8.2 通知的发送和接收

UCS 使用操作流来实现通知的发送和接受功能。在 “NotificationDemo” 项目中包含下面两个操作流：

- SendOutOneNotification.opf

这个操作流用于连接一个事件提供者 supplier 到事件通道，并发送一条通知。最后断开连接以释放资源。

- AttachSequencePushConsumer.opf

这个操作流用于连接一个事件消费者 consumer 到事件通道，并等待接受通知。收到的通知可以显示在“Server output”窗口 (如果设置 CorbaMNQ.seqPushConsumerInTable＝true ， 通知将显示在 “notification output”窗口)

下面把两个操作流结合在一起演示发送和接收通知。这里我们用 JacORB 的通知服务作演示。具体步骤如下：

第一步

执行下面命令启动 JacORB 通知服务

C: \program\JacORB\bin>ntfy.bat -writeIOR c:\ntf.ior

然后通知服务的 ior 将输出到 c:\ntf.ior 中。

第二步

从 UCS NtfManager 右键点击菜单“ConnectNotificationFactory”, 选择 c:\ntf.ior.

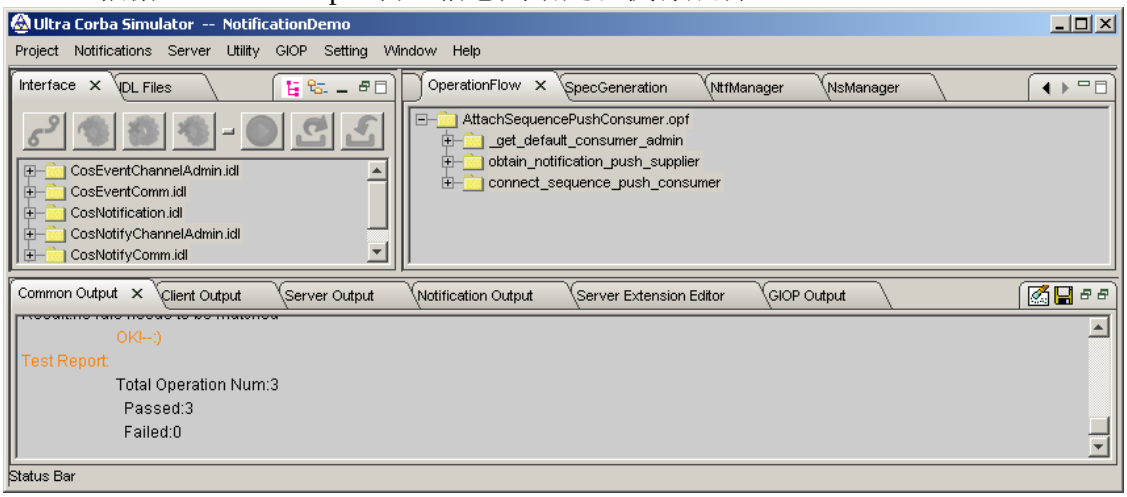
第三步

在 NtfManager 中 EventChannelFactory 对象上右键 CreateChannel 创建事件通道。在新创建的 channel 上右键保存 IOR 到文件 c:\ch.ior。

第四步

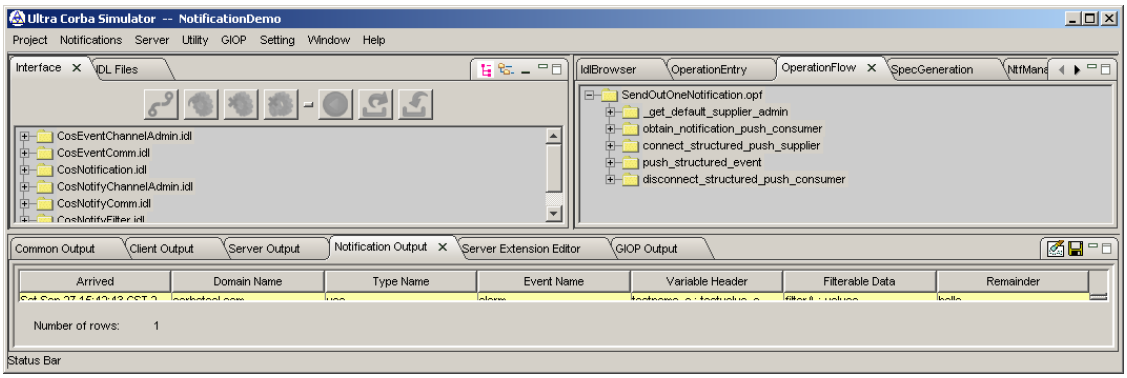
载入操作流 AttachSequencePushConsumer.opf 并执行

根据 CommonOutput 窗口信息判断是否执行成功



第五步

载入 SendOutOneNotification.opf 并执行，结果是通知面板中收到一条通知



注意：
#1 如果直接拿到了事件通道 ior, 可以跳过第二步和第三步，但 ior 要保存到 c:\ch.ior
#2 如果象改变 ior 路径，或者文件名，可以修改下面的操作 **AttachSequencePushConsumer\1_get_default_consumer_admin.op** 和 **SendOutOneNotification\1_get_default_supplier_admin.op 因为他们指向 c:\ch.ior.**
比如打开 1_get_default_supplier_admin.op 文件：

```
<Operation opname="_get_default_supplier_admin"
cname="::CosNotifyChannelAdmin::EventChannel::_get_default_supplier_admin"
ior="file:/c:\\ch.ior" >
<Parameter id="1" value="None" ></Parameter>
</Operation>
```

#3 用户可以通过操作流创建更加复杂的用例，比如增加事件质量，事件过滤等。

3.9 命名服务

UCS 只提供命名服务管理功能，但是 UCS 集成了 JacOrb 的核心库 jacorb.jar。而 JacORB 核心库提供了命名服务，所以用户仍然可以从 UCS 使用命名服务。

3.9.1 命名服务

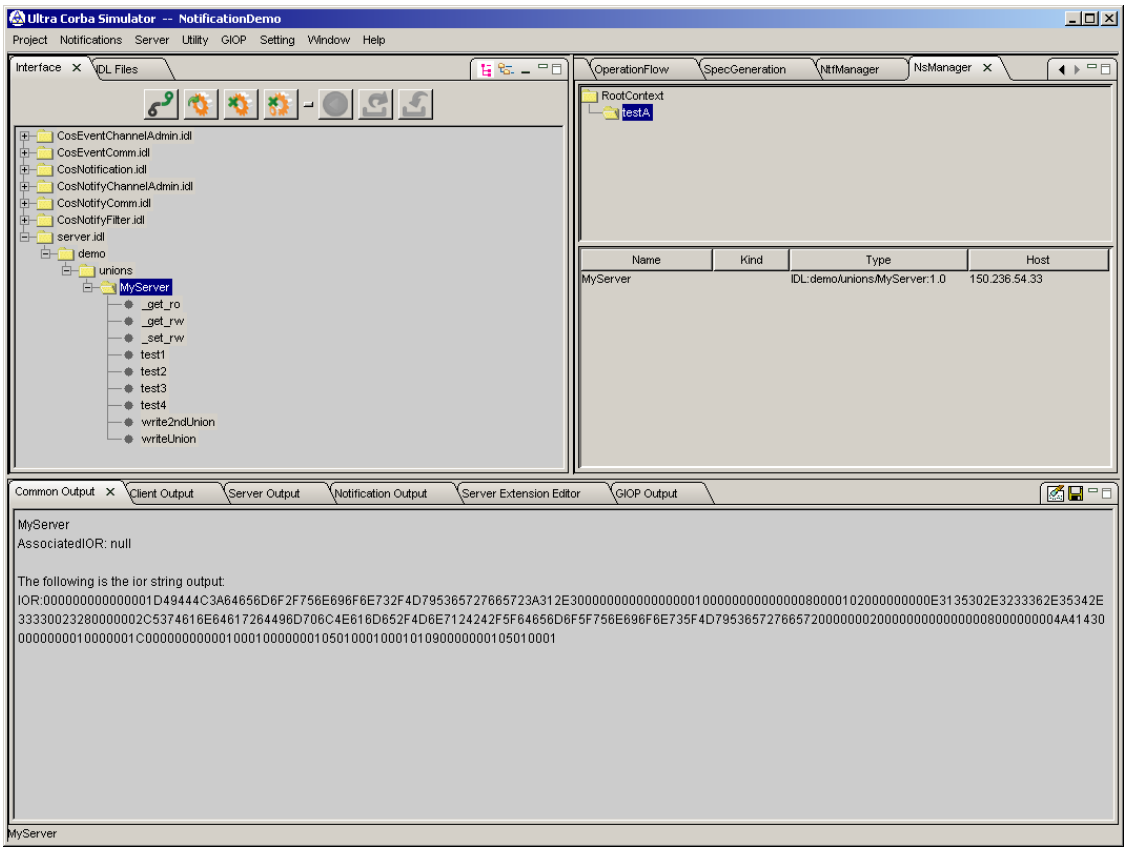
启动命名服务前首先修改 ns.bat 文件中 JAVA_HOME 指向 java 安装路径，UCS_HOME 指向 UCS 安装路径。然后从命令行运行 ns.bat。命名服务的 IOR 生成路径在%UCS_HOME%/etc/jacorb.properties 中指定。比如:

```
jacorb.naming.ior_filename=C:/ns.ior
```

3.9.2 命名服务管理

命名服务启动后，从 UCS NsManager 窗口右键连接到命名服务。UCS 提供创建上下文，绑定对象 IOR 等功能。

在下面的例子中，我们创建一个新的上下文，“testA”, 然后从左侧 Interface 窗口中创建一个 servant，最后以“MyServer”为名字把生成的 ior 绑定到命名服务上。



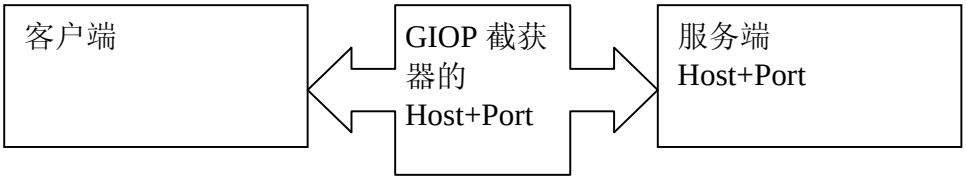
3.10 GIOP 消息截获器

3.10.1 GIOP 截获器模型

通常的 CORBA 客户服务器模式如下：

客户端通过服务端 IOR 中的 IP 和端口和服务端通信。

GIOP 截获器模式如下：

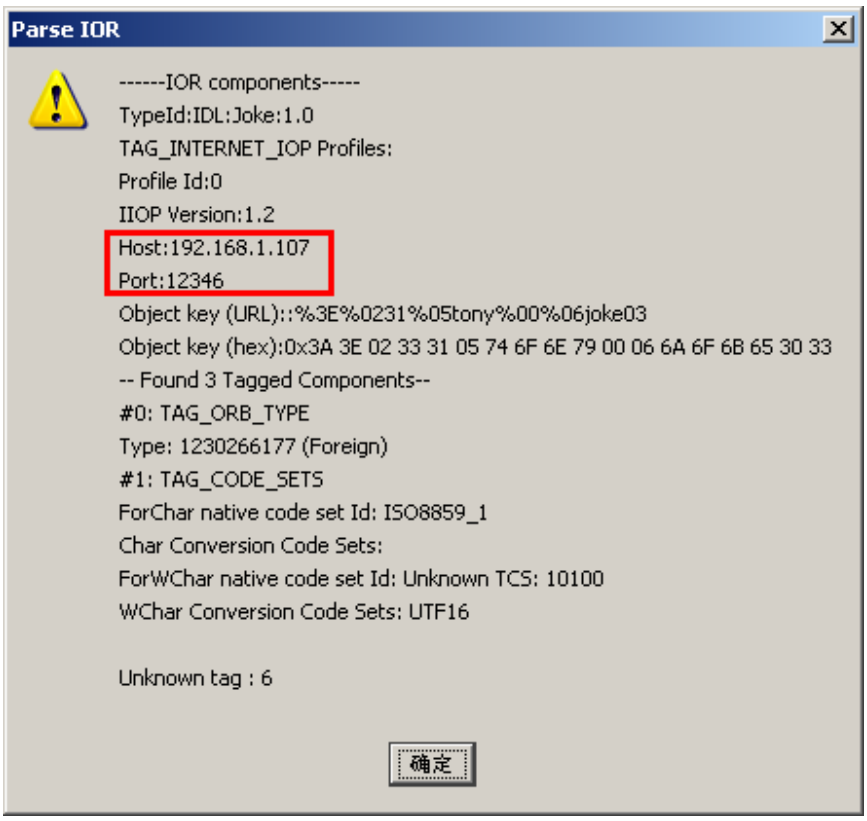


这种模式下，首先把服务端 IOR 中 Host 和 port 改成 GIOP 截获器的 IP 和监听端口。然后在这个监听端口上启动截获器。最后客户端使用改过的 IOR 文件调用服务端。实际消息是先发到截获器上，截获之后，又转发到服务端。反之亦然。
对截获的消息进行底层协议分析，尤其在 CORBA 互联不通时非常有用。

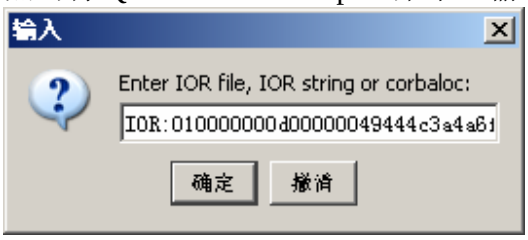
3.10.2 快速启动截获器

注：这种方式，只指定 GIOP 截获器监听端口，UCS 使用自动找到的第一个 IP 作为截获器监听 IP。

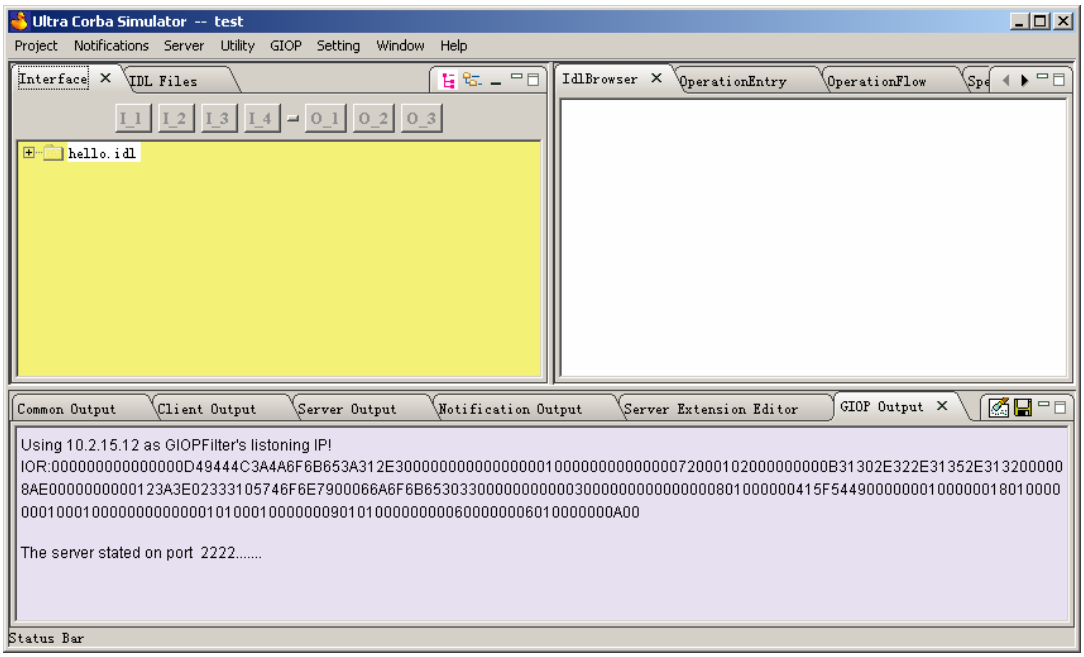
以下列原始服务端 IOR 为例
IOR:010000000d00000049444c3a4a6f6b653a312e3000000000010000000000000072000000010102000e0000003139322e3136382e312e313037003a30120000003a3e02333105746f6e7900066a6f6b6530330000030000000000000008000000010000004 15f5449010000018000000010000000100010000000000000101000100000009010100060000000600000010000000a00
解析这个 IOR 如下：



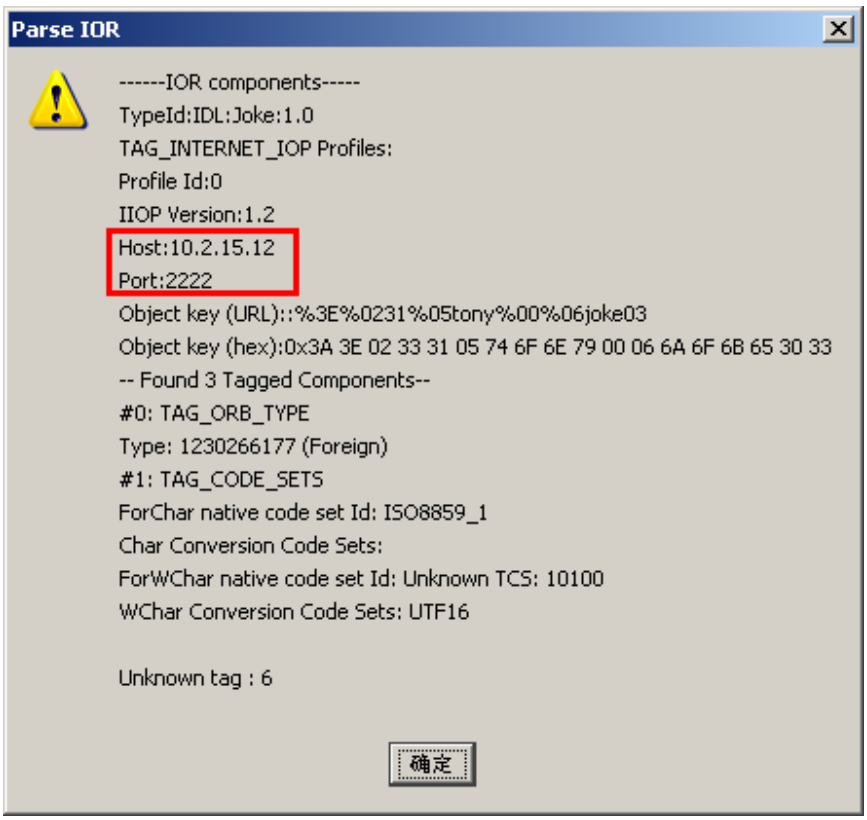
点击将 Quick start Interceptor 菜单，输入上述原始服务端 IOR



再输入 GIOP 监听端口

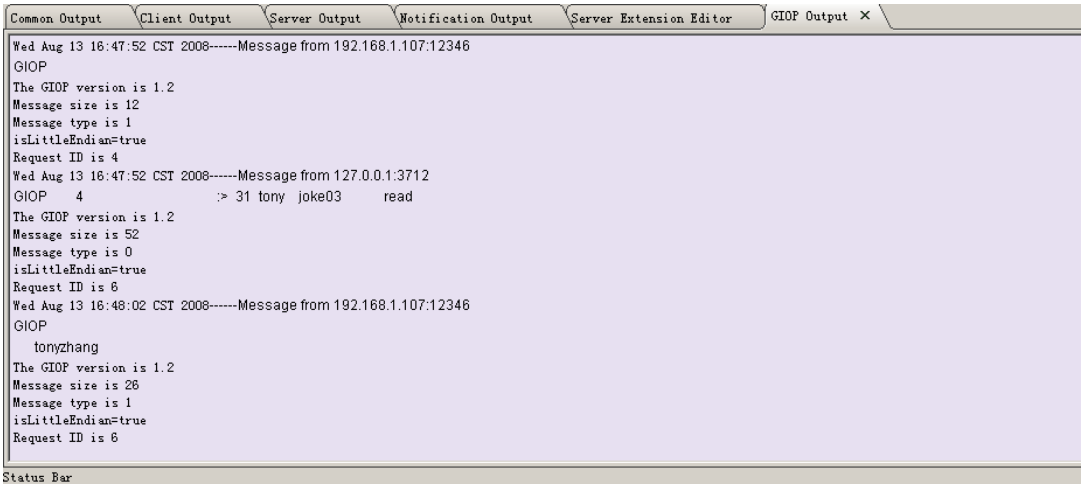


After above steps, there is an IOR be generated.
IOR:000000000000000D49444C3A4A6F6B653A312E3000000000000000010000000000000720001020000000000B31302E322E31352E3132000008AE0000000000123A3E02333105746F6E7900066A6F6B6530330000000000030000000000000000801000000415F54490000000100000018010000000100010000000000010100010000000901010000000060000000601000000A00



This means GIOP filter is listening on 2222, and its IP is 10.2.15.12. It will dispatch all the patch to the destination of 192.168.1.107:12346.

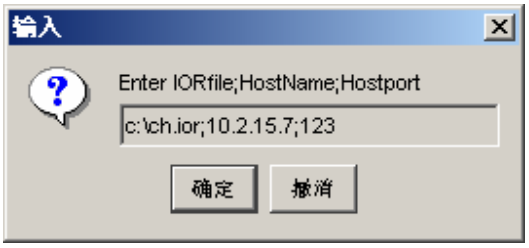
So offer client the generated IOR, and the GIOP tools and record the message between client and server.



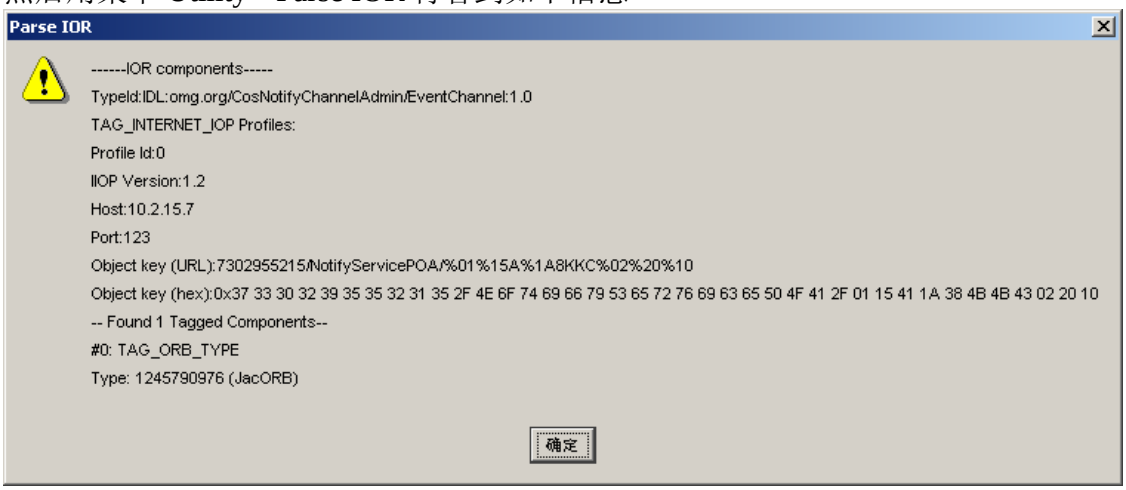
3.10.3 正常使用截获器

3.10.3.1 修改 IOR

此工具和启动截获器配合使用，来实现 GIOP 截获。点击此菜单要求输入下列参数：
IOR 文件；主机名；主机端口。三个字段用分号分开。
比如.



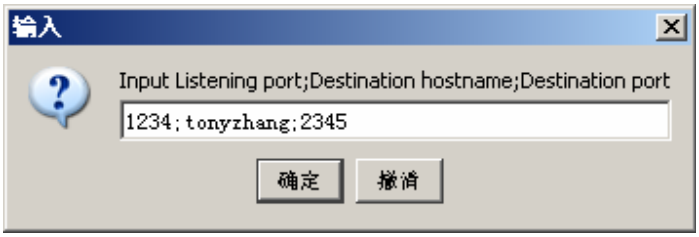
然后用菜单 Utility->Parse IOR 将看到如下信息

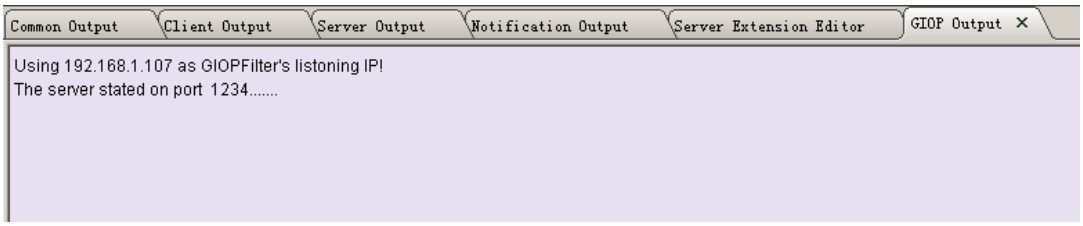


这里可以看到 Host 和 Port 变成了我们修改的值。

3.10.3.2 启动截获器

这里指定监听端口（和 Fix IOR 时指定的端口一致），目的主机名和端口（即原始服务端 IOR 中解析出来的主机名和端口）。



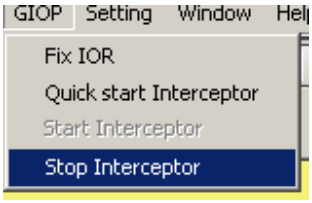


这样 GIOP 截获器将监听端口 1234, 并转发消息到 tonyzhang:2345。

此时把 Fix IOR 中被修改的 IOR 文件设置在客户端，然后调用服务端方法，UCS 将显示截获的信息。

3.10.3.3 停止截获器

停止监听功能。



3.11 其他工具

3.11.1 测试 IOR 连通性

这里使用 FactoryDemo 为例。
运行 server.bat，将在项目目录下产生 MyAccount.ior 文件
点击菜单 Utility->Ping IOR，输入 MyAccount.ior 中内容确认

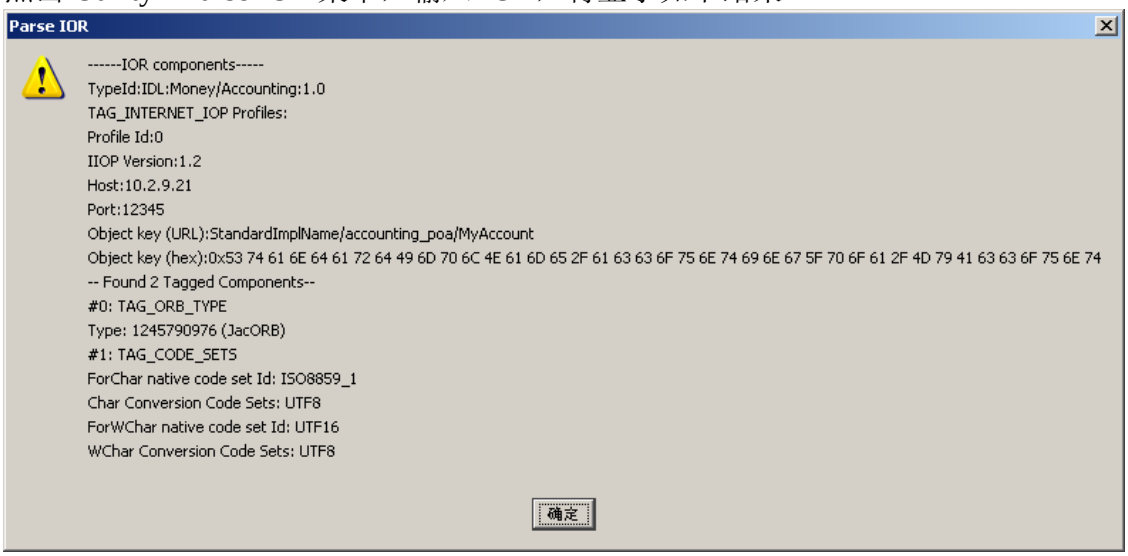
结果显示对象是存在的。

关闭 server.bat 窗口，再次执行 Utility->Ping IOR，结果如下

表明对象已经不存在。

3.11.2 解析 IOR

点击 Utility->Parse IOR 菜单，输入 IOR，将显示如下结果



这里可以看到 IOR 中 IP，端口，IIOP 版本等信息。

3.12 命令行模式

3.12.1 客户端操作流

UCS 可以再命令行模式下运行操作流 (*.opf)，下面列出其用法：

```
Windows 用法: ucs [-options]
        Or      ucs.bat [-options]
Unix 用法: ucs.sh [-opitons]
```

参数包括:

-console <directory of project>	指定UCS运行在命令行模式
-c	同 -console
-scripts <List of abstract path of OperationFlow file>	这个参数需要和-console一起使用,操作流文件之间用分号分开。 比如: -scripts D:\ucs\projects\test\scripts\flow1.opf;D:\ucs\projects\test\scripts\flow2.opf
-s	同 -scripts
-port <integer>	servant 监听端口, 默认使用 8888
-p	同 -port
-help or -h or -?	打印帮助信息

3.12.2 通知工具

这些工具保存在 UCS_HOME/bin 中

3.12.2.1 前提

指定 JAVA_HOME，可以在环境变量中指定和直接修改脚本文件。

3.12.2.2 通知接收

该工具用于快速检查是否可以从通知服务接收通知。

如何启动通知接收器?

在命令行模式下输入下列命令：

```
.D:\myjava\ucs\bin>Notify_Push_Consumer_tester.
IOR:01feffff3a00000049444c3a6f6d672e6f72672f436f734e6f746966794368616e6e656c
41646d696e2f4576656e744368616e6e656c466163746f72793a312e30000000010000000
0000000840000000010102000e0000003139322e3136382e312e31303900881339000000a
bacab305f526f6f74504f41004e6f7469667953657276696365504f4100044656661756c7
44576656e744368616e6e656c466163746f72790000000100000001000000200000001f1
13000100010002000000200001000100010509010100010000000010100
```

上面的 IOR 是通知服务的 IOR。也可以用 corbaloc 形式的 IOR。

```
.D:\myjava\ucs\bin>Notify_Push_Consumer_tester corbaloc:iiop:localhost:5000/Defau
ltEventChannelFactory
```

回车后结果如下:

```
There are 0 channels.
Please enter 'C' to create a channel, or enter an number to choose an existing c
hannel.
-
```

按 C 键，然后回车将创建一个 channel

```
There are 0 channels.
Please enter 'C' to create a channel, or enter an number to choose an existing c
hannel.
c
Channel 14 created!
Please enter 'C' to create a ConsumerAdmin, or enter an number to choose an exis
ting ConsumerAdmin.
ConsumerAdmin[0]
```

输入 0 选择一个 channel admin.

```
ConsumerAdmin[0]
0
Ready to receive events
IOR:000000000000000003349444C3A6F6D672E6F72672F436F734E6F74696679436F6D6D2F53657175
656E636550757368436F6E73756D65723A312E300000000000001000000000000000480001020000000
000E3139322E3136382E312E313039001E61000000163731313138323731323332F00153E2437463E
29012C250000000000001000000000000000800000000004A414300
```

现在进入接收通知状态。

3.12.2.3 通知发送

此工具用于发送通知到通知服务

```
D:\myjava\ucs\bin>Notify_Push_Supplier_tester corbaloc:iiop:localhost:5000/Defau
ltEventChannelFactory
```

启动后将看到下列结果

```
There are 1 channels.
Channel[14]
Please enter 'C' to create a channel, or enter an number to choose an existing c
hannel.
```

Input 输入 14 来选择 channel 14

```
Please enter 'C' to create a SupplierAdmin, or enter an number to choose an exis
ting SupplierAdmin.
SupplierAdmin[0]
```

选择 admin 0.

```
Please enter 'C' to create a SupplierAdmin, or enter an number to choose an exis
ting SupplierAdmin.
SupplierAdmin[0]
0
Please enter 'Q' to quit
Please enter Event name:
```

这里可以输入任何字符串

```
Please enter 'Q' to quit
Please enter Event name:
UCS
Please enter Event body:
UCS is very cool!!!
send finished!
Please enter 'Q' to quit
Please enter Event name:
```

如果通知接收端打开，将收到下列结果

```
Ready to receive events
IOR:000000000000000349444C3A6F6D672E6F72672F436F734E6F74696679436F6D6D2F53657175
656E6365507573684368F6E73756D65723A312E3000000000000010000000000000004800010200000000
000E3139322E3136382E312E313039001E61000000163731313138323732332F00153E2437463E
29012C250000000000001000000000000000800000004A414300
Domain name is UCS domain
Type name is UCStest
Event name is UCS
Event body is UCS is very cool!!!
```

3.12.2.4 通知对象删除

可用此工具删除 channel, admin, 或 proxy 对象

```
D:\kb\jdk1.5.0_04\bin\java -classpath ..\lib\CorbaMNQ.jar;D:\kb\jdk1.5.0_04\lib\
tools.jar -Xms2m -Xmx256m  -Dprocess.name="UCS Notify Destroy tools"  -Djava.end
orsed.dirs=..\oemlib -Djacorb.home=. -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
 -Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton com.corba.mnq.Not
ifyTestTools.DestroyTools  -ORBInitRef NotificationService=corbaloc:iiop:localho
st:5000/DefaultEventChannelFactory
There are 1 channels.
Channel[14]
Enter an number to choose an existing channel.
14
```

```
SupplierAdmin[0]
ConsumerAdmin[0]
Please enter 'D' to destory this channel. please enter 'S+number' to choose supp
lier admin, and 'C+number' to choose consumer admin.
For example C0,S1.
c0
Sequence Push Supplier[0]
Enter 'DP+number' to destory proxy.For example DP0
Enter 'D' to destory this admin.
```

回车 DP0

```
c0
Sequence Push Supplier[0]
Enter 'DP+number' to destory proxy.For example DP0
Enter 'D' to destory this admin.
dp0
Destroy Sequence Push Supplier [0] successfully
Press any key to contine,enter 'Q' to quit
```

回车

```
Destroy Sequence Push Supplier [0] successfully
Press any key to contine,enter 'Q' to quit

There are 1 channels.
Channel[14]
Enter an number to choose an existing channel.
```

输入 14

```
Channel[14]
Enter an number to choose an existing channel.
14
SupplierAdmin[0]
ConsumerAdmin[0]
Please enter 'D' to destory this channel. please enter 'S+number' to choose supp
lier admin, and 'C+number' to choose consumer admin.
For example C0,S1.
```

输入 d

```
SupplierAdmin[0]
ConsumerAdmin[0]
Please enter 'D' to destory this channel. please enter 'S+number' to choose supp
lier admin, and 'C+number' to choose consumer admin.
For example C0,S1.
d
Destory channel successfully.
```

3.13 UCS 插件

高级用户可以通过写插件来扩展 UCS 功能。插件用 java 实现特定的接口。接口文件位于 $(UCS_HOME)\ plugin\com\corba\mnq\plugin。实现文件位于 $(UCS_HOME)\plugin。这里可以通过 build.bat 来编译插件文件。

目前支持下列插件接口

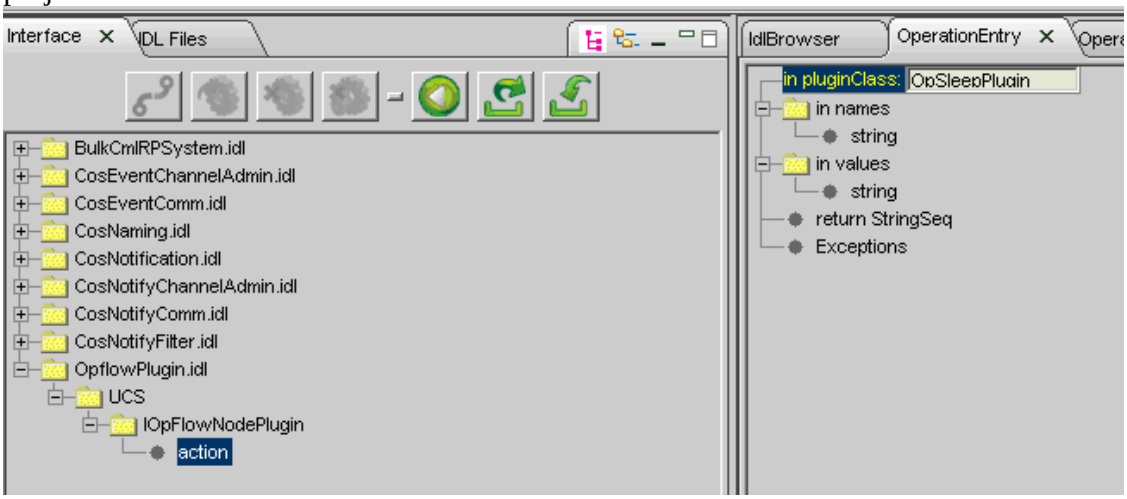
3.13.1 操作流节点插件 IOpFlowNodePlugin.java

在操作流中，如果需要实现测试功能，比如去检查磁盘文件，或者执行某个外部脚本，可以通过实现 IOpFlowNodePlugin 接口的插件来完成。下面给出 一个 Sleep 的简单实现(OpSleepPlugin.java)

```
import com.corba.mnq.plugin.*;

public class OpSleepPlugin implements IOpFlowNodePlugin {

	public String[] action(String[] names, String[] values, ILog log){
		log.output("--------- Enter OpSleepPlugin----------");
		String[] r=new String[1];
		r[0]="false";

		for( int i=0;i<names.length;i++){
			log.output("Input parameter "+names[i]+"="+values[i]);
		}
		if( values.length <1 ){
			log.output("please input sleep ms!");
		}else {
			log.output("sleep "+values[0]+" ms");
			try{
				Long x=new java.lang.Long(values[0]);
				Thread.sleep( x.longValue() );
				r[0]="true";
				log.output("sleep finished");
			}catch( Exception ex){
				ex.printStackTrace();
				log.output(ex.getMessage());
			}
		}
		log.output("--------- Exit OpSleepPlugin----------");
		return r;
	}
}
```

This is a pure Java file. After finishing that, compiling it using build.bat. Now it could be used from UCS.

Before use it the IDL file OpflowPlugin.idl has to be copied to idl directory for your UCS project.



填写 pluginClass 的值为 OpSleepPlugin (确保 OpSleepPlugin.class 存在于目录 $(UCS_HOME)\plugin 下)
填写 names 下的一个元素 值为“time”, 然后填写 values 下一个元素值为 “60000”，然后激活该操作这个操作会睡眠一分钟后返回，见下面的输出

--------- Enter OpSleepPlugin----------
Input parameter time=60000
sleep 60000 ms
sleep finished
--------- Exit OpSleepPlugin----------

当然可以保存这个操作，以便在操作流中使用。用法和其他操作一样，只是不必设置 IOR。

3.13.2 通知事件插件 IStructureEventActionPlugin.java

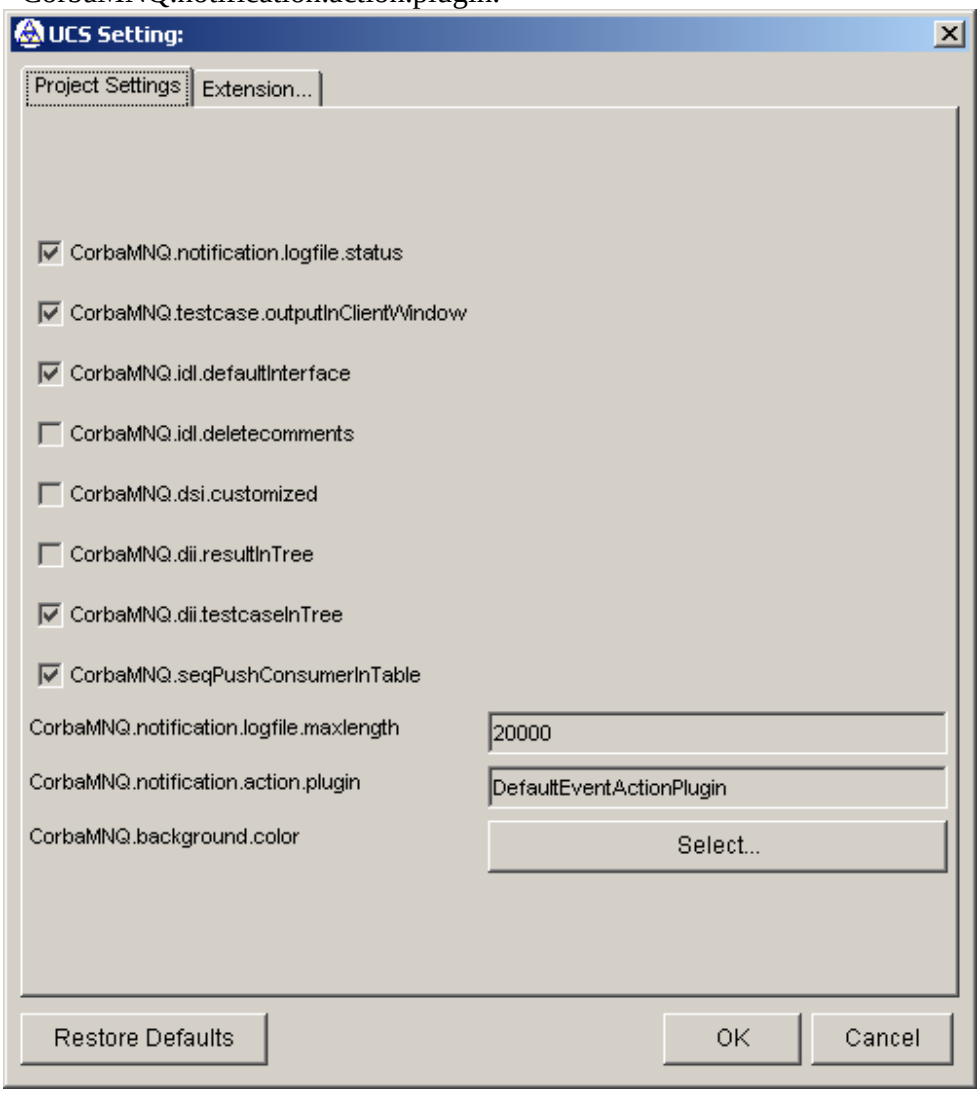
当接收到通知 StructureEvent 时， UCS 提供一个机会给用户来执行某些动作，比如格式化通知输出到某个文件。可以通过实现 IStructureEventActionPlugin 接口插件来完成。下面是个简单的例子。

(DefaultEventActionPlugin.java)

```
import org.omg.CosNotification.StructuredEvent;

import com.corba.mnq.plugin.ILog;
import com.corba.mnq.plugin.IStructureEventActionPlugin;

public class DefaultEventActionPlugin implements IStructureEventActionPlugin {


        public boolean action(StructuredEvent event, ILog log) {
                log.output("event has been received!!!!");
```

```
        return true;
    }
```

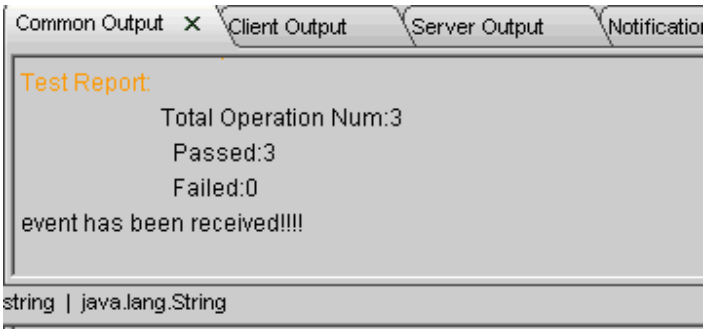
同样也要编译插件，最后在菜单中配置 Setting->Config Setting->CorbaMNQ.notification.action.plugin.



这里指定名字为 DefaultEventActionPlugin。

运行操作流 AttachSequencePushConsumer.opf 连接操作流到 channel，然后发送一条通知，在 CommonOutput 窗口中将看到下面输出

event has been received!!!!



3.14 如何配置 SSL 功能

UCS 可以运行在 SSL 模式和非 SSL 模式。取决于配置文件 (jacorb.properties)。默认是非 SSL 模式。

To enable SSL mode, pls. following the below steps:

1. Edit the jacorb.properties (located in UCSV1.2.0/etc)
   * make sure that below value is enabled (on), not disabled (off)
     jacorb.security.support_ssl=on
   * make sure that below value is corresponding to the real value for SSL
     jacorb.security.keystore=D:\\Users\\cn1wc0k0\\Desktop\\ucs.key
     jacorb.security.keystore_password=ucsucs
2. Normally, above modification is ok, For further detailed refined configuration for SSL, pls. see below sub chapters.

3.14.1 Key stores

SSL relies on public key certificates in the standard X.509 format. These certificates are presented in the authentication phase of the SSL handshake and used to compute and exchange session keys.
The Java 2 security API provides interfaces that access a persistent data structure called KeyStore. A key store is simply a file that contains public key certificates and the corresponding private keys. It also contains other certificates that can be used to verify public key certificates. All cryptographic data is protected using passwords and accessed using names called aliases.
The following section explains how to create key stores for Sun JSSE.

3.14.1.1 Setting up a JSSE key store

To set up key stores with JSSE you can use Java's keytool. In order to generate a simple public key infrastructure you can perform the following steps:

1. Create a new key store containing a new public/private key pair with keytool. The public key will be wrapped into a self-signed certificate.
2. Export the self-signed certificate from the key store into a file.
3. Import the self-signed certificate into a trust store (or configure that trustees shall be read from key store, see below).

To create a new key store containing a new public/private key pair type:

keytool -genkey -alias <alias> -keystore <keystore>

If you don't give a key store name keytool will create a key store with the name .keystore in the user's home directory. The command given above will ask for the following input:

Enter keystore password: **ucsucs**
What is your first and last name?
[Unknown]: **Developer**
What is the name of your organizational unit?
[Unknown]: **ucs**
What is the name of your organization?
[Unknown]: **ucs**
What is the name of your City or Locality?
[Unknown]: **Beijing**
What is the name of your State or Province?
[Unknown]: **Beijing**
What is the two-letter country code for this unit?
[Unknown]: **China**
Is CN=Developer, OU=ucs, O=ucs, L= Beijing, ST= Beijing, C= China correct?
[no]: **yes**
Enter key password for <testkey>
(RETURN if same as keystore password): **ucsucs**

You can view the entries of the newly created keystore by typing:

keytool -keystore <keystore> -list -storepass <password>

Now you have a public key certificate that you can present for authentication. The public key contained in the key store is wrapped into a self-signed certificate. This self-signed certificate has to be added to the Java trust store. To do this export the certificate from the key store and import it into the Java trust store located in <java_home>/jre/lib/security/cacerts.

To export the self-signed certificate into a file type:

keytool -export -keystore <keystore> -alias <alias> -file <filename>

3.14.2 Configuring SSL properties

When the ORB is initialized by the application, a couple of properties are read from files and the command line. To turn on SSL support, you have to set the following property to "on":

jacorb.security.support_ssl=on

This will just load the SSL classes on startup. The configuration of the various aspects of SSL is done via additional properties.
Configure which SSL socket factory and SSL server socket factory shall be used with the properties:

jacorb.ssl.socket_factory=qualified classname
jacorb.ssl.server_socket_factory=qualified classname

If you want to use JSSE, then configure the following as qualified classname of SSL Socket Factory and SSL server socket factory:

org.jacorb.security.ssl.sun_jsse.SSLSocketFactory
org.jacorb.security.ssl.sun_jsse.SSLServerSocketFactory

As explained in the previous section, cryptographic data (key pairs and certificates) is stored in a key store file. To configure the file name of the key store file, you need to define the following property:

jacorb.security.keystore=AKeystoreFileName

The key store file name can either be an absolute path or relative to the home directory. Key stores are searched in this order, and the first one found is taken. If this property is not set, the user will be prompted to enter a key store location on ORB startup.
To avoid typing in lots of aliases and passwords (one for the key store, and one for each entry that is used), you can define default aliases and passwords like this:

the name of the default key alias to look up in the key store
jacorb.security.default_user=brose
jacorb.security.default_password=ucsucs

Note that when using Sun JSSE: The javax.net.ssl.trustStore[Password] properties doesn't seem to take effect, so you may want to add trusted certificates to "normal" key stores. In this case configure JacORB to read certificates from the key store rather than from a dedicated trust store, please set the property

jacorb.security.jsse.trustees_from_ks=on

SSL settings can be further refined using security options as in the following property definitions:

jacorb.security.ssl.client.supported_options=0
jacorb.security.ssl.client.required_options=0
jacorb.security.ssl.server.supported_options=0
jacorb.security.ssl.server.required_options=0

The value of these security options is a bit mask coded as a hexadecimal integer. The meanings of the individual bits is defined in the CORBA Security Service Specification and reproduced here from the Security.idl file:

Table 5.1: Client side supported options

Property with value	Description
jacorb.security.ssl.client.supported_options=20 // EstablishTrustInTarget	This value indicates that the client can use SSL. Actually, this is default SSL behaviors and must always be supported by the client.
jacorb.security.ssl.client.supported_options=40 // EstablishTrustInClient	This makes the client load it's own key/certificate from it's key store, to enable it to authenticate to the server.

Table 5.2: Client side required options

Property with value	Description
jacorb.security.ssl.client.required_options=20 // EstablishTrustInTarget	This enforces SSL to be used.
jacorb.security.ssl.client.required_options=40	This enforces SSL to be used. Actually,

// EstablishTrustInClient	this is no meaningfully value, since in SSL, the client can't force it's own authentication to the server.

```
typedef unsigned short AssociationOptions;

const AssociationOptions NoProtection = 1;
const AssociationOptions Integrity = 2;
const AssociationOptions Confidentiality = 4;
const AssociationOptions DetectReplay = 8;
const AssociationOptions DetectMisordering = 16;
const AssociationOptions EstablishTrustInTarget = 32;
const AssociationOptions EstablishTrustInClient = 64;
const AssociationOptions NoDelegation = 128;
const AssociationOptions SimpleDelegation = 256;
const AssociationOptions CompositeDelegation = 512;
```

Table 5.3: Server side supported options

Property with value	Description
jacorb.security.ssl.server.supported_options=1 // NoProtection	This tells the clients that the server also supports unprotected connections. If No-Protection is set, no required options should be set as well, because they override this value.
jacorb.security.ssl.server.supported_options=20 // EstablishTrustInTarget	This value indicates that the server supports SSL. Actually, this is default SSL behavior and must always be supported by the server. This also makes the server load it's ey/certificate from the key store.
jacorb.security.ssl.server.supported_options=40 // EstablishTrustInClient	This value is ignored, because authenticating the client is either required, or not done at all (the client can't force its own authentication).

Table 5.4: Server side required options

Property with value	Description
jacorb.security.ssl.server.required_options=20 // EstablishTrustInTarget	This enforces SSL to be used.
jacorb.security.ssl.server.required_options=40 // EstablishTrustInClient	This enforces SSL to be used, and will request the client to authenticate. It also will load trusted certificates for the authentication process.

4 附录

4.1 常见问题

4.1.1 如何在同一个机器上启动多个 UCS?

指定不同端口号可以在同一个机器上启动多个 UCS。比如“ucs.bat –port 9000” 或“ucs.bat –p 9000”

4.1.2 为什么我的插件无法执行?

如果在老的 JRE 环境下运行使用比较新的 JDK 编译过的插件，将会遇到版本问题。因此最好保持版本一致。