## Generell

Domino stellt keine wie auch immer gearteten Möglichkeiten zur Verfügung, um Serveragenten in irgendeiner Weise zu synchronisieren. Dieses Problem ist auf keine Plattform beschränkt, sondern ist symptomatisch für den Agent Manager, der als Server Task isoliert auf jeweils einem Server läuft. Es stehen keine Pipes oder Semaphoren zur Verfügung, um eine Kommunikation mit einem Agentmanager auf einem anderen Server herzustellen. Das bedeutet im einzelnen, will man eine Applikation clusterfähig machen die Server Agenten verwendet, kommt man nicht umhin Code Anpassungen im Agentencode vorzunehmen, um auszuschließen das Agenten auf den gleichen Eingabedaten laufen, was schwere Probleme wie Replikaktionskonflikte, doppelte Dokumente etc. hervorruft.

## Was geht nicht

Agenten nur auf einem Server aktivieren.
Gleiches Problem wie im nächsten Punkt!

Unterschiedliche Startzeiten für Daily, Weekly, Monthly Agenten angeben.
Dadurch könnte man Agenten entkoppeln und sie soweit voneinander starten, das sie sich nicht in die Quere kommen.

Sobald man an den Agent-Eigenschaften Änderungen vornimmt, werden diese bei der nächsten Cluster-Replikation (normalerweise innerhalb der nächsten Sekunden) in auf alle anderen Server des Clusters repliziert, die diese Applikation als Cluster Replik beinhalten. Dies gilt ebenfalls für das Feld "Run on". Die einzige Möglichkeit, die man in diesem Kontext hat, ist einen festen Runserver vorzugeben, auf dem der Agent getriggert wird. Auf allen anderen Servern läuft der Agent dann aber nicht mehr, was insbesondere bei Failovers zum Problem wird.

## Funktionalität / Vorgehensweise

Man kann eine Funktion schreiben, die auf der Basis eines "bevorzugten RunServers" operiert. Dieser Server wird im Normalfall verwendet, um den scheduled (gilt auch für New Mail, new Document) Agent auszuführen. Im Failover Fall wird der nächste verfügbare Server innerhalb des Clusters genommen, der dann zum "bevorzugten RunServer" wird und ab diesem Zeitpunkt die Ausführung des Agenten übernimmt. Nachdem die Failover Situation beendet ist, kann wieder auf den vorher gültigen Runserver umgeschaltet werden.

Soweit erst einmal die Theorie. Auf dem Weg in die Praxis gibt es jedoch einige Probleme, da es nicht so einfach ist, überhaupt eine Failover Situation festzustellen. Man kann nämlich per Lotus Script/Formula keine Abfrage auf eine anderen Server als den aktuellen machen, auf dem der Agent ausgeführt wird. Abfrage in diesem Sinne heißt ganze einfach: Öffne eine wohlbekannte Datenbank auf dem abzufragenden Server. Kann man Sie öffnen, dann ist der Server verfügbar, wenn nicht, haben wir eine Failover Situation. Eine wohlbekannte Applikation ist die aktuelle Datenbank, da sie auf dem anderen Server im gleichen Verzeichnis vorhanden sein muss.

Hierzu haben wir zwei Lösungsmöglichkeiten entwickelt. die sowohl einzeln, als auch in Kombination ausgeführt werden können, um die Sicherheit zu erhöhen.

Öffnen der wohlbekannten Applikation auf einem anderen Server des Clusters per API Funktion NSFDBOpen(<Filepath>)

Hiezu muss in der Applikation auf jedenfall alle Server des Clusters in der ACL eingetragen sein. (Normalerweise immer!) und die Server müssen unrestricted Access aufeinander haben, was in den Sicherheitseinstellungen des NAB Server Dokuments einstellbar ist.
Wenn diese Voraussetzungen erfüllt sind, dann kann man beim Start eines Agenten auf einem anderen Server nachsehen, ab man die Cluster-Replik dort öffnen kann. Wenn ja, befindet man sich in einem normalen Status, wenn nicht, dann hat man eine Failover Situation.

Ermittlung der Verfügbarkeit eines anderen Servers über die Kombination Profile/Probe-Agenten.
Ein scheduled Agent mit einem Laufzeit Intervall von 5 Minuten schreibt in ein Server spezifisches Profile (Name der Profile: Common Servername) der Applikation einen Timestamp in ein Feld "TimeStamp". Durch den Cluster-Replikator wird diese Profile innerhalb von Sekunden auf jede andere Cluster-Replik verteilt. Nimmt man nun an, das alle 5 Minuten die aktuelle Zeit in das Profile geschrieben wird, kann man davon ausgehen, das wenn dieser Wert älter als 5 Minuten + maximale Agent Laufzeit (Eintrag im NAB) ist, der Server nicht mehr läuft, wir also eine Failover Situation haben. Wenn nicht, ist der Server höchstwahrscheinlich verfügbar.
Ein wenig Unsicherheit ist da allerdings schon da, da eventuelle Power Agents mehr als 5 Minuten laufen könnten und dadurch diesen Wert verfälschen würden, daher der Sicherheitszuschlag "maximale Agent Laufzeit", da nach dieser Zeit ein Agent vom Agent Manager automatisch aus dem Speicher geworfen wird und alle anderen Agenten dann wieder dran kommen.

Nun fehlt uns nur noch ein Profile, indem die Werte "bevorzugter Server" und Server Liste des Clusters abgelegt werden. Diese Werte kann man in ein sowieso vorhandenes Applikationsprofile eintragen, oder eine separate Profile per Form generieren, wo es dem Applikationsverantwortlichen möglich ist, die Cluster-Liste zu editieren und den "bevorzugten Server" einzutragen. Dies kann allerdings auch automatisch geschehen, dann wird als bevorzugter Server eben der eingetragen, auf dem das Profile das erste mal angelegt wird bzw. angepasst wird.

Nachdem wir die nötigen Bestandteile definiert haben, kann man daraus eine Klasse zusammen bauen, die einfach nur noch in jedem Scheduled Agent als erstes Statement instanziiert wird und anschließend eine Check Methode aufgerufen wird, welche True oder False zurück liefert. Abhängig von Returnwert wird der Agent verlassen oder weiter ausgeführt. Die funktioniert in allen Fällen und bedarf ansonsten keine großen Code Änderung.

Beispiel:

```
dim runEnabled as new CheckRunClass(session.currentDatabase)
if runEnabled.Check() = False then exit
```

## Ablaufbeschreibung

Angenommen, wir haben eine Profile, in der es ein folgende Felder gibt:

| | |
|---|---|
| RunServername | (Inhalt : Server_A) |
| RunServernameOriginal | (Inhalt : leer) |
| Clustermembers | (Inhalt: Server_A, Server_B, Server_C) |
| Maximale_Agentlaufzeit | (Inhalt: 10 Minuten) |

In den Felden sind auch schon die richtigen Werte enthalten, dann könnte der Ablauf folgendermaßen beschrieben werden:

Agent X wird auf Server_B vom Agentmanager ausgeführt. Die Check Methode prüft, ob der aktuelle Server (in diesem Fall Server_B) dem RunServername entspricht. Das Ergebnis ist False, da Server_B <> Server_A ist, also darf der Agent eigentlich nicht laufen, aber es muss zusätzlich geprüft werden, ob keine Failover Situation vorliegt. Also versuchen wir mit einem NSFDBOpen("Server_A!!<Applicationpath)>") eine wohlbekannte Datenbank auf dem bevorzugten Runserver zu öffnen. Gelingt uns das, liegt eine normale Run Situation vor und der Agent darf nicht laufen, das heiß die Check() Methode liefert False zurück.

Konnte das API Kommando die Datenbank nicht öffnen, müssen wir eine Fallunterscheidung machen:

„Cannot open Database“ als Fehlermeldung -> Failover Situation

Wir setzen den aktuellen Server als RunServer und schreiben den ursprünglichen Runserver in das Feld RunServernameOrignal, warten ein paar Sekunden (per For Schleife) um sicherzustellen, das kein anderer Server das Profile innerhalb unserer Laufzeit verändert hat, lesen das Profile erneut ein, checken nochmals das Runserver Feld, stehen wir noch drin, dann können wir sicher sein, das der Cluster-Replikator in der Zwischenzeit die Änderung verteilt hat. Dadurch werden alle anderen Agenten gesperrt und wir lassen die Abarbeitung des Agentcode zu indem die Check Methode True zurück liefert.

Command is not allowed in this Context/Session

API-Calls sind aktuell nicht erlaubt, also verwenden wir das Probe verfahren. Wir öffnen das entsprechende Profile des abzufragenden Servers, ist der Timestamp älter als 5 Minuten + Maximale Agentlaufzeit, dann liegt eine Failover Situation vor und wir verfahren wie unter 1. beschrieben. Ansonsten verlassen wir die Methode Check mit False.

Wird Agent X auf Server_A gestartet, ergibt der Vergleich aktueller Server, RunServer True, die Check Methode wird sofort mit True verlassen.

Nachdem ein Failover eingetreten ist, (Ersichtlich daran, dass das Feld „RunServerOriginal“ nicht leer ist), muss natürlich bei jedem Aufruf der Check Methode, in der festgestellt wird, das wir der aktuelle Runserver sind, geprüft werden, ob der Failover noch vorhanden ist, also der alte Runserver immer noch nicht erreichbar ist. Im falle, das wir wieder normale Run Situation haben, wir der alte Runserver wieder in das Feld „RunServername“ eingetragen und der Inhalt des Feldes „RunServernameOrignal“ gelöscht. Danach liefert die Check Methode False zurück und unterbindet damit die weitere Ausführung das aktuellen Agenten.

Diese Schritt ist aber optional und kann auch unterlassen werden. Dadurch würde nur ein anderer Server bis zum nächsten Failover der bevorzugte Server sein. Hier liegt auch eine Gefahr begründet, die bei Daily, Weekly oder Monthly Agents zum Tragen kommt. Ist der Agent auf dem Original Runserver nämlich schon gelaufen, wird die Datenverarbeitung erst beim nächsten Scheduling ausgeführt, da der aktuelle Agent nicht läuft!