

PIMR: Руководство программиста
Версия 1.0.0
www.gradsoft.kiev.ua

October 24, 2002

Contents

1	Введение	2
1.1	Свойства сервиса и операции над ним	2
1.2	Динамическая часть	3
2	API репозитария	4
3	Взаимодействие репозитария и объекта-сервиса	6
4	Взаимодействие репозитария и объекта-клиента, использующего сервис	6
5	Изменения	7

1 Введение

PIMR ("репозиторий") предоставляет собой способ повысить доступность и облегчить конфигурирование взаимосвязанных CORBA-сервисов. Репозиторий также можно использовать вместо `nameservice` для получения инициальных объектных ссылок на необходимые сервисы. Репозиторий регистрирует в своей базе данных сервисы и при получении запроса (далее "заявки") - перенаправляет его подходящему сервису. Кроме того репозиторий отслеживает доступность известных ему сервисов и при необходимости делает попытку перезапустить сбойный сервис.

В данной реализации (версия 1.0.0) отслеживание доступности сервисов производится периодически через определённый промежуток времени. Частота проверки настраивается в конфигурационном файле, с помощью команды `checkfreq {число}`. При обнаружении недоступности сервиса производится попытка возобновления его работы. Если сервис не подал сигнал о успешной перезагрузке и остаётся недоступным по истечении таймаута, он помечается как сбойный и более не перезапускается.

Прикладным программистам репозиторий предоставляет API, описываемое с помощью idl-интерфейсов.

1.1 Свойства сервиса и операции над ним

Со статической точки зрения сервис характеризуется набором взаимосвязанных свойств и операций. Некоторые свойства могут быть не установлены, поэтому они представлены множеством {флаг,свойство[, свойство...]}, в котором флаг определяет определённость свойств. Если флаг установлен - группа свойств определена и ими можно пользоваться. Если соответствующий флаг не установлен - значения свойств этой группы не имеют определённого смысла (хотя их и можно прочесть).

Свойства сервиса:

- IOR ссылка на реализацию сервиса. Невозможно работать с объектом не имея на него ссылки. Механизм IOR(Interoperability Object References) представляет собой один из видов объектных ссылок. IOR определяется в ПОР - адаптации GIOP, использующей стек протоколов TCP/IP. Например `corbaloc::127.0.0.1:22000/NameService`. (Используется для перенаправления запросов и для проверки работоспособности сервисов помимо callback-интерфейса)
- ID IDL-интерфейса объекта. Например `IDL:omg.org/CosNaming/NamingContext:1.0`. (Используется при привязке CORBA-объекта к серванту-приманке)
- IOR ссылка на CheckCallback интерфейс, реализуемый объектом-сервисом. (необязательно)
- `restart_flag`. Флаг, контролирующий наличие свойств: команда перезапуска сервиса, хост/порт. Флаг устанавливается если установлена команда

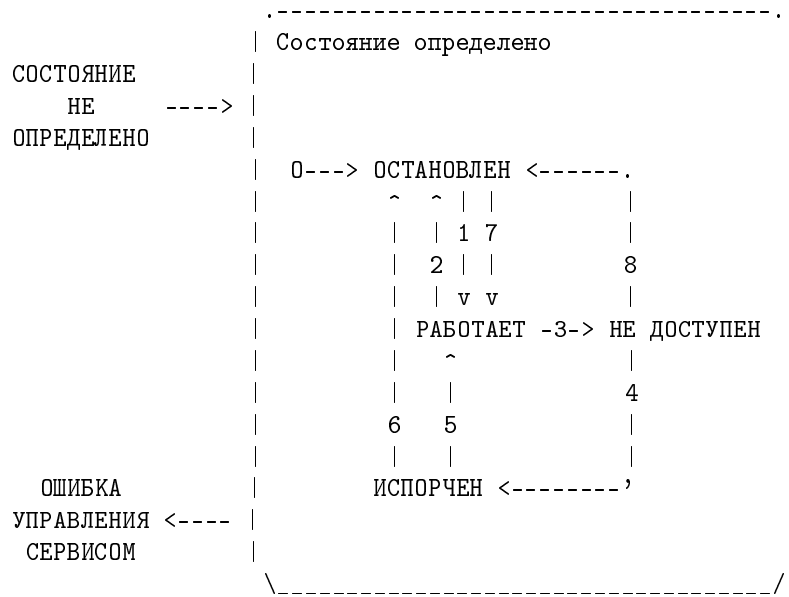


Рисунок 1: Диаграмма основных состояний

запуска сервиса в случае его сбоя ("команда перезапуска", условимся называть запуск сервиса в таких обстоятельствах "перезапуском"). Установка флага означает что сервис должен быть перезапущен если он окажется недоступным. Другое название флага - "всегда доступен".

- Хост (и порт), на котором запущен сервер. Например `127.0.0.1:22000` или `localhost:22000`.
- Команда перезапуска сервиса. Например `myservice -parameter1 parameter2`.

1.2 Динамическая часть

Диаграмма основных состояний сервиса изображена на рисунке 1. На ней изображены основные состояния и возможные переходы между ними:

- 1 Запуск сервиса (переход через промежуточное состояние)
- 2 Остановка сервиса (переход через промежуточное состояние)
- 3 Работающий сервис стал недоступен
- 4 Недоступный сервис должен быть доступен в этот момент
- 5 Перезапуск сервиса (переход через промежуточное состояние)
- 6 Остановка сбойного сервиса (переход через промежуточное состояние)

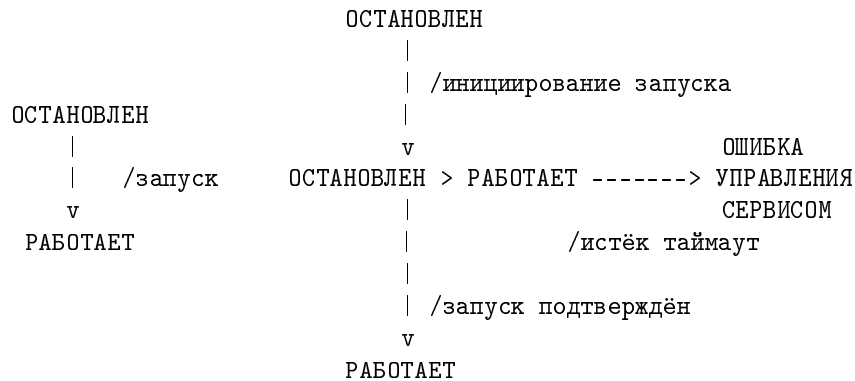


Рисунок 2: Переход через промежуточное состояние

7 Сервис запустили

8 Сервис остановили

При этом переход 8 запрещен для сервисов с флагом "всегда доступен". Сервисы со сброшенным флагом "всегда доступен" при обнаружении их недоступности считаются остановленными.

Диаграмма содержит два разных перехода из состояния ОСТАНОВЛЕН в РАБОТАЕТ и это не ошибка так как переходы 1,2,5,6 производятся через дополнительно введённые состояния (2).

2 API репозитария

API репозитария представлено в файле pimr.idl:

```

#pragma prefix "gradsoft.kiev.ua"

module PIMR
{

interface CheckCallback;

interface Repository
{

exception UnknownServiceName {};

/**
 * must be call by client, which know about PIMR after
 * startup.

```

```

    **/
void activated(in string service, in Object instance)
    raises(UnknownServiceName);

/**
 * we can call checker for service
 **/
void setCheck(in string service, in CheckCallback callback)
    raises(UnknownServiceName);

};

enum ServiceStatus { Unknown, NotRunning, Correct };

//
interface CheckCallback
{
    ServiceStatus    checkStatus(in string service);
};

};

#endif

```

Интерфейс Repository (а также RepositoryAdmin, определённый в файле PIMRAdmin.idl) реализуется CORBA-объектом репозитория, доступным по ссылке:

```
corbaloc::хост_репозитория:порт_репозитория/Repository
```

A CheckCallback предназначен для реализации в сервисе, поддерживающем репозиторий. Задача этого интерфейса - реализовать своеобразный CORBA-ring – проверку доступности сервиса. При этом операции `activated()` и(или) `setCheck()` могут быть вызваны объектом-сервисом при старте. Эти операции сообщают репозиторию о активизации данного объекта-сервиса. Отличие между ними состоит в том, что первая операция сообщает репозиторию информацию о (возможно, изменившемся) расположении объекта-сервиса, а вторая - о расположении объекта, реализующего интерфейс CheckCallback и принадлежащего вызывающему сервису.

Примеры, использующие эти интерфейсы, расположены в инсталляции репозитория в поддиректории `demo\`, ознакомьтесь с ними.

Хотя вызывать их и не обязательно, для полной поддержки репозитория необходимо после запуска сервиса вызвать сначала `activated(myIOR)` а затем `setCheck(callBack)`. Это гарантирует, что репозиторий будет знать о расположении объекта-сервиса и сможет проверить его доступность.

При использовании сервисов, реализующих неполную поддержку репозитория, возможен вариант, когда репозиторий имеет правильную ссылку на сервис

но неправильную ссылку на callback-интерфейс (или наоборот). В таком случае сервис будет считаться недоступным.

3 Взаимодействие репозитория и объекта-сервиса

Взаимодействие репозитория и объекта-сервиса возможно в нескольких вариантах:

- Никакого взаимодействия, за исключением того, что кто-то должен будет внести всю необходимую информацию в репозиторий. Например через конфигурационный файл. Этот вариант можно использовать для сервисов, ничего не знающих о репозитории и это его плюс. Увеличенное время реакции и не такая надёжность, как в других случаях - это минус.
- Сервис после запуска вызывает `activated(*)` но не реализует `CheckCallback` интерфейс.
- Сервис реализует интерфейс `CheckCallback` (и возможно вызывает `setCallback(*)` после запуска, иначе кто-то должен сообщить репозиторию ссылку на `Callback` интерфейс сервиса) но не использует `activated(*)`. В этом случае кто-то должен сообщить репозиторию ссылку на объект-сервис.
- Сервис полностью поддерживает репозиторий в том смысле, что реализует `CheckCallback` интерфейс (и возможно вызывает `setCallback(*)` после запуска, иначе кто-то должен сообщить репозиторию ссылку на него) и вызывает `activated()` после запуска.

Дополнительно в конфигурационном файле можно предоставить репозиторию информацию о перезапуске сервиса. Тогда этот сервис будет иметь установленным флаг `restart_flag`, называющийся ещё "всегда доступен" и репозиторий будет перезапускать его, если обнаружит что он недоступен.

Будьте осторожны при совмещении нескольких вариантов взаимодействия для одного и того же сервиса - возможно возникновение варианта, когда репозиторий имеет правильную ссылку на объект сервиса и неправильную - на его callback-интерфейс. Такой сервис будет недоступным.

4 Взаимодействие репозитория и объекта-клиента, использующего сервис

Предварительные условия: репозиторий запущен. Подразумевается, что сервис зарегистрирован в репозитории, репозиторий получил объектную ссылку на него. Если это так, то сервис будет доступен по ссылке:

```
corbaloc::хост_репозитория:порт_репозитория/имя_необходимого_сервиса
```

Клиент направляет заявку используя этот корбалок. Орб клиента связывается с репозитарием и передаёт заявку. Орб клиента получает:

- Указание перенаправить заявку. Это значит всё прошло хорошо. Указание содержит адрес объекта, а механизм перенаправления реализован в CORBA скрыто для объекта-клиента.
- Исключение OBJECT_NOT_EXIST, что означает либо отсутствие у репозитория записи о сервисе либо то, что сервис остановлен в этот момент. Это также значит что при обработке запроса не произошло никаких ошибок.
- Исключение IMP_LIMIT если у репозитория истёк период ожидания (возможно при извлечении информации или при оперировании сервисом) PS. Возможно TRANSIENT лучше подходит здесь, это исключение может измениться в следующей версии репозитория.
- Исключение OBJ_ADAPTER если сервис испорчен (неправильно сконфигурирован) и поэтому недоступен.
- Системное исключение CORBA с соответствующей семантикой.

Если орб клиента плучает указание перенаправить заявку, то эта (и последующие) заявка отправляется скрыто для объекта-клиента по ссылке, предоставленной репозиторием.

Детали реализации: Орб клиента связывается с репозитарием и передаёт заявку. Репозитарий использует динамический сервант-приманку для ассоциирования с объектом, доступным по ссылкам указанного выше вида.

Поэтому заявка клиента направляется орбом репозитория серванту-приманке, но ее перехватывает интерцептор. Интерцептор извлекает **имя_сервиса**, содержащееся в заявке, затем обращается к внутреннему реестру для получения информации о сервисе. В зависимости от полученной информации интерцептор может произвести определенные операции над сервисом, согласно алгоритму работы репозитория, до истечения таймаута **redirecttimeout**. Затем интерцептор возвращает результат клиенту.

Для лучшего понимания рассмотрите примеры в директории **demo\дистрибутива** репозитория.

5 Изменения

15.08.2002 Создана предварительная версия.