Logging

ASYS verwendet für das Logging 'log4j'. Dies bietet eine sehr gute Skalierbarkeit und (für Citrix-Systeme) die Möglichkeit, für jeden Anwender eigene Log-Dateien erstellen zu lassen.

Für die verschiedenen Komponenten ('Anwendungsoberflächen', 'Kommunikationsserver', 'Webservice Transportkontrolle') gibt es hierbei unterschiedliche Property-Dateien, in denen die Loggingkonfigurationen eingestellt werden können.

Die folgende Tabelle gibt einen Überblick, um welche Dateien es sich hierbei handelt und wo diese zu finden sind bzw. abgelegt werden müssen.

Es werden folgende Abkürzungen verwendet:

- [USER_HOME]: Userspezifisches 'ASYS-Dokumente und Einstellungen-Verzeichnis' ('c:\Dokumente und Einstellungen\<Anmeldename> (ggf.+.<Domänenname>)\Asys7'; ist betriebssystemseitig meist auf 'c:\user\<Anmeldename>...' gemappt.
- [ASYS_HOME]: Installationsverzeichnis von ASYS (Standard: 'c:\Programme\Asys7')
- [TOMCAT]: Verzeichnis des Apache-Tomcat-Appletservers für den ASYS-Kommunikationsserver bzw. ASYS-Webserviceserver (z.B. ...\apache-tomcat-9 im Verzeichnis ...\AsysServer\; die Tomcat-Versionsnummer ist Bestandteil des Verzeichnisnamens und variiert daher je nach eingesetzter Version).

Datei	für Komponente	Ort der Datei	Bemerkungen
MbsLog2.properties	Anwendungsoberfläche	[USER_HOME]	s.u.
MbsAdminLog2.properties	Administrator Service-Routinen	[USER_HOME]	s.u.
MbsAdminClientLog2.properties	Admin Datenbank- und Testtool	[USER_HOME]	s.u.
AsysComServer7.properties	Kommunikationsserver	...\AsysServer\[TOMCAT]\conf\	
AsysWebService7.properties	WebService-Server	...\AsysServer\[TOMCAT]\conf\ oder bei Standalone-Lösung: ...\AsysWebService\[TOMCAT]\conf\	
AsysJobServer.properties	Funktionsserver	...\AsysServer\[TOMCAT]\conf\	

MbsLog2.properties

Um nicht für jeden Nutzer eine Log-Datei manuell erstellen zu müssen, bietet ASYS hierfür einen gewissen Komfort.

Beim Start von ASYS erfolgt die Suche nach den 'MbsLog2.properties'

- 1. Im [USER_HOME]-Verzeichnis.
- 2. Darüber hinaus erfolgt die Suche nach einer **Vorlage**-Datei im [ASYS_HOME]-Verzeichnis:

Gibt es im [ASYS_HOME]-Verzeichnis eine 'MbsLog.properties', die jünger ist, als die Datei in [USER_HOME], wird die Datei nach [USER_HOME] kopiert und überschreibt die dort vorhandene Datei; gibt es im [USER_HOME] noch keine 'MbsLog.properties' wird die Datei einfach kopiert. Genutzt wird somit die Datei in [USER_HOME], die Datei in [ASYS_HOME] dient nur als Kopiervorlage.
Nimmt der Fachadministrator Änderungen an der Vorlagendatei vor, so werden die Änderungen für jeden einzelnen Nutzer bei dessen nächstem Login in dessen [USER_HOME]-Verzeichnis übertragen. Dem Fachadministrator wird im Administrator-Programm die Vorlagendatei zur Bearbeitung angeboten.

- 3. Insbesondere für Citrix/WTS-Systeme: In die Pfadnamen für die einzelnen Logger kann ein Platzhalter-Zeichen aufgenommen werden (**<*LOGIN_NAME*>**). Außer dem Kopieren der Datei ersetzt ASYS diesen Platzhalter dann beim Start durch den echten Loginnamen des Anwenders, der sich gerade bei ASYS angemeldet hat.

```
Beispiel:  Aus
...
log4j.appender.debug=org.apache.log4j.FileAppender
log4j.appender.debug.File=logs/asys_<*LOGIN_NAME*>_debug.log
log4j.appender.debug.Append=false
...

wird, wenn sich der Anwender 'itu' anmeldet
...
log4j.appender.debug=org.apache.log4j.FileAppender
log4j.appender.debug.File=logs/asys_itu_debug.log
log4j.appender.debug.Append=false
...
```

- 4. Sofern es eine userspezifische 'MbsLog2_[Loginname].properties' im [ASYS_HOME]-Verzeichnis gibt, wird sie ins [USER_HOME]-Verzeichnis verschoben und in 'MbsLog2.properties' umbenannt (auch dies könnte insbesondere für Citrix/WTS-Systeme interessant sein).

Logging-Einstellungen für einzelne Arbeitsplätze lassen sich auch einfach über den Repository-Administrator ändern.

Allgemeines zum Logging

In der ASYS-Mittelschicht werden vier Logging-Stufen verwendet (dies ist auch durchaus Standard):

- info: Niedrigste Logging-Stufe - wenige Informationen, die einen korrekten Programmablauf anzeigen.
- warn: Warnung - wird verwendet, wenn eine (meist fehlende) Einstellung zu einem Fehler führen könnte.
- error: Fehler - wird 'geworfen', wenn im Programm ein Fehler auftritt.
- debug: Höchste Logging-Stufe - sehr umfangreiche Informationen; in der Implementierungs- und Testphase kann hierüber ein korrekter Programmablauf kontrolliert werden; im Betrieb wird das 'debug' für die Fehlersuche verwendet und ist hierfür häufig essentiell.

Die prinzipielle Einstufung in 'info', 'warn' und 'debug' ('error' ergibt sich normalerweise automatisch) erfolgt im Programmcode und kann auch nur dort geändert werden. Konfiguriert werden kann, welche

Informationen 'geloggt' werden.

Die Konfiguration wird in den oben genannten Property-Dateien vorgenommen.

Die Ausgabe wird hierbei über sogenannte 'Appender' definiert und gesteuert. In ASYS kommen als Appender die Java Console ('ConsoleAppender') und zwei Varianten von Datei-Appendern ('FileAppender' und 'DailyRollingFileAppender') zum Einsatz. Prinzipiell gäbe es auch noch weitere mögliche Appender.

Je nach Appender-Typ sind unterschiedlich viele Konfigurationseinstellungen erforderlich.

Erforderliche Einstellungen für alle Appender:

- log4j.appender.[NAME]=org.apache.log4j.[TYP]
  - NAME: Name des Appenders - sinnvollerweise ein 'sprechender' Name (z.B. 'info' oder 'debug') in Kleinbuchstaben
  - TYP: 'ConsoleAppender', 'FileAppender' oder 'DailyRollingFileAppender'; (einen Appender, der automatisch Mails verschicken könnte, gibt es wohl auch, ist aber im Kontext von nicht getestet worden.)
- log4j.appender.debug.Threshold=[THRESHOLD]
  - THRESHOLD: Welche Logging-Stufe nimmt dieser 'Appender auf'; in ASYS werden normalerweise 'INFO' oder 'DEBUG' eingestellt ('WARN' und 'ERROR' werden automatisch mit geloggt); es könnte prinzipiell aber auch z.B. 'ERROR' eingegeben werden; in diesen Appender würden damit dann nur alle Fehler geloggt werden
- log4j.appender.debug.layout=[LAYOUT]
  - LAYOUT: in ASYS immer nur 'org.apache.log4j.PatternLayout'
- log4j.appender.debug.layout.ConversionPattern=[CONVERSIONPATTERN] (z.B. %d [%t] %-5p %-50c %x - %m%n)
  - CONVERSIONPATTERN: Im Beispiel beinhaltet das Ausgabeformat Datum und Zeit (%d), den aktuellen Thread (%t), die Priorität (%-5p), die Kategorie (%-50c), den „nested diagnostic content“ wenn ein Fehler aufgetreten ist (%x) und die Log-Nachricht gefolgt von einem Zeilenumbruch. Die Modifizierer (-5 bzw. -50) legen fest bis zu welcher maximalen Länge mit Leerzeichen aufgefüllt wird, bzw. Teile der Zeichenkette nicht mit ausgegeben werden (%-50p).

Für die zwei Varianten von Datei-Appendern ('FileAppender' und 'DailyRollingFileAppender') sind zwei weitere Einstellungen vorzunehmen:

- log4j.appender.debug.File=[FILE]
  - FILE: Pfad und Name der Ausgabedatei; der Pfad kann auch relativ angegeben werden (z.B. 'logs/asys_debug.log')
- log4j.appender.debug.Append=[BOOLEAN]
  - BOOLEAN: true oder false; soll beim erneuten Start des Programms der Inhalt der Log-Datei erhalten bleiben (true=Anhängen) oder verworfen werden (false)

Für den speziellen 'DailyRollingFileAppender' (jeden Tag wird eine neue Log-Datei erstellt; die alten Log-Dateien bleiben erhalten) ist eine Konfigurationseinstellung für den (alten) Dateinamen vorzunehmen:

- log4j.appender.info.datePattern=[DATEPATTERN]
  - DATEPATTERN: '.'yyyy-MM-dd'.log'; an den 'File-Namen' der Log-Datei wird eine Endung anhand dieses 'Patterns' angehängt

Für das prinzipielle Logging ist eine weitere wichtige Konfigurationseinstellung vorzunehmen:

- log4j.rootCategory=[ROOTCATEGORY]
  - ROOTCATEGORY: In die RootCategory wird eingetragen, welcher Inhalt (Threshold) als Standard geloggt werden soll (**INFO** oder **DEBUG** in Großbuchstaben); dahinter wird angegeben, welche 'Appender' die Standard-Logger sind (z.B. 'console', 'debug', 'info' (in Kleinbuchstaben - Namen der Appender)); die Standard-Logger nehmen den gesamten Log-Inhalt einer Logging-Stufe (INFO bzw. DEBUG) auf; 'Appender', die nur speziellen Inhalt aufnehmen sollen (SQL, Regeln, VG ...) sind hier nicht einzutragen

Als Standard für die 'RootCategory' sollte immer nur **INFO** eingetragen werden, da bei 'DEBUG' innerhalb kürzester Zeit riesige Log-Dateien entstehen könnten. Das 'DEBUG' sollte nur auf Paket- bzw. Klassenebene eingeschränkt zugeschaltet werden.

Der **rootCategory-Eintrag** sollte also folgendermaßen aussehen:

log4j.rootCategory=INFO, console, info

bzw.

log4j.rootCategory=INFO, console, info, debug

sofern der **debug**-Appender definiert ist.

In den Auslieferungsversionen sind die Ausgaben auf das wesentliche beschränkt.

Ausgabe auf Paket- bzw. Klassenebene:

- Je Paket- bzw. Klasse kann eingestellt werden, welcher Inhalt (Threshold) ausgegeben werden soll ('INFO' oder 'DEBUG' in Großbuchstaben); optional kann noch angegeben werden, ob der Inhalt in einen speziellen Appender ausgegeben werden soll (z.B. ', script, ...' (in Kleinbuchstaben - Name(n) der(s) Appender(s)))

Die wichtigsten Pakete und Klassen, für die das DEBUG ggf. eingestellt werden sollte (für Fehleranalysen der Governikus ITU) stehen in auskommentierten Zustand in der 'ACSComDienst.prop'.

Der Sql- und der Rule-Logger sind selbstdefinierte Kategorien, die bereits im Programmcode definiert sind. Diese Kategorien nehmen Informationen über bestimmte inhaltliche Bereiche auf (der Inhalt kann sich dabei über verschiedene Pakete und Klassen erstrecken). Über eine Schalter läßt sich damit dieser gesamte inhaltliche Bereich in einen bestimmten Appender schreiben.

Beispiele

Beispiel für ein Paket und eine Klasse, die in einen bestimmten Appender schreiben:

```
log4j.appender.skript=org.apache.log4j.FileAppender
log4j.appender.skript.File=logs/asys_script.log
```

```
log4j.appender.skript.Append=false
log4j.appender.skript.Threshold=DEBUG
log4j.appender.skript.layout=org.apache.log4j.PatternLayout
log4j.appender.skript.layout.ConversionPattern=%d %x - %m%n
log4j.logger.de.condat.script=DEBUG, skript (DEBUG des Paketes 'script'
schreibt in den Appender 'skript')
log4j.logger.de.condat.ddo.session.GenericScriptImpl=DEBUG, skript ('DEBUG'
der Klasse 'GenericScriptImpl' schreibt in den Appender 'skript')
```

Beispiel für ein SQL-Log:

```
log4j.appender.sql=org.apache.log4j.FileAppender
log4j.appender.sql.File=logs/asys_sql.log
log4j.appender.sql.Append=false
log4j.appender.sql.Threshold=DEBUG
log4j.appender.sql.layout=org.apache.log4j.PatternLayout
log4j.appender.sql.layout.ConversionPattern=%d %x - %m%n
log4j.logger.Sql-Logger=DEBUG, sql (die 'DEBUG'-Ausgabe der Sql-Logger-
Kategorie wird in den 'Appender' 'sql' (und damit die Datei 'asys_sql.log')
geschrieben)
```

Beispiel für ein DEBUG-Log (die Pakete, die auf DEBUG gestellt werden sollen, werden von der Governikus ITU benannt):

```
log4j.rootCategory=INFO, console, debug (als rootCategory INFO)

log4j.appender.debug=org.apache.log4j.FileAppender
log4j.appender.debug.File=logs/asys_debug.log
log4j.appender.debug.Append=false
log4j.appender.debug.Threshold=DEBUG
log4j.appender.debug.layout=org.apache.log4j.PatternLayout
log4j.appender.debug.layout.ConversionPattern=%d [%t] %-5p %-50c %x - %m%n
log4j.logger.de.condat.mbs=DEBUG (Paket auf DEBUG gestellt)
log4j.logger.de.condat.ddo.session=DEBUG (Paket auf DEBUG gestellt)
```

Beispiel für ein DEBUG-Log, wie es **NICHT** gemacht werden sollte:

```
log4j.rootCategory=DEBUG, console, debug (als rootCategory 'DEBUG'; führt
innerhalb kürzester Zeit zu einer 'riesigen' Log-Datei)

log4j.appender.debug=org.apache.log4j.FileAppender
log4j.appender.debug.File=logs/asys_debug.log
log4j.appender.debug.Append=false
log4j.appender.debug.Threshold=DEBUG
log4j.appender.debug.layout=org.apache.log4j.PatternLayout
log4j.appender.debug.layout.ConversionPattern=%d [%t] %-5p %-50c %x - %m%n
```

Weitere ausführlichere Informationen über das 'log4j' können bei Interesse dem beiliegenden Dokument

Nutzerhandbuch für Log4j 2.x (englisch)

[1] entnommen werden.

[1] Eine deutsche Übersetzung des Handbuchs ist nicht verfügbar.