

CICS Transaction Server for z/OS



XPI-Funktionsreferenz

Version 5 Release 5

CICS Transaction Server for z/OS



XPI-Funktionsreferenz

Version 5 Release 5

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen“ auf Seite 139 gelesen werden.

Inhaltsverzeichnis

Informationen zur vorliegenden Veröffentlichung im PDF-Format v

Kapitel 1. XPI-Funktion für Geschäftsanwendungsmanagerdomäne 1

Aufruf INQUIRE_ACTIVATION 1

Kapitel 2. XPI-Funktionen für Verzeichnisdomäne 3

Aufruf BIND_LDAP 3

Aufruf END_BROWSE_RESULTS 5

Aufruf FLUSH_LDAP_CACHE 6

Aufruf FREE_SEARCH_RESULTS 6

Aufruf GET_ATTRIBUTE_VALUE 7

Aufruf GET_NEXT_ATTRIBUTE 8

Aufruf GET_NEXT_ENTRY 9

Aufruf SEARCH_LDAP 10

Aufruf START_BROWSE_RESULTS 12

Aufruf UNBIND_LDAP 13

Kapitel 3. XPI-Funktionen für Dispatcher 15

Synchronisationsprotokolle für SUSPEND- und RESUME-Verarbeitung 15

 Protokoll für normale Synchronisation 15

 Synchronisationsprotokoll und Taskbereinigung 16

Aufruf ADD_SUSPEND 17

Aufruf CHANGE_PRIORITY 19

Aufruf DELETE_SUSPEND 20

Aufruf RESUME 21

Aufruf SUSPEND 22

Aufruf WAIT_MVS 26

Kapitel 4. XPI-Funktionen für Speicherauszugssteuerung 31

Aufruf SYSTEM_DUMP 31

Aufruf TRANSACTION_DUMP 32

Kapitel 5. XPI-Funktionen für Enqueue-Domäne 37

Funktion DEQUEUE 37

Funktion ENQUEUE 37

Kapitel 6. XPI-Funktionen für Kerneldomäne 41

Funktion START_PURGE_PROTECTION 41

Funktion STOP_PURGE_PROTECTION 41

Aufrufe für Bereinigungsschutz verschachteln 42

Kapitel 7. XPI-Funktionen für Ladeprogramme 43

Aufruf ACQUIRE_PROGRAM 43

Aufruf DEFINE_PROGRAM 45

Aufruf DELETE_PROGRAM 49

Aufruf IDENTIFY_PROGRAM 50

Aufruf RELEASE_PROGRAM 52

Kapitel 8. XPI-Funktionen für Protokollmanager 55

Aufruf INQUIRE_PARAMETERS 55

Aufruf SET_PARAMETERS 55

Kapitel 9. XPI-Funktionen für Überwachung 57

Aufruf INQUIRE_APP_CONTEXT 57

Aufruf INQUIRE_MONITORING_DATA 59

Aufruf MONITOR 60

Aufruf SET_TRACKING_DATA 63

Kapitel 10. XPI-Funktionen für Objekttransaktionen 65

Aufruf IMPORT_TRAN 65

Aufruf COMMIT_ONE_PHASE 66

Aufruf PREPARE 67

Aufruf COMMIT 67

Aufruf ROLLBACK 68

Aufruf SET_ROLLBACK_ONLY 68

Aufruf SET_COORDINATOR 69

Kapitel 11. XPI-Funktion für Parameterdomäne 71

Aufruf INQUIRE_FEATUREKEY 71

Kapitel 12. XPI-Funktionen für Programmverwaltung 73

Aufruf INQUIRE_PROGRAM 73

Aufruf INQUIRE_CURRENT_PROGRAM 82

Aufruf SET_PROGRAM 85

Aufruf START_BROWSE_PROGRAM 88

Aufruf GET_NEXT_PROGRAM 89

Aufruf END_BROWSE_PROGRAM 90

Aufruf INQUIRE_AUTOINSTALL 91

Aufruf SET_AUTOINSTALL 92

Aufruf BIND_CHANNEL 93

Kapitel 13. XPI-Funktionen für Statusdatenzugriff 95

Aufruf INQ_APPLICATION_DATA 95

Aufruf INQUIRE_SYSTEM 97

Aufruf SET_SYSTEM 102

Kapitel 14. XPI-Funktionen für Speichersteuerung 105

Aufruf GETMAIN 105

Aufruf FREEMAIN 108

Aufruf INQUIRE_ACCESS	109
Aufruf INQUIRE_ELEMENT_LENGTH	109
Aufruf INQUIRE_SHORT_ON_STORAGE	110
Aufruf INQUIRE_TASK_STORAGE	111
Aufruf SWITCH_SUBSPACE	112

Kapitel 15. XPI-Funktion für Tracesteuerung	113
Aufruf TRACE_PUT	113

Kapitel 16. XPI-Funktionen für Transaktionsverwaltung	115
Aufruf INQUIRE_CONTEXT	115
Aufruf INQUIRE_DTRTRAN	116
Aufruf INQUIRE_MXT	117

Aufruf INQUIRE_TCLASS	118
Aufruf INQUIRE_TRANDEF	120
Aufruf INQUIRE_TRANSACTION	129
Aufruf SET_TRANSACTION	133

Kapitel 17. XPI-Funktion für Benutzerjournalführung	135
Aufruf WRITE_JOURNAL_DATA	135

Kapitel 18. Threadsichere XPI-Befehle	137
--	------------

Bemerkungen	139
------------------------------	------------

Index	145
------------------------	------------

Informationen zur vorliegenden Veröffentlichung im PDF-Format

Die vorliegende Veröffentlichung im PDF-Format enthält Referenzinformationen zu den XPI-Makrofunktionen, die von globalen Benutzerexitprogrammen für den Zugriff auf verschiedene CICS-Services verwendet werden können. Die XPI-Funktionen werden nach funktionaler Zugehörigkeit, hauptsächlich nach CICS-Domäne, gruppiert. Informationen zur Verwendung dieser Funktionen in Programmen finden Sie in der PDF-Veröffentlichung *Developing CICS System Programs*. Vor CICS TS V5.4 waren die in der vorliegenden PDF-Veröffentlichung enthaltenen Informationen Bestandteil der Veröffentlichung *Customization Guide*.

Ausführliche Informationen zu der in dieser Veröffentlichung verwendeten Begrifflichkeit und Schreibweise finden Sie im IBM Knowledge Center im Abschnitt *Conventions and terminology used in the CICS documentation*.

Erstellungsdatum der Veröffentlichung

Die vorliegende Veröffentlichung im PDF-Format wurde am 14. Dezember 2018 erstellt.

Kapitel 1. XPI-Funktion für Geschäftsanwendungsmanagerdomäne

Die XPI stellt eine Funktion für die Geschäftsanwendungsmanagerdomäne bereit. Dabei handelt es sich um den DFHBABRX-Aufruf INQUIRE_ACTIVATION.

Aufruf INQUIRE_ACTIVATION

Die Funktion INQUIRE_ACTIVATION wird für den Makroaufruf DFHBABRX bereitgestellt. Mit dem Aufruf INQUIRE_ACTIVATION können Sie den Aktivitätsnamen und den Prozesstyp für die Geschäftstransaktionsaktivität der aktuellen Transaktion abrufen.

INQUIRE_ACTIVATION

```
DFHBABRX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(INQUIRE_ACTIVATION),
           [TRANSACTION_TOKEN(name8),]]
          [RETURNED_ACTIVITYID(buffer_descriptor)]
          [RETURNED_PROCESS_NAME(buffer_descriptor)]
          [OUT,
           [ACTIVITY_NAME(name16)]
           [PROCESS_TYPE(name8)]
           RESPONSE (name1 | *),
           REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

ACTIVITY_NAME(name16)

Gibt den aus 16 Zeichen bestehenden Namen der BTS-Aktivität zurück, der der Aktivität vom Benutzer zugeordnet wurde.

PROCESS_TYPE(name8)

Gibt die aus 8 Zeichen bestehende Kennung der Typdefinition des BTS-Prozesses zurück.

RETURNED_ACTIVITYID(buffer_descriptor)

Gibt die aus 52 Zeichen bestehende Kennung der BTS-Aktivität zurück, die der Aktivität von CICS zugeordnet wurde. Bei RETURNED_ACTIVITYID handelt es sich um einen vom Geschäftsanwendungsmanager (BAM) zurückgegebenen Ausgabeparameter. Der Datentyp ist Puffer, sodass das aufrufende Modul einen als Puffer zu verwendenden Bereich als Eingabe für den Aufruf bereitstellen muss.

RETURNED_PROCESS_NAME(buffer_descriptor)

Gibt den aus 36 Zeichen bestehenden Namen des BTS-Prozesses zurück. Bei RETURNED_PROCESS_NAME handelt es sich um einen vom Geschäftsanwendungsmanager zurückgegebenen Ausgabeparameter. Der Datentyp ist Puffer, sodass das aufrufende Modul einen als Puffer zu verwendenden Bereich als Eingabe für den Aufruf bereitstellen muss.

TRANSACTION_TOKEN(name8)

Gibt das Transaktionstoken für die Task an, für die die Informationen abgerufen werden.

RESPONSE- und REASON-Werte für INQUIRE_ACTIVATION

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	ACTIVITY_NOT_FOUND
DISASTER	----
INVALID	INVALID_BUFFER_LENGTH
KERNERROR	----
PURGED	----

Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Kapitel 2. XPI-Funktionen für Verzeichnisdomäne

Die XPI stellt Funktionen für die Verzeichnisdomäne bereit, mit denen Sie LDAP-Sitzungen öffnen und schließen, Ergebnisse für Berechtigungsnachweise durchsuchen, Ergebnisse durchsuchen und lokalisieren, die Anzeige beenden, den richtigen Wert zurückgeben und die Suche beenden können.

Bei den Verzeichnisdomänenfunktionen handelt es sich um die folgenden DFHD-DAPX-Aufrufe:

- BIND_LDAP
- END_BROWSE_RESULTS
- FLUSH_LDAP_CACHE
- FREE_SEARCH_RESULTS
- GET_ATTRIBUTE_VALUE
- GET_NEXT_ATTRIBUTE
- GET_NEXT_ENTRY
- SEARCH_LDAP
- START_BROWSE_RESULTS
- UNBIND_LDAP

Aufruf BIND_LDAP

Über den Aufruf BIND_LDAP wird eine Sitzung mit einem LDAP-Server aufgebaut.

Der LDAP-Server wird auf eine der folgenden Weisen angegeben:

- Über die LDAP-URL sowie den definierten Namen (DN) und das Kennwort des Benutzers, der zum Extrahieren der erwarteten Daten berechtigt ist.
- Über ein RACF-Profil in der Klasse LDAPBIND, das die LDAP-URL sowie den definierten Namen und das Kennwort enthält. Diese Option ist vorzuziehen, da sich auf diese Weise ein Codieren der LDAP-Berechtigungsnachweise in Ihrer Anwendung erübrigt.

BIND_LDAP

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
    FUNCTION(BIND_LDAP),
    {LDAP_BIND_PROFILE(block-descriptor) |
    LDAP_SERVER_URL((block-descriptor),DISTINGUISHED_NAME((block-descriptor),
    PASSWORD(block-descriptor),}
    [CACHE_SIZE(name4),CACHE_TIME_LIMIT(name4),]]
  [OUT,
    LDAP_SESSION_TOKEN(name4),
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

CACHE_SIZE(name4)

Ein Vollwort, das die Anzahl der für das Zwischenspeichern von LDAP-Suchergebnissen verfügbaren Byte angibt. Der Wert Null gibt an, dass die Cachegröße nicht begrenzt ist. Bei Angabe von CACHE_SIZE muss auch CACHE-

_TIME_LIMIT angegeben werden. Ist keiner der beiden Parameter angegeben, werden die Ergebnisse nicht im Cache gespeichert.

CACHE_TIME_LIMIT(name4)

Ein Vollwort, das den Zeitraum (in Sekunden) angibt, in dem LDAP-Suchergebnisse im Cache gespeichert werden. Der Wert Null gibt an, dass die Cachezeit nicht begrenzt ist.

DISTINGUISHED_NAME(block-descriptor)

Gibt die Position des definierten LDAP-Namens des Benutzers an, der zum Binden an den ausgewählten Server berechtigt ist. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte.

Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_BIND_PROFILE(block-descriptor)

Gibt die Position des Namens eines RACF-Profiles in der Klasse LDAPBIND mit der URL und den Berechtigungsnachweisen für den LDAP-Server an, auf den zugegriffen wird. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte.

Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax. Geben Sie entweder LDAP_BIND_PROFILE oder die drei Parameter LDAP_SERVER_URL, DISTINGUISHED_NAME und PASSWORD gemeinsam an.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API als Antwort auf den Empfang von URL und Benutzerberechtigungsdaten gesendet wird.

LDAP_SERVER_URL(block-descriptor)

Gibt die Position der LDAP-URL des LDAP-Servers (im Format 'ldap://server:port') an, auf den zugegriffen wird. Werden Doppelpunkt und Portnummer nicht angegeben, wird standardmäßig der Port 389 verwendet. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte.

Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_SESSION_TOKEN(name4)

Der Name des Vollworttokens, das die LDAP-Verbindung angibt.

PASSWORD(block-descriptor)

Gibt die Position des Kennworts für den Benutzer an, der im Eingabeparameter DISTINGUISHED_NAME angegeben ist. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte.

Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

RESPONSE- und REASON-Werte für BIND_LDAP

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_BUFFER_LENGTH
	INVALID_LDAP_PROFILE

<i>RESPONSE</i>	<i>REASON</i>
	INVALID_LDAP_URL
	LDAP_INACTIVE
	NOTAUTH
	NOTFOUND
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf END_BROWSE_RESULTS

Mit dem Aufruf END_BROWSE_RESULTS können Sie die Anzeigesitzung beenden, die über den Aufruf START_BROWSE_RESULTS gestartet wurde.

END_BROWSE_RESULTS

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
  FUNCTION(END_BROWSE_RESULTS),
  SEARCH_TOKEN(name4),]
  [OUT,
  [LDAP_RESPONSE(name4),]
  RESPONSE(name1 | *),
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

RESPONSE- und REASON-Werte für END_BROWSE_RESULTS

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
	INVALID_CALLING_SEQUENCE
	NOTFOUND
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf FLUSH_LDAP_CACHE

Mit dem Aufruf FLUSH_LDAP_CACHE wird der Inhalt der zwischengespeicherten Suchantworten für die angegebene LDAP-Verbindung entfernt.

FLUSH_LDAP_CACHE

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
  FUNCTION(FLUSH_LDAP_CACHE),
  LDAP_SESSION_TOKEN(name4),]
  [OUT,
  [LDAP_RESPONSE(name4),]
  RESPONSE(name1 | *),
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

LDAP_SESSION_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion BIND_LDAP zurückgegeben wurde.

RESPONSE- und REASON-Werte für FLUSH_LDAP_CACHE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
	LDAP_INACTIVE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf FREE_SEARCH_RESULTS

Mit dem Aufruf FREE_SEARCH_RESULTS wird der gesamte Speicher freigegeben, der von der Funktion SEARCH_LDAP gehalten wurde. Es werden keine Suchergebnisse mehr ermittelt und das Suchtoken wird inaktiviert. Wird die Funktion FREE_SEARCH_RESULTS nicht von der Anwendung aufgerufen, wird sie von CICS aufgerufen, wenn die Task beendet ist.

FREE_SEARCH_RESULTS

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
  FUNCTION(FREE_SEARCH_RESULTS),
  SEARCH_TOKEN(name4),]
  [OUT,
  [LDAP_RESPONSE(name4),]
  RESPONSE(name1 | *),
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

RESPONSE- und REASON-Werte für FREE_SEARCH_RESULTS

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf GET_ATTRIBUTE_VALUE

Mit dem Aufruf GET_ATTRIBUTE_VALUE können Sie den Wert abrufen, der einem vom Aufruf SEARCH_LDAP zurückgegebenen Attribut zugeordnet ist. Ein Eintrag stellt einen LDAP-Datensatz dar und ein Attribut ist eines der Elemente innerhalb eines Eintrags. Das Attribut kann über die Funktion GET_NEXT_ATTRIBUTE oder die Angabe des Attributnamens zurückgegeben werden.

GET_ATTRIBUTE_VALUE

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
  FUNCTION(GET_ATTRIBUTE_VALUE),
  SEARCH_TOKEN(name4),
  LDAP_ATTRIBUTE_NAME(block-descriptor),
  LDAP_ATTRIBUTE_VALUE(buffer-descriptor),
  [ATTRIBUTE_TYPE(name4),]
  [VALUE_ARRAY_POSITION(name4),]]
  [OUT,
  [LDAP_RESPONSE(name4),]
  RESPONSE(name1 | *),
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ATTRIBUTE_TYPE(name4)

Gibt das Schlüsselwort CHARACTER oder BINARY und damit das Format des Attributs an. Ist dieser Parameter nicht angegeben, wird dies als Angabe des Werts CHARACTER gewertet.

LDAP_ATTRIBUTE_NAME(block-descriptor)

Gibt die Position des LDAP-Attributnamens an. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse des Attributnamens, das zweite die Länge des Attributnamens in Byte. Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_ATTRIBUTE_VALUE(buffer-descriptor)

Gibt den Puffer an, in dem der Attributwert zurückgegeben werden soll. Für den Pufferdeskriptor wird eine Gruppe von drei Vollworten angegeben:

- Die Adresse, an die das Ergebnis zurückgegeben wird.
- Die maximale Größe der zurückgegebenen Daten in Byte.
- Die tatsächliche Länge des Ergebnisses in Byte. Für diesen Maximalwert kann ein Stern (*) angegeben werden. Die Länge wird in diesem Fall anschließend in DDAP_LDAP_ATTRIBUTE_VALUE_N zurückgegeben.

Weitere Informationen zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

VALUE_ARRAY_POSITION(name4)

Gibt die Position des angeforderten Werts in der Wertfeldgruppe für das aktuelle Attribut an. Dieser Parameter ist nur erforderlich, wenn mehrere Werte erwartet werden. Die Array-Indexierung beginnt bei Position 1.

RESPONSE- und REASON-Werte für GET_ATTRIBUTE_VALUE

RESPONSE	REASON
OK	----
EXCEPTION	INVALID_TOKEN
	NOTFOUND
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf GET_NEXT_ATTRIBUTE

Mit dem Aufruf GET_NEXT_ATTRIBUTE können Sie das nächste Attribut in einer Reihe von Attributen in einem Eintrag abrufen, der vom Aufruf SEARCH_LDAP zurückgegeben wurde. Ein Eintrag stellt einen LDAP-Datensatz dar und ein Attribut ist eines der Elemente innerhalb eines Eintrags.

GET_NEXT_ATTRIBUTE

```
DFHDDAPX [CALL],
          [CLEAR],
          [IN,
           FUNCTION(GET_NEXT_ATTRIBUTE),
           SEARCH_TOKEN(name4),
           LDAP_ATTRIBUTE_NAME(buffer-descriptor),]
          [OUT,
           [LDAP_RESPONSE(name4),]
           [VALUE_COUNT(name4),]
           RESPONSE(name1 | *),
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

LDAP_ATTRIBUTE_NAME(buffer-descriptor)

Gibt den Puffer an, in den der Attributwert zurückgegeben werden soll. Für den Pufferdeskriptor wird eine Gruppe von drei Vollworten angegeben:

- Die Adresse, an die die Daten zurückgegeben werden.
- Die maximale Größe der zurückgegebenen Daten in Byte.
- Die tatsächliche Länge der Daten in Byte. Für diesen Maximalwert kann ein Stern (*) angegeben werden. Die Länge wird in diesem Fall anschließend in DDAP_LDAP_ATTRIBUTE_NAME_N zurückgegeben.

Weitere Informationen zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

VALUE_COUNT(name4)

Ein Vollwort, das die Anzahl der Werte enthält, die für das Attribut zurückgegeben werden. In der Regel wird ein einziger Wert zurückgegeben.

RESPONSE- und REASON-Werte für GET_NEXT_ATTRIBUTE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	BROWSE_END
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
	INVALID_TOKEN
	NOT_FOUND
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf GET_NEXT_ENTRY

Mit dem Aufruf GET_NEXT_ENTRY können Sie den nächsten Eintrag in einer Reihe von Einträgen abrufen, die vom Aufruf SEARCH_LDAP zurückgegeben wurde. Ein Eintrag stellt einen LDAP-Datensatz dar. Mit diesem Aufruf wird der definierte Name (DN) zurückgegeben, der dem Eintrag zugeordnet ist.

GET_NEXT_ENTRY

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
    FUNCTION(GET_NEXT_ENTRY),
    SEARCH_TOKEN(name4),
    [DISTINGUISHED_NAME(buffer-descriptor),]]
  [OUT,
```

```
[LDAP_RESPONSE(name4),]
[ATTRIBUTE_COUNT(name4),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ATTRIBUTE_COUNT(name4)

Gibt die Anzahl der Attribute in dem abgerufenen Eintrag an.

DISTINGUISHED_NAME(buffer-descriptor)

Gibt den Puffer an, in den der definierte Name des nächsten Eintrags zurückgegeben werden soll. Für den Pufferdeskriptor wird eine Gruppe von drei Vollworten angegeben:

- Die Adresse, an die die Daten zurückgegeben werden.
- Die maximale Größe der zurückgegebenen Daten in Byte.
- Die tatsächliche Länge der Daten in Byte. Für diesen Maximalwert kann ein Stern (*) angegeben werden. Die Länge wird in diesem Fall anschließend in DDAP_DISTINGUISHED_NAME_N zurückgegeben.

Weitere Informationen zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

RESPONSE- und REASON-Werte für GET_NEXT_ENTRY

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
	BROWSE_END
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf SEARCH_LDAP

Mit dem Aufruf SEARCH_LDAP wird eine Suchanforderung an einen angegebenen LDAP-Server gesendet. Die Suchanforderung enthält einen definierten LDAP-Namen, der das Ziel der Suchanforderung angibt.

Für die Suchanforderung wird eine Reihe von Ergebnissen (Attribute oder Einträge) zurückgegeben, die durchsucht oder ausgewählt werden können. Ein Eintrag stellt einen LDAP-Datensatz dar und ein Attribut ist eines der Elemente innerhalb eines Eintrags.

SEARCH_LDAP

```
DFHDDAPX [CALL],
  [CLEAR],
  [IN,
  FUNCTION(SEARCH_LDAP),
  LDAP_SESSION_TOKEN(name4),
  DISTINGUISHED_NAME(block-descriptor),
  [FILTER(block-descriptor),]
  [SEARCH_TIME_LIMIT(name4),]]
  [OUT,
  SEARCH_TOKEN(name4),
  [LDAP_RESPONSE(name4),]
  [ENTRY_COUNT(name4),]
  RESPONSE(name1 | *),
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

DISTINGUISHED_NAME(block-descriptor)

Gibt die Position des definierten LDAP-Namens an. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte. Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

ENTRY_COUNT(name4)

Die Anzahl der bei der Suche zurückzugebenden LDAP-Einträge.

FILTER(block-descriptor)

Gibt die Position einer LDAP-Filterzeichenfolge an, die die Suche eingrenzt. Wird dieser Parameter nicht oder als Nullwert angegeben, wird der Suchfilter (objectClass=*) verwendet. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte. Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

LDAP_SESSION_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion BIND_LDAP zurückgegeben wurde.

SEARCH_TIME_LIMIT(name4)

Gibt das Zeitlimit (in Sekunden) für die Suche an. Werden in dem angegebenen Zeitraum keine Ergebnisse gefunden, wird die Suche beendet. Wird dieser Parameter nicht oder als Nullwert angegeben, ist der Zeitraum für die Suche nicht begrenzt.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das die aktuelle Position in der Suchanforderung kennzeichnet und hält.

RESPONSE- und REASON-Werte für SEARCH_LDAP

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_BUFFER_LENGTH
	INVALID_TOKEN
	NOTFOUND
	TIMED_OUT
	LDAP_INACTIVE

<i>RESPONSE</i>	<i>REASON</i>
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf START_BROWSE_RESULTS

Mit dem Aufruf START_BROWSE_RESULTS können Sie die Ergebnisse (Attribute oder Einträge) durchsuchen, die vom SEARCH_LDAP-Aufruf zurückgegeben wurden. Werden mehrere Einträge zurückgegeben, wird mit START_BROWSE_RESULTS zunächst der erste Eintrag durchsucht. Mit dem Aufruf GET_NEXT_ENTRY können Sie weitere Einträge abrufen.

START_BROWSE_RESULTS kann mehrmals für ein Suchtoken (SEARCH_TOKEN) abgesetzt werden. Wird der Aufruf nach einem Aufruf GET_NEXT_ENTRY oder GET_NEXT_ATTRIBUTE abgesetzt, wird die Anzeigeposition an den Anfang der Suchergebnisse zurückgesetzt.

START_BROWSE_RESULTS

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(START_BROWSE_RESULTS),
    SEARCH_TOKEN(name4),
    [DISTINGUISHED_NAME(buffer-descriptor),]]
    [OUT,
    [LDAP_RESPONSE(name4),]
    [ATTRIBUTE_COUNT(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

DISTINGUISHED_NAME(buffer-descriptor)

Gibt den Puffer an, in den der definierte Name des ersten bzw. einzigen gefundenen Suchergebnisses zurückgegeben werden soll. Für den Pufferdeskriptor wird eine Gruppe von drei Vollworten angegeben:

- Die Adresse, an die die Daten zurückgegeben werden.
- Die Länge des Puffers in Byte, in den die Daten zurückgegeben werden.
- Die maximale Länge der Daten in Byte. Für diesen Maximalwert kann ein Stern (*) angegeben werden. Die Länge wird in diesem Fall anschließend in DDAP_DISTINGUISHED_NAME_N zurückgegeben.

Weitere Informationen zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

ATTRIBUTE_COUNT(name4)

Ein Vollwort, das die Anzahl der Attribute angibt, die in dem aktuellen Eintrag durchsucht werden können.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

SEARCH_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion SEARCH_LDAP zurückgegeben wird.

RESPONSE- und REASON-Werte für START_BROWSE_RESULTS

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf UNBIND_LDAP

Mit dem Aufruf UNBIND_LDAP wird eine Sitzung mit einem LDAP-Server beendet.

UNBIND_LDAP

```
DFHDDAPX [CALL],  
  [CLEAR],  
  [IN,  
   FUNCTION(UNBIND_LDAP),  
   LDAP_SESSION_TOKEN(name4),]  
  [OUT,  
   [LDAP_RESPONSE(name4),]  
   RESPONSE(name1 | *),  
   REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

LDAP_RESPONSE(name4)

Gibt den Rückgabecode an, der von der LDAP-API gesendet wird.

LDAP_SESSION_TOKEN(name4)

Der Name des Vollworttokens, das von der Funktion BIND_LDAP zurückgegeben wurde.

RESPONSE- und REASON-Werte für UNBIND_LDAP

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN
	LDAP_INACTIVE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Kapitel 3. XPI-Funktionen für Dispatcher

Die XPI stellt fünf Dispatcherfunktionen bereit. Bei diesen Funktionen handelt es sich um die DFHDSSRX-Aufrufe ADD_SUSPEND, SUSPEND, RESUME, DELETE_SUSPEND und WAIT_MVS sowie um den DFHDSATX-Aufruf CHANGE_PRIORITY.

Die Einsatzmöglichkeiten für diese Dispatcheraufrufe sind begrenzt. Informieren Sie sich vor der Verwendung dieser Funktionen anhand des Abschnitts Global user exit programs über die näheren Details für die einzelnen Exitpunkte.

Anmerkung:

1. Zum Erstellen eines Aussetztokens müssen Sie den Aufruf ADD_SUSPEND absetzen, **bevor** Sie einen Aufruf SUSPEND oder RESUME absetzen.
2. Wird eine ausgesetzte Task storniert, schlägt der Aufruf SUSPEND mit dem RESPONSE-Wert PURGED und dem REASON-Wert TASK_CANCELLED fehl. Für den entsprechenden RESUME-Aufruf wird der RESPONSE-Wert EXCEPTION sowie der REASON-Wert TASK_CANCELLED zurückgegeben.
3. Überschreitet eine ausgesetzte Task ein zulässiges Zeitlimit, schlägt der Aufruf SUSPEND mit dem RESPONSE-Wert PURGED und dem REASON-Wert TIMED_OUT fehl. Für den entsprechenden RESUME-Aufruf wird der RESPONSE-Wert EXCEPTION sowie der REASON-Wert TIMED_OUT zurückgegeben.
4. Dispatcherprotokolle erfordern es, dass ein Aufruf RESUME abgesetzt wird, auch wenn der Aufruf SUSPEND aufgrund einer Stornierung der ausgesetzten Task oder aufgrund einer Zeitlimitüberschreitung bereinigt wurde. Sie dürfen für jeden Aufruf SUSPEND nur einen einzigen Aufruf RESUME absetzen.

Synchronisationsprotokolle für SUSPEND- und RESUME-Verarbeitung

Bei Verwendung der von der XPI bereitgestellten SUSPEND- und RESUME-Verarbeitung müssen Sie die richtigen Protokolle einhalten, damit die Taskbereinigung effektiv durchgeführt werden kann.

Protokoll für normale Synchronisation

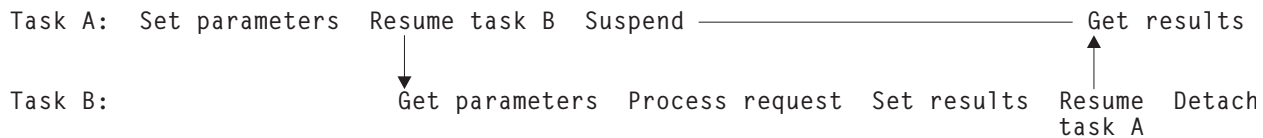
Im Normalfall umfasst die Synchronisation zwei Tasks und drei Operationen.

In den folgenden Beispieloperationen handelt es sich bei den Tasks um die Task A zum Anfordern eines Service und die Task B zum Verarbeiten einer Anforderung von Task A.

1. Task A startet die Anforderung auf folgende Weise:
 - Die von Task B zu verwendenden Parameter werden festgelegt.
 - Der Aufruf RESUME für Task B wird abgesetzt.
 - Der Aufruf SUSPEND wird abgesetzt.
2. Task B verarbeitet die Anforderung auf folgende Weise:
 - Die Parameter werden abgerufen.
 - Die Aktion wird durchgeführt.
 - Die Ergebnisse werden festgelegt.
 - Task B wird beendet (oder wartet auf weitere Arbeit).
3. Task A beendet die Interaktion auf folgende Weise:

- Die von Task B überlassenen Ergebnisse werden abgerufen.

Diese Sequenz lässt sich folgendermaßen darstellen:



Ohne RESUME und SUSPEND gestaltet sich die Ausführung wie folgt:

Set parameters; Get parameters; Process request; Set results; Get results

Dabei gilt, dass diese Aktionen immer **sequenziell** sind.

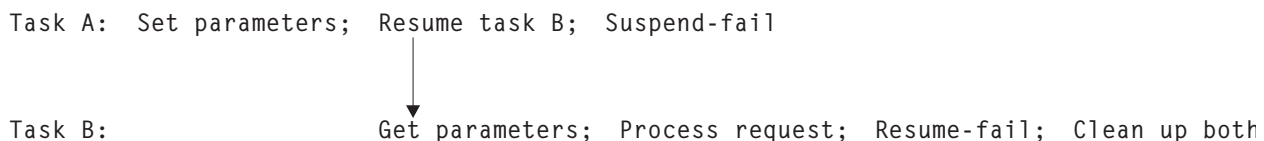
Synchronisationsprotokoll und Taskbereinigung

Wenn eine der Tasks gelöscht werden soll, handelt es sich dabei auf jeden Fall um Task A, da Task A die ausgesetzte Task ist. In diesem Fall würde die Ausführung von Task A nach dem fehlgeschlagenen Aufruf SUSPEND parallel zur Ausführung von Task B erfolgen und die richtige Serialisierung wäre nicht mehr gegeben. Ohne Programmänderung würde die Verarbeitung der Anforderung und das Festlegen von Ergebnissen zu derselben Zeit erfolgen wie das Abrufen von Ergebnissen. Dies hätte unvorhersehbare Ergebnisse zur Folge.

Alternativer Ansatz für Taskbereinigung

Eine Möglichkeit, dieses Problem zu vermeiden, besteht darin, dass Task A, wenn sie gelöscht werden soll, keine Aktion ausführt, die Task B beeinträchtigen könnte. Dies könnte bedeuten, dass Task A nicht abgehängt werden darf, wenn dadurch Speicherplatz freigegeben wird, auf den Task B zugreifen muss. Da Task B nun die einzige beteiligte Task ist, ist Task B für die Bereinigung für beide Tasks zuständig.

Die folgende Abbildung veranschaulicht die Sequenz:



Da die Taskbereinigung nur effektiv ist, wenn sie zwischen den Aufrufen SUSPEND und RESUME erfolgt, geht der Aufruf SUSPEND mit Fehlschlag dem Aufruf RESUME mit Fehlschlag voraus. Mit derselben Integritätsbedingung in Bezug auf die Serialisierung wie beim normalen Synchronisationsprotokoll kann das Protokoll für die Taskbereinigung logisch auf die folgende Sequenz reduziert werden:

Set parameters; Get parameters; Process request; Clean up

Der Unterschied besteht darin, dass die Aktionen zum Festlegen und Abrufen von Ergebnissen durch das Bereinigen ersetzt werden. Es ist von zentraler Bedeutung, dass nur diese beiden Sequenzen durchlaufen werden können. Beide Programme

müssen deshalb korrekt codiert werden. Von CICS wird sichergestellt, dass beide Tasks informiert werden, ob die SUSPEND- und RESUME-Verarbeitung erfolgreich war oder fehlgeschlagen ist.

Im Folgenden sind die Programmierschritte aufgeführt, die diesen Regeln entsprechen:

Programm für Task A	Programm für Task B
SET PARAMETERS;	
RESUME B;	GET PARAMETERS;
SUSPEND A;	PROCESS REQUEST;
	RESUME A;
if	if
RESPONSE = OK	RESPONSE != OK
then	then
GET RESULTS;	CLEAN UP;
endif	endif

Wird von SUSPEND und RESUME 'OK' zurückgegeben, folgt das Beispiel den Regeln für die normale Synchronisation. Die Verarbeitung wird mit dem Abrufen von Ergebnissen beendet. Wird weder von SUSPEND noch von RESUME der Wert 'OK' zurückgegeben, folgt das Beispiel den Regeln für das Protokoll für Taskbereinigung und die Verarbeitung wird mit der Bereinigung beendet.

Die zuvor beschriebene Sequenz stellt nur einen Lösungsansatz für den Umgang mit dem Problem der Taskbereinigung dar. Bei Verwendung dieser Methode ist Task B bei der Verarbeitung der Anforderung nicht bekannt, ob Task A gelöscht wurde. Task B muss deshalb bei der Verwendung von Ressourcen, deren Eigner Task A ist, mit besonderer Vorsicht vorgehen (für den Fall, dass Task A gelöscht wurde). In einigen Fällen kann diese Einschränkung zu Problemen führen.

Ein anderer Ansatz gestaltet sich wie im Folgenden beschrieben. Wenn Task A gelöscht werden soll, geschieht Folgendes:

1. Task A informiert Task B, dass Task A nicht mehr verfügbar ist, damit Task B keine Ressourcen mehr verwendet, deren Eigner Task A ist.
2. Task A führt eine eigene Bereinigungsverarbeitung durch (einschließlich des Absetzens des Aufrufs RESUME für den Aufruf SUSPEND mit der Antwort PURGED, wie für die Dispatcherprotokolle erforderlich) und wird abnormal beendet.
3. B führt eine eigene Bereinigungsverarbeitung durch.

Aufruf ADD_SUSPEND

Mit ADD_SUSPEND wird ein Aussetztoken angefordert, das anschließend zur Kennzeichnung eines SUSPEND/RESUME-Paars (Aussetzen/Wiederaufnehmen) verwendet werden kann.

ADD_SUSPEND

```
DFHDSSRX [CALL,]
    [CLEAR,]
    [IN,
    FUNCTION(ADD_SUSPEND),
    [RESOURCE_NAME(name16 | string | 'string'),]
    [RESOURCE_TYPE(name8 | string | 'string'),]]
    [OUT,
    SUSPEND_TOKEN(name4 | (Rn)),
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

RESOURCE_NAME(name16 | string | "string")

Gibt eine aus 16 Zeichen bestehende Zeichenfolge an, mit der die beim Aussetzen und Wiederaufnehmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name16

Der Name der Position, an der ein aus 16 Byte bestehender Wert gespeichert wird.

string Eine aus 16 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung: Für RESOURCE_NAME ist bei ADD_SUSPEND ein Standardwert vorgesehen, der verwendet wird, wenn der Parameter RESOURCE_NAME in einem SUSPEND-Aufruf nicht angegeben ist.

RESOURCE_TYPE(name8 | string | "string")

Gibt eine aus 8 Zeichen bestehende Zeichenfolge an, mit der die beim Aussetzen und Wiederaufnehmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name8 Der Name der Position, an der ein aus 8 Byte bestehender Wert gespeichert wird.

string Eine aus 8 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung: Für RESOURCE_TYPE ist bei ADD_SUSPEND ein Standardwert vorgesehen, der verwendet wird, wenn der Parameter RESOURCE_TYPE in einem SUSPEND-Aufruf nicht angegeben ist.

SUSPEND_TOKEN(name4 | (Rn))

Gibt ein vom System zugeordnetes Token zurück, um das SUSPEND/RESUME-Operationspaar für die Task zu kennzeichnen.

name4 Der Name eines aus 4 Byte bestehenden Felds, in dem das Token gespeichert wird.

(Rn) Ein Register, in das der Tokenwert geladen wird.

RESPONSE- und REASON-Werte für ADD_SUSPEND

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf CHANGE_PRIORITY

Mit dem Aufruf CHANGE_PRIORITY kann die Task, von der der Aufruf abgesetzt wird, die eigene Priorität ändern. Ein Ändern der Priorität einer anderen Task ist mit diesem Aufruf nicht möglich. Der Befehl führt dazu, dass die Task, von der der Aufruf abgesetzt wird, die Steuerung freigibt und so die Ausführung anderer Tasks möglich wird.

CHANGE_PRIORITY

```
DFHDSATX [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(CHANGE_PRIORITY),  
    PRIORITY(name1 | (Rn) | decimalint | literalconst),]  
    [OUT,  
    [OLD_PRIORITY(name1 | (Rn)),]  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

OLD_PRIORITY(name1 | (Rn))

Gibt die vorherige Priorität der Task, von der der Aufruf abgesetzt wird, zurück.

name1 Der Name eines aus 1 Byte bestehenden Felds mit der vorherigen Priorität der Task.

(Rn) Ein Register, bei dem das niedrigstwertige Byte den vorherigen Prioritätswert empfängt und für die übrigen Bytes Nullwerte festgelegt werden.

PRIORITY(name1 | (Rn) | decimalint | literalconst)

Gibt die neue Priorität an, die der Task, von der der Aufruf abgesetzt wird, zugeordnet werden soll.

name1 Der Name eines aus 1 Byte bestehenden Felds mit einem Wert im Bereich von 0 bis 255.

(Rn) Ein Register mit dem niedrigstwertigen Byte, das den neuen Prioritätswert enthält.

decimalint

Eine ganze Dezimalzahl kleiner oder gleich 255. Der Wert darf weder als Ausdruck noch in Hexadezimalschreibweise angegeben sein.

literalconst

Eine Zahl in Form eines Literals, z. B. B'00000000', X'FF', X'FCF4', "0" oder ein Gleichsetzungszeichen mit einem entsprechenden Wert.

RESPONSE- und REASON-Werte für CHANGE_PRIORITY

RESPONSE	REASON
OK	----
DISASTER	----
INVALID	----
KERNERROR	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf DELETE_SUSPEND

Mit dem Aufruf DELETE_SUSPEND wird ein einer Task zugeordnetes Aussetztoken freigegeben.

DELETE_SUSPEND

```
DFHDSSRX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(DELETE_SUSPEND),  
           SUSPEND_TOKEN(name4 | (Rn)),]  
          [OUT,  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

SUSPEND_TOKEN(name4 | (Rn))

Gibt ein vom System zugeordnetes Token zur Kennzeichnung des für die Task verwendeten SUSPEND/RESUME-Operationspaars an.

name4 Der Name eines aus 4 Byte bestehenden Felds, in dem das über einen Aufruf ADD_SUSPEND abgerufene Token gespeichert wurde.

(Rn) Ein Register mit dem zuvor erhaltenen Tokenwert.

RESPONSE- und REASON-Werte für DELETE_SUSPEND

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf RESUME

Mit dem Aufruf RESUME wird die Ausführung einer Task erneut gestartet, die ausgesetzt oder aufgrund einer Zeitlimitüberschreitung unterbrochen wurde.

Für jede SUSPEND-Anforderung darf es nur eine RESUME-Anforderung geben. Da es sich bei der Schnittstelle um eine asynchrone Schnittstelle handelt, kann eine SUSPEND-Anforderung vor oder nach der entsprechenden RESUME-Anforderung empfangen werden. Sie müssen sicherstellen, dass Sie den Überblick über die von Ihrem Exitprogramm abgesetzten SUSPEND- und RESUME-Anforderungen behalten.

RESUME

```
DFHDSSRX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(RESUME),  
           SUSPEND_TOKEN(name4 | (Rn)),  
           [COMPLETION_CODE(name1 | (Rn)),]  
          [OUT,  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

COMPLETION_CODE(name1 | (Rn))

Gibt einen benutzerdefinierten Ursachencode für RESUME bei der Aussetzungs- und Wiederaufnahmeverarbeitung an.

name1 Der Name eines aus 1 Byte bestehenden Bereichs für den zu empfangenden Code.

(Rn) Ein Register, bei dem das niedrigstwertige Byte den Beendigungscode enthält und die übrigen Bytes Nullwerte darstellen.

SUSPEND_TOKEN(name4 | (Rn))

Gibt ein vom System zugeordnetes Token zur Kennzeichnung des für die Task verwendeten SUSPEND/RESUME-Operationspaars an.

name4 Der Name einer Position mit einem aus 4 Byte bestehenden Token, das zuvor als Ausgabe eines Aufrufs ADD_SUSPEND empfangen wurde.

(Rn) Ein Register mit dem Tokenwert.

RESPONSE- und REASON-Werte für RESUME

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	TASK_CANCELLED
	TIMED_OUT
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

2. 'TASK_CANCELLED' bedeutet, dass die Task während des Aussetzens durch eine Bedieneraktion abgebrochen wurde und dass das Aussetztoken wieder zur Verfügung steht.

Aufruf SUSPEND

Mit dem Aufruf SUSPEND wird die Ausführung einer aktiven Task ausgesetzt.

Ausgesetzte Tasks können auf zwei Arten erneut gestartet werden. Die Ausführung einer Task kann durch einen von Ihnen abgesetzten XPI-Aufruf RESUME wieder aufgenommen werden und die Wiederaufnahme der Taskausführung erfolgt automatisch, wenn der von Ihnen für das Makro DFHDSSRX angegebene Wert für INTERVAL abläuft. Ausgesetzte Tasks können auch vom Bediener, von einer Anwendung oder über die Deadlock-Zeitlimitfunktion gelöscht werden.

SUSPEND

```
DFHDSSRX [CALL,]  
  [CLEAR,]  
  [IN,  
    FUNCTION(SUSPEND),  
    PURGEABLE(YES|NO),  
    SUSPEND_TOKEN(name4 | (Rn)),  
    [INTERVAL(name4 | (Rn)),]  
    [RESOURCE_NAME(name16 | string | 'string'),]  
    [RESOURCE_TYPE(name8 | string | 'string'),]  
    [TIME_UNIT(SECOND|MILLI_SECOND),]  
    [WLM_WAIT_TYPE,]]  
  [OUT,  
    [COMPLETION_CODE(name1 | (Rn)),]  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

COMPLETION_CODE (name1 | (Rn))

Gibt einen benutzerdefinierten Aktionsursachencode bei der Aussetzungs- und Wiederaufnahmeverarbeitung zurück.

name1 Der Name eines aus 1 Byte bestehenden Bereichs für den zu empfangenden Code. Der Wert in diesem Feld ist benutzerdefiniert und wird von CICS ignoriert.

(Rn) Ein Register, bei dem das niedrigstwertige Byte den Beendigungscode enthält und die übrigen Bytes Nullwerte darstellen.

INTERVAL(name4 | (Rn))

Gibt die Zeit in Sekunden oder Millisekunden an, nach der die Task automatisch wiederaufgenommen wird und den RESPONSE-Wert PURGED sowie REASON-Wert TIMED_OUT erhält. Die bei der Option INTERVAL verwendete Zeiteinheit richtet sich nach der Einstellung für die Option TIME_UNIT. Der Wert für INTERVAL überschreibt den für die Transaktion angegebenen Zeitlimitwert (DTIMOUT), soweit vorhanden.

name4 Der Name eines aus 4 Byte bestehenden Bereichs, der als Vollwort-Binärwert interpretiert wird.

(Rn) Ein Register mit dem Intervallwert, der einen Vollwort-Binärwert darstellt.

PURGEABLE(YES|NO)

Gibt an, ob Ihr Code eine abnormale Beendigung der Anforderung infolge ei-

ner Bereinigung verarbeiten kann. Es gibt vier Bereinigungstypen (siehe Tabelle 1). Die Angabe PURGEABLE(NO) bewirkt Folgendes beim Dispatcher:

- Alle Versuche, die Task zu löschen (PURGE), werden zurückgewiesen.
- Die Deadlock-Zeitlimitfunktion DTIMOUT (falls für die Task anwendbar) wird für die Dauer dieser Anforderung ausgesetzt.

Tabelle 1. Aufruf SUSPEND - RESPONSE(PURGED)

REASON	CONDITION	PURGEABLE (NO)	PURGEABLE (YES)
TASK_CANCELLED	PURGE	Abbruch	Fortfahren mit normaler Ausführung
	FORCEPURGE	Fortfahren mit normaler Ausführung	Fortfahren mit normaler Ausführung
TIMED_OUT	DTIMOUT	Abbruch	Fortfahren mit normaler Ausführung
	INTERVAL	Fortfahren mit normaler Ausführung	Fortfahren mit normaler Ausführung

Anmerkung: Bei FORCEPURGE wird immer davon ausgegangen, dass das Löschen der Task vom Benutzer gewünscht wird. Diese Option überschreibt deshalb die Option PURGEABLE(NO). Wurde vom Benutzer ein Wert für INTERVAL festgelegt, überschreibt auch dieser Wert die Option PURGEABLE(NO).

RESOURCE_NAME(name16 | string | "string")

Gibt eine aus 16 Zeichen bestehende Zeichenfolge an, mit der die beim Aussetzen und Wiederaufnehmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name16

Der Name der Position, an der ein aus 16 Byte bestehender Wert gespeichert wird.

string

Eine aus 16 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung:

1. Die Information RESOURCE_NAME wird von CICS nicht verwendet, jedoch in Traceeinträgen eingefügt und auf entsprechenden CEMT-Anzeigen angezeigt, um Ihnen einen Einblick in die Verarbeitung Ihrer Task zu ermöglichen. Verwenden Sie andere Werte als die, die von internen CICS-Anforderungen angegeben werden, um Mehrdeutigkeiten zu vermeiden. Beschreibungen zu Werten interner CICS-Anforderungen finden Sie im Abschnitt The resources that CICS tasks can wait for in Troubleshooting.
2. Ist RESOURCE_NAME nicht angegeben, wird der Standardwert, soweit vorhanden, von ADD_SUSPEND verwendet.

RESOURCE_TYPE(name8 | string | "string")

Gibt eine aus 8 Zeichen bestehende Zeichenfolge an, mit der die beim Ausset-

zen und Wiederaufnahmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name8 Der Name der Position, an der ein aus 8 Byte bestehender Wert gespeichert wird.

string Eine aus 8 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung:

1. Die Information RESOURCE_TYPE wird von CICS nicht verwendet, jedoch in Traceeinträgen eingefügt und auf entsprechenden CEMT-Anzeigen angezeigt, um Ihnen einen Einblick in die Verarbeitung Ihrer Task zu ermöglichen. Verwenden Sie andere Werte als die, die von internen CICS-Anforderungen angegeben werden, um Mehrdeutigkeiten zu vermeiden. Werte interner CICS-Anforderungen sind im Abschnitt The resources that CICS tasks can wait for in Troubleshooting dokumentiert.
2. Ist RESOURCE_TYPE nicht angegeben, wird der Standardwert, soweit vorhanden, von ADD_SUSPEND verwendet.

SUSPEND_TOKEN(name4 | (Rn))

Gibt ein vom System zugeordnetes Token zur Kennzeichnung des für die Task verwendeten SUSPEND/RESUME-Operationspaars an.

name4 Der Name einer Position mit einem aus 4 Byte bestehenden Token, das zuvor als Ausgabe eines Aufrufs ADD_SUSPEND empfangen wurde.

(Rn) Ein Register mit dem Tokenwert.

TIME_UNIT(SECOND | MILLI_SECOND)

Gibt die Zeiteinheit für die Option INTERVAL an.

SECOND

Die Option INTERVAL gibt die Anzahl der Sekunden bis zur Zeitlimitüberschreitung an.

MILLI_SECOND

Die Option INTERVAL gibt die Anzahl der Millisekunden bis zur Zeitlimitüberschreitung an.

WLM_WAIT_TYPE(name1)

Gibt an einer aus 1 Byte bestehenden Position die Ursache für das Aussetzen der Task an. Diese Ursache gibt dem MVS-Workload-Manager den Typ des Wartestatus an.

Für den Typ des Wartestatus werden folgende Wertentsprechungen verwendet:

CMDRESP

Es wird auf eine Befehlsantwort gewartet.

CONV

Es wird auf einen Datenaustausch gewartet.

DISTRIB

Es wird auf eine verteilte Anforderung gewartet.

IDLE

Für eine als Work Manager verwendete CICS-Task liegt keine Verarbeitungsanforderung vor, die in der Überwachungsumgebung ausgeführt werden darf. Dabei kann es sich z. B. um einen Journalführungscode handeln, der automatisch ausgesetzt wird, wenn keine auszuführenden E/A-Journalführungsoperationen vorliegen.

IO Es wird auf eine E/A-Operation oder eine unbestimmte, zur Ein-/Ausgabe gehörige Operation (Sperrern, Puffer, Zeichenfolge etc.) gewartet.

LOCK

Es wird auf eine Sperre gewartet.

MISC

Es wird auf eine nicht identifizierte Ressource gewartet.

Anmerkung: Bei diesem Wert handelt es sich um die Standardursache, die dem Wartestatus zugewiesen wird, wenn Sie die Ausführung einer Task aussetzen und den Parameter WLM_WAIT_TYPE nicht angeben.

OTHER_PRODUCT

Es wird auf den Abschluss einer Verarbeitung durch ein anderes Programm gewartet. Dies ist z. B. der Fall, wenn die Workload an Db2 übergeben wurde.

SESS_LOCALMVS

Es wird auf die Einrichtung einer Sitzung in dem MVS-Image gewartet, in dem die CICS-Region aktiv ist.

SESS_NETWORK

Es wird auf die Einrichtung einer Sitzung an einer anderen Position im Netz gewartet (d. h. nicht im MVS-Image).

SESS_SYSPLEX

Es wird auf die Einrichtung einer Sitzung im Sysplex (d. h. nicht im MVS-Image) gewartet.

TIMER

Es wird auf den Ablauf des Zeitlimits eines Zeitgebers gewartet. Ein Beispiel hierfür ist eine Task, die sich selbst in den Ruhemodus versetzt.

Geben Sie die Ursache für das Aussetzen der Task im Parameter WLM_WAIT_TYPE an, wenn CICS in einer MVS-Workload-Management-Umgebung im Modus 'Goal' (d. h. unter Verwendung eines zielorientierten Leistungsmanagements) ausgeführt wird.

Tabelle 2. RESPONSE- und REASON-Werte für SUSPEND

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	TASK_CANCELLED
	TIMED_OUT

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. TASK_CANCELLED bedeutet, dass die Task durch eine Bedieneraktion oder über einen Anwendungsbefehl abgebrochen wurde.
3. Nach dem Empfang des RESPONSE-Werts PURGED darf das Aussetztoken erst in einem weiteren Aufruf SUSPEND verwendet werden, wenn es über einen Aufruf RESUME, der als Entsprechung zu dem Aufruf SUSPEND mit der Antwort PURGED verwendet wird, zurückgesetzt wurde.
4. TIMED_OUT bedeutet, dass die Task automatisch wiederaufgenommen wurde, da der über den Wert für INTERVAL angegebene Zeitraum (oder das beim Anhängen der Task angegebene Zeitlimit) abgelaufen ist. Das Token bleibt jedoch ein Aussetztoken und muss Objekt eines Aufrufs RESUME sein, bevor es das Objekt eines Aufrufs DELETE_SUSPEND sein kann.

Aufruf WAIT_MVS

Mit dem Aufruf WAIT_MVS wird ein Wartestatus für einen MVS-Ereignissteuerblock (ECB = Event Control Block) oder eine Liste mit MVS-Ereignissteuerblöcken angefordert. Sie können den Aufruf WAIT_MVS beispielsweise absetzen, um den Abschluss einer MVS-Task abzuwarten, für die Sie einen Aufruf ATTACH abgesetzt und einen Ereignissteuerblock für Taskausführung bereitgestellt haben.

Der Inhalt der Ereignissteuerblöcke wird beim Empfang einer WAIT_MVS-Anforderung nicht vom Dispatcher gelöscht. Wurde ein Ereignissteuerblock bereits gesendet, wird die Steuerung sofort mit der Antwort 'OK' an das Exitprogramm zurückgegeben.

Ein Ereignissteuerblock darf nicht gleichzeitig für mehrere Wartestatus angegeben sein. Wird beim Empfang einer WAIT_MVS-Anforderung bereits auf einen Ereignissteuerblock gewartet, wird die Anforderung zurückgewiesen. Der RESPONSE-Code lautet 'DSSR_INVALID' und der REASON-Code 'DSSR_ALREADY_WAITING'.

Anmerkung: In WAIT_MVS-Anforderungen verwendete Ereignissteuerblöcke müssen immer mit dem MVS-Makro POST gesendet werden.

WAIT_MVS

```
DFHDSSRX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(WAIT_MVS),
           {ECB_ADDRESS(name4 | (Ra)) | ECB_LIST_ADDRESS(name4 | (Ra)),}
           PURGEABLE(YES|NO),
           [INTERVAL(name4 | (Rn)),]
           [RESOURCE_NAME(name16 | string | 'string'),]
           [RESOURCE_TYPE(name8 | string | 'string'),]
           [TIME_UNIT(SECOND|MILLI_SECOND),]
           [WLM_WAIT_TYPE,]
           [OUT,
            RESPONSE(name1 | *),
            REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ECB_ADDRESS(name4 | (Ra))

Gibt die Adresse des Ereignissteuerblocks an, auf den gewartet werden soll.

name4 Der Name einer Position, die eine ECB-Adresse enthält.

(Ra) Ein Register mit der Adresse eines Ereignissteuerblocks.

ECB_LIST_ADDRESS(name4 | (Ra))

Gibt die Adresse einer Liste von Ereignissteuerblockadressen an, auf die gewartet werden soll.

name4 Der Name einer Position, die eine ECB-Adresse - möglicherweise gefolgt von weiteren ECB-Adressen - enthält. Bei dem letzten Adresswort in der Liste ist das höchstwertige Bit mit '1' definiert.

(Ra) Ein Register, das auf eine derartige Adressliste verweist.

INTERVAL(name4 | (Rn))

Gibt die Zeit in Sekunden oder Millisekunden an, nach der die Task automatisch wiederaufgenommen wird und den RESPONSE-Wert PURGED sowie den REASON-Wert TIMED_OUT erhält. Die bei der Option INTERVAL verwendete Zeiteinheit richtet sich nach der Einstellung für die Option TIME_UNIT.

Der Wert für INTERVAL überschreibt den für die Transaktion angegebenen Zeitlimitwert (DTIMOUT), soweit vorhanden.

name4 Der Name eines aus 4 Byte bestehenden Bereichs, der als Vollwort-Binärwert interpretiert wird.

(Rn) Ein Register mit dem Intervallwert, der einen Vollwort-Binärwert darstellt.

PURGEABLE(YES|NO)

Gibt an, ob Ihr Code eine abnormale Beendigung der Anforderung infolge einer Bereinigung verarbeiten kann. Es gibt vier Bereinigungstypen (siehe Tabelle 3). Die Angabe PURGEABLE(NO) bewirkt Folgendes beim Dispatcher:

- Alle Versuche, die Task zu löschen (PURGE), werden zurückgewiesen.
- Die Deadlock-Zeitlimitfunktion DTIMOUT (falls für die Task anwendbar) wird für die Dauer dieser Anforderung ausgesetzt.

Tabelle 3. Aufruf SUSPEND - RESPONSE(PURGED)

REASON	CONDITION	PURGEABLE (NO)	PURGEABLE (YES)
TASK_CANCELLED	PURGE	Abbruch	Fortfahren mit normaler Ausführung
	FORCEPURGE	Fortfahren mit normaler Ausführung	Fortfahren mit normaler Ausführung
TIMED_OUT	DTIMOUT	Abbruch	Fortfahren mit normaler Ausführung
	INTERVAL	Fortfahren mit normaler Ausführung	Fortfahren mit normaler Ausführung

Anmerkung: Bei FORCEPURGE wird immer davon ausgegangen, dass das Löschen der Task vom Benutzer gewünscht wird. Diese Option überschreibt deshalb die Option PURGEABLE(NO). Wurde vom Benutzer ein Wert für INTERVAL festgelegt, überschreibt auch dieser Wert die Option PURGEABLE(NO).

RESOURCE_NAME(name16 | string | "string")

Gibt eine aus 16 Zeichen bestehende Zeichenfolge an, mit der die beim Aussetzen und Wiederaufnehmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name16

Der Name der Position, an der ein aus 16 Byte bestehender Wert gespeichert wird.

string Eine aus 16 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung: Die Information RESOURCE_NAME wird von CICS nicht verwendet, jedoch in Traceeinträgen eingefügt und auf entsprechenden CEMT-Anzeigen angezeigt, um Ihnen einen Einblick in die Verarbeitung Ihrer Task zu ermöglichen. Verwenden Sie andere Werte als die, die von internen CICS-Anforderungen angegeben werden, um Mehrdeutigkeiten zu vermeiden. Werte interner CICS-Anforderungen sind im Abschnitt The resources that CICS tasks can wait for in Troubleshooting dokumentiert.

RESOURCE_TYPE(name8 | string | "string")

Gibt eine aus 8 Zeichen bestehende Zeichenfolge an, mit der die beim Aussetzen und Wiederaufnehmen beteiligte Ressource dokumentiert und verfolgt werden kann. Die Adresse der Zeichenfolge kann nicht über eine Registernotation angegeben werden.

name Der Name der Position, an der ein aus 8 Byte bestehender Wert gespeichert wird.

string Eine aus 8 Byte bestehende Zeichenfolge ohne eingebettete Leerzeichen. Kürzere Zeichenfolgen werden mit Leerzeichen aufgefüllt, längere abgeschnitten.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Zeichenfolge kann Leerzeichen enthalten. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

Anmerkung: Die Information RESOURCE_TYPE wird von CICS nicht verwendet, jedoch in Traceeinträgen eingefügt und auf entsprechenden CEMT-Anzeigen angezeigt, um Ihnen einen Einblick in die Verarbeitung Ihrer Task zu ermöglichen. Verwenden Sie andere Werte als die, die von internen CICS-Anforderungen angegeben werden, um Mehrdeutigkeiten zu vermeiden. Werte interner CICS-Anforderungen sind im folgenden Abschnitt dokumentiert: The resources that CICS tasks can wait for in Troubleshooting.

TIME_UNIT(SECOND | MILLI_SECOND)

Gibt die Zeiteinheit für die Option INTERVAL an.

SECOND

Die Option INTERVAL gibt die Anzahl der Sekunden bis zur Zeitlimitüberschreitung an.

MILLI_SECOND

Die Option INTERVAL gibt die Anzahl der Millisekunden bis zur Zeitlimitüberschreitung an.

WLM_WAIT_TYPE(name1)

Gibt an einer aus 1 Byte bestehenden Position die Ursache für das Aussetzen der Task an. Diese Ursache gibt dem MVS-Workload-Manager den Typ des Wartestatus der Task an.

Für den Typ des Wartestatus werden folgende Wertentsprechungen verwendet:

CMDRESP

Es wird auf eine Befehlsantwort gewartet.

CONV

Es wird auf einen Datenaustausch gewartet.

DISTRIB

Es wird auf eine verteilte Anforderung gewartet.

IDLE

Für eine als Work Manager verwendete CICS-Task liegt keine Verarbeitungsanforderung vor, die in der Überwachungsumgebung ausgeführt werden darf. Dabei kann es sich z. B. um einen Journalführungscode handeln, der automatisch ausgesetzt wird, wenn keine auszuführenden E/A-Journalführungsoperationen vorliegen.

IO Es wird auf eine E/A-Operation oder eine unbestimmte, zur Ein-/Ausgabe gehörige Operation (Sperrern, Puffer, Zeichenfolge etc.) gewartet.

LOCK

Es wird auf eine Sperre gewartet.

MISC

Es wird auf eine nicht identifizierte Ressource gewartet. Bei diesem Wert handelt es sich um die Standardursache, die dem Wartestatus zugewiesen wird, wenn Sie die Ausführung einer Task aussetzen und den Parameter WLM_WAIT_TYPE nicht angeben.

OTHER_PRODUCT

Es wird auf den Abschluss einer Verarbeitung durch ein anderes Programm gewartet. Dies ist z. B. der Fall, wenn die Workload an Db2 übergeben wurde.

SESS_LOCALMVS

Es wird auf die Einrichtung einer Sitzung in dem MVS-Image gewartet, in dem die CICS-Region aktiv ist.

SESS_NETWORK

Es wird auf die Einrichtung einer Sitzung an einer anderen Position im Netz gewartet (d. h. nicht im MVS-Image).

SESS_SYSPLEX

Es wird auf die Einrichtung einer Sitzung im Sysplex (d. h. nicht im MVS-Image) gewartet.

TIMER

Es wird auf den Ablauf des Zeitlimits eines Zeitgebers gewartet. Ein Beispiel hierfür ist eine Task, die sich selbst in den Ruhemodus versetzt.

Wenn CICS in einer MVS-Workload-Management-Umgebung im Modus 'Goal' (d. h. unter Verwendung eines zielorientierten Leistungsmanagements) ausgeführt wird, empfiehlt es sich, die Ursache für das Aussetzen der Task im Parameter WLM_WAIT_TYPE anzugeben.

Tabelle 4. RESPONSE- und REASON-Werte für WAIT_MVS

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	TASK_CANCELLED
	TIMED_OUT

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. TIMED_OUT wird zurückgegeben, wenn das Intervall oder ein Deadlock-Zeitlimitintervall abläuft.
3. TASK_CANCELLED bedeutet, dass die Task durch eine Bedieneraktion oder über einen Anwendungsbefehl storniert wurde.

Kapitel 4. XPI-Funktionen für Speicherauszugssteuerung

Die XPI stellt zwei Funktionen für Speicherauszugssteuerung bereit. Dabei handelt es sich um die Aufrufe `SYSTEM_DUMP` und `TRANSACTION_DUMP` des Makros `DFHDUDUX`.

Einschränkung: `DFHDUDUX`-Aufrufe können in keinem Exitprogramm verwendet werden, das über einen globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Statistikdomäne
- Monitordomäne
- Speicherauszugsdomäne
- Dispatcherdomäne
- Programm für transiente Daten (TDP)

Aufruf `SYSTEM_DUMP`

Der Aufruf `SYSTEM_DUMP` bewirkt einen Systemspeicherauszug. Ist der von Ihnen in der Eingabe übergebene Systemspeicherauszugscode in der Tabelle für Systemspeicherauszugscodes enthalten, kann der Speicherauszug unterdrückt werden.

Informationen zur Speicherauszugstabelle und zur Funktionsweise der Tabelle finden Sie in den Abschnitten `Using dumps in problem determination` und `SET SYSDUMPCODE`.

SYSTEM_DUMP

```
DFHDUDUX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(SYSTEM_DUMP),  
           SYSDUMPCODE(name8 | string | "string"),  
           CALLER(block-descriptor),]  
          [TITLE(block-descriptor),]  
          [OUT,  
           DUMPID(name9 | *),  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

CALLER(block-descriptor)

Gibt die Quelle einer Systemspeicherauszugsanforderung an. Die von Ihnen an dieser Stelle angegebene Information erscheint im Systemspeicherauszugheader und kann dazu verwendet werden, das Exitprogramm kenntlich zu machen, von dem die Systemspeicherauszugsanforderung initiiert wurde. Eine Beschreibung gültiger Blockdeskriptoren finden Sie im Abschnitt `XPI syntax`.

DUMPID(name9 | *)

Gibt die Speicherauszugs-ID zurück.

name9 Der Name eines aus 9 Byte bestehenden Felds, in dem die zugeordnete ID empfangen wird.

SYSTEM_DUMPCODE(name8 | string | "string")

Gibt den Code für den Fehler an, der den Aufruf für den Systemspeicheraus-
zug verursacht hat. Systemspeicherauszugscodes werden in der Speicheraus-
zugstabelle gespeichert.

name8 Der Name einer Position, die eine aus 8 Byte bestehende Zeichenfolge
enthält.

string Eine Zeichenfolge ohne eingebettete Leerzeichen. Aus der Zeichenfolge
wird vom Makro eine aus 8 Byte bestehende Literalkonstante generiert.
Die Zeichenfolge wird dazu je nach Bedarf abgeschnitten oder mit
Leerzeichen verlängert.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge, die möglicherweise
Leerzeichen enthält. Dieser Wert wird auf die gleiche Weise verarbeitet
wie die vorangehende Zeichenfolge ("string").

TITLE(block-descriptor)

Gibt einen Bereich mit dem Text an, der in der Ausgabe des Systemspeicher-
auszugs im Speicherauszugsheder angezeigt werden soll.

RESPONSE- und REASON-Werte für SYSTEM_DUMP

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	FESTAE_FAILED
	INSUFFICIENT_STORAGE
	IWMWQWRK_FAILED
	NO_DATASET
	PARTIAL_SYSTEM_DUMP
	SDUMP_BUSY
	SDUMP_FAILED
	SDUMP_NOT_AUTHORIZED
	SUPPRESSED_BY_DUMPOPTION
	SUPPRESSED_BY_DUMPTABLE
	SUPPRESSED_BY_USEREXIT
DISASTER	----
INVALID	INVALID_DUMPCODE
	INVALID_PROBDESC
	INVALID_SVC_CALL
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und
REASON im Abschnitt Making an XPI call.

Aufruf TRANSACTION_DUMP

Der Aufruf TRANSACTION_DUMP bewirkt einen Transaktionsspeicherauszug. Ist
der von Ihnen in der Eingabe übergebene Transaktionsspeicherauszugscode in der
Tabelle für Transaktionsspeicherauszugscodes enthalten, kann der Speicherauszug
unterdrückt und bei Bedarf ein Systemspeicherauszug erstellt werden.

Informationen zur Speicherauszugstabelle und zur Funktionsweise der Tabelle fin-
den Sie in den Abschnitten Using dumps in problem determination und SET
TRANDUMPCODE.

Gültige Zeichen sind Großbuchstaben (A-Z), Kleinbuchstaben (a-z) und Ziffern (0-9) sowie die Sonderzeichen \$ @ # / % & ? ! : | ; , ¢ + * ~ - und _ . Je nach Position sind auch die folgenden Zeichen gültig: < > . = und ". Von Ihnen eingegebene Kleinbuchstaben werden in Großbuchstaben umgewandelt.

Wichtig

Bei der Verwendung der XPI in einem frühen Stadium der Initialisierung ist eine Einschränkung zu beachten. Starten Sie Exitprogramme, die die XPI-Funktionen TRANSACTION_DUMP, WRITE_JOURNAL_DATA, MONITOR und INQUIRE_MONITOR_DATA verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt Writing initialization and shutdown programs.

TRANSACTION_DUMP

```
DFHDUDUX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(TRANSACTION_DUMP),
           TRANSACTION_DUMP_CODE(name4 | string | 'string')
           [CSA(NO|YES),]
           [PROGRAM(NO|YES),]
           [SEGMENT(block-descriptor),]
           [SEGMENT_LIST(block-descriptor),]
           [TCA(NO|YES),]
           [TERMINAL(NO|YES),]
           [TRANSACTION(NO|YES),]
           [TRT(NO|YES),]]
          [OUT,
           DUMPID(name9 | *),
           RESPONSE(name1 | *),
           REASON(name1 | *)]
```

Anmerkung: Dieser Befehl ist **NICHT** threadsicher.

CSA(NO|YES)

Gibt an, ob der allgemeine Systembereich (Common System Area, CSA) in den Transaktionsspeicherauszug eingeschlossen wird. Der Standardwert ist 'NO'.

DUMPID(name9 | *)

Gibt die Speicherauszugs-ID zurück.

name9 Der Name eines aus 9 Byte bestehenden Felds, in dem die zugeordnete ID empfangen wird.

PROGRAM(NO|YES)

Gibt an, ob alle Programmspeicherbereiche, die zu der Task gehören, in den Transaktionsspeicherauszug eingeschlossen werden. Der Standardwert ist 'NO'.

SEGMENT(block-descriptor)

Gibt die Adresse und die Länge eines einzelnen Speicherblocks an, für den ein Speicherauszug erstellt werden soll. Eine Beschreibung gültiger Blockdeskriptoren finden Sie im Abschnitt XPI syntax. Die Optionen SEGMENT und SEGMENT_LIST schließen sich gegenseitig aus.

SEGMENT_LIST(block-descriptor)

Gibt Adresse und Länge eines *Sets* zusammenhängender Wortpaare an. Das erste Wort in den einzelnen Paaren gibt die Länge (**length**) eines Speichersegments an (in Byte), für das ein Speicherauszug erstellt werden soll. Das zweite Wort enthält die Adresse (**address**) des Speichersegments. Das Ende der Auf-

zählung muss durch ein Wort gekennzeichnet sein, das die Zeichenfolge X'FFFFFFFF' enthält. Die Optionen SEGMENT und SEGMENT_LIST schließen sich gegenseitig aus.

TCA(NO|YES)

Gibt an, ob der Tasksteuerbereich (Task Control Area, TCA) in den Transaktionsspeicherauszug eingeschlossen werden soll. Der Standardwert ist 'NO'.

TERMINAL(NO|YES)

Gibt an, ob alle Terminalspeicherbereiche, die zu der Task gehören, in den Transaktionsspeicherauszug eingeschlossen werden. Der Standardwert ist 'NO'.

TRANSACTION(NO|YES)

Gibt an, ob alle Transaktionsspeicherbereiche, die zu der Task gehören, in den Transaktionsspeicherauszug eingeschlossen werden. Der Standardwert ist 'NO'.

TRANSACTION_DUMP CODE(name4 | string | "string")

Gibt den Code für den Fehler an, der den Aufruf für den Transaktionsspeicherauszug verursacht hat. Transaktionsspeicherauszugscodes werden in der Speicherauszugstabelle gespeichert.

name4 Der Name einer Position, die eine aus 4 Byte bestehende Zeichenfolge enthält.

string Eine Zeichenfolge ohne eingebettete Leerzeichen. Aus der Zeichenfolge wird vom Makro eine aus 4 Byte bestehende Literalkonstante generiert. Die Zeichenfolge wird dazu je nach Bedarf abgeschnitten oder mit Leerzeichen verlängert.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge, die möglicherweise Leerzeichen enthält. Dieser Wert wird auf die gleiche Weise verarbeitet wie die vorangehende Zeichenfolge ("string").

TRT(NO|YES)

Gibt an, ob die Tracetabelle (Trace Table, TRT) in den Transaktionsspeicherauszug eingeschlossen werden soll. Der Standardwert ist 'NO'.

RESPONSE- und REASON-Werte für TRANSACTION_DUMP

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	FESTAE_FAILED
	INSUFFICIENT_STORAGE
	IWMWQWRK_FAILED
	NOT_OPEN
	OPEN_ERROR
	PARTIAL_SYSTEM_DUMP
	PARTIAL_TRANSACTION_DUMP
	SDUMP_BUSY
	SDUMP_FAILED
	SDUMP_NOT_AUTHORIZED
	SUPPRESSED_BY_DUMPOPTION
	SUPPRESSED_BY_DUMPTABLE
	SUPPRESSED_BY_USEREXIT
DISASTER	----
INVALID	INVALID_DUMP CODE
	INVALID_PROBDESC
	INVALID_SVC_CALL
KERNERROR	----

RESPONSE
PURGED

REASON

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu *RESPONSE* und *REASON* im Abschnitt Making an XPI call.
2. *NOT_OPEN* bedeutet, dass die CICS-Speicherauszugsdatei inaktiv ist.
3. *OPEN_ERROR* bedeutet, dass beim Öffnen der CICS-Speicherauszugsdatei ein Fehler aufgetreten ist.
4. *PARTIAL* bedeutet, dass der aufgrund der Anforderung erstellte Transaktions-speicherauszug nicht vollständig ist.

Kapitel 5. XPI-Funktionen für Enqueue-Domäne

Die XPI stellt zwei Enqueue-Domänenfunktionen bereit. Dabei handelt es sich um die DFHNQEDX-Aufrufe DEQUEUE und ENQUEUE.

Funktion DEQUEUE

Die Funktion DEQUEUE wird für den Makroaufruf DFHNQEDX bereitgestellt. Sie gibt eine Ressource frei, für die zuvor mit einem Funktionsaufruf ENQUEUE eine Einreihung eingerichtet wurde.

DEQUEUE

```
DFHNQEDX [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(DEQUEUE),  
    {ENQUEUE_TOKEN(name4),  
    ENQUEUE_NAME1(address,length),[ENQUEUE_NAME2(address,length),]}  
    MAX_LIFETIME(DISPATCHER_TASK),]  
    [ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR),]  
    [OUT,  
    RESPONSE (name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Die Parameter ENQUEUE_TOKEN, ENQUEUE_NAME1, ENQUEUE_NAME2, MAX_LIFETIME (DISPATCHER_TASK) und ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR) stimmen mit den Parametern des Funktionsaufrufs ENQUEUE überein.

RESPONSE- und REASON-Werte für DEQUEUE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	
	ENQUEUE_NOT_OWNED
	ENQUEUE_LOCKED

Funktion ENQUEUE

Die Funktion ENQUEUE wird für den Makroaufruf DFHNQEDX bereitgestellt. ENQUEUE ermöglicht das Einreihen in eine Warteschlange für eine angegebene Ressource.

Standardmäßig werden alle über einen XPI-Befehl ENQUEUE erstellten Einreihungen zu einem speziellen Einreihungspool namens DISPATCH zugeordnet und als CICS-interne Einreihungen behandelt. XPI-Einreihungen kollidieren nicht mit Einreihungen, die über **EXEC CICS ENQ**-Befehle erstellt werden. Diese Einreihungen werden ausgerichtet am angegebenen Einreihungsmodell zu anderen Einreihungspools hinzugefügt. Eine aktive Einreihung über einen Befehl EXEC CICS ENQ für eine Zeichenfolge verhindert beispielsweise nicht, dass ein XPI-Befehl ENQUEUE für dieselbe Zeichenfolge erfolgreich ausgeführt werden kann.

Anmerkung:

- XPI-Einreihungen können nicht mit der CICS-SPI (System Programming Interface) durchsucht werden.
- XPI-Einreihungen können nicht über Einreihungsmodelle (Ressource ENQMODEL) gesteuert werden.

Wenn Sie den optionalen Parameter **ENQUEUE_TYPE** verwenden, kann mit dem XPI-Befehl **ENQUEUE** eine Einreihung für dieselbe Ressource erfolgen, für die auch ein Befehl **EXEC CICS ENQ** abgesetzt wurde, und umgekehrt. Anwendungen können Prozesse mit **EXEC CICS**- und **EXEC XPI**-Befehlen synchronisieren.

ENQUEUE

```
DFHNQEDX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(ENQUEUE),
           ENQUEUE_NAME1(address,length),
           [ENQUEUE_NAME2(address,length),]
           MAX_LIFETIME(DISPATCHER_TASK),
           [ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR),]
           [WAIT(YES|NO),]
           [PURGEABLE(YES|NO),]
           [OUT,
            [ENQUEUE_TOKEN(name4),]
            [DUPLICATE_REQUEST,]
            RESPONSE (name1 | *),
            REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

DUPLICATE_REQUEST

Gibt an dass die anfordernde Dispatcher-Task bereits Eigner der Ressource ist, auf die sich die Einreihungsanforderung bezieht.

ENQUEUE_NAME1(address,length)

Gibt den höchstwertigen Teil des Namens an, auf den sich die Einreihungsanforderung bezieht.

ENQUEUE_NAME2(address,length)

Gibt den niedrigstwertigen Teil des Namens an, auf den sich die Einreihungsanforderung bezieht.

ENQUEUE_TOKEN(name4)

Ermöglicht es einer nachfolgenden **DEQUEUE**-Anforderung, zur Kennzeichnung der Ressource anstelle des Einreihungsnamens ein Token zu verwenden. Auf diese Weise kann die NQ_Domäne den Einreihungssteuerblock direkt und somit effizienter lokalisieren.

ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR)

Gibt den Typ der Ressource an, auf die sich die Einreihungsanforderung bezieht. Die XPI-Option gibt das typische **DFHNQEDX**-Verhalten an. Der verwendete Ressourcenpool ist ausschließlich für die XPI vorgesehen und ein Zugriff über die CICS-API ist nicht möglich. Verwenden Sie **EXECSTRN** oder **EXECADDR**, um anzuzeigen, dass **ENQUEUE_NAME1** eine Einreihungsressource angibt, die sich in demselben Namespace befindet wie die von **EXEC CICS ENQ** verwendete Ressource. Weitere Informationen zu **EXECSTRN** und **EXECADDR** finden Sie im Abschnitt *The resources that CICS tasks can wait for in Troubleshooting*.

MAX_LIFETIME(DISPATCHER_TASK)

Die Option MAX_LIFETIME (DISPATCHER_TASK) ist erforderlich und gibt an, dass die anfordernde Dispatcher-Task der Eigner aller XPI-Einreihungen ist.

Wenn Sie den Aufruf ENQUEUE XPI verwenden, um sicherzustellen, dass Ihre globalen Benutzerexitprogramme threadsicher sind, wird empfohlen, Ressourcen während des Aufrufs des globalen Benutzerexitprogramms freizugeben (dequeue), in dem die entsprechenden Einreihungen erfolgt sind. Da keine Wiederherstellungsservices zum Stoppen globaler Benutzerexits bereitgestellt werden, stellt CICS jedoch sicher, dass alle ausstehenden XPI-Einreihungen automatisch freigegeben werden, wenn die Dispatcher-Task beendet wird. Führt die Dispatcher-Task eine CICS-Transaktion aus, wird die Dispatcher-Task beendet, wenn die CICS-Transaktion beendet wird, unabhängig davon, ob es sich um eine normale oder abnormale Beendigung handelt.

Normalerweise ist die anfordernde Transaktion Eigner einer Einreihung und die in der Transaktion enthaltenen Arbeitseinheiten werden als Ankerpunkt für die Einreihungssteuerblöcke verwendet. Die XPI benötigt jedoch keine Transaktionsumgebung und globale Benutzerexits können unter Dispatcher-Tasks aufgerufen werden, die keine Transaktionen oder Arbeitseinheiten aufweisen.

PURGEABLE(YES|NO)

Gibt an, ob eine Bereinigungsanforderung oder Zeitlimitüberschreitung für die Task berücksichtigt wird, wenn die anfordernde Dispatcher-Task auf die Einreihung warten muss.

WAIT(YES|NO)

Gibt an, ob die Dispatcher-Task wartet, wenn die Ressource zum aktuellen Zeitpunkt Gegenstand einer Einreihungsanforderung einer anderen Dispatcher-Task ist.

RESPONSE- und REASON-Werte für ENQUEUE

RESPONSE	REASON
OK	----
EXCEPTION	ENQUEUE_BUSY ENQUEUE_LOCKED ENQUEUE_DISABLED LIMIT_EXCEEDED SYSEQ_FAILURE INVALID_PHASE
PURGED	TASK_CANCELLED TIMED_OUT

Kapitel 6. XPI-Funktionen für Kerneldomäne

Die XPI stellt zwei Kerneldomänenfunktionen bereit. Dabei handelt es sich um die DFHKEDSX-Aufrufe START_PURGE_PROTECTION und STOP_PURGE_PROTECTION.

Funktion START_PURGE_PROTECTION

Die Funktion START_PURGE_PROTECTION wird für den Makroaufruf DFHKEDSX bereitgestellt. Mit dieser Funktion kann die Bereinigung für die aktuelle Task unterdrückt werden, sofern es sich nicht um eine erzwungene Bereinigung handelt. Die Funktion kann von allen globalen Benutzerexitprogrammen verwendet werden, um die Bereinigung während eines globalen Benutzerexitaufrufs zu verhindern.

Zu jedem Aufruf START_PURGE_PROTECTION sollte generell ein entsprechender Funktionsaufruf STOP_PURGE_PROTECTION vorliegen, um den Zeitraum für den Bereinigungsschutz beim Ausführen von Programmlogik, die diesen Schutz benötigt, zu beenden.

START_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]  
          [CLEAR,]  
          [IN,  
          FUNCTION(START_PURGE_PROTECTION),]  
          [OUT,  
          RESPONSE (name1 | *)]
```

Dieser Befehl ist threadsicher.

Für diesen Aufruf sind keine Eingabe- oder Ausgabeparameter vorgesehen. Es gibt lediglich einen RESPONSE-Wert.

Funktion STOP_PURGE_PROTECTION

Die Funktion STOP_PURGE_PROTECTION wird für den Makroaufruf DFHKEDSX bereitgestellt. Mit dieser Funktion wird die Bereinigung für die aktuelle Task erneut aktiviert, nachdem sie über einen vorherigen Funktionsaufruf START_PURGE_PROTECTION ausgesetzt wurde.

STOP_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]  
          [CLEAR,]  
          [IN,  
          FUNCTION(STOP_PURGE_PROTECTION),]  
          [OUT,  
          RESPONSE (name1 | *)]
```

Dieser Befehl ist threadsicher.

Für diesen Aufruf sind keine Eingabe- oder Ausgabeparameter vorgesehen. Es gibt lediglich einen RESPONSE-Wert.

Aufrufe für Bereinigungsschutz verschachteln

Die Funktionen `START_PURGE_PROTECTION` und `STOP_PURGE_PROTECTION` können verschachtelt sein. Wenn mehrere Aufrufe `START_PURGE_PROTECTION` für eine Task abgesetzt werden, müssen Sie darauf achten, dass die richtige Anzahl von Aufrufen `STOP_PURGE_PROTECTION` abgesetzt wird, um den Schutz vor Bereinigung aufzuheben.

Wenn Sie zwei Startaufrufe und lediglich einen Stoppaufruf absetzen, besteht der Bereinigungsschutz für die aktuelle Task weiterhin.

Für eine aktuelle Task können mehrere globale Benutzerexitprogramme vorliegen. Entwerfen Sie Ihre Exitprogramme so, dass der Schutz vor Bereinigung ordnungsgemäß aufgehoben wird. Das folgende Beispiel veranschaulicht die Verschachtelung:

XEIIN:

EXIT_PROG1: Gibt den Aufruf `START_PURGE_PROTECTION` aus.

XFCREQ:

EXIT_PROG2: Gibt den Aufruf `START_PURGE_PROTECTION` aus.

XFCREQC:

EXIT_PROG3: Gibt den Aufruf `STOP_PURGE_PROTECTION` aus.

XEIOUT:

EXIT_PROG4: Gibt den Aufruf `STOP_PURGE_PROTECTION` aus.

Kapitel 7. XPI-Funktionen für Ladeprogramme

Die XPI stellt fünf Ladeprogrammfunktionen bereit. Bei diesen Funktionen handelt es sich um die DFHLDLXD-Aufrufe ACQUIRE_PROGRAM, DEFINE_PROGRAM, DELETE_PROGRAM, IDENTIFY_PROGRAM und RELEASE_PROGRAM.

Die CICS-Ladeprogrammservices, die XPI eingeschlossen, erkennen nicht LE-konforme Assemblerprogramme (LE = Language Environment), die mit AMODE(64) verknüpft sind. Der Adressierungsmodus des Moduls wird im zurückgegebenen Einstiegspunktparameter angegeben. AMODE(64) wird angegeben, wenn das Bit 0 den Wert '0' und das Bit 31 den Wert '1' hat (entspricht der Adressierungsmoduskonvention des Betriebssystems z/OS).

Einschränkung: DFHLDLXD-Aufrufe können in keinem Exitprogramm verwendet werden, das an einem globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Statistikdomäne
- Monitordomäne
- Speicherauszugsdomäne
- Dispatcherdomäne
- Programm für transiente Daten (TDP)

Aufruf ACQUIRE_PROGRAM

ACQUIRE_PROGRAM gibt die Einstiegs- und die Ladepunktadresse sowie die Länge und ein neues Programmtoken für eine verwendbare Kopie des angegebenen Programms zurück, das durch den Namen oder ein Programmtoken ausgewiesen werden kann.

ACQUIRE_PROGRAM

```
DFHLDLXD [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(ACQUIRE_PROGRAM),  
    {PROGRAM_NAME(name8 | string | 'string')|  
    PROGRAM_TOKEN(name8)},  
    [SUSPEND(NO|YES),]  
    [OUT,  
    ENTRY_POINT(name4 | (Ra)),  
    [LOAD_POINT(name4 | (Ra)),]  
    [NEW_PROGRAM_TOKEN(name8),]  
    [PROGRAM_ATTRIBUTE(name1 | (Rn)),]  
    [PROGRAM_LENGTH(name4 | (Rn)),]  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ENTRY_POINT(name4 | (Ra))

Gibt die Einstiegspunktadresse des Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende, aus 31 Bit bestehende Einstiegsadresse.

(Ra) Ein Register für die zu empfangende Einstiegsadresse.

LOAD_POINT(name4 | (Ra))

Gibt die Ladepunktadresse des Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende Ladeadresse.

(Ra) Ein Register für die Ladeadresse.

NEW_PROGRAM_TOKEN(name8)

Gibt das neue Programmtoken für eine verwendbare Kopie des angegebenen Programms zurück.

name8 Der Name einer Position für das zu empfangende, aus 8 Byte bestehende Token, das das Programm und die Instanz kennzeichnet.

PROGRAM_ATTRIBUTE(name1 | (Rn))

Gibt das Programmattribut zurück.

name1 Der Name einer aus 1 Byte bestehenden Position für das zu empfangende Programmattribut.

(Rn) Ein Register, bei dem das niedrigstwertige Byte das Programmattribut empfängt und für die übrigen Bytes Nullwerte festgelegt werden. Mögliche Werte: RELOAD, RESIDENT, REUSABLE und TRANSIENT.

RELOAD

Das Programm ist nicht wiederverwendbar. Es werden deshalb möglicherweise mehrere Kopien des Programms geladen. Wird ein Aufruf RELEASE_PROGRAM für eine Kopie abgesetzt, wird die betreffende Kopie aus dem Speicher entfernt.

RESIDENT

Es gibt eine einzige Kopie des Programms, die erst aus dem Speicher entfernt wird, wenn sie gelöscht wird. Als RESIDENT definierte Programme müssen zumindest quasiwiedereintrittsfähig sein. Programme des Typs PROGRAM_TYPE_SHARED verfügen standardmäßig über das Attribut RESIDENT. Bei diesem Typ von residenten Programmen hat der Aufruf DELETE_PROGRAM keine Auswirkungen.

REUSABLE

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut REUSABLE, das nicht verwendet wird, von CICS zur Speicheroptimierung aus dem Speicher entfernt werden kann.

TRANSIENT

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut TRANSIENT aus dem Speicher entfernt wird, sobald es nicht genutzt wird (Nutzungszähler gleich null).

PROGRAM_LENGTH(name4 | (Rn))

Gibt die Länge des angegebenen Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die in Byte angegebene Länge als Binärwert empfangen wird.

(Rn) Ein Register für den Binärwert mit der in Byte angegebenen Länge.

PROGRAM_NAME(name8 | string | "string")

Gibt den Namen des Programms an, zu dem Details angefordert werden.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

PROGRAM_TOKEN(name8),

Gibt ein Token zur Kennzeichnung des Programms an, zu dem Details angefordert werden.

name8 Der Name einer Position mit dem aus 8 Byte bestehenden Token, das über einen vorherigen Aufruf DEFINE_PROGRAM oder ACQUIRE_PROGRAM abgerufen wurde.

SUSPEND(NO|YES)

Gibt an, ob die Ausführung ausgesetzt werden soll, bis die Anforderung erfüllt werden kann.

RESPONSE- und REASON-Werte für ACQUIRE_PROGRAM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	NO_STORAGE
	PROGRAM_NOT_DEFINED
	PROGRAM_NOT_FOUND
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. Der REASON-Wert NO_STORAGE in Verbindung mit dem RESPONSE-Wert EXCEPTION bedeutet, dass der Speicher nicht zum Erfüllen der Anforderung ausreicht und SUSPEND(NO) angegeben war.
3. Der REASON-Wert PROGRAM_NOT_FOUND wird zurückgegeben, wenn das Programm nicht in die Bibliotheksverkettung aufgenommen wurde oder das Herstellen der Programmverbindung fehlgeschlagen ist. In diesem Fall muss für das als nicht ausführbar (not executable) gekennzeichnete Programm erneut eine Programmverbindung hergestellt werden, bevor Details zum Programm angefordert werden können.

Aufruf DEFINE_PROGRAM

Mit dem Aufruf DEFINE_PROGRAM können Sie neue Programme für die Ladedomäne definieren oder die Details bereits definierter Programme ändern. Die von Ihnen bereitgestellten Details werden im lokalen Katalog aufgezeichnet und sind sofort verfügbar. Sie werden für alle nachfolgenden ACQUIRE-Anforderungen für das angegebene Programm verwendet.

Mit dem Aufruf DEFINE_PROGRAM erstellte Programmdefinitionen werden bei einer XRF-Übernahme nicht beibehalten. Außerdem ist zu beachten, dass nur die Ladedomänendefinitionen aktualisiert werden, nicht die CICS-Systemdefinitionsdatei (CSD).

DEFINE_PROGRAM

```
DFHDLDX [CALL,]
        [CLEAR,]
        [IN,
        FUNCTION(DEFINE_PROGRAM),
        PROGRAM_NAME(name8 | string | 'string' ),
        [EXECUTION_KEY(CICS|USER),]
        [PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
        [PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY),]
        [REQUIRED_AMODE(24|31|AMODE_ANY|64),]
        [REQUIRED_RMODE(24|RMODE_ANY),]]
        [OUT,
        [NEW_PROGRAM_TOKEN(name8),]
        RESPONSE(name1 | *),
        REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

EXECUTION_KEY(CICS|USER)

Gibt in Verbindung mit anderen Programmattributen den Typ des dynamischen Speicherbereichs (Dynamic Storage Area, DSA) an, in den das Programm vom Ladeprogramm geladen werden soll.

CICS Bei nicht wiedereintrittsfähigen Programmen wird das Programm in einen dynamischen CICS-Speicherbereich ober- oder unterhalb der 16-MB-Grenze, d. h. in CDSA oder ECDSA, geladen. Die Auswahl des dynamischen CICS-Speicherbereichs richtet sich nach dem Residenzmodusattribut des Programms (RMODE), wie dem Verbindungseditor angegeben.

Bei wiedereintrittsfähigen RMODE(24)-Programmen wird das Programm in den dynamischen CICS-Speicherbereich (CDSA) geladen.

USER Bei nicht wiedereintrittsfähigen Programmen wird das Programm in einen dynamischen Benutzerspeicherbereich ober- oder unterhalb der 16-MB-Grenze, d. h. in UDSA oder EUDSA, geladen. Die Auswahl des dynamischen Benutzerspeicherbereichs richtet sich nach dem Residenzmodusattribut des Programms (RMODE), wie dem Verbindungseditor angegeben.

Bei wiedereintrittsfähigen RMODE(24)-Programmen wird das Programm in den dynamischen Benutzerspeicherbereich (UDSA) geladen.

Wiedereintrittsfähiges Programme, die oberhalb der 16-MB-Grenze geladen werden können: Ist ein Programm für den Verbindungseditor mit AMODE(31),RMODE(ANY) als wiedereintrittsfähiges Programm definiert, wird die Option EXECUTION_KEY ignoriert und das Programm wird in einen schreibgeschützten dynamischen Speicherbereich, d. h. RDSA oder ERDSA, geladen. Details zu dem Speichertyp, der für ERDSA zugeordnet ist, können Sie den Erläuterungen zum Systeminitialisierungsparameter RENTPGM entnehmen.

Eine Übersicht zu den Auswirkungen der Option EXECUTION_KEY in Verbindung mit anderen Faktoren finden Sie in Tabelle 5.

Tabelle 5. Übersicht über Attribute zur Definition der DSA-Eignung

Option EXECUTION_KEY	Wiedereintrittsfähig	Ober- oder unterhalb der 16-MB-Grenze	Dynamischer Speicherbereich (DSA)
CICS	Nein	Unterhalb	CDSA
CICS	Ja	Unterhalb	RDSA

Tabelle 5. Übersicht über Attribute zur Definition der DSA-Eignung (Forts.)

Option EXECUTION_KEY	Wiedereintrittsfähig	Ober- oder unter- halb der 16-MB- Grenze	Dynamischer Speicherbereich (DSA)
CICS	Nein	Oberhalb	ECDSA
CICS	Ja	Oberhalb	ERDSA
USER	Nein	Unterhalb	UDSA
USER	Ja	Unterhalb	RDSA
USER	Nein	Oberhalb	EUDSA
USER	Ja	Oberhalb	ERDSA

NEW_PROGRAM_TOKEN(name8)

Gibt das Token für das neu definierte Programm zurück.

name8 Der Name einer Position für das abgerufene, aus 8 Byte bestehende Token.

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)

Gibt den Residenzstatus des Programms an.

RELOAD

Jede ACQUIRE_PROGRAM-Anforderung für dieses Programm wird durch das Laden einer neuen Kopie in den Speicher erfüllt. Wird eine RELEASE-Anforderung für eine Programmkopie abgesetzt, wird die Kopie aus dem Speicher entfernt.

Anmerkung: Verwenden Sie dieses Attribut nicht, wenn Sie ein Exitprogramm definieren.

RESIDENT

Es gibt eine einzige Kopie des Programms, die erst aus dem Speicher entfernt wird, wenn sie gelöscht wird. Als RESIDENT definierte Programme müssen zumindest quasiwiedereintrittsfähig sein.

REUSABLE

Das Programm ist zumindest quasiwiedereintrittsfähig. Eine einzelne Kopie im Speicher kann von verschiedenen Tasks im System verwendet werden. Ein als REUSABLE definiertes Programm kann im Rahmen des normalen Komprimierungsschemas für dynamischen Programmspeicher gelöscht werden, wenn es nicht genutzt wird (Nutzungszähler gleich null).

TRANSIENT

Entspricht REUSABLE, abgesehen davon, dass das Programm sofort aus dem Speicher entfernt wird, wenn es nicht genutzt wird (Nutzungszähler gleich null). Geben Sie diese Option nur für selten verwendete Programme oder für Programme in Systemen an, bei denen ein kritischer Speichermangel vorliegt.

PROGRAM_NAME(name8 | string | "string")

Gibt den Namen des zu definierenden Programms an.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine als Programmname dienende Zeichenfolge ohne eingebettete Leerzeichen.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY)

Gibt an, wo das Programm geladen wird.

PRIVATE

Das Programm ist in der DFHRPL-Bibliotheksverkettung oder einer dynamischen Bibliotheksverkettung enthalten. Ein privates Programm muss nicht unbedingt wiedereintrittsfähig sein und erhält nur begrenzten Schutz gegen unbefugtes Überschreiben. Der Umfang des Schutzes richtet sich nach dem Typ des dynamischen Speicherbereichs (DSA), in den das Programm geladen wird (siehe Option EXECUTION_KEY):

DSA Schutz vor unbefugtem Überschreiben.

CDSA Kann nicht durch Benutzertasks überschrieben werden.

ECDSA

Kann nicht durch Benutzertasks überschrieben werden.

ERDSA

Vollständig: Kann nicht durch Benutzertasks oder CICS-Tasks überschrieben werden.

EUDSA

Kein Schutz.

RDSA Vollständig: Kann nicht durch Benutzertasks oder CICS-Tasks überschrieben werden.

UDSA Kein Schutz.

SHARED

Das Programm befindet sich im Link-Pack-Bereich (LPA), ist wiedereintrittsfähig und wird geschützt.

TYPE_ANY

Es kann die Kopie in der DFHRPL-Bibliotheksverkettung oder einer dynamischen Bibliotheksverkettung oder die LPA-Kopie des Programms verwendet werden. Bevorzugte Option: LPA-Kopie.

REQUIRED_AMODE(24|31|AMODE_ANY|64)

Gibt den Adressierungsmodus des Programms an. Wird bei der nachfolgenden ACQUIRE_PROGRAM-Verarbeitung festgestellt, dass keine Kopie des Programms gefunden werden kann, die die definierten Adressierungsanforderungen erfüllt, empfängt der Aufruf ACQUIRE_PROGRAM eine Ausnahmeantwort (EXCEPTION) und den Wert PROGRAM_NOT_FOUND für REASON.

Anmerkung:

1. AMODE_ANY und AMODE 31 haben bei dieser Funktion eine identische Bedeutung.
2. Sie können mit dieser Option nicht den für den Verbindungseditor angegebenen Adressierungsmodus des Programms überschreiben.

REQUIRED_RMODE(24|RMODE_ANY)

Gibt den Residenzmodus des Programms an. Wird bei der nachfolgenden ACQUIRE_PROGRAM-Verarbeitung festgestellt, dass keine Kopie des Programms gefunden werden kann, die die definierten Adressierungsanforderungen erfüllt,

empfängt der Aufruf ACQUIRE_PROGRAM eine Ausnahmeantwort (EXCEPTION) und den Wert PROGRAM_NOT_FOUND für REASON.

Anmerkung: Sie können mit dieser Option nicht den für den Verbindungseditor angegebenen Residenzmodus des Programms überschreiben.

RESPONSE- und REASON-Werte für DEFINE_PROGRAM

RESPONSE	REASON
OK	----
EXCEPTION	CATALOG_ERROR
	CATALOG_NOT_OPERATIONAL
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf DELETE_PROGRAM

Mit dem Aufruf DELETE_PROGRAM wird die Definition eines angegebenen Programms im Katalog und in der Liste der aktuellen Programme entfernt. Wird diese Anforderung erfolgreich ausgeführt, schlagen nachfolgende ACQUIRE_PROGRAM-Anforderungen mit dem REASON-Wert PROGRAM_NOT_DEFINED fehl.

DELETE_PROGRAM

```
DFHDLDX [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(DELETE_PROGRAM),  
    PROGRAM_NAME(name8 | string | 'string' ),]  
    [OUT,  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

PROGRAM_NAME(name8 | string | "string")

Gibt den Namen des zu löschenden Programms an.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

RESPONSE- und REASON-Werte für DELETE_PROGRAM

RESPONSE	REASON
OK	----
EXCEPTION	PROGRAM_NOT_DEFINED
DISASTER	----

<i>RESPONSE</i>	<i>REASON</i>
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf IDENTIFY_PROGRAM

Mit dem IDENTIFY_PROGRAM wird ein Programm anhand einer zugeordneten Adresse lokalisiert. Gehört die Adresse zu einem in CICS definierten Programm, werden vom Aufruf Informationen zu dem betreffenden Programm zurückgegeben.

Ist die Adresse nicht einem in der CICS-Ladedomäne definierten Programm zugeordnet, schlägt die Anforderung mit dem REASON-Wert INSTANCE_NOT_FOUND fehl.

IDENTIFY_PROGRAM

```
IDENTIFY_PROGRAM
DFHDLDX [CALL,]
    [CLEAR,]
    [IN,
    FUNCTION(IDENTIFY_PROGRAM),
    ADDRESS(name4 | (Rn) | *),]
    [OUT,
    [PROGRAM_NAME(name8 | *),]
    [PROGRAM_ATTRIBUTE(name1 | (Rn) | *),]
    [PROGRAM_LENGTH(name4 | (Rn) | *),]
    [LOAD_POINT(name4 | (Ra) | *),]
    [ENTRY_POINT(name4 | (Ra) | *),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ADDRESS(name4 | (Rn) | *)

Die zur Kennzeichnung des Programms verwendete Speicheradresse.

name4 Der Name eines aus 4 Byte bestehenden Vollworts mit der gespeicherten Speicheradresse.

(Rn) Ein Register, das mit der Speicheradresse definiert ist.

PROGRAM_NAME(name8 | *)

Gibt den Namen des Programms zurück, das die Speicheradresse enthält. PROGRAM_NAME entspricht dem in CICS definierten Programmnamen, nicht einem Programmabschnitt (CSECT).

name8 Der Name einer Position für den aus 8 Byte bestehenden Programmnamen.

PROGRAM_ATTRIBUTE(name1 | (Rn) | *)

Gibt das Programmattribut zurück.

name1 Der Name einer aus 1 Byte bestehenden Position für das zu empfangende Programmattribut.

(Rn) Ein Register, bei dem das niedrigstwertige Byte das Programmattribut

empfängt und für die übrigen Bytes Nullwerte festgelegt werden. Mögliche Registerwerte: RELOAD, RESIDENT, REUSABLE und TRANSIENT.

RELOAD

Das Programm ist nicht wiederverwendbar. Es werden deshalb möglicherweise mehrere Kopien des Programms geladen. Wird ein Aufruf RELEASE_PROGRAM für eine Kopie abgesetzt, wird die betreffende Kopie aus dem Speicher entfernt.

RESIDENT

Es gibt eine einzige Kopie des Programms, die erst aus dem Speicher entfernt wird, wenn sie gelöscht wird. Als RESIDENT definierte Programme müssen zumindest quasiwiedereintrittsfähig sein. Programme des Typs PROGRAM_TYPE_SHARED verfügen standardmäßig über das Attribut RESIDENT. Bei diesem Typ von residenten Programmen hat der Aufruf DELETE_PROGRAM keine Auswirkungen.

REUSABLE

Das Programm gleicht einem Programm mit dem Attribut RESIDENT, abgesehen davon, dass es von CICS zur Optimierung der Speichernutzung aus dem Speicher entfernt werden kann, wenn es nicht genutzt wird.

TRANSIENT

Das Programm gleicht einem Programm mit dem Attribut RESIDENT, abgesehen davon, dass das Programm aus dem Speicher entfernt wird, sobald es nicht genutzt wird (Nutzungszähler gleich null).

PROGRAM_LENGTH(name4 | (Rn) | *)

Gibt die Länge des angegebenen Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die in Byte angegebene Länge als Binärwert empfangen wird.

(Rn) Ein Register für den Binärwert mit der in Byte angegebenen Länge.

LOAD_POINT(name4 | (Ra) | *)

Gibt die Ladepunktadresse des Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende Ladeadresse.

(Ra) Ein Register für die Ladeadresse.

ENTRY_POINT(name4 | (Ra) | *)

Gibt die Einstiegspunktadresse des Programms zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende, aus 31 Bit bestehende Einstiegsadresse.

(Ra) Ein Register für die zu empfangende Einstiegsadresse.

RESPONSE- und REASON-Werte für IDENTIFY_PROGRAM

RESPONSE	REASON
OK	----
EXCEPTION	INSTANCE_NOT_FOUND
DISASTER	----
INVALID	----

RESPONSE	REASON
KERNERROR	----
PURGED	----

Anmerkung: Weitere Informationen finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf RELEASE_PROGRAM

Der Aufruf RELEASE_PROGRAM bewirkt, dass sich der Nutzungszähler für ein aktuell geladenes Programm um 1 verringert.

Wurde das Programm mit dem Attribut RELOAD definiert, wird der von der jeweiligen Kopie des Programms belegte Speicher freigegeben.

Setzen Sie die ACQUIRE_PROGRAM- und RELEASE_PROGRAM-Anforderungen für ein einzelnes Programm während derselben Ausführung des Exitprogramms ab. Wenn Sie das nicht möchten, fordern Sie stattdessen während der CICS-Initialisierung ein einziges Mal Details zum Programm an und halten Sie das Programm bis zur Beendigung von CICS als residentes Programm im Speicher zurück.

RELEASE_PROGRAM

```
DFHDLDX [CALL,]
        [CLEAR,]
        [IN,
        FUNCTION(RELEASE_PROGRAM),
        ENTRY_POINT(pointer),
        {PROGRAM_NAME(name8 | string | 'string')|
        PROGRAM_TOKEN(name8)},]
        [OUT,
        RESPONSE(name1 | *),
        REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ENTRY_POINT(pointer)

Gibt die Adresse des Einstiegspunkts für die Kopie des angegebenen Programms an.

PROGRAM_NAME(name8 | string | "string")

Gibt den Namen des freizugebenden Programms an.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

PROGRAM_TOKEN(name8),

Gibt ein Token zur Kennzeichnung des freizugebenden Programms an.

name8 Der Name einer Position mit dem aus 8 Byte bestehenden Token, das über einen vorherigen Aufruf DEFINE_PROGRAM oder ACQUIRE_PROGRAM abgerufen wurde.

RESPONSE- und REASON-Werte für RELEASE_PROGRAM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	PROGRAM_NOT_DEFINED
	PROGRAM_NOT_IN_USE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. PROGRAM_NOT_DEFINED wird zurückgegeben, wenn das von Ihnen angegebene Programm im System nicht bekannt ist.
3. PROGRAM_NOT_IN_USE wird zurückgegeben, wenn der Nutzungszähler für das angegebene Programm bereits den Wert '0' aufweist.

Kapitel 8. XPI-Funktionen für Protokollmanager

Die XPI stellt zwei Protokollmanagerfunktionen bereit. Dabei handelt es sich um die DFHLGPAX-Aufrufe INQUIRE_PARAMETERS und SET_PARAMETERS. Mit diesen Aufrufen können Sie Informationen zum Protokollmanagerparameter KEYPOINT_FREQUENCY abrufen und diesen Parameter festlegen. Der Parameter gibt die Häufigkeit der Aktivitätsschlüsselpunkte in der CICS-Region an.

Aufruf INQUIRE_PARAMETERS

Mit dem Aufruf INQUIRE_PARAMETERS werden Informationen zur Häufigkeit von Aktivitätsschlüsselpunkten im System zurückgegeben.

INQUIRE_PARAMETERS

```
DFHLGPAX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(INQUIRE_PARAMETERS),  
           [OUT,  
            KEYPOINT_FREQUENCY(name4 | *),]  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

KEYPOINT_FREQUENCY(name4 | *)

Gibt die Häufigkeit für das Setzen von Aktivitätsschlüsselpunkten in der CICS-Region zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für den zu empfangenden Häufigkeitswert.

RESPONSE- und REASON-Werte für INQUIRE_PARAMETERS

RESPONSE	REASON
OK	----
DISASTER	----
INVALID	----
KERNERROR	----

Aufruf SET_PARAMETERS

Mit dem Aufruf SET_PARAMETERS können Sie die Häufigkeit von Aktivitätsschlüsselpunkten für die CICS-Region festlegen.

SET_PARAMETERS

```
DFHLGPAX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(SET_PARAMETERS),  
           KEYPOINT_FREQUENCY(name4 | (Rn) ),]  
          [OUT,  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

KEYPOINT_FREQUENCY(name4 | *)

Gibt die Häufigkeit für das Setzen von Aktivitätsschlüsselpunkten in der CICS-Region an.

Zulässige Werte sind 0 und ganze Zahlen zwischen 200 und 65535 (einschließlich).

name4 Der Name einer aus 4 Byte bestehenden Position mit dem neuen Häufigkeitswert.

(Rn) Ein Register mit dem neuen Häufigkeitswert.

RESPONSE- und REASON-Werte für SET_PARAMETERS

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	OUT_OF_RANGE
DISASTER	----
INVALID	----
KERNERROR	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Kapitel 9. XPI-Funktionen für Überwachung

Die XPI stellt vier Überwachungsfunktionen bereit: die DFHMNMNX-Aufrufe INQUIRE_MONITORING_DATA und MONITOR, den DFHMNTDX-Aufruf SET_TRACKING_DATA sowie den DFHMNIAX-Aufruf INQUIRE_APP_CONTEXT.

Einschränkung:

DFHMNMNX-, DFHMNTDX- und DFHMNIAX-Aufrufe können in keinem Exitprogramm verwendet werden, das über einen globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Dispatcherdomäne
- Speicherauszugsdomäne
- Monitordomäne
- Statistikdomäne
- Programm für transiente Daten (TDP)

Darüber hinaus können die Aufrufe SET_TRACKING_DATA und INQUIRE_APP_CONTEXT in keinem Exitprogramm verwendet werden, das über einen globalen Benutzerexitpunkt in den folgenden Domänen aufgerufen wird:

- Transaktionsmanagerdomäne

Darüber hinaus ist zu beachten, dass die Aufrufe INQUIRE_APP_CONTEXT, INQUIRE_MONITORING_DATA und SET_TRACKING_DATA in keinem Exitprogramm verwendet werden können, das über einen globalen Benutzerexitpunkt in DFHTCP oder DFHZCP aufgerufen wird.

Aufruf INQUIRE_APP_CONTEXT

Mit dem Aufruf INQUIRE_APP_CONTEXT werden die Anwendungskontextdaten zu der aktuellen Anwendung mit der Task, von der der Aufruf abgesetzt wird, zum Exitprogramm zurückgegeben.

Beschränkung

Starten Sie Exitprogramme, die die Funktion INQUIRE_APP_CONTEXT verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt Writing initialization and shutdown programs.

INQUIRE_APP_CONTEXT

```
DFHMNIAX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(INQUIRE_APP_CONTEXT),  
           CONTEXT(INITIAL | CURRENT)]  
          [OUT,  
           [APPLNAME(name64),]  
           [PLATNAME(name64),]  
           [OPERNAME(name64),]  
           [MAJORVER(name4 | (Rn)),]
```

```

[MINORVER(name4 | (Rn)),]
[MICROVER(name4 | (Rn)),]
RESPONSE(name1 | *),
REASON(name1 | *)]

```

Dieser Befehl ist threadsicher.

APPLNAME(name64)

Gibt den aus 64 Byte bestehenden Namen der Anwendung zurück, die der Task zugeordnet ist.

CONTEXT(INITIAL | CURRENT)

Der Parameter **CONTEXT(INITIAL | CURRENT)** legt fest, ob die zurückgegebenen Anwendungskontextwerte zu dem ursprünglichen Kontext oder dem aktuellen Kontext der Task gehören. Ist der Parameter **CONTEXT** nicht angegeben, wird standardmäßig der aktuelle Anwendungskontext der Task zurückgegeben.

MAJORVER(name4 | (Rn))

Gibt die Hauptversionsnummer der Anwendung zurück, die der Task zugeordnet ist.

name4 Der Name eines aus 4 Byte bestehenden Felds, das die Hauptversionsnummer in Form eines Binärwerts enthält.

(Rn) Ein Register für die zu empfangende Hauptversionsnummer.

MICROVER(name4 | (Rn))

Gibt die Mikroversionsnummer der Anwendung zurück, die der Task zugeordnet ist.

name4 Der Name eines aus 4 Byte bestehenden Felds, das die Mikroversionsnummer in Form eines Binärwerts enthält.

(Rn) Ein Register für die zu empfangende Mikroversionsnummer.

MINORVER(name4 | (Rn))

Gibt die Nebenversionsnummer der Anwendung zurück, die der Task zugeordnet ist.

name4 Der Name eines aus 4 Byte bestehenden Felds, das die Nebenversionsnummer in Form eines Binärwerts enthält.

(Rn) Ein Register für die zu empfangende Nebenversionsnummer.

OPERNAME(name64)

Gibt den aus 64 Byte bestehenden Namen der Operation zurück, die der Task zugeordnet ist.

PLATNAME(name64)

Gibt den aus 64 Byte bestehenden Namen der Plattform zurück, die der Task zugeordnet ist.

RESPONSE- und REASON-Werte für INQUIRE_APP_CONTEXT

RESPONSE	REASON
OK	----
EXCEPTION	APP_CONTEXT_UNAVAILABLE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Weitere Informationen finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf INQUIRE_MONITORING_DATA

Mit dem Aufruf INQUIRE_MONITORING_DATA werden die Überwachungsdaten der Leistungsklasse, die für die Task, von der der Aufruf abgesetzt wird, aufgelaufen sind, an das Exitprogramm zurückgegeben.

Der DFHMNTDS-Pseudoabschnitt (DSECT), über den die Daten zugeordnet werden, hat ein festes Format. Beachten Sie Folgendes:

- Es werden *alle* systemdefinierten CICS-Felder in den Leistungsdatensätzen (einschließlich der Felder, die Sie mithilfe der Option EXCLUDE des Makros DFHMCT TYPE=RECORD als auszuschließende Felder angegeben haben) aufgelistet.
- Es werden *keine* benutzerdefinierten Datenfelder aufgelistet.

INQUIRE_MONITORING_DATA

```
DFHMNMNX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(INQUIRE_MONITORING_DATA),  
           DATA_BUFFER(buffer-descriptor),]  
          [OUT,  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Wichtig

Bei der Verwendung der XPI in einem frühen Stadium der Initialisierung ist eine Einschränkung zu beachten. Starten Sie Exitprogramme, die die XPI-Funktionen TRANSACTION_DUMP, WRITE_JOURNAL_DATA, MONITOR und INQUIRE_MONITOR_DATA verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt Writing initialization and shutdown programs.

DATA_BUFFER(buffer-descriptor)

Gibt Adresse und Länge eines Puffers für die zurückgegebenen Überwachungsdaten an. Eine vollständige Definition eines Pufferdeskriptors finden Sie im Abschnitt XPI syntax. Die Überwachungsdaten werden über den DFHMNTDS-Pseudoabschnitt zugeordnet.

RESPONSE- und REASON-Werte für INQUIRE_MONITORING_DATA

RESPONSE	REASON
OK	----
EXCEPTION	LENGTH_ERROR
	MONITOR_DATA_UNAVAILABLE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. 'LENGTH_ERROR' bedeutet, dass die im Pufferdeskriptor angegebene Länge zu kurz für die Überwachungsdaten war, die vom XPI-Aufruf zurückgegeben wurden.

Aufruf MONITOR

Der XPI-Aufruf MONITOR entspricht dem Befehl **EXEC CICS MONITOR**. Dieser Aufruf ermöglicht es Ihnen, benutzerdefinierte EMPs (Event-Monitoring Points, Ereignisüberwachungspunkte) in Ihrem Exitprogramm aufzurufen.

Die benutzerdefinierten EMPs müssen in der Überwachungssteuertabelle (Monitoring Control Table, MCT) mit dem Makro DFHMCT TYPE=EMP definiert oder über den Parameter **APPLNAME** für das Makro DFHMCT TYPE=INITIAL generiert werden. Weitere Informationen zur CICS-Überwachung finden Sie im Abschnitt CICS monitoring facility: Performance and tuning .

An einem benutzerdefinierten EMP können Sie eigene Daten (bis zu 256 Zähler oder Taktgeber und eine einzelne Zeichenfolge mit einer Länge von bis zu 256 Byte) zu Feldern in Datensätzen für Überwachungsdaten der Leistungsklasse hinzufügen, die ohne weitere Bedingungen für Sie reserviert sind. Sie können an den DFHAPPL-EMPs auch bis zu 12 Byte an Benutzerinformationen hinzufügen.

MONITOR

```
DFHMNMNX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(MONITOR),
          POINT(expression | name2 | (Rn)),
          [DATA1(expression | name4 | (Ra) | *),]
          [DATA2(expression | name4 | (Ra) | *),]
          [ENTRYNAME(name8 | string | 'string'),]]
          [OUT,
          RESPONSE(name1 | *),
          REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Wichtig

Bei der Verwendung der XPI in einem frühen Stadium der Initialisierung ist eine Einschränkung zu beachten. Starten Sie Exitprogramme, die die XPI-Funktionen TRANSACTION_DUMP, WRITE_JOURNAL_DATA, MONITOR und INQUIRE_MONITOR_DATA verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt Writing initialization and shutdown programs.

DATA1(expression | name4 | (Ra) | *)

Gibt eine Vollwort-Binärvariable an, deren Inhalt vom Typ des verwendeten benutzerdefinierten EMPs abhängt:

- Enthält die benutzerdefinierte EMP-Definition der Überwachungssteuertabelle die Option ADDCNT, SUBCNT, NACNT, EXCNT oder ORCNT, stellt die Variable DATA1 einen Bereich dar, der wie in der benutzerdefinierten EMP-Definition festgelegt verwendet wird.

- Enthält die benutzerdefinierte EMP-Definition der Überwachungssteuertabelle die Option MLTCNT, stellt die Variable DATA1 einen Bereich mit der Adresse einer Reihe angrenzender Vollworte mit den Werten dar, die zu den in der benutzerdefinierten EMP-Definition definierten Benutzeranzahlfeldern hinzuzufügen sind.
- Enthält die benutzerdefinierte EMP-Definition die Option MOVE, stellt die Variable DATA1 einen Bereich mit der Adresse der zu versetzenden Zeichenfolge dar. Diese Regel gilt auch für die DFHAPPL-EMPs.

Nähere Angaben zu den Optionen in benutzerdefinierten EMPs finden Sie im Abschnitt Monitoring control table (MCT).

expression

Ein gültiger Assemblerausdruck, der die Vollwort-Binärvariable für den EMP angibt.

name4 Ein aus 4 Byte bestehendes Feld, das die Vollwort-Binärvariable für den EMP enthält.

(Ra) Ein Register mit der Vollwort-Binärvariable für den EMP.

***** Der Wert für diese Option ist bereits in der Parameterliste enthalten oder die Option ist nicht für den EMP angegeben.

DATA2(expression | name4 | (Rn) | *)

Gibt eine Vollwort-Binärvariable an, deren Inhalt vom Typ des verwendeten benutzerdefinierten EMPs abhängt:

- Enthält die benutzerdefinierte EMP-Definition der Überwachungssteuertabelle die Option ADDCNT, SUBCNT, NACNT, EXCNT oder ORCNT, stellt die Variable DATA2 einen Bereich dar, der wie in der benutzerdefinierten EMP-Definition festgelegt verwendet wird.
- Enthält die benutzerdefinierte EMP-Definition die Option MLTCNT, stellt die Variable DATA2 einen Bereich mit der Anzahl der zu aktualisierenden Benutzeranzahlfelder dar.

Die in DATA2 angegebene Anzahl überschreibt den in der Überwachungssteuertabelle definierten Standardwert für die Operation. Der Wert '0' weist die Überwachung an, den Standardwert zu verwenden. Die MLTCNT-Operation kann auch ohne Angabe eines Werts für DATA2 erfolgreich ausgeführt werden. Nach einer erfolgreichen Ausführung wird in diesem Fall jedoch die Ausnahmeantwort DATA2_NOT_SPECIFIED zurückgegeben. Siehe Anmerkung 5.

- Enthält die benutzerdefinierte EMP-Definition die Option MOVE, stellt die Variable DATA2 einen Bereich mit der Länge der zu versetzenden Zeichenfolge dar.

Die in DATA2 angegebene Länge überschreibt den in der Überwachungssteuertabelle definierten Standardwert für die Operation. Der Wert '0' weist die Überwachung an, den Standardwert zu verwenden. Die MOVE-Operation kann auch ohne Angabe eines Werts für DATA2 erfolgreich ausgeführt werden. Nach einer erfolgreichen Ausführung wird in diesem Fall jedoch die Ausnahmeantwort DATA2_NOT_SPECIFIED zurückgegeben. Siehe Anmerkung 5.

Nähere Angaben zu den Optionen in benutzerdefinierten EMPs finden Sie im Abschnitt Monitoring control table (MCT).

expression

Ein gültiger Assemblerausdruck, der die Vollwort-Binärvariable für den EMP angibt.

name4 Ein aus 4 Byte bestehendes Feld, das die Vollwort-Binärvariable für den EMP enthält.

(Rn) Ein Register mit der Vollwort-Binärvariable für den EMP.

***** Der Wert für diese Option ist bereits in der Parameterliste enthalten oder die Option ist nicht für den EMP angegeben.

ENTRYNAME(name8 | string | "string")

Gibt den Namen des Überwachungspunkteintrags an, der den Wert für POINT näher bestimmt und in der Überwachungssteuertabelle definiert ist.

name8 Der Name einer Position, die eine aus 8 Byte bestehende Zeichenfolge enthält.

string Eine Zeichenfolge ohne eingebettete Leerzeichen. Aus der Zeichenfolge wird vom Makro eine aus 8 Byte bestehende Literalkonstante generiert. Die Zeichenfolge wird dazu je nach Bedarf abgeschnitten oder mit Leerzeichen verlängert.

"string"

Eine in Anführungszeichen gesetzte Zeichenfolge, die möglicherweise Leerzeichen enthält. Dieser Wert wird auf die gleiche Weise verarbeitet wie die vorangehende Zeichenfolge ("string").

Anmerkung: Wenn Sie bei der Definition des EMPs in der Überwachungssteuertabelle keinen Eintragsnamen angeben, nimmt der Eintragsname standardmäßig den Wert 'USER' an. ENTRYNAME nimmt ebenfalls standardmäßig den Wert 'USER' an, wenn die Variable nicht angegeben ist.

POINT(expression | name2 | (Rn))

Gibt die in der Überwachungssteuertabelle definierte Kennung des Überwachungspunkts an, die im Bereich von 0 bis 255 liegt. Überwachungspunktkennungen im Bereich von 200 bis 255 sind für IBM® Programmprodukte reserviert.

expression

Ein gültiger Assemblerausdruck, der in 2 Byte ausgedrückt werden kann.

name2 Der Name einer aus 2 Byte bestehenden Quelle für Überwachungspunktdateien.

(Rn) Ein Register, das die Überwachungspunktdateien in den 2 niedrigstwertigen Byte enthält.

RESPONSE- und REASON-Werte für MONITOR

RESPONSE	REASON
OK	----
EXCEPTION	DATA1_NOT_SPECIFIED DATA2_NOT_SPECIFIED POINT_NOT_DEFINED INVALID_DATA1_VALUE INVALID_DATA2_VALUE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Nähere Informationen finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. POINT_NOT_DEFINED bedeutet, dass der von Ihnen angegebene EMP nicht in der Überwachungssteuertabelle definiert wurde.
3. INVALID_DATA1_VALUE und INVALID_DATA2_VALUE werden in der Regel durch die Angabe ungültiger Adressen verursacht. Dies führt zu einem Programmfehler.
4. DATA1_NOT_SPECIFIED bzw. DATA2_NOT_SPECIFIED bedeutet, dass Sie DATA1 bzw. DATA2 nicht angegeben haben, obwohl die Variable für die Operation erforderlich war. Siehe dazu die Beschreibung zu DATA2.
5. Bei allen Fehlerantworten wird die Verarbeitung des EMPs beendet. Operationen, deren Ausführung vor dem Fehlerpunkt vorgesehen war, werden ausgeführt, spätere Operationen werden abgebrochen.

Aufruf SET_TRACKING_DATA

Der Funktionsaufruf SET_TRACKING_DATA legt den Tag für die Ursprungsdaten der Transaktionsüberwachung für die Task, von der der Aufruf abgesetzt wird, fest.

SET_TRACKING_DATA

```
DFHMNTDX [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(SET_TRACKING_DATA),  
    {TRACKING_TAG (MOBILE)|TRACKING_TAG_VALUE(name1 | *)},]  
    [OUT,  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Wichtig

Bei der Verwendung der XPI in einem frühen Stadium der Initialisierung ist eine Einschränkung zu beachten. Starten Sie Exitprogramme, die die Funktion SET_TRACKING_DATA verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt Writing initialization and shutdown programs.

TRACKING_TAG (MOBILE)

Gibt an, dass die Taginformationen für die Ursprungsdaten der Transaktionsüberwachung in den Ursprungsdaten der Transaktionsüberwachung festgelegt werden sollen.

MOBILE

Der Tag für die Ursprungsdaten der Transaktionsüberwachung ist für mobile Geräte geeignet.

RESPONSE- und REASON-Werte für SET_TRACKING_DATA

RESPONSE	REASON
OK	----
EXCEPTION	NO_ASSOCIATION_DATA

RESPONSE	REASON
	TRACKING_TAG_ALREADY_SET
	INVALID_TRACKING_TAG
	INVALID_TRACKING_TAG_VALUE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. NO_ASSOCIATION_DATA bedeutet, dass die Transaktion, unter der dieser XPI-Aufruf aufgerufen wurde, keine Zuordnungsdaten zur Transaktionsüberwachung aufweist. Weitere Informationen zu Zuordnungsdaten und Transaktionsüberwachung finden Sie im Abschnitt Introduction to CICS intercommunication.
3. TRACKING_TAG_ALREADY_SET bedeutet, dass der Tag für die Ursprungsdaten der Transaktionsüberwachung für die Task, von der der Aufruf abgesetzt wird, bereits festgelegt wurde.
4. INVALID_TRACKING_TAG bedeutet, dass der Tag für die Ursprungsdaten der Transaktionsüberwachung einen ungültigen Wert aufweist.
5. INVALID_TRACKING_TAG_VALUE bedeutet, dass der Wert des Tags für die Ursprungsdaten der Transaktionsüberwachung nicht im Bereich von 129 bis 255 liegt.

Kapitel 10. XPI-Funktionen für Objekttransaktionen

Mit den XPI-Aufrufen für Objekttransaktionen können Sie ein TRUE-Programm implementieren, das auf Aufrufe zwischen einer CICS-Arbeitseinheit und einem fernen Transaktionskoordinator antwortet. Dabei handelt es sich um die DFHOTTRX-Aufrufe COMMIT, COMMIT_ONE_PHASE, IMPORT_TRAN, PREPARE, ROLLBACK und SET_ROLLBACK_ONLY sowie den DFHOTCOX-Aufruf SET_COORDINATOR. Diese Funktionen ermöglichen eine leistungsfähige Steuerung der Synchronisationspunktverarbeitung einer CICS-Arbeitseinheit. Werden diese Aufrufe nicht ordnungsgemäß verwendet, wird CICS sofort vom CICS-Wiederherstellungsmanager beendet und es wird eine Wiederherstellungs- und Resynchronisationsverarbeitung benötigt.

Aufruf IMPORT_TRAN

Verknüpft die aktuelle Arbeitseinheit einer Task mit einer externen Transaktion. Verschiedene Informationen zu der externen Transaktion werden in der aktuellen Arbeitseinheit aufgezeichnet.

IMPORT_TRAN

```
DFHOTTRX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(IMPORT_TRAN),  
           [FORMAT_ID,(name4|Rn),]  
           [BQUAL_LEN,(name4|Rn),]  
           [TID_BLOCK_IN,(block-descriptor),]  
           [TIMEOUT,(name4|Rn),]  
           [LOGICAL_SERVER,(name4|string|'string'),]  
           [PUBLIC_ID,(name64|string|'string'),]  
          [OUT,  
           UOW_ID,(name8 | *),  
           RESPONSE (name1 | *),  
           REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

BQUAL_LEN (name4 | Rn)

Gibt die Länge des Verzweigungsqualifikationsmerkmals der OTS-Transaktions-ID (TID) an.

name4 Der Name einer aus 4 Byte bestehenden Position, die die Länge des Verzweigungsqualifikationsmerkmals enthält.

Rn Ein Register mit der Länge des Verzweigungsqualifikationsmerkmals.

FORMAT_ID (name4 | Rn)

Gibt die Kennung des OTS-Transaktionsformats an.

name4 Der Name einer aus 4 Byte bestehenden Position, die die Format-ID enthält.

Rn Ein Register mit der Format-ID.

LOGICAL_SERVER (name4 | string | 'string')

Gibt den Namen des logischen Servers an, in dem die Transaktion ausgeführt wird. Bei dieser XPI-Funktion wird empfohlen, nur Werte zu verwenden, die keiner im CICS-System installierten CorbaServer-Definition gleichen.

PUBLIC_ID (name64 | string | 'string')

Gibt die der Transaktion zugeordnete öffentliche ID an.

TID_BLOCK_IN (block-descriptor)

Gibt die eindeutige OTS-Transaktions-ID (TID) der externen Transaktion an,

die der Arbeitseinheit der Task zugeordnet werden soll. Bei dem Blockdeskriptor handelt es sich um zwei Vollworte für Daten. Das erste Vollwort enthält die Adresse der Daten, das zweite die Länge der Daten in Byte.

TIMEOUT (name4 | Rn)

Gibt das Zeitlimit für die OTS-Transaktion in Sekunden an.

name4 Der Name einer aus 4 Byte bestehenden Position, die das Zeitlimit enthält.

Rn Ein Register mit dem Zeitlimit.

UOW_ID (name8 | *)

Gibt die ID der CICS-Arbeitseinheit an, in die die OTS-Transaktion importiert wurde.

RESPONSE- und REASON-Werte für IMPORT_TRAN

RESPONSE	REASON
OK	----
EXCEPTION	TID_TOO_LONG
	OTS_TRAN_ALREADY
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf COMMIT_ONE_PHASE

Führt einen Synchronisationspunkt für die Arbeitseinheit der aktuellen Task aus, ohne auf einen externen Koordinator zu verweisen. Sie können den Aufruf COMMIT_ONE_PHASE nicht verwenden, wenn Sie mit dem Aufruf SET_COORDINATOR Koordinatorinformationen zu der Arbeitseinheit der aktuellen Task hinzugefügt haben.

COMMIT_ONE_PHASE

```
DFHOTTRX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(COMMIT_ONE_PHASE),]]
          [OUT,
          STATUS (name1 | *),
          RESPONSE (name1 | *),
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

STATUS (name1 | *)

Das Ergebnis der CICS-Arbeitseinheit.

Mögliche Werte für diesen Parameter:

COMMITTED
ROLLEDBACK

RESPONSE- und REASON-Werte für COMMIT_ONE_PHASE

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----

RESPONSE
PURGED

REASON

Aufruf PREPARE

Führt für eine OTS-Transaktion die erste Phase des Synchronisationspunkts bei der CICS-Arbeitseinheit durch. Die von dieser Funktion zurückgegebene Entscheidung soll es dem Koordinator der OTS-Transaktion ermöglichen, das Ergebnis der Gesamttransaktion zu bestimmen. Die CICS-Arbeitseinheit wechselt bei Aufruf von PREPARE in einen unbestätigten Status (*indoubt*) und behält diesen Status bis zu einem nachfolgenden Aufruf der Funktion COMMIT oder ROLLBACK bei. Schlägt das CICS-System fehl und erfordert einen Neustart, wird die CICS-Arbeitseinheit über das Systemprotokoll wiederhergestellt und Sie müssen eine Resynchronisation durchführen, um den Status der unbestätigten Arbeitseinheit zu ändern.

PREPARE

```
DFHOTTRX [CALL,]  
          [CLEAR,]  
          [IN,  
          FUNCTION(PREPARE),]  
          [OUT,  
          VOTE (name1 | *),  
          RESPONSE (name1 | *),  
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

VOTE (name1 | *)

Das Ergebnis der ersten Phase der OTS-Transaktion.

Mögliche Werte für diesen Parameter:

YES
NO
READ_ONLY
HEURISTIC_MIXED

RESPONSE- und REASON-Werte für PREPARE

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf COMMIT

Führt die zweite Phase des Synchronisationspunkts einer OTS-Transaktion aus, um das Festschreiben der Transaktion sicherzustellen.

COMMIT

```
DFHOTTRX [CALL,]  
          [CLEAR,]  
          [IN,
```

```

FUNCTION(COMMIT),]
[OUT,
RESPONSE (name1 | *),
REASON (name1 | *)]

```

Dieser Befehl ist threadsicher.

RESPONSE- und REASON-Werte für COMMIT

RESPONSE	REASON
OK	----
EXCEPTION	UOW_ROLLEDBACK
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf ROLLBACK

Macht eine OTS-Transaktion rückgängig.

ROLLBACK

```

DFHOTTRX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(ROLLBACK),]
          [OUT,
          RESPONSE (name1 | *),
          REASON (name1 | *)]

```

Dieser Befehl ist threadsicher.

RESPONSE- und REASON-Werte für ROLLBACK

RESPONSE	REASON
OK	----
EXCEPTION	UOW_COMMITTED
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf SET_ROLLBACK_ONLY

Markiert die CICS-Arbeitseinheit, sodass der OTS-Koordinator beim Ausführen eines Synchronisationspunktprotokolls eine Entscheidung (NO) erhält, die dazu führt, dass ein Rollback der globalen Transaktion erzwungen wird. Die CICS-Arbeitseinheit verbleibt weiterhin im Status einer aktuell ausgeführten Arbeitseinheit, für alle nachfolgenden wiederherstellbaren Ressourcenaktualisierungen wird jedoch mit dem Rest der globalen Transaktion ein Rollback durchgeführt.

SET_ROLLBACK_ONLY

```

DFHOTTRX [CALL,]
          [CLEAR,]
          [IN,

```

```

FUNCTION(SET_ROLLBACK_ONLY),]
[OUT,
RESPONSE (name1 | *),
REASON (name1 | *)]

```

Dieser Befehl ist threadsicher.

RESPONSE- und REASON-Werte für SET_ROLLBACK_ONLY

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf SET_COORDINATOR

Ordnet der Arbeitseinheit der aktuellen Task eine Verknüpfung zu, um einen fernen Koordinator anzugeben.

SET_COORDINATOR

Stellen Sie sicher, dass die XPI-Objekttransaktionsfunktionen für Phase 1 (OTTR_PREPARE) und Phase 2 (OTTR_COMMIT oder OTTR_ROLLBACK) dazu verwendet werden, um die aktuelle Arbeitseinheit ausgerichtet an den Aktionen des fernen Koordinators durch die Synchronisationspunktverarbeitung zu steuern. Sie können die Funktion OTTR_COMMIT_ONE_PHASE nicht verwenden, wenn Sie mit der Funktion SET_COORDINATOR Informationen zu einem Koordinator zu der Arbeitseinheit der aktuellen Task hinzugefügt haben, und Sie müssen verhindern, dass die Task beendet wird, ohne die richtigen XPI-Objekttransaktionsfunktionen für die Synchronisationspunktverarbeitung für die aktuelle Arbeitseinheit zu verwenden.

```

DFHOTCOX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(SET_COORDINATOR),
          [IOR_BLOCK,(block-descriptor)]
          [HOST_BLOCK,(block-descriptor)]]
          [OUT,
          COORDINATOR_TOKEN,(name1 | *),
          RESPONSE (name1 | *),
          REASON (name1 | *)]

```

Dieser Befehl ist threadsicher.

HOST_BLOCK (block-descriptor)

Ein Verweis auf eine Zeichenfolge, die die Kennung des Systems darstellt, das die Koordinatorinstanz enthält, sowie die Länge der Zeichenfolge. Bei diesem Parameter wird eine maximale Länge von 4096 Byte unterstützt.

IOR_BLOCK (block-descriptor)

Ein Verweis auf eine Zeichenfolge, die die Koordinatorinstanz im Hostsystem darstellt, sowie die Länge der Zeichenfolge. Bei diesem Parameter wird eine maximale Länge von 4096 Byte unterstützt.

COORDINATOR_TOKEN (name1 | *)

Ein Token, das den Koordinator darstellt.

RESPONSE- und REASON-Werte für SET_COORDINATOR

RESPONSE	REASON
OK	----
EXCEPTION	IOR_TOO_LONG
	HOST_TOO_LONG
	LINK_UNKNOWN
	COORDINATOR_NOT_FOUND
	COORDINATOR_ALREADY
	INVALID_SYNCPOINT_STATE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Kapitel 11. XPI-Funktion für Parameterdomäne

Die XPI stellt eine Parameterdomänenfunktion bereit. Dabei handelt es sich um den DFHPAIQX-Aufruf INQUIRE_FEATUREKEY für Funktionsumschaltungen.

Aufruf INQUIRE_FEATUREKEY

Mit INQUIRE_FEATUREKEY wird der Wert für einen Feature-Umschalter abgerufen.

Eine nach CICS-Release geordnete Liste der umschaltbaren Features finden Sie in Toggle-enabled features, support by release. Rufen Sie mithilfe der Links in der Tabelle mit der Featureliste Beschreibungen zu den Feature-Umschaltern für das Aktivieren und Festlegen von Konfigurationsoptionen für ein bestimmtes umschaltbares Feature auf.

INQUIRE_FEATUREKEY

```
DFHPAIQX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(INQUIRE_FEATUREKEY),  
           OPTION(name255|'string'),  
           [STRING(buffer-descriptor),]  
           [UPPERCASE (YES|NO),]  
           [OUT,  
            [NUMBER(name4),]  
            [BOOLEAN(name1)]]  
          RESPONSE(name1 | *),  
          REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

OPTION(name255|'string')

Gibt den Namen des Feature-Umschalters an.

STRING(buffer-descriptor)

Gibt Adresse und Länge eines Puffers für den zurückgegebenen Zeichenfolge-wert des Umschalters an. Eine vollständige Definition eines Pufferdeskriptors finden Sie in XPI syntax.

UPPERCASE (YES|NO)

Gibt an, ob der zurückgegebene Wert in STRING in Großbuchstaben geändert wird.

NUMBER(name4)

Gibt einen numerischen Umschaltwert im Binärformat zurück.

BOOLEAN(name1)

Gibt einen booleschen Umschaltwert zurück.

TRUE Der Wert lautet 'true'.

FALSE

Der Wert lautet 'false'.

RESPONSE- und REASON-Werte für INQUIRE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	NOT_FOUND
	BUFFER_TOO_SMALL
	BAD_OPTION
	NOT_BOOLEAN
	NOT_NUMBER
	NO_LIST
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Kapitel 12. XPI-Funktionen für Programmverwaltung

Die XPI stellt einschließlich der Aufrufe DFHPGISX, DFHPGAQX und DFHPGCHX acht Programmverwaltungsfunktionen bereit. In Verbindung mit den Ladeprogrammfunktionen stellen diese Funktionen eine umfassende Gruppe von Tools für die Bearbeitung von Programmen dar.

Zu den Programmverwaltungsfunktionen gehören die folgenden DFHPGISX-Aufrufe:

- END_BROWSE_PROGRAM
- GET_NEXT_PROGRAM
- INQUIRE_CURRENT_PROGRAM
- INQUIRE_PROGRAM
- SET_PROGRAM
- START_BROWSE_PROGRAM

Zu den Programmverwaltungsfunktionen gehören die folgenden DFHPGAQX-Aufrufe:

- INQUIRE_AUTOINSTALL
- SET_AUTOINSTALL

Zu den Programmverwaltungsfunktionen gehört der folgende DFHPGCHX-Aufruf:

- BIND_CHANNEL

Sie können diese Funktionen in Verbindung mit den Ladeprogrammfunktionen (DFHLDLDX-Aufrufe) zur Bearbeitung von Programmen einsetzen. Die in den NEW_PROGRAM_TOKEN-Feldern der DFHPGISX-Aufrufe zurückgegebenen Token unterscheiden sich von den Token, die von DFHLDLDX-Aufrufen zurückgegeben werden. Verwenden Sie kein Token in einem Aufruf DFHLDLDX, das Sie über einen Aufruf DFHPGISX erhalten haben, und umgekehrt.

Aufruf INQUIRE_PROGRAM

INQUIRE_PROGRAM gibt Informationen zu den Attributen eines angegebenen Programms zurück.

INQUIRE_PROGRAM

```
DFHPGISX [CALL,]  
    [CLEAR,]  
    [IN,  
    FUNCTION(INQUIRE_PROGRAM),  
    [AC_APPLICATION_NAME(block-descriptor),]  
    [AC_MAJOR_VERSION(name4),]  
    [AC_MICRO_VERSION(name4),]  
    [AC_MINOR_VERSION(name4),]  
    [AC_PLATFORM_NAME(block-descriptor),]  
    [{PROGRAM_NAME(name8 | string | 'string')|  
    PROGRAM_TOKEN(name4)},],  
    [SHOW_PROGRAMS(PRIVATE|PRIVATE_AND_PUBLIC),]  
    [OUT,  
    [ACCESS(CICS|NONE|READ_ONLY|USER),]  
    [APIST(CICSAPI|OPENAPI),]  
    [AVAIL_STATUS(DISABLED|ENABLED),]
```

```

[CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
[CONCURRENCY(QUASIRENT|THREADSAFE),]
[DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
[DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC),]
[ENTRY_POINT(name4),]
[EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
[EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
[HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
[INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
[LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                  COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
[LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                  LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
[LIBRARY(name8),]
[LIBRARYDSN(name44),]
[LOAD_POINT(name4),]
[LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
[LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA),]
[MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
[NEW_PROGRAM_TOKEN(name4),]
[PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
[PROGRAM_LENGTH(name4),]
[PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY),]
[PROGRAM_USAGE(APPLICATION|NUCLEUS),]
[PROGRAM_USE_COUNT(name4),]
[PROGRAM_USER_COUNT(name4),]
[REMOTE_DEFINITION(LOCAL|REMOTE),]
[REMOTE_PROGID(name8),]
[REMOTE_SYSID(name4),]
[REMOTE_TRANID(name4),]
[SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64),]
[SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED),]
RESPONSE(name1 | *),
REASON(name1 | *)]

```

Dieser Befehl ist threadsicher.

AC_APPLICATION_NAME(block-descriptor)

Gibt Adresse und Länge des Namens der Anwendung an, die dem Programm zugeordnet ist. Sollen Informationen zu privaten Programmen für auf Plattformen bereitgestellten Anwendungen angefragt werden, müssen Sie die Felder AC_APPLICATION_NAME, AC_MAJOR_VERSION, AC_MINOR_VERSION, AC_MICRO_VERSION und AC_PLATFORM_NAME bereitstellen, um einen vollständigen Anwendungskontext bereitzustellen. Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

AC_MAJOR_VERSION(name4)

Gibt die Hauptversion der Anwendung im Binärformat an.

AC_MICRO_VERSION(name4)

Gibt die Mikroversion der Anwendung im Binärformat an.

AC_MINOR_VERSION(name4)

Gibt die Nebenversion der Anwendung im Binärformat an.

AC_PLATFORM_NAME(block-descriptor)

Gibt Adresse und Länge des Namens der Plattform an, die dem Programm zugeordnet ist. Weitere Informationen zu Blockdeskriptoren finden Sie im Abschnitt XPI syntax.

ACCESS(CICS|NONE|READ_ONLY|USER)

Gibt einen Wert für den Typ des Speichers zurück, in den das Programm geladen wurde.

CICS CICS-Schlüssel.

NONE

Das Programm wurde nicht geladen.

READ_ONLY

Schreibgeschützt.

USER Benutzerschlüssel.

APIST(CICSAPI|OPENAPI)

Gibt einen Wert für das API-Attribut der installierten Programmdefinition zurück.

CICSAPI

Bei dem Programm ist nur eine Verwendung der von CICS zugelassenen Anwendungsprogrammierschnittstellen möglich.

OPENAPI

Das Programm ist nicht auf die Verwendung der von CICS zugelassenen Anwendungsprogrammierschnittstellen beschränkt. Das Programm muss gemäß threadsicherer Standards codiert und mit CONCURRENCY (THREADSAFE) definiert sein.

AVAIL_STATUS(DISABLED|ENABLED)

Gibt einen Wert zurück, der angibt, ob das Programm verwendet werden kann (ENABLED) oder nicht (DISABLED).

CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC)

Gibt den EDF-Status (EDF = Execution Diagnostic Facility) des Programms zurück.

CEDF Wird das Programm unter der Steuerung der Execution Diagnostic Facility (EDF) von CICS ausgeführt, werden EDF-Diagnoseanzeigen angezeigt.

NOCEDF

Es werden keine EDF-Diagnoseanzeigen angezeigt.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich bei dem Modul um eine Zuordnungsgruppe, eine Partitionsgruppe oder ein fernes Programm handelt.

CONCURRENCY(QUASIRENT|THREADSAFE)

Gibt einen Wert für das Attribut der installierten Programmdefinition für Nebenläufigkeit zurück.

QUASIRENT

Das Programm ist als quasiwiedereintrittsfähig definiert und kann nur unter dem QR-Tasksteuerblock von CICS (QR = Quasi-Reentrant, quasiwiedereintrittsfähig) ausgeführt werden.

THREADSAFE

Das Programm ist als threadsicher definiert und kann unter jedem Tasksteuerblock ausgeführt werden, der von der zugehörigen Benutzer-task verwendet wird, wenn das Programm die Steuerung erhält. Dabei kann es sich um einen offenen Tasksteuerblock oder den QR-Tasksteuerblock von CICS handeln.

Anmerkung: Bei einem LE-konformen Programm (LE = Language Environment) kann die ursprünglich definierte Nebenläufigkeit überschrieben werden, wenn das Programm anschließend geladen wird.

DATA_LOCATION(ANY|BELOW|NOT_APPLIC)

Gibt einen Wert zurück, der angibt, ob das Programm auf Daten oberhalb der 16-MB-Grenze zugreifen kann.

ANY Das Programm kann 31-Bit-Adressen verarbeiten und kann Daten erhalten, die sich oberhalb oder unterhalb der 16-MB-Grenze befinden.

BELOW

Das Programm kann nur 24-Bit-Adressen verarbeiten. An dieses Programm dürfen deshalb nur Daten übergeben werden, die sich unterhalb der 16-MB-Grenze befinden.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich bei dem Modul um eine Zuordnungsgruppe, eine Partitionsgruppe oder ein fernes Programm handelt.

DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC)

Gibt einen Wert zurück, der angibt, ob die Anforderung dynamisch weitergeleitet werden kann, wenn das Programm Gegenstand einer programmgesteuerten LINK-Anforderung ist.

DYNAMIC

Ist das Programm Gegenstand einer programmgesteuerten LINK-Anforderung, wird das CICS-Programm für dynamisches Routing aufgerufen. Sofern für die Option SYSID im Befehl EXEC CICS LINK nicht explizit eine bestimmte ferne Serverregion angegeben ist, kann das Routing-Programm die Anforderung an die Region weiterleiten, in der das Programm ausgeführt werden soll.

NOT_DYNAMIC

Ist das Programm Gegenstand einer programmgesteuerten LINK-Anforderung, wird das Programm für dynamisches Routing nicht aufgerufen.

Bei einer DPL-Anforderung (DPL = Distributed Program Link, Verbindung zu verteilten Programmen) muss die Serverregion, in der das Programm ausgeführt werden soll, explizit für die Option REMOTE-SYSTEM der Programmdefinition oder die Option SYSID des Befehls EXEC CICS LINK angegeben sein. Andernfalls wird standardmäßig die lokale Region verwendet.

Informationen zum dynamischen Routing von DPL-Anforderungen finden Sie im Abschnitt Dynamically routing DPL requests.

ENTRY_POINT(name4)

Gibt die Einstiegspunktadresse des Programms wie von einem Ladedomänenaufruf ACQUIRE_PROGRAM zurückgegeben zurück.

EXECUTION_KEY(CICS|NOT_APPLIC|USER)

Gibt den Schlüssel zurück, in dem CICS die Steuerung an das Programm übergibt, wodurch festgelegt wird, ob der CICS-Schlüsselspeicher vom Programm geändert werden kann.

CICS CICS übergibt die Steuerung an das Programm im CICS-Schlüssel. Das Programm wird in einen dynamischen CICS-Speicherbereich (Dynamic Storage Area, DSA) ober- oder unterhalb der 16-MB-Grenze, d. h. in CDSA oder ECDSA, geladen. Die Entscheidung für CDSA oder ECDSA richtet sich nach dem Residenzmodusattribut RMODE, wie für den Verbindungseditor definiert.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich bei dem Modul um eine Zuordnungsgruppe, eine Partitionsgruppe oder ein fernes Programm handelt.

USER CICS übergibt die Steuerung an das Programm im Benutzerschlüssel. Das Programm wird in einen dynamischen Benutzerspeicherbereich ober- oder unterhalb der 16-MB-Grenze, d. h. in UDSA oder EUDSA, geladen. Die Entscheidung für UDSA oder EUDSA richtet sich nach dem Residenzmodusattribut RMODE, wie für den Verbindungseditor definiert.

EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC)

Gibt einen Wert zurück, der angibt, ob CICS eine Verbindung zum Programm herstellt und das Programm so ausführt, als ob es in einer fernen CICS-Region ausgeführt würde.

DPLSUBSET

CICS stellt eine Verbindung zum Programm her und führt das Programm mit den API-Einschränkungen eines fernen DPL-Programms aus. Das Programm kann nur eine Untergruppe der Befehle der CICS-API verwenden.

FULLAPI

CICS stellt eine Verbindung zum Programm her und führt das Programm ohne die API-Einschränkungen eines fernen DPL-Programms aus. Das Programm kann die CICS-API in vollem Umfang verwenden.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich bei dem Modul um eine Zuordnungsgruppe, eine Partitionsgruppe oder ein fernes Programm handelt. (Die Option EXECUTIONSET von DEFINE PROGRAM gilt nur für lokale Programmdefinitionen. Sie dient dazu, Programme in einer lokalen CICS-Umgebung so zu testen, als ob sie als DPL-Programme ausgeführt würden.)

HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE)

Gibt einen Wert zurück, der angibt, wie lange das Programm geladen bleiben soll.

CICS_LIFE

Das Programm bleibt geladen, bis CICS beendet wird.

NOT_APPLIC

Die Option ist nicht anwendbar, da das Programm nicht geladen wurde oder es sich um ein fernes Programm handelt.

TASK_LIFE

Das Programm bleibt die gesamte Laufzeit der Task geladen.

INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO)

Gibt die Methode zurück, die für die Installation der PROGRAM-Ressourcen-Definition verwendet wurde.

AUTO

Automatische Installation.

CATALOG

Der globale CICS-Katalog, nach einem Neustart.

GROUPLIST

Die CICS-Startgruppenliste.

MANUAL

Bei dem Programm handelt es sich um ein internes CICS-Modul, das von einer CICS-Komponente explizit für den Programmmanager definiert wurde.

RDO RDO-Befehle.

SYSAUTO

Automatische Systeminstallation, d. h. automatisch von CICS ohne Aufruf des Benutzerprogramms für automatische Installation installiert. Bei dem Programm kann es sich um ein internes CICS-Modul oder beispielsweise um ein PLTPI-Programm der ersten Phase handeln.

LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI)

Gibt die von CICS abgeleitete Programmiersprache für das Programm zurück. COBOL gibt OS/VS COBOL an, das nicht unter der vorliegenden CICS-Version ausgeführt werden kann, und COBOL2 entweder Enterprise COBOL oder VS COBOL II.

LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|LE370|NOT_APPLIC|NOT_DEFINED|PLI)

Gibt die in der Ressourcendefinition definierte Programmiersprache zurück.

LIBRARY(name)

Gibt den aus 8 Zeichen bestehenden Namen der LIBRARY-Ressource zurück, aus der das Programm geladen wurde. Dieses Feld ist leer, wenn das Programm nicht geladen wurde oder der LPASTATUS LPA lautet. LPA gibt an, dass das Programm aus dem Link-Pack-Bereich (Link Pack Area, LPA) geladen wurde.

LIBRARYDSN(name44)

Gibt den aus 44 Zeichen bestehenden Namen des Datasets zurück, aus dem das Programm geladen wurde. Dieses Feld ist leer, wenn das Programm nicht geladen wurde oder der LPASTATUS LPA lautet. LPA gibt an, dass das Programm aus dem Link-Pack-Bereich (Link Pack Area, LPA) geladen wurde.

- Wurde das Programm aus einer installierten Bibliothek geladen, werden der LIBRARY- und der LIBRARYDSN-Name zurückgegeben.
- Wurde das Programm aus einer Bibliothek geladen, die inaktiviert wurde, wird der LIBRARY-Name zurückgegeben, während für LIBRARYDSN ein Leerwert zurückgegeben wird.
- Wurde das Programm aus einer Bibliothek geladen, die gelöscht wurde, wird sowohl für den LIBRARY- als auch für den LIBRARYDSN-Namen ein Leerwert zurückgegeben.

LOAD_POINT(name4)

Gibt die von einem Ladedomänenaufruf ACQUIRE_PROGRAM zurückgegebene Ladepunktadresse des Programms zurück.

LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED)

Gibt einen Wert zurück, der angibt, ob das Programm geladen werden kann.

LOADABLE

Das Programm kann geladen werden.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich um ein fernes Programm handelt.

NOT_LOADABLE

Von CICS wurde vergeblich versucht, das Programm zu laden. Das Programm ist nicht in der Bibliothek enthalten.

NOT_LOADED

Von CICS wurde noch nicht versucht, das Programm zu laden.

LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA)

Gibt einen Wert zurück, der angibt, an welcher Position sich die zuletzt geladene Kopie des Programms befindet.

CDSA Der dynamische CICS-Speicherbereich.

ECDSA

Der erweiterte dynamische CICS-Speicherbereich.

ELPA Der erweiterte Link-Pack-Bereich.

ERDSA

Der erweiterte schreibgeschützte dynamische Speicherbereich.

ESDSA

Der erweiterte gemeinsam genutzte dynamische Speicherbereich.

LPA Der Link-Pack-Bereich.

NONE

Das Programm wurde nicht geladen.

RDSA Der schreibgeschützte dynamische Speicherbereich.

SDSA Der gemeinsam genutzte dynamische Speicherbereich.

MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM)

Gibt den Typ der Programmressource zurück.

NEW_PROGRAM_TOKEN(name4)

Gibt ein Token zur Kennzeichnung des angegebenen Programms zurück.

name4 Der Name einer Position für das zu empfangende, aus 4 Byte bestehende Token, das das Programm kennzeichnet.

Ist PROGRAM_NAME in der Anforderung angegeben, wird für NEW_PROGRAM_TOKEN ein Programmtoken festgelegt, das für nachfolgende Anforderungen für dasselbe Programm verwendet werden kann. Ist PROGRAM_TOKEN in der Anforderung angegeben, wird für NEW_PROGRAM_TOKEN derselbe Wert festgelegt.

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)

Gibt den Residenzstatus des Programms zurück bzw. gibt an, wann der zugehörige Speicher freigegeben wird.

RELOAD

Das Programm ist nicht wiederverwendbar. Es werden deshalb möglicherweise mehrere Kopien geladen. Wird ein Aufruf RELEASE_PROGRAM für eine Kopie abgesetzt, wird die betreffende Kopie aus dem Speicher entfernt.

RESIDENT

Es gibt eine einzige Kopie des Programms, die erst aus dem Speicher entfernt wird, wenn sie gelöscht wird. Als RESIDENT definierte Programme müssen zumindest quasiwiedereintrittsfähig sein. Programme des Typs PROGRAM_TYPE SHARED verfügen standardmäßig über das Attribut RESIDENT.

REUSABLE

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut REUSABLE, das nicht verwendet wird, von CICS zur Speicheroptimierung aus dem Speicher entfernt werden kann.

TRANSIENT

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut TRANSIENT aus dem Speicher entfernt wird, sobald es nicht genutzt wird (Benutzeranzahl gleich null).

PROGRAM_LENGTH(name4)

Gibt die Länge des Programms in Byte als Binärwert zurück.

PROGRAM_NAME(name8 | string | 'string')

Gibt den Namen des Programms an, für das Informationen angefragt werden.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

'string'

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

PROGRAM_TOKEN(name4)

Gibt ein Token zur Kennzeichnung des Programms an, für das Informationen angefragt werden.

name4 Der Name einer Position mit einem aus 4 Byte bestehenden Token, das aus einem vorherigen Aufruf INQUIRE_PROGRAM stammt.

PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY)

Gibt einen Wert zurück, der angibt, aus welcher Position die nächste neue Kopie des Programms geladen werden soll.

NOT_APPLIC

Die Option ist nicht anwendbar, da es sich um ein fernes Programm handelt.

PRIVATE

Das Programm wird aus der DFHRPL-Bibliotheksverkettung oder einer dynamischen Bibliotheksverkettung geladen. Ein privates Programm muss nicht unbedingt wiedereintrittsfähig sein und erhält nur einen begrenzten Schutz gegen unbefugtes Überschreiben. Der Umfang des Schutzes richtet sich nach dem Typ des dynamischen Speicherbereichs (DSA), in den das Programm geladen wird (siehe dazu die Beschreibung der Option PROGRAM_TYPE für den Aufruf DEFINE_PROGRAM).

SHARED

Das Programm wird aus dem Link-Pack-Bereich (LPA) geladen. Gemeinsam genutzte Programme müssen wiedereintrittsfähig sein und sind geschützt.

Beim nächsten Empfang der Option NEWCOPY oder PHASEIN wird eine LPA-Kopie des Programms verwendet, soweit verfügbar. Ist keine LPA-Version verfügbar, wird das Programm aus der DFHRPL-Bibliotheksverkettung oder einer dynamischen Bibliotheksverkettung geladen.

TYPE_ANY

Es kann die Kopie in der DFHRPL-Bibliotheksverkettung oder einer dynamischen Bibliotheksverkettung oder die LPA-Kopie des Programms verwendet werden. Bevorzugte Option: LPA-Kopie.

PROGRAM_USAGE(APPLICATION|NUCLEUS)

Gibt einen Wert zurück, der angibt, ob das Programm als CICS-Nukleusprogramm oder als Benutzeranwendungsprogramm verwendet wird.

PROGRAM_USE_COUNT(name4)

Gibt die Anzahl der verschiedenen Benutzer zurück, die das Programm aufgerufen haben.

PROGRAM_USER_COUNT(name4)

Gibt die aktuelle Anzahl der Benutzer des Programms zurück.

REMOTE_DEFINITION(LOCAL|REMOTE)

Gibt einen Wert zurück, der angibt, ob es sich bei dem Programm um eine lokale oder eine ferne Ressource handelt. Wenn es sich um eine ferne Ressource handelt, werden Anforderungen für eine Verbindung zum Programm von CICS als DPL-Anforderungen (DPL = Distributed Program Link) behandelt und an die ferne Region weitergegeben.

REMOTE_PROGID(name8)

Gibt den Namen zurück, unter dem das Programm in der fernen CICS-Region bekannt ist, wenn es sich bei dem Programm um eine ferne Ressource handelt. Wurde REMOTESYSTEM in der Programmdefinition angegeben, REMOTENAME jedoch nicht, wird der lokale Name als ferner Name verwendet (d. h. REMOTE_PROGID nimmt standardmäßig den Wert von PROGRAM_NAME an).

REMOTE_SYSID(name4)

Gibt den Namen der fernen CICS-Region zurück, wenn es sich bei dem Programm um eine ferne Ressource handelt.

REMOTE_TRANID(name4)

Gibt den Namen der von der fernen CICS-Region angehängten Transaktion zurück, unter der das Programm ausgeführt wird, wenn es sich bei dem Programm um eine ferne Ressource handelt.

SHOW_PROGRAMS(PRIVATE|PRIVATE_AND_PUBLIC)

Wenn Anwendungskontextfelder für INQUIRE_PROGRAM angegeben werden, definiert SHOW_PROGRAMS den Umfang der Suche.

PRIVATE

Sucht nur nach privaten Programmen.

PRIVATE_AND_PUBLIC

Sucht zunächst nach privaten Programmen und anschließend nach öffentlichen Programmen.

SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64)

Gibt den Adressierungsmodus zurück, der in einem Aufruf DEFINE_PROGRAM angegeben wurde.

SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED)

Gibt den Residenzmodus zurück (d. h. die Angabe, ob das Programm oberhalb oder unterhalb der 16-MB-Grenze geladen werden soll), der in einem Aufruf DEFINE_PROGRAM angegeben wurde.

RESPONSE- und REASON-Werte für INQUIRE_PROGRAM

RESPONSE

OK

EXCEPTION

REASON

PROGRAM_NOT_DEFINED_TO_LD

PROGRAM_NOT_DEFINED_TO_PG

APP_CONTEXT_NOT_FOUND

<i>RESPONSE</i>	<i>REASON</i>
DISASTER	ABEND
	LOCK_ERROR
INVALID	INVALID_PROGRAM_TOKEN
KERNERROR	----
PURGED	----

Aufruf INQUIRE_CURRENT_PROGRAM

Mit dem Aufruf INQUIRE_CURRENT_PROGRAM werden Informationen zu den Attributen des gerade in Ausführung befindlichen Programms zurückgegeben. Wird dieser Aufruf in einem globalen oder taskbezogenen Benutzerexit abgesetzt, gibt er die Attribute des globalen oder taskbezogenen Benutzerexitprogramms selbst zurück.

INQUIRE_CURRENT_PROGRAM

```
DFHPGISX [CALL,]
    [CLEAR,]
    [IN,
    FUNCTION(INQUIRE_CURRENT_PROGRAM),]
    [IGNORE_EXITS(YES|NO),]
    [OUT,
    [AVAIL_STATUS(DISABLED|ENABLED),]
    [CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
    [CURRENT_AMODE(24|31|64),]
    [CURRENT_CEDF_STATUS(CEDF|NOCEDF),]
    [CURRENT_ENTRY_POINT(name4),]
    [CURRENT_ENVIRONMENT(EXEC|GLUE|PLT|SYSTEM|TRUE|URM),]
    [CURRENT_EXECUTION_SET(DPLSUBSET|FULLAPI),]
    [CURRENT_LOAD_POINT(name4),]
    [CURRENT_PROGRAM_LENGTH(name4),]
    [CURRENT_PROGRAM_NAME(name8),]
    [DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
    [DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC),]
    [EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
    [EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
    [HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
    [IGNORE_EXITS(YES|NO),]
    [INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
    [INVOKING_ENVIRONMENT (EXEC|GLUE|PLT|SYSTEM|TRUE|URM),]
    [INVOKING_PROGRAM_NAME(name8),]
    [LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                     COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
    [LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                     LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
    [LIBRARY(name8),]
    [LIBRARYDSN(name44),]
    [LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
    [MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
    [NEW_PROGRAM_TOKEN(name4),]
    [REMOTE_DEFINITION(LOCAL|REMOTE),]
    [REMOTE_PROGID(name8),]
    [REMOTE_SYSID(name4),]
    [REMOTE_TRANID(name4),]
    [RETURN_PROGRAM_NAME(name8),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Anmerkung: Die in der folgenden Liste nicht beschriebenen Optionen stimmen mit den entsprechenden Optionen des Aufrufs INQUIRE_PROGRAM überein. Siehe „Aufruf INQUIRE_PROGRAM“ auf Seite 73.

CURRENT_AMODE(24|31|64)

Gibt den Adressierungsmodus zurück, den das aktive Programm aktuell verwendet.

CURRENT_CEDF_STATUS(CEDF|NOCEDF)

Gibt den EDF-Status der aktuellen Instanz des Programms zurück. Der zurückgegebene Wert stimmt mit dem Wert für CEDF_STATUS überein, bei dem es sich um den in der Programmdefinition angegebenen EDF-Status handelt. Siehe dazu den Abschnitt zur Option CEDF_STATUS für den Aufruf INQUIRE_PROGRAM.

CURRENT_ENTRY_POINT(name4)

Gibt die Einstiegspunktadresse des aktuellen Programms zurück.

CURRENT_ENVIRONMENT(EXEC|GLUE|PLT|SYSTEM|TRUE|URM)

Gibt die Umgebung bzw. den Typ des aktuellen Programms zurück.

EXEC Benutzeranwendungsprogramm.

GLUE Globales Benutzerexitprogramm.

PLT Programm in der Programmlistentabelle.

SYSTEM

CICS-Systemcode.

TRUE Taskbezogenes Benutzerexitprogramm.

URM Durch den Benutzer austauschbares Programm.

CURRENT_EXECUTION_SET(DPLSUBSET|FULLAPI)

Gibt die von der aktuellen Instanz des Programms verwendete API-Ausführungsgruppe zurück. Der zurückgegebene Werte entspricht dem Wert für EXECUTION_SET (die in der Programmdefinition angegebene API-Ausführungsgruppe), *sofern die Instanz nicht* das erste Programm in einer Transaktion ist. In diesem Fall können sich die Werte unterscheiden. Dies ist dadurch bedingt, dass das Attribut DPLSUBSET nur für Programme gilt, zu denen eine Programmverbindung hergestellt wurde. Es wird für das erste Programm in einer Transaktion ignoriert, da es sich bei diesem Programm nicht um das Ziel eines DPL-Aufrufs handeln kann. Wird für das erste Programm in einer Transaktion der Wert DPLSUBSET für EXECUTION_SET zurückgegeben, gibt CURRENT_EXECUTION_SET deshalb dennoch FULLAPI zurück. Siehe dazu den Abschnitt zur Option EXECUTION_SET für den Aufruf INQUIRE_PROGRAM.

CURRENT_LOAD_POINT(name4)

Gibt die Ladepunktadresse des aktuellen Programms zurück.

CURRENT_PROGRAM_LENGTH(name4)

Gibt die Länge des aktuellen Programms in Byte als Binärwert zurück.

CURRENT_PROGRAM_NAME(name8)

Gibt den Namen des Programms zurück, das aktuell ausgeführt wird.

IGNORE_EXITS(YES|NO)

Gibt an, ob globale Benutzerexitprogramme und taskbezogene Benutzerexitprogramme ignoriert werden, wenn Informationen über das Programm zurückgegeben werden, das das aktuelle Programm aufgerufen hat und an das die Steuerung später zurückgegeben wird. Ihre Einstellung für diese Option wirkt sich auf die Werte aus, die von den Optionen INVOKING_ENVIRONMENT, INVO-

KING_PROGRAM_NAME und RETURN_PROGRAM_NAME zurückgegeben werden. Bei Angabe von YES (Standardeinstellung) werden globale Benutzerexitprogramme und taskbezogene Benutzerexitprogramme für diese Optionen ignoriert. Bei Angabe von NO geben die Optionen Informationen zum Exitprogramm zurück, wenn ein globales Benutzerexitprogramm oder ein taskbezogenes Benutzerexitprogramm beteiligt ist.

INVOKING_ENVIRONMENT (EXEC|GLUE|PLT|SYSTEM|TRUE|URM)

Gibt die Umgebung zurück, von der aus das aktuelle Programm aufgerufen wurde. Dabei handelt es sich um die Umgebung, die dem in INVOKING_PROGRAM_NAME angegebenen Programm entspricht. Die Werte entsprechen den für CURRENT_ENVIRONMENT beschriebenen Werten.

INVOKING_PROGRAM_NAME(name8)

Gibt den Namen des letzten Programms zurück, von dem das aktuelle Programm aufgerufen wurde. Ist IGNORE_EXITS(NO) angegeben, kann es sich um ein globales Benutzerexitprogramm oder ein taskbezogenes Benutzerexitprogramm handeln, soweit ein derartiges Programm beteiligt ist. Ist die Standardeinstellung IGNORE_EXITS(YES) angegeben, handelt es sich um das letzte Programm, das *kein* globales Benutzerexitprogramm oder taskbezogenes Benutzerexitprogramm darstellt.

LIBRARY(name)

Gibt den aus 8 Zeichen bestehenden Namen der LIBRARY-Ressource zurück, aus der das Programm geladen wurde. Dieses Feld ist leer, wenn das Programm nicht geladen wurde oder der LPASTATUS LPA lautet. LPA gibt an, dass das Programm aus dem Link-Pack-Bereich (Link Pack Area, LPA) geladen wurde. Wurde das Programm aus einer installierten Bibliothek geladen, werden der LIBRARY- und der LIBRARYDSN-Name zurückgegeben.

LIBRARYDSN(data-area)

Gibt den aus 44 Zeichen bestehenden Namen der Datei zurück, aus der das Programm geladen wurde. Dieses Feld ist leer, wenn das Programm nicht geladen wurde oder der LPASTATUS LPA lautet. LPA gibt an, dass das Programm aus dem Link-Pack-Bereich (Link Pack Area, LPA) geladen wurde. Wurde das Programm aus einer installierten Bibliothek geladen, werden der LIBRARY- und der LIBRARYDSN-Name zurückgegeben.

RETURN_PROGRAM_NAME(name8)

Gibt den Namen des Programms zurück, an das die Steuerung zurückgegeben wird. Ist IGNORE_EXITS(NO) angegeben, kann es sich um ein globales Benutzerexitprogramm oder ein taskbezogenes Benutzerexitprogramm handeln. Ist die Standardeinstellung IGNORE_EXITS(YES) angegeben, handelt es sich um das Programm, das erneut die Steuerung erhält, nachdem zwischengeschaltete globale Benutzerexitprogramme oder taskbezogene Benutzerexitprogramme, soweit beteiligt, abgeschlossen wurden.

RESPONSE- und REASON-Werte für INQUIRE_CURRENT_PROGRAM

RESPONSE	REASON
OK	----
EXCEPTION	NO_CURRENT_PROGRAM
DISASTER	LOCK_ERROR
	ABEND
INVALID	----
KERNERROR	----
PURGED	----

Aufruf SET_PROGRAM

Mit dem Aufruf SET_PROGRAM können Sie ausgewählte Attribute in der Definition eines angegebenen Programms festlegen.

SET_PROGRAM

```
DFHPGISX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(SET_PROGRAM),
           {PROGRAM_NAME(name8 | string | 'string')|
            PROGRAM_TOKEN(name4)},,]
          [AVAIL_STATUS(DISABLED|ENABLED),]
          [CEDF_STATUS(CEDF|NOCEDF),]
          [EXECUTION_KEY(CICS|USER),]
          [EXECUTION_SET(DPLSUBSET|FULLAPI),]
          [PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
          [PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY),]
          [PROGRAM_USAGE(APPLICATION|NUCLEUS),]
          [REQUIRED_AMODE(24|31|AMODE_ANY|64),]
          [REQUIRED_RMODE(24|RMODE_ANY),]
          [OUT,
           RESPONSE(name1 | *),
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

AVAIL_STATUS(DISABLED|ENABLED)

Gibt an, ob das Programm verwendet werden kann (ENABLED) oder nicht (DISABLED).

CEDF_STATUS(CEDF|NOCEDF)

Gibt an, ob EDF-Diagnoseanzeigen angezeigt werden, wenn das Programm unter der Steuerung der Execution Diagnostic Facility (EDF) von CICS ausgeführt wird.

EXECUTION_KEY(CICS|USER)

Gibt den Schlüssel an, in dem CICS die Steuerung an das Programm übergibt. Der Schlüssel legt fest, ob das Programm den CICS-Schlüsselspeicher ändern kann.

CICS CICS übergibt die Steuerung an das Programm im CICS-Schlüssel. Das Programm wird in einen dynamischen CICS-Speicherbereich (Dynamic Storage Area, DSA) ober- oder unterhalb der 16-MB-Grenze, d. h. in CDSA oder ECDSA, geladen. Die Entscheidung für CDSA oder ECDSA richtet sich nach dem Residenzmodusattribut RMODE, wie für den Verbindungseditor definiert.

USER CICS übergibt die Steuerung an das Programm im Benutzerschlüssel. Das Programm wird in einen dynamischen Benutzerspeicherbereich ober- oder unterhalb der 16-MB-Grenze, d. h. in UDSA oder EUDSA, geladen. Die Entscheidung für UDSA oder EUDSA richtet sich nach dem Residenzmodusattribut RMODE, wie für den Verbindungseditor definiert.

Anmerkung: Wurde ein Programm für den Verbindungseditor mit AMODE(31),RMODE(ANY) als wiedereintrittsfähiges Programm definiert, wird die Option EXECUTION_KEY ignoriert und das Programm wird in den erweiterten schreibgeschützten dynamischen Speicherbereich (ERDSA) geladen. Details zu dem Speichertyp, der für ERDSA zugeordnet ist, können Sie dem Abschnitt RENTPGM system initialization parameter entnehmen.

EXECUTION_SET(DPLSUBSET|FULLAPI)

Gibt an, ob CICS eine Verbindung zum Programm herstellt und das Programm so ausführt, als ob es in einer fernen CICS-Region ausgeführt würde.

Anmerkung: Die Option EXECUTIONSET gilt nur für lokale Programmdefinitionen. Sie dient dazu, Programme in einer lokalen CICS-Umgebung so zu testen, als ob sie als DPL-Programme ausgeführt würden.

DPLSUBSET

CICS stellt eine Verbindung zum Programm her und führt das Programm mit den API-Einschränkungen eines fernen DPL-Programms aus. Das Programm kann nur eine Untergruppe der Befehle der CICS-API verwenden.

FULLAPI

CICS stellt eine Verbindung zum Programm her und führt das Programm ohne die API-Einschränkungen eines fernen DPL-Programms aus. Das Programm kann die CICS-API in vollem Umfang verwenden.

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)

Gibt den Residenzstatus des Programms an. Damit wird angegeben, wann der zugehörige Speicher freigegeben wird.

RELOAD

Das Programm ist nicht wiederverwendbar. Es werden deshalb möglicherweise mehrere Kopien geladen. Wird ein Aufruf RELEASE_PROGRAM für eine Kopie abgesetzt, wird die betreffende Kopie aus dem Speicher entfernt.

RESIDENT

Es befindet sich jeweils immer nur eine einzige Kopie des Programms im Speicher und diese Kopie wird nur entfernt, wenn sie gelöscht wird. Als RESIDENT definierte Programme müssen zumindest quasi-wiedereintrittsfähig sein. Programme des Typs PROGRAM_TYPE SHARED verfügen standardmäßig über das Attribut RESIDENT.

REUSABLE

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut REUSABLE, das nicht verwendet wird, von CICS zur Speicheroptimierung aus dem Speicher entfernt werden kann.

TRANSIENT

Entspricht RESIDENT, abgesehen davon, dass ein Programm mit dem Attribut TRANSIENT aus dem Speicher entfernt wird, sobald es nicht genutzt wird (Benutzeranzahl gleich null).

PROGRAM_NAME(name8 | string | 'string')

Gibt den Namen des Programms an, dessen Attribute geändert werden sollen.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

'string'

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

PROGRAM_TOKEN(name4)

Gibt ein Token zur Kennzeichnung des Programms an.

name4 Der Name einer Position mit einem aus 4 Byte bestehenden Token, das aus einem vorherigen Aufruf INQUIRE_PROGRAM, INQUIRE_CURRENT_PROGRAM, START_BROWSE_PROGRAM oder GET_NEXT_PROGRAM stammt.

PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY)

Gibt an, wo das Programm geladen wird.

PRIVATE

Das Programm ist in der DFHRPL-Bibliotheksvernetzung oder einer dynamischen Bibliotheksvernetzung enthalten. Ein privates Programm muss nicht unbedingt wiedereintrittsfähig sein und erhält nur einen begrenzten Schutz gegen unbefugtes Überschreiben. Der Umfang des Schutzes richtet sich nach dem Typ des dynamischen Speicherbereichs (DSA), in den das Programm geladen wird (siehe dazu die Beschreibung der Option PROGRAM_TYPE für den Aufruf DEFINE_PROGRAM).

SHARED

Das Programm befindet sich im Link-Pack-Bereich (LPA), ist wiedereintrittsfähig und wird geschützt.

TYPE_ANY

Es kann die Kopie in der DFHRPL-Bibliotheksvernetzung oder einer dynamischen Bibliotheksvernetzung oder die LPA-Kopie des Programms verwendet werden. Bevorzugte Option: LPA-Kopie.

PROGRAM_USAGE(APPLICATION|NUCLEUS)

Gibt an, ob das Programm als CICS-Nukleusprogramm oder als Benutzeranwendungsprogramm verwendet wird.

REQUIRED_AMODE(24|31|AMODE_ANY|64)

Gibt den Adressierungsmodus des Programms an. Wird bei der nachfolgenden Verarbeitung festgestellt, dass keine Kopie des Programms gefunden werden kann, die die definierten Adressierungsanforderungen erfüllt, tritt eine Ausnahmebedingung ein.

Anmerkung:

1. AMODE_ANY und AMODE 31 haben bei dieser Funktion eine identische Bedeutung.
2. Sie können mit dieser Option nicht den für den Verbindungseditor angegebenen Adressierungsmodus des Programms überschreiben.

REQUIRED_RMODE(24|AMODE_ANY)

Gibt den Residenzmodus des Programms an (d. h., ob das Programm oberhalb oder unterhalb der 16-MB-Grenze geladen wird). Wird bei der nachfolgenden Verarbeitung festgestellt, dass keine Kopie des Programms gefunden werden kann, die die definierten Residenzanforderungen erfüllt, tritt eine Ausnahmebedingung ein.

Anmerkung: Sie können mit dieser Option nicht den für den Verbindungseditor angegebenen Residenzmodus des Programms überschreiben.

RESPONSE- und REASON-Werte für SET_PROGRAM

RESPONSE

OK

EXCEPTION

REASON

CEDF_STATUS_NOT_FOR_MAPSET

CEDF_STATUS_NOT_FOR_PTNSSET

<i>RESPONSE</i>	<i>REASON</i>
	CEDEF_STATUS_NOT_FOR_REMOTE
	EXEC_KEY_NOT_FOR_MAPSET
	EXEC_KEY_NOT_FOR_PTNSSET
	EXEC_KEY_NOT_FOR_REMOTE
	EXEC_SET_NOT_FOR_MAPSET
	EXEC_SET_NOT_FOR_PTNSSET
	EXEC_SET_NOT_FOR_REMOTE
	INCOMPATIBLE_BUNDLE_SET
	PROGRAM_NOT_DEFINED_TO_LD
	PROGRAM_NOT_DEFINED_TO_PG
<i>DISASTER</i>	ABEND
	CATALOG_ERROR
	CATALOG_NOT_OPERATIONAL
	LOCK_ERROR
<i>INVALID</i>	INVALID_MODE_COMBINATION
	INVALID_PROGRAM_NAME
	INVALID_PROGRAM_TOKEN
	INVALID_TYPE_ATTRIB_COMBIN
<i>KERNERROR</i>	----
<i>PURGED</i>	----

Aufruf **START_BROWSE_PROGRAM**

Mit dem Aufruf **START_BROWSE_PROGRAM** können Sie ein Token abrufen, das es Ihnen ermöglicht, Programmdefinitionen zu durchsuchen und dabei optional mit der Definition eines angegebenen Programms zu beginnen.

START_BROWSE_PROGRAM

```
DFHPGISX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(START_BROWSE_PROGRAM),
          [PROGRAM_NAME(name8 | string | 'string'),]
          [OUT,
          BROWSE_TOKEN(name4)
          RESPONSE(name1 | *),
          REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

BROWSE_TOKEN(name4)

Gibt ein Token zurück, das in einem Aufruf von **GET_NEXT_PROGRAM** verwendet werden soll, um ein sequenzielles Durchsuchen von Programmdefinitionen einzuleiten.

name4 Der Name einer Position für das zu empfangende, aus 4 Byte bestehende Token.

PROGRAM_NAME(name8 | string | 'string')

Gibt den Namen des Programms an, dessen Definition zuerst angezeigt werden soll. Die Suchsequenz ist alphabetisch. Gibt es kein Programm mit dem angegebenen Namen, gibt CICS ein Token für die Definition zurück, die dem Alphabet nach als nächstes folgt. Wenn Sie kein Programm angeben, gibt CICS ein Token für die erste Definition zurück.

name8 Der Name einer Position mit einem aus 8 Byte bestehenden Programmnamen.

string Eine Zeichenfolge, die das Programm benennt.

'string'

Eine in Anführungszeichen gesetzte Zeichenfolge. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt. Diese Länge wird ggf. durch Auffüllen mit Leerzeichen oder Abschneiden erreicht.

RESPONSE- und REASON-Werte für START_BROWSE_PROGRAM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	
DISASTER	ABEND
	INVALID_DIRECTORY
	LOCK_ERROR
INVALID	----
KERNERROR	----
PURGED	----

Aufruf GET_NEXT_PROGRAM

Mit dem Aufruf GET_NEXT_PROGRAM können Sie in einer Suchsequenz, die über den Aufruf START_BROWSE_PROGRAM initiiert wird, die nächste Programmdefinition anfragen. Die Suchsequenz ist alphabetisch. Das Ende der alphabetischen Liste der Definitionen wird mit einer END_LIST-Ausnahmeantwort angezeigt.

GET_NEXT_PROGRAM

```
DFHPGISX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(GET_NEXT_PROGRAM),
          BROWSE_TOKEN(name4),]
          [OUT,
          PROGRAM_NAME(name8),
          [ACCESS(CICS|NONE|READ_ONLY|USER),]
          [AVAIL_STATUS(DISABLED|ENABLED),]
          [CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
          [DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
          [ENTRY_POINT(name4),]
          [EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
          [EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
          [HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
          [INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
          [LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                           COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
          [LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                           LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
          [LOAD_POINT(name4),]
          [LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
          [LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA),]
          [MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
          [NEW_PROGRAM_TOKEN(name4),]
          [PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
          [PROGRAM_LENGTH(name4),]
          [PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY),]
          [PROGRAM_USAGE(APPLICATION|NUCLEUS),]
          [PROGRAM_USE_COUNT(name4),]
          [PROGRAM_USER_COUNT(name4),]
          [REMOTE_DEFINITION(LOCAL|REMOTE),]
```

```
[REMOTE_PROGID(name8),]
[REMOTE_SYSID(name4),]
[REMOTE_TRANID(name4),]
[SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64),]
[SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Anmerkung: Die in der folgenden Liste nicht beschriebenen Optionen stimmen mit den entsprechenden Optionen des Aufrufs INQUIRE_PROGRAM überein. Siehe „Aufruf INQUIRE_PROGRAM“ auf Seite 73.

BROWSE_TOKEN(name4)

Gibt ein Token zur Kennzeichnung der zu durchsuchenden Definition an. Dabei kann es sich um ein Token handeln, das im Feld NEW_PROGRAM_TOKEN des Aufrufs GET_NEXT_PROGRAM oder im Feld BROWSE_TOKEN des Aufrufs START_BROWSE_PROGRAM zurückgegeben wird. (Dieses Token wird nach jedem Aufruf GET_PROGRAM aktualisiert.)

name4 Der Name einer Position, die ein aus 4 Byte bestehendes Token enthält.

NEW_PROGRAM_TOKEN(name4)

Gibt ein Token zurück, das die nächste Definition in der Suchsequenz kennzeichnet. Sie können das Token im Feld BROWSE_TOKEN Ihres nächsten Aufrufs GET_NEXT_PROGRAM (oder Aufrufs END_BROWSE_PROGRAM, wenn die Sequenz beendet werden soll) verwenden. Sie können das Token auch im Feld PROGRAM_TOKEN der Aufrufe INQUIRE_PROGRAM und SET_PROGRAM verwenden.

name4 Der Name einer Position für das zu empfangende, aus 4 Byte bestehende Token, das die nächste Programmdefinition kennzeichnet.

RESPONSE- und REASON-Werte für GET_NEXT_PROGRAM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	END_LIST
	INVALID_BROWSE_TOKEN
	PROGRAM_NOT_DEFINED_TO_LD
DISASTER	ABEND
	LOCK_ERROR
INVALID	----
KERNERROR	----
PURGED	----

Aufruf END_BROWSE_PROGRAM

Mit dem Aufruf END_BROWSE_PROGRAM kann das Durchsuchen von Programmdefinitionen beendet werden, die von START_BROWSE_PROGRAM initiiert wurde.

END_BROWSE_PROGRAM

```
DFHPGISX [CALL,]
          [CLEAR,]
          [IN,]
          FUNCTION(END_BROWSE_PROGRAM),
```

```

BROWSE_TOKEN(name4),]
[OUT,
RESPONSE(name1 | *),
REASON(name1 | *)]

```

Dieser Befehl ist threadsicher.

BROWSE_TOKEN(name4)

Gibt das Token an, das im Feld NEW_PROGRAM_TOKEN des Aufrufs GET_NEXT_PROGRAM oder im Feld BROWSE_TOKEN des Aufrufs START_BROWSE_PROGRAM zurückgegeben wird. (Dieses Token wird nach jedem Aufruf GET_NEXT_PROGRAM aktualisiert.)

RESPONSE- und REASON-Werte für END_BROWSE_PROGRAM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_BROWSE_TOKEN
DISASTER	ABEND
	LOCK_ERROR
INVALID	----
KERNERROR	----
PURGED	----

Aufruf INQUIRE_AUTOINSTALL

Mit dem Aufruf INQUIRE_AUTOINSTALL werden Informationen zu den aktuellen Einstellungen der Funktion für die automatische Installation von Programmen, Zuordnungsgruppen und Partitionsgruppen zurückgegeben.

INQUIRE_AUTOINSTALL

```

DFHPGAQX [CALL,]
[CLEAR,]
[IN,
FUNCTION(INQUIRE_AUTOINSTALL),]
[OUT,
[AUTOINSTALL_CATALOG (ALL|MODIFY|NONE),]
[AUTOINSTALL_EXIT_NAME(name8),]
[AUTOINSTALL_STAT (ACTIVE|INACTIVE),]
RESPONSE(name1 | *),
REASON(name1 | *)]

```

Dieser Befehl ist threadsicher.

AUTOINSTALL_CATALOG(ALL|MODIFY|NONE)

Gibt den Katalogstatus für automatisch installierte Programmdefinitionen zurück.

ALL Alle automatisch installierten Programm-, Zuordnungs- und Partitionsgruppeneinstellungen werden katalogisiert.

MODIFY

Automatisch installierte Programm-, Zuordnungs- und Partitionsgruppeneinstellungen werden nur im globalen CICS-Katalog aufgezeichnet, wenn sie nach der automatischen Installation über einen Befehl SET PROGRAM geändert werden.

NONE

Es werden keine automatisch installierten Programm-, Zuordnungs- oder Partitionsgruppeneinstellungen katalogisiert.

AUTOINSTALL_EXIT_NAME(name8)

Gibt den Namen des durch Benutzer austauschbaren Steuerprogramms für die automatische Installation von Programmen, Zuordnungsgruppen und Partitionsgruppen zurück.

AUTOINSTALL_STATE(ACTIVE|INACTIVE)

Gibt den Status der Funktion für die automatische Installation von Programmen zurück.

ACTIVE

Die automatische Installation ist für Programme, Zuordnungsgruppen und Partitionsgruppen aktiviert.

INACTIVE

Die automatische Installation ist nicht für Programme, Zuordnungsgruppen und Partitionsgruppen aktiviert.

RESPONSE- und REASON-Werte für INQUIRE_AUTOINSTALL

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	----
DISASTER	----
INVALID	INVALID_FUNCTION
KERNERROR	----
PURGED	----

Aufruf SET_AUTOINSTALL

Mit dem Aufruf SET_AUTOINSTALL können Sie die Einstellungen der Funktion für die automatische Installation von Programmen, Zuordnungsgruppen und Partitionsgruppen ändern.

SET_AUTOINSTALL

```
DFHPGAQX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(SET_AUTOINSTALL),
           [AUTOINSTALL_CATALOG (ALL|MODIFY|NONE),]
           [AUTOINSTALL_EXIT_NAME(name8),]
           [AUTOINSTALL_STATE (ACTIVE|INACTIVE),]
           [LANGUAGES_AVAILABLE(NO|YES),]]
          [OUT,
           RESPONSE(name1 | *),
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

AUTOINSTALL_CATALOG(ALL|MODIFY|NONE)

Gibt den Katalogstatus für automatisch installierte Programmdefinitionen an.

ALL Alle automatisch installierten Programm-, Zuordnungs- und Partitionsgruppendifinitionen werden katalogisiert.

MODIFY

Automatisch installierte Programm-, Zuordnungs- und Partitionsgruppendifinitionen werden nur im globalen CICS-Katalog aufgezeichnet, wenn sie nach der automatischen Installation über einen Befehl SET PROGRAM geändert werden.

NONE

Keine automatisch installierten Programm-, Zuordnungs- und Partitionsgruppensdefinitionen werden katalogisiert.

AUTOINSTALL_EXIT_NAME(name8)

Gibt den Namen des durch Benutzer austauschbaren Steuerprogramms für die automatische Installation von Programmen, Zuordnungsgruppen und Partitionsgruppen an.

AUTOINSTALL_STATE(ACTIVE|INACTIVE)

Gibt den Status der Funktion für die automatische Installation von Programmen an.

ACTIVE

Die automatische Installation wird für Programme, Zuordnungsgruppen und Partitionsgruppen aktiviert.

INACTIVE

Die automatische Installation wird für Programme, Zuordnungsgruppen und Partitionsgruppen inaktiviert.

LANGUAGES_AVAILABLE(NO|YES)

Gibt an, ob das Steuerprogramm für die automatische Installation aufgerufen werden kann. Das Steuerprogramm kann erst nach der Einrichtung von Language Environment aufgerufen werden.

NO Das Steuerprogramm kann nicht aufgerufen werden.

YES Das Steuerprogramm kann aufgerufen werden.

RESPONSE- und REASON-Werte für SET_AUTOINSTALL

RESPONSE	REASON
OK	----
EXCEPTION	----
DISASTER	----
INVALID	INVALID_FUNCTION
KERNERROR	----
PURGED	----

Aufruf BIND_CHANNEL

Mit dem Aufruf BIND_CHANNEL wird ein Kanal an eine Task gebunden. Der Aufruf muss vor dem ersten Programm in einer Task abgesetzt werden.

BIND_CHANNEL

```
DFHPGCHX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(BIND_CHANNEL),  
           CHANNEL_TOKEN(name4)]  
          [OUT,  
           RESPONSE(name1 | *),  
           REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

CHANNEL_TOKEN(name4)

Gibt den Namen einer Position an, die ein aus 4 Byte bestehendes Token für den an die Task zu bindenden Kanal enthält.

RESPONSE- und REASON-Werte für BIND_CHANNEL

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_TOKEN CHANNEL_ALREADY_SET CHANNEL_ON_RESTART
DISASTER	----
INVALID	INVALID_LINK_LEVEL
KERNERROR	----
PURGED	----

Kapitel 13. XPI-Funktionen für Statusdatenzugriff

Die XPI stellt Funktionen für den Zugriff auf Statusdaten bereit, mit denen Sie bestimmte Systemdaten in der AP-Domäne anfragen und festlegen können. Dabei handelt es sich um die DFHAPIQX-Aufrufe INQ_APPLICATION_DATA, INQUIRE_SYSTEM und SET_SYSTEM.

Aufruf INQ_APPLICATION_DATA

Mit dem Aufruf INQ_APPLICATION_DATA können Sie Informationen zu Anwendungssystemdaten der AP-Domäne anfragen.

INQ_APPLICATION_DATA

```
DFHAPIQX [CALL,]  
[CLEAR,]  
[IN,  
FUNCTION(INQ_APPLICATION_DATA),]  
[OUT,  
[ACEE(name4 | (Rn) | * ),]      [DSA(name4 | (Rn) | * ),]  
[EIB(name4 | (Rn) | * ),]  
[RSA(name4 | (Rn) | * ),]  
[SYSEIB(name4 | (Rn) | * ),]  
[TCTUA(name4 | (Rn) | * ),]  
[TCTUASIZE(name4 | * ),]  
[TWA(name4 | (Rn) | * ),]  
[TWASIZE(name4 | (Rn) | * ),]  
RESPONSE (name1 | * ),  
REASON (name1 | * )]
```

Dieser Befehl ist threadsicher.

ACEE(name4 | (Rn) | *)

Gibt die Adresse des Zugriffssteuerungsumgebungselements (Access Control Environment Element, ACEE) zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende ACEE-Adresse.

(Rn) Ein Register für die zu empfangende ACEE-Adresse.

***** Die Parameterliste selbst (Name: APIQ_ACEE) wird zum Beibehalten der Adresse verwendet.

DSA(name4 | (Rn) | *)

Gibt den Anfang der Kette der dynamischen Speicherbereiche zurück, die von Anwendungsprogrammen verwendet wird, um die Programme wiedereintrittsfähig zu machen (für Assemblerprogramme z. B. der Speicher DFHEISTG).

name4 Der Name eines aus 4 Byte bestehenden Bereichs, in dem die Adresse für den Anfang der Kette der dynamischen Speicherbereiche (Dynamic Storage Area, DSA) empfangen wird.

(Rn) Ein Register für die zu empfangende DSA-Adresse.

***** Die Parameterliste selbst (Name: APIQ_DSA) wird zum Beibehalten der Adresse verwendet.

EIB(name4 | (Rn) | *)

Gibt die Adresse des EXEC-Schnittstellenblocks (EXEC Interface Block, EIB) für die aktuelle Task zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende EIB-Adresse.

(Rn) Ein Register für die zu empfangende EIB-Adresse.

* Die Parameterliste selbst (Name: APIQ_EIB) wird zum Beibehalten der Adresse verwendet.

RSA(name4 | (Rn) | *)

Gibt die Adresse des Registersicherungsbereichs für die aktuelle Task zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende Adresse des Registersicherungsbereichs (Register Save Area, RSA).

(Rn) Ein Register für die zu empfangende Adresse des Registersicherungsbereichs.

* Die Parameterliste selbst (Name: APIQ_RSA) wird zum Beibehalten der Adresse verwendet.

SYSEIB(name4 | (Rn) | *)

Gibt die Adresse des EXEC-Schnittstellenblocks des Systems für die aktuelle Task zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende Adresse des EXEC-Schnittstellenblocks des Systems.

(Rn) Ein Register für die zu empfangende Adresse des EXEC-Schnittstellenblocks des Systems.

* Die Parameterliste selbst (Name: APIQ_SYSEIB) wird zum Beibehalten der Adresse verwendet.

TCTUA(name4 | (Rn) | *)

Gibt die Adresse des Benutzerbereichs für die Terminalsteuertabelle (Terminal Control Table User Area, TCTUA) für die aktuelle Task zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende TCTUA-Adresse.

(Rn) Ein Register für die zu empfangende TCTUA-Adresse.

* Die Parameterliste selbst (Name: APIQ_TCTUA) wird zum Beibehalten der Adresse verwendet.

TCTUASIZE(name4 | (Rn) | *)

Gibt die TCTUA-Länge für die aktuelle Task in Byte zurück.

name4 Der Name eines aus 4 Byte bestehenden Bereichs für die zu empfangende TCTUA-Länge in Byte.

(Rn) Ein Register für die zu empfangende TCTUA-Länge.

* Die Parameterliste selbst (Name: APIQ_TCTUASIZE) wird zum Beibehalten der TCTUA-Länge verwendet.

TWA(name4 | (Rn) | *)

Gibt die Adresse des Transaktionsarbeitsbereichs (Transaction Work Area, TWA) zurück.

name4 Der Name eines Vollwortbereichs für die zu empfangende TWA-Adresse.

(Rn) Ein Register für die zu empfangende TWA-Adresse.

- * Die Parameterliste selbst (Name: APIQ_TWA) wird zum Beibehalten der Adresse verwendet.

TWASIZE(name4 | (Rn) | *)

Gibt die TWA-Länge in Byte zurück.

name4 Der Name eines aus 4 Byte bestehenden Bereichs für die zu empfangende TWA-Länge in Byte.

(Rn) Ein Register für die zu empfangende TWA-Länge.

- * Die Parameterliste selbst (Name: APIQ_TWASIZE) wird zum Beibehalten der TWA-Länge verwendet.

RESPONSE- und REASON-Werte für INQ_APPLICATION_DATA

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	DPL_PROGRAM
	NO_TRANSACTION_ENVIRONMENT
	TRANSACTION_DOMAIN_ERROR
DISASTER	ABEND
	LOOP
	INQ_FAILED
INVALID	INVALID_FUNCTION
KERNERROR	----
PURGED	----

Aufruf INQUIRE_SYSTEM

Mit dem Aufruf INQUIRE_SYSTEM erhalten Sie Zugriff auf CICS-Systemdaten in der AP-Domäne.

INQUIRE_SYSTEM

```
DFHSAIQX [CALL,]
  [CLEAR,]
  [IN,
  FUNCTION(INQUIRE_SYSTEM),
  [GMMTEXT(name4),]]
  [OUT,
  [CICSREL(name4 | *),]
  [CICSSTATUS(ACTIVE | FINALQUIESCE |
               FIRSTQUIESCE | INITIALIZING),]
  [CICSSYS(name1 | *),]
  [CICTSLEVEL(name6 | *),]
  [CWA(name4 | (Rn) | *),]
  [CWALENGTH(name2 | *),]
  [DATE(name4|*),]
  [DTRPRGRM(name8 | *),]
  [GMMLENGTH(name2 | *),]
  [GMMTRANID(name4 | *),]
  [INITSTATUS(FIRSTINIT | INITCOMPLETE | SECONDINIT |
               THIRDINIT),]
  [JOBNAME(name8 | *),]
  [OPREL(name2 | *),]
  [OPSYS(name1 | *),]
  [OSLEVEL(name4 | *),]
  [PLTPI(name2 | *),]
  [SDTRAN(name4 | *),]
  [SECURITYMGR(EXTSECURITY | NOSECURITY),]
  [SHUTSTATUS(CONTROLSHUT | NOTSHUTDOWN | SHUTDOWN),]
  [STARTUP(COLDSTART | EMERGENCY | WARMSTART),]
```

```
[STARTUPDATE(name4 | *),]
[TERMURM(name8 | *),]
[TIMEOFDAY(name4 | *),]
[XRFSTATUS(NOXRF | PRIMARY | TAKEOVER),]
RESPONSE (name1 | * ),
REASON (name1 | * )]
```

Dieser Befehl ist threadsicher.

CICSREL(name4 | *)

Gibt die Versionsnummer des CICS-Codes zurück, unter dem die CICS-Region ausgeführt wird.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die Zeichen für die Versionsnummer als Hexadezimalwerte empfangen werden.

CICSSTATUS(ACTIVE|FINALQUIESE|FIRSTQUIESCE|INITIALIZING)

Gibt den Status der CICS-Region zurück.

ACTIVE

Die CICS-Region ist aktiv und einsatzbereit.

FINALQUIESCE

Die CICS-Region wird beendet und befindet sich in der letzten Phase des Eintritts in den Quiescemodus.

FIRSTQUIESCE

Die CICS-Region wird beendet und befindet sich in der ersten Phase des Eintritts in den Quiescemodus.

INITIALIZING

Die CICS-Region wird initialisiert.

CICSSYS(name1 | *)

Gibt das Betriebssystem zurück, für das die aktive CICS-Instanz erstellt wurde.

name1 Der Name eines aus 1 Byte bestehenden Bereichs, in dem das Hexadezimalzeichen für das Betriebssystem empfangen wird. Der Wert „X“ steht für MVS.

CICSTSLEVEL(name6 | *)

Gibt das CICS Transaction Server-Release zurück, unter dem CICS ausgeführt wird.

name6 Der Name eines aus 6 Byte bestehenden Bereichs, in dem die Zeichen für das Release als Hexadezimalwerte empfangen werden.

CWA(name4 | (Rn) | *)

Gibt die Adresse des gemeinsamen Arbeitsbereichs (Common Work Area, CWA) zurück.

name4 Der Name eines aus 4 Byte bestehenden Felds für die zu empfangende CWA-Adresse.

(Rn) Ein Register für die zu empfangende CWA-Adresse.

CWALENGTH(name2 | *)

Gibt die Länge des CWA in Byte zurück.

name2 Der Name eines aus 2 Byte bestehenden Felds für die zu empfangende CWA-Länge.

DATE(name4 | *)

Gibt das aktuelle Datum im gepackten, aus 4 Byte bestehenden Dezimalformat 0Cyyddds zurück. Dabei gilt Folgendes:

- C gibt das Jahrhundert an. 0 = 1900, 1 = 2000, 2 = 2100 etc.
- yy gibt die Jahre an.
- ddd gibt die Tage an.
- s gibt das Vorzeichen an.

name4 Der Name einer aus 4 Byte bestehenden Position für das zu empfangende Datum.

DTRPRGRM(name8 | *)

Gibt den Namen des Programms für dynamisches Routing zurück.

name8 Der Name eines aus 8 Byte bestehenden Bereichs, in dem der Name des Programms für dynamisches Routing empfangen wird.

GMMLENGTH(name2 | *)

Gibt die Länge der Begrüßungsnachricht,,“ (Good Morning Message, GMM) in Byte zurück.

name2 Der Name eines aus 2 Byte bestehenden Bereichs, in dem die Länge der Begrüßungsnachricht empfangen wird.

GMMTEXT(name4)

Gibt die Adresse eines Speicherbereichs mit einer Länge von mindestens 244 Byte an, dessen Eigner das aufrufende Modul ist und in das CICS die Begrüßungsnachricht zurückgibt.

name4 Die Adresse eines Speicherbereichs für die zu empfangende Begrüßungsnachricht.

Anmerkung: Der Parameter GMMTEXT muss auf die Anweisung IN als Eingabeparameter folgen.

GMMTRANID(name4 | *)

Gibt die Transaktions-ID der CICS-Begrüßungsnachricht zurück.

name4 Der Name eines aus 4 Byte bestehenden Bereichs, in dem die Transaktions-ID der CICS-Begrüßungsnachricht empfangen wird.

INITSTATUS(FIRSTINIT|INITCOMPLETE|SECONDINIT|THIRDINIT)

Gibt einen Wert für die bei der CICS-Initialisierung erreichte Phase zurück.

FIRSTINIT

Die erste Phase der CICS-Initialisierung.

INITCOMPLETE

Die CICS-Initialisierung ist abgeschlossen.

SECONDINIT

Die zweite Phase der CICS-Initialisierung. Diese Phase entspricht dem Zeitraum, in dem die PLTPI-Programme der ersten Phase ausgeführt werden. Dabei handelt es sich um die Programme in einer Programm-listentabelle (Program List Table, PLT), die **vor** der Anweisung DFHDELIM definiert sind.

THIRDINIT

Die dritte Phase der CICS-Initialisierung. Diese Phase entspricht dem Zeitraum, in dem die PLTPI-Programme der zweiten Phase ausgeführt werden. Dabei handelt es sich um die Programme in einer Programm-listentabelle, die **nach** der Anweisung DFHDELIM definiert sind.

JOBNAME(name8 | *)

Gibt den aus 8 Zeichen bestehenden Namen des MVS-Jobs zurück, unter dem die CICS-Region ausgeführt wird.

name8 Der Name eines aus 8 Byte bestehenden Bereichs für den zu empfangenden MVS-Jobnamen.

OPREL(name2 | *)

Gibt die letzten beiden Ziffern der Versionsnummer des MVS-Elements von z/OS zurück, unter dem die CICS-Region ausgeführt wird.

name2 Der Name eines aus 2 Byte bestehenden Bereichs, in dem die Versionsnummer des MVS-Elements von z/OS als Halbwort-Binärwert empfangen wird. '03' steht beispielsweise für z/OS Release 3 MVS.

Anmerkung: Dieses Feld wird nur aus Kompatibilitätsgründen unterstützt. Diese Information wird aus den beiden letzten Ziffern im MVS-Feld CVT-PRODN abgeleitet. Der CVTPRODN-Wert 'SP5.2.2' steht beispielsweise für MVS/ESA SP Version 5 Release 2.2 (OPREL gibt in diesem Fall '22' zurück) und 'SP6.0.3' für z/OS Release 3. Es wird empfohlen, das Feld OSLEVEL für die vollständige Versions- und Releasenummer des z/OS-Produkts zu verwenden.

OPSYS(name1 | *)

Gibt den Typ des Betriebssystems zurück, unter dem die CICS-Region ausgeführt wird.

name1 Der Name eines aus 1 Byte bestehenden Bereichs, in dem das Hexadezimalzeichen für das Betriebssystem empfangen wird, unter dem CICS ausgeführt wird. Der Wert „X“ steht für MVS.

OSLEVEL(name4 | *)

Gibt die Version, das Release und die Modifikationsstufe des z/OS-Produkts an, unter dem CICS ausgeführt wird.

name1 Der Name eines aus 4 Byte bestehenden Bereichs, in dem die z/OS-Versions- und Releasenummer empfangen wird, unter der CICS ausgeführt wird. Der Wert „0240“ steht für z/OS Release 4.

PLTPI(name2 | *)

Gibt das Suffix zurück, das die Programmliistentabelle mit der Liste der Programme kennzeichnet, die während der Initialisierung von CICS ausgeführt werden sollen. Diese Liste wird als PLTPI (Program List Table Post Initialization) bezeichnet.

name2 Der Name eines aus 2 Byte bestehenden Bereichs für das zu empfangende Suffix.

SDTRAN(name4 | *)

Gibt den Namen der Transaktion für Beendigungsunterstützung zurück, die zu Beginn einer normalen oder sofortigen Beendigung ausgeführt werden soll. Informationen zur Transaktion für Beendigungsunterstützung finden Sie im Abschnitt The shutdown assist utility program, DFHCESD.

name4 Der Name eines aus 4 Byte bestehenden Bereichs für den zu empfangenden Namen.

SECURITYMGR(EXTSECURITY|NOSECURITY)

Gibt zurück, ob ein Sicherheitsmanager aktiv ist.

EXTSECURITY

CICS verwendet einen externen Sicherheitsmanager, z. B. RACF.

NOSECURITY

In der CICS-Region wird kein Sicherheitsmanager verwendet. 'SEC=NO' ist als Systeminitialisierungsparameter angegeben.

SHUTSTATUS(CTRLSHUT|NOTSHUTDOWN|SHUTDOWN)

Gibt den Beendigungsstatus der CICS-Region zurück.

CTRLSHUT

CICS führt ein kontrolliertes Beenden, d. h. ein normales Beenden mit einem Schlüsselpunkt für normales Beenden, durch.

NOTSHUTDOWN

CICS befindet sich nicht im Beendigungsmodus.

SHUTDOWN

CICS führt eine sofortige Beendigung durch.

STARTUP(COLDSTART|EMERGENCY|WARMSTART)

Gibt den aktuellen Starttyp der CICS-Region zurück.

COLDSTART

Von CICS wurde ein Kaltstart wie explizit über den Systeminitialisierungsparameter angegeben durchgeführt oder von CICS wurde aufgrund des Status des globalen Katalogs ein Kaltstart erzwungen.

EMERGENCY

Von CICS wurde ein außerplanmäßiger Neustart durchgeführt, da die vorherige Ausführung nicht normal mit einem normalen Schlüsselpunkt beendet wurde.

WARMSTART

Von CICS wurde nach einer normalen Beendigung der vorherigen Ausführung ein normaler Neustart durchgeführt.

STARTUPDATE(name4 | *)

Gibt das Startdatum der CICS-Region im gepackten, aus 4 Byte bestehenden Dezimalformat **00yydddc** zurück. Dabei steht **yy** für die Jahre, **ddd** für die Tage, **c** für das Vorzeichen.

name4 Der Name einer aus 4 Byte bestehenden Position, an der das Startdatum des CICS-Systems empfangen wird.

TERMURM(name8 | *)

Gibt den Namen des Benutzerprogramms für die automatische Installation von Terminals zurück.

name8 Der Name eines aus 8 Byte bestehenden Bereichs, in dem der Name Benutzerprogramms für die automatische Installation von Terminals empfangen wird.

TIMEOFDAY(name4 | *)

Gibt die aktuelle Uhrzeit im aus 4 Byte bestehenden, gepackten Dezimalformat **hhmmssstc** zurück. Dabei steht **hh** für die Stunden, **mm** für die Minuten, **ss** für die Sekunden, **t** für die Zehntelsekunden und **c** für das Vorzeichen.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende Uhrzeit.

XRFSTATUS(NOXRF|PRIMARY|TAKEOVER)

Gibt den XRF-Status der CICS-Region zurück.

NOXRF

CICS wurde mit dem Systeminitialisierungsparameter 'XRF=NO' gestartet. XRF ist nicht aktiv.

PRIMARY

Die CICS-Region wurde als aktive CICS-Region in einer XRF-Umgebung gestartet.

TAKEOVER

Die CICS-Region wurde als alternative CICS-Region mit dem Systeminitialisierungsparameter START=STANDBY gestartet.

RESPONSE- und REASON-Werte für INQUIRE_SYSTEM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
INVALID	INVALID_FUNCTION
EXCEPTION	LENGTH_ERROR
	UNKNOWN_DATA
DISASTER	INQ_FAILED
PURGED	----

Aufruf SET_SYSTEM

Mit dem Aufruf SET_SYSTEM können Sie Werte für CICS-Systemdaten in der AP-Domäne festlegen.

SET_SYSTEM

```
DFHSAIQX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(SET_SYSTEM),
          [DTRPRGRM(name8 | string | 'string'),]
          [GMMLENGTH(name2 | (Rn) | expression),]
          [GMMTEXT(name8 | (Rn)),]]
          [OUT,
          RESPONSE (name1 | * ),
          REASON (name1 | * )]
```

Dieser Befehl ist threadsicher.

DTRPRGRM(name8 | string | 'string')

Gibt den Namen des Programms für dynamisches Routing an.

name8 Der Name eines aus 8 Byte bestehenden Bereichs mit dem Namen des Programms für dynamisches Routing.

string Eine Zeichenfolge ohne eingebettete Leerzeichen, die den Namen des festzulegenden Programms für dynamisches Routing definiert.

'string'

Eine Zeichenfolge ohne eingebettete Leerzeichen. Verwenden Sie dieses Format, wenn ein Name (eine Bezeichnung) in Ihrem Programm dokumentiert werden soll.

GMMLENGTH(name2 | (Rn))

Gibt die Länge der neuen „Begrüßungsnacht (Good Morning Message, GMM) an, die über den Parameter GMMTEXT übergeben wird.

name2 Der Name eines aus 2 Byte bestehenden Bereichs, der die Länge der neuen Begrüßungsnachricht in Form eines Halbwort-Binärwerts enthält.

(Rn) Ein Register mit der Länge der neuen Begrüßungsnachricht.

GMMTEXT(name4 | (Rn))

Gibt die neue Begrüßungsnachricht an.

name4 Der Name einer aus 4 Byte bestehenden Position mit der Adresse eines Speicherbereichs mit einer Länge von bis zu 246 Byte, der die Begrüßungsnachricht enthält.

(Rn) Ein Register mit der Adresse eines Speicherbereichs mit einer Länge von bis zu 246 Byte, der die Begrüßungsnachricht enthält.

RESPONSE- und REASON-Werte für SET_SYSTEM

<i>RESPONSE</i>	<i>REASON</i>
OK	----
INVALID	INVALID_FUNCTION
EXCEPTION	AKP_SIZE_ERROR
	NO_KEYPOINT
DISASTER	SET_FAILED
PURGED	----

Kapitel 14. XPI-Funktionen für Speichersteuerung

Die XPI stellt sieben Funktionen für Speichersteuerung bereit. Dabei handelt es sich um die Aufrufe GETMAIN, FREEMAIN, INQUIRE_ELEMENT_LENGTH und INQUIRE_TASK_STORAGE des Makros DFHSMCMX sowie die Aufrufe INQUIRE_ACCESS, INQUIRE_SHORT_ON_STORAGE und SWITCH_SUBSPACE des Makros DFHMSRX.

DFHSMCMX-Aufrufe können in keinem Exitprogramm verwendet werden, das an einem globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Dispatcherdomäne
- Speicherauszugsdomäne
- Monitordomäne
- Statistikdomäne
- Programm für transiente Daten (TDP)

Aufruf GETMAIN

Mit dem Aufruf GETMAIN wird ein Speicherelement für Ihr Exitprogramm angefordert. Sie können eine bestimmte Speicherklasse anfragen und die Initialisierung des Elements mit einem Einzelbytwert anfordern.

Mit einem Aufruf GETMAIN angeforderter Speicher in den folgenden Klassen wird von CICS freigegeben, wenn der bei der Speicherübernahme verwendete Tasksteuerbereich (Task Control Area, TCA) beendet wird:

- CICS
- CICS24
- USER
- USER24

Im Gegensatz dazu wird Speicher in den folgenden Klassen bei Taskende nicht automatisch freigegeben. Dieser Speicher muss mit dem Aufruf FREEMAIN freigegeben werden.

- SHARED_CICS
- SHARED_CICS24
- SHARED_USER
- SHARED_USER24
- TERMINAL

Außerdem ist zu beachten, dass einige Benutzerexits möglicherweise über Systemtasks aufgerufen werden. In diesen Fällen erfolgt die Speicherfreigabe erst beim nächsten Beenden von CICS. Geben Sie deshalb alle über einen Aufruf GETMAIN angeforderten Speicherbereiche mit dem Aufruf FREEMAIN frei, sobald die Speicherbereiche nicht mehr benötigt werden.

GETMAIN

```
DFHSMCX [CALL,]  
  [CLEAR,]  
  [IN,  
  FUNCTION(GETMAIN),  
  GET_LENGTH(name4 | (Rn) | expression),  
  STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|  
    SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL),  
  SUSPEND(NO|YES),  
  [INITIAL_IMAGE(name1 | literalconst),]  
  [TCTTE_ADDRESS(name4 | (Ra)),]]  
  [OUT,  
  ADDRESS(name4 | (Rn) | *),  
  RESPONSE(name1 | *),  
  REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ADDRESS(name4 | (Rn) | *)

Gibt die Adresse des über den Aufruf erhaltenen Speichers zurück.

name4 Der Name eines Vollworts zum Speichern der erhaltenen Speicheradresse.

(Rn) Ein Register, das auf den erhaltenen Speicher verweist.

***** Die Parameterliste selbst (Name: SMMC_ADDRESS) wird zum Beibehalten der Adresse verwendet.

GET_LENGTH(name4 | (Rn) | expression)

Gibt die gewünschte Anzahl an Speicherbyte auf eine der folgenden Weisen an:

name4 Der Name eines Vollworts, das die Anzahl der Bytes im Binärformat angibt.

(Rn) Ein Register, das die Anzahl der Bytes im Binärformat enthält.

expression

Ein gültiger Assemblerausdruck, z. B. eine Zahl, ein symbolischer Ausdruck oder eine Kombination aus beidem.

Beim Anfordern von TERMINAL-Speicher enthält die von Ihnen angegebene Länge nicht den Speicherabrechnungsbereich (Storage Accounting Area, SAA). Die maximale Länge, die Sie angeben können, beträgt 65.515 Byte. Von der CICS-Speicherverwaltung wird ein SAA mit einer Länge von 8 Byte hinzugefügt. Die vom XPI-Aufruf zurückgegebene Adresse gibt den Anfang des SAA an.

Wenn Sie Speicher in den Klassen CICS24, CICS, USER24, USER, SHARED_CICS24, SHARED_CICS, SHARED_USER24 oder SHARED_USER anfordern, müssen Sie lediglich die von Ihrem Programm benötigte Länge angeben. Die zurückgegebene Adresse ist der Anfang Ihres Datenspeichers. Die maximale Größe des Speichers für diese Speicherklassen entspricht der Größe des dynamischen Speicherbereichs (Dynamic Storage Area, DSA), von dem aus die Zuordnung erfolgt.

INITIAL_IMAGE(name1 | literalconst)

Gibt das Initialisierungsmuster an. Sie können den erhaltenen Speicher z. B. mit binären Nullen belegen.

name1 Der Name einer Position, an der das aus einem Byte bestehende Initialisierungsmuster gespeichert wird.

literalconst

Eine Zahl in Form eines Literals (z. B. B'00000000', X'FF', X'FC', "0" oder ein Gleichsetzungssymbol mit einem entsprechenden Wert).

STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL)

Gibt die Klasse des Speichers an, auf den sich der Aufruf bezieht. Eine Auflistung der Werte, die Sie über diese Option zuordnen können, sowie die Speichertypen zu den Werten finden Sie in Tabelle 6.

Tabelle 6. CICS-Speicherklassen

STORAGE_CLASS	Speichertyp
CICS	Tasklaufzeitbezogener CICS-Schlüsselspeicher oberhalb von 16 MB und unterhalb von 2 GB.
CICS24	Tasklaufzeitbezogener CICS-Schlüsselspeicher unterhalb von 16 MB.
SHARED_CICS	Gemeinsam genutzter CICS-Schlüsselspeicher oberhalb von 16 MB und unterhalb von 2 GB.
SHARED_CICS24	Gemeinsam genutzter CICS-Schlüsselspeicher unterhalb von 16 MB.
SHARED_USER	Gemeinsam genutzter Benutzerschlüsselspeicher oberhalb von 16 MB und unterhalb von 2 GB.
SHARED_USER24	Gemeinsam genutzter Benutzerschlüsselspeicher unterhalb von 16 MB.
TERMINAL	Diese Speicherklasse hat einen SAA mit einer Länge von 8 Byte.
USER	Tasklaufzeitbezogener Benutzerschlüsselspeicher oberhalb von 16 MB und unterhalb von 2 GB.
USER24	Tasklaufzeitbezogener Benutzerschlüsselspeicher unterhalb von 16 MB.

Sie müssen in einer Anforderung GETMAIN eine Speicherklasse angeben. Bei der Anforderung FREEMAIN ist die Speicherklasse ein optionaler Parameter und der von Ihnen angegebene Wert wird nicht von CICS überprüft.

SUSPEND(YES|NO)

Gibt an, ob die Anforderung ausgesetzt werden soll, wenn der über die Option GET_LENGTH angeforderte Speicher nicht verfügbar ist.

TCTTE_ADDRESS(name4 | (Ra))

Gibt die Adresse des Terminaleintrags in der Terminalsteuertabelle (Terminal Control Table Terminal Entry, TCTTE) an. Bei GETMAIN-Anforderungen müssen Sie diese Option festlegen, wenn Sie die Klasse TERMINAL für die Option STORAGE_CLASS angeben. Bei FREEMAIN-Anforderungen müssen Sie diese Option festlegen, wenn Sie Speicher der Klasse TERMINAL freigeben.

Anmerkung: Überprüfen Sie vor dem Anfordern von Speicher der Klasse TERMINAL das TCAFCI-Bit 7 (Task Control Area Facility Control Indicator), um sicherzustellen, dass dieser TCA unter einem Terminal ausgeführt wird.

name4 Der Name eines Vollworts, das die Adresse enthält.

(Ra) Ein Register, das auf den Terminaleintrag in der Terminalsteuertabelle verweist.

RESPONSE- und REASON-Werte für GETMAIN

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INSUFFICIENT_STORAGE
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung:

1. Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.
2. INSUFFICIENT_STORAGE wird zurückgegeben, wenn die Anforderung GETMAIN mit SUSPEND(NO) angegeben wurde und der Speicher nicht zum Erfüllen der Anforderung ausreicht.
3. PURGED wird zurückgegeben, wenn die Anforderung GETMAIN mit SUSPEND(YES) angegeben wurde, der Speicher nicht zum Erfüllen der Anforderung ausreicht und die Task gelöscht wurde.

Aufruf FREEMAIN

Mit dem Aufruf FREEMAIN wird ein Speicherbereich freigegeben, der Ihrem Exitprogramm aktuell zugeordnet ist.

FREEMAIN

```
DFHSMCX [CALL,]  
  [CLEAR,]  
  [IN,  
    FUNCTION(FREEMAIN),  
    ADDRESS(name4 | (Rn) | *),  
    [STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|  
      SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL),]  
    [TCTTE_ADDRESS(pointer),]]  
  [OUT,  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Eine Erläuterung der Optionen finden Sie in „Aufruf GETMAIN“ auf Seite 105.

RESPONSE- und REASON-Werte für FREEMAIN

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	----
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Aufruf INQUIRE_ACCESS

INQUIRE_ACCESS gibt den Zugriffsschlüssel eines Speicherelements zurück, das durch Startadresse und Länge angegeben ist. Ist das Element nicht vollständig in einem einzigen dynamischen CICS-Speicherbereich (Dynamic Storage Area, DSA) enthalten, gibt CICS eine Ausnahmeantwort zurück.

INQUIRE_ACCESS

```
DFHMSRX [CALL,]  
  [CLEAR,]  
  [IN,  
    FUNCTION(INQUIRE_ACCESS),  
    ELEMENT_ADDRESS(name4 | (Rn) | *),  
    ELEMENT_LENGTH(name4 | (Rn) | *),]  
  [OUT,  
    ACCESS(CICS | READ_ONLY | USER),  
    RESPONSE(name1 | *),  
    REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

ACCESS(CICS|READ_ONLY|USER)

Gibt den Zugriffsschlüssel des Speicherelements zurück.

CICS CICS-Schlüssel.

READ_ONLY

Schreibgeschützter Speicher.

USER Benutzerschlüssel.

ELEMENT_ADDRESS(name4 | (Rn) | *)

Gibt die Adresse des Speicherelements an.

ELEMENT_LENGTH(name4 | (Rn) | *)

Gibt die Länge des Speicherelements in Byte an. Eine Länge von Null wird als Länge von 1 Byte behandelt.

RESPONSE- und REASON-Werte für INQUIRE_ACCESS

RESPONSE	REASON
OK	----
EXCEPTION	INVALID_ELEMENT
DISASTER	----
INVALID	----
KERNERROR	----

Aufruf INQUIRE_ELEMENT_LENGTH

Mit dem Aufruf INQUIRE_ELEMENT_LENGTH können Sie die Adresse eines beliebigen Teils eines Elements des Tasklaufzeitspeichers übergeben und von CICS Startadresse und Länge des Speicherelements abrufen, das die übergebene Adresse enthält.

INQUIRE_ELEMENT_LENGTH

```
DFHSMCX [CALL,]  
  [CLEAR,]  
  [IN,  
    FUNCTION (INQUIRE_ELEMENT_LENGTH),  
    ADDRESS (name4 | (Rn) | *),]
```

```
[OUT,
ELEMENT_ADDRESS(name4 | (Rn) | *),
ELEMENT_LENGTH(name4 | (Rn) | *),
RESPONSE (name1 | *),
REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

ADDRESS(name4 | (Rn) | *)

Gibt eine Adresse in einem Element des Tasklaufzeitspeichers für die aktuelle Task an.

Von CICS werden Adressen akzeptiert, die auf die führenden und abschließenden Prüfzonen verweisen, in denen bestätigt wird, dass es sich um gültige Adressen für das Element des Speichers handelt, für das Sie Informationen anfragen.

ELEMENT_ADDRESS(name4 | (Rn) | *)

Gibt die Startadresse des Elements des Tasklaufzeitspeichers zurück, auf das über den Parameter ADDRESS verwiesen wird. Die zurückgegebene Startadresse beinhaltet **nicht** die führende Prüfzone.

ELEMENT_LENGTH(name4 | (Rn) | *)

Gibt die Länge des Elements des Tasklaufzeitspeichers zurück, auf das über den Parameter ADDRESS verwiesen wird. Die zurückgegebene Länge beinhaltet **nicht** die führende und die abschließende Prüfzone.

RESPONSE- und REASON-Werte für INQUIRE_ELEMENT_LENGTH

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INVALID_ADDRESS
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf INQUIRE_SHORT_ON_STORAGE

Mit dem Aufruf INQUIRE_SHORT_ON_STORAGE können Sie ermitteln, ob der 64-Bit-Speicher, der Speicher oberhalb der 16-MB-Linie unter 2 GB oder der Speicher unterhalb der 16-MB-Linie für CICS nicht ausreicht.

INQUIRE_SHORT_ON_STORAGE

```
DFHMSRX [CALL,]
        [CLEAR,]
        [IN,
        FUNCTION(INQUIRE_SHORT_ON_STORAGE),]
        [OUT,
        SOS_ABOVE_THE_BAR(NO|YES),
        SOS_ABOVE_THE_LINE(NO|YES),
        SOS_BELOW_THE_LINE(NO|YES),
        RESPONSE (name1 | *),
        REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

SOS_ABOVE_THE_BAR(NO|YES),

Gibt YES zurück, wenn der 64-Bit-Speicher (oberhalb der Grenze) für CICS nicht ausreicht. NO wird zurückgegeben, wenn der Speicher ausreicht.

SOS_ABOVE_THE_LINE(NO|YES),

Gibt YES zurück, wenn der Speicher oberhalb der 16-MB-Grenze unter 2 GB für CICS nicht ausreicht. NO wird zurückgegeben, wenn der Speicher ausreicht.

SOS_BELOW_THE_LINE(NO|YES),

Gibt YES zurück, wenn der Speicher unterhalb der 16-MB-Grenze für CICS nicht ausreicht. NO wird zurückgegeben, wenn der Speicher ausreicht.

RESPONSE- und REASON-Werte für INQUIRE_SHORT_ON_STORAGE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	----
KERNERROR	----

Aufruf INQUIRE_TASK_STORAGE

Mit dem Aufruf INQUIRE_TASK_STORAGE können Sie Details zu allen Elementen des Tasklaufzeitspeichers einer Task anfordern. Sie können die Transaktionsnummer der Task explizit im Aufruf angeben oder auf die Standardeinstellung zurückgreifen (aktuelle Task).

INQUIRE_TASK_STORAGE

```
DFHSMCX [CALL,]  
        [CLEAR,]  
        [IN,  
        FUNCTION (INQUIRE_TASK_STORAGE),  
        [TRANSACTION_NUMBER(name4 | (Rn) | *),]  
        ELEMENT_BUFFER(buffer-descriptor),  
        LENGTH_BUFFER(buffer-descriptor),]  
        [OUT,  
        NUMBER_OF_ELEMENTS(name4 | (Rn) | *),  
        RESPONSE (name1 | *),  
        REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

ELEMENT_BUFFER(buffer-descriptor)

Definiert Adresse und Länge eines Puffers, in den CICS eine Liste der Startadressen der Elemente in dem Tasklaufzeitspeicher zurückgibt, der zu der angegebenen Task oder zur aktuellen Task (Standardeinstellung) gehört.

Die zurückgegebenen Startadressen beinhalten **nicht** die führende Prüfzone. Eine Beschreibung zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

LENGTH_BUFFER(buffer-descriptor)

Definiert Adresse und Länge eines Puffers, in den CICS eine Liste mit der Länge der Elemente in dem Tasklaufzeitspeicher zurückgibt, der zu der angegebenen Task oder zur aktuellen Task (Standardeinstellung) gehört. Die zurückgegebene Länge beinhaltet **nicht** die führende und die abschließende Prüfzone.

Eine Beschreibung zu Pufferdeskriptoren finden Sie im Abschnitt XPI syntax.

NUMBER_OF_ELEMENTS(name4 | (Rn) | *)

Gibt die Anzahl der Einträge in den beiden Puffern (ELEMENT_BUFFER und LENGTH_BUFFER) jeweils als Vollwort-Binärwert zurück.

TRANSACTION_NUMBER(name4 | (Rn) | *)

Gibt die Transaktionsnummer der Task, zu der der Speicher gehört, als aus 4 Byte bestehenden, gepackten Dezimalwert an.

Wenn Sie die Transaktionsnummer (Tasknummer) nicht angeben, setzt CICS voraus, dass es um die aktuelle Task geht.

RESPONSE- und REASON-Werte für INQUIRE_TASK_STORAGE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	INSUFFICIENT_STORAGE
	NO_TRANSACTION_ENVIRONMENT
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Aufruf SWITCH_SUBSPACE

Der Aufruf SWITCH_SUBSPACE bewirkt, dass CICS von einem Unterbereich zum Basisspeicherbereich wechselt, wenn die jeweilige Task nicht bereits im Basisbereich ausgeführt wird. Wird die Task bereits im Basisspeicherbereich ausgeführt, wird der Aufruf vom Speichermanager ignoriert.

Die Funktion kann von globalen Benutzerexitprogrammen verwendet werden, die die Steuerung in einem Unterbereich erhalten und aus bestimmten Gründen in den Basisspeicherbereich wechseln müssen.

SWITCH_SUBSPACE

```
DFHMSRX [CALL,]
        [CLEAR,]
        [IN,
        FUNCTION(SWITCH_SUBSPACE),
        SPACE(BASESPACE),]
        [OUT,
        RESPONSE (name1 | *),
        REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

SPACE(BASESPACE)

Gibt an, dass CICS die Task, von der der Aufruf abgesetzt wird, auf den Basisspeicherbereich umstellen soll, wenn die Task aktuell in einem Unterbereich ausgeführt wird. Dies ermöglicht der Task Lese- und Schreibzugriffe auf den tasklaufzeitbezogenen Benutzerschlüsselspeicher einer anderen Task.

RESPONSE- und REASON-Werte für SWITCH_SUBSPACE

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	----
KERNERROR	----

Kapitel 15. XPI-Funktion für Tracesteuerung

Die XPI stellt eine Funktion für Tracesteuerung bereit. Dabei handelt es sich um den DFHTRPTX-Aufruf TRACE_PUT.

Einschränkung: DFHTRPTX-Aufrufe können in keinem Exitprogramm verwendet werden, das über einen globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Dispatcherdomäne
- Speicherauszugsdomäne
- Monitordomäne
- Statistikdomäne
- Programm für transiente Daten (TDP)

Aufruf TRACE_PUT

Mit dem Aufruf TRACE_PUT wird ein Traceeintrag in die aktiven Traceziele geschrieben.

Geben Sie einen Aufruf TRACE_PUT nur aus, wenn UEPTRON anzeigt, dass die Traceerstellung für die Funktion, die das Exitprogramm enthält, aktiv ist (siehe dazu UEPTRON in der DFHUEPAR-Parameterliste). Möglicherweise ziehen Sie es vor, Einträge im Ausnahmebedingungs-Trace für schwerwiegende Fehler zu erstellen, ohne UEPTRON zu testen.

Wenn Sie mit TRACE_PUT Einträge für den Ausnahmebedingungs-Trace schreiben, kennzeichnen Sie diese Einträge, sodass diese Einträge vom Dienstprogramm für die Traceformatierung als Ausnahmebedingungs-Traceeinträge hervorgehoben werden. Geben Sie zur Kennzeichnung eines Eintrags für den Ausnahmebedingungs-Trace die Literalzeichenfolge 'USEREXC' im Blockdeskriptorfeld DATA1 für den DFHTRPTX-Aufruf an.

TRACE_PUT

```
DFHTRPTX [CALL,]
          [CLEAR,]
          [IN,
           FUNCTION(TRACE_PUT),
           POINT_ID(literalconst | name2 | (Rn)),
           [DATA1(block-descriptor),]
           [DATA2(block-descriptor),]
           [DATA3(block-descriptor),]
           [DATA4(block-descriptor),]
           [DATA5(block-descriptor),]
           [DATA6(block-descriptor),]
           [DATA7(block-descriptor),]
           [RETURN_ADDR(expression | name4 | (Ra)),]]
          [OUT,
           RESPONSE(name1 | *)]
```

Dieser Befehl ist threadsicher.

DATA_n(block-descriptor)

Gibt bis zu sieben Bereiche an, die in den Datenabschnitt des Traceeintrags eingeschlossen werden sollen. Eine Beschreibung gültiger Blockdeskriptoren fin-

den Sie im Abschnitt XPI syntax. Wenn Sie einen gegebenen Blockdeskriptor DATA n angeben, müssen DATA1 bis DATA($n-1$) vor DATA n codiert sein. Die angegebenen DATA-Elemente werden in der Traceausgabe in der angegebenen Reihenfolge, d. h. von DATA1 bis DATA n , ausgegeben. Vor dem Datenfeld selbst wird ein Feld mit einer Länge von 2 Byte ausgegeben. Die maximale Gesamtlänge der Daten, für die in einem einzigen Aufruf ein Trace erstellt werden kann, beträgt 4000 Byte. Die Gesamtlänge der Datenfelder und der zugehörigen Felder mit einer Länge von 2 Byte muss innerhalb dieser Begrenzung liegen.

POINT_ID(literalconst|name2|(Rn))

Gibt die Traceeinträge an, die als Ergebnis der Anforderung erstellt werden. Jeder Aufruf TRACE_PUT in einer aufrufenden Domäne sollte einen eindeutigen Wert für POINT_ID angeben. Auf diese Weise können Sie den Ursprung eines Traceaufrufs ermitteln, wenn Sie einen formatierten Trace untersuchen. Die Werte für POINT_ID müssen im Bereich der Dezimalwerte von 256 bis 511 (X'100' bis X'1FF') liegen. Dieser Bereich ist für Benutzerexits reserviert und wird nicht in CICS-Modulen verwendet.

literalconst

Eine Zahl in Form eines Literals, das die ID enthält.

name2 Der Name eines aus 2 Byte bestehenden Felds, das die ID enthält.

(Rn) Ein Register, bei dem die beiden niedrigstwertigen Bytes die ID enthalten.

RETURN_ADDR(expression|name4|(Ra))

Gibt den Wert an, der im Feld für die Rückkehradresse des Traceeintrags erscheint.

expression

Ein gültiger Assemblerausdruck, der die Adresse ergibt.

name4 Der Name eines Vollworts mit der Adresse.

(Ra) Ein Register mit der Adresse.

Kapitel 16. XPI-Funktionen für Transaktionsverwaltung

Die XPI stellt Funktionen für die Transaktionsverwaltung bereit, mit denen Sie Informationen zu Transaktionen anfragen und bestimmte Transaktionsparameter festlegen können.

Aufruf INQUIRE_CONTEXT

Mit dem Aufruf INQUIRE_CONTEXT werden Informationen zu der Umgebung, in der eine Transaktion ausgeführt wird, zurückgegeben. Der Aufruf stellt insbesondere Informationen für Transaktionen in einer Umgebung mit Brücke bereit.

INQUIRE_CONTEXT

```
DFHBR1QX [CALL,]  
          [CLEAR,]  
          [IN,  
           FUNCTION(INQUIRE_CONTEXT),]  
          [OUT,  
           [CONTEXT(byte1),]  
           [BRIDGE_TRANSACTION_ID(name4),]  
           [BRIDGE_EXIT_PROGRAM(name8),]  
           [BFB_TOKEN(name4),]  
           [BRXA_TOKEN(name4),]  
           [FACILITYTOKEN(name8),]  
           [START_TYPE(byte1),]  
           RESPONSE (name1 | *),  
           REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

BFB_TOKEN(name4)

Gibt einen Zeiger zurück, der die Adresse der Bridgefunktion enthält, die von dieser Task verwendet wird. Obwohl die Bridgefunktion kein reales Terminal ist, wird sie durch eine Datenstruktur dargestellt, die das gleiche Format wie ein TCTTE (Terminal Control Table Terminal Entry) hat, und kann mit Hilfe des DFHTCTTE-Pseudoabschnitts zugeordnet werden. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

Anmerkung: In früheren Releases von CICS wurde dieses Feld als BRIDGE_FACILITY_TOKEN bezeichnet.

name4 Der Name einer aus 4 Byte bestehenden Position für das zu empfangende Token.

BRIDGE_EXIT_PROGRAM(name8)

Gibt den Namen des Brückenexitprogramms zurück, das von dieser Task verwendet wird. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

name8 Der Name einer aus 8 Byte bestehenden Position für den zu empfangenden Namen des Brückenexitprogramms.

BRIDGE_TRANSACTION_ID(name4)

Gibt den Namen der Überwachungstransaktion für die Brücke zurück, über die der Befehl START BREXIT TRANSID zum Starten der Transaktion abgesetzt wurde. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

name4 Der Name einer aus 4 Byte bestehenden Position, an der der Name der Überwachungstransaktion für die Brücke empfangen wird.

BRXA_TOKEN(name4)

Gibt ein Token mit der Adresse des Brückenexitbereichs zurück (BRXA), der von der Task verwendet wird. Der Brückenexitbereich kann nicht auf die Link3270-Brücke angewendet werden (START_TYPE=BRIQ_LINK). Das BRXA-Format wird über die DFHBRARx-Copybooks definiert. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

name4 Der Name einer aus 4 Byte bestehenden Position für das zu empfangende Token.

CONTEXT(byte1)

Gibt an einer aus 1 Byte bestehenden Position (*byte1*) den Typ der Umgebung zurück, in dem die Transaktion ausgeführt wird.

BRIDGE

Eine Benutzertransaktion, die unter Verwendung einer Brücke gestartet wurde.

BREXIT

Ein Brückenexitprogramm.

NORMAL

Eine Transaktion, die nicht in einer Brückenumgebung ausgeführt wird.

FACILITYTOKEN(name8)

Gibt das Funktionstoken (eine der Bridgefunktion zugeordnete Kennung) zurück. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

name8 Der Name einer aus 8 Byte bestehenden Position für das zu empfangende Funktionstoken.

START_TYPE(byte1)

Gibt an einer aus 1 Byte bestehenden Position (*byte1*) zurück, wie die Link3270-Brücke gestartet wurde. Wird für CONTEXT der Wert NORMAL zurückgegeben, ist der Inhalt dieses Feldes irrelevant.

BRIQ_START

Die Brücke wurde mit START BREXIT gestartet.

BRIQ_LINK

Die Bridge wurde mit Hilfe des Link3270-Mechanismus gestartet.

RESPONSE- und REASON-Werte für INQUIRE_CONTEXT

RESPONSE	REASON
OK	----
DISASTER	ABEND
	LOOP
INVALID	----
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
KERNERROR	----

Aufruf INQUIRE_DTRTRAN

Mit dem Aufruf INQUIRE_DTRTRAN wird der Name der DTR-Transaktionsdefinition (DTR = Dynamic Transaction Routing) zurückgegeben.

Die DTR-Transaktionsdefinition stellt allgemeine Attribute für dynamisch weiterzuleitende Transaktionen bereit, die keine bestimmte Transaktionsdefinition aufweisen. Die Transaktionsdefinition wird im Systeminitialisierungsparameter DTRTRAN angegeben. Von CICS wird die Standarddefinition CRTX bereitgestellt.

INQUIRE_DTRTRAN

```
DFHXMSRX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(INQUIRE_DTRTRAN),]
          [OUT,
          DTRTRAN(name4),
          RESPONSE (name1 | *),
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

DTRTRAN(name4)

Gibt den Namen der DTR-Transaktionsdefinition für Routing-Transaktionen zurück, die nicht durch eine explizite Transaktionsressourcendefinition definiert sind.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende DTR-Transaktionsdefinition. Wurde für den Systeminitialisierungsparameter DTRTRAN 'NO' angegeben, stellt 'NO' auch den Wert in diesem Feld dar.

RESPONSE- und REASON-Werte für INQUIRE_DTRTRAN

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	ABEND
	LOGIC_ERROR
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	----
PURGED	----

Aufruf INQUIRE_MXT

Die Funktion INQUIRE_MXT wird für den Makroaufruf DFHXMSRX bereitgestellt. Diese Funktion dient dazu, den aktuellen Wert des Parameters MXT bereitzustellen.

INQUIRE_MXT

```
DFHXMSRX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(INQUIRE_MXT),]
          [OUT,
          CURRENT_ACTIVE(name4 | (Rn) ),
          MXT_LIMIT(name4 | (Rn)),
          MXT_QUEUED(name4 | (Rn) ),
          TCLASS_QUEUED(name4 | (Rn) ),
          RESPONSE (name1 | *),
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

CURRENT_ACTIVE(name4 | (Rn))

Gibt die aktuelle Anzahl aller aktiven Benutzertasks zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle Anzahl der aktiven Benutzertasks als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuelle Anzahl der aktiven Benutzertasks als Binärwert empfangen wird.

MXT_LIMIT(name4 | (Rn))

Gibt die aktuelle Anzahl des Parameters MXT zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuell zulässige maximale Anzahl aller Benutzertasks als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuell zulässige maximale Anzahl aller Tasks als Binärwert empfangen wird.

MXT_QUEUED(name4 | (Rn))

Gibt die aktuelle Anzahl der Benutzertransaktionen zurück, die sich in einer Warteschlange befinden, da die maximale Anzahl der Tasks (MXT) erreicht ist.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle Anzahl der Benutzertasks, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuelle Anzahl der Benutzertasks, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

TCLASS_QUEUED(name4 | (Rn))

Gibt die aktuelle Anzahl aller Transaktionen in einer Warteschlange für die Transaktionsklassenzugehörigkeit zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle Anzahl der Transaktionsklassenmitglieder, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuelle Anzahl der Transaktionsklassenmitglieder, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

RESPONSE- und REASON-Werte für INQUIRE_MXT

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	LOGIC_ERROR
	ABEND
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	----
PURGED	----

Aufruf INQUIRE_TCLASS

Die Funktion INQUIRE_TCLASS wird für den Makroaufruf DFHXMCLX bereitgestellt. Diese Funktion dient dazu, aktuelle Informationen zur angegebenen Transaktionsklasse (TCLASS) bereitzustellen.

INQUIRE_TCLASS

```
DFHXMCLX [CALL,]  
        [CLEAR,]  
        [IN,  
        FUNCTION(INQUIRE_TCLASS),  
        INQ_TCLASS_NAME(name8 | string | 'string'),]  
        [OUT,  
        [CURRENT_ACTIVE(name4 | (Rn)),]  
        [CURRENT_QUEUED(name4 | (Rn)),]  
        [MAX_ACTIVE(name4 | (Rn)),]  
        [PURGE_THRESHOLD(name4 | (Rn)),]  
        RESPONSE (name1 | *),  
        REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

CURRENT_ACTIVE(name4 | (Rn))

Gibt die aktuelle Anzahl der aktiven Benutzertasks in der Transaktionsklasse zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle Anzahl der aktiven Benutzertasks für diese Transaktionsklasse als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuelle Anzahl der aktiven Benutzertasks für diese Transaktionsklasse als Binärwert empfangen wird.

CURRENT_QUEUED(name4 | (Rn))

Gibt die aktuelle Anzahl der Benutzertasks, die sich in einer Warteschlange befinden, zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle Anzahl der aktiven Benutzertasks für die Transaktionsklasse, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

(Rn) Ein Register, in dem die aktuelle Anzahl der Benutzertasks, die sich in einer Warteschlange befinden, als Binärwert empfangen wird.

INQ_TCLASS_NAME(name8 | string | 'string')

Gibt den Namen der Transaktionsklasse für die Anfrage an.

name8 Der Name einer aus 8 Byte bestehenden Position, an der der Name der Transaktionsklasse, auf die sich die Anfrage bezieht, empfangen wird.

string Eine Zeichenfolge ohne eingebettete Leerzeichen zur Kennzeichnung der Transaktionsklasse.

'string'

Eine in Anführungszeichen gesetzte Zeichenfolge zur Kennzeichnung der Transaktionsklasse. Die Länge der Zeichenfolge ist auf 8 Byte festgelegt und wird erreicht, indem Leerzeichen innerhalb der Anführungszeichen hinzugefügt werden.

MAX_ACTIVE(name4 | (Rn))

Gibt die aktuelle maximale Anzahl aktiver Tasks zurück, die für die Transaktionsklasse zulässig sind.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die aktuelle maximale Anzahl der aktiven Tasks als Binärwert empfangen wird, die für die Transaktionsklasse zu diesem Zeitpunkt zulässig sind.

(Rn) Ein Register, in dem die aktuelle maximale Anzahl der aktiven Tasks als Binärwert empfangen wird, die zu diesem Zeitpunkt für die Transaktionsklasse zulässig sind.

PURGE_THRESHOLD(name4 | (Rn))

Gibt den Bereinigungsschwellenwert für die Transaktionsklasse zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der der aktuelle Bereinigungsschwellenwert für diese Transaktionsklasse als Binärwert empfangen wird.

(Rn) Ein Register, in dem der aktuelle Bereinigungsschwellenwert für diese Transaktionsklasse als Binärwert empfangen wird.

RESPONSE- und REASON-Werte für INQUIRE_TCLASS

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	LOGIC_ERROR
INVALID	----
EXCEPTION	UNKNOWN_CLASS

Aufruf INQUIRE_TRANDEF

Die Funktion INQUIRE_TRANDEF wird für den Makroaufruf DFHXMIDX bereitgestellt. Diese Funktion ermöglicht es Ihnen, Informationen zur angegebenen Transaktionsdefinition abzurufen. Insgesamt entspricht dieser Funktionsaufruf, von einigen Unterschieden abgesehen, dem Befehl EXEC CICS INQUIRE TRANSACTION.

INQUIRE_TRANDEF

```
DFHXMIDX [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(INQUIRE_TRANDEF),
          INQ_TRANSACTION_ID(name4 | string | 'string'),]
          [OUT,
          [BEXIT(name8),]
          [CMDSEC(name1),]
          [DTIMEOUT(name4 | (Rn)),]
          [DUMP(name1),]
          [DYNAMIC(name1),]
          [INDOUBT(name1),]
          [INDOUBT_WAIT(name1),]
          [INDOUBT_WAIT_TIME(name4),]
          [INITIAL_PROGRAM(name8),]
          [ISOLATE(name1),]
          [LOCAL_QUEUEING(name1),]
          [OTSTIMEOUT(name4 | (Rn)),]
          [PARTITIONSET(name1),]
          [PARTITIONSET_NAME(name8),]
          [PROFILE_NAME(name8),]
          [REMOTE(name1),]
          [REMOTE_NAME(name8),]
          [REMOTE_SYSTEM(name4),]
          [RESSEC(name1),]
          [RESTART(name1),]
          [ROUTABLE_STATUS(ROUTABLE|NOT_ROUTABLE),]
          [RUNAWAY_LIMIT(name4 | (Rn)),]
          [SHUTDOWN(name1),]
          [SPURGE(name1),]
          [STATUS(name1),]
          [STORAGE_CLEAR(name1),]
          [STORAGE_FREEZE(name1),]
          [SYSTEM_ATTACH(name1),]
          [SYSTEM_RUNAWAY(name1),]
```



```

[TASKDATAKEY(name1),]
[TASKDATALOC(name1),]
[TCLASS(name1),[TCLASS_NAME(name8),]]
[TPURGE(name1),]
[TRACE(name1),]
[TRAN_PRIORITY(name4 | (Rn)),]
[TRAN_ROUTING_PROFILE(name8),]
[TRANSACTION_ID(name4),]
[TWASIZE(name4 | (Rn)),]
RESPONSE (name1 | *),
REASON (name1 | *)]

```

Dieser Befehl ist threadsicher.

Die folgenden Parameterbeschreibungen enthalten kurze Erläuterungen zu den Werten, die von einem Aufruf INQUIRE_TRANDEF zurückgegeben werden können. Eine ausführlichere Erläuterung zu einigen Parametern finden Sie in den Parameterbeschreibungen für die Transaktionsressourcendefinition im Abschnitt TRANSACTION attributes.

BREXIT(name8)

Gibt den Namen des Standardbrückenexitprogramms zurück, das für die genannte Transaktion angegeben ist. Ist kein Brückenexit angegeben, werden Leerzeichen zurückgegeben.

name8 Der Name einer aus 8 Byte bestehenden Position für den zu empfangenden Namen des Brückenexitprogramms.

CMDSEC(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob eine Befehlssicherheitsprüfung für die Transaktion erforderlich ist.

XMxD_YES

Eine Befehlssicherheitsprüfung ist erforderlich.

XMxD_NO

Eine Befehlssicherheitsprüfung ist nicht erforderlich.

DTIMEOUT(name4)

Gibt das Deadlockzeitlimit für die Transaktion zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der das Deadlockzeitlimit als Binärwert empfangen wird.

(Rn) Ein Register, in dem das Deadlockzeitlimit als Binärwert empfangen wird.

Dabei ist zu beachten, dass ein Nullwert bedeutet, dass in der Transaktionsressourcendefinition DTIMOUT(NO) angegeben ist.

DUMP(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob von CICS bei einer abnormalen Beendigung der Transaktion ein Transaktionsspeicherauszug erstellt wird.

XMxD_YES

Es ist ein Transaktionsspeicherauszug erforderlich.

XMxD_NO

Ein Transaktionsspeicherauszug ist nicht erforderlich.

DYNAMIC(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion für dynamisches Transaktionsrouting definiert ist.

XMxD_YES

Die Transaktion wird dynamisch an ein fernes CICS-System weitergeleitet.

XMxD_NO

Die Transaktion wird nicht dynamisch weitergeleitet.

INDOUBT(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der die Aktion angibt, die erfolgen soll, wenn die CICS-Region fehlschlägt oder die Verbindung zum zugehörigen Koordinator unterbrochen wird, während eine Arbeitseinheit sich im unbestätigten Zeitraum befindet. Die Aktion basiert auf dem Attribut ACTION der Transaktionsressourcendefinition.

Die Aktion richtet sich nach den in INDOUBT_WAIT und INDOUBT_WAIT_TIME zurückgegebenen Werten. Gibt INDOUBT_WAIT den Wert XMxD_YES zurück, erfolgt die Aktion erst, wenn die in INDOUBT_WAIT_TIME zurückgegebene Zeit abgelaufen ist.

XMxD_BACKOUT

Von der Transaktion an wiederherstellbaren Ressourcen vorgenommene Änderungen werden zurückgesetzt.

XMxD_COMMIT

Von der Transaktion an wiederherstellbaren Ressourcen vorgenommene Änderungen werden festgeschrieben.

INDOUBT_WAIT(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, wie eine Arbeitseinheit (Unit Of Work, UOW) antwortet, wenn die Arbeitseinheit den Status 'Unbestätigt' (Indoubt) aufweist und währenddessen ein Fehler auftritt.

XMxD_NO

Die Arbeitseinheit wartet nicht auf die anstehende Wiederherstellung nach dem Fehler. Von CICS wird sofort die im Attribut ACTION der Transaktionsdefinition angegebene Aktion ausgeführt, unabhängig davon, um welche Aktion es sich handelt.

XMxD_YES

Die Arbeitseinheit wartet auf die anstehende Wiederherstellung nach dem Fehler, um zu ermitteln, ob wiederherstellbare Ressourcen zurückgesetzt oder festgeschrieben werden.

INDOUBT_WAIT_TIME(name4)

Gibt die Anzahl der Minuten zurück, die nach einem Fehler im unbestätigten Zeitraum verstreichen, bevor die Transaktion die im Feld INDOUBT zurückgegebene Aktion durchführt. Der zurückgegebene Wert ist nur gültig, wenn die Arbeitseinheit unbestätigt ist und für INDOUBT_WAIT die Antwort XMxD_YES zurückgegeben wird.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die Verzögerungszeit als Binärwert empfangen wird.

Siehe dazu auch INDOUBT und INDOUBT_WAIT.

INITIAL_PROGRAM(name8)

Gibt den Namen des ursprünglichen Programms zurück, an das die Steuerung für die Transaktion übertragen wird.

name8 Der Name einer aus 8 Byte bestehenden Position, an der der Name des ursprünglichen Programms empfangen wird.

INQ_TRANSACTION_ID(name4 | string | 'string')

Gibt die Transaktionskennung für die Transaktionsdefinitionsanfrage an.

name4 Der Name einer aus 4 Byte bestehenden Position mit dem Namen der Transaktionskennung.

string Eine Zeichenfolge ohne eingebettete Leerzeichen, die die Transaktionskennung angibt.

'string'

Eine in Anführungszeichen gesetzte Zeichenfolge für die Transaktionskennung. Die Länge der Zeichenfolge ist auf 4 Byte festgelegt und wird erreicht, indem Leerzeichen innerhalb der Anführungszeichen hinzugefügt werden.

ISOLATE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob eine Transaktionsisolation für den tasklaufzeitbezogenen Benutzerschlüsselspeicher der Transaktion erforderlich ist.

XMxD_NO

Für den Benutzerschlüsselspeicher für die Tasklaufzeit ist keine Transaktionsisolation erforderlich.

XMxD_YES

Für den Benutzerschlüsselspeicher für die Tasklaufzeit ist eine Transaktionsisolation erforderlich.

LOCAL_QUEUEING(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob eine Startanforderung für die Transaktion in eine lokale Warteschlange gestellt werden kann, wenn die Transaktion auf einem anderen System gestartet werden soll, das ferne System jedoch nicht verfügbar ist.

XMxD_NO

Die Anforderung kann nicht in eine lokale Warteschlange gestellt werden.

XMxD_YES

Die Anforderung kann in eine lokale Warteschlange gestellt werden.

OTSTIMEOUT(name4)

Gibt den Standardzeitraum in Sekunden zurück, in dem die Ausführung einer OTS-Transaktion (OTS = Object Transaction Service), die in einer EJB-Umgebung (EJB = Enterprise JavaBeans) erstellt und unter der CICS-Transaktion ausgeführt wird, zugelassen wird, ohne dass der Initiator der OTS-Transaktion einen Synchronisationspunkt erstellt (oder ein Rollback für die OTS-Transaktion durchführt).

name4 Der Name einer aus 4 Byte bestehenden Position, an der die Einstellung für die Zeitlimitüberschreitung als Binärwert empfangen wird.

(Rn) Ein Register, in dem die Einstellung für die Zeitlimitüberschreitung als Binärwert empfangen wird.

Ein Nullwert bedeutet, dass in der Transaktionsressourcendefinition OTSTIME-OUT(NO) angegeben ist.

PARTITIONSET(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der die in der Transaktionsdefinition angegebene Partitionsgruppe angibt.

XMxD_KEEP

Für die Partitionsgruppe ist der reservierte Name KEEP angegeben. Dies bedeutet, dass Tasks, die unter der Transaktionsdefinition ausgeführt werden, die Anwendungspartitionsgruppe für das der Transaktion zugeordnete Terminal verwenden.

XMxD_NAMED

Der Name der Partitionsgruppe ist in der Transaktionsdefinition ausdrücklich angegeben. Der Name wird im Parameter PARTITIONSET_NAME zurückgegeben.

XMxD_NONE

Für die Transaktionsdefinition ist keine Partitionsgruppe angegeben.

XMxD_OWn

Für die Partitionsgruppe ist der reservierte Name OWN angegeben. Dies bedeutet, dass Tasks, die unter der Transaktionsdefinition ausgeführt werden, eine eigene Partitionsgruppenverwaltung durchführen.

PARTITIONSET_NAME(name8)

Gibt den Namen der in der Transaktionsdefinition definierten Partitionsgruppe zurück.

name8 Der Name einer aus 8 Byte bestehenden Position für den zu empfangenden Partitionsgruppennamen.

PROFILE_NAME(name8)

Gibt den Namen der Profildefinition zurück, die der Transaktionsdefinition zugeordnet ist.

name8 Der Name einer aus 8 Byte bestehenden Position, an der der Name der Profildefinition empfangen wird, die der Transaktionsdefinition zugeordnet ist.

REMOTE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion als fern definiert ist.

XMxD_NO

Bei der Transaktion handelt es sich nicht um eine ferne Transaktion.

XMxD_YES

Bei der Transaktion handelt es sich um eine ferne Transaktion.

REMOTE_NAME(name8)

Gibt den Namen zurück, unter dem die Transaktion in einem fernen System bekannt ist.

name8 Der Name einer aus 8 Byte bestehenden Position, an der der Name des fernen Systems für die Transaktion empfangen wird.

REMOTE_SYSTEM(name4)

Gibt den Namen des fernen Systems, wie in der Transaktionsdefinition angegeben, zurück.

Wird vom Parameter DYNAMIC der Wert XMXD_YES zurückgegeben, gibt REMOTE_SYSTEM den Standardnamen zurück, der vom Programm für dynamisches Routing geändert werden kann.

Wird vom Parameter DYNAMIC der Wert XMXD_NO zurückgegeben, handelt es sich bei dem Namen um das ferne System, an das die Transaktion weitergeleitet wird.

name4 Der Name einer aus 4 Byte bestehenden Position, an der der definierte Name des fernen Systems empfangen wird.

RESSEC(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob eine Ressourcensicherheitsprüfung für die Transaktion erforderlich ist.

XMXD_NO

Eine Ressourcensicherheitsprüfung ist nicht erforderlich.

XMXD_YES

Eine Ressourcensicherheitsprüfung ist erforderlich.

RESTART(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob für die Transaktion ein Transaktionsneustart infrage kommt.

XMXD_NO

Die Transaktion kann nicht erneut gestartet werden.

XMXD_YES

Die Transaktion kann erneut gestartet werden.

ROUTABLE_STATUS(ROUTABLE|NOT_ROUTABLE)

Gibt einen Wert zurück, der angibt, ob die Transaktion mithilfe der erweiterten Routingmethode weitergeleitet wird, wenn sie Gegenstand eines geeigneten EXEC CICS START-Befehls ist.

NOT_ROUTABLE

Ist die Transaktion Gegenstand eines Startbefehls, wird sie unter Verwendung der 'traditionellen' Methode weitergeleitet.

ROUTABLE

Ist die Transaktion Gegenstand eines geeigneten Startbefehls, wird sie unter Verwendung der erweiterten Methode weitergeleitet.

Details zur erweiterten und zur 'traditionellen' Methode für die Weiterleitung von Transaktionen, die über EXEC CICS START-Befehle aufgerufen wurden, finden Sie im Abschnitt CICS transaction routing.

RUNAWAY_LIMIT(name4 | (Rn))

Gibt das in der Transaktionsdefinition angegebene Zeitlimit für nicht mehr steuerbare Tasks zurück. Ist für SYSTEM_RUNAWAY der Wert XMXD_YES angegeben, wird der über den Systeminitialisierungsparameter **ICVR** definierte Wert zurückgegeben.

name4 Der Name einer aus 4 Byte bestehenden Position, an der das Zeitlimit für nicht mehr steuerbare Tasks als Binärwert empfangen wird.

(Rn) Ein Register, in dem das Zeitlimit für nicht mehr steuerbare Tasks als Binärwert empfangen wird.

SHUTDOWN(*name1*)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion während des Beendens von CICS ausgeführt werden kann.

XMxD_DISABLED

Die Transaktion kann nicht beim Beenden von CICS ausgeführt werden.

XMxD_ENABLED

Die Transaktion kann beim Beenden von CICS ausgeführt werden.

SPURGE(*name1*)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion als vom System bereinigbar definiert ist.

XMxD_NO

Die Transaktion kann nicht im Rahmen einer vom System durchgeführten Bereinigung gelöscht werden.

XMxD_YES

Die Transaktion kann im Rahmen einer vom System durchgeführten Bereinigung gelöscht werden.

STATUS(*name1*)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der den Status der Transaktionsdefinition angibt.

XMxD_DISABLED

Die Transaktionsdefinition ist inaktiviert.

XMxD_ENABLED

Die Transaktionsdefinition ist aktiviert.

STORAGE_CLEAR(*name1*)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob der Inhalt des Tasklaufzeitspeichers von Tasks, die der Transaktionsdefinition zugeordnet sind, gelöscht wird, bevor der Speicher über einen Befehl FREEMAIN freigegeben wird.

XMxD_NO

Der Inhalt des Tasklaufzeitspeichers muss nicht gelöscht werden, bevor die Freigabe erfolgt.

XMxD_YES

Der Inhalt des Tasklaufzeitspeichers muss gelöscht werden, bevor die Freigabe erfolgt.

STORAGE_FREEZE(*name1* | (Rn))

Gibt in einem aus 1 Byte bestehenden Feld (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Speicherblockierung für die Transaktion über die Option STGFRZ der von CICS bereitgestellten Transaktion für Servicemitarbeiter (CICS Supplied Field Engineering Transaction, CSFE-Transaktion) definiert wird.

XMxD_NO

Speicher wird bei der Ausführung der Transaktion normal freigegeben.

XMxD_YES

Speicher, der normalerweise während der Ausführung einer Transaktion freigegeben wird, wird blockiert.

SYSTEM_ATTACH(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die dieser Transaktions-ID zugeordneten Tasks immer als Systemtasks angehängt werden.

XMxD_NO

Eine Benutzertask wird für die Transaktion angehängt.

XMxD_YES

Eine Systemtask wird für die Transaktion angehängt.

SYSTEM_RUNAWAY(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktionsdefinition das Systemstandardzeitlimit für nicht mehr steuerbare Tasks enthält, das im Systeminitialisierungsparameter **ICVR** festgelegt ist.

XMxD_NO

Die Transaktion unterliegt nicht dem Systemzeitlimit für nicht mehr steuerbare Tasks.

XMxD_YES

Die Transaktionsdefinition gibt das Systemstandardzeitlimit für nicht mehr steuerbare Tasks an.

TASKDATAKEY(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der den Speicherschlüssel für den Tasklaufzeitspeicher für Tasks angibt, die der Transaktionsdefinition zugeordnet sind.

XMxD_CICS

Für Tasklaufzeitspeicher ist die Ausführung mit CICS-Schlüssel angegeben.

XMxD_USER

Für Tasklaufzeitspeicher ist die Ausführung mit Benutzerschlüssel angegeben.

TASKDATALOC(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der die Datenposition für den Tasklaufzeitspeicher für Tasks angibt, die der Transaktionsdefinition zugeordnet sind.

XMxD_ANY

Tasklaufzeitspeicher kann sich oberhalb der 16-MB-Grenze im virtuellen Speicher befinden.

XMxD_BELOW

Tasklaufzeitspeicher muss sich unterhalb der 16-MB-Grenze im virtuellen Speicher befinden.

TCLASS(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion zu einer Transaktionsklasse gehört.

XMxD_NO

Die Transaktion gehört keiner Transaktionsklasse an.

XMxD_YES

Die Transaktion gehört der im Parameter **TCLASS_NAME** angegebenen Transaktionsklasse an.

TCLASS_NAME(name8)

Gibt den Namen der Transaktionsklasse zurück, zu der die Transaktion gehört.

name8 Der Name einer aus 8 Byte bestehenden Position für den zu empfangenden Namen der Transaktionsklasse, zu der die Transaktion gehört.

TPURGE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion im Fall eines Terminalfehlers beim z/OS Communications Server als bereinigbar definiert ist.

XMxD_NO

Die Transaktion kann nicht gelöscht werden, wenn ein Terminalfehler auftritt.

XMxD_YES

Die Transaktion kann gelöscht werden, wenn ein Terminalfehler auftritt.

TRACE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der die für die Transaktion definierte Tracestufe angibt:

XMxD_SPECIAL

Spezieller CICS-Trace. Diese Tracestufe ist aktiviert, wenn mit dem CICS-Befehl SET TRANSACTION ein spezielles Tracing angefordert wurde.

XMxD_STANDARD

CICS-Standardtrace. Dieser Wert entspricht der Einstellung TRACE(YES) in der Transaktionsressourcendefinition.

XMxD_SUPPRESSED

Tracing wird für die Transaktion unterdrückt. Dieser Wert entspricht der Einstellung TRACE(NO) in der Transaktionsressourcendefinition.

TRAN_PRIORITY(name4 | (Rn))

Gibt die in der Transaktionsdefinition angegebene Transaktionspriorität zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die Transaktionspriorität als Binärwert empfangen wird.

(Rn) Ein Register, in dem die Transaktionspriorität als Binärwert empfangen wird.

TRAN_ROUTING_PROFILE(name8)

Gibt den Namen des Profils zurück, das von CICS für die Weiterleitung der Transaktion an ein fernes System verwendet wird.

name8 Der Name einer aus 8 Byte bestehenden Position, an der das Profil für Transaktionsrouting empfangen wird.

TRANSACTION_ID(name4)

Gibt die primäre Transaktionskennung für die Transaktionsdefinitionsanfrage zurück.

name4 Der Name einer aus 4 Byte bestehenden Position mit dem Namen der Transaktionskennung.

TWASIZE(name4 | (Rn))

Gibt die in der Transaktionsdefinition angegebene Größe des Transaktionsarbeitsbereichs (Transaction Work Area, TWA) zurück.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die Größe des Transaktionsarbeitsbereichs als Binärwert empfangen wird.

(Rn) Ein Register, in dem die Größe des Transaktionsarbeitsbereichs als Binärwert empfangen wird.

RESPONSE- und REASON-Werte für INQUIRE_TRANDEF

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	UNKNOWN_TRANSACTION_ID
INVALID	----
DISASTER	LOGIC_ERROR
PURGED	----

Aufruf INQUIRE_TRANSACTION

Die Funktion INQUIRE_TRANSACTION wird für den Makroaufruf DFHXMIQX bereitgestellt. Diese Funktion ermöglicht es Ihnen, Informationen zu einer angehängten Transaktion (Task) abzurufen. Insgesamt entspricht dieser Funktionsaufruf, von einigen Unterschieden abgesehen, dem Befehl EXEC CICS INQUIRE TASK.

INQUIRE_TRANSACTION

```
DFHXMIQX [CALL,]
          [CLEAR,]
          [IN,]
          FUNCTION(INQUIRE_TRANSACTION),
          [TRANSACTION_TOKEN(name8),]
          [OUT,]
          [ATTACH_TIME(name8),]
          [CICS_UOW_ID(name8),]
          [DTIMEOUT(name4 | (Rn)),]
          [DYNAMIC(name1),]
          [FACILITY_NAME(name4),]
          [FACILITY_TYPE(name1),]
          [INITIAL_PROGRAM(name8),]
          [NETNAME(name8),]
          [ORIGINAL_TRANSACTION_ID(name4),]
          [OUT_TRANSACTION_TOKEN(name8),]
          [RE_ATTACHED_TRANSACTION(name1),]
          [REMOTE(name1),]
          [REMOTE_NAME(name8),]
          [REMOTE_SYSTEM(name4),]
          [RESOURCE_NAME(name16),]
          [RESOURCE_TYPE(name8),]
          [RESTART(name1),]
          [RESTART_COUNT(name2 | (Rn)),]
          [SPURGE(name1),]
          [START_CODE(name1),]
          [STATUS(name1),]
          [SUSPEND_TIME(name4 | (Rn)),]
          [SYSTEM_TRANSACTION(name1),]
          [TASK_PRIORITY(name1),]
          [TCLASS(name1), [TCLASS_NAME(name8),]]
          [TERMINATE_PROTECTED(name1),] [TPURGE(name1),]
          [TRANNUM(name4 | string | 'string'),]
          [TRAN_PRIORITY(name1),]
          [TRAN_ROUTING_PROFILE(name8),]
          [TRANSACTION_ID(name4),]
          [USERID(name8),]
          RESPONSE (name1 | *),
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

Die Beschreibungen der folgenden Parameter stimmen mit den Beschreibungen zu den entsprechenden Parametern für den Funktionsaufruf INQUIRE_TRANDEF überein.

DTIMEOUT
DYNAMIC
INITIAL_PROGRAM
REMOTE
REMOTE_NAME
REMOTE_SYSTEM
RESTART
SPURGE
STATUS
TCLASS
TRAN_ROUTING_PROFILE
TRANSACTION_ID

Die folgenden Parameterbeschreibungen enthalten kurze Erläuterungen zu den Werten, die von einem Aufruf INQUIRE_TRANSACTION zurückgegeben werden können. Eine ausführlichere Erläuterung zu diesen Parametern finden Sie in den Parameterbeschreibungen für die Transaktionsressourcendefinition im Abschnitt TRANSACTION attributes.

ATTACH_TIME(name8)

Gibt die Zeit in Millisekunden zurück, die seit dem Anhängen der Task verstrichen ist.

name8 Der Name einer aus 8 Byte bestehenden Position für die zu empfangende Zeit im gepackten ABSTIME-Dezimalformat.

CICS_UOW_ID(name8)

Gibt die CICS-Arbeitseinheitenkennung für die Task zurück.

name8 Der Name einer aus 8 Byte bestehenden Position für die zu empfangende Arbeitseinheiten-ID.

FACILITY_NAME(name4)

Gibt den Namen der Hauptfunktion zurück, die der Task zugeordnet ist.

name4 Der Name einer aus 4 Byte bestehenden Position für den zu empfangenden Namen der Hauptfunktion.

FACILITY_TYPE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der den Typ der Hauptfunktion angibt, die der Task zugeordnet ist.

XMIQ_NONE

Es gibt keine Hauptfunktion.

XMIQ_START

Die Hauptfunktion ist ein Intervallsteuerungselement.

XMIQ_TD

Die Hauptfunktion ist eine Warteschlange mit transienten Daten.

XMIQ_TERMINAL

Die Hauptfunktion ist ein Terminal.

NETNAME(name8)

Gibt den Netznamen der Hauptfunktion zurück, die der Task zugeordnet ist.

name8 Der Name einer aus 8 Byte bestehenden Position für den zu empfangenden Netznamen.

ORIGINAL_TRANSACTION_ID(name4)

Gibt die Transaktions-ID zurück, die zum Anhängen der Transaktion verwendet wurde. Wurde an einem Terminal beispielsweise ein Aliasname verwendet, wird in diesem Feld der betreffende Aliasname zurückgegeben.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die ursprüngliche Transaktionskennung empfangen wird.

OUT_TRANSACTION_TOKEN(name8)

Gibt das Token zurück, das die Task darstellt.

name8 Der Name einer aus 8 Byte bestehenden Position für das zu empfangende Transaktionstoken für die Task.

RE_ATTACHED_TRANSACTION(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion erneut angehängt wurde.

XMIQ_NO

Die Transaktion wurde nicht erneut angehängt und das globale Benutzerexitprogramm wird in derselben Umgebung aufgerufen wie beim ursprünglichen Anhängen der Task.

XMIQ_YES

Die Transaktion wurde erneut angehängt und die Umgebung beim Aufruf des globalen Benutzerexitprogramms unterscheidet sich von der Umgebung beim ursprünglichen Anhängen der Task.

RESOURCE_NAME(name16)

Gibt den Namen einer Ressource zurück, auf die die (ausgesetzte) Transaktion wartet.

name16

Der Name einer aus 16 Byte bestehenden Position, an der der Name der Ressource empfangen wird, auf die die Transaktion wartet.

RESOURCE_TYPE(name8)

Gibt den Typ der Ressource zurück, auf die die (ausgesetzte) Transaktion wartet.

name8 Der Name einer aus 8 Byte bestehenden Position, an der der Typ der Ressource empfangen wird, auf die die Transaktion wartet.

RESTART_COUNT(name2 | (Rn))

Gibt zurück, wie oft die Instanz der Transaktion erneut gestartet wurde.

name2 Der Name einer aus 2 Byte bestehenden Position, an der die Anzahl der erneuten Transaktionsstartvorgänge als Halbwort-Binärwert empfangen wird.

(Rn) Ein Register, in dem die Anzahl der erneuten Transaktionsstartvorgänge als Halbwort-Binärwert empfangen wird.

START_CODE(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, wie die Task gestartet wurde:

C Internes Anhängen durch CICS.

XMIQ_DF

Der Startcode ist noch nicht bekannt und wird später angegeben.

XMIQ_QD

Anhängen aufgrund einer erreichten Auslöserebene mit transienten Daten.

XMIQ_S

START-Befehl ohne Daten.

XMIQ_SD

START-Befehl mit Daten.

XMIQ_SZ

FEPI-Anhängen (FEPI = Front End Programming Interface).

XMIQ_T

Anhängen aufgrund von Terminaleingabe.

XMIQ_TT

Anhängen über permanente Terminaltransaktion

SUSPEND_TIME(name4 | (Rn))

Gibt die Zeit zurück, die seit dem Eintritt der Task in den Aussetzstatus verstrichen ist.

name4 Der Name einer aus 4 Byte bestehenden Position, an der die gerundete Anzahl der Sekunden, die seit dem Eintritt der Task in den Aussetzstatus verstrichen sind, als Binärwert empfangen wird.

(Rn) Ein Register, in dem die gerundete Anzahl der Sekunden, die seit dem Eintritt der Task in den Aussetzstatus verstrichen sind, als Binärwert empfangen wird.

SYSTEM_TRANSACTION(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob es sich bei der Task um eine CICS-Systemtask handelt.

XMIQ_NO

Die Task ist keine CICS-Systemtask.

XMIQ_YES

Die Task ist eine CICS-Systemtask.

TASK_PRIORITY(name1)

Gibt die kombinierte Taskpriorität zurück, bei der es sich um die Summe der Prioritäten handelt, die für das Terminal, die Transaktion und den Bediener definiert sind.

name1 Der Name einer aus 1 Byte bestehenden Position, an der die Taskpriorität als Binärzahl empfangen wird.

TERMINATE_PROTECTED(name1)

Gibt an einer aus 1 Byte bestehenden Position (*name1*) einen Gleichsetzungswert zurück, der angibt, ob die Transaktion abgebrochen werden kann.

XMIQ_NO

Die Transaktion kann abgebrochen werden.

XMIQ_YES

Die Transaktion kann nicht abgebrochen werden.

TRANNUM(name4)

Gibt die Tasknummer der Transaktion zurück.

name4 Der Name einer aus 4 Byte bestehenden Position für die zu empfangende Tasknummer.

TRANSACTION_TOKEN(name8)

Gibt das Transaktionstoken für die Task an, für die die Informationen abgerufen werden. Dieser Parameter ist optional. Ist der Parameter nicht angegeben, wird von der aktuellen Task ausgegangen.

Wenn Sie diesen Aufruf in einem globalen XXMATT-Benutzerexitprogramm angeben, kann es sich bei der aktuellen Task um eine CICS-Systemtask handeln. Zum Anfragen von Informationen zu der Benutzertask, für die XXMATT aufgerufen wurde, müssen Sie das Transaktionstoken angeben, das in der speziellen Parameterliste für den Exit XXMATT übergeben wurde.

name8 Der Name einer aus 8 Byte bestehenden Position mit dem Transaktionstoken.

USERID(name8)

Gibt die Benutzer-ID zurück, die der Task zugeordnet ist.

name8 Der Name einer aus 8 Byte bestehenden Position für die zu empfangende Benutzer-ID.

RESPONSE- und REASON-Werte für INQUIRE_TRANSACTION

<i>RESPONSE</i>	<i>REASON</i>
OK	----
DISASTER	ABEND
	LOOP
INVALID	----
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
	BUFFER_TOO_SMALL
	INVALID_TRANSACTION_TOKEN
KERNERROR	----

Aufruf SET_TRANSACTION

Die Funktion SET_TRANSACTION wird für den Makroaufruf DFHXMIQX bereitgestellt. Diese Funktion ermöglicht es Ihnen, Taskpriorität und Transaktionsklasse der aktuellen Task zu ändern.

Dabei ist zu beachten, dass Sie den Transaktionsklassennamen TCLASS_NAME nur mit diesem Aufruf ändern können, wenn der Aufruf von einem globalen XXMATT-Benutzerexitprogramm ausgeht.

SET_TRANSACTION

```
DFHXMIQX  [CALL,]
          [CLEAR,]
          [IN,
          FUNCTION(SET_TRANSACTION),
          [TASK_PRIORITY(name4),]
          [TCLASS_NAME(name8),]
          [TRANSACTION_TOKEN(name8),]]
          [OUT,
          RESPONSE (name1 | *),
          REASON (name1 | *)]
```

Dieser Befehl ist threadsicher.

TASK_PRIORITY(name4)

Gibt die neue Taskpriorität für die durch TRANSACTION_TOKEN gekennzeichnete Task an.

name4 Der Name einer aus 4 Byte bestehenden Position, die die neue Taskprioritätszahl als Binärwert enthält.

TCLASS_NAME(name8)

Gibt den Namen der neuen Transaktionsklasse an, der die Task zugeordnet werden soll. Geben Sie den speziellen Standardsystemnamen DFHTCL00 an, um festzulegen, dass die Task in keiner Transaktionsklasse enthalten sein soll.

name8 Der Name einer aus 8 Byte bestehenden Position mit dem Namen der neuen Transaktionsklasse. Verwenden Sie den Wert DFHTCL00 für dieses Feld, wenn die Task keiner Transaktionsklasse zugeordnet werden soll.

TRANSACTION_TOKEN(name8)

Gibt das Transaktionstoken für die Task an, die geändert wird. Wenn Sie diesen Parameter nicht angeben, bezieht sich der Aufruf standardmäßig auf die aktuelle Task.

name8 Der Name einer aus 8 Byte bestehenden Position mit dem Transaktionstoken.

RESPONSE- und REASON-Werte für SET_TRANSACTION

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
	UNKNOWN_TCLASS
	INVALID_TRANSACTION_TOKEN
DISASTER	ABEND
	LOOP
INVALID	----
KERNERROR	----

Kapitel 17. XPI-Funktion für Benutzerjournalführung

Die XPI stellt eine Benutzerjournalfunktion zur Verfügung. Dabei handelt es sich um den DFHJCJCX-Aufruf `WRITE_JOURNAL_DATA`.

Einschränkung: DFHJCJCX-Aufrufe können in keinem Exitprogramm verwendet werden, das über einen globalen Benutzerexitpunkt in den folgenden Domänen oder dem folgenden Programm aufgerufen wird:

- Statistikdomäne
- Monitordomäne
- Speicherauszugsdomäne
- Dispatcherdomäne
- Programm für transiente Daten (TDP)

Aufruf `WRITE_JOURNAL_DATA`

Mit dem Aufruf `WRITE_JOURNAL_DATA` wird ein einzelner Journalsatz in das in der Journalmodelldefinition angegebene Journal geschrieben, das dem Journalnamen entspricht. Es kann sich um ein Journal in einem Protokolldatenstrom der MVS-Systemprotokollfunktion oder einen SMF-Datensatz handeln. Ist `DUMMY` in der Definition definiert, wird kein Datensatz geschrieben.

`WRITE_JOURNAL_DATA`

```
DFHJCJCX [CALL,]  
        [CLEAR,]  
        [IN,  
        FUNCTION(WRITE_JOURNAL_DATA),  
        FROM(block-descriptor),  
        JOURNALNAME(name8 | string | 'string') |  
        JOURNAL_RECORD_ID(name2 | string | 'string'),  
        WAIT(YES|NO),  
        [RECORD_PREFIX(block-descriptor),]]  
        [OUT,  
        RESPONSE(name1 | *),  
        REASON(name1 | *)]
```

Dieser Befehl ist threadsicher.

Wichtig

Bei der Verwendung der XPI in einem frühen Stadium der Initialisierung ist eine Einschränkung zu beachten. Starten Sie Exitprogramme, die die XPI-Funktionen `TRANSACTION_DUMP`, `WRITE_JOURNAL_DATA`, `MONITOR` und `INQUIRE_MONITOR_DATA` verwenden, nicht vor der zweiten PLTPI-Phase. Weitere Informationen zu PLTPI (Program List Table Post Initialization) finden Sie im Abschnitt *Writing initialization and shutdown programs*.

FROM(block-descriptor)

Gibt Adresse und Länge des Journalsatzes an.

Der Blockdeskriptor umfasst 8 Datenbyte. Die ersten 4 Byte enthalten die Adresse der Daten, die geschrieben werden sollen. Die zweiten 4 Byte enthalten

die Länge der Daten. Der Blockdeskriptor wird über den Aufruf des Makros DFHJCJCX an die Position JCJC_FROM versetzt, die über den DFHJCJCY-Pseudoabschnitt (DSECT) zugeordnet wird.

JOURNALNAME(name8 | string | "string")

Gibt den Namen des CICS-Journals oder Protokolls an, in das die FROM-Daten geschrieben werden sollen.

JOURNAL_RECORD_ID(name2 | string | "string")

Gibt einen aus 2 Zeichen bestehenden Wert an, der in den Journalsatz geschrieben werden soll, um den zugehörigen Ursprung zu kennzeichnen.

name2 Der Name einer aus 2 Byte bestehenden Position.

string Eine auf eine Länge von 2 Byte begrenzte Zeichenfolge im generierten Code.

"string"

Eine auf eine Länge von 2 Byte begrenzte, in Anführungszeichen gesetzte Zeichenfolge im generierten Code.

RECORD_PREFIX(block-descriptor)

Gibt das optionale Benutzerpräfix an.

WAIT(YES|NO)

Gibt an, ob die Steuerung von CICS erst an das Exitprogramm zurückgegeben werden soll, wenn der Datensatz in das Journal bzw. Protokoll geschrieben wurde.

RESPONSE- und REASON-Werte für WRITE_JOURNAL_DATA

<i>RESPONSE</i>	<i>REASON</i>
OK	----
EXCEPTION	IO_ERROR
	JOURNAL_NOT_FOUND
	JOURNAL_NOT_OPEN
	LENGTH_ERROR
	STATUS_ERROR
DISASTER	----
INVALID	----
KERNERROR	----
PURGED	----

Anmerkung: Weitere Details finden Sie in den Erläuterungen zu RESPONSE und REASON im Abschnitt Making an XPI call.

Kapitel 18. Threadsichere XPI-Befehle

Die meisten, jedoch nicht alle XPI-Befehle sind threadsicher. Ein nicht threadsicherer Befehl bewirkt, dass der QR-Tasksteuerblock von CICS zur Sicherung der Serialisierung eingesetzt wird.

Die threadsicheren XPI-Befehle sind in den Befehlssyntaxdiagrammen mit dem Hinweis „Dieser Befehl ist threadsicher“ gekennzeichnet. Folgende Befehle sind threadsicher:

Threadsichere Befehle

- DFHAPIQX INQ_APPLICATION_DATA
- DFHBRIQX INQUIRE_CONTEXT
- DFHDDAPX BIND_LDAP
- DFHDDAPX END_BROWSE_RESULTS
- DFHDDAPX FLUSH_LDAP_CACHE
- DFHDDAPX FREE_SEARCH_RESULTS
- DFHDDAPX GET_ATTRIBUTE_VALUE
- DFHDDAPX GET_NEXT_ATTRIBUTE
- DFHDDAPX GET_NEXT_ENTRY
- DFHDDAPX SEARCH_LDAP
- DFHDDAPX START_BROWSE_RESULTS
- DFHDDAPX UNBIND_LDAP
- DFHDSATX CHANGE_PRIORITY
- DFHDSSRX ADD_SUSPEND
- DFHDSSRX DELETE_SUSPEND
- DFHDSSRX RESUME
- DFHDSSRX SUSPEND
- DFHDSSRX WAIT_MVS
- DFHDUDUX SYSTEM_DUMP
- DFHJCJCX WRITE_JOURNAL_DATA
- DFHKEDSX START_PURGE_PROTECTION
- DFHKEDSX STOP_PURGE_PROTECTION
- DFHLDLDX ACQUIRE_PROGRAM
- DFHLDLDX DEFINE_PROGRAM
- DFHLDLDX DELETE_PROGRAM
- DFHLDLDX IDENTIFY_PROGRAM
- DFHLDLDX RELEASE_PROGRAM
- DFHLGPAX INQUIRE_PARAMETERS
- DFHLGPAX SET_PARAMETERS
- DFHMNMNX INQUIRE_MONITORING_DATA
- DFHMNMNX MONITOR
- DFHNQEDX DEQUEUE
- DFHNQEDX ENQUEUE
- DFHPGAQX INQUIRE_AUTOINSTALL
- DFHPGAQX SET_AUTOINSTALL
- DFHPGISX END_BROWSE_PROGRAM
- DFHPGISX GET_NEXT_PROGRAM
- DFHPGISX INQUIRE_CURRENT_PROGRAM
- DFHPGISX INQUIRE_PROGRAM
- DFHPGISX SET_PROGRAM
- DFHPGISX START_BROWSE_PROGRAM
- DFHSAIQX INQUIRE_SYSTEM

- DFHSAIQX SET_SYSTEM
- DFHSMMCX GETMAIN
- DFHSMMCX FREEMAIN
- DFHSMMCX INQUIRE_ELEMENT_LENGTH
- DFHSMMCX INQUIRE_TASK_STORAGE
- DFHMSRX INQUIRE_ACCESS
- DFHMSRX INQUIRE_SHORT_ON_STORAGE
- DFHMSRX SWITCH_SUBSPACE
- DFHTRPTX TRACE_PUT
- DFHXMCLX INQUIRE_TCLASS
- DFHXMIQX INQUIRE_TRANSACTION
- DFHXMIQX SET_TRANSACTION
- DFHXMSRX INQUIRE_DTRTRAN
- DFHXMSRX INQUIRE_MXT
- DFHMXDX INQUIRE_TRANDEF

Nicht threadsichere Befehle

- DFHDUDUX TRANSACTION_DUMP

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. IBM stellt dieses Material möglicherweise auch in anderen Sprachen zur Verfügung. Für den Zugriff auf das Material in einer anderen Sprache kann eine Kopie des Produkts oder der Produktversion in der jeweiligen Sprache erforderlich sein.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

*IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes 2, avenue Gambetta
92066 Paris La Defense
France*

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America*

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesen Informationen beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden und jede Ähnlichkeit mit Namen und Adressen tatsächlicher Personen oder Unternehmen ist rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellsprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielpprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielpprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielpprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielpprogramme entstehen.

Informationen zu Programmierschnittstellen

Die von CICS zur Verfügung gestellte Dokumentation kann teilweise als Programmierschnittstelle betrachtet werden und zum Teil nicht.

Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zur Nutzung der Services von CICS Transaction Server for z/OS, Version 5 Release 5 zu schreiben, sind in folgenden Abschnitten der Online-Produktdokumentation enthalten:

- Developing applications

- Developing system programs
- Securing overview
- Developing for external interfaces
- Reference: application development
- Reference: system programming
- Reference: connectivity

Informationen, die NICHT zur Verwendung als Programmierschnittstelle von CICS Transaction Server for z/OS, Version 5 Release 5 bestimmt sind, die aber als Programmierschnittstelle missverstanden werden können, sind in folgenden Abschnitten der Online-Produktdokumentation enthalten:

- Troubleshooting and support
- Reference: diagnostics

Wenn Sie auf die CICS-Dokumentation in Handbüchern im PDF-Format zugreifen, sind Programmierschnittstellen, die es dem Kunden ermöglichen, Programme zur Nutzung der Services von CICS Transaction Server for z/OS, Version 5 Release 5 zu schreiben, in den folgenden Handbüchern enthalten:

- Application Programming Guide und Application Programming Reference
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- Supplied Transactions
- CICSplex SM Managing Workloads
- CICSplex SM Managing Resource Usage
- CICSplex SM Application Programming Guide and Application Programming Reference
- Java™ Applications in CICS

Wenn Sie auf die CICS-Dokumentation in Handbüchern im PDF-Format zugreifen, sind Informationen, die NICHT zur Verwendung als Programmierschnittstelle von CICS Transaction Server for z/OS, Version 5 Release 5 bestimmt sind, die aber als Programmierschnittstelle missverstanden werden können, in den folgenden Handbüchern enthalten:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSplex SM Problem Determination Guide

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite Copyright and trademark information unter www.ibm.com/legal/copytrade.shtml.

Adobe, das Adobe-Logo, PostScript und das PostScript-Logo sind Marken oder eingetragene Marken von Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder ihrer Tochtergesellschaften in den USA oder anderen Ländern.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Nutzungsbedingungen für die Produktdokumentation

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit

Diese Bedingungen sind eine Ergänzung der Nutzungsbedingungen auf der IBM Website.

Persönliche Nutzung

Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung

Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens weder vervielfältigen, weitergeben oder anzeigen noch abgeleitete Werke davon erstellen.

Rechte

Abgesehen von den hier gewährten Berechtigungen werden keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum gewährt.

IBM behält sich das Recht vor, die hierin gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Vorschriften, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM übernimmt keine Gewährleistung für den Inhalt dieser Veröffentlichungen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendbarkeit für einen bestimmten Zweck oder die Freiheit von Rechten Dritter zur Verfügung gestellt.

IBM Online-Datenschutzerklärung

IBM Softwareprodukte, einschließlich Software as a Service-Lösungen („Softwareangebote“), können Cookies oder andere Technologien verwenden, um Informationen zur Produktnutzung zu erfassen, die Endbenutzererfahrung zu verbessern und Interaktionen mit dem Endbenutzer anzupassen oder zu anderen Zwecken. In vielen Fällen werden von den Softwareangeboten keine personenbezogenen Daten erfasst. Einige der IBM Softwareangebote können Sie jedoch bei der Erfassung personenbezogener Daten unterstützen. Wenn dieses Softwareangebot Cookies zur Erfassung personenbezogener Daten verwendet, sind nachfolgend nähere Informationen über die Verwendung von Cookies durch dieses Angebot zu finden:

Für die Webbenutzerschnittstelle von CICSplex System Manager (Hauptschnittstelle):

Abhängig von den bereitgestellten Konfigurationen kann dieses Softwareangebot Sitzungscookies und persistente Cookies zum Erfassen der Benutzernamen und anderer personenbezogener Daten einzelner Benutzer für das Sitzungsmanagement, die Authentifizierung, einen besseren Bedienungskomfort, zur Nutzungsüberwachung und für andere funktionale Zwecke verwenden. Diese Cookies können nicht inaktiviert werden.

Für die Webbenutzerschnittstelle von CICSplex System Manager (Datenschnittstelle):

Abhängig von den bereitgestellten Konfigurationen kann dieses Softwareangebot Sitzungscookies und persistente Cookies zum Erfassen der Benutzernamen und anderer personenbezogener Daten einzelner Benutzer für das Sitzungsmanagement, die Authentifizierung, einen besseren Bedienungskomfort, zur Nutzungsüberwachung und für andere funktionale Zwecke verwenden. Diese Cookies können nicht inaktiviert werden.

Für die Webbenutzerschnittstelle von CICSplex System Manager ("hello world"-Seite):

Abhängig von den bereitgestellten Konfigurationen kann dieses Softwareangebot Sitzungscookies verwenden, die keine personenbezogenen Daten erfassen. Diese Cookies können nicht inaktiviert werden.

Für CICS Explorer:

Abhängig von den bereitgestellten Konfigurationen kann dieses Softwareangebot persistente Vorgaben und Sitzungsvorgaben zum Erfassen der Benutzernamen und Kennwörter von Benutzern für das Sitzungsmanagement, die Authentifizierung und zur Single Sign-on-Konfiguration (einmalige Anmeldung) verwenden. Diese Vorgaben können nicht inaktiviert werden, auch wenn die Speicherung eines Benutzerkennworts auf ei-

nem Datenträger in verschlüsselter Form nur aktiviert werden kann, indem der Benutzer bei der Anmeldung explizit ein Kontrollkästchen aktiviert.

Wenn es die für dieses Softwareangebot bereitgestellten Konfigurationen Ihnen als Kunde ermöglichen, personenbezogene Daten von Endbenutzern über Cookies und andere Technologien zu erfassen, müssen Sie sich zu allen gesetzlichen Bestimmungen in Bezug auf eine solche Datenerfassung, einschließlich aller Mitteilungspflichten und Zustimmungsanforderungen, rechtlich beraten lassen.

Weitere Informationen zur Nutzung verschiedener Technologien, einschließlich Cookies, für diese Zwecke finden Sie unter IBM Privacy Policy und in der IBM Online Privacy Statement im Abschnitt „Cookies, Web-Beacons und sonstige Technologien“ sowie auf der Seite IBM Software Products and Software-as-a-Service Privacy Statement.

Index

A

ACQUIRE PROGRAM, XPI-Funktion 43
ADD SUSPEND, XPI-Funktion 17
Anwendungskontextdaten 57

B

Benutzerjournalfunktionen der XPI 135
BIND_CHANNEL, XPI-Funktion 93
BIND_LDAP, XPI-Funktion 3

C

CHANGE PRIORITY, XPI-Funktion 19
COMMIT, XPI-Funktion 67
COMMIT_ONE_PHASE, XPI-Funktion 66

D

DEFINE PROGRAM, XPI-Funktion 45
DELETE PROGRAM, XPI-Funktion 49
DELETE SUSPEND, XPI-Funktion 20
DEQUEUE, XPI-Funktion 37
DFHAPIQX, Makro 95
DFHBABRX, Makro 1
DFHBABRX-Makro 1
DFHBRIQX, Makro 115
DFHDDAPX, Makro 3, 5, 6, 7, 8, 9, 10, 12, 13
DFHDSATX, Makro 15
DFHDSSRX, Makro 15
DFHDUDUX, Makro 31
DFHJCJCX, Makro 135
DFHKEDSX, Makro 41
DFHLDLDX, Makro 43
DFHLGPAX, Makro 55
DFHMNMNX, Makro 57
DFHNQEDX, Makro 37
DFHOTCOX, Makro 69
DFHOTTRX, Makro 65, 66, 67, 68
DFHPGAQX, Makro 73
DFHPGCHX, Makro 73
DFHPGISX, Makro 73
DFHSAIQX, Makro 97, 102
DFHSMMCX, Makro 105
DFHMSRX, Makro 110
DFHTRPTX, Makro 113
DFHXMCLX, Makro 119
DFHXXMIQX, Makro 129, 133
DFHXXMSRX, Makro 117
DFHXXMDX, Makro 120
Dispatcherfunktionen der XPI 15

E

END_BROWSE_PROGRAM, XPI-Funktion 90

END_BROWSE_RESULTS, XPI-Funktion 5
ENQUEUE 37
Enqueue-Domänenfunktionen der XPI 37

F

FLUSH_LDAP_CACHE, XPI-Funktion 6
FREE_SEARCH_RESULTS, XPI-Funktion 6
FREEMAIN, XPI-Funktion 108

G

Geschäftsanwendungsmanagerfunktion der XPI 1
GET_ATTRIBUTE_VALUE, XPI-Funktion 7
GET_NEXT_ATTRIBUTE, XPI-Funktion 8
GET_NEXT_ENTRY, XPI-Funktion 9
GET_NEXT_PROGRAM, XPI-Funktion 89
GETMAIN, XPI-Funktion 105

I

IDENTIFY_PROGRAM, XPI-Funktion 50
IMPORT_TRAN, XPI-Funktion 65
INQ_APPLICATION_DATA, XPI-Funktion 95
INQUIRE_ACCESS, XPI-Funktion 109
INQUIRE_ACTIVATION, XPI-Funktion 1
INQUIRE APP CONTEXT, XPI-Funktion 57
INQUIRE_AUTOINSTALL, XPI-Funktion 91
INQUIRE_CONTEXT, XPI-Funktion 115
INQUIRE_CURRENT_PROGRAM, XPI-Funktion 82
INQUIRE_DTRTRAN, XPI-Funktion 117
INQUIRE_ELEMENT_LENGTH, XPI-Funktion 109
INQUIRE MONITORING DATA, XPI-Funktion 59
INQUIRE_MXT, XPI-Funktion 117
INQUIRE_PARAMETERS, XPI-Funktion 55
INQUIRE_PROGRAM, XPI-Funktion 73
INQUIRE_SHORT_ON_STORAGE, XPI-Funktion 110
INQUIRE_SYSTEM, XPI-Funktion 97
INQUIRE_TASK_STORAGE, XPI-Funktion 111
INQUIRE_TCLASS, XPI-Funktion 119
INQUIRE_TRANDEF, XPI-Funktion 120

INQUIRE_TRANSACTION, XPI-Funktion 129

K

Kerneldomänenfunktionen der XPI 41

L

Ladeprogrammfunktionen der XPI 43
LIBRARYDSN, Option
Befehl INQUIRE_CURRENT PROGRAM 84

M

MONITOR, XPI-Funktion 60

O

Objekttransaktionsfunktionen der XPI 65

P

PREPARE, XPI-Funktion 67
Programmverwaltungsfunktionen der XPI 73
Protokollmanagerfunktionen der XPI 55

R

RELEASE PROGRAM, XPI-Funktion 52
RENTPGM, Systeminitialisierungsparameter 45, 85
RESUME, XPI-Funktion 21
ROLLBACK, XPI-Funktion 68

S

SEARCH_LDAP, XPI-Funktion 10
SET_AUTOINSTALL, XPI-Funktion 92
SET_COORDINATOR, XPI-Funktion 69
SET_PARAMETERS, XPI-Funktion 55
SET_PROGRAM, XPI-Funktion 85
SET_ROLLBACK_ONLY, XPI-Funktion 68
SET_SYSTEM, XPI-Funktion 102
SET_TRACKING_DATA, XPI-Funktion 63
SET_TRANSACTION, XPI-Funktion 133
Speicherauszugsteuerungsfunktionen der XPI 31
Speichersteuerungsfunktionen der XPI 105, 110
START_BROWSE_RESULTS, XPI-Funktion 12

START_PURGE_PROTECTION, XPI-Funktion 41
 Statusdatenzugriffsfunktionen der XPI 95
 STOP_PURGE_PROTECTION, XPI-Funktion 41
 SUSPEND, XPI-Funktion 22
 SWITCH_SUBSPACE, XPI-Funktion 112
 SYSTEM DUMP, XPI-Funktion 31
 Systeminitialisierungsparameter
 RENTPGM 45, 85

T

Threadsichere XPI-Befehle 137
 TRACE_PUT, XPI-Funktion 113
 Tracesteuerungsfunktionen der XPI 113
 TRANSACTION DUMP, XPI-Funktion 32
 Transaktionsverwaltungsfunktionen der XPI 115

U

Überwachung
 XPI-Funktionen 57
 UNBIND_LDAP, XPI-Funktion 13

V

Verzeichnisdomänenfunktionen der XPI 3

W

WRITE JOURNAL DATA, XPI-Funktion 135

X

XPI, Exitprogrammierschnittstelle
 Dispatcherfunktionen
 ADD SUSPEND 17
 DELETE SUSPEND 20
 PRIORITÄT ÄNDERN 19
 RESUME 21
 SUSPEND 22
 WAIT_MVS 26
 Enqueue-Domänenfunktionen
 DEQUEUE 37
 ENQUEUE 37
 Funktionen für Speicherauszugssteuerung
 SYSTEM DUMP 31
 TRANSACTION DUMP 32
 Funktionen für Speichersteuerung
 FREEMAIN 108
 GETMAIN 105
 INQUIRE_ACCESS 109
 INQUIRE_ELEMENT_LENGTH 109
 INQUIRE_SHORT_ON_STORAGE 110
 INQUIRE_TASK_STORAGE 111
 SWITCH_SUBSPACE 112

XPI, Exitprogrammierschnittstelle (*Forts.*)
 Funktionen für Statusdatenzugriff
 INQ_APPLICATION_DATA 95
 INQUIRE_SYSTEM 97
 SET_SYSTEM 102
 Journalführungsfunktion
 WRITE JOURNAL DATA 135
 Kerndomänenfunktionen
 START_PURGE_PROTECTION 41
 STOP_PURGE_PROTECTION 41
 Ladeprogrammfunktionen
 ACQUIRE PROGRAM 43
 DEFINE PROGRAM 45
 DELETE PROGRAM 49
 IDENTIFY_PROGRAM 50
 RELEASE PROGRAM 52
 Programmverwaltungsfunktionen
 BIND_CHANNEL 93
 END_BROWSE_PROGRAM 90
 GET_NEXT_PROGRAM 89
 INQUIRE_AUTOINSTALL 91
 INQUIRE_CURRENT_PROGRAM 82
 INQUIRE_PROGRAM 73
 SET_AUTOINSTALL 92
 SET_PROGRAM 85
 Protokollmanagerfunktionen
 INQUIRE_PARAMETERS 55
 SET_PARAMETERS 55
 threadsichere Befehle 137
 Tracesteuerungsfunktion
 TRACE_PUT 113
 Transaktionsverwaltungsfunktionen
 COMMIT 67
 COMMIT_ONE_PHASE 66
 IMPORT_TRAN 65
 INQUIRE_CONTEXT 115
 INQUIRE_DTRTRAN 117
 INQUIRE_MXT 117
 INQUIRE_TCLASS 119
 INQUIRE_TRANDEF 120
 INQUIRE_TRANSACTION 129
 PREPARE 67
 ROLLBACK 68
 SET_COORDINATOR 69
 SET_ROLLBACK_ONLY 68
 SET_TRANSACTION 133
 Überwachungsfunktionen
 INQUIRE_APP_CONTEXT 57
 INQUIRE_MONITORING_DATA 59
 MONITOR 60
 SET_TRACKING_DATA 63
 Verzeichnisdomänenfunktionen
 BIND_LDAP 3
 END_BROWSE_RESULTS 5
 FLUSH_LDAP_CACHE 6
 FREE_SEARCH_RESULTS 6
 GET_ATTRIBUTE_VALUE 7
 GET_NEXT_ATTRIBUTE 8
 GET_NEXT_ENTRY 9
 INQUIRE_ACTIVATION 1
 SEARCH_LDAP 10
 START_BROWSE_RESULTS 12
 UNBIND_LDAP 13

