KDMID1PCFactory
用户手册

深圳市金证科技股份有限公司

项目名称	KDMID1PCFactory 接口转换引擎
类别	文档
子类别	用户手册
摘要	
当前版本	V3.4
日期	2011.7.6
作者	杜玉巍
文档拥有者	杜玉巍
送交人员	用户

修改历史			
版本	日期	修改人	摘要
V1.0	2004/07/12	杜玉巍	1.0 版本
V2.0	2005/06/24	杜玉巍	从 KDMID1PC 中分离出 KDMIDAPI
V3.0	2006/12/11	杜玉巍	增加业务组合。 泛化 KDMIDAPI 为 Adapter。
V3.1	2007/1/15	杜玉巍	增加系统变量、表达式、函数、SQL 语句等功能。
V3.2	2007/1/30	杜玉巍	增加表定义，为 invoke 结果增加名称，增加 output 的 cache 功能
V3.3	2008/1/23	杜玉巍	增加对 switch、while 语句、条件表达式支持。
V3.4	2010/10/28	杜玉巍	增强 invoke 语句功能，支持调用 LBM 及其他程序包中的业务。

目录

1. 引言

1.1 目标

KCBP 接口工厂，又名 KDMID1PCFactory，目标是提供一个符合 KCBPXA 规范，能完成 KCBP 和各种第三方系统(包括柜台系统)之间接口转换，提供跨中间件的业务调用功能，提供系统内或系统间的业务流程编排功能，实现异种系统集成的程序。这个接口程序，不依赖具体的目标系统，可复用，可扩充，既可以在 KCBP 上使用，又可以在 KDMID 上使用，还可以用于任何支持 KCBPXA 规范的系统，也支持与 KCBP 客户端 API 集成。

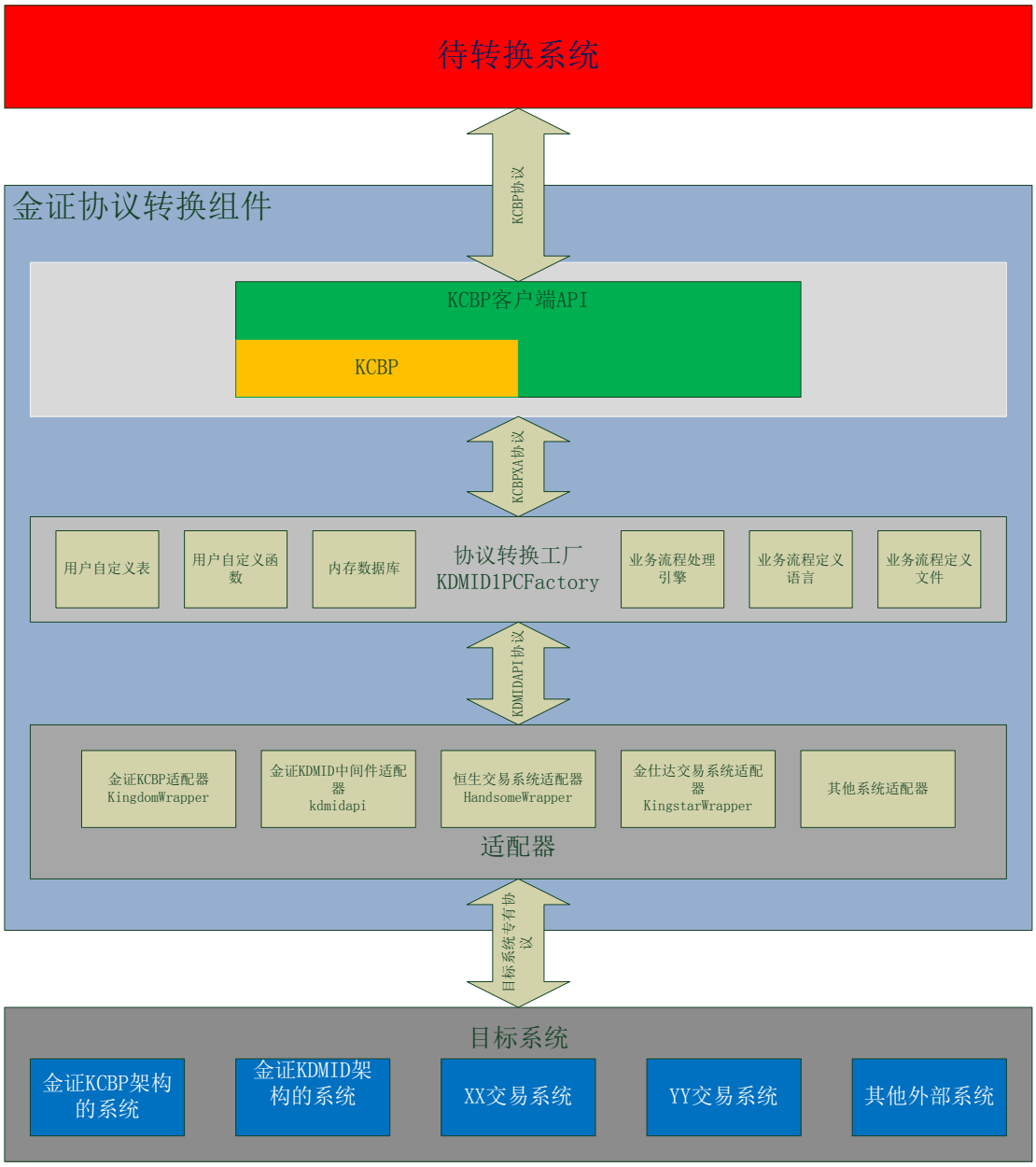
1.2 术语

- KCBP：金证公司开发的通用交易中间件。
- MID：金证公司开发的证券行业专用中间件。
- SPD：英文 Service program definition 的缩写
- KCBPXA：KCBP 操作资源管理器(RM)的规范，它定义了一套操作资源管理器的方法，它扩充了 X/Open 的 XA 标准。
- KDMID1PC：符合 KCBPXA 规范的接口实现。
- KDMID1PCFactory：接口转换工厂。符合 KCBPXA 规范的接口实现，扩充了 KDMID1PC 的功能，将协议转换和接口分开设计，Factory 完成协议转换和接口调用，Adapter 完成业务调用。
- KDMIDAPI：基于 KDMID 的 API 封装的访问柜台业务的接口，Adapter 提供这种形式的接口。后面章节会详细说明 KDMIDAPI。

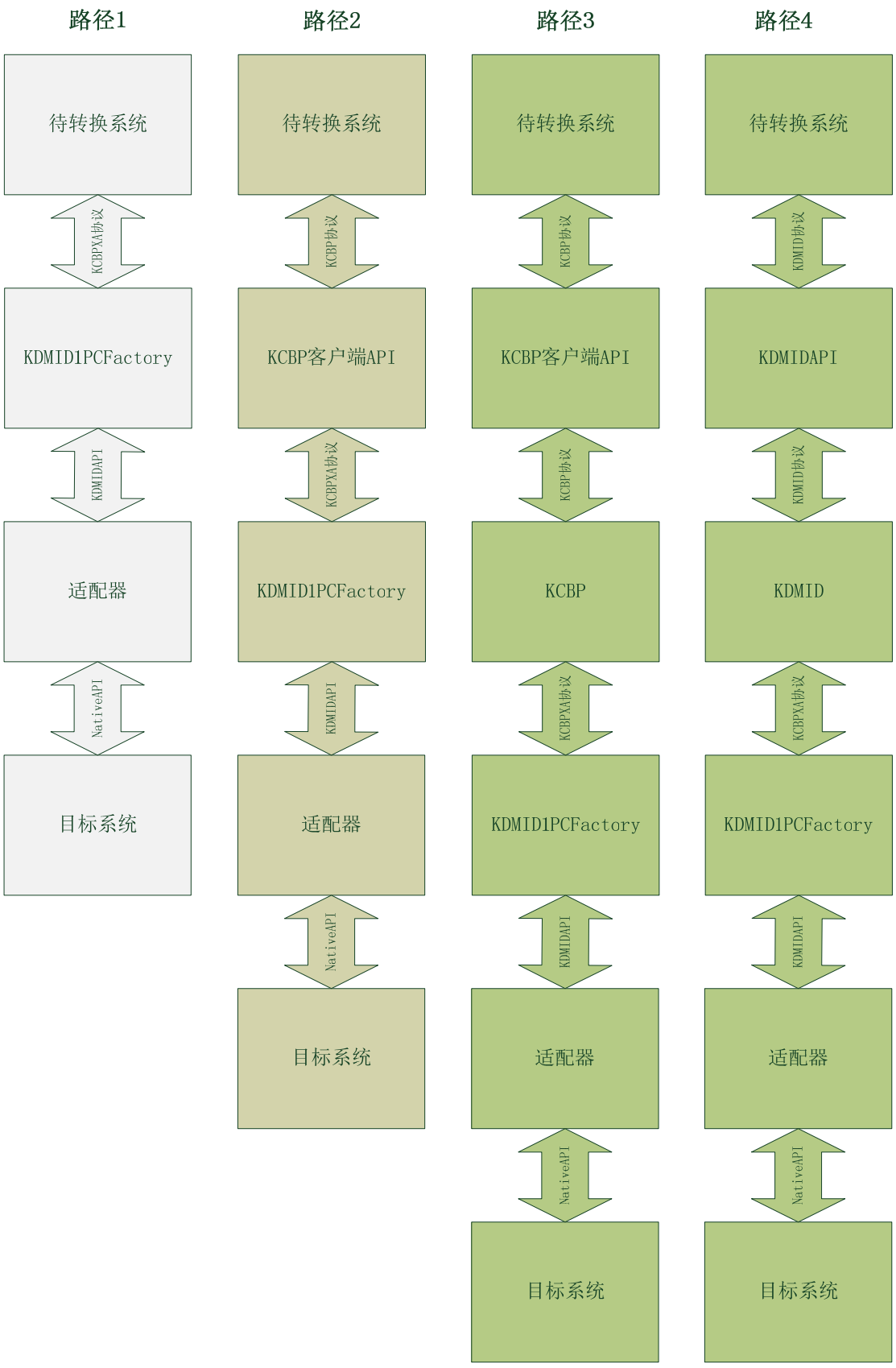
2. 系统结构介绍

KCBP 接口工厂由处理引擎、转换规则、适配器三部分构成。
转换规则是 XML 文件，支持金证 BPEL 定义的接口转换规则，金证 BPEL 可编排业务流程。适配器是依据 KDMIDAPI 接口协议开发的通讯组件。处理引擎解释规则，通过适配器与第三方系统通讯。引擎内嵌金证内存数据库，可支持表定义、表操作、结果缓存、结果筛选与组合功能。

2.1 KCBP接口工厂（协议转换组件）内部构成示意图

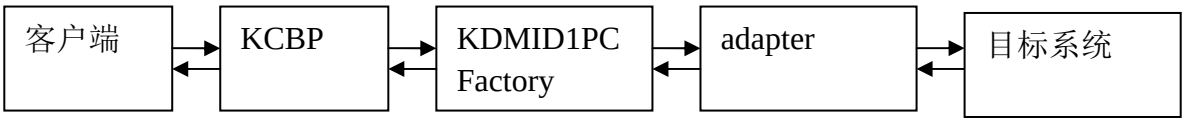


2.2 KCBP协议转换4种处理流程



2.3 处理流程简要描述

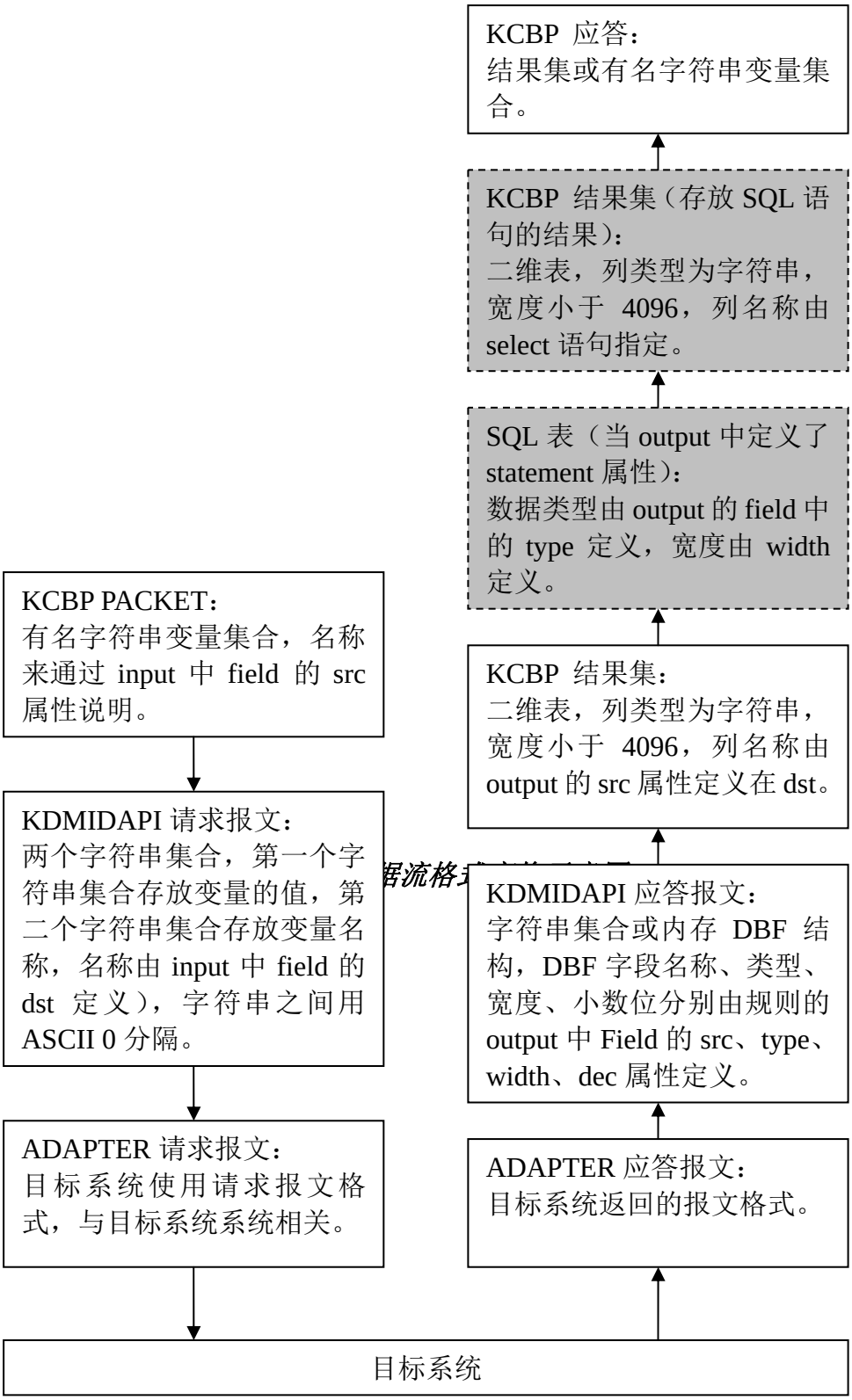
当 KDMID1PCFactory 的输入是 KCBP 报文时，Factory 按照下面流程处理：



KDMID1PCFactory 根据转换规则，完成输入报文格式向 adapter 报文格式(KDMID 格式)的转换，adapter 将转换后的报文进一步组织成目标系统需要的格式，发送给目标系统。Adapter 收到目标系统的应答后，将其转换成 KDMID 的应答报文格式（变量串或二维表），发送给 Factory。Factory 收到应答后，按照出参转换规将报文转换成 KCBP 格式。

当 KDMID1PCFactory 的输入是 KDMID 报文时，Factory 先将其转换成 KCBP 的报文格式，变量名称取自转换规则的入参定义；然后 Factory 再按照前面的流程进行处理，一直到得到 KCBP 格式的应答，再将 KCBP 格式的应答转换成 KDMID 格式的应答。实际上，KDMID1PCFactory 处理 KDMID 格式的报文比处理 KCBP 格式的报文，在入参处理前增加了一个报文转换环节，出参处理后增加了一个报文转换环节。增加的入参转换环节，完成了变量命名处理（变量名称来自转换规则的入参定义）。这个转换，既保证了后续代码及逻辑的复用，又为流程组合提供了基础。

2.3.1.1 KCBP到目标系统数据流格式变换示意图



3. 金证BPEL语法说明

BPEL 用于定义转换规则。 转换规则文件, 也可以理解 BPEL 程序包，每个业务转换规则，可包含变量、表、函数、业务处理逻辑等内容。例子见后面章节--金证新一代 win 柜台协议转 KB32 柜台协议例子。

3.1 一个最简单规则

以下规则只定义接口消息转换方法, 不涉及流程、表、字典、函数、结果筛选等复杂内容。

```
<message src="410301" dst="20102905">
  <input format="!table">
    <field src="inputtype" dst="1" dict="inputtype_in"/>
    <field src="inputid" dst="2"/>
    <field src="trdpwd" dst="3"/>
    <field src="mode" dst="4" default="2"/>
    <field src="netaddr" dst="5"/>
  </input>
  <output format="table">
    <field src="jysdm" dst="market" dict="market_out"/>
    <field src="gddm" dst="secuid"/>
    <field src="gdxm" dst="name"/>
    <field src="zjzh" dst="fundid"/>
    <field src="yybdm" dst="orgid"/>
  </output>
</message>
```

3.2 保留字

业务转换规在 message 元素中定义。

message 的属性 src 表示源业务名称；dst 属性表示目标业务名称，组合业务 dst 属性为空串；indicator 属性只用于金仕达接口，表示在多包返回时，第一包中说明后续行数的字段位置。

input 描述输入变量转换规则集，output 描述输出变量转换规则集。field 描述单个变量转换规则。sequence 描述业务组合，invoke 调用业务。发送请求时，KDMID1PCFactory 将 input 的 src 转换成 dst，发送给目标系统。接收应答时，KDMID1PCFactory 将目标系统的返回作为 output 的 src，转换成 KCBP 系统的 dst。

input 和 output 属性 format 值含义：table 表示入参 src 格式是 2 维表，!table 表示入参 src 格式不是二维表。format 值还可以是数字，0 表示 src 和 dst 都不是二维表，1 表示 src 是二维表；2 表示 dst 是二维表，相当于!table；3 表示 src 和

dst 都是二维表，等同 table。

output 属性 statement 存放 SQL 语句集，SQL 语句作用在 output 结果集和 input 变量之上，output 结果集存放在@output 表中，input 变量存放在@input 表中，SQL 语句可以完成对结果集的筛选、统计、排序等操作。@output 表中，列的数据类型采用 field 中的 type 属性值，如果没有定义 type 属性，则列类型取缺省值 char，宽度由 width 属性定义（如果没有定义 width，宽度缺省 32 字节）。注意，如果需要使用 sql 的聚集函数（如 sum）对某些列计算时，需要将列定义成数值类型（如 int,double）。statement 中可以包含多个 SQL 语句，每个 SQL 语句之间以分号;间隔，多个 SELECT 的结果纵向合并返回。处理@input 和@output 之外，还提供了一个 dual 表，满足语法需要（参考 Oracle 的语法）。使用 SQL 语句时，DATE、CHAR 等数据库用到的保留字不能作为列名使用。

output 的属性 cache 表示是否将结果集保存到内存数据库的表中,yes 保存，no 不保存,lifetime 表示是保存内容的有效期，以秒为单位。如果保存结果到内存数据库中，那么以后的业务调用，在内容有效期内，不会向目标系统发请求，直接从内存表返回数据。内存表的名称是 output_消息的 src 名称，如 output_20102900。

switch 语句用来流程控制，允许从一组分支中只选择一个活动分支。switch 由 case 元素定义的一个或多个条件分支的有序列表组成，后面可跟也可以不跟一个 otherwise 分支。以 case 分支的出现顺序检查，第一个条件是 true 的分支被选择并被作为被执行的活动。如果有条件的分支都未被选择，那么 otherwise 分支将被选择。condition 是条件表达式，兼容 C++语法格式。

```
<switch>
<case condition="jysdm=='0'">
  <invoke message="20102900" input="jysdm='0'" return="yes"/>
</case>
<case condition="jysdm=='1'">
  <invoke message="20102900" input="jysdm='1'" return="yes"/>
</case>
<otherwise>
  <invoke message="20102900" input="jysdm=' '" return="yes"/>
</ otherwise >
</switch>
```

<while>指定反复执行一个活动，直到某个成功条件被满足为止。

```
<while condition="bool-expr" >
      activity
</while>
```

execute 语句用来执行 SQL 语句。SQL 语句一般用来处理多个中间结果集，生成 output 结果集,中间结果集的名称与@input 类似，以@开始，用法举例如下：

```
<invoke message="20102900" input="jysdm='0'" output="@result1"/>
<invoke message="20102900" input="jysdm='1'" output="@result2"/>
<execute sql="insert into @output(jysdm,jysjc) select jysdm,jysjc from @result1; insert into @output(jysdm,jysjc) select jysdm,jysjc from @result2"/>
```

注意：当需要使用聚合语句处理输入、中间、输出结果集时，请定义列的类

型，如不定义数据类型，引擎将列当作字符串处理，而字符串难以进行聚合操作。

3.3 field属性说明

src 定义转换引擎的输入名称(不可缺少)，内容格式是编号、名称、或名称#编号，如: jysdm#1。

以@开始的 input 变量表示系统变量，系统变量@messagesrc 和@messagedst 表示输入的请求编号和目标请求编号，@netaddress 取本机网卡地址，@errorcode 错误码，@errormsg 错误信息，其它来自 XA 的 option 中，目前包括：

userid,password,srcnodeid,destnodeid,opcode,timeout,secretlevel,rule,concurrence,locktimeout,key,keepaliveinterval,connecttimeout,sendqueue,receivequeue,backupip,backupport,trace,tradesystem

trace 的值是 yes 或 no，如果是 yes,记录请求和应答数据，系统性能会下降。

当连接金证新一代 UNIX 版系统时，tradesystem= KDNGUNIX。

如果这些不能满足要求，那么需要修改 KDMID1PCFactory 程序。

对于来自 KCBP 的 input，src 属性要求配置请求变量的名称，编号可不配置。

对于来自 MID 的 input，src 属性要求配置变量的顺序（从 1 开始），名称需要配置。

当 KCBP 上使用传统接口时，从兼容 KDMID 的规则角度考虑，建议名称和编号都配置，这样可以增强规则的适应性，减少编写规则的工作量。

dst 定义转换引擎的输出名称(不可缺少)，内容格式是编号、名称、或名称#编号，如:jysdm#1，金正 3.2 柜台和金仕达柜台输入参数的 dst 属性可以只用编号，恒生柜台必须是名称#编号格式。

当 input 变量转向 KDMID、金仕达系统时，dst 属性要求配置请求的编号,名称可不配置。

当 input 变量转向恒生、金证新一代/UNIX 时，dst 属性要求配置请求的名称,编号可不配置。

dict 定义语义转换用到的字典

default 定义缺省值

type 定义数据类型,包括 int,char,numeric,double,bool,short 等。

width 定义数据宽度

dec 定义小数点后位数

type,width,dec 这三项内容在 KCBP 与其它接口转换时会用到，并且 SQL 语句也可能需要这些内容，配置此项可以增强规则的复用性，减少 KDMID 转换规则编写的工作量，因此建议配置此项。

3.4 invoke属性说明

return="yes"表示结果返回给前端；return="no"表示结果不返回给前端，输出的结果作为下一个请求的输入参数，如果输出结果集有多行，取第一行中的内容，如果结果变量名称与输入变量有重复，则用结果中变量的值替代输入变量的值。

input="jysdm= '0 ',hbdm=@input.hbdm" 对入参赋值，其中，变量值前面可以使用入参的名称或前面的 invoke 返回结果集名称。

output="resultname"结果集的名称，输出结果按名称保存。

whenerror="continue"表示当调用失败时继续后面的调用，whenerror="stop"表示当调用失败时停止处理流程，直接返回错误给前端。缺省动作是 stop。

message="20102900"，表示调用本程序包中的 20102900 逻辑。message 的全格式"[PackageName.]ProgramName"，其中 PackageName 是限定词，是可选项，当 PackageName 未声明时，表示调用本 Package 中的 ProgramName。在 KCBP 上，每个 KDMID1PCFactory XA 定义项都是一个 Package, 其中包含一组服务和业务逻辑定义；一个 KCBP 上所有 LBM 的集合，也是一个 Package。一个 Package 中的 invoke 语句，除了可以调用本 Package 中的服务外，还可以调用另一个 Package 中的服务以及宿主 KCBP 上的 LBM 服务。宿主 KCBP 的 LBM Package 名称是"this"，其他 KDMID1PCFactory XA Package 名称为该 XA 名称。比如，message="20102900"、message="this.20102900"、message="kdmid.20102900"都是符合语法要求的定义项。**注意，当前版本引擎，invoke 调用 LBM 或其它 package 业务时，在当前逻辑范围内，输入/出变量直接使用，不做报文转换。**

注意，output 属性的优先级大于 return 属性的优先级。当 output 属性不为空时，结果集按 output 的名称保存，后续调用可以按名称引用其中的字段。当 output 属性没有定义或为空时，结果集按照 return 声明的方式处理。

3.5 字典

字典用来转换语义。
字典定义例子：

```
<dict name="inputtype_in">
    <entry name="1" value="1"/>
    <entry name="2" value="0"/>
    <entry name="D" value="3"/>
    <entry name="H" value="2"/>
    <entry name="Z" value="Z"/>
    <entry name="K" value="C"/>
</dict>
```

这个字典的名称是 inputtype_in，有 6 个词条，定义了新一代 Windows 版登陆方式与 KB32 登录方式的对应关系，如新一代系统登录方式 K 表示磁卡，对应 KB32 的登录方式 C。

Factory 有一个保留字典 errorcode，定义不同系统之间的错误号对应关系。

3.6 表

表用来定义一个二维表，这个二维表可以被转换规则中的 SQL 语句使用。
表定义例子：

```
<table name="jys" key="jysdm">
```

```
    <row jysdm="0" jysjc="深圳A" jysbs="SZA" hbdm="0" gddmcd="10" nbgddmcd="10" zqdmcd="6" nbzqdmcd="6" jymmcd="6"/>
    <row jysdm="1" jysjc="上海A" jysbs="SHA" hbdm="0" gddmcd="10" nbgddmcd="10" zqdmcd="6" nbzqdmcd="6" jymmcd="6"/>
    <row jysdm="2" jysjc="深圳B" jysbs="SZB" hbdm="1" gddmcd="10" nbgddmcd="10" zqdmcd="6" nbzqdmcd="6" jymmcd="6"/>
    <row jysdm="3" jysjc="上海B" jysbs="SHB" hbdm="2" gddmcd="10" nbgddmcd="10" zqdmcd="6" nbzqdmcd="6" jymmcd="6"/>
</table>
```

key 属性表示索引字段名称，目前只支持一个字段。key 可以省略。

表使用例子：

```
<message src="20102900" dst="">
  <input format="!table">
    <field src="jysdm#1" dst="jysdm"/>
    <field src="hbdm#2" dst="hbdm"/>
  </input>
  <output format="table" statement="select * from jys">
    <field src="jysdm" dst="jysdm" type="C" width="1" dec="0"/>
    <field src="jysjc" dst="jysjc" type="C" width="6" dec="0"/>
    <field src="hbdm" dst="hbdm" type="C" width="1" dec="0"/>
    <field src="gddmcd" dst="gddmcd" type="N" width="2" dec="0"/>
    <field src="nbgddmcd" dst="nbgddmcd" type="N" width="2" dec="0"/>
    <field src="zqdmcd" dst="zqdmcd" type="N" width="2" dec="0"/>
    <field src="nbzqdmcd" dst="nbzqdmcd" type="N" width="2" dec="0"/>
    <field src="jysbs" dst="jysbs" type="C" width="4" dec="0"/>
    <field src="jymmcd" dst="jymmcd" type="N" width="2" dec="0"/>
  </output>
</message>
```

3.7 函数

函数用来处理 XML 难于表示的数据转换，比如密码加密。函数在 DLL 中实现，在转换规则定义文件中声明之后就可以使用。
函数编写需要遵循一定的规范。目前，KDMID1PCFactory 使用的函数，入参个数>=0 个，参数类型是字符串或整形，返回值是一个动态 Buffer。

3.7.1 函数定义

```
<function name="encryptw" path="NGWTools.dll" entrypoint="KDEncodeW">
    <return name="cipherpassword" type="string"/>
    <argument>
        <argv name="key" type="string"/>
```

```
        <argv name="plainpassword" type="string"/>
    </argument>
</function>
```

3.7.2 函数使用

20102905.BEX 参数

```
GETREQ
  1 @khbslx
  2 @khbs
  3 @jymm
  4 @mode
  5 @wldz
ENDGETREQ

<message src="20102905" dst="410301">
    <input format="!table">
        <field src="funcid#-1" dst="funcid" default="410301"/>
        <field src="custid#-1" dst="custid"/>
        <field src="custorgid#-1" dst="custorgid"/>
        <field src="jymm#3=encryptw(funcid, jymm)" dst="trdpwd"/>
        <field src="wldz#5" dst="netaddr"/>
        <field src="@srcnodeid" dst="orgid"/>
        <field src="@opcode" dst="operway"/>
        <field src="ext#-1" dst="ext" default="0"/>
        <field src="khbslx#1" dst="inputtype" dict="khbslx"/>
        <field src="khbs#2" dst="inputid"/>
    </input>
    <output format="table">
        <field src="market" dst="jysdm" dict="market"/>
        <field src="market" dst="jysjc" dict="marketname"/>
        <field src="secuid" dst="gddm" />
        <field src="name" dst="gdxm"/>
        <field src="fundid" dst="zjzh"/>
        <field src="hbdm" dst="hbdm"/>
    </output>
</message>
```

其中，转换 Jymm 时，调用函数进行了加密处理。

3.7.3 函数编写

下面的 VC++代码实现了新一代 Windows 密码加密功能。

```
#include "KDEncodeCli.h"
#pragma comment(lib, "KDEncodeCli.lib")
```

```
#pragma comment(linker, "/export:KDEncodeW=_KDEncodeW@8")
extern "C" __declspec(dllexport) char * WINAPI KDEncodeW(char *szKey, char *szPlain)
{
    char szCipher[64];
    char *pReturn;

    memset(szCipher, 0, sizeof(szCipher));

    KDEncode(KDCOMPLEX_ENCODE, (unsigned char *)szPlain, strlen(szPlain),
        (unsigned char *)szCipher, sizeof(szCipher) - 1, szKey, strlen(szKey));
    pReturn = (char *)malloc(strlen(szCipher) + 1);
    if(pReturn != NULL)
    {
        strcpy(pReturn, szCipher);
    }
    return pReturn;
}
```

这个函数，返回了一个动态分配的 Buffer，其中存放了返回值，这个 Buffer 会被 KDMID1PCFactory 释放。
注意：
这段代码编译时，Debug 版编译选项设置/MDd，Release 版编译选项设置/MD，如果过不设置这个编译选项，KDMIF1PCFactory 释放内存时会抛异常。
返回的 Buffer 不能用 new 申请。

3.8 业务组合sequence例子

```
<message src="20112900" dst="">
  <sequence>
    <invoke message="20102900" return="yes" input="jysdm=3"/>
    <invoke message="20102900" return="yes" input="jysdm=2"/>
    <invoke message="20102900" return="yes" input="jysdm=1"/>
    <invoke message="20102900" return="yes" input="jysdm=0"/>
  </sequence>
</message>
```

3.9 编写和调试注意事项

定义规则之后，需要用 IE 打开规则文件，检查 XML 文件语法的正确性。
程序运行时，可使用 DbgView 察看程序的输出信息，注意，这些运行信息不会输出的中间件屏幕上，只能通过 DbgView 工具察看。

4. KCBP上KDMID1PCFactory配置

4.1 安装KCBP

1 安装 KCBP 新安装包
2 安装。如果升级已前的 KCBP，安装时不要选配置文件。安装后将附件解压到 kcbp 的 bin 目录。如果调试恒生系统,regsvr32 hscommx.dll,运行 hscommxsetup 配置通讯参数.
3 安装后在图形管理器中设置 KCBP License。

4.2 配置XA

4.2.1 恒生接口

<xa entry="KCBP_KDMID_FACTORY" name="kdmid" switchloadfile="KDMID1PCFactory.dll" xaclose="" xaopen="adapter=handsomewrapper.dll,rule=ngw2hs.xml,timeout=30" xaoption="" xaserial="all_operation"/>

4.2.2 金仕达接口

<xa entry="KCBP_KDMID_FACTORY" name="kdmid" switchloadfile="KDMID1PCFactory.dll" xaclose="" xaopen="ip=192.168.40.65,port=9000,adapter=kingstarwrapper.dll,rule=ngw2ks.xml, connecttimeout=5,timeout=30" xaoption="" xaserial="all_operation"/>

4.2.3 金证3.2接口

<xa entry="KCBP_KDMID_FACTORY" name="kdmid" switchloadfile="KDMID1PCFactory.dll" xaclose="" xaopen="ip=192.168.40.65,port=28946,adapter=kdmidapi.dll,rule=ngw2mid.xml,connecttimeout=5,userid=9999,password=3Mc+w9uU5lM=,timeout=30" xaoption="" xaserial="all_operation"/>
其中参数含义参见 KCBP 用户手册 KDMID1PC 配置说明，注意,在 KDMID1PCFactory 中，option 不配参数，参数通过 open 设置。

用户可自定义接口适配器，具体方法见《KDMID1PCFactory 适配器开发指南》。
关于授权注意事项：
在 KCBP 上使用组件时，KCBP 授权号即组件授权号，授权中必须开通组件功能才能正常运行。如果单独使用 KDMID1PCFactory, 在 option 选项中需要增加授权号, 如:license=adadfaasdfasdfasfasdfas, 否则组件无法正常使用。

4.3 配置program

将待转发的业务配置为：
type="deputy",xa="kdmid"
其中 kdmid 可用户自定义。

5. 金证新一代win柜台协议转KB32柜台协议例子

例子中的命令，通过 kcbpcp 输入。

5.1 字典定义

```
<dict name="market_in">
  <entry name="1" value="1"/>
  <entry name="2" value="0"/>
  <entry name="D" value="3"/>
  <entry name="H" value="2"/>
</dict>

<dict name="market_out">
  <entry name="0" value="2"/>
  <entry name="1" value="1"/>
  <entry name="2" value="H"/>
  <entry name="3" value="D"/>
</dict>

<dict name="inputtype_in">
  <entry name="1" value="1"/>
  <entry name="2" value="0"/>
  <entry name="D" value="3"/>
  <entry name="H" value="2"/>
  <entry name="Z" value="Z"/>
  <entry name="K" value="C"/>
</dict>

<dict name="moneytype_out">
  <entry name="0" value="0"/>
  <entry name="1" value="2"/>
  <entry name="2" value="1"/>
</dict>
```

5.2 登录

funcid:410301,custid:,custorgid:0,trdpwd:888888,netaddr:123456789ABC,orgid:0000,operway:7,ext:0,inputtype:Z,inputid:1

```
<message src="410301" dst="20102905">
```

```
  <input format="!table">
    <field src="inputtype" dst="1" dict="inputtype_in"/>
    <field src="inputid" dst="2"/>
    <field src="trdpwd" dst="3"/>
    <field src="mode" dst="4" default="2"/>
    <field src="netaddr" dst="5"/>
  </input>
  <output format="table">
    <field src="jysdm" dst="market" dict="market_out"/>
    <field src="gddm" dst="secuid"/>
    <field src="gdxm" dst="name"/>
    <field src="zjzh" dst="fundid"/>
    <field src="yybdm" dst="orgid"/>
  </output>
</message>
```

inputtype 的语义使用 inputtype_in 字典转换，如果输入 0，输出转换成 2。
注意，字典的处理顺序在缺省值之后。如果变量值不在字典词条范围内，则变量值会被设成空串""。例如，新一代支持下面的输入方式，而 KB32 不支持：
‘B’表示以银行帐户登录　-登录标识为银行帐户
‘C’表示以客户代码登录　-登录标识为客户代码（含代理人代码）
‘X’表示代理人登录　　-登录标识为代理人
‘N’表示用股东内码登录　-登录标识为股东内码
当输入方式为这些值时，经过字典转换，inputtype 被设置为空串。

5.3 委托

5.3.1 上海委托

funcid:410411,custid:,custorgid:0,trdpwd:888888,netaddr:123456789ABC,orgid:0000,operway:7,ext:0,market:1,secuid:A000000001,fundid:,stkcode:600000,bsflag:B,price:9.01,qty:100,ordergroup:-1,bankcode:,remark:test

5.3.2 深圳委托

funcid:410411,custid:,custorgid:0,trdpwd:888888,netaddr:123456789ABC,orgid:0000,operway:7,ext:0,market:2,secuid:0000000001,fundid:,stkcode:000001,bsflag:B,price:9.01,qty:100,ordergroup:-1,bankcode:,remark:test

5.3.3 规则定义

```
<message src="410411" dst="20103900">
  <input format="!table">
    <field src="market" dst="1" dict="market_in"/>
    <field src="secuid" dst="2"/>
    <field src="bsflag" dst="3"/>
```

```
      <field src="stkcode" dst="4"/>
      <field src="qty" dst="5"/>
      <field src="price" dst="6"/>
      <field src="remark" dst="7"/>
      <field src="netaddr" dst="8"/>
   </input>
   <output format="!table">
      <field src="1" dst="ordersno"/>
   </output>
</message>
```

5.4 当日委托查询

funcid:410510,custid:,custorgid:0,trdpwd:888888,netaddr:123456789ABC,orgid:0000,operway:7,ext:0,market:,fundid:1,secuid:,stkcode:,ordersno:,bankcode:,qryflag:1,count:100,poststr:

```
<message src="410510" dst="20102913">
   <input format="!table">
      <field src="khbslx" dst="1" default="Z"/>
      <field src="fundid" dst="2"/>
      <field src="market" dst="3" dict="market_in"/>
      <field src="stkcode" dst="4"/>
      <field src="ordersno" dst="5"/>
      <field src="gdms" dst="6" default="2"/>
      <field src="cdms" dst="7" default="0"/>
   </input>
   <output format="table">
      <field src="wtrq" dst="orderdate"/>
      <field src="wtsj" dst="opertime"/>
      <field src="jysdm" dst="market" dict="market_out"/>
      <field src="gddm" dst="secuid"/>
      <field src="zjzh" dst="fundid"/>
      <field src="hbdm" dst="moneytype" dict="moneytype_out"/>
      <field src="htxh" dst="ordersno"/>
      <field src="zqdm" dst="stkcode"/>
      <field src="zqmc" dst="stkname"/>
      <field src="mmlb" dst="bsflag"/>
      <field src="wtsl" dst="orderqty"/>
      <field src="wtjg" dst="orderprice"/>
      <field src="zjfss" dst="orderfrzamt"/>
      <field src="sbsj" dst="reporttime"/>
      <field src="cdbz" dst="cancelflag"/>
```

```
      <field src="cjsl" dst="matchqty"/>
      <field src="cdsl" dst="cancelqty"/>
      <field src="cjje" dst="matchamt"/>
      <field src="fzdm" dst="orgid"/>
      <!--field src="poststr" dst="poststr"/>
      <field src="custid" dst="custid"/>
      <field src="custname" dst="custname"/>
      <field src="orderid" dst="orderid"/>
      <field src="orderstatus" dst="orderstatus"/>
      <field src="seat" dst="seat"/-->
   </output>
</message>
```

5.5 撤单

5.5.1 命令格式

funcid:410413,custid:,custorgid:0,trdpwd:888888,netaddr:123456789ABC,orgid:0000,operway:7,ext:0,orderdate:,fundid:0000000001,ordersno:000001

5.5.2 规则定义

```
<message src="S410413" dst="20103901">
   <input format="!table">
      <field src="gdms" dst="1" default="0"/>
      <field src="market" dst="2"/>
      <field src="secuid" dst="3"/>
      <field src="ordersno" dst="4"/>
      <field src="bzxx" dst="5"/>
      <field src="netaddr" dst="6"/>
   </input>
   <output format="!table">
      <field src="1" dst="msgok"/>
   </output>
</message>

<message src="410413" dst="">
     <sequence>
          <invoke message="410510" return="no" input=""/>
          <invoke message="S410413" return="yes" input=""/>
     </sequence>
</message>
```

5.5.3 说明

KB32 撤单需要股东代码，新一代/Win 版撤单使用资金帐号，因此，需要使

用组合规则，先根据资金帐号＋合同号查委托，获得股东代码，然后再撤单。

这是一个典型的业务组合调用。第一个业务调用，410510 按委托合同号查询委托单信息，返回的 secuid 股东代码、market 市场代码作为第二个调用 S410413 的入参，S410413 的结果输出给前端。

注意，410510 返回的结果是二维表，如果存在该合同号的委托单时，会有一行返回，否则没有返回。return="no"属性，告诉 KDMID1PCFactory，把这个二维表列名转换成变量名称，第一行各列的值转换成变量值，丢弃第一行之外的其余行；如果列名与输入变量重合，则将变量值赋为列值。

还有一点要特别提醒，关于涉及到资金、股份发生变化的业务，要保证安全性。由于 KB32 委托/撤单没有输入要求交易密码，安全依赖于登录 session，当外围不能提供 Session 功能时，如使用 JZAPI 开发的网上交易处理机，不同客户复用连接，这时会有安全隐患。要消除这个隐患，可以通过组合业务实现，先验证用户的交易密码，成功后再调用实际业务。这项功能，要求前端在委托/撤单时，输入参数中要包含交易密码，如果没有输入交易密码，要修改程序提供密码，毕竟，保证系统安全是最重要的。