Symantec™ ApplicationHA 6.2 汎用アプリケーションエージェント設定ガイド - KVM での Linux

3 月 2015

Symantec™ ApplicationHA 汎用アプリケーションエージェント設定ガイド

この本で説明されているソフトウェアは使用許諾契約の下で提供され、同意条項に従う場合にのみ使うことができます。

エージェントバージョン: 6.2

マニュアルバージョン:6.2 Rev 1.

法的通知と登録商標

Symantec、Symantec ロゴ、Checkmark ロゴ、Veritas、Veritas Storage Foundation、CommandCentral、NetBackup、Enterprise Vault、LiveUpdate は、Symantec Corporation または同社の米国およびその他の国における関連会社の商標または登録商標です。その他の会社名、製品名は各社の登録商標または商標です。

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043

http://www.symantec.com

目次

1

Symantec ApplicationHA の汎用アプリケーションエージェント

この章では以下の項目について説明しています。

汎用エージェントについて

汎用エージェントは、カスタムアプリケーションのオンライン化とオフライン化を行い、その状態を監視します。さまざまなプログラムのオンライン、オフライン、監視ルーチンに対して異なる実行可能ファイルを指定するには、このエージェントを使います。 実行ファイルは、仮想マシンのローカルに置く必要があります。このエージェントを使うと、ApplicationHA がデフォルトでサポートしていないアプリケーションにも高可用性を提供できます。

サポート対象アプリケーションの一覧については、『Symantec ApplicationHA インストールガイド』を参照してください。

アプリケーションはルートのデフォルト環境で実行されます。

アプリケーションは次の方法で管理できます。

- 監視プログラムを使用する
- プロセスのリストを指定する
- プロセス ID ファイルのリストを指定する
- 上記の方法を任意に組み合わせる

メモ: ApplicationHA カスタムアプリケーションウィザードで設定できるのは監視プログラムのみです。プロセスまたはプロセス ID ファイルのリストを指定する方法について詳しくは、p.29 の「PID ファイルを使った複数の処理を監視するカスタム監視プログラムの書き込み」を参照してください。を参照してください。

IMF 対応

アプリケーション エージェントは IMF（Intelligent Monitoring Framework）対応であり、IMF 通知に非同期の AMF（Asynchronous Monitoring Framework）カーネルドライバを使います。Symantec ApplicationHA 設定ウィザードを使ってエージェントを設定すると、ウィザードにより IMF がデフォルトで有効になります。

IMF とインテリジェントなリソースの監視について詳しくは、『Symantec Cluster Server 管理者ガイド』を参照してください。

IMF 関連の アプリケーション エージェントの機能について詳しくは、「エージェント関数」を参照してください。

汎用エージェントでの IMF の使用

インテリジェントな監視は特定の設定においてのみ汎用エージェントでサポートされます。サポートされる設定の完全なリストを次の表に示します。

表 1-1

MonitorProgram	MonitorProcesses	PidFiles	IMF 監視モード
未設定	未設定	未設定	適用不可能
未設定	未設定	設定済み	online、offline
未設定	設定済み	未設定	online、offline
未設定	設定済み	設定済み	online、offline
設定済み	未設定	未設定	オフラインのみ
設定済み	未設定	設定済み	online、offline
設定済み	設定済み	未設定	online、offline
設定済み	設定済み	設定済み	online、offline

メモ: MonitorProcesses を設定しない場合、IMF はオフラインノードの StartProgram のみを監視します。そのため、IMF がオフラインノードのリソースを監視サイクルごとに監視するように IMF 属性の MonitorFreq を 1 に設定する必要があります。

MonitorProcesses 属性で複数のプロセスが設定されていて、それらの一部のみが実行されている場合、RegisterRetryLimit に到達するまで IMF へのオフライン登録が繰り返し失敗します。そのような場合、IMF ではリソースがいつ ONLINE になったかを判断できないので、エージェントは従来からの方法でリソースを監視します。

エージェント関数

Online

StartProgram 属性の値で指定した実行可能ファイルを実行します。指定したユーザー環境において、指定したパラメータで実行可能ファイルを実行します。

リソースをオンラインにするため、エージェント関数は次のコマンドを実行します。

```
su [-] user -c executable_to_online_resource
```

Offline

StopProgram 属性の値で指定した実行可能ファイルを実行します。指定したユーザー環境において、指定したパラメータで実行可能ファイルを実行します。

リソースをオフラインにするため、エージェント関数は次のコマンドを実行します。

```
su [-] user -c executable_to_offline_resource
```

Monitor

MonitorProgram 属性が指定されている場合、エージェントは、ユーザーが指定した環境でユーザー定義の MonitorProgram を実行します。PidFiles 属性が指定されている場合は、リストに表示された各ファイル内にあるプロセス ID が実行中であることを確認します。MonitorProcesses 属性が指定されている場合、ルーチンは、ユーザーが指定した環境で、リストに示された各プロセスが実行中であることを確認します。

これらの属性（MonitorProgram、PidFiles、または MonitorProcesses）の任意の組み合わせは、アプリケーションを監視するために使います。

PidFiles または MonitorProcesses のいずれかで指定したプロセスの 1 つでも動作していないことを監視プロセスが見つけた場合、monitor は OFFLINE を返します。プロセスが正常でない状態で終了している場合は、monitor は OFFLINE を返し、フェールオーバーが発生します。

リソースを監視するため、エージェント関数は次のコマンドを実行します。

```
su [-] user -c executable_to_monitor_resource
```

imf_init	AMF（Asynchronous Monitoring Framework）カーネルドライバと連動するようにエージェントを初期化します。この機能は、エージェントが起動すると実行されます。
imf_getnotification	リソースの状態の変更についての通知を取得します。この機能は、AMF カーネルドライバと連動するようにエージェントが初期化された後に実行されます。エージェントは継続的に通知を待機し、通知後にリソースでアクションを実行します。
imf_register	エージェントで監視する必要があるリソースエンティティを AMF カーネルドライバに登録します。たとえば、この関数はプロセスのオンライン監視のための PID を登録します。この機能は、リソースが安定した状態（オンラインまたはオフライン）に入った後に、リソースごとに実行されます。Application エージェントは PidFiles 属性と MonitorProcesses 属性で設定されたプロセスに対して IMF を使います。
Clean	PidFiles または MonitorProcesses で指定した処理を終了します。User 属性で指定されたユーザー ID で実行中のプロセス（MonitorProcesses 属性で指定）のみが強制終了（kill）されます。CleanProgram が定義されている場合、エージェントは、プロセスを強制終了するために、CleanProgram を実行します。 リソースを強制的に停止するため、エージェント関数は次のコマンドを実行します。 `su [-] user -c executable_to_clean_resource` エージェントが `su -` オプションを使うのは、UseSUDash 属性が有効（1）になっている場合のみであることに注意してください。UseSUDash 属性はデフォルトでは無効（0）になっています。

Action

action エントリポイントの各種の機能には次のものがあります。

- program.vfd
  指定したプログラムが使用可能かどうかと、指定したプログラムに対する実行権限を調べます。
- user.vfd
  ホストでの特定ユーザーの存在の有無を調べます。
- cksum.vfd
  すべてのノードにおける同一バイナリの存在を確認します。
- propcv
  (内部使用のみ)クラスタ内においてプロセスが VCS 制御外でアプリケーションリソースに対して起動することを許可または防止するかどうかを決定するために、引数を使って AMF 呼び出しを実行します。MonitorProcesses に設定され、オフライン監視用 AMF に登録される StartProgram とその他のプロセスは、オフラインノードでの開始を妨げられます。これは初期段階での同時性違反を防ぐのに役立ちます。
- getcksum
  指定したプログラムのチェックサムを返します。

状態の定義

ONLINE

PidFiles 属性と MonitorProcesses 属性で指定したすべてのプロセスが動作中であることを示します。または、MonitorProgram から ONLINE が返されたことを示します。

OFFLINE

PidFiles 属性または MonitorProcesses で指定したプロセスの少なくとも 1 つが動作していないことを示します。または、MonitorProgram が OFFLINE が返されたことを示します。

UNKNOWN

アプリケーションの状態が確定不能か、または設定が無効であることを示します。

2

汎用アプリケーションエージェントの設定

この章では以下の項目について説明しています。

ApplicationHA でのアプリケーション監視の設定について

この章では、仮想化環境内の ApplicationHA でアプリケーション監視を設定する手順について説明します。

先に進む前に、次の点を確認してください。

- 仮想マシンで監視するアプリケーションを設定するには、Symantec ApplicationHA 設定ウィザードを使います。
- Symantec ApplicationHA 設定ウィザードは、VOM（Veritas Operations Manager）Management Server コンソールの[Symantec High Availability]ビューで[アプリケーション監視の設定（Configure Application Monitoring）]をクリックすると起動します。
- このリリースでは、このウィザードから監視を設定できるのは、仮想マシンあたり 1 つのアプリケーションのみです。
  このウィザードを使って別のアプリケーションを設定するには、まず既存のアプリケーション監視を設定解除する必要があります。

- ウィザードでアプリケーションの監視を設定したら、VCS（Symantec Cluster Server）コマンドを使って、同じ仮想マシンにある他のアプリケーションの監視を設定できます。詳しくは、次の TechNote を参照してください。
  http://www.symantec.com/docs/TECH159846
- アプリケーションに監視を設定した後で、アプリケーションの別のインスタンスを作成しても、これらの新しいコンポーネントは既存の設定での監視対象にはなりません。このような場合は、まず既存の設定を解除し、次にウィザードを使ってアプリケーションを再設定する必要があります。その後、すべてのインスタンスを監視対象として選択できます。

アプリケーションのアプリケーション監視を設定する前に

仮想マシンの アプリケーション のアプリケーション監視を設定する前に、次の手順を完了してください。

- VOM（Veritas Operations Manager）Management Server をインストールします。VOM での作業について詳しくは、『Symantec ApplicationHA ユーザーズガイド』を参照してください。［Symantec High Availability］ビューへのアクセスについて詳しくは、p.11 の「［Symantec High Availability］ビューへのアクセス」を参照してください。を参照してください。
- 監視する必要がある仮想マシンに ApplicationHA ゲストコンポーネントをインストールします。
- アプリケーション監視を設定する仮想マシンにログオンしているユーザーに、ApplicationHA のアプリケーション監視の設定（管理者）権限を割り当てます。
- 仮想マシンで監視したいアプリケーションとそれに関連するコンポーネントをインストールします。
- ファイアウォールを設定した場合は、ファイアウォールの設定で必ず ApplicationHA インストーラ、ウィザード、サービスで使うポートにアクセスすることを許可します。使われるポートとサービスの一覧については、『Symantec ApplicationHA インストールガイド』を参照してください。

［Symantec High Availability］ビューへのアクセス

KVM 環境で実行している仮想マシンのアプリケーションを管理するには、VOM（Veritas Operations Manager）Management Server コンソールの［Symantec High Availability］ビューにアクセスする必要があります。

［Symantec High Availability］ビューでは、次のような管理操作を実行できます。

- アプリケーションを開始する
- アプリケーションを停止する

- アプリケーション監視を設定する
- アプリケーション監視を設定解除する
- アプリケーションハートビートを有効にする
- アプリケーションハートビートを無効にする
- 保守モードを開始する
- 保守モードを終了する

[Symantec High Availability]ビューにアクセスするには

1. VOM Management Server コンソールにログオンします。
2. [サーバー(Server)]パースペクティブを選択し、左ペインの[管理(Manage)]を展開します。
3. [組織(Organization)]または[未分類のホスト(Uncategorized Hosts)]を展開して仮想マシンに移動します。
4. 必要な仮想マシンを右クリックし、[ApplicationHA の管理(Manage ApplicationHA)]をクリックします。

   [Symantec High Availability]ビューが表示されます。

アプリケーション に対するアプリケーション監視の設定

仮想マシンでアプリケーションの監視を設定するには、次の手順を実行します。

アプリケーションのアプリケーション監視を設定するには

1. Veritas Operations Manager Management Server コンソールの[Symantec High Availability]ビューで、[アプリケーション監視の設定(Configure Application Monitoring)]をクリックします。

   Symantec ApplicationHA 設定ウィザードが起動します。
2. [ようこそ(Welcome)]画面の情報を確認して、[次へ(Next)]をクリックします。

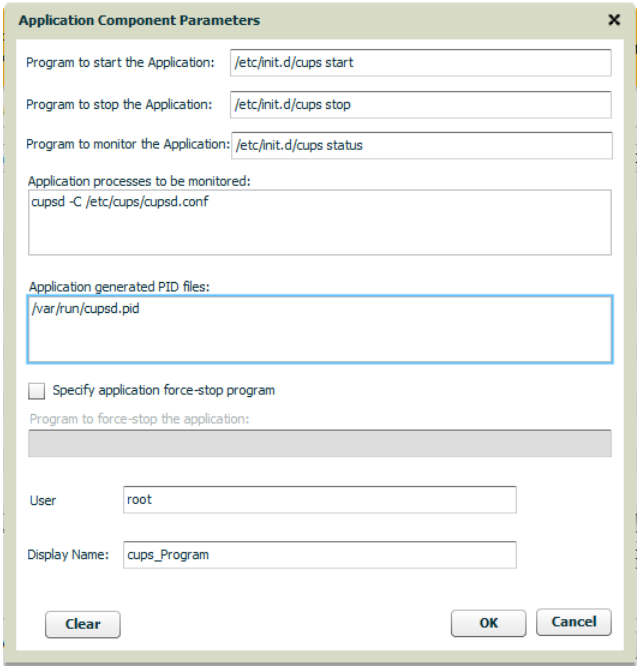
   ウィザードに、システムでサポートされるすべてのアプリケーションが表示されます。
3. [カスタムアプリケーション(Custom Application)]を選択して[次へ(Next)]をクリックします。

   メモ: ウィザードで、IMF(Intelligent Monitoring Framework)を使ってカスタムアプリケーションを監視するように ApplicationHA を設定します。

   プログラム選択の画面が表示されます。

4 監視するアプリケーションコンポーネントを指定するには、[コンポーネントの追加(Add Component)]をクリックします。

[アプリケーションコンポーネントパラメータ(Application Component Parameters)]ダイアログボックスが表示されます。



5 次の詳細を指定して、監視するコンポーネントを設定します。

- 起動プログラム(Start program): 起動プログラムスクリプトの絶対パス
- 停止プログラム(Stop program): 停止プログラムスクリプトの絶対パス
- 強制停止プログラム(Force-stop program): 強制的にアプリケーションを停止するプログラムスクリプトの絶対パス
- 少なくとも次の項目を 1 つ以上指定します。
  - 監視プログラム(Monitor program): 監視プログラムスクリプトの絶対パス
  - 監視するアプリケーション関連の処理(Application-related processes to monitor): 監視する必要があるアプリケーション処理の名前
  - アプリケーション生成 PID ファイル(Application-generated PID files): アプリケーションのプロセス ID(PID)ファイルのパス名

- ユーザー(User): ユーザー名。アプリケーションを設定する仮想マシンの適切な権限がある有効なユーザーを指定します。指定しないと、アプリケーションの監視に失敗することがあります。

[コンポーネント(Component)]フィールドからコンポーネントを削除するには、[削除(Remove)]アイコンを使います。

メモ: ウィザードは、指定されたコンポーネントの[表示名(Display Name)]を自動的にポピュレートします。ただし、このフィールドの情報は編集できます。また、汎用エージェントを設定する仮想マシン(ゲスト)で、有効なクレデンシャルと適切な権限を持つユーザーが指定されていることを確認します。指定しないと、アプリケーションの監視に失敗することがあります。

---

6 [OK]をクリックします。

指定したコンポーネントがプログラムの選択画面に表示されます。

7 監視するアプリケーションコンポーネントをさらに指定する場合は、手順 4 から手順 6 を繰り返します。指定しない場合は、[次へ(Next)]をクリックします。

手順 4 から 7 で指定したその他のコンポーネントがコンポーネントリストに表示されます。

[開始または停止順序の定義(Define Start Stop Order)]画面が表示されます。以前選択したコンポーネントが画面に表示されます。

8 親コンポーネントのリストからコンポーネントをクリックします。

9 指定した親コンポーネントとの依存関係を設定するには、コンポーネントリストでコンポーネントをクリックします。すべての親コンポーネントに対してこの手順を繰り返します。

10 [設定(Configure)]をクリックします。

ウィザードはアプリケーションの監視設定タスクを実行します。ApplicationHA の設定画面に各タスクの状態が表示されます。

すべてのタスクを完了したら、[次へ(Next)]をクリックします。

メモ: 設定タスクが失敗した場合は、[診断情報(Diagnostic information)]をクリックして失敗の詳細を確認します。

次に、ウィザードを再度実行してアプリケーション監視を設定する必要があります。

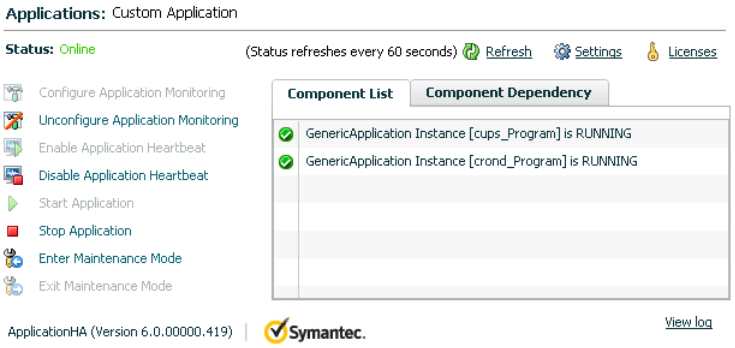
---

11 [完了(Finish)]をクリックしてウィザードを終了します。

これで、監視するアプリケーションの設定が完了します。p.23 の「init とカスタムの処理の設定例」を参照してください。

12 仮想マシンの設定済みのアプリケーションの状態を表示するには、Veritas Operations Manager Management Server コンソールで適切な仮想マシンを右クリックして、[ApplicationHA の管理(Manage ApplicationHA)]をクリックします。

[Symantec High Availability]ビューが表示されます。

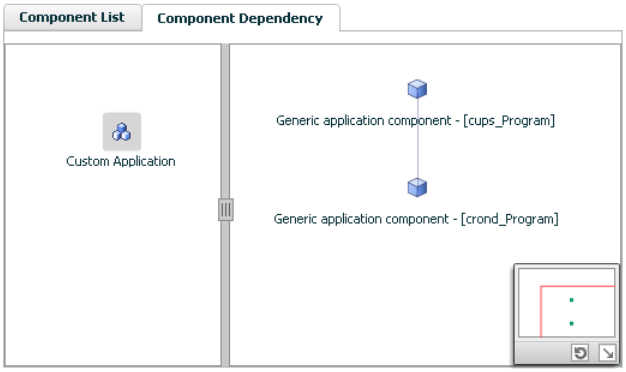


デフォルトでは、[コンポーネントリスト(Component List)]タブが表示されます。このタブには、設定済みのアプリケーションの各コンポーネントと、各コンポーネントの状態の説明が表示されます。

Veritas Operations Manager を使ったアプリケーションの表示と管理について詳しくは、『Symantec ApplicationHA ユーザーズガイド』を参照してください。

13 監視対象アプリケーションのコンポーネントの依存関係を表示するには、[コンポーネントの依存関係(Component Dependency)]タブをクリックします。

コンポーネントの依存関係を示すグラフが表示されます。



このグラフは、設定済みのアプリケーションに対して、選択したコンポーネントグループ(アプリケーションまたは相互に関連するコンポーネントのグループ)とそのコンポーネントの依存関係を表します。左ペインには、コンポーネントグループや設定済みのアプリケーションが表示されます。右ペインには、選択したコンポーネントグループまたはアプリケーションのコンポーネントが表示されます。

設定済みのアプリケーションに対するコンポーネントの依存関係の表示について詳しくは、『Symantec ApplicationHA ユーザーズガイド』を参照してください。

A

リソースタイプの定義

この付録では以下の項目について説明しています。

リソースタイプの定義

```
type Application (
        static keylist SupportedActions = { "program.vfd", "user.vfd",
        "cksum.vfd", getcksum, propcv }
        static str ArgList[] = { User, StartProgram, StopProgram,
        CleanProgram, MonitorProgram, PidFiles, MonitorProcesses,
        EnvFile, UseSUDash, State, IState }
        static int IMF{} = { Mode = 3, MonitorFreq = 1,
        RegisterRetryLimit = 3 }
        static str IMFRegList[] = { MonitorProcesses, User, PidFiles,
        MonitorProgram, StartProgram, LevelTwoMonitorFreq }
        static int LevelTwoMonitorFreq = 1
        str User = root
        str StartProgram
        str StopProgram
        str CleanProgram
        str MonitorProgram
        str PidFiles[]
        str MonitorProcesses[]
        str EnvFile
        boolean UseSUDash = 0
)
```

エージェントの属性

表 A-1 必須属性

必須属性	説明
StartProgram	アプリケーションを起動する実行ファイル。この属性は、実行ファイルの絶対パスを指定します。該当のコマンドライン引数は実行ファイルの名前の後に続きます。実行ファイルのパスと引数はスペースで区切ります。 たとえば、次のような StartProgram 属性の場合: /usr/sbin/vxnotify -g dg00 -m >> /var/log/vxnotify.log vxnotify がブロッキングコマンドの場合、次のように設定します。 /usr/sbin/vxnotify -g dg00 -m >> /var/log/vxnotify.log & **メモ:** エージェントは StartProgram 実行可能ファイルの戻り値をログに記録します。エージェントはゼロ以外の戻り値を実行のエラーと扱わず、リソースをオンラインにします。 **メモ:** この文字列では、開始と終了の (){} 波カッコの記号を使わないでください。 **メモ:** スクリプトでは、0 から 255 までの間の戻り値を指定します。 データ形式と値のタイプ: 文字列 - スカラー 例: "/usr/sbin/sample_app start"
StopProgram	アプリケーションを停止する実行ファイル。この属性は、実行ファイルの絶対パスを指定します。該当のコマンドライン引数は実行ファイルの名前の後に続きます。 **メモ:** エージェントは StopProgram 実行可能ファイルの戻り値をログに記録します。エージェントはゼロ以外の戻り値を実行のエラーと扱わず、リソースをオフラインにします。 **メモ:** この文字列では、開始と終了の (){} 波カッコの記号を使わないでください。 **メモ:** スクリプトでは、0 から 255 までの間の戻り値を指定します。 データ形式と値のタイプ: 文字列 - スカラー 例: "/usr/sbin/sample_app stop"

必須属性	説明
次のいずれか(1 つ以上)の属性 ■ MonitorProcesses ■ MonitorProgram ■ PidFiles	p.19 の 表 A-2 を参照してください。

表 A-2 オプション属性

オプション属性	説明
CleanProgram	アプリケーションを強制的に停止する実行ファイル。この属性は、実行ファイルの絶対パスを指定します。該当のコマンドライン引数は実行ファイルの名前の後に続きます。実行ファイルのパスと引数はスペースで区切ります。 **メモ:** ストレージの接続が失われた場合にアプリケーションを停止する適切なアクションを ApplicationHA が実行できるように、CleanProgram をローカルストレージに配置することをお勧めします。 **メモ:** CleanProgram 実行可能ファイルがゼロ以外の値を返す場合、エージェントはクリーンエラーとして扱い、リソースはエラーになりません。 データ形式と値のタイプ: 文字列 - スカラー 例: "/usr/sbin/sample_app stop"
MonitorProcesses	監視とクリーンの対象とするプロセスのリスト。各プロセス名は、実行ファイルの名前です。パスを指定して実行ファイルを起動する場合は、実行ファイルの名前を絶対パスで指定する必要があります。 プロセス名は、`ps -ef` コマンドで表示される名前を指定します。 データ形式と値のタイプ: 文字列 - ベクトル 例: "nmbd"

オプション属性	説明
MonitorProgram	アプリケーションを監視する実行ファイル。この属性は、実行ファイルの絶対パスを指定します。該当のコマンドライン引数は実行ファイルの名前の後に続きます。実行ファイルのパスと引数はスペースで区切ります。 MonitorProgram は値を返すことができ、OFFLINE 値は 100 または 1、ONLINE 値は信頼性レベルに応じて 101 から 110 の範囲（110 は信頼性レベルが 100%）または 0 になります。その他の値はすべて UNKNOWN です。 **メモ:** この文字列では、開始と終了の（{}）波カッコの記号を使わないでください。 MonitorProgram が設定されていても利用可能ではない場合、リソースの状態は次のようになります。 ■ OFFLINE - リソースが OFFLINE 状態で、アクションを待機していない場合 ■ UNKNOWN - リソースがその他の状態であるか、または何らかのアクションを待機している場合 データ形式と値のタイプ: 文字列 - スカラー 例: "/usr/sbin/sample_app_monitor all"
PidFiles	監視とクリーンの対象とするプロセスのプロセス ID（PID）が含まれている PID ファイルのリスト。これらのファイルは、アプリケーションによって生成されます。各 PID ファイルには、監視下にある PID が 1 つ含まれます。属性値には、絶対パスで各 PID ファイルを指定します。 プロセス ID は、プロセスの再起動時に変更される可能性があります。PID ファイルの更新に時間がかかると、エージェントの monitor 関数から不正な結果が返される場合があります。誤った結果になった場合は、リソース定義の ToleranceLimit の値を大きくします。 データ形式と値のタイプ: 文字列 - ベクトル
User	StartProgram、StopProgram、MonitorProgram、CleanProgram を実行するためのユーザー ID。MonitorProcesses で指定されたプロセスは、この属性値で指定されたユーザー環境で実行される必要があります。監視時に、プロセスが指定されたユーザー環境で実行されていることを確認します。 データ形式と値のタイプ: 文字列 - スカラー **メモ:** 設定されたユーザーが存在しないか、設定されたユーザーに対してホームディレクトリが設定されていない場合、リソースの状態は UNKNOWN になります。 デフォルト: root 例: user1

オプション属性	説明
EnvFile	StartProgram、StopProgram、MonitorProgram、CleanProgram のいずれかを実行する前に供給される必要がある環境ファイル。 データ形式と値のタイプ: 文字列 - スカラー デフォルト: "" **メモ:** 設定したユーザーのデフォルトシェル構文に EnvFile が従っていることを確認してください。 例: /home/username/envfile
UseSUDash	この属性の値が 0 の場合、エージェントは、StartProgram、StopProgram、MonitorProgram、CleanProgram のエージェント関数を実行する前に `su user` コマンドを実行します。 この属性の値が 1 の場合、エージェントは、StartProgram、StopProgram、MonitorProgram、CleanProgram のエージェント関数を実行する前に `su - user` コマンドを実行します。 データ形式と値のタイプ: ブール - スカラー デフォルト: 0 例: 1

表 A-3 に汎用アプリケーションエージェントの内部属性を示します。

表 A-3 汎用アプリケーションのエージェントの内部属性

内部属性	説明
IMF	このリソースタイプレベルの属性は、汎用アプリケーションエージェントがインテリジェントなリソースの監視を実行する必要があるかどうかを決定します。 この属性は次のキーを含んでいます。 ■ Mode: この属性を定義して、インテリジェントなリソースの監視を有効または無効にします。 有効な値は次のとおりです: ■ 0 - インテリジェントなリソースの監視を実行しない ■ 1 - オフラインリソースについてはインテリジェントなリソースの監視を実行し、オンラインリソースについてはポーリングベースの監視を実行する ■ 2 - オンラインリソースについてはインテリジェントなリソースの監視を実行し、オフラインリソースについてはポーリングベースの監視を実行する ■ 3 - オンラインリソースとオフラインリソースの両方についてインテリジェントなリソースの監視を実行する デフォルト: 3 ■ MonitorFreq: このキー値は、エージェントが監視エージェント機能を呼び出す頻度を指定します。このキーの値は整数です。 デフォルト: 1 エージェントがポーリングベースの監視とインテリジェントなリソースの監視の両方を実行する必要がある場合は、このキーをゼロ以外の値に設定できます。 値が 0 の場合、エージェントはポーリングベースのプロセスチェック監視を実行しません。リソースが AMF カーネルドライバに登録されると、エージェントは次のように監視エージェント機能を呼び出します。 ■ オンラインリソースの場合は（MonitorFreq x MonitorInterval）秒おき ■ オフラインリソースの場合は（MonitorFreq x OfflineMonitorInterval）秒おき ■ RegisterRetryLimit: インテリジェントなリソース監視を有効にすると、エージェントは imf_register エージェント関数を呼び出して、リソースを AMF カーネルドライバに登録します。RegisterRetryLimit キーの値によって、エージェントがリソースの登録を再試行する必要がある回数が決まります。エージェントが、指定された制限内にリソースを登録できない場合、リソースの状態が変化するか、Mode キーの値が変更されるまで、インテリジェントな監視は無効になります。 デフォルト: 3

B

設定例

この付録では以下の項目について説明しています。

- init とカスタムの処理の設定例

init とカスタムの処理の設定例

この項では、Symantec ApplicationHA を使って、httpd や sendmail などの init プロセスを設定するための手順と、高可用性を実現するためにカスタムプロセスを設定する手順を説明します。

サンプルのカスタムアプリケーションコンポーネント MyComponent1 の開始、停止、強制停止、監視を、`startMyComponent1`、`stopMyComponent1`、`forcestopMyComponent1`、`monitorMyComponent1` のそれぞれのスクリプトを使って行います。`monitorMyComponent1` スクリプトは、アプリケーションの MonitorProgram 属性に準拠するために書き込まれます。

p.18 の「エージェントの属性」を参照してください。

init プロセスのアプリケーション監視を設定するには

1. Veritas Operations Manager Management Server コンソールの［Symantec High Availability］ビューで、［アプリケーション監視の設定（Configure Application Monitoring）］をクリックします。

   Symantec ApplicationHA 設定ウィザードが起動します。

2. ［ようこそ（Welcome）］画面の情報を確認して、［次へ（Next）］をクリックします。

   ウィザードに、システムでサポートされるすべてのアプリケーションが表示されます。

3. ［カスタムアプリケーション（Custom Application）］を選択して［次へ（Next）］をクリックします。

   プログラム選択の画面が表示されます。

4 監視する httpd の詳細を指定するには、[コンポーネントの追加(Add Component)]をクリックします。

[アプリケーションコンポーネントパラメータ(Application Component Parameters)]ダイアログボックスが表示されます。

メモ: httpd などの init プロセスでは、特別な監視スクリプトは不要です。ApplicationHA では、監視に init スクリプトの status オプションが使われます。ただし、独自のプログラムスクリプトを使用して、そのようなプロセスを監視することもできます。

5 それぞれのフィールドに次の値を入力して、[OK]をクリックします。

アプリケーションを開始するプログラム	`/etc/init.d/httpd start`
アプリケーションを停止するプログラム	`/etc/init.d/httpd stop`
アプリケーションを監視するプログラム	`etc/init.d/httpd status`

メモ: 「forcestop」オプションを選択しない場合、ApplicationHA はアプリケーションを停止するために選択したプログラムスクリプトを使います。

6 監視するアプリケーションコンポーネントを指定するには、[コンポーネントの追加(Add Component)]をクリックします。

[アプリケーションコンポーネントパラメータ(Application Component Parameters)]ダイアログボックスが表示されます。

7 それぞれのフィールドに次の値を入力します。

アプリケーションを開始するプログラム	/home/user1/myapplication/bin/startMyComponent1
アプリケーションを停止するプログラム	/home/user1/myapplication/bin/stopMyComponent1
アプリケーションを監視するプログラム	/home/user1/myapplication/bin/monitorMyComponent1
アプリケーションを強制的に停止するプログラム	/home/user1/myapplication/bin/forcestopMyComponent1
アプリケーションを管理するときに使うユーザー名	*username* デフォルトユーザー名は「root」です。

8 [OK]をクリックします。

指定したコンポーネントがプログラムの選択画面に表示されます。

9 監視するアプリケーションコンポーネントをさらに指定する場合は、手順 4 から手順 8 を繰り返します。

httpd コンポーネントと MyApplication コンポーネント間の関係を定義する場合は、[次へ(Next)]をクリックします。[開始または停止順序の定義(Define Start Stop Order)]画面が表示されます。以前選択したコンポーネントが画面に表示されます。

10 最初に httpd プログラム、次に MyApplication プログラムをオンラインにするには、親コンポーネントのリストで[startMyComponent1_Program]をクリックします。

11 コンポーネントリストで、[httpd_Program]をクリックして選択し、次に[設定(Configure)]をクリックします。

ウィザードはアプリケーションの監視設定タスクを実行します。ApplicationHA の設定画面に各タスクの状態が表示されます。すべてのタスクを完了したら、[次へ(Next)]をクリックします。

12 [完了(Finish)]をクリックしてウィザードを終了します。これで、監視するアプリケーションの設定が完了します。

C

カスタムアプリケーションのサンプルスクリプト

この付録では以下の項目について説明しています。

カスタムアプリケーションを開始、停止、監視するためのサンプルスクリプト

汎用エージェント用の独自のスクリプトを記述して、カスタムアプリケーションのオンライン化とオフライン化や、カスタムアプリケーションの状態の監視を行うことができます。また、次のサンプルスクリプトを修正し、それらをカスタムアプリケーションの開始、停止、監視に使うこともできます。

- カスタムアプリケーションを開始するためのサンプルスクリプトは次のとおりです。

```
#!/bin/sh
touch /tmp/sampleapp    # add any steps, if required
exit 0
```

サンプルの開始スクリプトをカスタムアプリケーションの必要条件に合うように修正できます。開始スクリプトを `startsampleapp` という名前で保存した場合は、カスタムアプリケーションをオンライン化するため、エージェント機能で次のコマンドを実行します。

```
su - root -c /root/customapp/startsampleapp
```

- カスタムアプリケーションを停止するためのサンプルスクリプトは次のとおりです。

```
#!/bin/sh
rm -f /tmp/sampleapp # add any steps, if required
exit 0
```

サンプルの停止スクリプトをカスタムアプリケーションの必要条件に合うように修正できます。停止スクリプトを stopsampleapp という名前で保存した場合は、カスタムアプリケーションをオフライン化するため、エージェント機能で次のコマンドを実行します。

```
su - root -c /root/customapp/stopsampleapp
```

メモ: 開始スクリプトと停止スクリプトに対する戻りコードの値は 0 である必要があります。他の戻りコードはサポートされていません。

- カスタムアプリケーションを監視するためのサンプルスクリプトは次のとおりです。

```
#!/bin/sh
APPLICATION_IS_ONLINE=110
APPLICATION_IS_OFFLINE=100
if [ -f /tmp/sampleapp ] ; then  # add any steps, if required
        exit $APPLICATION_IS_ONLINE
else
        exit $APPLICATION_IS_OFFLINE
fi
```

監視スクリプトを monitorsampleapp という名前で保存した場合は、カスタムアプリケーションを監視するため、エージェント機能で次のコマンドを実行します。

```
su - root -c/root/customapp/monitorsampleapp
```

D

カスタム監視プログラム

この付録では以下の項目について説明しています。

複数の処理を監視するカスタム監視プログラムの書き込み

アプリケーション監視設定ウィザードの[カスタムアプリケーション(Custom Application)]オプションでは、複数の処理の監視を可能にできません。ApplicationHA を使って複数の処理の監視に使うことができる監視プログラムを書き込むには、次の手順を実行します。

プロセスパス名を使ってカスタム監視プログラムを書き込むには

1. 次のコマンドを実行すると各処理が動作するかを確認します。

   ```
   ps -ef | grep ProcessName
   ```

2. すべての処理が動作したら、戻りコード 110 で監視プログラムを終了します。

   処理のいずれかが動作しない場合は、戻りコード 100 で監視プログラムを終了します。

3. シェルスクリプトにこの処理を保存し、スクリプトに実行権を割り当てます。

4. アプリケーション監視設定ウィザードの[アプリケーションコンポーネントパラメータ(Application Component Parameters)]ダイアログボックスで、監視する処理のリストの前にある[監視プログラム(Monitor Program)]フィールドにスクリプトの絶対パスを入力します。

PID ファイルを使った複数の処理を監視するカスタム監視プログラムの書き込み

アプリケーション監視設定ウィザードの[カスタムアプリケーション(Custom Application)]オプションでは、PID ファイルを使った複数の処理の監視を可能にできません。ApplicationHA で PID ファイルを使った複数の処理の監視に使うことができる監視プログラムを書き込むには、次の手順を実行します。

PID ファイルを使ってカスタム監視プログラムを書き込むには

1 次のコマンドを実行すると各 PID ファイルで指定した処理が動作するかを確認します。

```
ps -ef | grep ProcessID
```

ProcessID は PID ファイルの内容です。

2 すべての処理が動作したら、戻りコード 110 で監視プログラムを終了します。

処理のいずれかが動作しない場合は、戻りコード 100 で監視プログラムを終了します。

3 シェルスクリプトにこの処理を保存し、スクリプトに実行権を割り当てます。

4 アプリケーション監視設定ウィザードの[アプリケーションコンポーネントパラメータ(Application Component Parameters)]ダイアログボックスで、スペースで区切った PID ファイルのリストの前にある[監視プログラム(Monitor Program)]フィールドにスクリプトの絶対パスを入力します。