PuA Release-Notes

Roland Schmitz, Jonathan Haas

9. März 2016

Kappich Systemberatung

Inhaltsverzeichnis

1 Einleitung

Im folgenden Dokument werden die Release-Notes zu PuA in umgekehrter chronologischer Reihenfolge aufgelistet, so dass die Informationen zum letzten Release zuerst aufgeführt werden.

2 PuA 2.10.0

Release-Datum: 09.03.2016

2.1 Änderungen

Folgende Erweiterungen und Änderungen gegenüber vorhergehenden Versionen wurden durchgeführt:

- Die Kommunikation zwischen PuA-Server und PuA-Client wurde dahingehend erweitert, dass nun beim Verbindungsaufbau gegenseitig die Version des verwendeten Kommunikationsprotokolls (aktuell 4) ausgetauscht wird. Zukünftige Änderungen des Kommunikationsprotokolls werden damit vereinfacht.
- Nerz-FM–173: Die Ersetzung von im Skript angegeben Typen durch die konkreten Objekte wurde überarbeitet. Der Typ den Hauptobjekts muss nun nicht mehr mit dem im Skript angegebenem Typ identisch sein, so kann man jetzt auch beispielsweise im Skript den Typ `typ.messQuerschnittAllgemein` verwenden, aber ein Hauptobjekt vom davon abgeleiteten Typ `typ.messQuerschnitt` angeben.
- Nerz-ÄM–103: Es wurden folgende künstliche Beschränkungen bei der Verwendung von temporären Attributen entfernt:
  - Temp-Attribute müssen nicht mehr im Spalten-Bereich angegeben werden.
  - Temp-Attribute müssen nicht mehr zwingend auf reelle Attribute verweisen.

  Dadurch können temporäre Attribute nun auch zur Bestimmung von Zwischenergebnissen oder als Konstanten verwendet werden. Beispiel:

```
spalten:
    tempAttribut messstelleQKfz
definitionen:
    alias mq = typ.messQuerschnitt
    alias abfahrt1 = typ.messQuerschnitt
    tempAttribut mqQKfz: att.verkehrsStärkeStunde =
        mq:atg.verkehrsDatenKurzZeitMq:asp.analyse:QKfz.Wert
    tempAttribut abfahrt1QKfz: att.verkehrsStärkeStunde =
```

```
    abfahrt1:atg.verkehrsDatenKurzZeitMq:asp.analyse:QKfz.Wert
tempAttribut korrekturFaktor: att.zahl = potenz(2, 0.5)
tempAttribut messstelleQKfz: att.verkehrsStärkeStunde =
    (mqQKfz + abfahrt1QKfz) * korrekturFaktor
```

- Nerz-FM–176:

  1. Abbruch des Skripts, wenn bei der Auswertung des Ausdrucks hinter `einschränkung` einen Fehler auftritt. Clientseitig wird eine entsprechende Fehlermeldung mit der Erläuterung des Problems zurückgegeben werden. Serverseitig wird dann eine Debug-Nachricht mit der Erläuterung des Problems und dem Skriptnamen ausgegeben werden.
  2. Der Ergebnistyp eines Ausdrucks kann mit neuen Funktionen in PuA-Skripten geprüft werden. Alle neuen Funktionen liefern (bei Anwendung mit genau einem Argument) einen Wahrheitswert zurück. Folgende Funktionen wurden implementiert:
     - `istFehler()` liefert `true` zurück, wenn das Argument zu einem Fehler ausgewertet wurde.
     - `istZahl()` liefert `true` zurück, wenn das Argument eine Zahl darstellt.
     - `istZustand()` liefert `true` zurück, wenn das Argument einen Zustand darstellt.
     - `istLeer()` liefert `true` zurück, wenn das Argument leer ist (wie z.B. bei einem leeren Datensatz).
     - `istText()` liefert `true` zurück, wenn das Argument eine Zeichenkette darstellt.
     - `istWahrheitswert()` liefert `true` zurück, wenn das Argument einen Wahrheitswert darstellt.
  3. Änderung der Semantik von `und` bzw. `oder` Verknüpfungen. Beide Verknüpfungen liefern nicht nur mehr dann einen Wahrheitswert zurück, wenn beide Operatoren Wahrheitswerte sind, sondern auch dann, wenn nur ein Operator bei der Auswertung einen Wahrheitswert liefert und bei der Auswertung des anderen ein Fehler aufgetreten ist.
     - Die `oder` Verknüpfung liefert genau dann `wahr` zurück, wenn mindestens einer der beiden Operanden `wahr` ist. Auch dann, wenn der andere Operand nicht ausgewertet werden konnte (Fehler).
     - Die `und` Verknüpfung liefert genau dann `falsch` zurück, wenn mindestens einer der beiden Operanden `falsch` ist. Auch dann, wenn der andere Operand nicht ausgewertet werden konnte (Fehler).

  - Wenn beide Operatoren nicht ausgewertet können, dann liefern sowohl `und` als auch `oder` einen entsprechenden Fehler zurück.

4. Änderung der Semantik von Vergleichsoperatoren, wenn einer der Operatoren ein Zustand und der andere eine Zahl ist. Wenn ein Zustandswert mit einem Zahlenwert verglichen wird, dann wird bei den Operatoren *kleiner*, *kleiner-gleich*, *größer*, *größer-gleich* und *gleich* der Wahrheitswert `falsch` bzw. beim Operator *ungleich* der Wahrheitswert `wahr` zurückgeliefert.

3 PuA 2.9.0

Release-Datum: 07.01.2016

3.1 Änderungen

Folgende Erweiterungen und Änderungen gegenüber vorhergehenden Versionen wurden durchgeführt:

- Nerz-Ä–98: Die bisherige `getIsActive()`-Methode im PuA-Client wurde durch einen Listener-Mechanismus ersetzt, der den aktuellen Status von PuA und dem von PuA verwendeten Archivsystem ohne Wartezeit bestimmt. Hierzu gibt es die neuen Methoden `isPuaOnline()`, `isArchiveOnline()`, `addStatusListener(listener)` und `removeStatusListener(listener)`. Voraussetzung ist ein gleichzeitiges Update des PuA-Servers, da sonst dynamische Änderungen beim Zustand des Archivsystems nicht korrekt bestimmt werden können. Ob PuA läuft, kann unabhängig von der Version des Servers in jedem Fall mit `isPuaOnline()` bestimmt werden.
- Nerz-Ä–98: Ebenso wie der Status-Listener können jetzt auch die Protokoll-Listener (`addProtocolListener(listener)`) clientseitig jederzeit registriert werden, unabhängig davon ob PuA aktuell läuft oder nicht. Der eigentliche Listener beim PuA-Server wird automatisch angemeldet sobald PuA verfügbar ist und automatisch abgemeldet, sobald PuA nicht mehr verfügbar ist.
- Nerz-Ä–99: Die Methoden `PuaClient.suspendProtocol()` und `PuaClient.resumeProtocol()`, die zur manuellen Flusskontrolle gedacht waren, werden nicht mehr benötigt und wurden auf `@deprecated` gesetzt, weil mittlerweile eine automatische Flusskontrolle zwischen PuA-Server und PuA-Client implementiert ist.
- Nerz-Ä–100: Die Klasse ProtocolResultData, die clientseitig eine Ergebniszeile abbildet, wurde überarbeitet und dokumentiert. Es gibt jetzt neue Methoden, um für jeden Spaltenindex die folgenden Attribute abzufragen:
  - Datenzeit
  - Archivzeit

  - Datenindex
  - Datenzustand (Keine Daten, Keine Quelle, Fehler, Leer, ...)
  - Archivdatenzustand

  Temporäre Attribute erben den Datenzustand (soweit sinnvoll) von den Attributen, die in die Berechnung eingehen. Kann eine Berechnung beispielsweise nicht durchgeführt werden, weil ein Eingangswert "Keine Daten" hat, dann hat das Ergebnis ebenfalls den Zustand "Keine Daten".

  Der Datenzustand kann nur abgefragt werden, wenn der PuA-Server aktualisiert wird (und ein neues Protokoll erstellt wird). Sonst wird immer der Zustand `PuADataState.NOT_AVAILABLE` zurück gegeben.

- Nerz-Ä–100: Die Berechnungen, die der PuA-Server bei der Auswertung von Skripten durchführt, wurden überarbeitet, sodass nach Möglichkeit in Fehlersituationen eine sinnvolle Fehlermeldung und ein sinnvoller Datenzustand an den Client übertragen wird.

  Die Methode `ResultValue.performOperationOn()` wurde von PuA nicht verwendet und aufgrund der mangelhaften Fehlerbehandlung auf `@Deprecated` gesetzt. Als Ersatz sollte (beispielsweise bei der Implementierung von benutzerdefinierten Funktionen und Aggregationen) die Klasse `de.bsvrz.pua.prot.util.ArithmeticOperation` oder `de.bsvrz.pua.prot.util.ArithmeticOperationEx` verwendet werden.

- Die Abfrage nach Skript-Standardwerten gibt die Aliase, Pseudoaspekte und Aspektbindungen nun in der Reihenfolge zurück, in der sie im Skript definiert wurden. Hierfür müssen sowohl Server als auch Client die neue PuA-Version und Kernsoftware ab 3.7.0 verwenden. Ansonsten ist die Reihenfolge wie bisher undefiniert.

- Die PuA-Datenobjekte aus `ProtocolResultData.getData()` erhalten jetzt eine klarere toString()-Ausgabe bei Fehlerwerten. Die zurückgegebenen Objekte vom Typ `ProtocolData` besitzen jetzt die neue Methode `isError()` mit denen abgefragt werden kann, ob es sich bei diesem Wert um einen Fehlerwert handelt. Die Fehlermeldung selbst kann dann wie bisher mit `asTextValue().getValueText()` abgerufen werden.

- Zur Vereinfachung von Berechnungen können Wahrheitswerte jetzt mit den Werten Wahr=1 und Falsch=0 (entsprechend `att.jaNein`) in Berechnungen eingehen. Hierdurch ändern sich die standardmäßigen Aggregationsfunktionen, sodass beispielsweise bei der Summe-Aggregation als Ergebnis die Anzahl der wahren Werte berechnet wird und bei der Durchschnittsaggregation der Anteil der wahren Werte. Diese Werte kann man entsprechend auch über das ProtocolData-Interface abfragen (z. B. als skalierten Zahlenwert).

- Die Klasse PuaClient implementiert jetzt die Schnittstelle `AutoCloseable` und bietet daher die Methode `close()`, mit der sämtliche belegte Resourcen und Datenverteiler-Anmeldungen des PuA-Clients freigegeben werden können.
- Die Klasse ProtocolResultStream implementiert jetzt die Schnittstelle `AutoCloseable` und bietet daher die Methode `close()`, mit der die Übertragung der Protokolldaten beendet wird, sofern der Client noch nicht alle Daten entgegengenommen hat. Die bisherige Methode `abort()` mit der gleichen Funktionalität wurde auf `@Deprecated` gesetzt, da der Name vermuten lässt, die gesamte Protokollerstellung würde abgebrochen werden, was nicht der Fall ist.
- ProtocolResultStream.getJobId() liefert jetzt nur noch `null` zurück, wenn es keine gültige Auftrags-Id gibt, PuA also beispielsweise den Auftrag nicht angenommen hat. Es muss nicht mehr selbst auf die Übertragung der Auftrags-Id gewartet werden.
- Mit dem neuen Aufrufargument `-clientTimeout=` kann ein Minutenwert angegeben werden, nach dem bei nicht reagierenden Clients ein Protokollabruf automatisch von PuA abgebrochen wird. Wenn das Argument nicht verwendet wird, dann wird als Default-Wert 15 (Minuten) verwendet. Mit dem Wert `-1` kann der Timeout deaktiviert werden.
- Nerz-F–160: Wenn in einem Protokoll mehrere Archivdatenarten („Online-Daten“, „nachgeforderte Online-Daten“, „nachgelieferte Daten“ und „nachgeforderte nachgelieferte Daten“) angefordert wurden, wurden diese Archivdatenarten von PuA einzeln nacheinander abgefragt, was dazu führte, dass diese im Protokoll nacheinander und nicht zeitlich sortiert ausgegeben wurden. Das Verhalten wurde so geändert, dass die Archivdatenarten gemeinsam abgefragt werden und das Archivsystem die Daten entsprechend nach Datenzeit sortiert.

3.2 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- Nerz-F–157: Die Verwendung des Moduls SimpleSender in PuA wurde durch ein verbessertes Modul ersetzt, wodurch Benachrichtigungen (z. B. über fertiggestellte Protokolle) an Clients nun asynchron versendet werden und PuA bei negativer Sendesteuerung nicht mehr in Timeouts läuft. Clients, die sich beenden, ohne entsprechende Listener abzumelden, werden nun von PuA automatisch aus den internen Datenstrukturen entfernt und verursachen keine Blockierungen mehr.
- Nerz-F–158: Bei dem Abruf eines nicht vorhandenen Protokolls wurde die Fehlermeldung des PuA-Servers unter der falschen ID verschickt, wodurch der Client diese nicht zurordnen konnte und in einen Timeout lief. Hier wurde sowohl

der PuA-Server angepasst, sodass dieser die richtige ID zum Versand der Fehlermeldung benutzt, als auch der Client, der jetzt auch Fehlermeldungen unter der falschen ID (Anfrage-ID statt Auftrags-ID) empfängt.

- Nerz-F–159: In der Klasse ArchiveUser wurden mehrere Thread-Synchronisierungsprobleme korrigiert, die zu einem Absturz des PuA-Servers führen konnten.
- Nerz-F–160: Wenn mehr als ein Zeitintervall und mehrere Archivdatenarten angefragt wurden, lieferte PuA nur für das erste Zeitintervall alle Archivdatenarten.
- Die Aggregation von Boolean-Werten in Attributlisten oder Arrays konnte fehlerhafte Werte zurückliefern.
- Bei Protokollen mit Archivanfragen konnte es bei vielen abgefragten Daten unter bestimmten Umständen zu einer fehlerhaften Wiederholung von Datensätzen am Ende vom Protokoll kommen.
- Beim clientseitigen Abbruch des ProtokollResultStreams wird nun explizit eine Nachricht an den PuA-Server zum Protokollabbruch gesendet, statt sich darauf zu verlassen, dass PuA die Protokollerstellung automatisch anhält.

4 PuA 2.8.4

Release-Datum: 19.05.2015

4.1 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- Beim Beenden von PuA konnte es wegen eines Thread-Synchronisierungsfehlers dazu kommen, dass einzelne Threads von PuA nicht beendet wurden und entsprechende Warnungen ausgegeben wurden.

5 PuA 2.8.3

Release-Datum: 18.05.2015

5.1 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- Beim Start von PuA konnte es wegen eines Thread-Synchronisierungsfehlers zu einer ConcurrentModificationException kommen.

6 PuA 2.8.2

Release-Datum: 12.05.2015

6.1 Hinweis

Das bisher zusammen mit PuA ausgelieferte Distributionspaket `de.bsvrz.sys.funclib.losb` wird seit Version 3.6.3 mit der Kernsoftware ausgeliefert.

6.2 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- PuA hat versucht, den Applikationsnamen in den ClientDavParameters nach dem Verbindungsaufbau mit dem Datenverteiler zu ändern. Dies hat mit neueren Kernsoftware-Version (>=3.6.3) zu Warnungen bzw. Fehlermeldungen geführt.
- Der optionale Aufrufparameter "-konfigurationsVerantwortlicher=", mit dem man das für Auswertungen zu verwendende Archivsystem angeben kann, wurde nicht an allen erforderlichen Stellen richtig ausgewertet.

7 PuA 2.8.1

Release-Datum: 09.10.2014

7.1 Voraussetzungen

7.1.1 Datenmodelle

Die Nutzung der erweiterten Funktionalität setzt eine Aktualisierung folgender Konfigurationsbereiche vorraus:

- kv.bea:
  - kb.tmVewProtokolleGlobal in Version 5
- kv.kappich:
  - kb.systemModellGlobal in Version 33

7.1.2 Bibliotheken

Es wird eine aktualisierte Version des Distributionspakets `de.bsvrz.sys.funclib.losb` mit gleichem (oder neuerem) Releasedatum vorrausgesetzt.

7.2 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- Ich seltenen Fällen konnte es passieren, dass die Flusskontrolle im PuA-Server den Datenversand zu lange gebremst hat. Der PuA-Server setzt den Versand von Datensätzen nun unmittelbar fort, sobald eine Flusskontroll-Nachricht eingetroffen ist.

8 PuA 2.8.0

Release-Datum: 08.10.2014

8.1 Voraussetzungen

8.1.1 Datenmodelle

Die Nutzung der erweiterten Funktionalität setzt eine Aktualisierung folgender Konfigurationsbereiche vorraus:

- kv.bea:
  - kb.tmVewProtokolleGlobal in Version 5
- kv.kappich:
  - kb.systemModellGlobal in Version 33

8.1.2 Bibliotheken

Es wird eine aktualisierte Version des Distributionspakets `de.bsvrz.sys.funclib.losb` mit gleichem (oder neuerem) Releasedatum vorrausgesetzt.

8.2 Neue Features

Folgende Erweiterungen gegenüber vorhergehenden Versionen wurden durchgeführt:

- Nerz-Ä–13: PuA verwendet eine Warteschlange für Protokollaufträge, wenn keine freien Threads zur Bearbeitung mehr frei sind. Die Klasse PuaClient und analog der PuA-Server wurden um eine Methode erweitert den Inhalt dieser Warteschlange und die aktuell in Bearbeitung befindlichen Aufträge abzurufen. Hierfür wurde am PuaClient die Methode `getJobList()` hinzugefügt. Über die zurückgegebenen JobInProgress-Objekte können damit alle weiteren Informationen über die Aufträge abgefragt werden.
  - Jeder Protokollauftrag enthält nun zur Identifizierung eine eindeutige Auftrags-ID, die die Protokoll-ID clientseitig an mehreren Stellen ersetzt.
    * Bei Protokoll-Erstellen-Aufträgen ist die Auftrags-ID identisch zur Protokoll-ID des erstellten Protokolls.
    * Bei Aufträgen, die gespeicherte oder ungelesene Protokolle abrufen, ergibt sich eine neue eindeutige Auftrags-ID.
    * Die Auftrags-ID lässt sich jeweils über die neue Methode `ProtocolResultStream.getJobId()` abrufen.
  - Die Methode `abortProtocol()` wurde so erweitert, dass mit der Methode nun beliebige protokollbezogene Aufträge anhand ihrer Auftrags-ID (JobId) abgebrochen werden können. Zudem können nun auch Aufträge abgebrochen werden, die sich noch in der Warteschlange befinden.
  - Die Methode `getStatus()` arbeitet nun ebenfalls mit Auftrags-IDs und wurde so erweitert, dass nun auch der Status von Aufträgen in der Warteschlange abgerufen werden kann.
  - Die Methoden zur Flusskontrolle `suspendProtocol()`/`resumeProtocol()` arbeiten nun ebenfalls mit Auftrags-IDs und erlauben auch das Anhalten und Fortsetzen von anderen protokollbezogenen Aufträgen wie dem Abruf von Protokollen.
- Nerz-Ä–14: Die bisherige Listener-Funktionalität im PuaClient wurde erweitert und erlaubt es nun auch, über das Gelöschtwerden und Gelesenwerden eines Protokolls informiert zu werden. Hierzu kann das bisherige Interface `ProtocolListener` durch das neue Interface `ProtocolListenerEx` ersetzt werden wodurch die zusätzlichen Methoden `protocolRemoved()` und `protocolRead()` bereitgestellt werden, die beliebig implementiert werden können. An den Funktionsaufrufen im PuaClient ändert sich nichts, die Methode `addProtocolListener()` wurde überladen und akzeptiert beide Interfaces. Die bisherige Methode `protocolFinished()` wurde im neuen Interface in `jobFinished()` umbenannt und enthält nun (sofern auch ein aktueller PuA-Server eingesetzt wird) umfangreiche Informationen über den fertiggestellten Auftrag, beispielsweise den Typ des Auftrags (Protokollerstellung, Protokollabruf, etc.), ob er erfolgreich abgeschlossen war und ob ein Protokoll als Resultat des Auftrags gespeichert wurde.

- Nerz-Ä–14: Mit der Methode `PuaClient.deleteSaveProtocol()` können nun auch ungelesene Protokolle gelöscht werden.
- Nerz-Ä–14: Der PuaClient verwendet in Verbindung mit einem aktuellen PuA-Server nun eine automatische Flusskontrolle, die die Anzahl der serverseitig versendeten Protokolldaten bei Protokoll-Erstellen- und Protokoll-Abrufen-Aufträgen begrenzt. Dafür wurde ein "Sliding-Window"-Mechanismus implementiert, welcher automatisch verwendet wird. Um die Parameter der Flusskontrolle einzustellen oder die automatische Flusskontrolle optional zu deaktivieren gibt es die neuen Funktionen `PuaClient.get/setFlowControlWindowSize()`.
- Nerz-Ä–15: Bei Zustandsprotokollen besteht nun die Möglichkeit, anzugeben, ob "Keine Änderung"- bzw. NoChange-Markierungen je Zeile (wie bisher) oder je Zelle/Datensatz generiert werden sollen. Hierfür gibt es in der Klasse `ProcessingParameter` die neuen Methoden `get/setNoChangeMarker`, mit denen die Art der Markierung festgelegt werden kann.
  - Daraus folgend kann nun in PuA-Scripten nun auch eine der beiden Markierungsarten als Standard festgelegt werden. Hierzu gibt es im Standards-Block die neuen Definitionen `unveraendertkennung pro zeile` und `unveraendertkennung pro zelle`. (Statt `unveraendertkennung` ist auch die Variante mit `ä` möglich.)
  - Bei Verwendung der neuen zellenweisen Markierungen werden keine zeilenweisen `ProtocolNoChanges`-Datensätze mehr übertragen, stattdessen erhalten die einzelnen Datumswerte, bei denen sich keine Änderungen zum vorherigen Wert ergeben haben, im ExpressionResult die neue Kennung `ResultType.NO_CHANGE`. Beim Zugriff über `ProtocolData`-Objekte gibt die `isDefined()`- Methode bei solchen Datensätzen `false` zurück und die neue Methode `ProtocolData.isNoChange()` liefert `true`.
  - Einzelne Datensätze mit NoChange-Markierung werden standardmäßig durch ein einzelnes Hochkomma (`'`) dargestellt.
  - Damit erfolgreich die neue Markierungsart verwendet werden kann, muss sowohl der Client als auch der Server aktualisiert werden. Wird ein neuer Client und ein veralteter PuA-Server eingesetzt, wird der Server immer zeilenweise Markierungen erstellen. Ruft ein veralteter Client ein Protokoll mit datensatzweisen NoChange-Markierungen ab, erscheinen diese für den Client so, als wären dort keine Daten vorhanden.
- Nerz-Ä–52: Es wurde die Möglichkeit hinzugefügt, Ereignisprotokolle zu erzeugen. Ereignisprotokolle bieten die Möglichkeit, Daten genau so abzurufen, wie sie im Archivsystem stehen, insbesondere werden Datensätze nicht aufgefüllt und identische Datensätze werden nicht durch NoChange-Markierungen ersetzt. Dazu wurden in der ProcessingParameter-Klasse neue Methoden `get/setProtocolType()`

ergänzt, mit denen diese neue Protokollart festgelegt werden kann. Die bisherigen Methoden zur Festlegung der Protokollart `get/setDeltaProtocol()` werden daher nur noch aus Kompatibilitätsgründen unterstützt.

- Daraus folgend kann nun in PuA-Scripten nun auch die Protokollart "Ereignisprotokoll" als Standard festgelegt werden. Zum Abruf des neuen Standardwertes wurde die Klasse `AtlDefaults` analog um die Methode `getProtocolType()` ergänzt. Die bisherige Methode `getDeltaProtocol()` sollte nicht mehr verwendet werden.
- Damit erfolgreich Ereignisprotokolle erstellt werden können, muss sowohl der Client als auch der Server aktualisiert werden. Wird nur der Client aktualisiert, erstellt der Server stattdessen Zustandsprotokolle. Wird der Server aktualisiert kann ein alter Client gespeicherte Ereignisprotokolle lesen, die Methode `getDeltaProtocol()` liefert dann `false`. Zum Abruf des Default-Wertes "Ereignisprotokoll" muss außerdem das verwendete Datenmodell `kb.tmVewProtokolleGlobal` aktualisiert werden.

8.3 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- `ExpressionResult` hatte einen statischen `NumberFormat`-Member. Da die Java-Klasse NumberFormat nicht threadsicher ist, konnte das zu seltenen, zufällig auftretenden, Problemen wie falschen Textausgaben und Exceptions führen.
- Serialisierung der Klasse `DataInformation` angepasst, sodass private Attribute wie Protokollart nicht mehr überflüssigerweise an den Client übertragen werden, was die Performance verbessern sollte.
- Synchronisierung in ThreadPool-Klasse korrigiert, Methode `getWaitingProtocols()` war nicht threadsicher und konnte z. B. eine `ConcurrentModificationException` erzeugen.
- Klasse Processing hat auf `Boolean.TRUE/Boolean.FALSE` synchronisiert, was zu schwer diagnostizierbaren Problemen wie Deadlocks führen konnte.
- Bei `PuaClient.abortProtocol()` wurde nicht in jedem Fall eine entsprechende (Fehler-)Meldung ("Der Versand von Protokolldatensätzen wurde abgebrochen") an den Client übertragen, der gerade das Protokoll abruft. Diese Fehlermeldung wurde ergänzt.
- Die Erzeugung von Protokoll-IDs wurde angepasst und ist nun unabhängig von irgendwelchen Zeitstempeln, sodass eventuell auftretende Probleme durch Erzeugung von mehreren Protokollen zur exakt selben Zeit vermieden werden.

- Beim PuaClient konnten Protokollantworten mit Benachrichtigungen zu fertiggestellten Protokollen durcheinandergebracht werden, was bei der Erweiterung der Benachrichtigungen gemäß Nerz-Ä–14 zu Exceptions bei Protokollanfragen geführt hätte. Das Problem wurde korrigiert, alte Clients sind nicht betroffen da erweiterte Benachrichtigungen nur an neue Clients gesendet werden.
- Nerz-Ä–15: Die Interpretation des Zeitstempels des Ende-Archiv-Datensatzes wurde angepasst. Der tatsächliche Zeitstempel des Ende-Archiv-Datensatzes wird ignoriert und für die weitere Verarbeitung durch den Zeitstempel des Intervallendes ersetzt. Dies korrigiert die bisher häufig vorkommenden überflüssigen Datensätze und Lücken gegen Ende eines Intervalls.
- Nerz-Ä–15: Datenindexsprünge in Änderungsprotokollen (die beispielsweise archivseitig durch weggelassene, identische Daten bei Delta-Anfragen verursacht werden können) werden jetzt nicht mehr fäschlicherweise als Datenlücke erkannt. Dadurch wurde das Auffüllen bei Änderungsprotokollen korrigiert und es werden keine überflüssigen "LEERER DATENSATZ"- Datensätze mehr erzeugt.
- Nerz-Ä–15: Beim Vergleich von Datensätzen auf Gleichheit werden Zahlwerte nicht mehr nach Double konvertiert. Dies behebt Probleme die beim Vergleich von großen Objekt-IDs aufgetreten sind, da sich Long-Werte nicht verlustfrei auf Doubles abbilden lassen.
- Die Logik zum Auffüllen von Datensätzen wurde korrigiert. Fehlerdatensätze, Enddatensätze und ähnliches werden nicht mehr zum Auffüllen verwendet.
- Der PuaClient und der PuA-Server verhalten sich jetzt besser bei der Kommunikation über gekoppelte Datenverteilersysteme, insbesondere warten die Klassen SimpleSender und Sender jetzt eine gewisse Zeit auf positive Sendesteuerung, da die Suche nach Zentraldatenverteilern und Übermittlung der Sendesteuerung an den Zieldatenverteiler u.U. eine kurze Zeit benötigt.
- In Skripten kam es zu Fehlern, wenn Spaltennamen, Definitionen von temporären Attributen und ähnliches das Schlüsselwort "`nicht`" enthielten. Die Grammatik-Definition wurde korrigiert.

9 PuA 2.7.0

Release-Datum: 25.02.2014

9.1 Änderungen

Folgende Änderungen zu vorhergehenden Versionen wurden durchgeführt:

- Nerz-Ä–12: Begrenzung der Anzahl parallel zu verarbeitenden Archivanfragen seitens einer einzelnen Applikation.
  - Die im Archivsystem eingestellte maximale Anzahl von gleichzeitig für eine Applikation bearbeiteten Archivanfragen wird zur Laufzeit abfragt und entsprechend berücksichtigt.
  - PuA wurde so erweitert , dass eine einzelne PuA-Anfrage nicht mehr zu mehreren gleichzeitig ausgeführten Archivanfragen führt, sondern eine einzelne Archivanfrage mit entsprechend vielen verschiedenen Teilanfragen gestellt wird.
- Nerz-Ä–16: Performance-Verbesserungen bei der Verwendung von ByteArrays in den Datenverteiler-Applikationsfunktionen. Verwendung der neuen Methoden des Data-Interface zum Setzen von `byte[]`, `short[]`, `int[]` etc.
- Nerz-Ä–6: Bei versionierten Konfigurationsänderungen werden für die betroffenen Objekte neue Konfigurationsobjekte mit neuer ID und gleicher Pid in der Konfiguration erzeugt. Um eine Archivanfrage für ein so geändertes Objekt über den Versionswechsel hinaus durchzuführen, musste die anfragende Applikation bisher die verschiedenen Konfigurationsobjekte mit der gewünschten Pid im Anfragezeitbereich ermitteln und alle gefundenen Objekte in der Archivanfrage angeben.
  - Die vom PuA-Server durchgeführten Archivanfragen wurden so angepasst, dass das neue API für Archivanfragen verwendet wird, bei dem die Pids der gewünschten Objekte angegeben werden.

9.2 Bugfixes

Folgende Probleme vorhergehender Versionen wurden behoben:

- Nerz-F–83: Wenn das Archiv auf eine Protokollanfrage von PuA nicht antwortet kann es passieren, dass das Abbrechen einer Protokollanfrage nicht dazu führt, dass die Archivanfrage abgebrochen wird. Zur Lösung des Problems wurde der Code in PuA so angepasst, dass bei Abbruch einer Auswertung der auf Archivantworten wartende Thread in PuA unterbrochen wird und die Archivanfrage abgebrochen wird.