BISSANTZ

SSIS-Logging in DeltaMaster

Autor	Bernd Werther
Datum der Veröffentlichung	24.02.2017

SSIS-Logging in DeltaMaster

Gerade in Projekten mit vielen Datei-Schnittstellen kommt das Thema „Fehler in der täglichen Aufbereitung“ in schöner Regelmäßigkeit wieder auf. Vom Endanwender kommt in solchen Fällen häufig nur die Information, dass Daten in DeltaMaster fehlen.

Für den Techniker bedeutet das dann meist, sich im SQL Server-Agent auf Spurensuche zu begeben und sich über viele, sehr aufgeblasen wirkende Logs, zur eigentlichen Fehlerursache „durchzuhangeln“. Verständlich, dass viele Endanwender sich davor scheuen, die Fehler selbst anzusehen.

Daher soll in diesem Beitrag ein Ansatz gezeigt werden, diese Komplexität zu reduzieren indem Fehler, die in SSIS-Paketen auftreten, in einer DeltaMaster-Anwendung dargestellt werden. Durch die gesunkene Komplexität soll der Anwender im Idealfall selbst dazu in der Lage sein, den Fehler zu analysieren und sogar zu beheben.

Dafür sind drei Schritte notwendig: Zunächst muss das SSIS-Logging außerhalb des SQL-Server-Agents verfügbar gemacht werden. Anschließend müssen die Logging-Resultate relational aufbereitet werden um sie dann im dritten Schritt in einer relationalen DeltaMaster-Anwendung anwenderfreundlich darzustellen.

1 Logging im SSIS-Paket aktivieren

Zunächst muss das Logging für das betroffene dtsx-Paket aktiviert werden. Dafür macht man unter „Control Flow“ einen Rechtsklick im freien (grauen) Bereich und wählt im sich öffnenden Menü den Punkt „Logging“ aus.

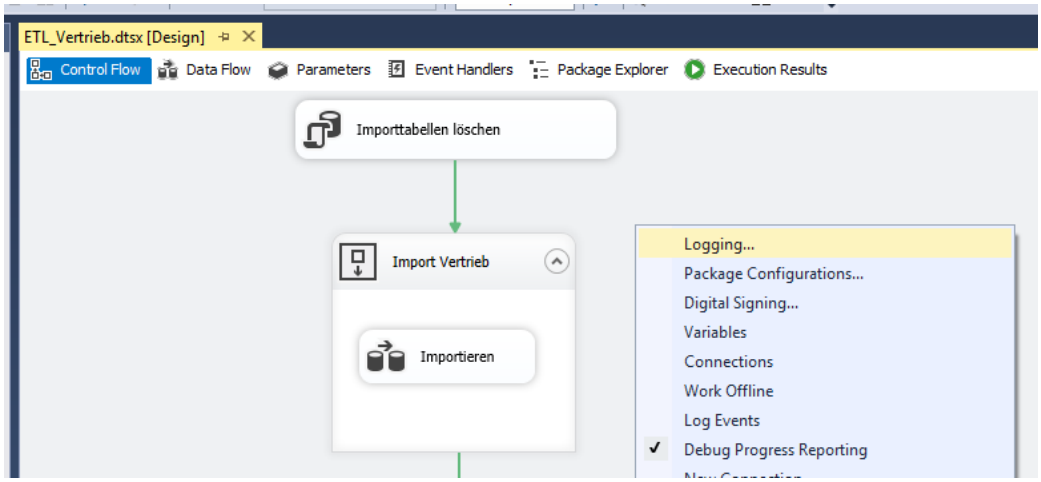


Abbildung 1 Logging im dtsx-Paket aktivieren

Hier müssen ein Provider und die zu loggenden Komponenten festgelegt werden. Im einfachsten Fall wird das Logging für das gesamte Paket aktiviert (Haken bei „ETL_Vertrieb“ im Beispiel). Die anderen Komponenten sind standardmäßig erst einmal nur implizit angehakt. Diese können auch separat aktiviert werden. Dadurch kann die Detaillierung des Loggings individuell für diese Komponenten gesteuert werden.

Als Provider empfiehlt es sich, „SSIS-Protokollanbieter für SQL Server“ zu verwenden, da so die Logs in einer Datenbanktabelle abgelegt werden. Anschließend auf „Add…“ klicken und den neuen Eintrag in der Liste aktivieren, sinnvoll umbenennen und in der Spalte „Configuration“ einen passenden „Connection Manager“ aus dem Drop-Down-Menü auswählen. Für die meisten DeltaMaster-Projekte sollte es sich um einen „OLE DB Connection Manager“ handeln.

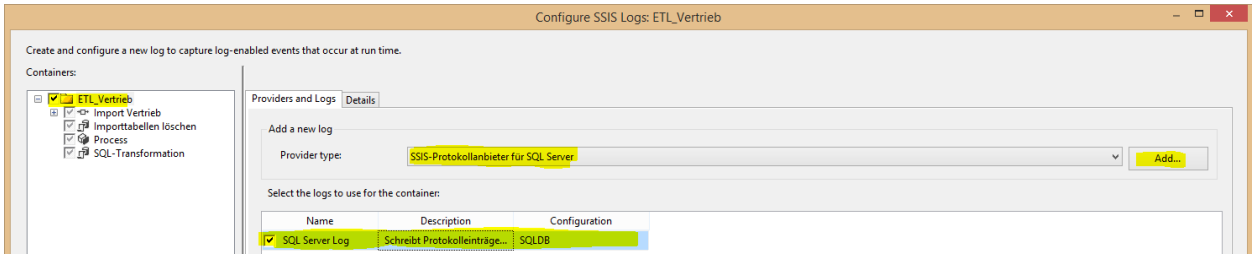


Abbildung 2 Festlegen der zu loggenden Komponenten und des Providers

Auf der Registerkarte „Details“ kann nun der Umfang des Loggings definiert werden. Das ist für alle Komponenten möglich, die in der Spalte „Containers“ nicht ausgegraut sind. Geloggt werden kann eine Vielzahl von Events. Von Informationen über Statusänderungen und Warnungen bis hin zu Fehlermeldungen. Es sollten hier nicht zu viele Events gewählt werden, um das Logging nicht zu überfrachten. Immer aktiviert werden sollte das Event „OnError“, um Fehler zu dokumentieren.

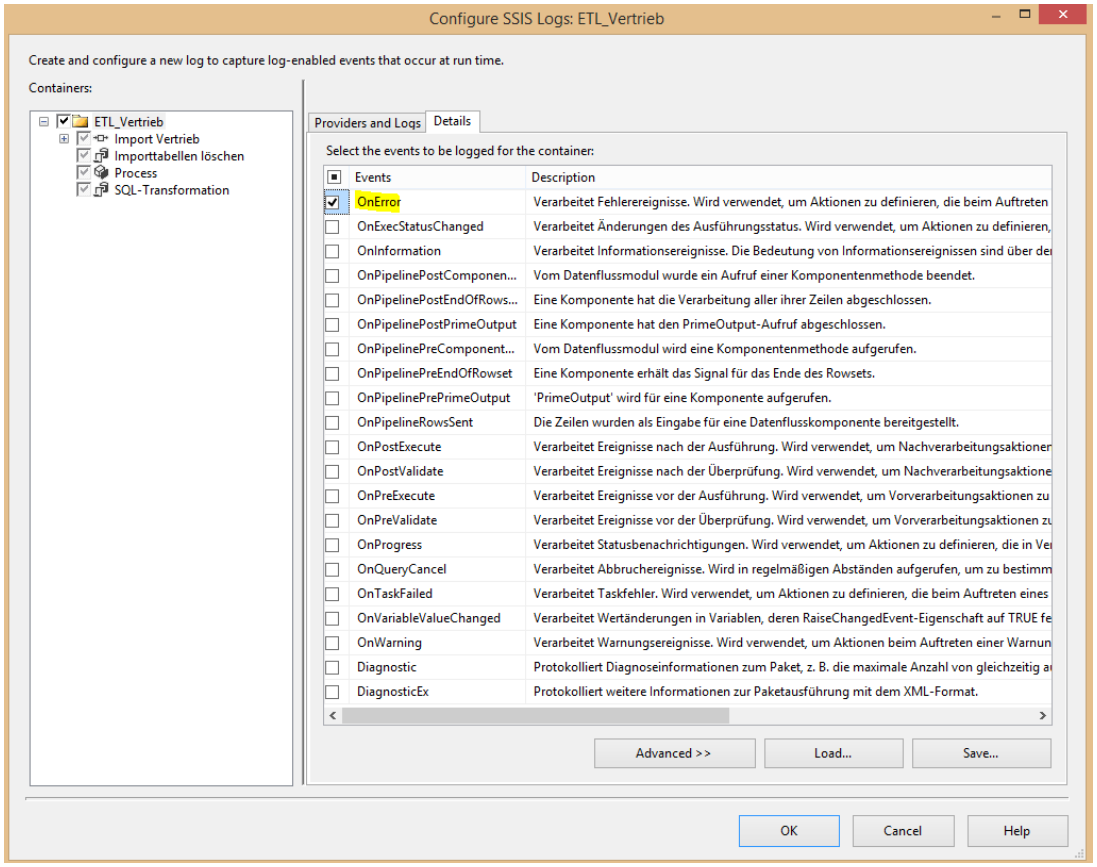


Abbildung 3 Detaillierungsgrad einstellen

Unter „Advanced>>“ kann zusätzlich eingestellt werden, welche Informationen zum jeweiligen Event gespeichert werden sollen (z. B. ausführender Nutzer, Meldungstext etc.). Standardmäßig sind hier alle Optionen aktiviert. Diese Einstellungen werden in diesem Beispiel unverändert gelassen.

Sind alle Einstellungen gesetzt, kann mit „OK“ bestätigt und das dtsx-Paket gespeichert werden. Bei der nächsten Ausführung des Pakets wird in der Datenbank des „Connection Managers“ eine Tabelle [dbo].[sysssislog] angelegt, falls sie noch nicht existiert. Standardmäßig ist sie unter „Systemtabellen“ zu finden.

Achtung: Wird die Tabelle gelöscht, so wird sie zwar bei der nächsten Ausführung wieder angelegt, allerdings ist sie dann direkt unter „Tabellen“ zu finden.

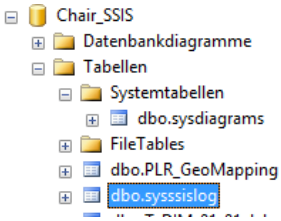
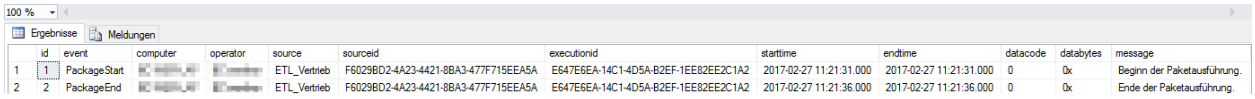


Abbildung 4 Tabelle [dbo].[sysssislog] im Objektbrowser des Management Studios

Der Aufbau der Tabelle sieht wie folgt aus:



id	event	computer	operator	source	sourceid	executionid	starttime	endtime	datacode	databytes	message
1	PackageStart	[redacted]	[redacted]	ETL_Vertrieb	F6029BD2-4A23-4421-8BA3-477F715EEA5A	E647E6EA-14C1-4D5A-B2EF-1EE82EE2C1A2	2017-02-27 11:21:31.000	2017-02-27 11:21:31.000	0	0x	Beginn der Paketausführung.
2	PackageEnd	[redacted]	[redacted]	ETL_Vertrieb	F6029BD2-4A23-4421-8BA3-477F715EEA5A	E647E6EA-14C1-4D5A-B2EF-1EE82EE2C1A2	2017-02-27 11:21:36.000	2017-02-27 11:21:36.000	0	0x	Ende der Paketausführung.

Abbildung 5 Inhalt der Tabelle [dbo].[sysssislog] nach einer erfolgreichen Ausführung

Interessant: Auch wenn, wie im Beispiel, lediglich Fehler geloggt werden sollen, werden auch bei fehlerfreier Paketausführung zwei Datensätze mit den Events „PackageStart" und „PackageEnd" generiert.

ACHTUNG: Handelt es sich um eine DeltaMaster ETL-Datenbank, dann wird die sysssislog-Tabelle während des Transforms „aufgeräumt". Es greift der Parameter „Transformation log tables lifetime (in days)" aus dem Bericht „Parameter". Steht der Parameter auf 0, so wird das „Event" „PackageStart" nie zu sehen sein, weil die Tabelle zu Beginn der Prozedur „P_Transform_all" geleert wird.

Transformation log tables lifetime (in days)	10

Abbildung 6 Einstellung der Logging-Tabellen Lebensdauer in DeltaMaster ETL

Da die von SSIS generierten Meldungen immer noch sehr technisch wirken, wird das SSIS-Paket noch um eine gröbere Logging-Stufe erweitert, die dem Anwender mitteilen soll, in welchem Bereich des SSIS-Pakets ein Fehler aufgetreten ist.

Hierfür wird zunächst in der Projektdatenbank eine Tabelle T_D_JobLog angelegt.

```
CREATE TABLE [dbo].[T_D_JobLog](
	[executionid] [UNIQUEIDENTIFIER] NOT NULL,
	[Time] [DATETIME] NOT NULL,
	[JobStepID] [INT] NULL,
	[Status] [INT] NULL
) ON [PRIMARY]
```

In dieser Tabelle wird der Fortschritt des Pakets protokolliert. Das Feld Status gibt an, ob der Schritt erfolgreich war, oder nicht (1 = erfolgreich, -1 = fehlerhaft). In einer weiteren Tabelle (T_S_JobSteps) werden die Schritte mit ID und Bezeichnung hinterlegt:

```
CREATE TABLE [dbo].[T_S_JobSteps](
	[JobstepID] [int] NOT NULL,
	[JobStepName] [varchar](50) NULL
) ON [PRIMARY]
```

Hier können je nach SSIS-Paket verschiedene Schritte eingetragen werden. Für ein SSIS-Paket, das aus Import, Transformation und Cube-Verarbeitung besteht, empfehlen sich beispielsweise die folgenden sechs Schritte:

JobstepID	JobStepName
1	Import gestartet
2	Import erfolgreich beendet
3	Transform gestartet
4	Transform erfolgreich beendet
5	Process gestartet
6	Process erfolgreich beendet

Um entsprechende Logeinträge zu generieren, werden einfach entsprechende „Execute SQL Task"-Komponenten in den Kontrollfluss des Pakets integriert, die per INSERT-Statement in die Tabelle T_D_JobLog schreiben.

Für Schritt 1 lautet das Statement beispielsweise:

```
INSERT INTO T_D_JobLog
	SELECT top 1
		s.executionid
		,getdate() AS Time
		,1 AS JobStepID
		,1 AS Status
	FROM sysssislog s
	ORDER BY s.id DESC
```

Als „executionid" wird die letzte bekannte „executionid" aus der Tabelle sysssislog gewählt. Das entspricht der „executionid" der aktuellen Paketausführung und hilft die beiden Logging-Varianten miteinander in Beziehung zu setzen.

Bei erfolgreicher Schrittausführung wird hier einfach eine 1 als Status protokolliert. Für den Fehlerfall wird ein weiterer „Execute SQL-Task" angelegt, dessen Verbindungspfeil auf „Fehler" eingestellt wird. Das SQL-Statement wird für diesen Fall dahingehend geändert, dass der Status mit „-1" geschrieben wird. Diese Fehlerfallbehandlung muss nur nach den Hauptschritten „Import", „Transform" und „Process" (für die Beispielhaft genannten Schritte sind dies die Schritte 2, 4 und 6) erfolgen und nicht beim Starten des jeweiligen Schritts.

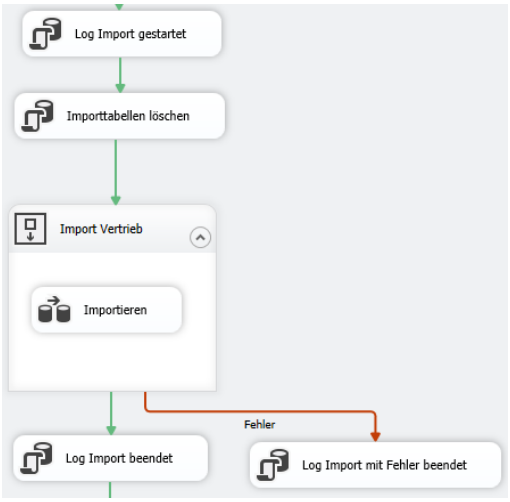


Abbildung 7: Beispiel für die drei "Execute SQL Task"-Komponenten zum Loggen des Import-Prozesses

Damit auch diese Tabelle regelmäßig geleert wird, empfiehlt es sich, in DeltaMaster ETL Modellen mit Hilfe des „Transformation log tables lifetime (in days)"-Parameters einen weiteren „Execute SQL-Task" zu Beginn des SSIS-Pakets aufzunehmen. Der SQL-Code des Schritts lautet wie folgt:

```
DECLARE @SYSLOG_Lifetime AS int;

SELECT @SYSLOG_Lifetime = convert(int,ParameterValue) * -1 FROM T_Model_Parameter
WHERE ParameterID = 800;

DELETE FROM T_D_JobLog

WHERE [Time] < dateadd(day,@SYSLOG_Lifetime,getdate())
```

2 Relationale Aufbereitung der Logging-Daten

Um dem Anwender eine relationale DeltaMaster Anwendung bereit zu stellen werden jetzt noch einige Dimensions- und Fact-Views angelegt.

Es werden die folgenden Dimensionen erstellt:

V_LOGGING_DIM_Event

```
CREATE VIEW [dbo].[V_LOGGING_DIM_Event] AS
	SELECT DISTINCT
		lo.[event]
	FROM [dbo].[sysssislog] lo
```

➔ Dimension für die Unterschiedlichen Event-Typen, z. B. „PackageStart", „OnError" etc.

V_LOGGING_DIM_Execution:

```
CREATE VIEW [dbo].[V_LOGGING_DIM_Execution] AS
	SELECT
		lo.executionid
		,min(lo.starttime) Starttime
		,max(lo.endtime) Endtime
		,YEAR(min(lo.starttime)) YearID
		,YEAR(min(lo.starttime)) * 100 + MONTH(min(lo.starttime)) MonthID
		,DATEADD(dd, DATEDIFF(dd, 0, min(lo.starttime)),0) DayID
		,dbo.F_BC_DateCODE('M', min(lo.starttime), 0) MonthBEZ
		,dbo.F_BC_DateCODE('D',min(lo.starttime), 0) DayBEZ
	FROM [dbo].[sysssislog] lo
	GROUP BY lo.executionid
```

➔ Dimension, die die unterschiedlichen Paket-Ausführungen unterscheidbar macht. Es erfolgt auch eine Aufgliederung nach der Zeit um später Ebenen bilden zu können.

V_LOGGING_DIM_JobSteps:

```
CREATE VIEW [dbo].[V_LOGGING_DIM_JobSteps] AS
	SELECT
		s.JobstepID
		,s.JobStepName
	FROM [dbo].[T_S_JobSteps] s
```

➔ Dimension zur Unterscheidung der einzelnen Schritte des SSIS-Pakets.

V_LOGGING_DIM_Source:

```
CREATE VIEW [dbo].[V_LOGGING_DIM_Source] AS
	SELECT DISTINCT
		lo.sourceid
		,lo.source
	FROM [dbo].[sysssislog] lo
```

➔ Dimension listet die Komponenten des SSIS-Pakets, in denen Events geloggt wurden.

V_LOGGING_FACT_Job:

```
CREATE VIEW [dbo].[V_LOGGING_FACT_Job] AS
	SELECT
		e.executionID
		,l.[Time]
		,s.JobStepID
		,CASE
			WHEN l.executionid IS NULL AND s.JobstepID = 1 THEN -5
			WHEN l.executionid IS NULL THEN 0	-- Schritt läuft noch
			WHEN s.JobstepID = 6 THEN 5 * l.status
			WHEN l.status = -1 THEN 5 * l.status
			ELSE l.[Status]
		END AS [Status]
	FROM dbo.[T_S_JobSteps] s
	LEFT JOIN  V_LOGGING_DIM_Execution e ON
		1=1
	LEFT JOIN [dbo].[T_D_JobLog] l ON
		l.JobStepID = s.JobStepID
		AND e.executionid = l.executionid
```

➔ Die View stellt den Status der einzelnen Job-Schritte dar. Dabei dient der Status des Schritts als Kennzahl. Für eine geeignete graphische Darstellung werden hier einige Ausnahmen definiert:

a) Falls es zu Schritt 1 keine Statusmeldung gibt, aber das Paket gestartet wurde, so wird der Status „-5“ zugewiesen.
b) Ist für einen Schritt, der nicht Schritt 1 ist, kein Status bekannt, so wird der Wert 0 zugewiesen. Der Schritt ist in diesem Fall noch ausstehend.
c) Handelt es sich um den letzten Schritt (im Beispiel Schritt 6), dann wird dessen Status mit dem Wert 5 multipliziert. Mögliche Ausprägungen sind also „5“ und „-5“.
d) Ist der Schritt fehlgeschlagen und trägt der Status somit den Wert “-1” dann wird der Wert mit 5 multipliziert.
e) Ansonsten wird der Status unverändert gelassen. Der „Sonst“-Fall gilt also für Schritte die erfolgreich abgeschlossen wurden aber nicht der letzte Schritt des Pakets waren.

➔ Die definierten Ausnahmen helfen in der relationalen DeltaMaster-Anwendung bei einer geeigneten graphischen Darstellung.

V_LOGGING_FACT_logs:

```
CREATE VIEW [dbo].[V_LOGGING_FACT_logs] AS
              SELECT
                     lo.[event]
                     ,lo.sourceid
                     ,lo.executionid
                     ,lo.starttime starttime
                     ,lo.endtime endtime
                     ,REPLACE(REPLACE(lo.[message],'.  ', '.' + CHAR(10)), '. ', '.' +
CHAR(10)) [message]
                     ,1 AS Counter
              FROM [dbo].[sysssislog] lo
```

Fact-View, die alle Events aus der Tabelle sysssislog dokumentiert. In der „Message"-Spalte werden zwecks besserer Lesbarkeit noch einige Zeilenumbrüche eingefügt.

3 Aufbau der relationalen DeltaMaster-Anwendung

In der relationalen Anwendung werden zunächst alle V_LOGGING-Views gemäß der Typen „Dimension" und „Fakt" aufgenommen und miteinander in Beziehung gesetzt.

Es entstehen die folgenden Dimensionen:

- **Jobausführung:** mit den Ebenen Jahr, Monat, Tag, Ausführung auf Basis der Startzeit. Diese Granularität ist nicht unbedingt nötig, wenn das Log, wie in Abschnitt 1 beschrieben, auf wenige Tage beschränkt wird.

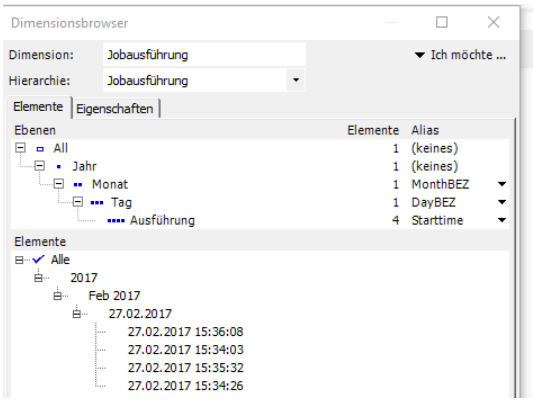


Abbildung 8 Struktur der Dimension "Jobausführung"

- **Ereignis**: listet die unterschiedlichen Ereignis-Typen (z. B. „OnError")

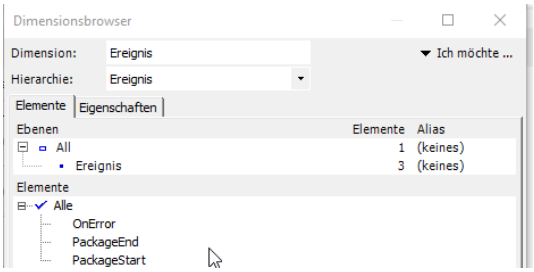


Abbildung 9 Struktur der Dimension "Ereignis"

- **Ereignisquelle**: Komponenten des SSIS-Pakets in denen „Events“ generiert wurden

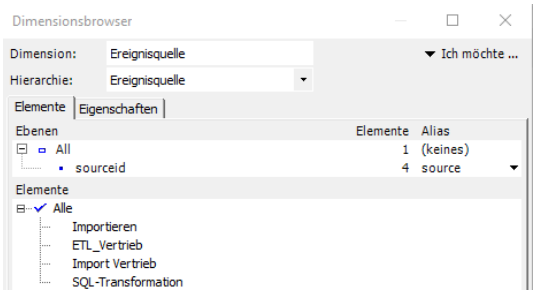


Abbildung 10 Struktur der Dimension "Ereignisquelle"

- **Jobschritt**: Stellt die Jobschritte dar, die im SSIS-Paket angelegt wurden

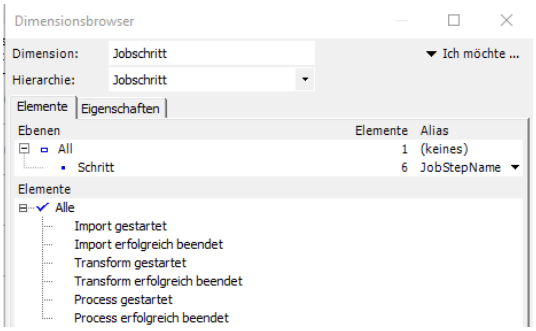


Abbildung 11 Struktur der Dimension "Jobschritt"

Diese Dimensionen werden an die beiden Fact-Views V_LOGGING_FACT_Job und V_LOGGING_ FACT_Logs entsprechend der Dimensionalität angebunden. In der V_Logging_FACT_Job-View wird der Status als Analysewert angelegt.

Anschließend werden zwei Berichte gebaut:

Der Startbericht „Job Übersicht“ zeigt die einzelnen Jobläufe in den Zeilen. In den Spalten sind die einzelnen Jobschritte zu finden. Als Kennzahl wird der „Status“ in den Filter genommen. Da die Kennzahl nicht aussagekräftig ist, verlassen wir uns in diesem Bericht ausschließlich auf das graphische Element „Kreis“ (globalskaliert). Der Wert der Kennzahl wird ausgeblendet, in dem unter Formatierung bei „Benutzerdefiniert“ folgendes eingetragen wird: **"";"";""**

Zusätzlich wird für die Analysesitzung unter „Optionen“ -> „Darstellung“ die Funktion „Grafische Elemente mit wertabhängiger Farbintensität anzeigen“ aktiviert.

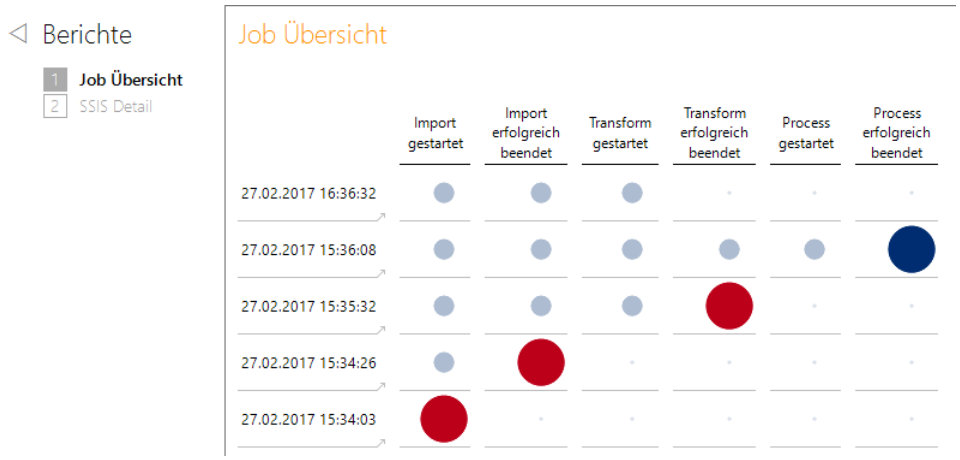


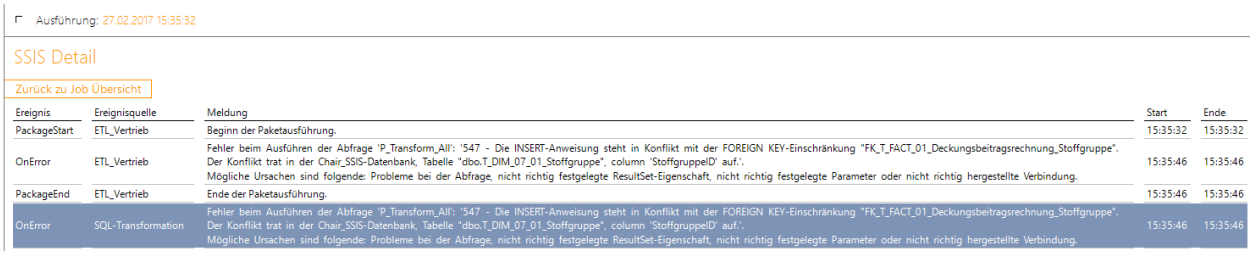
Abbildung 12 Startbericht "Job Übersicht", der Job um 16:36:32 läuft noch

Diese Einstellungen sorgen dafür, dass in Kombination mit den für die View V_LOGGING_FACT_Job beschriebenen Ausnahmen stets der letzte Schritt des Jobs als großer Kreis mit maximaler Farbintensität dargestellt wird. Jobs, die noch ausstehend, sind lediglich als kleine Pünktchen erkennbar und erfolgreich abgeschlossene Schritte, die nicht der letzte Schritt sind, werden als etwas kleinere Kreise dargestellt, die in einem dezenten Blau gehalten sind.

Der Anwender kann sich somit sofort am größten und farbintensivsten Kreis in jeder Zeile orientieren.

Läuft der Job noch, so wird dies dadurch deutlich, dass es keinen großen Kreis mit maximaler Farbintensität gibt (vgl. Zeile 1 in Abbildung 12).

Der Bericht enthält in jeder Zeile eine Verknüpfung auf den SQL-Durchgriffsbericht „SSIS Detail", der auf die jeweilige Ausführung gefiltert die detaillierten Logeinträge aus der Tabelle sysssislog anzeigt.


Ausführung: 27.02.2017 15:35:32

SSIS Detail

Zurück zu Job Übersicht

Ereignis	Ereignisquelle	Meldung	Start	Ende
PackageStart	ETL_Vertrieb	Beginn der Paketausführung.	15:35:32	15:35:32
OnError	ETL_Vertrieb	Fehler beim Ausführen der Abfrage 'P_Transform_All': '547 - Die INSERT-Anweisung steht in Konflikt mit der FOREIGN KEY-Einschränkung "FK_T_FACT_01_Deckungsbeitragsrechnung_Stoffgruppe". Der Konflikt trat in der Chair_SSIS-Datenbank, Tabelle "dbo.T_DIM_07_01_Stoffgruppe", column 'StoffgruppeID' auf.'. Mögliche Ursachen sind folgende: Probleme bei der Abfrage, nicht richtig festgelegte ResultSet-Eigenschaft, nicht richtig festgelegte Parameter oder nicht richtig hergestellte Verbindung.	15:35:46	15:35:46
PackageEnd	ETL_Vertrieb	Ende der Paketausführung.	15:35:46	15:35:46
OnError	SQL-Transformation	Fehler beim Ausführen der Abfrage 'P_Transform_All': '547 - Die INSERT-Anweisung steht in Konflikt mit der FOREIGN KEY-Einschränkung "FK_T_FACT_01_Deckungsbeitragsrechnung_Stoffgruppe". Der Konflikt trat in der Chair_SSIS-Datenbank, Tabelle "dbo.T_DIM_07_01_Stoffgruppe", column 'StoffgruppeID' auf.'. Mögliche Ursachen sind folgende: Probleme bei der Abfrage, nicht richtig festgelegte ResultSet-Eigenschaft, nicht richtig festgelegte Parameter oder nicht richtig hergestellte Verbindung.	15:35:46	15:35:46

Abbildung 13 über Verknüpfung aufgerufener SSIS-Detail Bericht

Im Vergleich zum Einstiegsbericht ist dieser Bericht jetzt deutlich technischer, da hier die Meldungen aus dem SSIS-Paket angezeigt werden. Dass Meldungen teilweise mehrfach auftauchen und sie sich nur durch die Ereignisquelle unterscheiden, ist dem „Zwiebelschalenprinzip" von SSIS geschuldet, das Fehlermeldungen an weiter außenliegende Komponenten weiterreicht.

Abschließend kann noch eine Kachel für die Anwendung definiert werden, die das Ergebnis der letzten Job-Ausführung anzeigt. Die Kachel zeigt an, wann die letzte Paket-Ausführung gelaufen ist, ob diese erfolgreich war oder nicht und ob sie noch läuft.

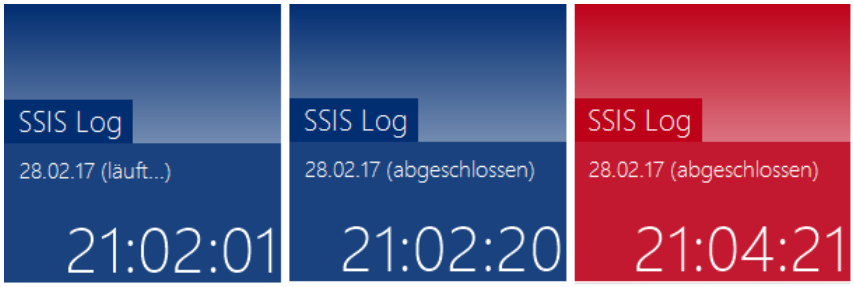


Abbildung 14 Mögliche Ausprägungen der Kachel

Die Definition der Kachel erfolgt in einer xml-Datei die den identischen Namen wie die relationale Anwendung trägt. Im beiliegenden Material ist auch diese Datei enthalten. Soll diese Kachel in einem Projekt eingesetzt werden, muss in der XML-Datei der Parameter ConnectionString entsprechend angepasst werden.

Im Materialordner liegen auch die SQL-Skripte zur Erstellung der relationalen Objekte, die relationale DeltaMaster-Anwendung, sowie die Beispiel-Datenbank und das Beispiel-SSIS-Paket bereit.