AtNotes Best Practices

Agenten in Lotus Notes

Version: 1.00 (05. Juni 2004)

© Copyright 2004 - Matthias TMC

INHALT

1 Über dieses Dokument ... 4
1.1 Copyright ... 4
1.2 Geltungsbereich ... 4
1.3 Designer- und Admin - Hilfe ... 4
1.4 Verwendete Symbole ... 4
2 Was sind Agenten ... 4
3 Agenten - Einstellungen ... 5
3.1 Einstellungensbereich ... 5
3.2 Name ... 5
3.3 Comment ... 5
3.4 Shared / Private ... 6
3.5 Trigger / When should this agent run? ... 6
3.5.1 Geplante periodische Auslösung (scheduled trigger) ... 6
3.5.2 Auslösung aufgrund eines Ereignisses (event trigger) ... 7
3.6 Target / Which document(s) should it act on? ... 9
3.7 Run on behalf of – Agenten im Namen eines anderen Users ausführen ... 10
3.8 Run as web user ... 10
3.9 Run in background client thread ... 11
3.10 Allow user activation ... 11
3.11 Programmiersprachen ... 12
3.11.1 Auswahl ... 12
3.11.2 Formelsprache ... 12
3.11.3 Lotus Script ... 12
4 Frontend- und Backend - Agenten ... 13
4.1 Einführung ... 13
4.2 Frontend-Klassen im Backend-Agenten ... 13
4.3 Benutzen eines Agenten mit UI-Klassen auch im Backend in R5 ... 14
4.4 Benutzen eines Agenten mit UI-Klassen auch im Backend ab R6 ... 14
5 Mit welchen Berechtigungen laufen die Agenten? ... 15
5.1 Allgemeines zur Agenten-Berechtigung (Agent permission) ... 15
5.2 Ereignisbasierende Agenten ... 15
5.3 Serverbasierende Agenten ... 15
5.4 Vom Webbrowser ausgelöste Agenten ... 15
6 Aufruf von Agenten ... 16
6.1 Aufruf von Agenten aus Formelsprache ... 16
6.1.1 ToolsRunMacro @Command ... 16
6.1.2 RunAgent @Command ... 16
6.1.3 AgentRun @Command ... 17
6.2 Aufruf von Agenten aus Lotus Script ... 17
6.2.1 NotesAgent-Klasse: Run ... 17
6.2.1.1 Syntax ... 17
6.2.1.2 Beispiele ... 18
6.2.2 NotesAgent-Klasse: RunOnServer ... 19
6.2.2.1 Syntax ... 19
7 Logging und Debugging von Agenten ... 20
7.1 Allgemeines zu Logging und Debugging von Agenten ... 20
7.2 Testen von Agenten im Designer ... 20
7.3 Ansehen des Agent Log im Designer ... 21
7.4 Logging von Agenten ... 21
7.5 Remote Debugger ... 21
8 Einschränkungen und Besonderheiten ... 22
8.1 Geplante periodische Agenten (scheduled agents) ... 22

INHALT

8.1.1 @DBColumn und @DBLookup 22
8.1.2 Lokale Agenten 22
9 Praxisbeispiele 23
9.1 Ausgehend vom Mailfile einen Agenten einer anderen DB aufrufen 23
9.1.1 Ausgangssituation 23
9.1.2 Problemstellung 23
9.1.3 Lösungsansatz 23
9.1.4 Die Umsetzung 23
9.1.5 Fazit 26
9.2 Dokumente refreshen 26
9.3 Agenten-Bausatz 27
9.3.1 Über dieses Kapitel 27
9.3.2 Script-Start 28
9.3.3 Selektionskriterien 28
9.3.4 Schleife 29
9.3.5 Bedingungen 29
9.3.6 Aktionen 30
9.3.7 Script-Ende 31
9.4 Weitere Einsatzmöglichkeiten von Agenten 32
9.4.1 Shared Actions – Gemeinsame Aktionen 32
9.5 Periodische Agenten im verborgenen Design 32
10 Links / Weiterführende Informationen 33
11 Autor 33
12 Index 34

1 Über dieses Dokument

1.1 Copyright

Diese Dokumentation darf ausschließlich als vorliegende PDF-Datei als ganzes weitergegeben werden. Dabei muss auf http://www.atnotes.de verwiesen werden.
Wer Auszüge aus dieser Dokumentation veröffentlichen will, muss sich die Genehmigung des Autors (Kontakt via E-Mail: atnotes@gmx.de) einholen.

1.2 Geltungsbereich

Dieses Dokument ist für alle Lotus Notes Versionen ab Release 5 gültig.

1.3 Designer- und Admin - Hilfe

Es ist nicht als Ersatz oder Umformulierung der Designer- und Adminhilfe zu verstehen – es wird hier auch an den entsprechenden Stellen auf die Hilfe verwiesen.

1.4 Verwendete Symbole

Neuerung ab R6

Wichtiger Hinweis

Zusätzliche Information

2 Was sind Agenten

Agenten sind grundsätzlich eigenständige Programme zur Ausführung von Aufgaben. Dabei stehen vielfältige Selektionskriterien zur Verfügung, um bestimmte Dokumente abzuarbeiten. Agenten können jedoch auch unabhängig von einer Dokumentenauswahl definiert werden.

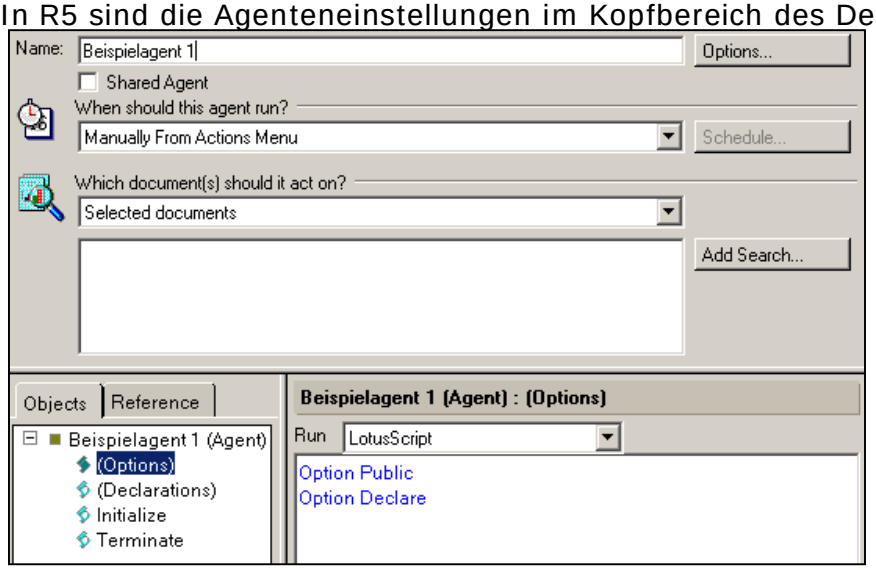
Agenten können wie folgt gestartet werden:

- durch einen User
- Periodisch geplant (‚scheduled'), z.B. jede Stunde, von Mo-Fr jeweils um 01:00 Uhr Nachts, etc.
- durch das Auftreten eines Ereignisses
- durch einen Web-Browser

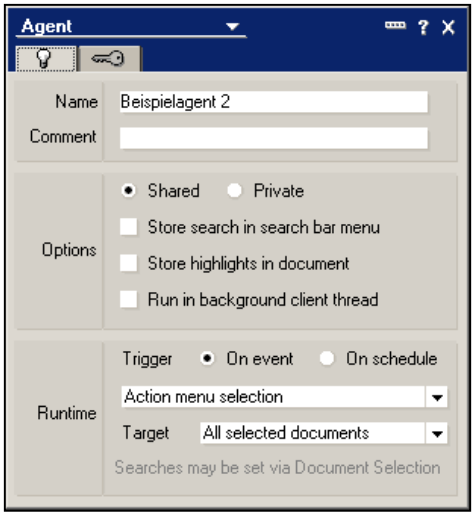
3 Agenten - Einstellungen

3.1 Einstellungensbereich

In R5 sind die Agenteneinstellungen im Kopfbereich des Designers:



Ab R6 gibt es ein eigenes Einstellungsfenster („Agent properties box“):



3.2 Name

Dies ist der Name des Agenten.

3.3 Comment

Dies ist eine optionale Angabe. Hier kann der Agent kommentiert werden.

3.4 Shared / Private

Ein *Shared Agent* ist öffentlich verfügbar und kann auch durch andere User ausgeführt werden.
Ein *Private Agent* ist ausschließlich für den Ersteller des Agenten verfügbar.

In R5 kann diese Eigenschaft nach dem ersten Speichern nicht mehr geändert werden! Um diese Eigenschaft in R5 zu ändern, muss ein neuer Agent erstellt werden und der Code und die Einstellungen vom bisherigen Agenten manuell in den neu erstellten Agenten übertragen werden.
Ab R6 kann diese Eigenschaft jederzeit geändert werden.

3.5 Trigger / When should this agent run?

3.5.1 Geplante periodische Auslösung (scheduled trigger)

Dies sind geplante, periodisch gestartete Agenten.

Alle Agenten mit geplanter periodischer Auslösung sind serverbasierende und somit Backend-Agenten – d.h. hier sind keine Frontend-Klassen und Formeln erlaubt (siehe auch Kap. 4 Frontend- und Backend - Agenten).

Trigger	Beschreibung	Berechtigung
More than once a day	Ein Intervall kleiner als 24 Stunden (von 1 Minute bis 11 h 59 Min.). Die Einstellung sollte mit Bedacht vorgeommen werden. Ein Agent, der z.B. alle 5 Minuten läuft, kann die Serverperformance sehr negativ beeinflussen!	➢ Berechtigungen die der Unterzeichner des Agenten hat (ist normalerweise der User, der den Agenten zuletzt gespeichert hat) ➢ Oder: Berechtigungen die der User hat, der in der „On behalf of“ – Option des Agenten eingetragen ist (siehe Kap. 3.7)
Daily	Täglich zur eingestellten Uhrzeit.	
Weekly	Wöchentlich – dabei gibt man das Startdatum und die Startuhrzeit als Basis vor.	
Monthly	Monatlich – dabei gibt man das Startdatum und die Startuhrzeit als Basis vor.	
Never	Keine Auslösung. Diese Einstellung ist z.B. sinnvoll, wenn man den Agenten programmatisch aufruft. Wobei ich in diesem Fall den Trigger „Agent List Selection / Manually From Agent List” empfehle.	

Bei R5 Lautet die Bezeichnung im Auswahldialog „On Schedule [Trigger]“, also „On Schedule Daily“, etc.

Generell gilt zu beachten, dass ein Agent, der beispielsweise auf ‚Daily’ mit Startzeit „01:00“ eingestellt ist, meist nicht exakt um 01:00 Uhr anläuft. Die Startzeit kann durchaus um ein paar Minuten abweichen.
Außerdem läuft ein Agent mit dem Trigger , More than once a day’ und dem Intervall „1 Minute“ nicht wirklich jede Minute. Hier verstreichen mindestens 3-4 Minuten zwischen den einzelnen Läufen.

3.5.2 Auslösung aufgrund eines Ereignisses (event trigger)

Diese Agenten werden durch ein bestimmtes Ereignis ausgelöst.

Trigger	Beschreibung	Berechtigung
Agent menu selection	(R5-Bezeichnung: Manually From Actions Menu) Damit erscheint der Agent im Menü ‘Actions’ der Datenbank und kann von dort aus aufgerufen werden.	Der Agent läuft mit der Berechtigung des jeweiligen Users.
Agent list selection	(R5-Bezeichnung: Manually From Agent List) Diese Einstellung verwendet man, wenn man den Agenten programmatisch aufruft.	Der Agent läuft mit der Berechtigung des Codes, mit dem dieser Agent aufgerufen wird. Wurde der ursprüngliche Code von einem User ausgeführt (z.B. über eine Aktionsschaltfläche mit dem @Command-Befehl [ToolsRunMacro]), so läuft der Agent mit dem Recht des jeweiligen Users.
Before new mail arrives	Diese Einstellung ist prinzipiell nur für Mail-Datenbanken geeignet. Damit werden Emails verarbeitet, <u>bevor</u> diese in Ansichten und Ordner in der Datenbank erscheinen. Agenten mit dieser Einstellung verwendet man z.B., um neue E-Mails nach bestimmten Kriterien (Subject-Text, etc.) zu bearbeiten (in Ordner ablegen etc.). Es ist nur 1 Agent mit diesem Trigger in einer Datenbank erlaubt.	Dieser Agent ist serverbasierend, die Berechtigungen sind in Kap. 5.3 (Serverbasierende Agenten) ersichtlich.
After new mail arrives	Diese Einstellung ist prinzipiell nur für Mail-Datenbanken geeignet. Damit werden Emails verarbeitet, <u>nachdem</u> diese in der Datenbank erscheinen.	Dieser Agent ist serverbasierend, die Berechtigungen sind in Kap. 5.3 (Serverbasierende Agenten) ersichtlich.

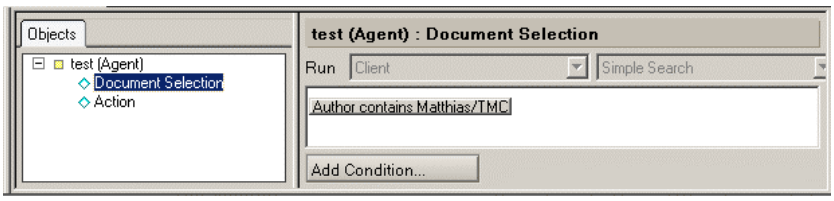
Trigger	Beschreibung	Berechtigung
	I.d.R. wird dieser Agent erst nach ein paar Minuten, nachdem eine neue E-Mail angekommen ist, gestartet.	
After documents are created or modified	(R5-Bezeichnung: If Documents Have Been Created or Modified) Dieser Agent wird ausgelöst, sobald ein neues Dokument erstellt oder ein bestehendes Dokument bearbeitet (gespeichert) wurde. Es können hier auch Zeitangaben vorgegeben werden. Der Agent startet nicht direkt, sondern oft erst einige Minuten nach dem Ereignis.	Dieser Agent ist serverbasierend, die Berechtigungen sind in Kap. 5.3 (Serverbasierende Agenten) ersichtlich.
When documents are pasted	(R5-Bezeichnung: If Documents Have Been Pasted) Der Agent wird gestartet, sobald ein Dokument in die Datenbank über die Zwischenablage eingefügt wird (Strg+V). Sind mehrere Agenten mit diesem Trigger in der Datenbank, so werden die Agenten in alphabetischer Reihenfolge anhand Agentenname ausgeführt.	Dieser Agent ist client-basierend, d.h. er läuft mit der Berechtigung des User, der ein Dokument via Zwischenablage in die Datenbank einfügt.

3.6 Target / Which document(s) should it act on?

Hier stellt man ein, für welche Dokumente der Agent gelten soll.

Target	Beschreibung
All documents in the database	Alle Dokumente in der Datenbank.
All new and modified documents since last run	Alle neuen und geänderten Dokumente in der Datenbank. Wenn man diese Einstellung verwendet, muss unbedingt die ‚UpdateProcessedDoc Methode' der NotesSession -Klasse verwendet werden, da sonst die vom Agenten bearbeiteten Dokumente beim nächsten Lauf immer noch als „modified" gelten und diese dann erneut bearbeitet werden. Ausserdem werden bei erneutem Aktivieren (oder wenn der Agent verändert wird) beim nächsten Lauf wieder alle Dokumente bearbeitet.
All unread documents in view	Alle ungelesenen Dokumente in der Ansicht. Die ungelesenen Dokumente beziehen sich dabei auf die Person, die den Agenten gemäß Notes-Logik ausführt (siehe Kap. 5 - Mit welchen Berechtigungen laufen die Agenten?).
All documents in view	Alle Dokumente der aktuell geöffneten Ansicht.
Selected documents	Alle ausgewählten Dokumente. Um die ausgewählten Dokumente zu bearbeiten, verwendet man die UnprocessedDocuments Eigenschaft der NotesDatabase (aktuelle Datenbank), um diese in eine Collection zu setzen.
None	(R5-Bezeichnung: Run once (@Commands may be used)) Auswahlkriterien für diese Einstellung sind z.B.: ➢ Man will nach einer eigenen Logik bestimmte Dokumente behandeln ➢ Man nutzt @Commands in diesem Agenten ➢ Eine View ist nicht geöffnet (also Backend-Agent)

Weitere Eingrenzungen können im ‚Document Selection' – Bereich vorgenommen werden, indem man auf den Button „Add Condition" klickt.

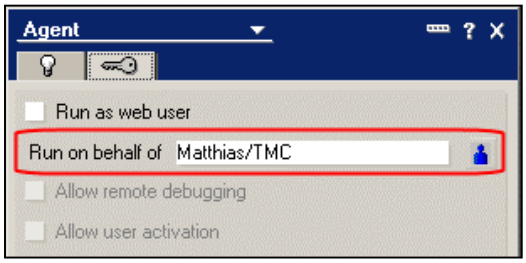


In R5 befindet sich dieser Bereich direkt unter der Einstellung ‚Which document(s) should it act on?', und der Button zum Hinzufügen neuer Bedingungen heißt hier „Add Search":

Details siehe ⇨ Designerhilfe.

3.7 Run on behalf of – Agenten im Namen eines anderen Users ausführen

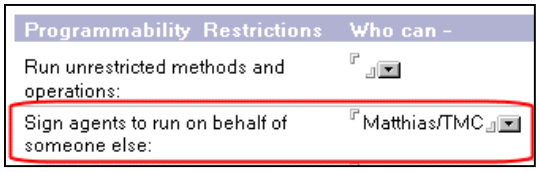
Die Option “Run on behalf of” ist ein neues Feature ab R6:



Wird hier eine Person eingetragen, so wird der Agent mit den Rechten der eingetragenen Person ausgeführt – und nicht mit den Rechten der Person, die den Agenten gespeichert hat. D.h. dem Agent wird vorgegeben, dass die eingetragene Person den Agenten signiert hat.

Dies kann beispielsweise bei E-Mail - Agenten praktisch sein – damit als Absender nicht der Designer erscheint sondern die gewünschte Person.

Der Designer (also die Person die den Agenten speichert und somit signiert) muss im Security-Tab des Server-Dokumentes im Domino Directory im folgenden Feld eingetragen sein:

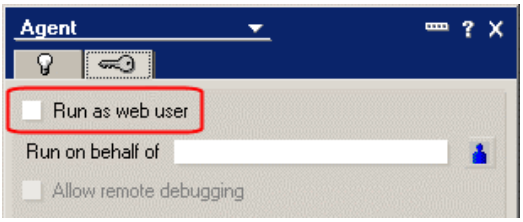


Steht der Unterzeichner des Agenten nicht in diesem Feld, dann läuft der Agent beim Ausführen auf einen Fehler.

Details zu „Run on behalf of“ siehe ⇨ R6 Designerhilfe.

3.8 Run as web user

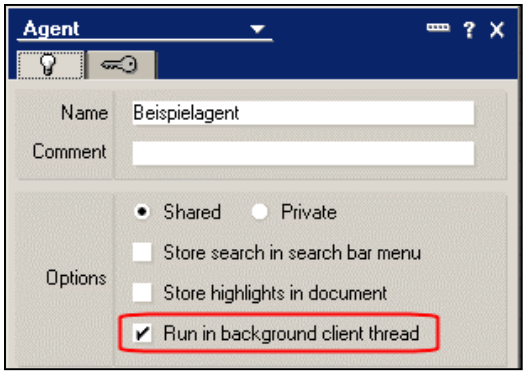
“Run as web user” ist ein neues Feature ab R6:



Wenn diese Option ausgewählt ist, läuft der Agent mit den Berechtigungen des authentifizierten Users.

3.9 Run in background client thread

Diese Option ist ebenfalls ein neues Feature ab R6:



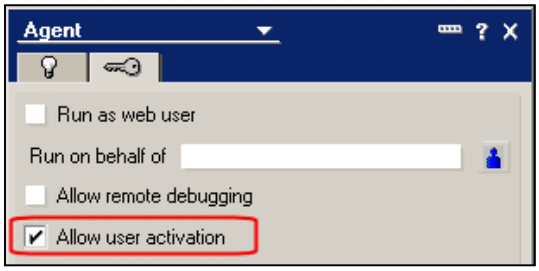
Hiermit wird ein Agent im Hintergrund ausgeführt und der User kann im Client weiterarbeiten während der Agent noch läuft.

Diese Option ist nur für den On event - Agenten „Action menu selection“ verfügbar.

3.10 Allow user activation

Generell werden periodische Agenten, die ein User aktiviert, automatisch mit der ID des Users signiert.

Ab R6 gibt es allerdings folgende Option für periodische Agenten:

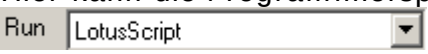


Ist diese Option aktiviert, wird ein Agent nicht signiert beim Aktivieren. Dabei muss der User zum Aktivieren von Agenten mindestens Editor-Rechte haben wenn diese Option aktiviert ist.

3.11 Programmiersprachen

3.11.1 Auswahl

Hier kann die Programmiersprache des Agenten ausgwählt werden:

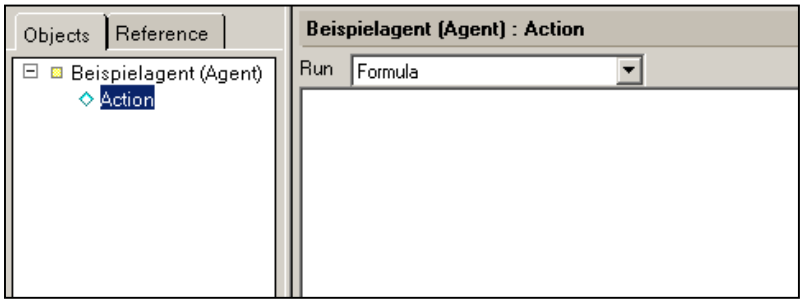


In dieser Dokumentation beschränken wir uns auf Formelsprache und Lotus Script.

Von einfachen Aktionen („Simple actions“) raten wir ab, da hier diverse Probleme auftreten können.

3.11.2 Formelsprache

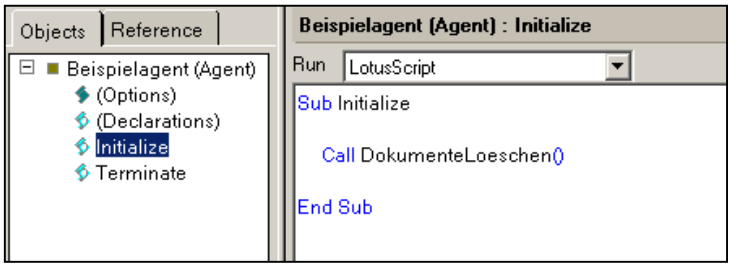
Hier findet man den gewöhnlichen Programmierbereich vor:



3.11.3 Lotus Script

Es gibt in Agenten nur 2 Events: Initialize und Terminate. Weitere (UI-basierende) Events sind nicht verfügbar.

Den auszuführenden Code übernimmt man in den Initialize – Event. Hierbei können auch andere Funktionen und Subroutinen aufgerufen werden.



4 Frontend- und Backend - Agenten

4.1 Einführung

Agenten in Lotus Script und Formelsprache können sowohl als Frontend-Agent als auch als Backend-Agent erstellt werden.

Backend- bzw. serverbasierende Agenten sind geplante periodische Agenten (scheduled agents) oder Agenten die mit der `RunOnServer` Methode (siehe Kap. 6.2.2) aufgerufen werden.

	Frontend-Agent	Backend- / server-basierter Agent
Lauf	Im Notes-Client	Auf dem Server
Erlaubte Lotus Script Klassen	Frontend- und Backend – Klassen [1])	Nur Backend-Klassen
Erlaubte Formeln	Frontend- und Backend – Formeln [2])	Nur Backend-Formeln
Zugriff [3])	Läuft mit den Rechten des Users, der den Agenten auslöst	Läuft generell mit den Rechten des Users, der den Agenten signiert hat. Zusätzlich gibt es die Möglichkeit, im Domino Directory (Security-Tab im Server-Dokument) spezielle Sicherheits-Einstellungen vorzunehmen. Details siehe ⇨ Administrator-Hilfe.

[1]) Frontend-Klassen sind alle Lotus Script - Klassen, deren Name ‚UI' enthält, also z.B. `NotesUIWorkspace`, `NotesUIDocument`, etc.
[2]) Frontend-Formeln sind `@Prompt`, `@Dialogbox` und alle `@Command`s.
[3]) Siehe auch Kap. 3.5.2 (Auslösung aufgrund eines Ereignisses (event trigger))

4.2 Frontend-Klassen im Backend-Agenten

In R5 laufen Backend-Agenten, die UI-Klassen enthalten, bereits beim Laden auf einen Fehler und diese Agenten werden erst gar nicht ausgeführt – ein einziges Dimmen einer UI-Klasse führt hier schon zum Abbruch. Fehler die bereits beim Laden des Agenten auftreten sind schwierig zu debuggen: diese erscheinen zwar auf der Serverkonsole und im Server-Log („Unknown LotusScript error" oder „Error loading USE or USELSX module: ...."), können aber programmatisch nicht abgefangen werden.

4.3 Benutzen eines Agenten mit UI-Klassen auch im Backend in R5

⇨ Siehe auch Kap. 4.4 (Benutzen eines Agenten mit UI-Klassen auch im Backend ab R6)

Um Backend-Agenten auch mit UI-Klassen in R5 benutzen zu können, kann man mit der Lotus Script `Execute` - Anweisung arbeiten.

Der Einsatz von UI-Klassen in Scheduled Agenten ist z.B. dann denkbar, wenn ein Backend-Agent zusätzlich zu der normalen, periodischen, Ausführung explizit von einem User ausgeführt werden soll, wobei dann andere Parameter übergeben werden sollen.

Beispiel:
Ein geplanter periodischer Agent bearbeitet alle Dokumente, die ein Jahr eines Konfigurationsdokument enthalten.
Nun soll aber ein identischer Code auch von einem User aufgerufen werden können, der über einen Dialog ein Jahr auswählt

Hier hat man nun mit dem `Execute` - Statement die Möglichkeit, UI-Klassen in den Backend-Agenten zu integrieren, um den Agenten auch manuell per Aktion aufrufen zu können.

Code-Auszug:

```
If iIsInteractive = True Then
'Display a prompt to select the desired year:
        strExecuteCode = _
        |
        Dim ws As New NotesUIWorkspace
        strYear = ws.Prompt (PROMPT_OKCANCELLIST, "Auswahl Jahr", "Für welches Jahr
            sollen Dokumente erstellt werden ?", vSetupValidYears (0), vSetupValidYears)
        |

        Execute strExecuteCode

        If strYear = "" Then
            Msgbox "Die Dokumenterstellung wurde abgebrochen !", 64, "Abbruch"
            Exit Sub
        End If
End If
```

Der geplante periodische Agent ruft diese Routine mit `iIsInteractive = False` auf. Damit bekommt der Server nicht mit, dass mit der UI-Klasse `NotesUIWorkspace` gearbeitet wird.

4.4 Benutzen eines Agenten mit UI-Klassen auch im Backend ab R6

Ab R6 werden Backend-Agenten mit UI-Klassen geladen und ausgeführt. Sobald ein UI-Objekt im Script gedimmt wird, folgt ein Laufzeitfehler Nr. 217: „Error creating product object“. Hier kann man den Fehler mit einem simplen Errorhandling abfangen.

5 Mit welchen Berechtigungen laufen die Agenten?

5.1 Allgemeines zur Agenten-Berechtigung (Agent permission)

Das Thema wird allgemein unterschätzt. Jedoch können hierdurch viele Probleme vermieden werden.
Beispielsweise hat ein Anwender keine Berechtigung, Dokumente in einer Datenbank zu löschen. Über die Agenten-Berechtigung lässt sich so etwas aber umgehen.

5.2 Ereignisbasierende Agenten

Berechtigungen siehe in Kap. 3.5.2 in der Spalte „Beschreibung“.

5.3 Serverbasierende Agenten

Serverbasierende Agenten (Definition siehe Kap. 3.5.1) laufen mit folgenden Berechtigungen:

- Berechtigungen die der Unterzeichner des Agenten hat (ist normalerweise der User, der den Agenten zuletzt gespeichert hat)
- Oder: Berechtigungen die der User hat, der in der „On behalf of“ – Option des Agenten eingetragen ist (siehe Kap. 3.7)

5.4 Vom Webbrowser ausgelöste Agenten

Vom Webbrowser ausgelöste Agenten sind immer Backend-Agenten, auch wenn die Auslösung durch Benutzerinteraktion erfolgt.

Dabei laufen die Agenten mit folgenden Berechtigungen:

- Berechtigungen die der Unterzeichner des Agenten hat (ist normalerweise der User, der den Agenten zuletzt gespeichert hat)
- Oder: mit den Berechtigungen die der User hat, der in der „On behalf of“ – Option des Agenten eingetragen ist (siehe Kap. 3.7)
- Oder: mit den Berechtigungen die der authentifizierte User im Web hat, wenn die Option „Run as web user“ ausgewählt wurde (siehe Kap. 3.8)

6 Aufruf von Agenten

6.1 Aufruf von Agenten aus Formelsprache

6.1.1 ToolsRunMacro @Command

Über den @Command-Befehl

```
@Command([ToolsRunMacro]; Agent )
```

wird ein Agent aus der Formelsprache heraus aufgerufen.
Der Parameter „Agent“ ist dabei der Name des zu startenden Agenten vom Typ String.

Wurde in den Agent-Eigenschaften „Trigger / When should this agent run?“ der Eintrag „Manually From Agent List” ausgewählt, so muss hier der Agentenname in Klammern übergeben werden, z.B.

```
_Agent1 := "(Mein Testagent 01)"
@Command([ToolsRunMacro]; _Agent1 )
```

Die Bezeichnung „ToolsRunMacro“ wird aus Abwärts-Kompatibilitätsgründen auch in aktuellen Releases verwendet und kann für Neueinsteiger in die Notes-Entwicklung irreführend sein. Dies ist aber tatsächlich der (bis R5 einzigste) Befehl, um Agenten aus der Formelsprache heraus aufzurufen. Hintergrund: In Notes R3 gab es noch ein Menü „Tools“, außerdem wurden damals Agenten noch als „Macro“ bezeichnet.
Ab R6 steht zusätzlich der @Command-Befehl RunAgent zur Verfügung, siehe Kap. 6.1.2.

6.1.2 RunAgent @Command

Über den ab R6 verfügbaren @Command-Befehl

```
@Command([RunAgent]; Agent )
```

wird ein Agent aus der Formelsprache heraus aufgerufen.
Der Parameter „Agent“ ist dabei der Name des zu startenden Agenten vom Typ String.

Abgrenzung zum ToolsRunMacro @Command (siehe Kap. 6.1.2):

- RunAgent: wird sofort ausgeführt
- ToolsRunMacro: wir erst nach allen anderen @Functions ausgeführt.

6.1.3 AgentRun @Command

Über den ab R6 verfügbaren @Command-Befehl

```
@Command([AgentRun])
```

wird der aktuell in der Agenten-Ansicht ausgewählte Agent gestartet. Sinnvoll ist dieser Befehl wohl nur in einem Smarticon.

6.2 Aufruf von Agenten aus Lotus Script

Die `NotesAgent` – Klasse bietet die Methoden `Run` und `RunOnServer`, um Agenten aus Lotus Script heraus aufzurufen.

Allerdings gibt es hier Einschränkungen:

- So aufgerufene Agenten können nicht debuggt werden
- UI-Klassen sind in dem aufgerufenen Agenten nicht erlaubt.
  Es kann auch nicht direkt mit dem aufgerufenen Agenten interagiert werden.
- Ein Agent kann sich nicht selbst (rekursiv) aufrufen.

6.2.1 NotesAgent-Klasse: Run

6.2.1.1 Syntax

status = *notesAgent*.**Run(** [*noteID*] **)**

notesAgent	Ist ein definierter Agent.
noteID:	Optionaler Parameter vom Typ String. Dies ist die NoteID eines Dokumentes. Man bekommt diese über die `NoteID` - Eigenschaft der `NotesDocument`- Klasse. Damit kann ein Dokument dem aufgerufenen Agenten übergeben werden. Im Zielagenten kann das übergebene Dokument mit der `ParameterDocID` - Eigenschaft der `NotesAgent` - Klasse definiert werden.
status:	Rückgabewert vom Typ Integer. `0` bedeutet, dass der Agent gelaufen ist.

Mit dem Parameter noteID ist es möglich, eine **Vielzahl von Informationen einem Agenten** zu **übergeben**.

Hierbei kann man beispielsweise auch temporär ein Dokument erzeugen als Informationsträger, welches man dann im aufgerufenen Agenten nach Abrufen der Informationen wieder löscht.

6.2.1.2 Beispiele

Beispiel 1: Einfacher Aufruf eines Agenten:

```
Sub Initialize
	Dim session As New NotesSession
	Dim db As NotesDatabase
	Dim agent As NotesAgent
	Set db = session.CurrentDatabase
	Set agent = db.GetAgent("MeinAgent")
	Call agent.Run
End Sub
```

Beispiel 2: Aufruf eines Agenten mit Prüfung:

```
Sub Initialize
	Dim session As New NotesSession
	Dim db As NotesDatabase
	Dim agent As NotesAgent
	Set db = session.CurrentDatabase
	Set agent = db.GetAgent("MeinAgent")

	If Not (agent Is Nothing) Then
		If Not agent.Run = 0 Then
			Messagebox "Agent did not run",48, "Failure"
		End If
	Else
		Messagebox "Agent not found.",48,"Failure"
	End If
End Sub
```

Beispiel 3a: Aufruf eines Agenten mit Prüfung und Übergabe der NoteID:

```
Sub Initialize
	Dim session As New NotesSession
	Dim db As NotesDatabase
	Dim agent As NotesAgent
	Dim uiws As New NotesUIWorkspace
	Dim uidoc As NotesUIDocument
	Set db = session.CurrentDatabase
	Set uidoc = uiws.CurrentDocument
	Set doc = uidoc.Document
	Set agent = db.GetAgent("MeinAgent")
	If Not (agent Is Nothing) Then
		If Not agent.Run(doc.NoteID) = 0 Then
			Messagebox "Agent did not run",48, "Failure"
		End If
	Else
		Messagebox "Agent not found.",48,"Failure"
```

```
        End If
End Sub
```

Beispiel 3b: Entgegennahme des Dokumentes via ParameterDocID, welches vom Code, der diesen Agenten aufruft, übergeben wurde:

```
Sub Initialize
        Dim session As New NotesSession
        Dim agent As NotesAgent
        Dim db As NotesDatabase
        Dim docPassed As NotesDocument
        Set db = session.CurrentDatabase
        Set agent = session.CurrentAgent
        Set docPassed = db.GetDocumentByID(agent.ParameterDocID)
        'Nun steht mit docPassed das Dokument zur Verfügung,
        'welches durch den Code, der diesen Agenten aufgerufen
        'hat, übergeben wurde.

        '......weiterer Code......
End Sub
```

6.2.2 NotesAgent-Klasse: RunOnServer

6.2.2.1 Syntax

status = *notesAgent*.**RunOnServer(** [*noteID*] **)**

Bedeutung der Syntax: Siehe Kap. 6.2.1 (NotesAgent-Klasse: Run)

Agenten, die vom Client aus aufgerufen werden (z.B. über `notesAgent.Run`), werden auf dem Client und nicht auf dem Server ausgeführt.
Mit `notesAgent.RunOnServer` wird ein Agent **explizit auf dem Server** ausgeführt. Dadurch läuft der Agent auch mit der Berechtigung des Unterzeichner des Agenten.

Wird RunOnServer auf einer lokalen Datenbank ausgeführt, wird der Agent wie ‚Run' ausgeführt.

7 Logging und Debugging von Agenten

7.1 Allgemeines zu Logging und Debugging von Agenten

In der aktuellen Version dieser Dokumentation wird auf dieses sehr umfangreiche Thema nur am Rande eingegangen.

Für Details können u.a. folgenden Ressourcen zu Rate gezogen werden:

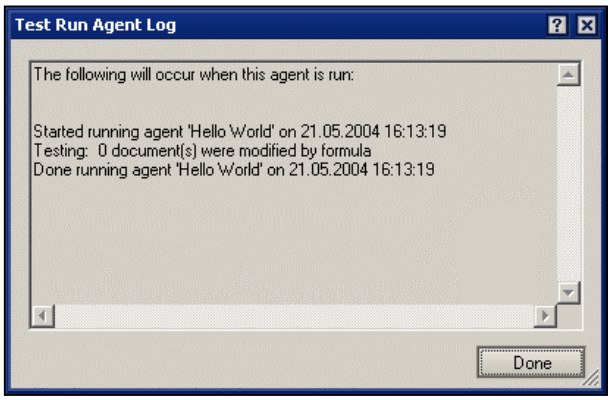
- LDD Today – Artikel „Troubleshooting agents in Notes/Domino 5 and 6" von Julie Kadashevich:
  http://www-10.lotus.com/ldd/today.nsf/62f62847467a8f78052568a80055b380/ef1565a0b202808285256c94004fd0fb?OpenDocument
- Designerhilfe
- Administratorhilfe

7.2 Testen von Agenten im Designer

Agenten können direkt im Designer getestet werden.

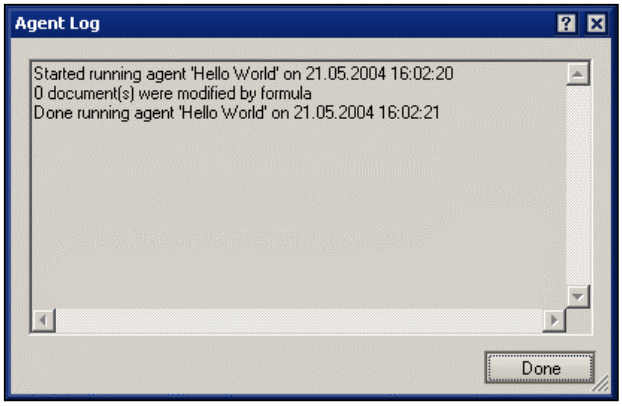
Dazu wählt man in der Agenten-Ansicht des Designers den Agenten aus und startet diesen über Menü Agent | Test (R5: Menü: Actions | Test).

Als Ergebnis erhält man folgendes Ausgabefenster:



7.3 Ansehen des Agent Log im Designer

Um ein Logging eines Agenten zu sehen, wählt man den Agenten in der Agentenansicht aus und rüft über Menü Agent | Log folgendes Fenster auf:



7.4 Logging von Agenten

In Lotus Notes gibt es Möglichkeiten, Agenten mitzuloggen und die Ergebnisse in eine Datenbank, in eine Textdatei oder in eine E-Mail zu schreiben.

Siehe hierzu die `NotesLog` – Klasse in der Designerhilfe.

7.5 Remote Debugger

Für clientbasierendes Debugging kann der Debugger (Menü: File | Tools | Debug LotusScript) verwendet werden.

Ab R6 gibt es den Remote Debugger, hiermit wird es ermöglicht, serverbasierendes Debugging durchzuführen.

Der Remote Debugger wird über Menü: File | Tools | Remote Debugger aufgerufen.

Hierzu müssen jedoch Voreinstellungen getroffen werden.

Details siehe ⇨ Admininstrator-Hilfe.

8 Einschränkungen und Besonderheiten

8.1 Geplante periodische Agenten (scheduled agents)

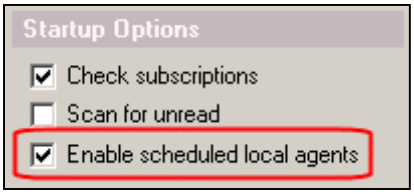
8.1.1 @DBColumn und @DBLookup

Ein geplanter periodischer Agent, der die Formeln @DbColumn oder @DbLookup verwendet, kann nur auf Datenbanken zugreifen, die auf derselben Workstation oder demselben Server gespeichert sind.

8.1.2 Lokale Agenten

Geplante, periodische Agenten können auch so konfiguriert werden, das diese nicht auf dem Server, sondern lokal, laufen.

Damit dies funktioniert, muss in den Client-Einstellungen (Menü „File | Preferences | User Preferences...) folgende Option aktiviert werden:



9 Praxisbeispiele

9.1 Ausgehend vom Mailfile einen Agenten einer anderen DB aufrufen

9.1.1 Ausgangssituation

Es sollen E-Mails per Knopfdruck als „RendertoRTItem“, also der Komplett-Inhalt der E-Mail, in ein Richtextfeld einer anderen Datenbank übernommen werden.

9.1.2 Problemstellung

Es müssen hierzu Eingriffe ins Design des Mailfiles vorgenommen werden (Erstellung eines Agenten), um beispielsweise einen Agenten zur Verfügung zu stellen der diese Aufgabe übernimmt.
Wenn allerdings Änderungen bei diesem Agenten vorgenommen werden (z.B. zusätzliche Informationen übernehmen, Änderungen der Feldbezeichnungen der Zieldatenbank, etc.), so müsste der Agenten-Quellcode aller betroffenen Anwender im Mailfile aktualisiert werden. Eine Möglichkeit der Agenten-Verteilung in beliebige Datenbanken ist uns auch nach www-Recherche nicht bekannt.

9.1.3 Lösungsansatz

Man hinterlegt in dem Mailfile-Agenten ausschließlich folgende Informationen:

- Zielserver und Zieldatenbank
- Aufzurufende Routine der Zieldatenbank

Den Rest erledigt dann die aufgerufene Routine. Man übergibt lediglich noch das zu behandelnde Maildokument.
Da ein Aufruf einer Sub/Function einer Scriptlibrary einer anderen Datenbank nicht möglich ist – aber Notes die Möglichkeit bietet, Agenten auch anderer Datenbanken aufzurufen – liegt es auf der Hand, dass man hier den Agenten mit dem eigentlichen Code in der Zieldatenbank platziert. Damit kann man Änderungen am Agenten zentral in der Zieldatenbank vornehmen.

9.1.4 Die Umsetzung

Mailfile

Zunächst erstellt man in der Mail-Datenbanken einen Agenten mit folgenden Eigenschaften:

- Trigger: Agent menu selection (R5: Manually From Actions Menu)
- Target: Selected documents

Als nächstes kopiert man folgendes Code-Beispiel in den Agenten:

```
Sub Initialize

	On Error Goto ERRORHANDLER

	Dim session As New NotesSession
	Dim agent As NotesAgent
	Dim strServerpath As String
	Dim strDatabase As String

	'----------> Set current doc (you can run this code either from a view or direct from a doc)
	Dim db As NotesDatabase
	Dim doc As NotesDocument
	Dim uiws As New NotesUIWorkspace
	Set db = session.CurrentDatabase
	If Not uiws.CurrentDocument Is Nothing Then
		Set doc = uiws.CurrentDocument.Document
	Elseif Not session.DocumentContext Is Nothing Then
		Set doc = session.DocumentContext
	End If
	'<----------

	'-------> Path of server & database
	strServerpath = "" 'Leave empty if you work local
	strDatabase = "F.\Notes\R5\MyDatabases\docu.nsf" 'incl. path!
	'<------

	'--------> Run Agent
	Dim dbNew As New NotesDatabase(strServerpath , strDatabase)
	Set agent = dbNew.GetAgent("(CopyMailInDocuDB)")
	If Not(agent Is Nothing) Then
		If Not agent.Run(doc.NoteID) = 0 Then 'runs the agent
			Messagebox "Agent did not run",48, "Failure"
		End If
	Else
		Messagebox "No existing agent found.",48,"Error"
	End If
	'<--------


EXIT_SCRIPT:
	Exit Sub

ERRORHANDLER:
	Msgbox "Error: " & Err & " - " & Error$ & Chr(10) & Chr(10)_
	& "Procedure: Mail-DB Agent 'Copy in docu DB '"& Chr(10) & Chr(10)_
	& "Line: " & Erl & Chr(10), _
	48, "An error occured"
	Resume EXIT_SCRIPT

End Sub
```

Was machen wir in diesem Agenten:

- Wir definieren als doc das Ausgangs-Maildokument. Hierbei kann der Agent sowohl aus einer Ansicht als auch aus dem geöffneten Maildokument heraus aufgerufen werden
- Wir setzen den Server und Pfad der Zieldatenbank und definieren die Zieldatenbank
- Wir starten den angegebenen Agenten der Zieldatenbank (inkl. Prüfung ob Agent überhaupt vorhanden und ob Agent auch ausgeführt wurde) – dabei übergeben wir dem Agenten das zuvor definierte doc via „agent.Run(doc.NoteID)“

Zieldatenbank

In der Zieldatenbank erstellt man ebenso einen Agenten mit folgenden Eigenschaften:

- Trigger: Agent list selection (R5: Manually From Agent List)
- Target: None (R5: Run once (@Commands may be used))

Als nächstes kopiert man folgendes Code-Beispiel in den Agenten:

```
Sub Initialize
	Dim session As New NotesSession
	Dim agent As NotesAgent
	Dim dbDocDB As NotesDatabase
	Dim docDocDB As NotesDocument
	Dim dbMail As NotesDatabase
	Dim docMail As NotesDocument
	Dim strForm As String
	Dim rtitemNew As NotesRichTextItem

	Set agent = session.CurrentAgent
	Set dbDocDB = session.CurrentDatabase
	strForm = "frm_DocDB" 'name of used form

	'-----> Set mail-database & get mail document (provided by previous agent)
	Set dbMail = New NotesDatabase("","")
	Call dbMail.OpenMail
	If Not dbMail.IsOpen Then Exit Sub
	Set docMail = dbMail.GetDocumentByID(agent.ParameterDocID)
	'<--------

	'-------->  New document
	Set docDocDB = dbDocDB.CreateDocument
	docDocDB.FORM = strForm
	docDocDB.Subject = docMail.Subject(0)
	Set rtitemNew = New NotesRichTextItem( docDocDB, "Body" )
	Call docMail.RenderToRTItem( rtitemNew )
	Call docDocDB.Save( False , False )
	'<----------
End Sub
```

Was machen wir in diesem Agenten:

- Wir definieren die Maildatenbank (also Quelle) und das vom vorigen Agenten übergebene Dokument via „`Set docMail = dbMail.GetDocumentByID(agent.ParameterDocID)`“
- Wir erzeugen ein neues Dokument in der Zieldatenbank und setzen Felder
- Wir verwenden „RenderToRTItem“, um das Ausgangs-Maildokument in das Ziel-Richtextfeld zu kopieren.

9.1.5 Fazit

Dieses Beispiel zeigt, dass es einfach möglich ist, Agenten anderer Datenbanken aufzurufen und dabei ein Dokument zu übergeben.

9.2 Dokumente refreshen

Um beispielsweise in ausgewählten Dokumenten berechnete Feldwerte neu berechnen zu lassen, kann man sich mit folgendem Formel-Agenten helfen:

```
@Command([ToolsRefreshSelectedDocs])
```

Target: None (R5: Run once (@Commands may be used)

Anmerkungen:

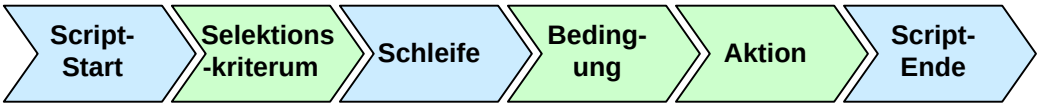
- Die Ausführung dieses Agenten kann bei vielen Dokumenten mit umfangreichen Feldern sehr viel Zeit in Anspruch nehmen.
- Ein Lotus Script – Pendant ist uns hierzu nicht bekannt, bsp. die `ComputeWithForm` Methode der `NotesDocument` - Klasse berechnet die in Dokumenten berechneten Werte <u>nicht</u> neu.

9.3 Agenten-Bausatz

9.3.1 Über dieses Kapitel

Hier werden diverse Codebeispiele gezeigt, um Agenten zur Datenbank-bereinigung, Feldwertänderung etc. einzusetzen. Ich verwende hier ausschließlich Backend-Klassen, somit können diese Agenten auch serverbasierend ausgeführt werden.

Die Agenten können beliebig zusammengesetzt werden, es muss aber exakt nach folgender Reihenfolge vorgegangen werden:



Hierbei muss man bei den grünen Bestandteilen eine Auswahl treffen.

Im jeweiligen Teil sieht man, wo man den nächsten Teil im Script einsetzen muss.

Selektionskritierien:

- Alle Dokumente in der Datenbank
- Alle Dokumente einer Ansicht
- Ausgewählte Dokumente

Bedingungen:

- Bestimmter Textfeldwert
- Bestimmtes Datumsfeld vs. Heutiges Datum – x Tage

Aktionen:

- Items löschen
- Feldinhalt ändern
- Dokumente löschen
- Dokumente in andere DB verschieben

9.3.2 Script-Start

```
Sub Initialize
        On Error Goto ERRORHANDLER
        Dim session As New NotesSession
        Dim db As NotesDatabase
        Dim dbTarget As NotesDatabase
        Dim collection As NotesDocumentCollection
        Dim doc As NotesDocument
        Dim docNext As NotesDocument 'for deleting docs
        Dim item As NotesItem
        Dim vItems As Variant
        Dim strItemName As String
        Dim strItemValue As String
        Dim strItemNameCh As String
        Dim strItemValueCh As String
        Dim strItemEntry As String
        Set db = session.CurrentDatabase
        '#######################################
        'HIER DAS SELEKTIONSKRITERUM EINFÜGEN
        '#######################################
```

9.3.3 Selektionskriterien

Alle Dokumente einer Datenbank:

- Agent-Target: All documents in the database

Code:

```
        Set collection = db.AllDocuments
        '#######################################################
        'HIER DIE SCHLEIFE FÜR DIE DOK-ABARBEITUNG EINFÜGEN
        '#######################################################
```

Alle ausgewählten Dokumente:

- Agent-Target: Selected documents

Code:

```
        Set collection = db.UnprocessedDocuments
        '#######################################################
        'HIER DIE SCHLEIFE FÜR DIE DOK-ABARBEITUNG EINFÜGEN
        '#######################################################
```

Alle Dokumente einer Ansicht:

- Agent-Target: All documents in view

Code:

```
'Hier setzen wir als Workaround die View-Collection in die Doc-Collection,
'damit der Bausatz - Copy&Paste funktioniert ohne Anpassungen.
Dim view As NotesView
Dim vc As NotesViewEntryCollection
Dim entry As NotesViewEntry
Set view = db.GetView("Meine View") 'Your view
Set vc = view.AllEntries
Set entry = vc.GetFirstEntry()
'We need an empty doccollection
Set collection = db.GetDocumentByUnid(db.Views(0).UniversalID).Responses
While Not (entry Is Nothing)
        Set doc = entry.Document
        Call collection.AddDocument(doc)
        Set entry = vc.GetNextEntry(entry)
Wend
'#######################################################
'HIER DIE SCHLEIFE FÜR DIE DOK-ABARBEITUNG EINFÜGEN
'#######################################################
```

9.3.4 Schleife

```
Set doc = collection.GetFirstDocument
While Not (doc Is Nothing)
        Set docNext = collection.GetNextDocument(doc)
        '##############################
        'HIER DIE BEDINGUNG EINFÜGEN
        '##############################
        Set doc = docNext
    Wend
```

9.3.5 Bedingungen

Bestimmter Textfeld-Wert oder Maske:

```
strItemName = "MyItem1" 'Item-name or Form
strItemValue = "Status 3" 'Item-content or form-name

strItemEntry = doc.GetItemValue(strItemName)
If strItemEntry (0) = strItemValue Then
        '#########################
        'HIER DIE AKTION EINFÜGEN
        '#########################
    End If
```

Nach Datumsfeld:

```
If doc.Datum(0) < (Now - 30) Then  'All docs older than 30 days acc. to your date field
        '##########################
        'HIER DIE AKTION EINFÜGEN
        '##########################
    End If
```

9.3.6 Aktionen

Items löschen:

```
'-----> Items you want to remove
Redim vDeleteItems(20) 'change this value if you process more than 20 items…
vDeleteItems(1) = "Feld1"
vDeleteItems(2) = "Feld2"
vDeleteItems(3) = "Itemname3"
vDeleteItems(4) = "Itemname4"
'........and so on.........
'<-----
Forall i In vDeleteItems
        While (doc.HasItem(i))
                Set item = doc.GetFirstItem(i)
                Call item.Remove
                Call doc.Save(False, False)
        Wend
    End Forall
```

Feldinhalt ändern:

```
strItemNameCh = "MyItem" 'item you want to change
strItemValueCh = "My new Value for the item" 'your new item value
Set item = doc.ReplaceItemValue ( strItemNameCh, strItemValueCh )
Call doc.Save(False, False)
```

Dokumente löschen:

```
Call doc.Remove( True )
```

Dokumente in andere Datenbank verschieben:

```
Set dbTarget = New NotesDatabase("" , "myarchive.nsf") 'server and database
Call doc.CopyToDatabase(dbTarget)
Call doc.Remove(True)
```

9.3.7 Script-Ende

```
EXIT_SCRIPT:
        Exit Sub

ERRORHANDLER:
        Msgbox "Error: " & Err & " - " & Error$ & Chr(10) & Chr(10)_
        & "Line: " & Erl & Chr(10), _
        48, "An error occured"
        Resume EXIT_SCRIPT

End Sub
```

9.4 Weitere Einsatzmöglichkeiten von Agenten

9.4.1 Shared Actions – Gemeinsame Aktionen

Es gibt vermehrt Probleme beim Einsatz von Shared Actions – wenn diese in Lotus Script geschrieben sind. Das Script wird nicht immer wie vorgesehen ausgeführt. Die Probleme treten noch häufiger auf, wenn im Code auf Script Libraries zurückgegriffen wird.

Als Abhilfe lagert man das Script in einen Agenten aus (Trigger: None / Run once (@Commands may be used)).

In die Shared Action schreibt man den Formelsprachebefehl `@Command([ToolsRunMacro]; Agent )` und ruft damit den erzeugten Agenten auf.

Mit dieser Vorgehensweise lassen sich Shared Actions zuverlässig verwenden.
Soll die Shared Action bzw. der Agent dabei sowohl für das aktuell geöffnete Dokument als auch in einer Ansicht für das erste der ausgewählten Dokumente verwendet werden, ist folgender Codeauszug hilfreich (dadurch wird als `doc` das jeweilige Dokument gesetzt):

```
Dim db As NotesDatabase
Dim doc As NotesDocument
Dim uiws As New NotesUIWorkspace
Set db = session.CurrentDatabase

If Not uiws.CurrentDocument Is Nothing Then
        Set doc = uiws.CurrentDocument.Document
Elseif Not session.DocumentContext Is Nothing Then
        Set doc = session.DocumentContext
End If
```

9.5 Periodische Agenten im verborgenen Design

Ist das Design einer Notes-Datenbank verborgen, hat man keine Möglichkeit mehr, die periodischen Einstellungen eines Agenten zu ändern.

Man kann jedoch das Auslösen des Agenten in eine zweite Datenbank auslagern.

Beim eigentlichen Agenten setzt man den Trigger auf „Agent menu selection“ (R5: Manually From Actions Menu).

In einer zweiten Datenbank mit offenem Design erstellt man einen gelanten periodischen Agenten, der den Agenten der 1. Datenbank aufruft.

10 Links / Weiterführende Informationen

Titel / Beschreibung	Link	Sprache
Das Notes Forum	http://www.atnotes.de	DE
LDD Forum Agent-FAQ	http://www-10.lotus.com/ldd/46dom.nsf/7e6c7e584a76331b85256a46006f083e/574c99ccb345839185256976004e811e?OpenDocument	EN
LDD Today: Troubleshooting agents in Notes/Domino 5 and 6 (von Julie Kadashevich)	http://www-10.lotus.com/ldd/today.nsf/62f62847467a8f78052568a80055b380/ef1565a0b202808285256c94004fd0fb?OpenDocument	EN
LDD Today: Decoding the new Notes/Domino 6 agent features (von Julie Kadashevich)	http://www-10.lotus.com/ldd/today.nsf/9148b29c86ffdcd385256658007aaa0f/177bbe55c6848ae000256c44003aee17?OpenDocument	EN

11 Autor

Ersteller und Initiator dieses Dokumentes ist Matthias TMC (atnotes@gmx.de).

12 Index

@

@Command AgentRun 18
@Command RunAgent 17
@Command ToolsRunMacro 17
@DBColumn 23
@DBLookup 23

A

Agent Log 22
Agent properties box 5
Agentenbausatz 28
Agenten-Einstellungen 5
AgentRun @Command 18
Allow user activation 12
Aufruf aus Formelsprache 17
Aufruf aus Lotus Script 18
Aufruf von Agenten 17
Autoren dieses Dokumentes 34

B

Backend 14
Bausatz 28
Beispiele 24
Berechtigungen 16
Besonderheiten 23

C

Comment 5

D

Debugging 21

E

Einsatzmöglichkeiten 24
Einschränkungen 23
Einstellungsbereich 5
Enable scheduled local agents 23
Ereignis-Auslösung 7
Ersteller dieses Dokumentes 34
Event Trigger 7

F

Formelsprache 13
Frontend 14

G

Gemeinsame Aktionen 33
Geplante Auslösung 6

L

Links 34
Logging 21
Lotus Script 13

N

Name 5
NotesLog 22

P

Periodische Auslösung 6
Praxisbeispiele 24
Private 6
Programmiersprachen 13

R

Refresh 27
Remote Debugger 22
Run 18
Run as web user 11
Run in background client thread ... 12
Run on behalf of 11
RunAgent @Command 17
RunOnServer 20

S

Scheduled Trigger 6
Shared 6
Shared Actions 33

T

Target 9
Testen von Agenten 21
ToolsRunMacro @Command 17
Trigger 6

Ü

Über dieses Dokument 4

W

Was sind Agenten 4
Weiterführende Infos..... 34
When should this agent run? 6
Which document(s) should it act on? 9