Nimbus分散サービス仕様書

2016年7月14日

目次

[1. 概要 1](#_Toc461467284)

[2. インタフェース設計 7](#_Toc461467285)

[2.1. jp.ossc.nimbus.service.keeplalive.KeepAliveChecker 7](#_Toc461467286)

[2.2. jp.ossc.nimbus.service.keeplalive.KeepAliveListener 7](#_Toc461467287)

[2.3. jp.ossc.nimbus.service.keeplalive.KeepAliveCheckerSelector 8](#_Toc461467288)

[2.4. jp.ossc.nimbus.service.keeplalive.ClusterListener 9](#_Toc461467289)

[2.5. jp.ossc.nimbus.service.proxy.invoker.KeepAliveCheckInvoker 10](#_Toc461467290)

[2.6. jp.ossc.nimbus.service.proxy.RemoteServerInvoker 11](#_Toc461467291)

[3. 実装設計 12](#_Toc461467292)

[3.1. jp.ossc.nimbus.service.keeplaive.ClusterService 12](#_Toc461467293)

[3.2. jp.ossc.nimbus.service.keepalive.KeepAliveChecker 15](#_Toc461467294)

[3.2.1. jp.ossc.nimbus.service.keepalive.jdbc.JDBCKeepAliveCheckerService 15](#_Toc461467295)

[3.2.2. jp.ossc.nimbus.service.keepalive.smtp.SmtpKeepAliveCheckerService 16](#_Toc461467296)

[3.2.3. jp.ossc.nimbus.service.jndi.CachedJndiFinderService 17](#_Toc461467297)

[3.3. jp.ossc.nimbus.service.keepalive.KeepAliveCheckerSelector 18](#_Toc461467298)

[3.3.1. jp.ossc.nimbus.service.keepalive.FirstAvailableKeepAliveCheckerSelectorService 18](#_Toc461467299)

[3.3.2. jp.ossc.nimbus.service.keepalive.RoundRobinKeepAliveCheckerSelectorService 19](#_Toc461467300)

[3.3.3. jp.ossc.nimbus.service.keepalive.ResourceAvailableKeepAliveCheckerSelectorService 20](#_Toc461467301)

[3.4. jp.ossc.nimbus.service.keepalive.ClusterListener 22](#_Toc461467302)

[3.4.1. jp.ossc.nimbus.service.keepalive.DefaultClusterListenerService 22](#_Toc461467303)

[3.5. jp.ossc.nimbus.service.proxy.invoker.KeepAliveCheckInvoker 23](#_Toc461467304)

[3.5.1. jp.ossc.nimbus.service.proxy.RemoteServiceServerService 23](#_Toc461467305)

[3.5.2. jp.ossc.nimbus.service.proxy.invoker.LocalClientMethodCallInvokerService 25](#_Toc461467306)

[3.5.3. jp.ossc.nimbus.service.proxy.invoker.RemoteClientMethodCallInvokerService 26](#_Toc461467307)

[3.6. jp.ossc.nimbus.service.proxy.RemoteServerInvoker 27](#_Toc461467308)

[3.6.1. jp.ossc.nimbus.service.proxy.RemoteServiceServerService 27](#_Toc461467309)

[3.7. jp.ossc.nimbus.service.proxy.RemoteClientService 28](#_Toc461467310)

[3.8. jp.ossc.nimbus.service.aop.Invoker 30](#_Toc461467311)

[3.8.1. jp.ossc.nimbus.service.proxy.invoker.ClusterInvokerService 30](#_Toc461467312)

[3.8.2. jp.ossc.nimbus.service.proxy.invoker.JMXClientRMICallInvokerService 31](#_Toc461467313)

# 概要

分散したリソースを監視及び管理し、透過的に呼び出しを行うためのサービス群です。死活監視やクラスタ制御を行うNimbus KeepAliveサービス群、リモート及びローカルのサービスをプロキシ化して、透過的なアクセスや、機能付加を行うNimbus Proxyサービス群から構成される。

Nimbus KeepAliveサービス群は、個々のリソースの死活監視を行うKeepAliveChecker、複数のKeepAliveCheckerから利用可能なKeepAliveCheckerを選択するKeepAliveCheckerSelector、分散環境でサービスをクラスタ化するClusterServiceで構成される。

死活監視

KeepAliveCheckerSelector

KeepAliveChecker

KeepAliveChecker

KeepAliveChecker

KeepAliveListener

KeepAliveCheckerは、各リソースの死活状態を監視する。監視方法は、監視するリソースとそれに対応するKeepAliveCheckerの実装に依存する。また、KeepAliveListenerを登録することで、監視対象のリソースの状態変化の通知を受けることができる。

KeepAliveCheckerSelectorは、複数のKeepAliveCheckerを管理及び監視し、利用可能なKeepAliveCheckerを１つ選択したり、利用可能なKeepAliveCheckerを全て取得したりする機能を持つ。１つ選択する場合の選択方法は、KeepAliveCheckerSelectorの実装に依存する。

サーバ２

サーバ1

ClusterService

ClusterService

ClusterListener

ClusterListener

死活監視

ClusterService

サーバ３

ClusterListener

各サーバ上のClusterServiceは、相互に通信してお互いの存在を通知して、クラスタに参加しているメンバを全員で共有し認識する。クラスタに参加するメンバが増減すると、各ClusterServiceは、登録されたClusterListenerにその変化を通知する。サービスは、ClusterListenerインタフェースを実装する事で、いわゆるクラスタ構成を組むことが可能になる。

Nimbus Proxyサービス群は、サービスのインタフェースをプロキシするRemoteClientService、プロキシに機能を付与するNimbus AOPサービス群、リモートのサービスをRMIで呼び出すRemoteServerInvokerをRMIレジストリにバインドするRemoteServiceServerService、RemoteServerInvokerを呼び出すInvokerで構成される。

クライアント

RemoteClientService

Interceptor

Interceptor

InterceptorChainList

Interceptor

InterceptorChain

Invoker

RemoteServerInvoker

（リモートオブジェクト）

任意のインタフェースA

サーバ

任意のインタフェースAを

実装したサービス

RemoteServiceServerService

RemoteServerInvoker

（サーバオブジェクト）

InterceptorChain

InterceptorChainList

Interceptor

Interceptor

Interceptor

Invoker

上図の構成で、リモートのサービスを透過的に呼び出すことが可能になる。

死活監視と呼び出しの両方の機能を持つKeepAliveCheckInvokerを、KeepAliveCheckerSelectorに設定することで、複数の呼び出し可能なリソースの死活監視を行い、利用可能なリソースを呼び出す事が可能になる。

このKeepAliveCheckerSelectorを利用する事で、複数の呼び出し可能なリソースから選択的にKeepAliveCheckInvokerを決定してInvokerとして振る舞うClusterInvokerServiceと、前述したNimbus Proxyサービス群を組み合わせる事で、リモートの複数のサーバ上のサービスをクラスタ化して、選択的に呼び出すことが可能になる。

クライアント

RemoteClientService

任意のインタフェースA

InterceptorChain

ClusterInvoker

KeepAliveCheckerSelector

KeepAliveCheckInvoker

（サーバ１）

KeepAliveCheckInvoker

RemoteServerInvoker

（リモートオブジェクト）

KeepAliveCheckInvoker

（サーバ２）

RemoteServerInvoker

（リモートオブジェクト）

死活監視

サーバ１

RemoteServerInvoker

（サーバオブジェクト）

サーバ２

RemoteServiceServerService

RemoteServiceServerService

RemoteServerInvoker

（サーバオブジェクト）

任意のインタフェースAを

実装したサービス

任意のインタフェースAを

実装したサービス

但し、この構成では、クライアント側でサーバの台数を意識して、静的にKeepAliveCheckInvokerのサービス定義を記述する必要がある。従って、スケールアウトのために、動的にサーバを追加するような運用には、対応できない。

そこで、この構成にClusterServiceを加えることで、KeepAliveCheckInvokerをクラスタメンバのオプション情報として取得可能になり、サーバの台数を意識して静的にKeepAliveCheckInvokerのサービス定義を記述する必要がなくなる。

クライアント

RemoteClientService

任意のインタフェースA

InterceptorChain

KeepAliveCheckerSelector

ClusterService

ClusterInvoker

KeepAliveCheckInvoker

（サーバ１）

KeepAliveCheckInvoker

RemoteServerInvoker

（リモートオブジェクト）

KeepAliveCheckInvoker

（サーバ２）

RemoteServerInvoker

（リモートオブジェクト）

死活監視

サーバ１

ClusterService

RemoteServerInvoker

（サーバオブジェクト）

RemoteServiceServerService

サーバ２

KeepAliveCheckInvoker

RemoteServerInvoker

（リモートオブジェクト）

RemoteServerInvoker

（サーバオブジェクト）

任意のインタフェースAを

実装したサービス

任意のインタフェースAを

実装したサービス

JavaDocは、以下を参照。

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/keepalive/package-summary.html>

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/keepalive/jdbc/package-summary.html>

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/keepalive/smtp/package-summary.html>

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/proxy/package-summary.html>

<http://nimbus.osdn.jp/reports/apidocs/jp/ossc/nimbus/service/proxy/invoker/package-summary.html>

サンプルコードは、以下のzipファイル内のkeepalive、proxyを参照。

<https://osdn.jp/projects/nimbus/downloads/64236/nimbus-sample-1.2.3.zip>

# インタフェース設計

## jp.ossc.nimbus.service.keeplalive.KeepAliveChecker

リソースの死活監視を行う機能を抽象化したインタフェースである。

リソースの死活状態を能動的に判定するisAlive()と、KeepAliveListenerを登録して受動的に死活状態を監視する機能がある。

**KeepAliveChecker**

isAlive()

onDead()

onRecover()

addKeepAliveListener()

KeepAliveListener

KeepAliveListener

以下に、KeepAliveCheckerインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.keepalive.jdbc.JDBCKeepAliveCheckerService | JDBCを使ってデータベースの死活監視を行う実装クラスである。 |
| 2 | jp.ossc.nimbus.service.keepalive.smtp.SmtpCheckerService | SMTPプロトコルを使ってSMTPサーバの死活監視を行う実装クラスである。 |
| 3 | jp.ossc.nimbus.service.jndi.CachedJndiFinderService | JNDIのlookupを使ってJNDIサーバの死活監視を行う実装クラスである。 |

## jp.ossc.nimbus.service.keeplalive.KeepAliveListener

リソースの死活監視をイベント通知として受け取るインタフェースである。

リソースが死んだ時に通知されるonDead()と、リソースが生き返った時に通知されるonRecover()がある。

## jp.ossc.nimbus.service.keeplalive.KeepAliveCheckerSelector

リソースの死活監視を行うKeepAliveCheckerを複数管理して、利用可能なKeepAliveCheckerを提供するインタフェースである。

ここで言う利用可能とは、KeepAliveChecker#isAlive()がtrueを返すリソースを指しており、利用可能な全てのKeepAliveCheckerを取得するgetSelectableCheckers()と、そのうち１つを取得するselectChecker()がある。１つを選択するロジックは、実装に依存する。

selectChecker()

getSelectableCheckers ()

**KeepAliveCheckerSelector**

KeepAliveChecker

KeepAliveChecker

KeepAliveChecker

以下に、KeepAliveCheckerSelectorインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.keepalive.FirstAvailableKeepAliveCheckerSelectorService | 生存している最初のKeepAliveCheckerを選択する実装クラスである。 |
| 2 | jp.ossc.nimbus.service.keepalive.RoundRobinKeepAliveCheckerSelectorService | 生存しているKeepAliveCheckerをラウンドロビン方式で選択する実装クラスである。 |
| 3 | jp.ossc.nimbus.service.keepalive.ResourceAvailableKeepAliveCheckerSelectorService | 生存しているKeepAliveCheckerからリソース使用量が小さなものを選択する実装クラスである。 |

## jp.ossc.nimbus.service.keeplalive.ClusterListener

ClusterServiceに登録する事で、クラスタのメンバ変更通知を受け取るインタフェースである。

クラスタのメンバ変更通知は、クラスタメンバの初期状態の通知を受け取るmemberInit()、クラスタメンバが変更された通知を受け取るmemberChange()、クラスタメンバ内で主系になった通知を受け取るchangeMain()、クラスタメンバ内で副系になった通知を受け取るchangeSub()がある。

ClusterService

addKeepAliveListener()

memberInit()

memberChange()

changeMain()

changeSub()

**ClusterListener**

**ClusterListener**

以下に、ClusterListenerインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.keepalive.DefaultClusterListenerService | 設定されたサービスをクラスタの主系の時に開始状態にする実装クラスである。 |
| 2 | jp.ossc.nimbus.service.keepalive.AbstractKeepAliveCheckerSelectorService | 分散したサーバ上のKeepAliveCheckerをClusterServiceでクラスタ化して呼び出すために実装している抽象クラスである。 |
| 3 | jp.ossc.nimbus.service.beancontrol.ClientBeanFlowInvokerFactoryService | 分散したサーバ上のBeanFlowInvokerServerをClusterServiceでクラスタ化して呼び出すために実装しているクラスである。 |
| 4 | jp.ossc.nimbus.service.publish.ClusterClientConnectionFactoryService | 分散したサーバ上のClientConnectionFactoryをClusterServiceでクラスタ化して呼び出すために内部クラスで実装している。 |
| 5 | jp.ossc.nimbus.service.publish.ClusterClientConnectionImpl | 分散したサーバ上のClientConnectionFactoryをClusterServiceでクラスタ化して呼び出すために実装しているクラスである。 |
| 6 | jp.ossc.nimbus.service.scheduler2. AbstractSchedulerService | 分散したサーバ上のSchedulerをClusterServiceでクラスタ化するために内部クラスで実装している抽象クラスである。 |
| 7 | jp.ossc.nimbus.service.scheduler2.DatabaseScheduleManagerService | 分散したサーバ上のDatabaseScheduleManagerServiceをClusterServiceでクラスタ化するために内部クラスで実装している。 |

## jp.ossc.nimbus.service.proxy.invoker.KeepAliveCheckInvoker

リソースの死活監視を行うKeepAliveCheckerと、Nimbus AOPサービス群の呼び出し機能であるInvokerの両方を実装したインタフェースである。

ClusterInvokerServiceなどから利用され、分散したサーバ上の呼び出し可能なリソースの死活監視を行い、生きているリソースを呼び出す事を可能にする。

リソースの死活監視をするisAlive()、リソースの呼び出しを行うinvoke()、リソースの利用量を取得するgetResourceUsage()などの機能がある。

selectChecker()

ClusterInvokerService

invoke()

KeepAliveCheckerSelector

**KeepAliveCheckInvoker**

isAlive()

getResourceUsage()

**KeepAliveCheckInvoker**

**KeepAliveCheckInvoker**

以下に、KeepAliveCheckInvokerインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.aop.invoker.MethodReflectionCallInvokerService | ローカルの呼び出し対象オブジェクトのメソッドを呼び出す実装クラスである。  死活監視は、自分自身が開始状態であるかどうかで判定する。 |
| 2 | jp.ossc.nimbus.service.beancontrol.BeanFlowInvokerServerService | リモートからのBeanFlowを呼び出し可能にする実装クラスである。  死活監視は、BeanFlowInvokerFactoryサービスが開始状態であるかどうかで判定する。 |
| 3 | jp.ossc.nimbus.service.proxy.RemoteServiceServerService | リモートからの任意のサービスの呼び出しを可能にする実装クラスである。  死活監視は、呼び出し対象のサービスが固定化されている場合は、そのサービスが開始状態であるかどうかで判定する。固定化されていない場合は、RMIサーバオブジェクトが存在すればtrue。 |
| 4 | jp.ossc.nimbus.service.proxy.invoker.LocalClientMethodCallInvokerService | ローカルのサービスのメソッドを呼び出す実装クラスである。  死活監視は、呼び出し対象のサービスが開始状態であるかどうかで判定する。 |
| 5 | jp.ossc.nimbus.service.proxy.invoker.RemoteClientMethodCallInvokerService | リモートのサービスのメソッドを呼び出す実装クラスである。  死活監視は、リモートのサービスを呼び出すRMIリモートオブジェクトのRemoteServerInvoke#isAlive()で判定する。 |

## jp.ossc.nimbus.service.proxy.RemoteServerInvoker

リモートのサービスの死活監視及び、抽象的な呼び出し機能を実装したRMIインタフェースである。KeepAliveCheckInvokerとほぼ同じ機能を持ち、RMIのリモートインタフェースとなっている。

KeepAliveCheckInvokerの実装クラスのうち、リモートの呼び出しを行う実装で、内部的に使われる。

以下に、RemoteServerInvokerインタフェースの実装クラスの一覧を示す。

|  |  |  |
| --- | --- | --- |
| # | 実装クラス名 | 概要 |
| 1 | jp.ossc.nimbus.service.proxy.RemoteServiceServerService | リモートからサービスを呼び出すためにRemoteServerInvokerをRMIサーバオブジェクトとして配置する。内部クラスで実装している。 |
| 2 | jp.ossc.nimbus.service.proxy.RemoteServiceServerEJBObject | リモートからサービスを呼び出すためのEJBオブジェクトインタフェース。RemoteServerInvokerを継承している。 |

# 実装設計

インタフェース設計で示した各インタフェースのうち、機能性のあるインタフェースの実装クラスの機能を説明する。

また、インタフェース設計に現れなかった、クラスの機能も説明する。

## jp.ossc.nimbus.service.keeplaive.ClusterService

複数のJavaVM間でクラスタを構成して、クラスタに参加しているメンバ情報を共有する。最初に起動したメンバは、クラスタの管理者となり、他のメンバを管理する立場になる。後続のメンバは、クラスタへの参加順で優先順位が決まり、管理者がメンバから外れると、次の優先順位のメンバが管理者へ昇格する。また、クラスタメンバ内には、メンバ中の１つだけが主系となり、他のメンバは副系になるという関係があり、管理者が主系となる。

ClusterService 1

（管理者=主系）

ClusterListener

UID 1

UID 3

メンバ情報

UID 2

UID 1

UID 2

UID 3

メンバ情報

UID 3

UID 1

UID 2

UID 3

ClusterService 2

（副系）

ClusterService 3

（副系）

メンバ情報

ハートビート

ハートビート

UID 1

UID 2

ハートビート

クラスタのメンバは、それぞれUID（Unified Identifier）を持ち、メンバ情報としてクラスタに参加しているメンバのUIDを管理、共有する。また、UIDに付属情報として、任意のオブジェクトを設定し、共有する事もできる。

クラスタのメンバは、メンバ情報の中の自分の次のメンバに対してハートビートを行い、死活監視を行う。メンバ情報の最後のメンバは、メンバ情報の先頭である管理者メンバを監視する事で、監視の輪を形成する。監視対象のメンバが死んだことを検知すると、管理者に通知してメンバ情報の更新を促す。通知を受けた管理者は、メンバ情報を更新し、クラスタメンバ全員にメンバ情報の更新を通知する。

クラスタ間の通信プロトコルは、設定によってマルチキャスト、ブロードキャスト、UDPユニキャストが選択可能である。

ClusterService には、ClusterListenerを設定でき、メンバの変更、主系/副系の切り替わりなどの通知を受け取る事ができる。

また、クライアントモードにすると、クラスタには参加しないが、Clusterのメンバ情報を共有する事ができる。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | MulticastGroupAddress | String | マルチキャストグループまたはブロードキャストのIPアドレスを設定する。 |
| 2 | MulticastPort | int | マルチキャストまたはブロードキャストのポート番号を設定する。  デフォルトは、1500。 |
| 3 | UnicastMemberAddresses | String[] | ユニキャストメンバーのIPアドレスを設定する。 |
| 4 | UnicastPort | int | ユニキャストのポート番号を設定する。  デフォルトは、1500。 |
| 5 | ReceiveBufferSize | int | パケットの受信バッファサイズを設定する。  デフォルトは、1024バイト。 |
| 6 | TimeToLive | int | マルチキャストパケットの有効期間を設定する。 |
| 7 | LocalAddress | String | ローカルIPアドレスを設定する。 |
| 8 | BindAddress | String | バインドアドレスを設定する。 |
| 9 | NetworkInterfaces | String[] | ネットワークインタフェースを設定する。 |
| 10 | Option | Serializable | UIDの付属情報を設定する。 |
| 11 | HeartBeatInterval | long | 隣のクラスタサービスとハートビートを行う間隔[ms]を設定する。  デフォルトは、1000[ms]。 |
| 12 | HeartBeatResponseTimeout | long | 隣のクラスタサービスにハートビートをリクエストした時のタイムアウト[ms]を設定する。  デフォルトは、500[ms]。 |
| 13 | HeartBeatRetryCount | int | 隣のクラスタサービスとのハートビートに失敗した場合に、相手が死んだと見なすまでのリトライ回数を設定する。  デフォルトは、1。 |
| 14 | AddMemberResponseTimeout | long | クラスタに参加するリクエストを行った時のタイムアウト[ms]を設定する。  デフォルトは、500[ms]。 |
| 15 | AddMemberRetryCount | int | クラスタに参加するリクエストを行った時のリトライ回数を設定する。  デフォルトは、0でリトライしない。 |
| 16 | LostTimeout | long | 受信したパケットが分割されていた場合に、分割された残りパケットが消失したとみなすタイムアウト[ms]を設定する。  デフォルトは、500[ms]。 |
| 17 | Client | boolean | クラスタクライアントかどうかを設定する。  クラスタクライアントになる場合は、trueに設定する。デフォルトは、false。  クラスタクライアントは、クラスタのメンバには参加せずクラスタの状況通知のみ受ける。 |
| 18 | JoinOnStart | boolean | サービスの開始時に、クラスタに参加するかどうかを設定する。  デフォルトは、true。 |

## jp.ossc.nimbus.service.keepalive.KeepAliveChecker

### jp.ossc.nimbus.service.keepalive.jdbc.JDBCKeepAliveCheckerService

JDBCを使ってデータベースの死活監視を行うKeepAliveCheckerインタフェースの実装クラスである。

設定された死活監視用のSQLが正常終了する事をもって、データベースが生存していることを監視する。

データベース

Connection

isAlive()

**JDBCKeepAliveCheckerService**

ConnectionFactory

CheckQuery

CheckInterval属性に有効な値が設定されている場合は、監視デーモンが起動し、定期的に死活監視を行い、isAlive()メソッドの呼び出しでは、監視デーモンが直近でチェックした死活状態を応答する。CheckInterval属性に有効な値が設定されていない場合は、監視デーモンは起動せず、isAlive()メソッドの呼び出し時に、死活監視のチェックを行う。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | CheckInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  この属性を設定しない場合は、定期監視を行わず毎回問い合わせを行う。  デフォルトは-1で、毎回問い合わせを行う。 |
| 2 | ConnectionFactoryServiceName | ServiceName | JDBCコネクションを取得するjp.ossc.nimbus.service.connection.ConnectionFactoryサービスのサービス名を設定する。 |
| 3 | CheckQuery | String | データベースの死活監視で使用するSQLを設定する。 |

### jp.ossc.nimbus.service.keepalive.smtp.SmtpKeepAliveCheckerService

SMTPプロトコルを使ってSMTPサーバの死活監視を行うKeepAliveCheckerインタフェースの実装クラスである。

設定されたSMTPサーバに、HEELOメッセージを送り、応答コードが2から始まるコードである事で、SMTPサーバが正常に稼働していることを監視する。

isAlive()

**SmtpKeepAliveCheckerService**

HELLO

SMTPサーバ

AliveCheckSMTPServer属性にtrueが設定されている場合は、監視デーモンが起動し、定期的に死活監視を行い、isAlive()メソッドの呼び出しでは、監視デーモンが直近でチェックした死活状態を応答する。AliveCheckSMTPServer属性にfalseが設定されている場合は、監視デーモンは起動せず、isAlive()メソッドの呼び出し時に、死活監視のチェックを行う。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | AliveCheckSMTPServer | boolean | 定期的に死活監視を行うかどうかを設定する。  デフォルトはfalseで、毎回問い合わせを行う。 |
| 2 | AliveCheckSMTPServerInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  デフォルトは60秒。 |
| 3 | HostName | String | SMTPサーバのホスト名を設定する。 |
| 4 | Port | int | SMTPサーバのポート番号を設定する。 |
| 5 | ConnectionTimeoutMillis | long | SMTPサーバへの接続タイムアウト[ms]を設定する。  デフォルトは、0で無限待ち。 |
| 6 | TimeoutMillis | long | SMTPサーバへの応答タイムアウト[ms]を設定する。  デフォルトは、1秒。 |

### jp.ossc.nimbus.service.jndi.CachedJndiFinderService

JNDIのlookup()を使ってJNDIサーバの死活監視を行うKeepAliveCheckerインタフェースの実装クラスである。JNDIサーバのルートコンテキストをlookup()できる事をもって、JNDIサーバが生存していることを監視する。

isAlive()

**CachedJndiFinderService**

JNDIサーバ

lookup()

CheckInterval属性に有効な値が設定されている場合は、監視デーモンが起動し、定期的に死活監視を行い、isAlive()メソッドの呼び出しでは、監視デーモンが直近でチェックした死活状態を応答する。CheckInterval属性に有効な値が設定されていない場合は、監視デーモンは起動せず、isAlive()メソッドの呼び出し時に、死活監視のチェックを行う。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | AliveCheckJNDIServer | boolean | 定期的に死活監視を行うかどうかを設定する。  この属性をfalseに設定した場合は、定期監視を行わず毎回問い合わせを行う。  デフォルトはfalseで、毎回問い合わせを行う。 |
| 2 | AliveCheckJNDIServerInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  デフォルトは60秒。 |
| 3 | Environment | Properties | 初期コンテキストのJNDI環境変数を設定する。 |

## jp.ossc.nimbus.service.keepalive.KeepAliveCheckerSelector

### jp.ossc.nimbus.service.keepalive.FirstAvailableKeepAliveCheckerSelectorService

利用可能なKeepAliveCheckerリソースのうち、最初のリソースを選択するKeepAliveCheckerSelectorインタフェースの実装クラスである。

**FirstAvailableKeepAliveCheckerSelectorService**

ClusterService

selectChecker ()

KeepAliveChecker

KeepAliveChecker

KeepAliveChecker

isAlive()

選択可能なKeepAliveCheckerは、サービス定義で静的に設定するか、ClusterServiceを設定して、クラスタメンバの付属情報からKeepAliveCheckerのリモートオブジェクトを取得する。

また、選択可能なKeepAliveCheckerそれぞれの死活監視は、CheckInterval属性に有効な値が設定されている場合は、監視デーモンを起動し、定期的にisAlive()を呼び出し、利用可能なKeepAliveCheckerを管理する。CheckInterval属性に有効な値が設定されていない場合は、監視デーモンを起動せず、selecChecker()呼び出し時に、選択可能なKeepAliveCheckerに対してisAlive()を呼び出し、利用可能なKeepAliveCheckerを選択する。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | CheckInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  この属性を設定しない場合は、定期監視を行わず毎回問い合わせを行う。  デフォルトは-1で、毎回問い合わせを行う。 |
| 2 | SelectableCheckerServiceNames | ServiceName[] | 選択可能なKeepAliveCheckerのサービス名を設定する。 |
| 3 | ClusterServiceName | ServiceName | 選択可能なKeepAliveCheckerを取得するClusterServiceのサービス名を設定する。 |
| 4 | KeepOrder | boolean | 選択可能なKeepAliveCheckerの順序を保証するかどうかを設定する。  デフォルトは、falseで、稼動→停止→稼動となったKeepAliveCheckerは、最後に並べられる。 |

### jp.ossc.nimbus.service.keepalive.RoundRobinKeepAliveCheckerSelectorService

利用可能なKeepAliveCheckerリソースのうち、ラウンドロビンでリソースを選択するKeepAliveCheckerSelectorインタフェースの実装クラスである。

**RoundRobinKeepAliveCheckerSelectorService**

ClusterService

selectChecker ()

KeepAliveChecker

KeepAliveChecker

KeepAliveChecker

isAlive()

選択可能なKeepAliveCheckerは、サービス定義で静的に設定するか、ClusterServiceを設定して、クラスタメンバの付属情報からKeepAliveCheckerのリモートオブジェクトを取得する。

また、選択可能なKeepAliveCheckerそれぞれの死活監視は、CheckInterval属性に有効な値が設定されている場合は、監視デーモンを起動し、定期的にisAlive()を呼び出し、利用可能なKeepAliveCheckerを管理する。CheckInterval属性に有効な値が設定されていない場合は、監視デーモンを起動せず、selecChecker()呼び出し時に、選択可能なKeepAliveCheckerに対してisAlive()を呼び出し、利用可能なKeepAliveCheckerを選択する。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | CheckInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  この属性を設定しない場合は、定期監視を行わず毎回問い合わせを行う。  デフォルトは-1で、毎回問い合わせを行う。 |
| 2 | SelectableCheckerServiceNames | ServiceName[] | 選択可能なKeepAliveCheckerのサービス名を設定する。 |
| 3 | ClusterServiceName | ServiceName | 選択可能なKeepAliveCheckerを取得するClusterServiceのサービス名を設定する。 |
| 4 | KeepOrder | boolean | 選択可能なKeepAliveCheckerの順序を保証するかどうかを設定する。  デフォルトは、falseで、稼動→停止→稼動となったKeepAliveCheckerは、最後に並べられる。 |

### jp.ossc.nimbus.service.keepalive.ResourceAvailableKeepAliveCheckerSelectorService

利用可能なKeepAliveCheckInvokerリソースのうち、最もリソース利用量が少ないリソースを選択するKeepAliveCheckerSelectorインタフェースの実装クラスである。

**ResourceAvailableKeepAliveCheckerSelectorService**

ClusterService

selectChecker ()

isAlive()

getResourceUsage()

KeepAliveCheckInvoker

KeepAliveChecker

KeepAliveCheckInvoker

選択可能なKeepAliveCheckInvokerは、サービス定義で静的に設定するか、ClusterServiceを設定して、クラスタメンバの付属情報からKeepAliveCheckInvokerのリモートオブジェクトを取得する。

また、選択可能なKeepAliveCheckInvokerそれぞれの死活監視は、CheckInterval属性に有効な値が設定されている場合は、監視デーモンを起動し、定期的にisAlive()及びgetResourceUsage()を呼び出し、利用可能なKeepAliveCheckInvokerを管理する。CheckInterval属性に有効な値が設定されていない場合は、監視デーモンを起動せず、selecChecker()呼び出し時に、選択可能なKeepAliveCheckInvokerに対してisAlive()及びgetResourceUsage()を呼び出し、利用可能なKeepAliveCheckInvokerを選択する。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | CheckInterval | long | 定期的に死活監視を行う間隔[ms]を設定する。  この属性を設定しない場合は、定期監視を行わず毎回問い合わせを行う。  デフォルトは-1で、毎回問い合わせを行う。 |
| 2 | SelectableCheckerServiceNames | ServiceName[] | 選択可能なKeepAliveCheckInvokerのサービス名を設定する。 |
| 3 | ClusterServiceName | ServiceName | 選択可能なKeepAliveCheckInvokerを取得するClusterServiceのサービス名を設定する。 |
| 4 | KeepOrder | boolean | 選択可能なKeepAliveCheckInvokerの順序を保証するかどうかを設定する。  デフォルトは、falseで、稼動→停止→稼動となったKeepAliveCheckInvokerは、最後に並べられる。 |

## jp.ossc.nimbus.service.keepalive.ClusterListener

### jp.ossc.nimbus.service.keepalive.DefaultClusterListenerService

クラスタの主系、副系の切り替わりに応じて、設定されたサービスを開始、停止するClusterListenerインタフェースのデフォルト実装クラスである。

ClusterService

changeMain()

or changeSub()

**DefaultClusterListenerService**

任意のService

start()

or stop()

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | ClusrterServiceName | ServiceName | ClusterServiceのサービス名を設定する。 |
| 2 | TargetServiceName | ServiceName | クラスタの主系/副系切り替えと同期するサービスのサービス名を設定する。 |

## jp.ossc.nimbus.service.proxy.invoker.KeepAliveCheckInvoker

### jp.ossc.nimbus.service.proxy.RemoteServiceServerService

RMIでリモートからサービスを呼び出す口を開くRemoteServerInvokerをリモートサーバとしてホスティングして、それを呼び出すKeepAliveCheckInvokerインタフェースの実装クラスをClusterServiceのメンバ情報の付属情報として設定するサービスである。

クライアント

サーバ

**RemoteServiceServerService**

ClusterService

メンバ情報

ClusterService

メンバ情報

KeepAliveCheckInvoker

KeepAliveCheckInvoker

RemoteServerInvoker

InterceptorChain

KeepAliveCheckInvoker

InterceptorChainList

Interceptor

Interceptor

Invoker

Interceptor

RemoteServerInvoker

(リモートオブジェクト)

isAlive()

getResourceUsage()

invoke()

lookup()

ResourceUsage

任意のサービス

Repository

Repository

RMIレジストリ

リモートのサービスを呼び出す側となるクライアント側にも、ClusterServiceを配置する事で、メンバ情報が共有され、そこからKeepAliveCheckInvokerを取得する事ができる。取得したKeepAliveCheckInvokerは、内部にRemoteServerInvokerのリモートオブジェクトを持っており、サーバ側にホスティングされているRemoteServerInvokerを呼び出す事で、死活監視（isAlive()）、リソース使用量の取得（getResourceUsage()）、リモートサービスの呼び出し（invoke()）などの機能を提供する。

また、Repositoryサービスを経由して、JNDIサーバ等のRMIレジストリにRemoteServerInvokerを登録することで、クライアント側でRepositoryサービス経由を経由して、RemoteServerInvokerのリモートオブジェクトを取得する事もできる。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | RemoteServiceName | ServiceName | リモートから呼び出されるサービスのサービス名を設定する。  設定しない場合は、呼び出し側で指定した任意のサービスが呼び出せる。 |
| 2 | InterceptorChainListServiceName | ServiceName | リモートから呼び出される時に挟み込むInterceptorChainListサービスのサービス名を設定する。 |
| 3 | InvokerServiceName | ServiceName | リモートから呼び出される時に、対象のサービスを呼び出すInvokerサービスのサービス名を設定する。  設定しない場合には、デフォルトでローカルのサービスをリフレクションで呼び出すMethodReflectionCallInvokerServiceが適用される。 |
| 4 | InterceptorChainFactoryServiceName | ServiceName | リモートから呼び出される時に挟み込むInterceptorChainを取得するInterceptorChainFactoryサービスのサービス名を設定する。 |
| 5 | JndiName | String | RepositoryサービスにRemoteServerInvokerを登録する際の、名前を設定する。 |
| 6 | JndiRepositoryServiceName | ServiceName | RemoteServerInvokerを登録するRepositoryサービスのサービス名を設定する。 |
| 7 | RMIPort | int | RemoteServerInvokerをRMI呼び出しする際に使用するポート番号を設定する。  デフォルトは、0で任意のポートが使用される。 |
| 8 | ClusterServiceName | ServiceName | KeepAliveCheckInvokerをクラスタのメンバ情報の付属情報として登録するためのClusterServiceのサービス名を設定する。 |
| 9 | ResourceUsageServiceName | ServiceName | リソース利用量を取得するResourceUsageサービスのサービス名を設定する。  設定しない場合は、getResourceUsage()の戻り値はnullとなる。 |
| 10 | RMIClientSocketFactoryServiceName | ServiceName | RMI呼び出し時に、クライアント側でソケットの生成に使用するRMIClientSocketFactoryサービスのサービス名を設定する。 |
| 11 | RMIServerSocketFactoryServiceName | ServiceName | RMI呼び出し時に、サーバ側でサーバソケットの生成に使用するRMIServerSocketFactoryサービスのサービス名を設定する。 |

### jp.ossc.nimbus.service.proxy.invoker.LocalClientMethodCallInvokerService

ローカルのサービスを呼び出すKeepAliveCheckInvokerインタフェースの実装クラスである。

**LocalClientMethodCallInvokerService**

isAlive()

getResourceUsage()

invoke()

ResourceUsage

任意のサービス

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | LocalServiceName | ServiceName | 呼び出されるサービスのサービス名を設定する。  設定しない場合は、呼び出し側で指定した任意のサービスが呼び出せる。 |
| 2 | ResourceUsageServiceName | ServiceName | リソース利用量を取得するResourceUsageサービスのサービス名を設定する。  設定しない場合は、getResourceUsage()の戻り値はnullとなる。 |

### jp.ossc.nimbus.service.proxy.invoker.RemoteClientMethodCallInvokerService

ローカルのサービスを呼び出すKeepAliveCheckInvokerインタフェースの実装クラスである。

**RemoteClientMethodCallInvokerService**

isAlive()

getResourceUsage()

invoke()

RemoteServerInvoker

（リモートオブジェクト）

lookup()

JndiFinder or Repository

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | RemoteServiceName | ServiceName | 呼び出したいリモートのサービスのサービス名を設定する。  設定しない場合は、呼び出し情報で指定された任意のサービスが呼び出せる。 |
| 2 | JndiFinderServiceName | ServiceName | RemoteServerInvokerをlookupするJndiFinderサービスのサービス名を設定する。 |
| 3 | JndiRepositoryServiceName | ServiceName | RemoteServerInvokerをlookupするRepositoryサービスのサービス名を設定する。 |
| 4 | RemoteServerJndiName | String | RemoteServerInvokerのJNDI名を設定する。 |

## jp.ossc.nimbus.service.proxy.RemoteServerInvoker

### jp.ossc.nimbus.service.proxy.RemoteServiceServerService

3.5.1を参照。

## jp.ossc.nimbus.service.proxy.RemoteClientService

任意のインタフェースを実装しているように見せかけるプロキシサービスである。

**RemoteClientService**

任意のインタフェースA

Invoker

Interceptor

InterceptorChainList

Interceptor

Interceptor

InterceptorChain

任意のメソッド

プロキシする任意のインタフェース上のメソッドの呼び出しに対して、処理を付与するInterceptorChainと、最終的な呼び出しの実行を行うInvokerを内部に持ち、処理を委譲する。

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | RemoteInterfaceClassName | String | プロキシするインタフェースのクラス名を設定する。 |
| 2 | RemoteServiceName | ServiceName | プロキシするサービスのサービス名を設定する。 |
| 3 | InterceptorChainListServiceName | ServiceName | プロキシに挟み込むjp.ossc.nimbus.service.aop.InterceptorChainListサービスのサービス名を設定する。 |
| 4 | InvokerServiceName | ServiceName | プロキシするサービスを呼び出すjp.ossc.nimbus.service.aop.Invokerサービスのサービス名を設定する。 |
| 5 | InterceptorChainFactoryServiceName | ServiceName | プロキシに挟み込むjp.ossc.nimbus.service.aop.InterceptorChainを生成するjp.ossc.nimbus.service.aop.InterceptorChainFactoryのサービス名を設定する。 |
| 6 | CreateNewProxy | boolean | プロキシを毎回生成するかどうかを設定する。  デフォルトは、false。 |
| 7 | CreateInterceptorChainByProxy | boolean | プロキシに紐付けてインターセプタチェインを生成するかどうかを設定する。  デフォルトは、false。 |

## jp.ossc.nimbus.service.aop.Invoker

このインタフェース自体は、Nimbus AOPのインタフェースであるが、RemoteClientServiceのInvokerとして利用する事を想定して実装されたInvokerを、jp.ossc.nimbus.service.proxy.invokerパッケージで提供するため、それらの実装の説明をする。

### jp.ossc.nimbus.service.proxy.invoker.ClusterInvokerService

KeepAliveCheckerSelectorを利用して、複数のKeepAliveCheckInvokerから１つを選んで呼び出したり、複数のKeepAliveCheckInvokerを全て呼び出したりするInvokerインタフェースの実装クラスである。

selectChecker()

or

getSelectableCheckers()

invoke()

**ClusterInvokerService**

invoke()

KeepAliveCheckerSelector

KeepAliveCheckInvoker

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | KeepAliveCheckerSelectorServiceName | ServiceName | KeepAliveCheckerSelectorサービスのサービス名を設定する。 |
| 2 | ExceptionConditions | String[] | リトライする例外のクラス名とその条件を設定する。  この条件を設定しない場合は、{@link jp.ossc.nimbus.service.proxy.RemoteServiceCallException}をキャッチした場合にリトライする。  条件式は、The Apache Jakarta Projectの Commons Jexl(http://jakarta.apache.org/commons/jexl/)を使用する。  例外のプロパティを参照する場合は、プロパティを表現する文字列を"@"で囲んで指定する。ここで言う、プロパティの概念は、Java Beansのプロパティの概念より広く、jp.ossc.nimbus.beans.PropertyFactoryの規約に従う。 |
| 3 | MaxRetryCount | int | リトライする回数を設定する。  デフォルトは、0でリトライしない。 |
| 4 | RetryInterval | long | リトライする間隔[ms]を設定する。  デフォルトは、0で間隔をあけずにリトライする。 |
| 5 | Broadcast | boolean | 呼び出し時点で生存している全てのKeepAliveCheckInvokerを呼び出すようにするかどうかを設定する。  trueの場合、呼び出し時点で生存している全てのKeepAliveCheckInvokerを順次呼び出す。その際、リトライ機能は無効となり、どれか１つのKeepAliveCheckInvokerの呼び出しで例外が発生すると、そこで終了する。  デフォルトは、false |

### jp.ossc.nimbus.service.proxy.invoker.JMXClientRMICallInvokerService

JMX（**J**ava **M**anagement e**X**tentions）のリモート呼び出し機能を使って、リモートMBean（Managed Bean）を呼び出すInvokerインタフェース実装クラスである。

invoke()

**JMXClientRMICallInvokerService**

invoke()

setAttribute()

getAttribute()

javax.management.MBeanServerConnection

以下にサービスの属性一覧を示す。

|  |  |  |  |
| --- | --- | --- | --- |
| # | 属性名 | 型 | 説明 |
| 1 | JndiFinderServiceName | ServiceName | javax.management.MBeanServerConnectionをJNDIからlookupするjp.ossc.nimbus.service.jndi.JndiFinderサービスのサービス名を設定する。  JMX規格のネットワーク層の接続方法が、規定される前の、JBossASのためのレガシー実装。 |
| 2 | RMIAdaptorName | String | javax.management.MBeanServerConnectionのJNDI名を設定する。 |
| 3 | ServiceURL | String | JMX API コネクタサーバーのアドレスを設定する。 |
| 4 | JMXConnectorEnvironment | Map<String,String> | JMX API コネクタ接続の確立方法を決定付ける属性のセットを設定する。 |
| 5 | ObjectName | String | 呼び出すMBeanのJMXオブジェクト名を設定する。 |
| 6 | ObjectNameDomain | String | 呼び出すMBeanのJMXドメイン名を設定する。  ObjectName属性を設定している場合は、不要。 |
| 7 | ObjectNameProperties | Properties | 呼び出すMBeanのJMXオブジェクト名のプロパティを設定する。  ObjectName属性を設定している場合は、不要。 |
| 8 | MBeanQuery | String | 呼び出すMBeanのJMXオブジェクト名を特定するためのクエリーを設定する。 |
| 9 | ObjectNameRegex | String | 呼び出すMBeanのJMXオブジェクト名を特定するための正規表現を設定する。 |