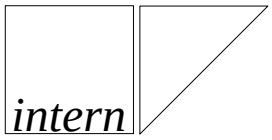


Thomas Bollmeier
IBS Financial Services
– Tools & Technology

Framework zur Parallelverarbeitung in der IBS Financial Services

Entwicklerdokumentation

28. Mai 2004

Version: 2.1

Änderungen zu Version 2.1.:

- Kapitel 4.12 : Ausnahme NOT_FOUND in Methode 1100
- Kapitel 4.17: neuer Parameter I_MSG_HANDLE in Methode 1400
- Kapitel 5.4: Erweiterungen zur Schnittstelle der FuB BANK_MAP_PP_START bzw. BANK_MAP_PP_RESTART
- Kapitel 5.6 „Nachrichtenbehandlung innerhalb der Parallelverarbeitung" hinzugefügt

Inhaltsverzeichnis:

1 Einleitung 4

2 Architektur des Parallelverarbeitungstools (PV-Tool) 4

2.1 Konzept 4

2.2 Begrifflichkeiten 5
2.2.1 Objekte 5
2.2.2 Paketvorlagen und Paketverwalter 5
2.2.3 Arbeitspakete und Arbeitsvorrat 5
2.2.4 Anwendungsart 5
2.2.5 Massenlauf 5

2.3 Klassendiagramm 6

3 Ablauf der Parallelverarbeitung 8

3.1 Vorbereitung und Start 8

3.2 Parallele Teilprozesse 8

3.3 Synchronisation 8

3.4 Objektsperren 8

3.5 Objektstatusänderungen und LUWs 9

3.6 Diagramme 10

4 Methoden der PV-Schnittstelle 13

4.1 Methode 0100: Start eines Massenlaufs (optional) 13

4.2 Methode 0110: Parameter eines Laufs holen (optional) 15

4.3 Methode 0120: Parameter setzen (optional) 15

4.4 Methode 0130: Parameter löschen (optional) 16

4.5 Methode 0150: Massenlauf für Restart auswählen/suchen (optional) 16

4.6 Methode 0160: Prüfungen vor Start eines Massenlaufs (optional) 17

4.7 Methode 0205: Paketvorlagen erzeugen 18

4.8 Methode 0206: Parameter zu Paketen sichern (optional) 19

4.9 Methode 0207: Parameter zu Paketen löschen (optional) 19

4.10 Methode 0300: Am Ende des Massenlaufs (optional) 20

4.11 Methode 1000: Initialisierung eines Arbeitspaketes 21

4.12 Methode 1100: Selektion von Anwendungsdaten 22

4.13 Methode 1200: Nachselektion von Anwendungsdaten 22

4.14 Methode 1260: Prüfung, ob Objekte gesperrt sind (optional) 23

4.15 Methode 1270: Löschen von Objekten aus Anwendungsdaten 24

4.16 Methode 1300: Objekte bearbeiten 24

4.17 Methode 1400: Start der Verarbeitung in einem parallelen Job (optional) 24

4.18 Methode 1410: Ende der Verarbeitung in einem parallelen Job (optional) 25

5 Anbindung einer Anwendung an die Parallelverarbeitung 25

5.1 Implementierung der PV-Schnittstelle 25

5.2 Persistente Speicherung der Anwendungsparameter 25

5.3 Customizing 26
5.3.1 PV-Klasse 26
5.3.2 Anwendungsart 26
5.3.3 Zuordnung von Funktionsbausteinen zu Methoden der PV-Schnittstelle 27
5.3.4 Kundencustomizing 27

5.4 Aufruf der Start-Funktionsbausteine 28
5.4.1 Erststart eines Massenlaufs 28
5.4.2 Wiederaufsetzen auf einer Massenverarbeitung 30

5.5 Wiederverwendung von Paketeinteilungen 32
5.5.1 Beispiel für die Generierung einer Paketeinteilung 33

5.6 Nachrichtenbehandlung innerhalb der Parallelverarbeitung 34

1 Einleitung

Diese Dokumentation gibt eine Einführung in die Grundlagen des Tools zur Parallelverarbeitung (PV), das im Bereich Financial Services zur Verfügung steht. Es werden die Schritte beschrieben, die ein Entwickler ausführen muß, um die Parallelverarbeitung einer Anwendung mit Hilfe dieses Tools zu implementieren.

2 Architektur des Parallelverarbeitungstools (PV-Tool)

2.1 Konzept

Das PV-Tool definiert eine Schnittstelle für Anwendungen zur Implementierung einer Parallelverarbeitung. Hierzu werden durch die Anwendung voneinander unabhängige Pakete gebildet. Das PV-Tool übernimmt dann die Abarbeitung dieser Pakete innerhalb einer parallel laufenden Menge von Hintergrundjobs. Diese Jobs werden vom PV-Tool automatisch generiert. Die Anzahl der parallelen Jobs kann kundenseitig im Customizing eingestellt werden. Das PV-Tool sorgt außerdem für eine Synchronisation der parallelen Jobs.

2.2 Begrifflichkeiten

2.2.1 Objekte

Das PV-Tool bietet eine Verwaltung des Verarbeitungsstatus einzelner Objekte. Damit verbunden ist eine Wiederaufsetzfähigkeit der Massenverarbeitung – d.h. nach Fehlern werden nur noch die offenen Objekte erneut bearbeitet.

Das PV-Tool macht keine Vorgaben über die Art der zu verarbeitenden Objekte. Objekte können z.B. Konten oder Aufträge sein. Um die verschiedenen Objekttypen unterscheiden zu können, werden im Customizing des PV-Tools Parallelverarbeitungsklassen (PV-Klassen) definiert. Die einzelnen Objekte werden vom PV-Tool über einen Schlüssel bestehend aus PV-Klasse und Objektnummer angesprochen. Die PV-Klasse enthält auch die Methoden zur Konvertierung des Objektschlüssels der Anwendung in den vom PV-Tool verwendeten und umgekehrt. Im Customizing müssen hierzu die Namen der jeweiligen Funktionsbausteine zur Konvertierung hinterlegt werden.

2.2.2 Paketvorlagen und Paketverwalter

Um eine Anwendung parallel verarbeiten zu können, muß es möglich sein, die Menge der zu bearbeitenden Objekte in voneinander unabhängige Paketvorlagen einzuteilen. Nach welchen Kriterien diese Einteilung vorgenommen werden soll, wird durch die Anwendung festgelegt (siehe 4.7). Das PV-Tool unterstützt mit eigenen DB-Tabellen die Definition der Paketvorlagen als Intervalle bezüglich einer Sortiernummer mit Angabe von Ober- und Untergrenze (40-stelliges Character-Feld) sowie die Definition als diskrete Liste von Objektschlüsseln. Sollen die Paketvorlagen nach anderen Kriterien gebildet werden, muß die Anwendung hierzu eigene Datenbanktabellen verwenden.

Die Gesamtmenge aller Paketvorlagen wird im Folgenden als Paketverwalter bezeichnet.

2.2.3 Arbeitspakete und Arbeitsvorrat

Da Paketvorlagen von mehreren Massenläufen gleichzeitig genutzt werden können, generiert das PV-Tool bei einem neuen Massenlauf zu jeder Paketvorlage ein Arbeitspaket, das die laufspezifischen Statusdaten enthält. Neben diesen Kopfdaten gehört zu einem Arbeitspaket auch die Liste der noch nicht abschließend bearbeiteten Objekte. Beim Wiederaufsetzen der Verarbeitung wird diese Objektliste herangezogen. Die Gesamtmenge aller Arbeitspakete bildet den Arbeitsvorrat.

Beispiel für einen Objekteintrag im Arbeitsvorrat(Schlüsselfelder fett):

Mandant	**Arbeitsvorrat-ID**	**Paketnummer**	**PV-Klasse**	**Objektnummer**	Bearbeitungsstatus
001	056103ABF	1234	1	4711	zurückgestellt

2.2.4 Anwendungsart

Während die PV-Klasse festlegt, *was* parallel verarbeitet wird, definiert die Anwendungsart, *wie* die Verarbeitung zu erfolgen hat. Beispiele für Anwendungsarten sind Kontoabschluß, Kontoauszugserstellung oder Cash Concentration. Auf Ebene der Anwendungsart wird definiert, wie die Methoden der PV-Schnittstelle implementiert werden.

2.2.5 Massenlauf

Als Massenlauf wird die Verarbeitung einer Menge von PV-Objekten zu einer Anwendungsart bezeichnet, die zu einem Zeitpunkt gestartet wurde.

Der Massenlauf wird durch einen Schlüssel aus den Feldern Programmname (Datenelement BKK_PROGN), Datum des ersten Starts (BKK_PRGDAT) und laufende Nummer pro Tag (BKK_PRGNO) eindeutig

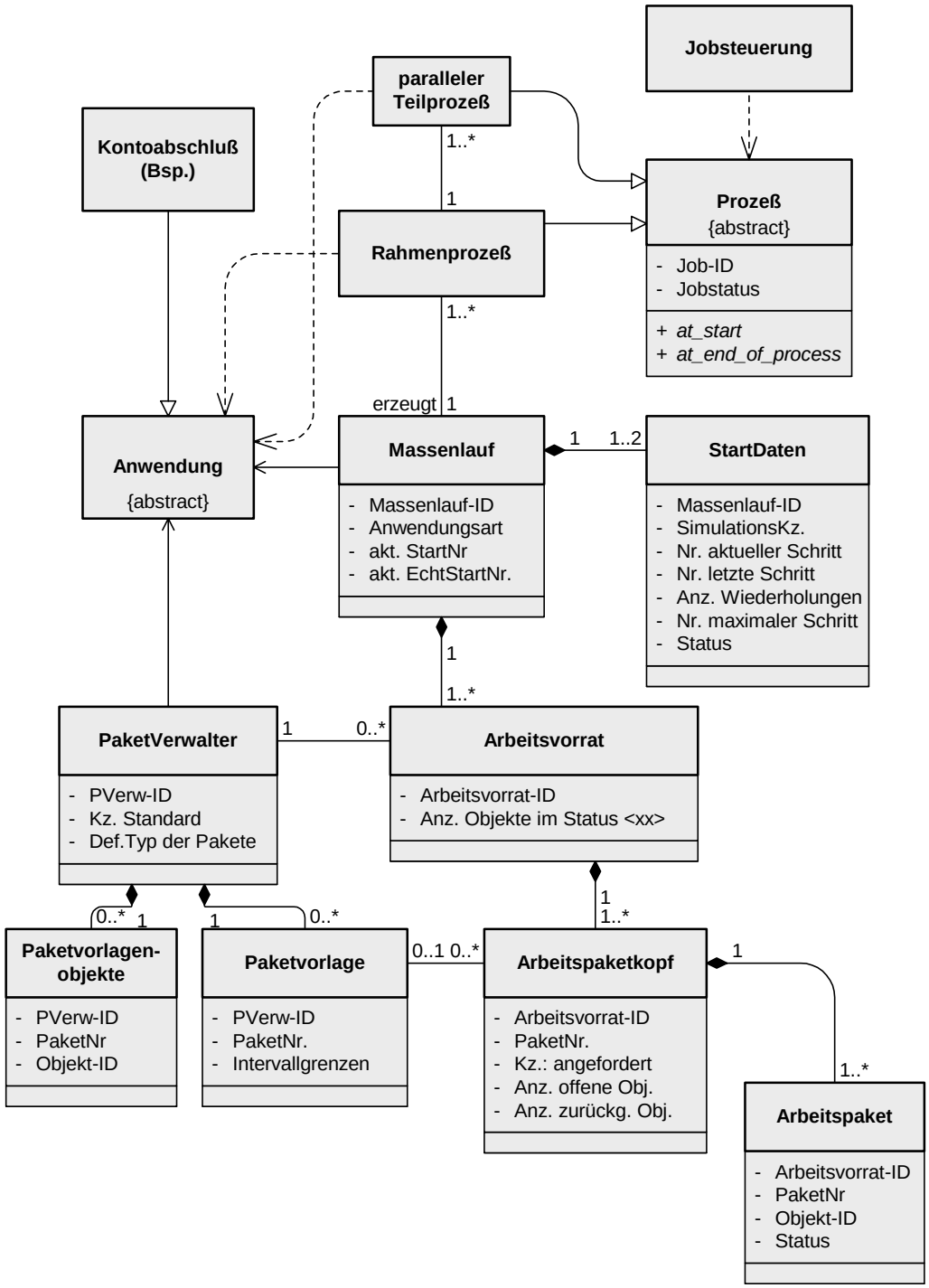
identifiziert. Dem Massenlauf sind als Attribute u.a. die Parameter zugeordnet, mit denen der Lauf gestartet wurde.

Ein Massenlauf kann bzw. muß wiederaufgesetzt werden, falls im Zuge der Verarbeitung Fehler bei einzelnen Objekten aufgetreten sind oder die Verarbeitung insgesamt – z.B. aufgrund eines Zusammenbruchs einer Netzwerkverbindung – abgebrochen wurde.

Die Verarbeitung eines Massenlaufs erfolgt in einem oder mehreren Schritten. Mehrere Schritte sind erforderlich, falls die Objekte in einer bestimmten Reihenfolge verarbeitet werden müssen. Am Beginn der Verarbeitung eines Schrittes wird die gesamte Verarbeitung in mehrere parallele Prozesse aufgespalten, die als Hintergrundjobs ablaufen. Mit der Synchronisation dieser Jobs endet die Verarbeitung des Schrittes und es beginnt – falls erforderlich - die Verarbeitung des nächsten Schritts. Ist der letzte Schritt fertig, endet damit auch der Massenlauf.

2.3 Klassendiagramm

Das folgende Diagramm veranschaulicht noch einmal die Architektur:


Jobsteuerung
paralleler Teilprozeß
Kontoabschluß (Bsp.)
1..*
1
Prozeß
{abstract}
- Job-ID
- Jobstatus
+ at_start
+ at_end_of_process
Rahmenprozeß
1..*
erzeugt 1
Anwendung
{abstract}
Massenlauf
- Massenlauf-ID
- Anwendungsart
- akt. StartNr
- akt. EchtStartNr.
1
1..2
StartDaten
- Massenlauf-ID
- SimulationsKz.
- Nr. aktueller Schritt
- Nr. letzte Schritt
- Anz. Wiederholungen
- Nr. maximaler Schritt
- Status
1
1..*
PaketVerwalter
- PVerw-ID
- Kz. Standard
- Def.Typ der Pakete
1
0..*
Arbeitsvorrat
- Arbeitsvorrat-ID
- Anz. Objekte im Status <xx>
0..* 1
1 0..*
1
1..*
Paketvorlagen-objekte
- PVerw-ID
- PaketNr
- Objekt-ID
Paketvorlage
- PVerw-ID
- PaketNr.
- Intervallgrenzen
0..1 0..*
Arbeitspaketkopf
- Arbeitsvorrat-ID
- PaketNr.
- Kz.: angefordert
- Anz. offene Obj.
- Anz. zurückg. Obj.
1
1..*
Arbeitspaket
- Arbeitsvorrat-ID
- PaketNr
- Objekt-ID
- Status

3 Ablauf der Parallelverarbeitung

3.1 Vorbereitung und Start

Der Start der Parallelverarbeitung erfolgt durch den Aufruf der MAPI-Funktionsbausteine

- **BANK_MAP_PP_START** beim Starten eines neuen Massenlaufs
- **BANK_MAP_PP_RESTART** beim Wiederaufsetzen auf einer abgebrochenen oder fehlerhaften Verarbeitung

Wird der aktuelle Schritt des Massenlaufs erstmals durchlaufen, wird hierfür ein neuer Arbeitsvorrat erzeugt. Sofern beim Start kein Standard-Paketverwalter als Parameter mitgegeben wurde, wird zu dem Arbeitsvorrat auch ein Paketverwalter erzeugt. Desweiteren werden Steuerungsdaten in den Verwaltungstabellen zur Parallelverarbeitung initialisiert bzw. aktualisiert.

Nach diesen vorbereitenden Schritten werden die parallelen Jobs generiert und sofort gestartet. Generierung, Start und Synchronisation der Jobs werden durch die Financial-Services-Jobsteuerung übernommen. Die Verteilung der Hintergrundjobs auf Server oder Servergruppen kann durch den Anwender im Customizing pro Anwendungsart eingestellt werden .

3.2 Parallele Teilprozesse

Innerhalb jedes Jobs wird eine Schleife durchlaufen, in der pro Durchlauf ein noch unbearbeitetes Arbeitspaket vom Arbeitsvorrat angefordert und danach durch die Anwendung bearbeitet wird. Die Schleife wird verlassen, sofern kein unbearbeitetes Paket mehr vorhanden ist. Der Job endet damit.

3.3 Synchronisation

Nach der Synchronisation der parallelen Jobs durch die Jobsteuerung, wird geprüft, ob der Massenlauf weitere Verarbeitungsschritte umfasst. Falls ja, werden diese analog zum ersten Schritt bearbeitet. Falls nein, ist das Ende des Massenlaufs erreicht. Die Verwaltungsdaten des Massenlaufobjekts sowie der abhängigen Objekte werden gelöscht.

3.4 Objektsperren

Objekte, die zu einem Arbeitspaket gehören, das aktuell in einem parallelen Job bearbeitet wird, werden durch das PV-Tool nicht gesperrt. Soll dies geschehen, muß die Anwendung über das Absetzen geeigneter SAP-Sperren für eine Änderungsschutz der aktuellen Objekte sorgen. Beispielsweise stehen für Vertragskonten in der Komponente FS-AM (Financial Services Account Management) die folgenden vier Funktionsbausteine zur Verfügung:

- BCA_API_LOCK_ACCTBAL_ENQ_SING
  versucht, eine einzelne Kontostandssperre abzusetzen.
- BCA_API_LOCK_ACCTBAL_ENQ_FLSH
  versucht, für eine Liste von Verträgen Kontostandssperren abzusetzen.
- BCA_API_LOCK_CONTRACT_ENQ_SING
  versucht, eine einzelne Stammdatensperre abzusetzen
- BCA_API_LOCK_CONTRACT_ENQ_FLSH
  versucht, für eine Liste von Verträgen Stammdatensperren abzusetzen.

Kommt es bei der Verarbeitung von Objekten durch die Anwendung zu Fehlersituationen, die ein vorläufiges Verarbeitungsende bedingen, müssen diese Objekte persistent gesperrt werden. Hierzu stellt das PV-Tool in seiner MAPI-Schnittstelle vier Funktionsbausteine zur Verfügung, mit denen derartige Objektsperren gesetzt, gelesen und gelöscht werden können. Wichtig: Der Aufruf dieser Funktionsbausteine liegt in der Verantwortung der Anwendung. Sie muß entscheiden, welches Objekt wann gesperrt oder entsperrt werden muß. Die Funktionsbausteine sind:

- BANK_MAP_PP_LOCKS_SET
  Setzen von Objektsperren
- BANK_MAP_PP_LOCKS_GET_MLT
  Lesen von Sperren zu einer Objektliste
- BANK_MAP_PP_LOCKS_GET_RUN_MLT
  liefert alle gesperrten Objekte eines Massenlaufs
- BANK_MAP_PP_LOCKS_RELEASE
  Aufheben von Objektsperren

3.5 Objektstatusänderungen und LUWs

Wie bereits erwähnt, verwaltet das PV-Tool den Status der einzelnen Objekte, sofern diese dem PV-Tool bekannt gemacht wurden, und bietet dadurch eine Möglichkeit zum Wiederaufsetzen. Um Dateninkonsistenzen zu vermeiden, müssen LUWs (= Logical Unit of Work) so gebildet werden, dass die Statusänderungen der Objekte immer in dergleichen LUW erfolgen wie die Änderungen in den Anwendungsdaten zu diesen Objekten. Setzt die Anwendungen keine COMMIT-WORK-Befehle ab, wird die Konsistenz zwischen Zustand der Anwendungsdaten und Objektstatus durch das PV-Tool gewährleistet. Dies wird empfohlen. Ist für eine Anwendung das Absetzen von COMMIT-WORK-Befehlen unumgänglich, muß zuvor für die geänderten Objekte der MAPI-Funktionsbaustein BANK_MAP_PP_CONFIRM_OBJECTS aufgerufen werden. Ihm wird in der Tabelle I_TAB_STATUS_CHANGE die Liste der in ihrem Status zu ändernden Objekte übergeben. Der Objektstatus kann folgende Werte annehmen, für die in der Include-Datei BANK_WORKLIST_CONSTANTS Konstanten definiert sind:

Wert	*Konstante*	*Bedeutung*
0	con_procstat_selected	Initial
1	con_procstat_restart	Objekt ist zum Wiederaufsetzen vorgemerkt
2	con_procstat_async	Objekt befindet sich in asynchroner Bearbeitung
3	con_procstat_finished	Objekt wurde erfolgreich bearbeitet
4	con_procstat_retry	Objekt ist zurückgestellt und wird nochmals bearbeitet. Der Anstoß dieser Bearbeitung erfolgt automatisch durch die Steuerung der Parallelisierung.
5	con_procstat_skipped	Objekt wurde übergangen und aus dem Arbeitsvorrat entfernt.

Außer dem Initialstatus „0“ werden die übrigen Statuswerte durch die Anwendung gesetzt. Beim Wiederaufsetzen wird der Status für alle durch die Steuerung des PV-Tool wieder auf den Initialwert gesetzt.

Bezüglich der Persistenz eines Status werden drei Kategorien unterschieden:

1. nicht persistent (***np***): In dieser Kategorie werden Objekte nicht persistent gespeichert bzw. aus der DB entfernt, falls sie bereits in der DB gespeichert waren.
2. optional persistent (***op***): In dieser Kategorie werden Objekte nicht neu in die DB aufgenommen. Sind sie bei einer Statusänderung in den neuen – optional persistenten Status – bereits vorhanden, wird der Status in der DB aktualisiert.
3. persistent (***p***): In dieser Kategorie werden Objekte in jedem Fall in der DB gespeichert.

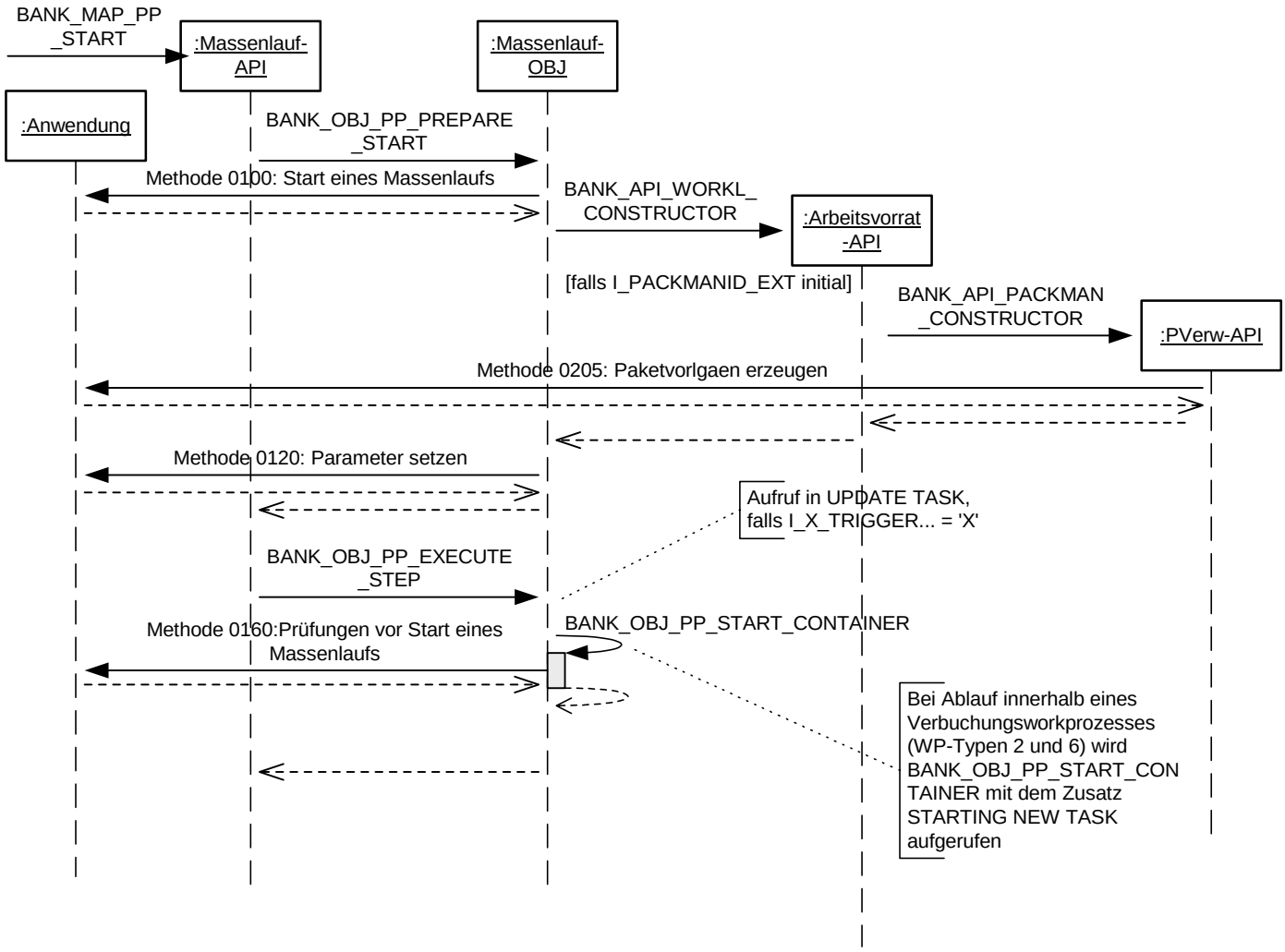
Bei einem Statuswechsel ergeben sich somit je nach Persistenzkategorie des neuen Status und Vorhandensein des Objekts in der DB folgende Konstellationen bzgl. einer DB-Operation (Update / Delete / Insert):

Persistenzkategorie des neuen Status	Objekt bereits in DB	DB-Operation
np	ja	DELETE
np	nein	-
op	ja	UPDATE
op	nein	-
p	ja	UPDATE
p	nein	INSERT

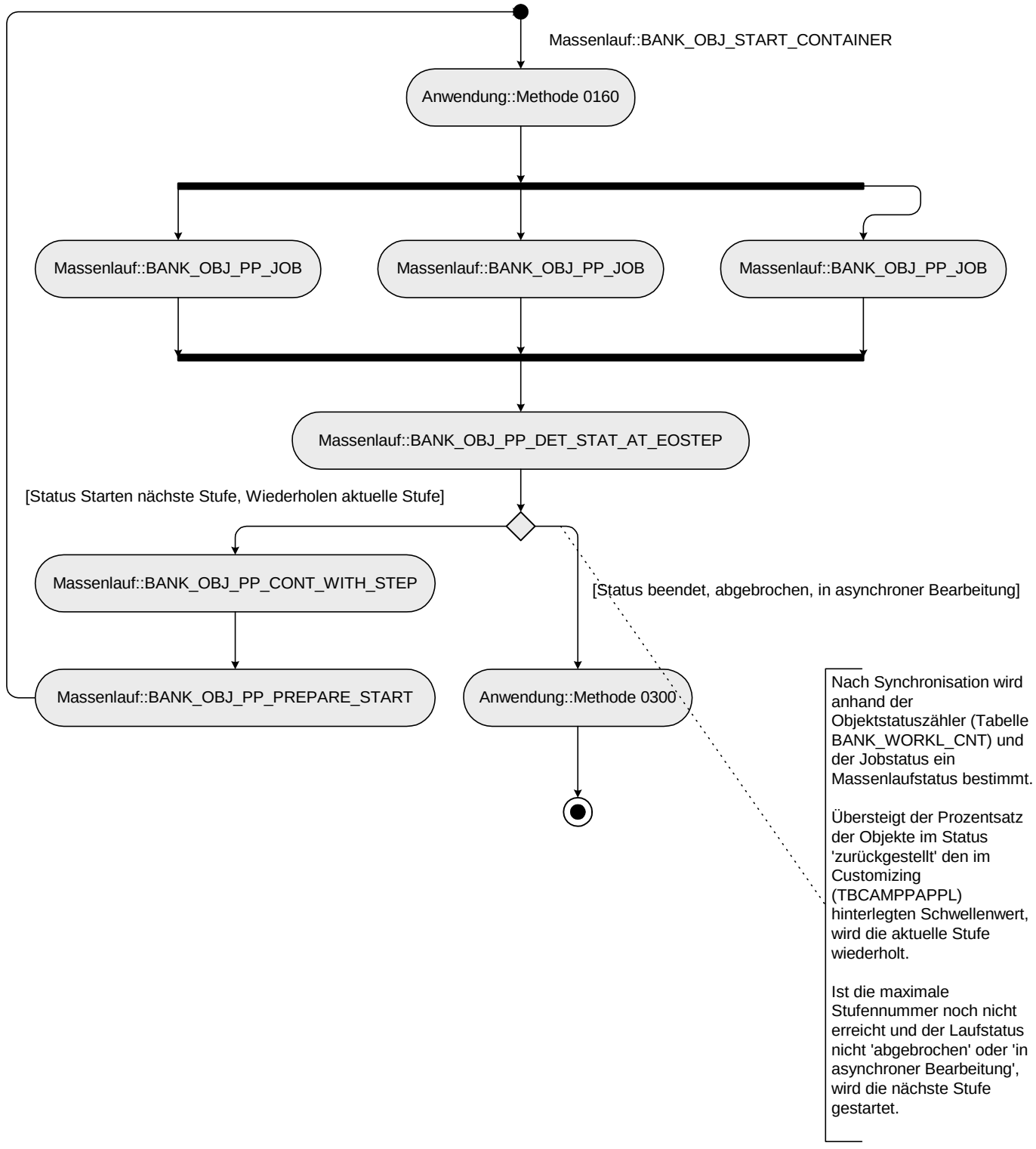
Bis auf den Status „initial“ sind alle Statuswerte fest einer Persistenzkategorie zugeordnet. Beim Status „initial“ entscheidet ein Kennzeichen, das im Kundencustomizing (siehe 5.3.4.1) gesetzt wird, darüber, ob der Status zur Persistenzkategorie „optional persistent“ (Voreinstellung) oder zur Kategorie „persistent“ gehört.

3.6 Diagramme

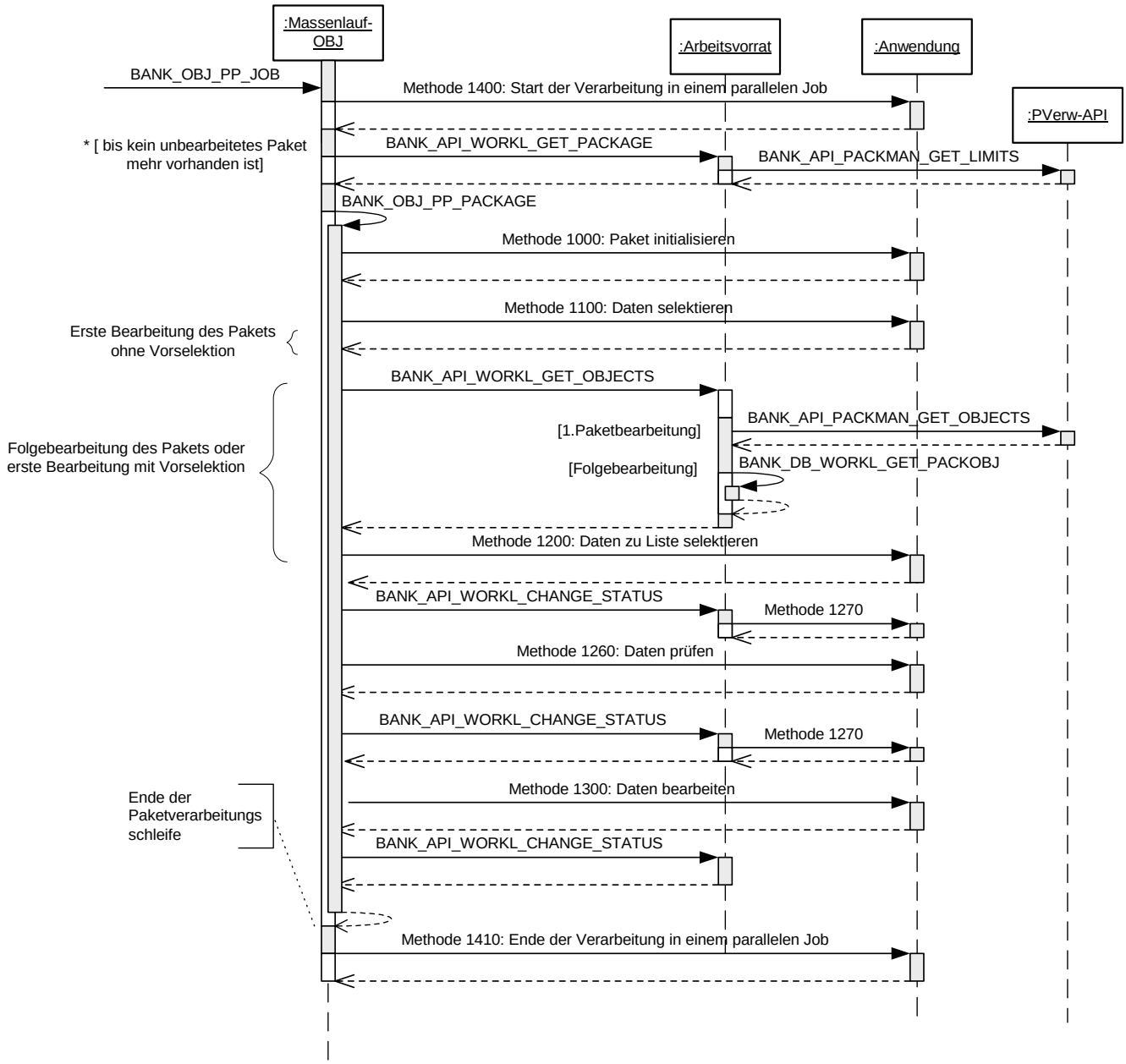
Die folgenden Diagramme geben einen Überblick über den Ablauf der Parallelverarbeitung und zeigen den Kontext, aus dem heraus die in Kapitel 0 detailliert beschriebenen Anwendungsmethoden aufgerufen werden.



Das nächste Diagramm zeigt den Ablauf innerhalb des Funktionsbausteins BANK_OBJ_PP_START_CONTAINER mit der Aufspaltung der Verarbeitung in parallele Jobs:



Das folgende Diagramm ist die Detailsicht auf die Abläufe in einem der parallelen Jobs:



4 Methoden der PV-Schnittstelle

Im Folgenden werden nun die einzelnen Methoden der Schnittstelle zur PV beschrieben, für die die zu parallelisierenden Anwendungen Funktionsbausteine zur Verfügung stellen müssen. Schnittstellen, die nicht zwingend implementiert werden müssen, sind durch den Zusatz „optional“ gekennzeichnet.

4.1 Methode 0100: Start eines Massenlaufs (optional)

Dieser Funktionsbaustein wird beim Beginn eines Massenlaufs aufgerufen um ggf. Änderungen von Parametern beim Wiederaufsetzen zu steuern oder alte, abgebrochen Läufe der Anwendung zu löschen, falls sie nicht mehr benötigt werden.

Parameter	Typ	Bedeutung	Import (I) /

			Export (E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_XNEWRUN	XFELD	Kennzeichen: Neuer Lauf	I
I_STR_CURRPARAM		aktuelle Anwendungsparameter	I
I_STR_OLDPARAM		Anwendungsparameter beim letzten Start	I
E_STR_NEWPARAM		neue Anwendungsparameter	E

Im Parameter I_APPLCATG wird die Anwendungsart übergeben.

Das Kennzeichen I_XNEWRUN signalisiert, daß der zu startende Lauf neu ist. In diesem Fall ist der alte Parametersatz I_STR_OLDPARAM ohne Bedeutung. Der aktuelle Parametersatz I_STR_CURRPARAM wird direkt in den Exportparameter E_STR_NEWPARAM kopiert.

Ist das Kennzeichen I_XNEWRUN nicht gesetzt, handelt es sich um ein Wiederaufsetzen. Der alte Parametersatz I_STR_OLDPARAM und der aktuelle Satz I_STR_CURRPARAM sind in diesem Fall miteinander zu vergleichen. Nicht änderbare Daten müssen in der Exportstruktur E_STR_NEWPARAM mit den Originalwerten aus I_STR_OLDPARAM, änderbare mit den Werten aus I_STR_CURRPARAM gefüllt werden.

Beispiel:

Eine Anwendung hat folgende Parameterstruktur:

- *OrgEinheit*
- *Stichtagsdatum*
- *Protokollmodus*

Beim ersten Start des Massenlaufs wurde der Lauf gestartet mit dem Satz I_STR_OLDPARAM:

- *OrgEinheit = 4711*
- *Stichtagsdatum = 11.11.2000*
- *Protokollmodus = 1*

Beim Wiederaufsetzen wurde im Selektionsbild der Parametersatz I_STR_CURRPARAM übergeben:

- *OrgEinheit = 4712*
- *Stichtagsdatum = 31.12.2000*
- *Protokollmodus = 2*

Die ersten beiden Parameter sind selektionskritisch und dürfen somit beim erneuten Start eines Massenlaufs nicht mehr verändert werden. Der neue Parametersatz E_STR_NEWPARAM ist somit

- *OrgEinheit = 4711*
- *Stichtagsdatum = 11.11.2000*
- *Protokollmodus = 2*

4.2 Methode 0110: Parameter eines Laufs holen (optional)

Dieser Baustein wird aufgerufen, um die Parameter eines Massenlaufs unter dem Schlüssel des Massenlaufs (I_PROGN, I_PROGDATE, I_PROGNO) zu lesen. Die Parameter werden in der Struktur E_S_PARAM zurückgeliefert

Da die Verarbeitung der Arbeitspackete in parallelen Hintergrundjobs erfolgt, müssen die globalen Anwendungsparameter, die beim Start der Massenverarbeitung mitgegeben wurden, persistent gespeichert werden. Andernfalls könnte aus den Batch-Prozessen heraus nicht auf sie zugegriffen werden.

Durch Implementierung der Methoden 0110, 0120 und 0130 kann die Anwendung ihre spezifischen Parameter in eigenen Datenbanktabellen sichern, aus diesen lesen und löschen. Als Schlüssel für den Datenbankzugriff muß dabei der Massenlaufschlüssel – bestehend aus den drei Feldern I_PROGN, I_PROGDATE und I_PROGNO – verwendet werden.

Die Methoden 0110, 0120 und 0130 sind nur insgesamt optional, d.h. sobald eine Methode implementiert wurde, müssen auch die anderen zur Verfügung stehen. Sind die Methoden 0110, 0120 und 0130 nicht implementiert, werden die Anwendungsparameter durch das Parallelisierungstool generisch gespeichert (siehe 5.2).

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I
I_PROGN	BKK_PROGN	1. Schlüsselfeld des Massenlaufs	I
I_PROGDATE	BKK_PRGDAT	2. Schlüsselfeld des Massenlaufs	I
I_PROGNO	BKK_PRGNO	3. Schlüsselfeld des Massenlaufs	I
E_S_PARAM		Struktur der Anwendungsparameter	E

4.3 Methode 0120: Parameter setzen (optional)

Dieser Baustein wird aufgerufen, um die Parameter eines Massenlaufs unter dem Schlüssel des Massenlaufs (I_PROGN, I_PROGDATE, I_PROGNO) auf der Datenbank zu sichern. Die Parameter werden in der Struktur I_S_PARAM übergeben.

Der Baustein muß als Verbuchungsbaustein implementiert sein.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I

I_PROGN	BKK_PROGN	1. Schlüsselfeld des Massenlaufs	I
I_PROGDATE	BKK_PRGDAT	2. Schlüsselfeld des Massenlaufs	I
I_PROGNO	BKK_PRGNO	3. Schlüsselfeld des Massenlaufs	I
I_S_PARAM		Anwendungsparameter	I

4.4 Methode 0130: Parameter löschen (optional)

Dieser Baustein wird aufgerufen, um die Parameter eines Massenlaufs unter dem Schlüssel des Massenlaufs (I_PROGN, I_PROGDATE, I_PROGNO) aus der Datenbank zu löschen.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I
I_PROGN	BKK_PROGN	1. Schlüsselfeld des Massenlaufs	I
I_PROGDATE	BKK_PRGDAT	2. Schlüsselfeld des Massenlaufs	I
I_PROGNO	BKK_PRGNO	3. Schlüsselfeld des Massenlaufs	I
E_RC	SY-SUBRC	Rückgabecode	E

4.5 Methode 0150: Massenlauf für Restart auswählen/suchen (optional)

Dieser Baustein wird beim Start des Wiederaufsetzens aufgerufen, falls dem MAPI-Baustein BANK_MAP_PP_RESTART kein vollständiger Schlüssel des wiederaufzusetzenden Massenlaufs übergeben wurde.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I
I_S_PARAM	anwendungsspezifisch	Anwendungsparameter	I
E_S_RUNKEY	BANK_STR_RUNKEY	Massenlauf-Schlüssel	E
NOT_FOUND			A
NOT_UNIQUE			A
NOT_SELECTED			A

Die Importparameter sind:

I_APPLCATG = Anwendungsart des wiederaufzusetzenden Laufs

I_S_PARAM = anwendungsspezifischer Parametersatz, der beim Start des Wiederaufsetzens mitgegeben wurde.

Durch Aufruf der MAPI-Funktion BANK_MAP_PP_GET_REST_RUNS kann eine Liste der aktuellen Massenläufe zur Anwendungsart I_APPLCATG geholt werden, die potentiell wiederaufgesetzt werden können.

Der Parametersatz I_S_PARAM kann dann dazu verwendet werden, um diese Liste nach anwendungsspezifischen Kriterien weiter zu filtern - z.B., um nur solche Läufe auszuwählen, die diegleichen Organisationseinheiten bearbeiteten, die auch beim Start des Wiederaufsetzens angegeben wurden. Die Anwendungsparameter zu den Massenläufen der selektierten Liste können durch Aufruf der MAPI-Funktion BANK_MAP_PP_GET_APPL_PARAM gelesen werden.

Wurde kein passender Lauf gefunden, muß die Ausnahme NOT_FOUND ausgelöst werden.

Wurden mehrere Läufe gefunden, hängt die Vorgehensweise davon ab, ob der Prozeß im Hintergrund läuft oder nicht:

- Hintergrund (SY-BATCH='X'): Die Ausnahme NOT_UNIQUE wird ausgelöst
- Dialog: Dem Anwender wird die Liste der wiederaufsetzbaren Massenläufe in einem Auswahl-Popup angeboten. Bricht der Anwender die Auswahl ab, wird die Ausnahme NOT_SELECTED ausgelöst.

Bei erfolgreicher Auswahl eines Massenlaufs wird der Schlüssel dieses Laufs im Parameter E_S_RUNKEY zurückgegeben.

4.6 Methode 0160: Prüfungen vor Start eines Massenlaufs (optional)

Der Baustein wird aufgerufen, um zu prüfen, ob der Massenlauf mit dem in den Parametern I_PROGN, I_PROGDATE und I_PROGNO übergebenen Schlüssel im aktuellen Umfeld gestartet werden darf. Zur Prüfung des Umfelds werden in der Tabelle I_TAB_CURR_RUNS die Daten der übrigen aktuellen Massenläufe übergeben. Die Daten enthalten Informationen über den Status der Läufe sowie über ihre Anwendungsart.

In diesem Baustein kann nun gemäß der Anwendungslogik geprüft werden, ob ein Start erlaubt ist. Das Ergebnis dieser Prüfung wird in dem Parameter E_XSTARTALLOWED zurückgegeben.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I
I_PROGN	BKK_PROGN	1. Schlüsselfeld des Massenlaufs	I
I_PROGDATE	BKK_PRGDAT	2. Schlüsselfeld des Massenlaufs	I
I_PROGNO	BKK_PRGNO	3. Schlüsselfeld des Massenlaufs	I
I_XSIMULRUN	XFELD	Kennzeichen Simulationslauf	I
I_TAB_CURR_RUNS	BANK_TAB_PP_RUNDETAILS	Aktuelle Massenläufe	I
E_XSTARTALLOWED	XFELD	Kennzeichen: Start ist erlaubt	E

4.7 Methode 0205: Paketvorlagen erzeugen

Dieser Baustein wird im Rahmen der Erzeugung einer Liste von Paketvorlagen innerhalb einer Schleife aufgerufen. Bei jedem Durchlauf werden die Daten der nächsten Paketvorlage angefordert. **Es ist Aufgabe der Anwendung, die Erzeugung neuer Paketvorlagen zu beenden. Hierzu muß dieser Baustein die Ausnahme NOT_FOUND auslösen.**

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_STR_PARAM		globale Anwendungsparameter	I
I_STR_PACKAGE_KEY	BANK_STR_PP_PACKAGEKEY	Schlüssel der Paketvorlage	I
E_LIMIT_HIGH	BKK_OBJNO	Obergrenze Paket	E
E_LIMIT_LOW	BKK_OBJNO	Untergrenze Paket	E
E_TAB_OBJKEY	BANK_TAB_OBJKEY	Objektliste	E
E_CNT_PACKAGES	BANK_DTE_CNT_PACKAGES	Anzahl der Pakete	E
E_STR_PACKPARAM		Anwendungsparameter zum Paket	E
NOT_FOUND			A

Eine Paketvorlage enthält die statischen Daten zu einem Paket. Die dynamischen Daten wie die Anzahl der Bearbeitungen eines Pakets oder die Anzahl der noch nicht abschließend bearbeiteten Objekte sind in den Arbeitspaketen des Arbeitsvorrats enthalten, die zur Laufzeit aus der Paketvorlage für einen Arbeitsvorrat generiert werden.

Zu den Parametern im einzelnen:

- I_APPLCATG: Anwendungsart
- I_STR_PARAM: Anwendungsparameter
- I_STR_PACKAGE_KEY: Schlüssel der Paketvorlage
  Falls die Anwendung neben den globalen Parametern I_STR_PARAM zusätzliche Parameter pro Paketvorlage benötigt, können diese in der Struktur E_STR_PACKPARAM zurückgegeben werden. In dieser exportierten Struktur muß der Paketvorlagenschlüssel enthalten sein, damit bei der späteren Bearbeitung des Pakets unter diesem Schlüssel auf die Parameter zum Paket zugegriffen werden kann.
- E_LIMIT_LOW, E_LIMIT_HIGH: Intervallgrenzen
  Eine Paketvorlage kann entweder durch die Angabe von Intervallgrenzen oder einer Liste der dem Paket zugeordneten Objekte oder durch die Zahl der vorhandenen Pakete definiert werden. Im letzten Fall werden die konkreten Intervallgrenzen eines Pakets durch eine Rechenfunktion ermittelt. Je nach Alternative werden daher entweder die Parameter E_LIMIT_LOW, E_LIMIT_HIGH oder die Objektliste E_TAB_OBJREL bzw. die Paketanzahl E_CNT_PACKAGES zurückgegeben.
- E_TAB_OBJKEY: Liste der Objekte
- E_CNT_PACKAGES: Anzahl der Pakete
- E_STR_PACKPARAM: anwendungsspezifische Parameter zum Paket. Die Struktur dieses Exportparameters muß dabei exakt der Zeilenstruktur des Dictionary-Tabellentyps entsprechen, der für die paketbezogenen Anwendungsdaten im Customizing hinterlegt ist.

- NOT_FOUND: keine weiteren Paketvorlagen

4.8 Methode 0206: Parameter zu Paketen sichern (optional)

Der Baustein wird bei jedem Sichern von Paketvorlagen im Rahmen der Erzeugung eines Paketverwalters aufgerufen. Zweck des Bausteins ist die Sicherung von Anwendungsdaten zu den einzelnen Paketvorlagen.

Der Baustein ist dann zwingend, falls beim Erzeugen der Pakete (Methode 0205) der Parameter E_STR_PACKPARAM gefüllt wurde.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_TAB_PACKPARAMS		Anwendungsparameter zu Paketen	I
I_X_UPDATE	XFELD	Kennzeichen: DB-Änderungen im Verbucher	I

Die den Paketen zugeordneten Anwendungsdaten müssen zuvor bei der Erzeugung der Paketvorlagen als Exportparameter an den Paketverwalter zurückgegeben worden sein, der daraufhin diese Parameter in eine interne Tabelle übernommen hat. In diesem Baustein wird der Inhalt dieser internen Tabelle in eine oder mehrere Datenbanktabellen der paketbezogenen Anwendungsdaten geschrieben, die von der Anwendung bereitgestellt werden.

Parameter:

- I_TAB_PACKPARAMS

  Tabelle der anwendungsspezifischen Parameter zu Paketen. Diese Tabelle hat den Dictionary-Tabellentyp, der im Customizing der Anwendungsart angegeben ist.

- I_X_UPDATE

  Falls das Kennzeichen gesetzt ist, müssen sämtliche Änderungsoperationen auf der Datenbank als Verbuchungsprozesse ausgefü hrt - d.h. die Funktionsbausteine für die DB-Zugriffe mit dem Zusatz `IN UPDATE TASK` gerufen werden.

4.9 Methode 0207: Parameter zu Paketen löschen (optional)

Der Baustein wird während des Löschens eines Paketverwalters aufgerufen, um die paketbezogenen Anwendungsparameter, die in einer oder mehreren Datenbanktabellen zu einem Paket gespeichert sind, aus der Datenbank zu entfernen.

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_PACKMANID	BANK_PACKMANID	ID des Paketverwalters	I

Parameter

- I_PACKMANID

  ID des Paketverwalters. Alle Einträge, bei denen diese ID im Schlüssel erscheint, sind zu löschen.

4.10 Methode 0300: Am Ende des Massenlaufs (optional)

In diesem Baustein wird die anwendungsspezifische Verarbeitung am Ende eines Massenlaufs durchlaufen. Typische Verarbeitungsschritte sind das Löschen anwendungsspezifischer Daten oder das Auslösen von Ereignissen für die Jobsteuerung.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart in der Parallelisierung	I
I_PROGN	BKK_PROGN	Name des Programms/Report zu einem Anwendungsprozeß	I
I_PROGDATE	BKK_PRGDAT	Datum des Programmlaufs	I
I_PROGNO	BKK_PRGNO	laufende Nummer des Programmlaufs	I
I_RUNSTARTNO	BKK_RUNSTARTNO	Anzahl der Starts eines Massenlaufs	I
I_RUNSTATUS	BANK_DTE_PP_RUNSTATUS	Status des Massenlaufs	I
I_XSIMULRUN	XFELD	Kennz.: Simulationslauf	I
I_PROCESS_ID	BANK_DTE_JC_PROC_ID	ID des Prozesses in einem Jobnetz, in dem die Massenverarbeitung läuft	I
E_APPL_RETURNCODE	SY-SUBRC	Rückgabecode der Anwendung	E

Der Schlüssel der Laufs wird in den Parametern I_PROGN, I_PROGDATE und I_PROGNO übergeben.

Der Parameter I_RUNSTATUS enthält den Status des Massenlaufs. Zu den möglichen Wertausprägungen existieren folgende Konstanten im Include BANK_PPCTRL_CONSTANTS

CON_RUN_UNDEFINED nicht definiert

CON_PROCESSING aktiv in Bearbeitung einer parallelen Stufe

CON_PROCESS_ASYNC es befinden sich noch Objekte in asynchroner Verarbeitung

CON_RUN_ERROR der Lauf ist abgebrochen

CON_RUN_CONTINUE der Lauf ist derzeit aktiv, es sind noch weitere Stufen zu starten

CON_RUN_PPROCESS Einzelfehler sind aufgetreten

CON_RUN_FINISHED Der Lauf ist fehlerfrei beendet worden

CON_RUN_RETRY Der Lauf ist aktiv. Wegen zurückgestellter Objekte wird die Verarbeitung der aktuellen Stufe erneut gestartet.

Der Parameter I_PROCESS_ID enthält die ID des Prozesses, in dem der Massenlauf im Rahmen eines Jobnetzes ausgeführt wird. Ist der Parameter nicht initial, muß innerhalb dieses Bausteins ein Returncode an die Jobsteuerung gemeldet werden. Hierzu wird I_PROCESS_ID benötigt.

4.11 Methode 1000: Initialisierung eines Arbeitspaketes

Der Baustein wird zu Beginn der Verarbeitung eines Arbeitspakets aufgerufen, um die Anwendung mit allen Parametern zu versehen, die für die folgenden Ablaufschritte - d.h. die Selektion, Prüfung und Verarbeitung von Anwendungsdaten - benötigt werden.

In diesem Baustein sollten außerdem Puffer der Anwendung, die Daten zu den Objekten eines Arbeitspaketes enthalten, gelöscht werden.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_PROGN	BKK_PROGN	Name des Programms/Report zu einem Anwendungsprozeß	I
I_PROGDATE	BKK_PRGDAT	Datum des Programmlaufs	I
I_PROGNO	BKK_PRGNO	laufende Nummer des Programmlaufs	I
I_CURRSTARTNO	BKK_RUNSTARTNO	Anzahl der Starts eines Massenlaufs	I
I_XSIMULATION	BKK_XSIMULRUN	Simulationskennzeichen	I
I_CURRSTEPNO	BANK_DTE_PP_STEPNO	aktuelle Stufennummer	I
I_STR_APPLPARAM		globale Anwendungsparameter	I
I_STR_PACKAGE_KEY	BANK_STR_PP_PACKAGEKEY	Schlüssel der Paketvorlage	I
I_LIMIT_LOW	BKK_OBJNO	Untergrenze	I
I_LIMIT_HIGH	BKK_OBJNO	Obergrenze	I
I_XRESTART	XFELD	Kennzeichen: Wiederaufsetzen	I

Im einzelnen werden folgende Parameter übergeben:

- I_APPLCATG: Anwendungsart
- I_PROGN, I_PROGDATE, I_PROGNO: Schlüssel des Massenlaufs
- I_CURRSTARTNO: aktuelle Startnummer des Massenlaufs.
- I_XSIMULATION: Kennzeichen: Lauf wird im Simulationsmodus durchgeführt
- I_CURRSTEPNO: Nummer der aktuellen Verarbeitungsstufe.
  Bei Abhängigkeiten in der Verarbeitung von Objekten (z.B. beim Abschluß von Unterkonten auf Referenzkonten) wird die Verarbeitung in zwei Stufen durchgeführt. Innerhalb einer Stufe läuft die

Verarbeitung parallel. Die Stufennummer wird benötigt, falls die Selektion der Daten abhängig von der Verarbeitungsstufe vorgenommen werden soll. *Beispiel: In Stufe 1 werden aus einer Paketvorlage nur Unterkonten in Stufe 2 nur Referenzkonten selektiert.*

- I_STR_APPLPARAM: anwendungsspezischer Parametersatz.
  Typischerweise enthält diese Struktur Daten, die für die Selektion und/oder Verarbeitung benötigt werden (Fälligkeitsdatum, Ausführungsdatum, etc.)
- I_STR_PACKAGE_KEY: Schlüssel der Paketvorlage
  Mit diesem Schlüssel kann auf anwendungsspezifische Daten zu einer Paketvorlage zugegriffen werden, die bei der Erzeugung von Paketvorlagen angelegt wurden. *Beispiel: Ein Paket ist einer Organisationseinheit zugeordnet.*
- I_LIMIT_LOW: Untere Grenze des Intervalls, aus dem Objekte selektiert werden sollen
- I_LIMIT_HIGH: Obere Grenze des Intervalls, aus dem Objekte selektiert werden sollen
- I_XRESTART: Das Kennzeichen signalisiert, daß das aktuell zu bearbeitende Paket schon einmal bearbeitet worden ist. Ggf. sind also bereits Daten zu den Objekten in Anwendungstabellen in der Datenbank gespeichert.

4.12 Methode 1100: Selektion von Anwendungsdaten

Der Baustein liest die zu einem Arbeitspaket gehörenden Daten.

Name	Type	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
E_TAB_OBJKEY	BANK_TAB_OBJKEY	selektierte Objekte	E
NOT_FOUND		keine Objekte gefunden	A

Sämtliche Selektionskriterien wurden bei der Initialisierung des Arbeitspakets übergeben und stehen im Puffer zur Verfügung.

Die Ergebnisse der Selektion werden in der Tabelle E_TAB_OBJKEY zurückgegeben. Hierzu müssen die selektierten Objekte aus dem Anwendungsformat in das für die Parallelverarbeitung gültige Format umgewandelt werden. Zur Konvertierung steht der MAPI-Baustein BANK_API_PPOBJ_CONV zur Verfügung.

4.13 Methode 1200: Nachselektion von Anwendungsdaten

Dieser Funktionsbaustein wird durch die Steuerung der Parallelverarbeitung aufgerufen, um zu einer gegebenen Liste von Objekten die Anwendungsdaten (für die spätere Verarbeitung) nachzulesen. Ein solcher Aufruf kann im Rahmen des Wiederaufsetzens auf einem mit Fehlern beendeten Massenlauf erfolgen. Ebenfalls durchlaufen wird der Baustein, falls die zu verarbeitenden Objekte aufgrund einer Vorselektion bereits bekannt sind.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_TAB_OBJKEY	BANK_TAB_OBJKEY	wiederaufzusetzende Objekte	I
E_TAB_OBJKEY_NOT_VALID	BANK_TAB_OBJKEY	ungültige Objekte	E

E_TAB_OBJKEY_NEW	BANK_TAB_OBJKEY	neue Objekte	E

Die Objektliste wird in der Tabelle I_TAB_OBJKEY übergeben. Mit Hilfe des MAPI-Funktionsbausteins BANK_API_PPOBJ_CONV wird diese Liste in das Anwendungsformat umgewandelt.

Beim Wiederaufsetzen kann es vorkommen, daß einige der beim ersten Start selektierten Objekte, deren Bearbeitung jetzt abgeschlossen werden soll, aus bestimmten Gründen mittlerweile nicht mehr gültig sind und nicht mehr bearbeitet werden können oder dürfen. Diese ungültigen Objekte müssen in der Tabelle E_TAB_OBJKEY_NOT_VALID an die Steuerung der PV zurückgegeben werden.

Ebenso ist es möglich, daß sich die Liste der beim Wiederaufsetzen zu bearbeitenden Objekte vergrößert. Diese neu zu bearbeitenden Objekte müssen der PV-Steuerung über den Exportparameter E_TAB_OBJKEY_NEW bekannt gemacht werden.

Die Konvertierung der Anwendungsdaten in die von der PV verwendete Darstellung erfolgt wie überall durch den Baustein BANK_API_PPOBJ_CONV.

4.14 Methode 1260: Prüfung, ob Objekte gesperrt sind (optional)

Im Rahmen der Verarbeitung eines Arbeitspaketes wird dieser Baustein durchlaufen, um die Objekte des Arbeitspaketes aus Anwendungssicht zu prüfen. Von Bedeutung sind hier vor allem Sperren, die von anderen Massenläufen auf die von ihnen bearbeiteten Objekte gesetzt wurden.

Name	Typ	Bedeutung	Import(I) Export(E) Changing (C) Ausnahme (A)
C_TAB_STATUS_CHANGE	BANK_TAB_PP_STATUS_CHANGE		C

In der Tabelle C_TAB_STATUS_CHANGE werden die Objekte zurückgegeben, bei denen sich der Status aufgrund der Prüfungen geändert hat. Beim Aufruf der Methode ist die Tabelle C_TAB_STATUS_CHANGE leer.

Der Bearbeitungsstatus eines Objektes kann folgende Werte annehmen, für die in der Include-Datei BANK_WORKLIST_CONSTANTS Konstanten definiert sind:

Wert	*Konstante*	*Bedeutung*
0	con_procstat_selected	Initial
1	con_procstat_restart	Objekt ist zum Wiederaufsetzen vorgemerkt
2	con_procstat_async	Objekt befindet sich in asynchroner Bearbeitung
3	con_procstat_finished	Objekt wurde erfolgreich bearbeitet
4	con_procstat_retry	Objekt ist zurückgestellt und wird nochmals bearbeitet. Der Anstoß dieser Bearbeitung erfolgt automatisch durch die Steuerung

		der Parallelisierung.
5	con_procstat_skipped	Objekt wurde übergangen und aus dem Arbeitsvorrat entfernt.

Als Ergebnis von Prüfungen kann der Objektstatus die Werte 1,4 und 5 annehmen.

4.15 Methode 1270: Löschen von Objekten aus Anwendungsdaten

Dieser Baustein wird aufgerufen, wenn sich der Bearbeitungsstatus von Objekten dahingehend geändert hat, daß eine momentane Bearbeitung nicht mehr möglich ist.

In diesem Zusammenhang ist ein Abgleich mit den durch die Anwendung gepufferten Daten erforderlich. Die aus dem Anwendungspuffer zu löschenden Objekte werden in der Tabelle I_TAB_DELETE übergeben.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_TAB_DELETE	BANK_TAB_OBJKEY	zu löschende Objekte	I

4.16 Methode 1300: Objekte bearbeiten

Dieser Baustein führt die eigentliche Verarbeitung der zuvor selektierten und validierten Objekte durch. Die zu diesem Zeitpunkt gültigen Objektdaten befinden sich in einem durch die Anwendung verwalteten Puffer.

Der Bearbeitungsstatus der Objekte wird in der Exporttabelle E_TAB_STATUS_CHANGE zurückgegeben.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
E_TAB_STATUS_CHANGE	BANK_TAB_PP_STATUS_CHANGE		E

4.17 Methode 1400: Start der Verarbeitung in einem parallelen Job (optional)

Diese Methode wird zu Beginn der Verarbeitung von Paketen innerhalb eines Jobs angesprochen. Sie kann seitens der Anwendung dazu benutzt werden, um z.B. Anwendungsparameter, die in der Struktur I_STR_APPLPARAM übergeben werden, im globalen Gedächtnis zu speichern oder um ein Anwendungsprotokoll zu erzeugen bzw. das zuvor automatisch generierte Anwendungsprotokoll unter dem Handle I_LOG_HANDLE zu modifizieren.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_PROGN	BKK_PROGN	Massenlaufschlüssel	I
I_PROGDATE	BKK_PRGDAT	Massenlaufschlüssel	I
I_PROGNO	BKK_PRGNO	Massenlaufschlüssel	I
I_STR_APPLPARAM		Globale Anwendungsparameter	I

I_MSG_HANDLE	EMSG_GEN-HANDLE	Handle, unter dem ein beim Jobstart durch die PV-Steuerung automatisch erzeugtes Standard-Nachrichtenobjekt angesprochen werden kann.	I

4.18 Methode 1410: Ende der Verarbeitung in einem parallelen Job (optional)

Die Methode wird aufgerufen, wenn alle Pakete bearbeitet sind. Innerhalb der Methode kann z.B. die Erstellung von Protokollen abgeschlossen werden.

Name	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_PROGN	BKK_PROGN	Massenlaufschlüssel	I
I_PROGDATE	BKK_PRGDAT	Massenlaufschlüssel	I
I_PROGNO	BKK_PRGNO	Massenlaufschlüssel	I

5 Anbindung einer Anwendung an die Parallelverarbeitung

Dieses Kapitel dient als „Kochbuch“, mit dessen Hilfe ein Anwendungsentwickler in die Lage versetzt werden soll, eine Parallelverarbeitung seiner Anwendung zu implementieren.

5.1 Implementierung der PV-Schnittstelle

Die Anwendung muß zunächst die in Abschnitt 4 beschriebenen Schnittstellenmethoden durch entsprechende Funktionsbausteine implementieren. Um innerhalb der FB auf gepufferte Daten gemeinsam zugreifen zu können, empfiehlt es sich, die Funktionsbausteine in einer eigenen Funktionsgruppe zusammenzufassen.

5.2 Persistente Speicherung der Anwendungsparameter

Da die eigentliche Verarbeitung in parallelen Hintergrundjobs erfolgt, müssen die globalen Anwendungsparameter, die beim Start der PV übergeben wurden, persistent gespeichert werden, um innerhalb der Jobs verfügbar zu sein. Die Anwendung kann dazu entweder den Persistenzdienst des PV-Tools nutzen oder eigene Datenbanktabellen zur Speicherung der Parameter anlegen.

Durch das PV-Tool werden die Parameter generisch als Werte mit Beschreibung der Datenstruktur in den Tabellen BANK_PP_PARDESC und BANK_PP_PARVALS gespeichert. Das generische Speichern kann allerdings nicht verwendet werden, falls die Inhalte einzelner Felder in der Parameterstruktur nicht in ein Character-Feld der Länge 45 passen (Datenelement RSDSSELOP_). Für die Anwendung entfällt bei generischer Speicherung jeglicher Aufwand, insbesondere eine Implementierung der Methoden 0110, 0120 und 0130. Dieser Vorteil wird allerdings erkauft durch eine relativ speicherplatzintensive Form der Speicherung.

Um eine eigene Datenhaltung der Anwendungsparameter eines Massenlaufs zu realisieren, muß die Anwendung eine oder ggf. mehrere DB-Tabellen nutzen. Der Schlüssel der Datenbanktabellen muß den Schlüssel des Massenlaufs – bestehend aus den drei Datenelementen BKK_PROGN, BKK_PRGDAT und BKK_PRGNO – umfassen. Außerdem ist bei eigenen Tabellen die Implementierung der Methoden 0110, 0120 und 0130 obligatorisch.

5.3 Customizing

Die Transaktion für das Entwickler-Customizing lautet **BANK_CUS_PP**. Man gelangt nach dem Aufruf zu einem Viewcluster, in dem die Einstellungen vorgenommen werden können.

5.3.1 PV-Klasse

Nach Navigation über den Eintrag „zu bearbeitende Klassen“ erscheint eine Pflegeview, in der Sie die PV-Klasse Ihrer Anwendung eintragen. Die PV-Klasse legt fest, was das Objekt der Verarbeitung ist. Beispiele aus dem Bankbereich sind Konten oder Daueraufträge.

Die Objekte werden durch das PV-Tool mit einem internen Schlüssel angesprochen (siehe 2.2.1). Somit muß der außerhalb des PV-Tools verwendete externe Schlüssel in den internen umgewandelt werden. Bei der Definition der PV-Klasse können hierzu die Namen der Konvertierungsbausteine für die Konvertierung von extern nach intern bzw. intern nach extern hinterlegt werden. In diesem Fall kann dann für die Schlüsselkonvertierungen, die von der Anwendung vorgenommen werden müssen, der Funktionsbaustein BANK_API_PPOBJ_CONV verwendet werden.

Die Parameterschnittstelle für die Schlüsselumwandlung hat folgende Struktur:

- Konvertierung intern -> extern

Name	Typ	Optional	Bedeutung	Import (I) / Export (E) / Ausnahme (A)
I_OBJCATG	BKK_OBJCATG		PV-Klasse	I
I_OBJNO	BKK_OBJNO		Objektnummer	I
E_S_EXTOBJECT	anwendungssperzifisch		Struktur für externen Objektschlüssel	E

- Konvertierung extern -> intern

Name	Typ	Optional	Bedeutung	Import (I) / Export (E) / Ausnahme (A)
I_OBJCATG	BKK_OBJCATG		PV-Klasse	I
I_S_EXTOBJECT	anwendungsspezifisch		Struktur für externen Objektschlüssel	I
E_OBJNO	BKK_OBJNO		Objektnummer	E

5.3.2 Anwendungsart

Navigieren Sie zum Eintrag ‚Anwendungsarten’. Wählen Sie im Änderungsmodus ‚Neue Einträge’.

Vergeben Sie einen maximal vierstelligen Schlüssel für Ihre Anwendung. Im Block „Anwendungsarten“ tragen Sie außerdem die Bezeichnung der Anwendungsart sowie die PV-Klasse ein, die durch die Anwendung bearbeitet wird. Soll automatisch ein Anwendungsprotokoll für die Parallelverarbeitung erzeugt

werden können, so nehmen Sie unter „Objekt“ und „Unterobjekt“ Einträge für Anwendungsprotokollobjekt und -unterobjekt vor.

5.3.3 Zuordnung von Funktionsbausteinen zu Methoden der PV-Schnittstelle

Navigieren Sie nun zu „implementierte Methoden“. Ordnen Sie den Methoden die Namen der Anwendungsfunktionsbausteine zu, die die Schnittstellenmethoden implementieren.

5.3.4 Kundencustomizing

Kundenseitig können pro Anwendungsart weitere Einstellungen zur Steuerung der Parallelverarbeitung in der Tabelle TBANK_PP_APPCUST getroffen werden. Die Pflege wird über die Transaktion SM30 aufgerufen.

5.3.4.1 Kennzeichen: alle Objekte persistent

Ist das Kennzeichen „alle Objekte persistent“ gesetzt, wird die Persistenzkategorie (siehe 3.5) des Initialstatus von „optional persistent“ auf „persistent“ geändert. Alle pro Paket selektierten Objekte stehen so im Abbruchfall zur Verfügung. Dieser Vorteil wird aber durch den Nachteil geringerer Performance erkauft, da bei erfolgreicher Verarbeitung die Objekteinträge wieder aus der DB gelöscht werden müssen, woraus häufige Einfüge- und Löschoperationen resultieren. Die Voreinstellung ist daher das ungesetzte Kennzeichen.

Die folgende Tabelle verdeutlicht noch einmal die Auswirkungen des Kennzeichens auf die Persistenzkategorie der verschiedenen Status. Wie erkennbar ändert sich die Persistenzkategorie nur für den Status „initial“ (0).

Status	Persistenzkategorie	
	Kz.“alle Obj.pers.“ = „ “	Kz.“alle Obj.pers.“ = „X“
0 (con_procstat_selected)	op	p
1 (con_procstat_restart)	p	p
2 (con_procstat_async)	p	p
3 (con_procstat_finished)	np	np
4 (con_procstat_retry)	p	p
5 (con_procstat_skipped)	np	np

5.3.4.2 Prozentsatz zurückgestellter Objekte

Mit dem Prozentsatz wird die Toleranzschwelle definiert, oberhalb derer die Steuerung der PV eine automatische Wiederholung der Verarbeitung für die zurückgestellten Objekte unmittelbar nach Synchronisation der parallelen Jobs erneut startet. Die Schwelle ist dabei definiert als prozentuale der Objekte im Status „zurückgestellt“ an den insgesamt noch offenen Objekten – d.h. Objekten im Status „initial“, „zurückgestellt“ oder „nachzubearbeiten“.

5.3.4.3 Anzahl der wiederholten Durchläufe einer Stufe

Mit dieser Anzahl kann bzw. muß die Zahl der automatischen Wiederholungen der PV für zurückgestellte Objekte begrenzt werden, um nicht in eine potentielle Endlosschleife zu geraten.

5.3.4.4 BTE-Anwendung

Die Methode 0205 zur Erzeugung von Paketvorlagen (siehe 4.7) kann durch einen Kunden-Funktionsbaustein überschrieben werden. Hierzu steht als Schnittstelle der Business Transaction Event (BTE) 0BANK010 zur Verfügung. Bei der Zuordnung des Kunden-FB zur Schnittstelle muß eine Anwendung

angegeben werden. Diese Anwendung muß ebenfalls im Kunden-Customizing zur Anwendungsart hinterlegt werden, damit der Kunden-FB bei der PV der Anwendungsart angesprochen wird. Ist hier nichts gepflegt, wird der Standard-FB aufgerufen, der der Methode 0205 zugeordnet wurde.

5.4 Aufruf der Start-Funktionsbausteine

5.4.1 Erststart eines Massenlaufs

Die parallele Massenverarbeitung wird durch Aufruf des FB BANK_MAP_PP_START. Üblicherweise geschieht dies aus einem Report heraus. Der Startbaustein hat folgende Schnittstelle

BANK_MAP_PP_START					
Name	Typ	Bedeutung	Vor-schlags wert	Optio-nal	Import(I) / Export(E) / Ausnahme (A)
I_PROGN	BKK_PROGN	Massenlaufs chlüssel, 1.Feld		X	I
I_PROGDATE	BKK_PRGDAT	Massenlaufs chlüssel, 2.Feld	SY-DATUM	X	I
I_RUNID_EXT	BANK_DTE_PP _RUNID_EXT	externe ID des Massenlaufs		X	I
I_APPLCATG	BKK_PAAPPLC ATG	Anwendungs art			I
I_PACKMAN_ID	BANK_DTE_PP _PMID_EXT	Schlüssel einer zu verwendende n (Standard-) Paketeinteilu ng		X	I
I_XSIMULRUN	BKK_XSIMULR UN	Kennz.: Simulationsla uf	SPACE	X	I
I_MAXSTEPNO	BANK_DTE_PP _STEPNO	Anzahl max. Parallelisieru ngsschritte	‚001'	X	I
I_STR_APPL_PARAM		Anwendungs parameter		X	I
I_STR_PRINT	PRI_PARAMS	Druckparame ter		X	I
I_XLOG	XFELD	Kennz.: Anwendungs protokoll erstellen		X	I
I_LOGEXTNUMBER	BALNREXT	Externe ID zu Anwendungs protokoll		X	I

I_TAB_LOG	BANK_TAB_JC_APPL_LOG_DATA	Daten zum Öffnen von Nachrichtenobjekten in einem Prozeß		X	I
I_X_SYNC	XFELD	Kennz.: synchroner Aufruf	‚X'	X	I
I_X_USE_DIALOG_WP	XFELD	Kennz.: Keine Hintergrundverarbeitung		X	I
I_X_TRIGGER_START_BY_COMMIT	XFELD	Kennz.: PV-Start durch COMMIT WORK auslösen		X	I
E_STR_RUNKEY	BANK_STR_RUNKEY	Schlüssel des Massenlaufs			E
E_RUNSTATUS	BANK_DTE_PP_RUNSTATUS	Status des Massenlaufs			E
E_RCD_APPL	SY-SUBRC	Rückgabecode der Anwendung			E
E_TAB_LOGH	BAL_T_LOGH	Liste der Handles zu erzeugten Anwendungsprotokollen			E
E_LOGEXTNUM	BALNREXT	externe ID zu Anwendungsprotokollen			E
E_TAB_JOBS	BANK_TAB_JC_JOBKEY	Liste der generierten Hintergrundjobs			E
NO_OUT_OF_SYNC					A
NO_EXPORT_ALLOWED					A
PACKMAN_INVALID					A
PREPARE_FAILED					A
START_FAILED					A

Zu den Parametern im einzelnen:

- In den Parameter I_PROGN, I_PROGDATE kann der Aufrufer die ersten beiden Felder des Massenlaufschlüssels vorbelegen. Der Baustein generiert durch Hinzufügen einer laufenden Nummer den vollständigen Schlüssel, der im Exportparameter E_RUNKEY zurückgegeben wird. Ist I_PROGN nicht gefüllt, wird in das Schlüsselfeld PROGN der Wert APPL_<*Anwendungsart*> geschrieben.

- I_APPLCATG: Anwendungsart
- I_PACKMAN_ID: ID einer bereits existierenden Paketeinteilung, die vom neu zu startenden Massenlauf verwendet werden soll. (Siehe auch 5.5).
- I_XSIMULRUN: Der Massenlauf wird als Simulation ausgeführt. Für Läufe, die in diesem Modus neu gestartet werden, gibt es keine Möglichkeit zum Wiederaufsetzen.
- I_MAXSTEPNO: Anzahl paralleler Verarbeitungsschritte (Siehe 2.2.5).
- I_STR_APPLPARAM: In diesem Parameter werden die globalen Anwendungsparameter übergeben.
- I_STR_PRINT: Druckparameter. Werden die Parameter nicht übergeben, gelten die Druckeinstellungen zum Benutzer, der den FB aufruft.
- I_XLOG: Kennzeichen, dass das PV-Tool ein Anwendungsprotokoll (pro parallelen Job) erzeugen soll. In diesem Fall müssen Objekt und ggf. Unterobjekt des Anwendungsprotokolls im Customizing der Anwendungsart angegeben worden sein (siehe 5.3.2) oder in der Tabelle I_TAB_LOG Einstellungen zur Erzeugung von Nachrichtenobjekten übergeben werden.
- I_LOGEXTNUMBER: Hier kann eine externe ID für das vom PV-Tool generierte Anwendungsprotokoll übergeben werden. Der Parameter ist veraltet. Stattdessen sollten die Daten zur Protokollierung in der Tabelle I_TAB_LOG übergeben werden.
- I_TAB_LOG: Liste der zu erzeugenden Nachrichtenobjekte. Die Übergabestruktur wird in Abschnitt 5.6 näher erläutert.
- I_X_SYNC: Ist das Kennzeichen gesetzt, wird der FB synchron durchlaufen, d.h. der FB wird erst nach Ende aller parallelen Jobs verlassen. Erfolgt der Aufruf des Start-FB aus einem Report heraus, der Teil eines Jobnetzes im Bereich IBS Financial Services ist, muß dieses Kennzeichen gesetzt sein.
- I_X_USE_DIALOG_WP: Falls gesetzt, werden keine Hintergrundjobs gestartet. Die Verarbeitung läuft stattdessen sequentiell in einem Dialog-Workprozeß. Dieser Betriebsmodus sollte möglichst nur zum Debuggen genutzt werden.
- I_X_TRIGGER_START_BY_COMMIT: Mit diesem Kennzeichen wird gesteuert, ob der Start der Parallelverarbeitung durch einen expliziten COMMIT WORK ausgelöst werden soll (‚X') oder nicht (SPACE). Der erste Fall ist bei einem Aufruf des FB im Rahmen eines BAPI gegeben. Hier wird die PV nicht durch den BAPI gestartet, stattdessen werden sämtliche Datenbankeinträge zur Steuerung der PV zunächst im Verbucher vorgemerkt. Der letzte Eintrag in der Liste des Verbuchers ist der eigentliche Start der parallelen Jobs. Das Schreiben in die Datenbank und der Jobstart werden dann durch einen Aufruf des FB BAPI_TRANSACTION_COMMIT getriggert.
- E_STR_RUNKEY: Schlüssel, unter dem der neu generierte Massenlauf angesprochen werden kann.
- E_RUNSTATUS, E_RCD_APPL: (technischer) Laufstatus und Rückgabecode der Anwendung nach Ende der parallelen Verarbeitung. Diese Parameter können im asnychronen Fall (I_X_SYNC='X') nicht abgefragt werden (Ausnahme: NO_EXPORT_ALLOWED).

⇒ Ein Beispiel für einen Aufruf der PV ist der Report RBANK_PP_DEMO_START.

5.4.2 Wiederaufsetzen auf einer Massenverarbeitung

Das Wiederaufsetzen auf einem abgebrochenen oder fehlerhaften Massenlauf erfolgt ähnlich wie der Erststart durch Aufruf eines Funktionsbausteins. In diesem Fall ist es der FB BANK_MAP_PP_RESTART mit folgender Schnittstelle:

BANK_MAP_PP_RESTART					
Name	Typ	Bedeutung	Vor-schlags wert	Optio-nal	Import(I) / Export(E) / Ausnahme (A)

I_PROGN	BKK_PROGN	Massenlaufschlüssel, 1.Feld		X	I
I_PROGDATE	BKK_PRGDAT	Massenlaufschlüssel, 2.Feld		X	I
I_PROGNO	BKK_PRGNO	Massenlaufschlüssel, 3.Feld		X	I
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart		X	I
I_PACKMAN_ID	BANK_DTE_PP_PMID_EXT	Schlüssel einer zu verwendenden (Standard-) Paketeinteilung		X	I
I_XSIMULRUN	BKK_XSIMULRUN	Kennz.: Simulationslauf	SPACE	X	I
I_MAXSTEPNO	BANK_DTE_PP_STEPNO	Anzahl max. Parallelisierungsschritte	‚001'	X	I
I_STR_APPL_PARAM		Anwendungsparameter		X	I
I_STR_PRINT	PRI_PARAMS	Druckparameter		X	I
I_XLOG	XFELD	Kennz.: Anwendungsprotokoll erstellen		X	I
I_LOGEXTNUMBER	BALNREXT	externe Protokollnummer		X	I
I_TAB_LOG	BANK_TAB_JC_APPL_LOG_DATA	Daten zum Öffnen von Nachrichtenobjekten in einem Prozeß		X	I
I_X_SYNC	XFELD	Kennz.: synchroner Aufruf	‚X'	X	I
I_X_USE_DIALOG_WP	XFELD	Kennz.: Keine Hintergrundverarbeitung		X	I
I_X_TRIGGER_START_BY_COMMIT	XFELD	Kennz.: PV-Start durch COMMIT WORK		X	I

		auslösen			
E_RUNSTATUS	BANK_DTE_PP_RUNSTATUS	Status des Massenlaufs			E
E_RCD_APPL	SY-SUBRC	Rückgabecode der Anwendung			E
E_TAB_LOGH	BAL_T_LOGH	Liste der Handles zu erzeugten Anwendungsprotokollen			E
E_LOGEXTNUMBER	BALNREXT	externe Protokollnummer			E
E_TAB_JOBS	BANK_TAB_JC_JOBKEY	Liste der generierten Hintergrundjobs			E
NO_OUT_OF_SYNC					A
NO_EXPORT_ALLOWED					A
PACKMAN_INVALID					A
PREPARE_FAILED					A
START_FAILED					A
RUNKEY_NOT_QUALIFIED					A
APPL_NOT_QUALIFIED					A
METHOD_NOT_IMPLEMENTED					A

Zur Bedeutung der einzelnen Parameter wird auf die Schnittstellenbeschreibung des FB BANK_MAP_PP_START verwiesen (5.4.1).

Wird beim Start des Wiederaufsetzens nicht der Schlüssel des wiederaufzusetzenden Massenlaufs in den Feldern I_PROGN, I_PROGDATE und I_PROGNO übergeben, darf die Anwendungsart nicht initial sein. Es muß in diesem Fall außerdem die Schnittstellenmethode 0150 (Massenlauf für Restart auswählen, siehe 4.5) implementiert sein. Andernfalls bricht das Wiederaufsetzen mit der Ausnahme RUNKEY_NOT_QUALIFED ab.

Soll der Massenlaufschlüssel an den FB übergeben werden, kann der Schlüssel z.B. in einem Benutzerdialog ausgewählt werden. Hierzu stellt das PV-Tool den MAPI-Funktionsbaustein BANK_MAP_PP_SELECTION_POPUP zur Verfügung. Die Auswahlliste für diesen Baustein kann durch den FB BANK_MAP_PP_GET_REST_RUNS ermittelt werden.

5.5 Wiederverwendung von Paketeinteilungen

Bei vielen periodischen Massenverarbeitungen, bei denen Objekte in Pakete aufgeteilt und diese Paket dann parallel verarbeitet werden, muß die Paketeinteilung für die periodisch durchzuführenden Massenläufe oftmals nicht verändert werden. In solchen Fällen bietet sich die Verwendung von Standard-Paketeinteilungen an, die beim Start eines Massenlaufs als Schnittstellenparameter übergeben werden (Parameter I_PACKMAN_ID, siehe 5.4.1).

Ein Paketeinteilung wird durch den MAPI-Funktionsbaustein BANK_MAP_PACKMAN_CONSTRUCTOR erzeugt. Die FB-Schnittstelle hat folgende Struktur:

BANK_MAP_PACKMAN_CONSTRUCTOR					
Name	Typ	Bedeutung	Vor-schlags wert	Optio-nal	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart			I
I_TAB_PARAMVALUES	BANK_TAB_DATAVAL	Parameterwerte			I
I_TAB_PARAMDESC	BANK_TAB_DATADESC	Parameterbeschreibung			I
I_PACKMANID	BANK_DTE_PP_PMID_EXT	ID der Paketeinteilung			I
I_PACKMANDESC	BANK_DTE_PP_PMID_EXT_DESC	Kurzbeschreibung der Paketeinteilung		X	I
PACKMANID_IN_USE					A
PACKMANID_NOT_QUALIFIED					A
APPLCATG_NOT_QUALIFIED					A
NO_METHOD					A
INTERNAL_ERROR					A

- I_APPLCATG: Die Anwendungsart wird benötigt, um die implementierte Methode zur Paketbildung aufzurufen (siehe 4.7). Allerdings kann der Kunde durch einen eigenen Funktionsbaustein zum Business Transaction Event 0BANK010 die Standard-Paketbildung gemäß Methode 0205 übersteuern. Hierzu muß im Customizing der Anwendung die BTE-Anwendungskategorie gepflegt sein.
- I_TAB_PARAMVALUES, I_TAB_PARAMDESC: Die Parameterwerte werden generisch als Character-Felder in der Tabelle I_TAB_PARAMVALUES übergeben. Die Metadaten – d.h. die Typ- und Längeninformationen zu den einzelnen Feldern – werden in der Tabelle I_TAB_PARAMDESC übergeben.
  Es steht der FB BANK_MAP_UTIL_DATA_TO_GENFORM zur Verfügung, mit dem eine Parameterstruktur in die generische Darstellung überführt werden kann.
- I_PACKMANID: In diesem Parameter übergibt der Aufrufer die eindeutige Kennung der zu erzeugenden Paketeinteilung. Dieser Parameter wird dann beim Start einer Massenverarbeitung mit Standard-Paketeinteilung an den FB BANK_MAP_PP_START übergeben (siehe 5.4.1).
- I_PACKMANDESC: Kurzbeschreibung der Standardpaketeinteilung

5.5.1 Beispiel für die Generierung einer Paketeinteilung

Der folgende Beispielreport demonstriert den Aufruf des Bausteins:

```
REPORT PACKAGE_CREATION_EXAMPLE.
PARAMETERS:
  p_pmid   TYPE BANK_DTE_PMID_EXT,
  p_pmdesc TYPE BANK_DTE_PMID_EXT_DESC.
…

CONSTANTS:
  con_my_appl TYPE bkk_paapplcatg VALUE ´abcd´.
…

DATA:
  g_str_my_params TYPE my_params,
  g_tab_val       TYPE bank_tab_dataval,
  g_tab_desc      TYPE bank_tab_datadesc.
…

START-OF-SELECTION.
…
  CALL FUNCTION ´BANK_MAP_UTIL_DATA_TO_GENFORM´
       EXPORTING
            I_STR_DATA     = g_str_my_params
       IMPORTING
            E_TAB_DATA     = g_tab_val
            E_TAB_DATADESC = g_tab_desc.

  CALL FUNCTION ´BANK_MAP_PACKMAN_CONSTRUCTOR´
       EXPORTING
            I_APPLCATG        = con_my_appl
            I_TAB_PARAMVALUES = g_tab_val
            I_TAB_PARAMDESC   = g_tab_desc
            I_PACKMANID       = p_pmid
            I_PACKMANDESC     = p_pmdesc.
```

5.6 Nachrichtenbehandlung innerhalb der Parallelverarbeitung

Das PV-Tool behandelt Nachrichten gemäß dem Nachrichtenkonzept innerhalb der IBS Financial Services. Zu den Einzelheiten dieses Konzeptes wird an dieser Stelle auf die Programmierrichtlinien sowie auf die Dokumentation zur Nachrichtenklasse verwiesen. Im Zusammenhang mit der PV sind allerdings folgende Punkte zu beachten:

Durch die Startbausteine der PV (BANK_MAP_PP_START bzw. BANK_MAP_PP_RESTART) werden keine Nachrichtenobjekte erzeugt, da durch den FuB-Aufruf kein neuer Kanal geöffnet wird. Es ist somit durch den Aufrufer der FuB sicherzustellen, dass Nachrichtenobjekte vorhanden sind.

Durch die vom PV-Tool generierten Hintergrundjobs wird jeweils ein neuer Nachrichtenkanal geöffnet. Um die innerhalb der parallelen Jobs erzeugten Protokolle mit den beim Aufruf der Start-FB geschriebenen Protokolle zusammenfassen zu können, ist es sinnvoll, dem PV-Tool mitzuteilen, welche Nachrichtenobjekte mit welchen Parametern am Beginn der parallelen Jobs instantiiert werden sollen. Hierzu muß beim Aufruf der PV in der Importtabelle I_TAB_LOG die Liste der zu erzeugenden Nachrichtenobjekte übergeben werden. Eine Zeile dieser Tabelle umfasst folgende Felder:

Name	Typ	Bedeutung
OBJECT	BALOBJ_D	Anwendungsprotokoll-Objekt
SUBOBJECT	BALSUBOBJ	Anwendungsprotokoll-Unterobjekt
EXTNUMBER	BALNREXT	externe Protokollnummer
ALPROG	BALPROG	Programmname für Anwendungsprotokoll
ALDATE_DEL	ALDATE_DEL	Verfallsdatum des Protokolls
FLG_STANDARD_LOG	XFELD	Kennzeichen: Nachrichtenobjekt ist Objekt zu Standard-Anwendungsprotokoll. Dieses Kennzeichen darf nur für genau einen Eintrag in der Tabelle I_TAB_LOG gesetzt werden,

Innerhalb der Parallelverarbeitung kann durch Aufruf des FuB BANK_API_PROCFL_GET_LOG_HDL das aktuelle Standard-Anwendungsprotokoll sowie ein Handle auf das zugehörige Nachrichtenobjekt ermittelt werden. Das Standard-Handle wird außerdem an die in Abschnitt 4.17 beschriebene Methode 1400 (Start der Verarbeitung in einem parallelen Job) übergeben.

Um die zu einem Massenlauf gehörenden Protokolle wiederfinden zu können, empfiehlt es sich, in der Tabelle I_TAB_LOG das Feld EXTNUMBER zu füllen. Zur Nummernvergabe steht der Funktionsbaustein BANK_API_PP_LOG_CREATE_LOGNO zur Verfügung. Weiterhin sollte das Feld ALPROG gefüllt sein, falls verschiedene Anwendungen gleiche Anwendungsprotokoll-Objekte nutzen. Andernfalls können die von verschiedenen Anwendungen generierten Protokolle in Anzeigetransaktion nicht voneinander separiert werden.