PMX 用户编程指南
（V1.0）

中国证券登记结算有限责任公司上海分公司

2009 年 4 月

版本修订历史

日期	版本	说明	作者
2009/04/02	1.0	发布	李栋良

目　录

第一章 概述

一 前言

1. 文档所涉内容及适用对象

本文档是中国证券登记结算有限责任公司上海分公司（以下简称中国结算上海分公司）PROP 消息交换系统（以下简称 PMX）的用户编程指南，用于指引用户如何通过 PMX 实现 PROP 用户间的消息级交换。

本文档的适用对象为：通过 PMX 进行消息交换的系统设计人员、软件开发人员。假定读者在使用本文档前已了解《PROP 数据交换系统-用户接口说明书》中的相关内容。

2. 术语释义

- PROP：参与人远程操作平台（Participant Remote Operating Platform），是中国结算上海分公司面向各类市场参与人推出的技术平台。
- PROP 用户：接入 PROP 系统的用户，一个市场参与人可能拥有一个或多个 PROP 用户，PROP 用户通过 PROP 用户名唯一标识。
- PROP 操作员：一个 PROP 用户可以包括多个 PROP 操作员，PROP 操作员通过 PROP 操作员号标识，用户号在该 PROP 用户下唯一。
- PROP 服务域、服务名及服务类型：PROP 服务通过服务域、服务名和服务类型的组合唯一标识。服务域（16 位）用以标识不同的服务提供者，如中国结算上海分公司的服务域为“SSCCRC”；服务名（16 位）用以标识用一服务域下的不同服务大类，如证券账户类服务的服务名为“ZHZLGLXT”；服务类型（2 位，其中‘FF’为系统保留值）则用以表示具体的服务类型，如证券账户开户的服务类型为“08”。

二 基于PMX的应用开发步骤

1. 在开发/测试环境中安装及配置PMX

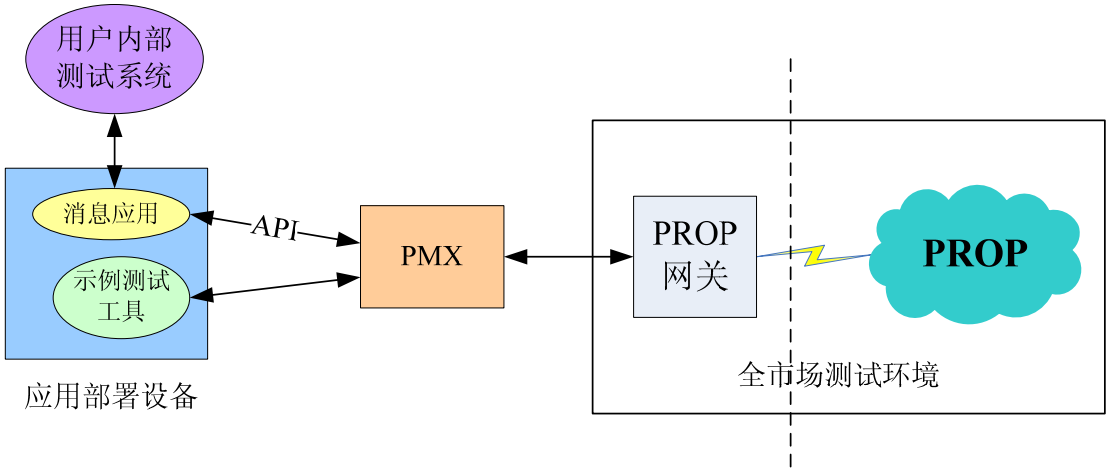
（1）确定开发/测试环境具备以下条件：

- 通过该环境能正常接入我公司的全市场技术测试环境。
- 确保开发/测试环境与生产系统隔离。
- 全市场测试环境专用的 PROP 网关不应作为 PMX 及开发应用的部署设备。

（2）从我公司网站www.chinaclear.cn或PROP公告板上下载最新的PMX软件。

（3）根据《PMX 操作手册》在开发/测试环境中安装 PMX。

（4）配置 PMX 的运行参数，具体步骤可参见《PMX 操作手册》。建议参照下图搭建应用的开发测试环境：



PMX应用开发部署示例图

说明：

- PMX 部署设备可以和应用部署设备合用。
- 示例测试工具由中国结算上海分公司提供，分服务端和客户端两个示例测试工具，可分别下载安装在应用部署设备上。关于示例测试工具的安装及配置，请参见相关技术文档。
- 全市场技术测试环境的服务时间并非 7x24，具体时间安排请参见我公司的相关通知。

2. 开发应用程序

（1）确定消息交换双方的应用模式、PROP 服务名、消息格式定义和处理流程。所谓应用模式是指谁是应用的请求方和应答方。对于请求方，应用程序应采用主动请求模式编程，而应答方则采用被动服务模式编程。PROP 服务名则由双方商定，如第三方存管服务可定义为“DSFCG”。服务名大小写敏感，用户在定义和使用服务名时应统一采用大写的方式。消息格式和处理流程由双方商定，PMX 对此完全透明。

（2）使用 PMX-API 完成应用程序的编码，相关内容请参见第二章。

（3）通过示例测试工具验证应用程序在消息交换方面的正确性：

- 应用程序采用主动请求模式，即是请求的发起方，则通过服务端示例测试工具进行测试。

- ✧ 应用程序采用被动服务模式，即是请求的处理方，则通过客户端示例测试工具进行测试。

3. 通过全市场测试环境与对方用户联调

消息交换双方在全市场测试环境服务时间内进行联合调试。在双方第一次调试之前请联系中国结算上海分公司，需由我公司相关人员在测试前进行权限及服务信息的配置。

在调试过程中如遇到 PROP 方面的技术问题，请致电 PROP 服务热线咨询。

4. 上线准备

在系统正式上线前，必须完成如下准备工作：

（1）双方已就上线时点协商一致，并准备了相应的回退方案。

（2）双方已向中国结算上海分公司申请了新增服务的注册权限及客户使用该服务的权限，并已由中国结算上海分公司完成了上述权限配置的调整。

（3）双方已在生产环境中部署完成 PMX，并正确设置了相应的配置信息。

（4）双方已在生产网络中开放了 PMX 侦听所需的端口，对于服务提供方，还需要开放服务提供方侦听的端口。关于网络端口的详细说明，可参见第二章。

（5）服务请求方可以通过 PMX 软件的 PING 功能验证到服务提供方的消息可达性。

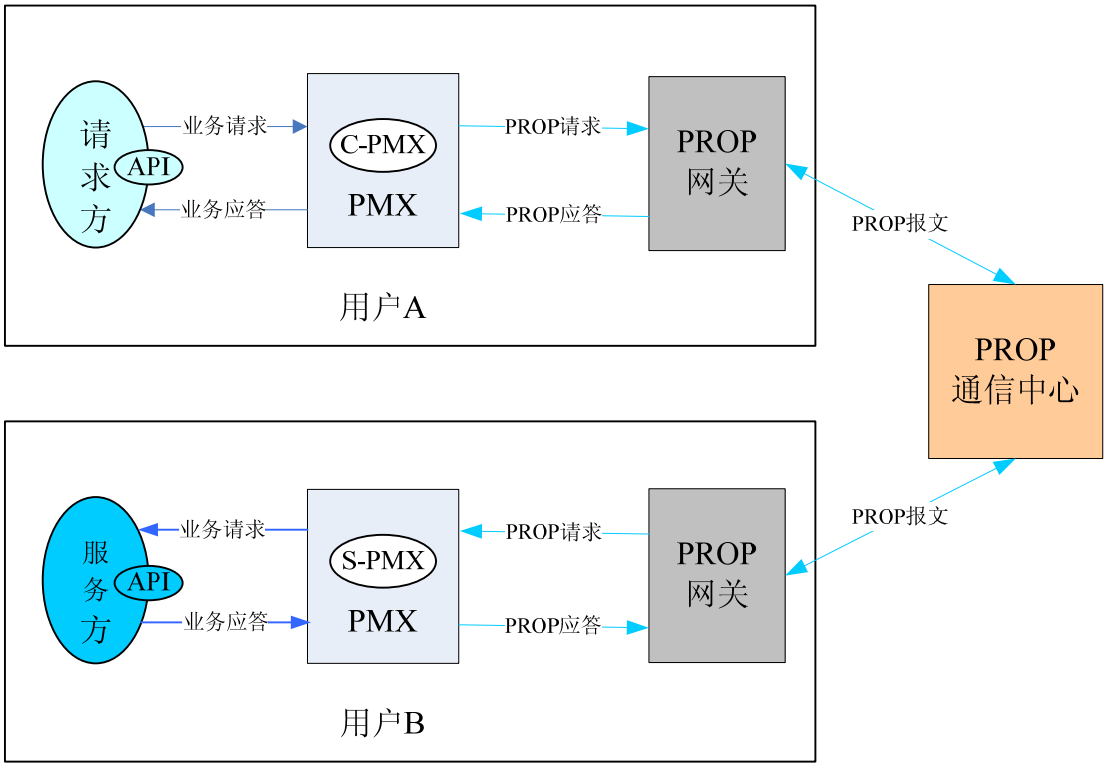
5. 上线及运行跟踪

在双方商定的上线时点完成应用上线后，双方通过查询类消息的交换验证上线的正确性。在系统正式投入生产运行后，双方进行运行跟踪，以及时发现运行过程中可能发生的异常。如遇到 PROP 方面的相关问题，请及时与中国结算上海分公司联系。

第二章 基于PMX的消息应用设计

一 基于PMX的消息交换原理

通过 PMX 在两个 PROP 用户之间进行消息交换，其消息交换过程如下图所示：



基于PMX的消息交换示意图

上图中 C-PMX 和 S-PMX 为 PMX 的两个功能模块，可以单独运行。对于服务请求方，必须启动 C-PMX 模块，而对于服务应答方，则必须启动 S-PMX 模块。

消息的交换过程说明如下：

（1）用户 A 的请求方应用进程向 PMX 中的 C-PMX 进程发起连接请求，并建立连接。

（2）请求方应用进程发送请求消息，请求消息中指定了该消息的接收者及消息对应的服务名。

（3）C-PMX 对请求消息以 PROP 交易请求的格式封装，并通过 PROP 网关将之发送到 PROP 系统的通信中心（PROP 通信中心由中国结算上海分公司负责运行和维护）。

（4）PROP 通信中心根据 PROP 请求的服务域、服务名寻找相应的服务提供者，并将 PROP 请求发送到用户 B 的网关，最终到达用户 B 的 S-PMX。

（5）S-PMX 对 PROP 交易请求解封，还原为业务请求。S-PMX 向用户 B 的服务方应用进程发起连接请求，在建立连接后将业务请求发送给对方。

（6）服务方应用进程收到业务请求后进行相应的业务处理，并将生成的业务应答消息通过第（5）步中建立的 socket 连接发送给 S-PMX。

（7）S-PMX 对应答消息以 PROP 交易应答的格式封装，并通过 PROP 网关将之发送到 PROP 系统的通信中心。在完成应答消息的发送后，S-PMX 将关闭与服务方进程之间的连接。

（8）PROP 通信中心将 PROP 应答报文发送给用户 A 的网关并最终到达 C-PMX。

（9）C-PMX 对 PROP 应答报文解封，还原为业务应答消息。C-PMX 通过第（1）步中建立的 socket 连接将应答消息发送给请求方应用进程。

（10）请求方应用进程接收应答消息。

（11）如果请求方采用长连接模式，重复步骤（2）-（10），否则关闭与 C-PMX 之间的连接。

从上述消息交换过程可知，在消息交换双方之间并未建立会话，整个消息交换机制实际是由 PROP 系统进行消息的转发。消息交换双方可以在应用层面建立会话机制，即由应用负责实现双方间的会话机制。

二 基于PMX的消息交换特点

1、请求应答模式

消息交换双方采用传统的 C/S 一应一答模式，服务请求方只能发起业务请求并且接收服务方返回的应答；服务方只能被动接收请求方发起的业务请求，并返回应答到请求方，服务方不能向请求方发送请求消息。

此外，对于服务请求方而言，对方返回的应答消息必然从原路返回，即从一个 PMX 连接上发出请求消息后，其对应的应答消息只能从该连接上接收。如果请求方在发送请求消息后关闭与 C-PMX 之间的连接，则其应答消息必然被丢弃，也无法从别的连接上收到应答消息。上述特点也意味着，在某一连接上不会收到应返回给别的连接的应答消息。

2、应用与 PMX 之间的连接模式

（1）服务请求方与 C-PMX 之间的连接模式：支持长连接模式和短连接模式。请求方采用长连接模式时可以连续在一个连接上发起一个或者多个请求但必须等到所有的请求均有应答时才能关闭连接，否则该应答将丢失；当采用短连接模式时一个连接上只能处理一个请求一个应答，处理完应答后必须关闭连接，当有一个新的请求时必须重新建

立一个连接。

（2）S-PMX 与服务应答方之间的连接模式：目前仅支持短连接模式。因此服务应答方在返回一条应答消息后不可能接收到另一条请求消息。

3、消息交换的高强度安全性

基于 PMX 进行 PROP 用户间的消息交换，具有很高强度的安全性，具体表现在：

（1）PROP 系统本身具备的高强度安全特性：

- ✧ 与本方交换消息的用户必然是一个合法的 PROP 用户，且通过身份认证机制保证了双方用户不可能被冒用。
- ✧ 通过 PROP 系统交换的数据通过 SSL 链路传输。

（2）PMX 是 PROP 系统的客户端扩充模块，其本身也具备了许多安全特征，具体表现在：

- ✧ 用户与 PMX 之间设计了令牌机制，确保非法应用无法通过 PMX 传输数据，包括非法应用无法冒充客户发送业务请求，也无法冒充服务应答方对外提供伪装的服务。
- ✧ 服务提供方可以指定本服务通道的服务对象，即可以拒绝对指定用户外的 PROP 用户提供服务。

4、消息交换的限制条件

（1）PROP 系统并不暂存用户间的消息，如果消息的指定接收方在消息到达时并不在线，消息将被丢弃。

（2）消息在传递过程中可能因各种原因被丢弃，底层系统不提供消息的重发机制，因此应用程序应在上层实现消息的重发机制。

（3）服务方在收到一个请求消息后，最多只能返回一条应答消息。因此消息交换双方必须采用一应一答模式，系统不支持应用之间的一应多答模式。

三 PMX应用的两种编程模式

1、主动请求模式

服务请求方是主动发出请求消息的一方，它必须使用主动请求模式的编程方式。主动请求模式不能取代被动服务模式，也就是说主动请求模式接收不到来自其它用户发送的请求消息，但能收到其它用户返回的应答消息。

2、被动服务模式

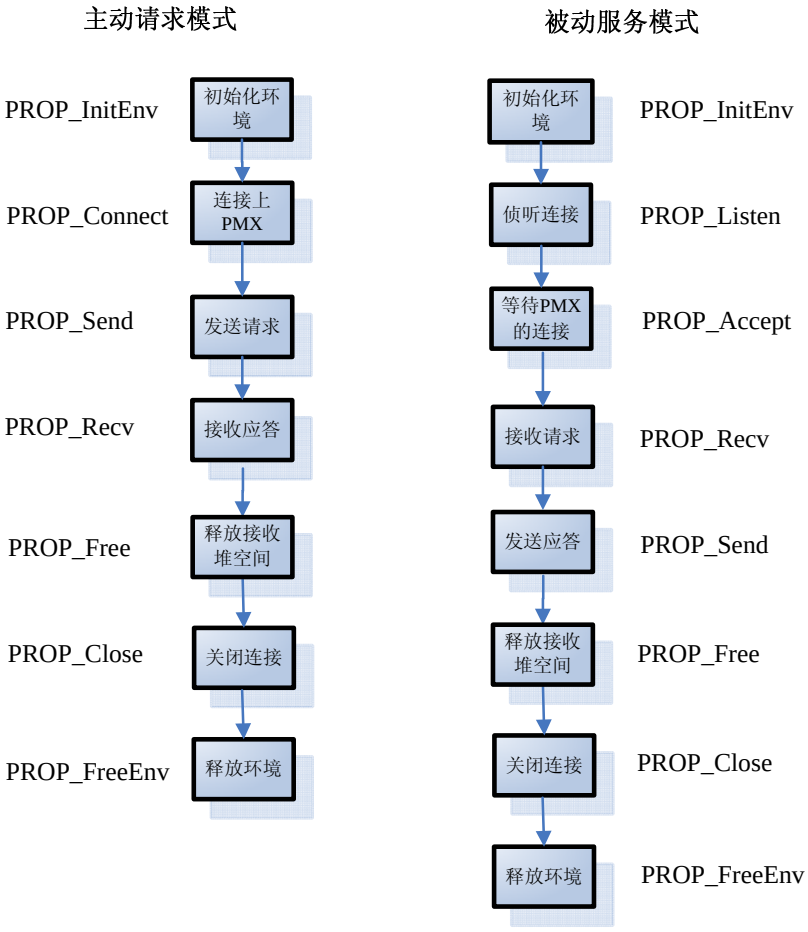
服务应答方是收到请求消息并返回应答消息的一方，它必须采用被动服务模式编程。被动服务模式下只能接收来自其它用户发送的业务请求消息，并可以返回应答消息。在此模式下不能主动向对方发出请求消息。

3、两种模式的关系

消息交换双方在消息交换过程中必须分别采用主动请求模式合被动服务模式，互相配合才能完成正常的实时消息交换。

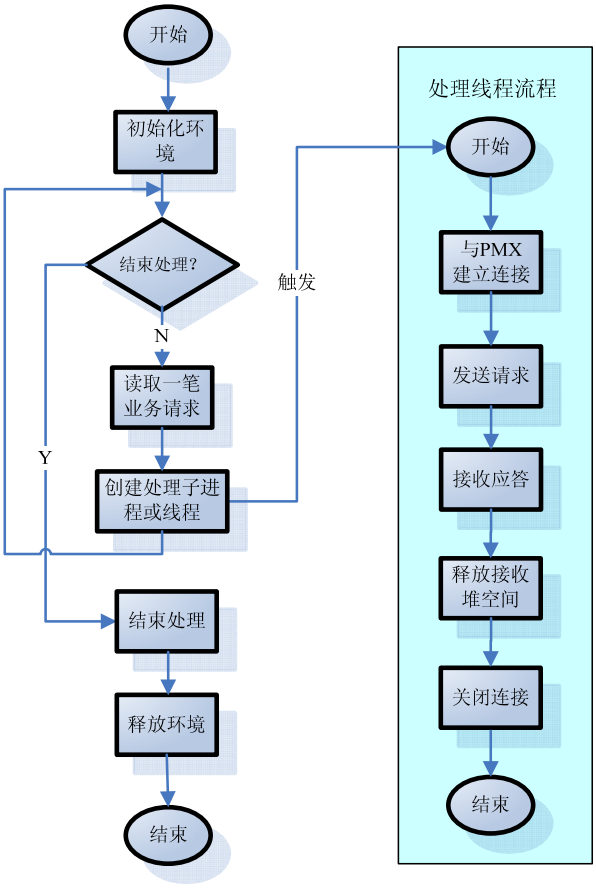
四 消息交换程序概要处理流程

消息交换双方为了进行一次消息交换，应分别按主动请求模式和被动服务模式进行处理。就一次完整的消息交换过程而言，两种模式的概要处理流程如下图所示：

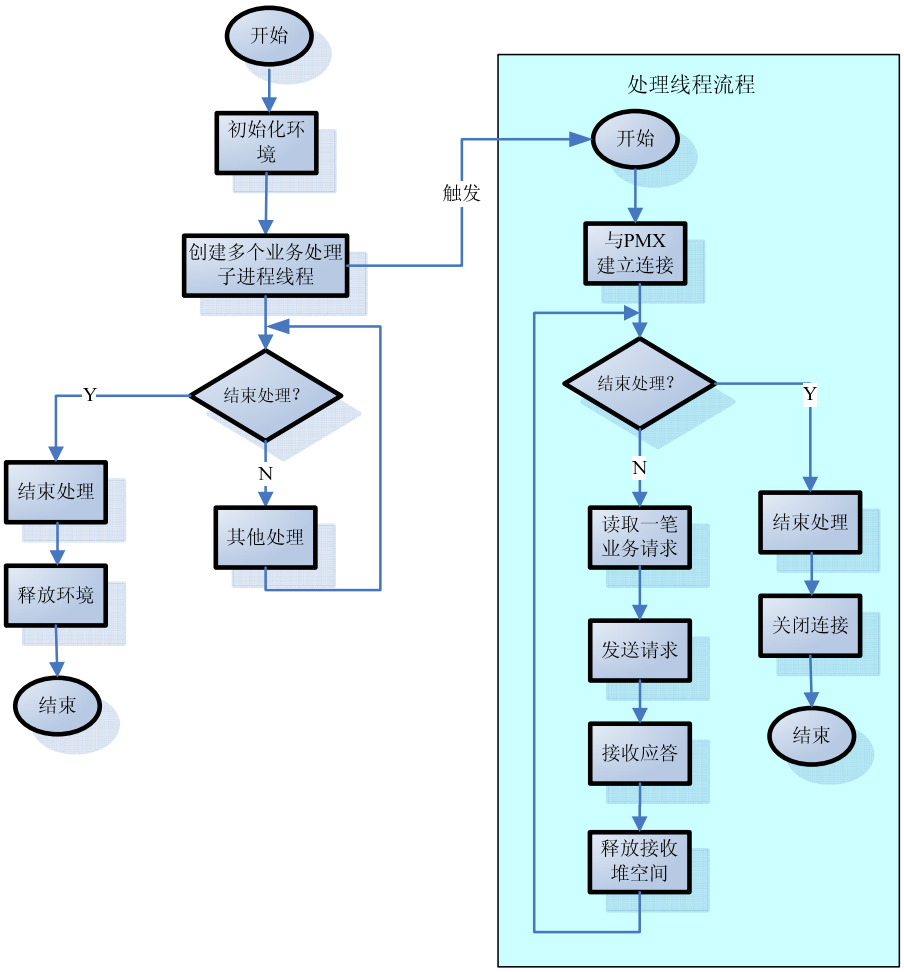


消息交换概要处理流程

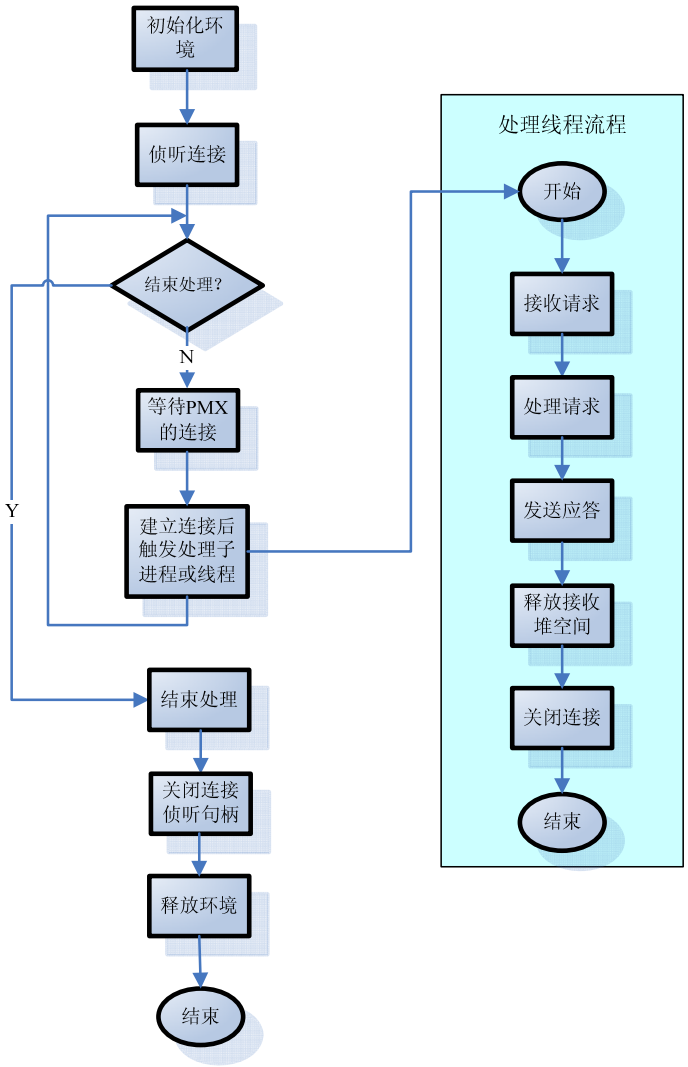
上述处理流程描述了从消息交换过程而言的一些必要的函数调用流程。在实际的应用设计中，设计人员还需要考虑很多相关的问题，如长连接/短连接的选择、同步/异步的选择、并发处理设计等。下面分别列出了服务请求方短连接处理模式、长连接处理模式和服务应答方典型处理流程图。



服务请求方短连接处理模式流程图



服务请求方长连接处理模式流程图



服务应答方典型处理流程

五 消息寻址

消息发送者在每一次的消息交换中均要指定一个唯一的消息接收者以及相应的服务名称。消息用户名以 PROP 用户名标识，而服务名称则由中国结算上海分公司统一编制和管理，用来唯一标识一类服务。

PROP 用户名的长度为 8 位，大小写敏感。PROP 用户名在系统内唯一。部分 PROP 用户名在尾部包含了‘*’串，消息应用在指定这些用户时应去除尾部的‘*’串。如某 PROP 用户名为“Q12345**”，在指定该用户时应填写为“Q12345”。

PROP 服务名称则由双方商定，如第三方存管服务可定义为“DSFCG”。服务名大小写敏感，用户在定义和使用服务名时应统一采用大写的方式。在确定 PROP 服务名称后，需要由中国结算上海分公司维护相应的服务权限。

对于同一类服务，服务提供方用户可以同时存在多个服务提供者。服务请求方发送的请求消息可能被发送到任一服务提供者（但该服务提供者必须指定允许接收来自该用户的请求），这取决于 PMX 的调度策略。

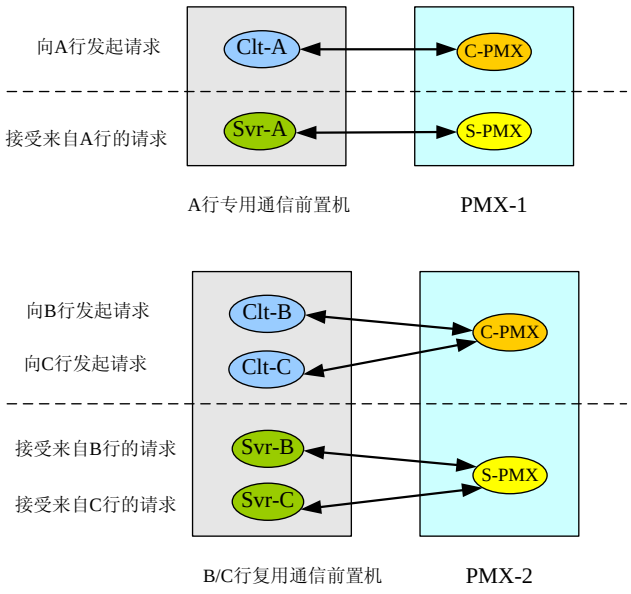
六 PMX的部署及服务提供策略

PMX 是 PROP 用户之间进行消息交换的核心功能模块，它由 C-PMX 和 S-PMX 两个子模块组成。这两个子模块可以单独运行。对于服务请求方，必须启动 C-PMX 模块，而对于服务应答方，则必须启动 S-PMX 模块。

无论是 C-PMX 还是 S-PMX 模块，其处理性能均不低于 80 笔/秒。如果上述性能无法满足用户的要求，用户可以部署多个 PMX 模块。在这种情况下，用户应用进程可同时连接多个 PMX 模块，以达到负载均衡和消除单点故障的目的。

对于服务提供方而言，还涉及到服务提供策略问题。为了提高某类服务的并发处理能力，服务方往往需要提供该服务的多个服务通道。对于某一特定的服务通道，可以指定专门的服务对象。用户应该根据实际的消息交换量和消息交换特点，制定合适的服务提供策略。

下面以客户资金第三方存管银证数据交换为例，给出某券商一种可行的 PMX 部署和服务提供策略。该券商与 A、B、C 三家银行进行消息交换，其中与 A 行的消息交换量较大，与另两家银行的消息交换量较小。该券商部署了 2 个 PMX 模块，并提供了对 A 行服务的专用通道，如下图所示：



PMX部署及服务提供策略示例图

与 PMX 部署相关的网络端口如下表所示：

端口用途	端口号	说明
C-PMX 侦听端口	6666，可配置	PMX 部署机对服务请求方的开放端口
S-PMX 侦听端口	6668，可配置	PMX 部署机对服务应答方的开放端口
服务侦听端口	服务方应用自定义	服务应答方对 PMX 部署机的开放的端口

第三章 基于PMX的消息应用编程

一 PMX应用环境的初始化和释放

通过 PMX 进行消息的交换之前，消息交换双方必须首先调用 PROP_InitEnv 函数以初始化 PMX 应用环境。而在程序的退出处理中，应用程序则必须调用释放 PROP_FreeEnv 函数以释放 PMX 应用环境相关的资源。下面分别介绍这两个函数的作用及使用方法。

1、初始化 PMX 应用环境

初始化 PMX 应用环境的目的主要有两个，一是初始化相应的系统资源，包括初始化 socket 环境、申请环境句柄相关的内存资源等。更重要的目的是通过调用 PROP_InitEnv 函数，应用程序向 PMX 系统进行了必要的身份验证（通过 PROP 操作员密码验证），并获取了代表身份的消息交换令牌，供以后的消息交换中使用。

PROP_InitEnv 函数的主要输入参数包括：

- ✧ PMX 所部署设备的 IP 地址
- ✧ C-PMX 或 S-PMX 的侦听端口号
- ✧ PROP 系统操作员名
- ✧ PROP 系统操作员密码

上述参数中，操作员名可以是任一合法的 PROP 操作员，而端口号参数则取决于应用程序所处的模式。对于发起交易请求的应用程序，应传入 C-PMX 的侦听端口号；对于接收交易请求的应用程序，则应传入 S-PMX 的侦听端口号。这两个端口号必须和 PMX 上配置的实际端口号一致。

该函数还有一个输出参数 pErrorInfo，它用于存储在函数调用失败时返回的出错信息，PMX-API 中的大部分函数均有该参数。如果用户可以关心调用失败的原因，则应负责申请该结构占用的空间，并传输该结构的指针。用户也可以传入空指针，但在函数调用失败时错误信息将被丢弃。

调用本函数成功后，将返回 PMX 环境句柄。环境句柄将作为后续的函数调用的输入参数。对服务请求方而言，在后续的 PROP_Connect 函数及 PROP_FreeEnv 函数将传入此参数；对服务应答方而言，在后续的 PROP_Listen 函数及 PROP_FreeEnv 函数将传入此参数。

一般而言，对一个应用程序而言，只需要调用一次初始化 PMX 应用环境函数，而且调用往往是在主进程中发生的，动态产生的线程或子进程无需调用本函数。尽管如此，在实际应用中，主进程可能需要多次调用该函数，因为应用程序可能同时连接了多个 PMX 模块，因而需要针对每个 PMX 分别调用一次。在多次调用本函数时，每次将返回一个新的 PMX 环境句柄，并不会覆盖原有的 PMX 环境句柄。

2、释放 PMX 应用环境句柄

在应用程序结束处理时，必须调用 PROP_Free 函数以释放 PMX 应用环境句柄相关的系统资源。主进程调用了多少次初始化函数，则应调用同样次数的释放 PMX 环境函数。

请编程人员注意的是，本函数绝非只是简单的堆内存释放函数，而是进行了一些保证 PMX 长时间正常运作的关键操作，如用户令牌的注销。因此在程序最后调用本函数并非可有可无，否则可能引发 PMX 在长时间运行后发生运行异常。

本函数的参数仅有一个，即在调用 PROP_InitEnv 函数后返回的 PMX 环境句柄。本函数无返回值。

二 PMX连接的建立和关闭

在完成初始化 PMX 应用环境后，服务请求方需要与 C-PMX 建立连接以传输请求消息，而服务应答方需要接受来自 S-PMX 的连接请求以接收对方用户发送的请求消息。PMX 连接的建立是消息得以传递的前提。

1、连接 PMX

连接 PMX 函数 PROP_Connect 函数适用于服务请求方调用，类似于 socket 编程中的 connect 调用。调用该函数时传入在 PROP_InitEnv 函数返回的 PMX 环境句柄，在调用成功时将返回 PMX 连接句柄，该句柄将作为后续的 PROP_Send、PROP_Recv 及 PROP_Close 调用的输入参数。

每次成功调用 PROP_Connect 后均会创建一个与 C-PMX 之间新的连接，该连接持续到程序调用 PROP_Close 函数后才关闭。PROP_Connect 和 PROP_Close 必须一一配对，每调用一次 PROP_Connect 均需要调用一次 PROP_Close 以释放资源，否则将造成内存泄露和 socket 资源泄漏。

2、侦听连接

侦听连接函数 PROP_Listen 函数适用于服务提供方调用，其作用类似于 socket 编程中的 listen 调用，但并不仅限于此。除了在指定的本地 IP 地址和端口上侦听连接外，该函数还可以限定侦听的服务名及服务对象。

PROP_Listen 函数的主要输入参数包括：

- ✧ 调用 PROP_InitEnv 时返回的 PMX 环境句柄
- ✧ 侦听的本地 IP 地址和端口号
- ✧ 侦听的服务名
- ✧ 可使用本服务的 PROP 用户名

上述参数中，侦听的本地 IP 地址为 NULL 时，应用侦听本机的所有地址。侦听的服务名则限定了能接受的服务，服务名不能为空。PROP 用户名则限定了可以使用本服务的用户，该参数为空时表示任何 PROP 用户均可使用本服务。假设某应用指定的服务名为“DSFCG”，用户名指定为“Q12345”，则只有来自 Q12345 用户且消息相关的服务名为“DSFCG”的消息才可能发送给本应用。

调用本函数成功将返回侦听句柄，该句柄将作为后续的 PROP_Accept 和 PROP_Close 调用的输入参数。

调用该函数成功后，PMX 监控软件的服务列表上将显示该服务。因此该函数实际上向 S-PMX 进行了一次服务注册，供后者完成到达消息的路由。因此，应用程序在退出处理中必须调用 PROP_Close 函数，并将侦听句柄作为其输入参数，这样侦听句柄将被关闭，服务将被注销。

3、接受 PMX 连接

服务应答方在调用 PROP_Listen 函数成功后，应继续调用 PROP_Accept 函数以接受来自 S-PMX 的连接请求。该函数的输入参数包括侦听句柄和超时时间。调用本函数后应用程序将阻塞在调用点，直到接受到 S-PMX 的连接请求并成功建立一个新的连接，或因超时而返回。

如果 PROP_Accept 函数调用成功，在应用程序和 S-PMX 之间就建立了一个新的连接。在传统的客户服务器模式下，服务端一般会在建立新的连接后创建新的子进程或线程以专门处理客户的请求。在多进程模式下，主进程在调用 PROP_Accept 成功后应立即关闭返回的

连接句柄，否则将引发因打开文件描述符太多而导致主进程无法正常运行。

当服务应答方进程因调用 PROP_Accept 函数而阻塞时，除了 S-PMX 外，其它应用试图连接该侦听端口则无法成功建立连接，因为该函数还对连接它的进程进行了身份验证。

4、关闭连接句柄或侦听句柄

应用程序在完成消息传输后，应调用 PROP_Close 函数关闭连接句柄，以释放 socket 资源。此外，应答方在退出前也应调用本函数关闭侦听句柄。

三 消息的发送

在应用程序与 PMX 之间建立连接后，请求方和应答方就可以通过调用 PROP_Send 函数来完成实时消息的发送。该函数的主要输入参数包括：连接句柄、消息描述体、消息正文、消息长度和超时值等。其中消息描述体用来定义被发送消息的关键要素，任何一条消息均与一个消息描述体相关。下表是对消息描述体结构各字段的详细说明。

名称	类型	说明
strMsgType	C2	消息类型，“00”表示请求，“01”表示应答
strSender	C8	消息的发送方用户名，在消息发送时无需填写
strReceiver	C8	消息的接收方用户名，必须填写
strService	C16	服务名，必须填写
strServiceType	C2	服务类型，一般固定填“00”
strMessageId	C10	请求消息发送方生成的消息序号
strTransId	C16	PROP 通迅中心生成的唯一交易流水号
strClientTransId	C10	PMX 生成的唯一交易流水号
strDateTime	C14	消息发送或接收的时间，无需填写

在上述字段中，有三个字段是与一条消息相关的消息序号或流水号，这三个字段的含义和用法是：

- ✧ strMessageId：请求消息发送方自用的消息编号，其编码规则由请求方自行确定，其唯一性由用户保证；应答方在返回应答消息时，应原样返回该字段的值。
- ✧ strClientTransId：由 PMX 生成的交易流水号，该流水号在本 PMX 中唯一。应答方在返回应答消息时，必须原样返回该字段的值，否则请求方无法收到该消息。
- ✧ strTransId：由 PROP 通迅中心生成的唯一交易流水号，在整个 PROP 系统中唯一。应答方在返回应答消息时，必须原样返回该字段的值，否则请求方无法收到该消息。

应用程序正确填写 PMX 消息描述体中的内容是进行消息发送的关键。对于消息长度参数，如果消息长度小于消息正文的实际长度，则发送函数将只发送指定长度的消息内容。超时值的单位是毫秒，建议该值为-1，即无限等待。

函数调用成功将返回 0，超时时返回-9，出错时则返回-1。基于本系统消息交换的特点，发送函数返回成功并不意味着对方一定能接收到该消息，如因对方不在线、系统繁忙等原因可能丢弃应用发送的消息。因此，消息交换双方必须自行建立必要的重发/确认机制，系统无法保证消息的可靠传递。

1、请求消息的发送

请求方在发送请求消息时，应按下表的要求填写 PMX 消息描述体的字段：

名称	填写要求
strMsgType	消息类型，填写为“00”
strSender	消息的发送方用户名，无需填写
strReceiver	消息的接收方用户名，必须填写
strService	服务名，必须填写
strServiceType	服务类型，一般固定填“00”
strMessageId	本方生成的消息序号
strTransId	PROP 通迅中心生成的唯一交易流水号，无需填写
strClientTransId	PMX 生成的唯一交易流水号，无需填写
strDateTime	消息发送或接收的时间，无需填写

应用程序在调用发送函数成功返回后，并不意味着本消息已被对方成功接收。一般的处理逻辑是：发送者继续调用接收函数，如接收函数返回成功，则可以确认请求消息必然被对方成功接收；如果返回失败，则进行请求的重发处理。

在发送请求消息时，应用程序在长连接模式下可采用异步发送模式，即发送一笔请求后，在未收到应答消息的情况下可以继续发送请求消息。由于等待应答消息的返回时间可能较长（一般在 5 秒之内，取决于应答方的处理速度和系统的繁忙程度），因此采用异步发送模式可以提高系统的处理速度。在该模式下需要应用自行匹配请求和应答消息。下面是一段可行的示例伪代码：

```
while(1)
{
    读取一条请求记录;
    iRet=PROP_Send(…);  //发送请求消息
    if (iRet == 0) //发送成功
    {
        iRet=PROP_Recv(…);  //接收应答消息
        if (iRet==0) //接收成功
        {
            匹配请求/应答;
            处理应答记录;
        }
        else if (iRet==-9) //接收超时
        {
            continue; //继续处理
        }
        else
        {
            接收出错处理;
        }
    }
    else
    {
        发送出错处理;
    }
```

}

2、应答消息的发送

应答方在发送应答消息时，应按下表的要求填写 PMX 消息描述体的字段：

名称	填写要求
strMsgType	消息类型，填写为“01”
strSender	消息的发送方用户名，无需填写
strReceiver	接收方用户名，必须是对应请求消息的发起方，必须填写
strService	服务名，原样返回
strServiceType	服务类型，原样返回
strMessageId	请求方消息序号，原样返回
strTransId	PROP 通迅中心生成的唯一交易流水号，原样返回
strClientTransId	PMX 生成的唯一交易流水号，原样返回
strDateTime	消息发送或接收的时间，无需填写

应答方在发送应答消息时，最重要的是必须将三个消息序号（流水号）字段原样返回，否则请求方将无法收到应答。

此外，由于 S-PMX 采用短连接模式，在 S-PMX 和应答方建立一次连接后，实际数据传输只有两次，即 S-PMX 发送请求消息给应答方，应答方返回应答消息给 S-PMX。在这一过程中，S-PMX 作为客户端，将主动关闭双方之间的 socket 连接。因此，为确保在连接关闭前应答消息被 S-PMX 完整接收，建议应答方在发送完应答消息后，不主动关闭连接，而应等待对方关闭连接后再将连接句柄关闭。建议的伪代码如下：

```
while(1)
{
    iRet=PROP_Recv(…);  //接收请求消息
    if (iRet == 0) //接收成功
    {
        处理请求;
        iRet=PROP_Send(…);  //发送应答
        …
    }
    else if (iRet==-2)  //S-PMX 主动关闭连接，正常情况
    {
        正常退出处理;
        return(0);
    }
    else if (iRet==-9)  //接收超时
    {
        超时处理;
        return(-1);
    }
    else //其他错误
    {
        错误处理;
```

```
            return(-1);
        }
    }
    return(0);
```

四 消息的接收

与 PROP_Send 函数相对应的是 PROP_Recv 函数，该函数的主要输入参数包括：连接句柄、接收超时值等，而输出参数则包括消息描述体、接收到的消息地址（双指针）、接收到的消息长度等。函数调用成功是返回 0，超时返回-9，PMX 主动关闭连接时返回-2，其他情况则返回-1。下面说明调用本函数的相关注意事项：

（1）请求方在发送一条请求消息后调用该函数接收应答消息时，一般情况下超时值不宜设置太短。由于 PMX 消息交换的特点，应答返回时间一般在 1-5 秒之内（取决于应答方的处理速度和系统的繁忙程度）。如果函数指定了较小的超时值，函数返回-9 则表明超时，程序应再次接收应答消息。在前文已经述及，应用程序也可能采用异步消息交换模式，在此类情况下，一般会设置较小的超时值，以提高系统的并行处理能力（参见本章第三部分的说明）。

（2）应答方通过调用本函数接收请求消息，由于 S-PMX 和应答方采用了短连接模式，因此建议应答方在接收应答消息时设置超时值为-1，即无限等待。此外，前文也有述及，S-PMX 会主动关闭连接，应答方在发送应答消息后，建议不主动关闭连接，而应再次调用 PROP_Recv 函数（调用函数将在 S-PMX 关闭连接时返回，返回值为-2，此时应用可正常关闭本方连接句柄后退出）。这样做一方面可以确保在连接关闭前应答消息被 S-PMX 完整接收，另一方面则可避免本方出现大量 TIME_WAIT 状态的 TCP 连接的情况。

（3）每次调用本函数接收消息后，API 动态申请了相应的内存空间。应用程序每调用一次 PROP_Recv 函数成功后，必须在使用完毕后相应地调用一次 PROP_Free 以释放内存，否则将造成内存的严重泄漏。

五 辅助函数的应用

在 PMX-API 库中，还提供了一些并非必须使用的辅助函数，包括服务状态侦测函数 PROP_Ping、连接句柄 I/O 状态检测函数 PROP_Select 和获取版本号函数 PROP_GetVersion。下面分别介绍这些函数的使用。

1、服务状态侦测函数

PROP_Ping 是供用户请求方使用的服务状态侦测函数，用于确定对方用户当前是否提供了某指定的服务。该函数的主要参数包括：调用 PROP_Connect 成功后返回的环境句柄、服务方用户名、服务名和超时时间等。应用程序使用该函数一般有两个用途：

（1）监控程序通过周期性地调用该函数，分别监测各消息交换对象的服务提供情况。

（2）请求方应用程序在启动时首先检测对方服务是否提供，如对方服务不在线则给出错误提示或延时再试。

2、连接句柄 I/O 状态检测函数

PROP_Select 函数的作用类似于 select 调用，它向应用程序提供了一种多路 I/O 选择手段。应用程序可能同时建立了多个数据交换通道，然后通过本函数检测各通道的 I/O 状态，决定是否进行后续的读写操作。

本函数的输入参数包括：连接句柄数组、句柄数量、操作类型和超时时间等。连接句柄

数组的成员是由调用 PROP_Connect 或 PORP_Accept 后返回的句柄组成的，成员数量至少为 1 个。操作类型包括：0-检测是否可读、1-检测是否可写和 2-检测是否有错误。函数成功返回时，返回值为某个符合检测条件的句柄在句柄数组中的偏移量（范围为 0~句柄数量-1）。此后，用户可以对该句柄进行相应的读写操作。

3、获取版本号函数

获取版本号函数用于获得 API 的当前版本号。通过调用该函数可以很方便地查询应用程序当前使用的 API 版本，避免出现因 API 版本过低导致程序运行错误。