HITACHI
Inspire the Next

OpenTP1 Version 7
分散トランザクション処理機能

# OpenTP1 解説

解説書

3000-3-D50-B0

# 前書き

## ■ 対象製品

・適用 OS：AIX V6.1，AIX V7.1，AIX V7.2，AIX V7.3

P-1M64-2141 uCosminexus TP1/Server Base 07-53
P-1M64-2341 uCosminexus TP1/FS/Direct Access 07-51
P-1M64-2441 uCosminexus TP1/FS/Table Access 07-51
P-1M64-2541 uCosminexus TP1/Client/W 07-52
P-1M64-2741 uCosminexus TP1/Online Tester 07-50
P-1M64-2841 uCosminexus TP1/Multi 07-50
P-1M64-2941 uCosminexus TP1/High Availability 07-50
P-1M64-8541 uCosminexus TP1/Extension 1 07-50
P-1M64-3141 uCosminexus TP1/Message Control 07-51
P-1M64-3241 uCosminexus TP1/NET/Library 07-51
P-1M64-C381 uCosminexus TP1/Message Queue 07-53
P-F1M64-31411 uCosminexus TP1/Message Control/Tester 07-50
P-F1M64-31412 uCosminexus TP1/Message Control - Extension 1 07-50
P-F1M64-32411 uCosminexus TP1/NET/User Agent 07-50
P-F1M64-32414 uCosminexus TP1/NET/OSI-TP 07-50
P-F1M64-32415 uCosminexus TP1/NET/XMAP3 07-50
P-F1M64-32418 uCosminexus TP1/NET/OSAS-NIF 07-50
P-F1M64-3241B uCosminexus TP1/NET/Secondary Logical Unit - TypeP2 07-50
P-F1M64-3241C uCosminexus TP1/NET/TCP/IP 07-50
P-F1M64-3241D uCosminexus TP1/NET/High Availability 07-50
P-F1M64-3241U uCosminexus TP1/NET/User Datagram Protocol 07-50
P-1M64-8141 uCosminexus TP1/Shared Table Access 07-50
P-1M64-8341 uCosminexus TP1/Resource Manager Monitor 07-50
P-1M64-1121 uCosminexus TP1/Server Base(64) 07-53
P-1M64-1321 uCosminexus TP1/FS/Direct Access(64) 07-51
P-1M64-1421 uCosminexus TP1/FS/Table Access(64) 07-51
P-1M64-1521 uCosminexus TP1/Client/W(64) 07-52
P-1M64-1921 uCosminexus TP1/High Availability(64) 07-50
P-1M64-1L21 uCosminexus TP1/Extension 1(64) 07-50
P-1M64-4121 uCosminexus TP1/Message Control(64) 07-51

P-1M64-4221 uCosminexus TP1/NET/Library(64) 07-51
P-F1M64-41212 uCosminexus TP1/Message Control - Extension 1(64) 07-50
P-F1M64-4221C uCosminexus TP1/NET/TCP/IP(64) 07-50
P-1M64-D121 uCosminexus TP1/Message Queue(64) 07-53
P-F1M64-4221D uCosminexus TP1/NET/High Availability(64) 07-50
P-F1M64-4221U uCosminexus TP1/NET/User Datagram Protocol(64) 07-50

・適用 OS：HP-UX 11i V3 (IPF)

P-1J64-2171 uCosminexus TP1/Server Base 07-51
P-1J64-2371 uCosminexus TP1/FS/Direct Access 07-51
P-1J64-2471 uCosminexus TP1/FS/Table Access 07-51
P-1J64-8571 uCosminexus TP1/Extension 1 07-50
P-1J64-8971 uCosminexus TP1/High Availability 07-50
P-1J64-C381 uCosminexus TP1/Message Queue 07-52
P-1J64-3171 uCosminexus TP1/Message Control 07-51
P-1J64-3271 uCosminexus TP1/NET/Library 07-51
P-F1J64-31712 uCosminexus TP1/Message Control - Extension 1 07-50
P-F1J64-32715 uCosminexus TP1/NET/XMAP3 07-51
P-F1J64-3271C uCosminexus TP1/NET/TCP/IP 07-51
P-F1J64-3271D uCosminexus TP1/NET/High Availability 07-50
P-1J64-1171 uCosminexus TP1/Server Base(64) 07-51
P-1J64-1371 uCosminexus TP1/FS/Direct Access(64) 07-51
P-1J64-1471 uCosminexus TP1/FS/Table Access(64) 07-51
P-1J64-8671 uCosminexus TP1/Extension 1(64) 07-50
P-1J64-8A71 uCosminexus TP1/High Availability(64) 07-50
P-1J64-C581 uCosminexus TP1/Message Queue (64) 07-52
P-1J64-4171 uCosminexus TP1/Message Control(64) 07-51
P-1J64-4271 uCosminexus TP1/NET/Library(64) 07-51
P-F1J64-41712 uCosminexus TP1/Message Control - Extension 1(64) 07-50
P-F1J64-4271C uCosminexus TP1/NET/TCP/IP(64) 07-51
P-F1J64-4271D uCosminexus TP1/NET/High Availability(64) 07-50

・適用 OS：Red Hat Enterprise Linux Server 6 (32-bit x86)，Red Hat Enterprise Linux Server 6 (64-bit x86_64)，Red Hat Enterprise Linux Server 7 (64-bit x86_64)，Red Hat Enterprise Linux Server 8 (64-bit x86_64)

P-8164-2111 uCosminexus TP1/Server Base 07-57

P-8164-2311 uCosminexus TP1/FS/Direct Access 07-51
P-8164-2411 uCosminexus TP1/FS/Table Access 07-51
P-8164-2911 uCosminexus TP1/High Availability 07-50
P-8164-8511 uCosminexus TP1/Extension 1 07-50
P-8164-3111 uCosminexus TP1/Message Control 07-52
P-8164-3211 uCosminexus TP1/NET/Library 07-52
P-8164-C311 uCosminexus TP1/Message Queue 07-53
P-F8164-31112 uCosminexus TP1/Message Control - Extension 1 07-50
P-F8164-32111 uCosminexus TP1/NET/User Agent 07-50
P-F8164-3211C uCosminexus TP1/NET/TCP/IP 07-52
P-F8164-3211D uCosminexus TP1/NET/High Availability 07-50
P-F8164-3211U uCosminexus TP1/NET/User Datagram Protocol 07-50
R-1945F-12 uCosminexus TP1/Web 07-50

・適用 OS：Red Hat Enterprise Linux Server 6 (64-bit x86_64)，Red Hat Enterprise Linux Server 7 (64-bit x86_64)，Red Hat Enterprise Linux Server 8 (64-bit x86_64)

P-8264-2111 uCosminexus TP1/Server Base(64) 07-57
P-8264-2311 uCosminexus TP1/FS/Direct Access(64) 07-51
P-8264-2411 uCosminexus TP1/FS/Table Access(64) 07-51
P-8264-2911 uCosminexus TP1/High Availability(64) 07-50
P-8264-8511 uCosminexus TP1/Extension 1(64) 07-50
P-8264-3111 uCosminexus TP1/Message Control(64) 07-52
P-8264-3211 uCosminexus TP1/NET/Library(64) 07-52
P-8264-C311 uCosminexus TP1/Message Queue(64) 07-53
P-F8264-31112 uCosminexus TP1/Message Control - Extension 1(64) 07-50
P-F8264-3211C uCosminexus TP1/NET/TCP/IP(64) 07-52
P-F8264-3211D uCosminexus TP1/NET/High Availability(64) 07-50
P-F8264-3211U uCosminexus TP1/NET/User Datagram Protocol(64) 07-50
R-1S45F-12 uCosminexus TP1/Web(64) 07-50

・適用 OS：Windows 7，Windows 7 x64 Edition，Windows 8，Windows 8 x64 Edition，Windows 8.1，Windows 8.1 x64 Edition，Windows 10，Windows 10 x64 Edition，Windows 11，Windows Server 2008 R2，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2464-2154 uCosminexus TP1/Client/P 07-51
P-2464-2284 uCosminexus TP1/Server Base 07-53

P-2464-2384 uCosminexus TP1/FS/Direct Access 07-51
P-2464-2484 uCosminexus TP1/FS/Table Access 07-51
P-2464-2554 uCosminexus TP1/Extension 1 07-51
P-2464-3164 uCosminexus TP1/Message Control 07-51
P-2464-3264 uCosminexus TP1/NET/Library 07-50
P-2464-7834 uCosminexus TP1/Client for .NET Framework 07-50
P-2464-C384 uCosminexus TP1/Message Queue 07-52
P-F2464-31642 uCosminexus TP1/Message Control - Extension 1 07-50
P-F2464-32645 uCosminexus TP1/NET/XMAP3 07-51
P-F2464-3264C uCosminexus TP1/NET/TCP/IP 07-51
P-F2464-3264D uCosminexus TP1/NET/High Availability 07-51
R-1545B-27 uCosminexus TP1/LiNK 07-51
R-15451-27 uCosminexus TP1/Connector for .NET Framework 07-50
R-15452-27 uCosminexus TP1/Extension for .NET Framework 07-50

・適用 OS：Windows 7 x64 Edition，Windows 8 x64 Edition，Windows 8.1 x64 Edition，Windows 10 x64 Edition，Windows 11，Windows Server 2008 R2，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2964-2124 uCosminexus TP1/Client/P(64) 07-51
P-2964-2224 uCosminexus TP1/Server Base(64) 07-53
P-2964-2324 uCosminexus TP1/FS/Direct Access(64) 07-51
P-2964-2424 uCosminexus TP1/FS/Table Access(64) 07-51
P-2964-2524 uCosminexus TP1/Extension 1(64) 07-50
P-2964-3124 uCosminexus TP1/Message Control(64) 07-51
P-2964-3224 uCosminexus TP1/NET/Library(64) 07-50
P-2964-C324 uCosminexus TP1/Message Queue(64) 07-52
P-F2964-3224C uCosminexus TP1/NET/TCP/IP(64) 07-51
P-F2964-3224D uCosminexus TP1/NET/High Availability(64) 07-50

・適用 OS：Windows Server 2008 R2，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2464-2944 uCosminexus TP1/High Availability 07-51
P-2964-2924 uCosminexus TP1/High Availability(64) 07-50

・適用 OS：Windows 10，Windows 10 x64 Edition，Windows 11，Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2464-2294 uCosminexus TP1/Server Base 07-60
P-2464-2564 uCosminexus TP1/Extension 1 07-60
P-2464-2164 uCosminexus TP1/Client/P 07-60

・適用 OS：Windows 10 x64 Edition，Windows 11，Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2964-2234 uCosminexusTP1/Server Base(64) 07-60
P-2964-2534 uCosminexus TP1/Extension 1(64) 07-60
P-2964-2134 uCosminexus TP1/Client/P(64) 07-60

・適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022

P-2464-2954 uCosminexus TP1/High Availability 07-60
P-2964-2934 uCosminexus TP1/High Availability(64) 07-60

・適用 OS：Java VM

P-2464-73B4 uCosminexus TP1/Client/J 07-53

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

## ■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。
なお、不明な場合は、弊社担当営業にお問い合わせください。

## ■ 商標類

## ■ 発行

2023 年 7 月　3000-3-D50-B0

## ■ 著作権

# 変更内容

## 変更内容（3000-3-D50-B0） uCosminexus TP1/Server Base 07-60，uCosminexus TP1/Server Base(64) 07-60

| 追加・変更内容 | 変更個所 |
|---|---|
| マニュアル訂正の内容を反映した。 | － |
| ソケット受信型サーバは廃止したため使用できないことを記載した。 | 3.4.1(4) |
| Docker および Kubernetes を使用したコンテナ環境に関する動作について説明を追加した。 | 8.2.1 |
| システム構成に関する次の説明を追加，変更した。<br>• システム構成図の矢印について誤りを訂正<br>• システム構成図に Windows プラットフォームを追加<br>• システム構成図に示す製品のバージョンに関する説明を追加<br>また，上記変更に伴い，次に示す表を変更した。<br>• 表 8-2 サポートバージョン<br>• 表 8-3 旧バージョンの TP1/Server Base からの RPC 通信可否<br>• 表 8-4 旧バージョンの TP1/Client/J からの RPC 通信可否<br>• 表 8-5 旧バージョンの TP1/Server Base への RPC 通信可否<br>• 表 8-6 TP1/Client/J から旧バージョンの TP1/Server Base への RPC 通信可否<br>• 表 8-7 他製品から TP1/Server Base への RPC<br>• 表 8-8 TP1/Server Base から他製品への RPC | 8.3，8.3.1(1)，8.3.2(1)，8.3.2(2)，8.3.3(1)，8.3.3(2) |

単なる誤字・脱字などはお断りなく訂正しました。

## 変更内容（3000-3-D50-A0） uCosminexus TP1/Server Base 07-57，uCosminexus TP1/Server Base(64) 07-57，uCosminexus TP1/Server Base 07-56，uCosminexus TP1/Server Base(64) 07-56

| 追加・変更内容 |
|---|
| マニュアル訂正の内容を反映した。 |
| Docker および Kubernetes を使用した OpenTP1 の構成と構築の説明を追加した。 |
| 次に示すバージョンの変更点の説明を追加した。<br>• TP1/Server Base 07-56 |

### 変更内容（3000-3-D50-91） uCosminexus TP1/Server Base 07-54，uCosminexus TP1/Server Base(64) 07-54，uCosminexus TP1/Server Base 07-53，uCosminexus TP1/Server Base(64) 07-53

| 追加・変更内容 |
| --- |
| マニュアル訂正の内容を反映した。 |
| 常駐/非常駐プロセスによるマルチサーバの概要図を変更した。 |
| 系切り替え機能を使用する場合の説明を追加した。 |
| キャッシュブロック数しきい値指定機能に関する説明を変更した。 |
| 回復対象外の DAM ファイルのキャッシュレスアクセスに関する説明を追加した。 |
| 共有ディスク装置の割り当てに関する説明を変更した。 |
| システムサービスプロセスの説明を追加および変更した。 |
| 次に示すバージョンの変更点の説明を変更した。<br>• TP1/Server Base 07-07 |

### 変更内容（3000-3-D50-90） uCosminexus TP1/Server Base 07-53，uCosminexus TP1/Server Base(64) 07-53

| 追加・変更内容 |
| --- |
| 販売終了製品に関連する機能の記載を削除した。 |

### 変更内容（3000-3-D50-80） uCosminexus TP1/Server Base 07-53，uCosminexus TP1/Server Base(64) 07-53，uCosminexus TP1/Server Base 07-52，uCosminexus TP1/Server Base(64) 07-52

| 追加・変更内容 |
| --- |
| OpenTP1 監視サービスの説明を追加した。 |
| OpenTP1 の状態確認に関する説明を追加した。 |
| OpenTP1 の監視に関する説明を追加した。 |
| 強制停止処理中の系切り替え抑止機能に関する説明を追加した。 |
| 入出力ファイルの記述を削除した。 |
| 共用メモリの使用状況の表示に関する記述を削除した。 |
| OpenTP1 監視サービスで使用する共用メモリの説明を追加した。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-53<br>• TP1/Server Base 07-52 |

### 変更内容（3000-3-D50-71） uCosminexus TP1/Server Base 07-51，uCosminexus TP1/Server Base(64) 07-51，uCosminexus TP1/Message Control 07-51，uCosminexus TP1/Message Control(64) 07-51，uCosminexus TP1/NET/Library 07-51，uCosminexus TP1/NET/Library(64) 07-51

| 追加・変更内容 |
| --- |
| マニュアル訂正の内容を反映した。 |

### 変更内容（3000-3-D50-70） uCosminexus TP1/Server Base 07-51，uCosminexus TP1/Server Base(64) 07-51，uCosminexus TP1/Message Control 07-51，uCosminexus TP1/Message Control(64) 07-51，uCosminexus TP1/NET/Library 07-51，uCosminexus TP1/NET/Library(64) 07-51

| 追加・変更内容 |
| --- |
| ERREVT2 が通知された原因に関する説明を追加した。 |
| MHP の再スケジュールに関する説明を追加した。 |
| ノードをわたった OpenTP1 プロセス間の TCP/IP 通信の場合に限り，通信処理を終了した時点で毎回コネクションを切断するノード間通信時の毎回コネクション断機能の説明を追加した。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-51<br>• TP1/Message Control 07-51 および TP1/NET/Library 07-51 |
| 次に示すバージョンの変更点の説明を変更した。<br>• TP1/Server Base 07-50 |
| 次に示すバージョンの変更点の説明を変更した。<br>• TP1/Server Base 07-00 |

### 変更内容（3000-3-D50-60） uCosminexus TP1/Server Base 07-50，uCosminexus TP1/Server Base(64) 07-50，uCosminexus TP1/Message Control 07-50，uCosminexus TP1/Message Control(64) 07-50，uCosminexus TP1/NET/Library 07-50，uCosminexus TP1/NET/Library(64) 07-50

| 追加・変更内容 |
| --- |
| XA リソースサービス使用時のタイマ監視範囲の説明を変更した。 |
| 注意事項に，起動通知機能を使用できない理由の説明を追加した。 |
| 注意事項に，ノード監視機能を使用できない理由の説明を追加した。 |
| メッセージ制御機能における OpenTP1 再開始時のメッセージの扱いに関する説明を追加した。 |
| 入力キューと出力キューの保留に関する説明を追加した。 |
| 閉塞されている論理端末のメッセージが未処理送信メッセージ滞留時間の対象とならないことを追加した。 |

| 追加・変更内容 |
| --- |
| dc_mcf_execap 関数でアプリケーションを起動する順序に関する説明を追加した。 |
| SPP のスケジュールでの，スケジュール対象の選択について説明を追加した。 |
| マルチサーバ負荷バランス機能の説明を変更した。 |
| ダイナミックコネクションスケジュールモードの説明を変更した。 |
| ジャーナル容量に対する考え方について，説明を追加した。 |
| TAM ファイルのバックアップとリストアの説明を変更した。 |
| TAM ファイルからのユーザデータの抽出と，TAM ファイルの再作成について説明を追加した。 |
| システムサービスプロセスに prctee を追加した。 |
| 受信用ポート番号を指定できるシステム定義のオペランドに，rpc_port_base を追加した。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-50<br>• TP1/Message Control 07-50 および TP1/NET/Library 07-50<br>• TP1/Server Base 07-07 |
| 次に示すバージョンの変更点の説明を変更した。<br>• TP1/Server Base 07-06<br>• TP1/Server Base 07-05<br>• TP1/Server Base 07-02<br>• TP1/Server Base 07-00 |

## 変更内容（3000-3-D50-50） uCosminexus TP1/Server Base 07-06，uCosminexus TP1/Server Base(64) 07-06

| 追加・変更内容 |
| --- |
| OpenTP1 の拡張機能を使うときに必要な製品（TP1/Extension 1，および TP1/Message Control - Extension1）に関する説明を追加した。 |
| ノード構成の変更（ノードの追加や削除）に自動的に対応する機能（ノード自動追加機能）を追加した。これに伴い，ノードリストファイルを追加した。 |
| ユーザサーバごとに，共有化したバッファの使用サイズを制限できるようにした。 |
| TP1/Client/J が DCCM3 と通信する場合の説明を変更した。 |
| rap クライアントマネジャの管理テーブルについての説明を追加した。 |
| 性能検証用トレースの情報を CSV 形式で出力し，トレース解析できるようにした。 |
| ネームサービス定義の name_audit_conf オペランドに 1 または 2 を指定している場合，実行形式ファイル namaudtd の稼働数が 1 になる旨を追加した。 |
| OpenTP1 で使用している受信用ポート番号 domain_masters_port を削除した。 |

| 追加・変更内容 |
| --- |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-06<br>• TP1/Server Base 07-05<br>• TP1/Server Base 07-02<br>• TP1/Message Control 07-01 および TP1/NET/Library 07-02<br>• TP1/Message Control 07-00 および TP1/NET/Library 07-00 |
| 一つのサービス要求ごとに実行するプロセスを起動し直せるようにした（非常駐 UAP プロセスのリフレッシュ機能）。 |

### uCosminexus TP1/Server Base 07-05，uCosminexus TP1/Server Base(64) 07-05，uCosminexus TP1/Message Control 07-05，uCosminexus TP1/Message Control(64) 07-05

| 追加・変更内容 |
| --- |
| 入力キューと出力キューについて説明を変更した。 |
| メッセージ出力通番について説明を追加した。 |
| 一つのリソースマネジャを複数の制御単位に分け，接続するユーザ名称などを変更してリソースマネジャに接続できるようにした（リソースマネジャ接続先選択機能）。 |
| システムの稼働数についての説明を追加した。 |

### uCosminexus TP1/Server Base 07-02，uCosminexus TP1/Message Control 07-01

| 追加・変更内容 |
| --- |
| 前回のオンライン停止時に残っていた未処理受信メッセージや未送信メッセージを引き継ぎ，TP1/Message Control の構成を変更できるようにした（MCF 構成変更再開始機能）。<br>また，OpenTP1 の日常的な運用に OpenTP1 の MCF 構成変更準備停止を追加した。 |

### uCosminexus TP1/Message Control 07-00，uCosminexus TP1/Message Control(64) 07-00

| 追加・変更内容 |
| --- |
| リモート MCF サービスに関連する記述を削除した。 |

### 変更内容（3000-3-D50-40） uCosminexus TP1/Server Base 07-04，uCosminexus TP1/Server Base(64) 07-04，uCosminexus TP1/Message Control 07-05，uCosminexus TP1/Message Control(64) 07-05，uCosminexus TP1/NET/Library 07-05，uCosminexus TP1/NET/Library(64) 07-05

| 追加・変更内容 |
| --- |
| 異常終了した MHP を，自動的に再スケジュールできるようにした。 |
| チェックポイントダンプ取得契機のスキップ回数を監視できるようにした。 |

| 追加・変更内容 |
| --- |
| 次に示すシステムサービスプロセスの説明を変更した。<br>• trnrvd<br>• jnlsdd<br>• admrsvre |
| OpenTP1 で使用しているポート番号の説明を変更した。 |
| サービス関数動的ローディング機能で使用する，UAP 共用ライブラリのサーチパスをオンライン中に変更できるようにした。 |
| OpenTP1 の起動コマンドがリターンした直後に MCF の運用コマンドを実行する場合，mcftlscom コマンドで MCF 通信サービスの開始を待ち合わせられるようにした。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Message Control 07-05 および TP1/NET/Library 07-05<br>• TP1/Server Base 07-04<br>• TP1/Message Control 07-02 および TP1/NET/Library 07-03<br>• TP1/Server Base 07-00 |

## 変更内容（3000-3-D50-30） uCosminexus TP1/Server Base 07-03，uCosminexus TP1/Server Base(64) 07-03，uCosminexus TP1/Message Control 07-03，uCosminexus TP1/Message Control(64) 07-03，uCosminexus TP1/NET/Library 07-04，uCosminexus TP1/NET/Library(64) 07-04

| 追加・変更内容 |
| --- |
| ジャーナルサービスで性能検証用トレース（JNL 性能検証用トレース）を出力できるようにした。 |
| ロックサービスを使用した排他制御の各種イベントの性能検証用トレース（LCK 性能検証用トレース）を出力できるようにした。 |
| グローバルドメインの説明を追加した。 |
| 特定のノードのサービス情報を優先的に使用する機能（サービス情報優先度指定機能）を追加した。 |
| UAP 異常終了通知イベント，および未処理送信メッセージ廃棄通知イベントの，MCF イベントが通知された原因の説明を変更した。 |
| 縮退運転の説明を変更した。 |
| メッセージが回復できない場合の説明を追加した。 |
| メッセージの送信順序の説明を変更した。 |
| dc_rpc_cltsend 関数で通知したデータを受け取る関数の説明を変更した。 |
| OpenTP1 ファイルシステムと OS が提供するファイルシステムとの違いの説明を変更した。 |
| OpenTP1 ファイルへのアクセス要求で，イベントトレース（FIL イベントトレース）を出力できるようにした。 |
| キャッシュレスアクセス指定の DAM ファイルを使用する場合の DAM サービス専用共用メモリサイズについて説明を変更した。 |

| 追加・変更内容 |
| --- |
| 性能検証用トレース取得サービスの稼働数を変更した。また，関連する定義を変更した。 |
| 受信用ポート番号を指定できるシステムサービスについて説明を追加した。 |
| アプリケーションに関するタイマ起動要求の状態を表示できるようにした。 |
| OpenTP1 の標準出力，標準エラー出力をリダイレクトする prctee プロセスを停止・再開始できるようにした。 |
| ユーザタイマ監視の状態を表示できるようにした。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-03<br>• TP1/Message Control 07-03 および TP1/NET/Library 07-04<br>• TP1/Message Control 07-02 および TP1/NET/Library 07-03<br>• TP1/Message Control 07-01 および TP1/NET/Library 07-01 |
| リモートプロシジャコールの処理の概要を追加した。 |

## uCosminexus TP1/Message Control 07-02，uCosminexus TP1/NET/Library 07-03

| 追加・変更内容 |
| --- |
| MHP でサービス関数動的ローディング機能を使用できるようにした。 |
| 次の操作を，ライブラリ関数でできるようにした。<br>• コネクションの状態表示，確立，および解放<br>• サーバ型コネクションの確立要求の状態表示，および受付開始・終了<br>• アプリケーションに関するタイマ起動要求の削除<br>• 論理端末の状態表示，閉塞，閉塞解除，および出力キューの削除<br>• MCF 通信サービスまたはアプリケーション起動サービスの状態表示 |
| MCF 静的共用メモリが不足し未使用領域から自動的に追加確保する場合に，メッセージを出力できるようにした。 |
| 相手システムとのメッセージ送受信に関するネットワークの状態を表示できるようにした。 |

## uCosminexus TP1/Message Control 07-01，uCosminexus TP1/NET/Library 07-01

| 追加・変更内容 |
| --- |
| メッセージ送受信での主なイベントで，性能検証用トレース（MCF 性能検証用トレース）を出力できるようにした。 |
| サーバ型コネクションの確立要求の受付開始・終了を手動でできるようにした。 |

## 変更内容（3000-3-D50-20） uCosminexus TP1/Server Base 07-02，uCosminexus TP1/Message Control 07-01，uCosminexus TP1/NET/Library 07-01

| 追加・変更内容 |
| --- |
| XA リソースサービスで性能検証用トレース（prf トレース）を出力できるようにした。 |

| 追加・変更内容 |
| --- |
| サービス関数を動的にローディングできる機能を追加した。 |
| タイマ監視機能によるタイマ監視の対象を追加した。また，監視対象に対応する OpenTP1 の定義を変更した。 |
| スケジュールサービスの動作をサービス単位で指定できるようにした。 |
| MS-DOS 用の TP1/Client/P でのユーザ認証機能の使い方を削除した。 |
| リモート API 機能に関する説明を変更した。 |
| 監査ログを出力する機能を追加した。 |
| システムジャーナルファイルの並列アクセス機能を追加した。 |
| ネームサービスでイベントトレースを出力できるようにした。 |
| プロセスサービスでイベントトレースを出力できるようにした。 |
| システムジャーナルファイルを使用しないでシステムを運用する機能（ジャーナルファイルレス機能）を追加した。<br>これに伴い，システムサービスプロセスの稼働数を変更した。 |
| 次に示すバージョンの変更点を記載した。<br>• TP1/Server Base 07-02<br>• TP1/Message Control 07-01 および TP1/NET/Library 07-01 |

# はじめに

このマニュアルは，分散トランザクション処理機能 OpenTP1 の概要について説明したものです。

本文中に記載されている製品のうち，このマニュアルの対象製品ではない製品については，OpenTP1 Version 7 対応製品の発行時期をご確認ください。

TP1/Message Queue は，米国 International Business Machines Corporation とのライセンス契約に基づき，IBM MQ（旧称：WebSphere MQ または MQSeries）の MQI，MQFAP，MQ クラスタの仕様をベースに実装しています。

次に示す製品，および各製品に示したバージョン以降で，ソケット受信型サーバに関する機能はすべて廃止しました。

- P-1M64-2141 uCosminexus TP1/Server Base：07-53-01 以降
- P-1M64-1121 uCosminexus TP1/Server Base(64)：07-53-01 以降
- P-1J64-2171 uCosminexus TP1/Server Base：07-51-02 以降
- P-1J64-1171 uCosminexus TP1/Server Base(64)：07-51-01 以降
- P-8164-2111 uCosminexus TP1/Server Base：07-57 以降
- P-8264-2111 uCosminexus TP1/Server Base(64)：07-57 以降
- P-2464-2294 uCosminexus TP1/Server Base：07-60 以降
- P-2964-2234 uCosminexus TP1/Server Base(64)：07-60 以降

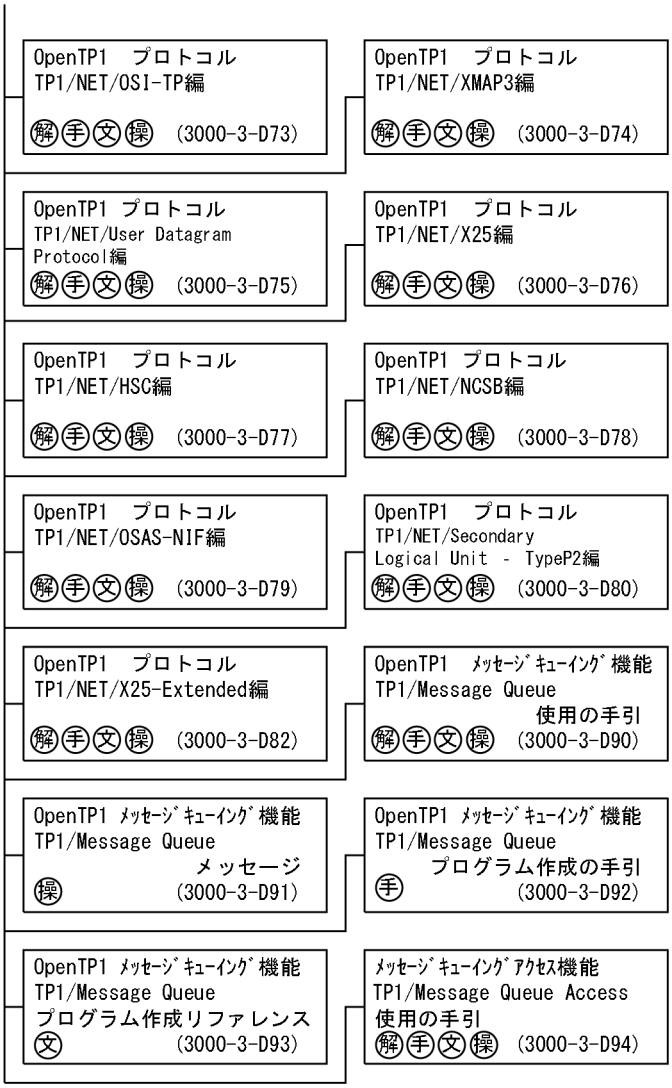
なお，該当する機能を使用した場合の動作は保証できないため，ご注意ください。

## ■ 対象読者

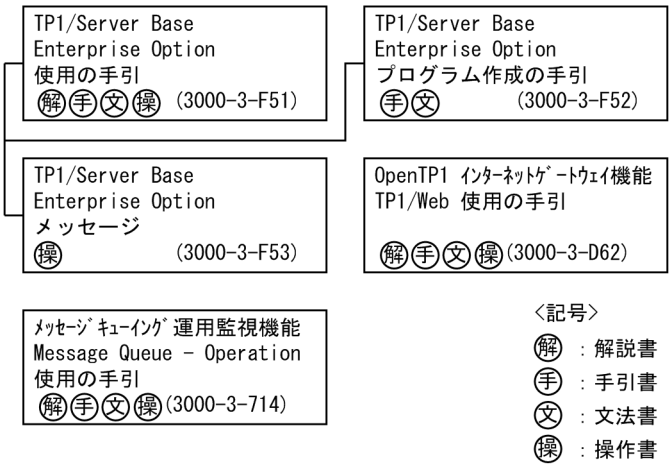
システム管理者，システム設計者，プログラマ，およびオペレータの方を対象としています。なお，オペレーティングシステム，オンラインシステムの基礎的な知識があることを前提としています

## ■ 関連マニュアル

●OpenTP1 Version 7




●そのほかのOpenTP1関連




マニュアル「OpenTP1 プロトコル」の各プロトコル編については，ご使用の製品のバージョンに対応するマニュアルの発行時期をご確認ください。

## ●関連製品

- 索引順編成ファイル管理 ISAM（3000-3-046）
- Linux(R)，HP-UX　通信管理　XNF/LS　使用の手引（3000-3-B51）
- AIX 通信管理 XNF/AS 解説・運用編（3000-3-B61）
- 高信頼化システム監視機能 HA モニタ AIX(R)編（3000-9-130）
- 高信頼化システム監視機能 HA モニタ AIX(R)編（3000-9-202）
- 高信頼化システム監視機能 HA モニタ Linux(R)編（3000-9-132）
- 高信頼化システム監視機能 HA モニタ HP-UX（IPF）編（3000-9-133）
- 高信頼化システム監視機能 HA モニタ メッセージ（3000-9-134）
- 高信頼化システム監視機能 HA モニタ Linux(R)（x86）編（3000-9-140）
- Hitachi HA Toolkit（3000-9-115）
- JP1 Version 8 JP1/Automatic Job Management System 2 - Scenario Operation（3020-3-K42）
- JP1 Version 8 JP1/ServerConductor/Deployment Manager（3020-3-L53）
- JP1 Version 9 JP1/ServerConductor/Deployment Manager（3020-3-T77）
- SEWB+ クライアントサーバシステム開発ガイド（3020-3-N83）
- HiRDB Version 9 システム導入・設計ガイド（UNIX(R)用）（3000-6-452）
- HiRDB Version 9 システム導入・設計ガイド（Windows(R)用）（3020-6-452）
- HiRDB Version 9 UAP 開発ガイド（3020-6-456）
- HiRDB Version 10 システム導入・設計ガイド（UNIX(R)用）（3020-6-552）
- HiRDB Version 10 システム導入・設計ガイド（Windows (R)用）（3020-6-553）
- HiRDB Version 10 UAP 開発ガイド（3020-6-560）
- XMAP3 Version 5 画面・帳票サポートシステム XMAP3 実行ガイド（3020-7-514）
- JP1 Version 10 JP1/Base 運用ガイド（3021-3-001）
- JP1 Version 10 JP1/Base メッセージ（3021-3-002）
- JP1 Version 10 JP1/Base 関数リファレンス（3021-3-003）
- JP1 Version 10 JP1/Automatic Job Management System 3 入門（3021-3-101）
- JP1 Version 10 JP1/Automatic Job Management System 3 導入ガイド（3021-3-102）
- JP1 Version 10 JP1/Automatic Job Management System 3 設計ガイド（システム構築編）（3021-3-103）
- JP1 Version 10 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）（3021-3-104）

- JP1 Version 10 JP1/Automatic Job Management System 3 構築ガイド 1（3021-3-105）
- JP1 Version 10 JP1/Automatic Job Management System 3 構築ガイド 2（3021-3-106）
- JP1 Version 10 JP1/Automatic Job Management System 3 運用ガイド（3021-3-107）
- JP1 Version 10 JP1/Automatic Job Management System 3 トラブルシューティング（3021-3-108）
- JP1 Version 10 JP1/Automatic Job Management System 3 操作ガイド（3021-3-109）
- JP1 Version 10 JP1/Automatic Job Management System 3 コマンドリファレンス 1（3021-3-110）
- JP1 Version 10 JP1/Automatic Job Management System 3 コマンドリファレンス 2（3021-3-111）
- JP1 Version 10 JP1/Automatic Job Management System 3 連携ガイド（3021-3-112）
- JP1 Version 10 JP1/Automatic Job Management System 3 メッセージ 1（3021-3-113）
- JP1 Version 10 JP1/Automatic Job Management System 3 メッセージ 2（3021-3-114）
- JP1 Version 10 JP1/Audit Management - Manager 構築・運用ガイド（3021-3-165）
- JP1 Version 11 JP1/Base 運用ガイド（3021-3-A01）
- JP1 Version 11 JP1/Base メッセージ（3021-3-A02）
- JP1 Version 11 JP1/Base 関数リファレンス（3021-3-A03）
- JP1 Version 11 ジョブ管理 基本ガイド（ジョブスケジューラー編）（3021-3-B11）
- JP1 Version 11 JP1/Automatic Job Management System 3 導入ガイド（3021-3-B12）
- JP1 Version 11 JP1/Automatic Job Management System 3 設計ガイド（システム構築編）（3021-3-B13）
- JP1 Version 11 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）（3021-3-B14）
- JP1 Version 11 JP1/Automatic Job Management System 3 構築ガイド（3021-3-B15）
- JP1 Version 11 JP1/Automatic Job Management System 3 運用ガイド（3021-3-B16）
- JP1 Version 11 JP1/Automatic Job Management System 3 トラブルシューティング（3021-3-B17）
- JP1 Version 11 JP1/Automatic Job Management System 3 操作ガイド（3021-3-B18）
- JP1 Version 11 JP1/Automatic Job Management System 3 コマンドリファレンス（3021-3-B19）
- JP1 Version 11 JP1/Automatic Job Management System 3 連携ガイド（3021-3-B20）
- JP1 Version 11 JP1/Automatic Job Management System 3 メッセージ（3021-3-B21）
- JP1 Version 11 JP1/Audit Management - Manager 構築・運用ガイド（3021-3-A17）
- JP1 Version 12 JP1/Base 運用ガイド（3021-3-D65）

- JP1 Version 12 JP1/Base メッセージ（3021-3-D66）
- JP1 Version 12 JP1/Base 関数リファレンス（3021-3-D67）
- JP1 Version 12 ジョブ管理 基本ガイド（ジョブスケジューラー編）（3021-3-D20）
- JP1 Version 12 JP1/Automatic Job Management System 3 導入ガイド（3021-3-D21）
- JP1 Version 12 JP1/Automatic Job Management System 3 設計ガイド（システム構築編）（3021-3-D22）
- JP1 Version 12 JP1/Automatic Job Management System 3 設計ガイド（業務設計編）（3021-3-D23）
- JP1 Version 12 JP1/Automatic Job Management System 3 構築ガイド（3021-3-D24）
- JP1 Version 12 JP1/Automatic Job Management System 3 運用ガイド（3021-3-D25）
- JP1 Version 12 JP1/Automatic Job Management System 3 トラブルシューティング（3021-3-D26）
- JP1 Version 12 JP1/Automatic Job Management System 3 操作ガイド（3021-3-D27）
- JP1 Version 12 JP1/Automatic Job Management System 3 コマンドリファレンス（3021-3-D28）
- JP1 Version 12 JP1/Automatic Job Management System 3 連携ガイド（3021-3-D29）
- JP1 Version 12 JP1/Automatic Job Management System 3 メッセージ（3021-3-D30）

このマニュアルでは，次のマニュアルについて，名称を省略して表記しています。マニュアルの正式名称とこのマニュアルでの表記を次に示します。

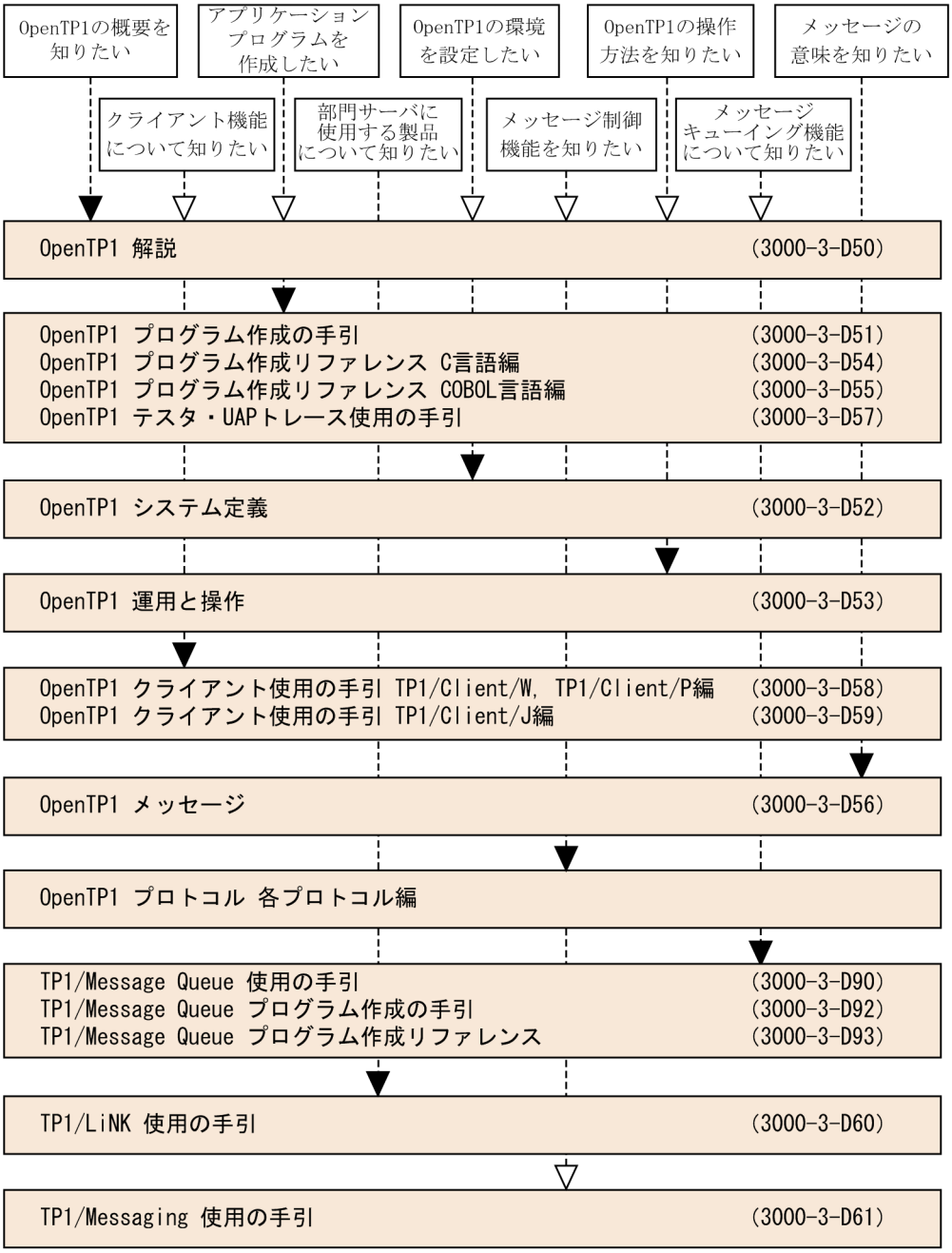
| 正式名称 | このマニュアルでの表記 |
|---|---|
| OpenTP1　クライアント使用の手引 TP1/Client/W，TP1/Client/P 編 | OpenTP1　クライアント使用の手引 |
| OpenTP1　クライアント使用の手引　TP1/Client/J 編 | |
| OpenTP1 メッセージキューイング機能 TP1/Message Queue 使用の手引 | TP1/Message Queue 使用の手引 |
| OpenTP1 メッセージキューイング機能 TP1/Message Queue プログラム作成の手引 | TP1/Message Queue プログラム作成の手引 |
| OpenTP1 メッセージキューイング機能 TP1/Message Queue プログラム作成リファレンス | TP1/Message Queue プログラム作成リファレンス |
| メッセージキューイングアクセス機能 TP1/Message Queue Access 使用の手引 | TP1/Message Queue Access 使用の手引 |
| メッセージキューイング運用監視機能 Message Queue - Operation 使用の手引 | Message Queue - Operation 使用の手引 |
| 高信頼化システム監視機能 HA モニタ AIX(R)編 | 高信頼化システム監視機能 HA モニタ |

| 正式名称 | このマニュアルでの表記 | |
|---|---|---|
| 高信頼化システム監視機能 HA モニタ Linux(R)編 | 高信頼化システム監視機能 HA モニタ | |
| 高信頼化システム監視機能 HA モニタ Linux(R)（x86）編 | | |
| 高信頼化システム監視機能 HA モニタ HP-UX（IPF）編 | | |
| 高信頼化システム監視機能 HA モニタ メッセージ | | |
| JP1 Version 10 JP1/Base 運用ガイド | JP1/Base 運用ガイド | JP1/Base |
| JP1 Version 11 JP1/Base 運用ガイド | | |
| JP1 Version 12 JP1/Base 運用ガイド | | |
| JP1 Version 10 JP1/Base メッセージ | JP1/Base メッセージ | |
| JP1 Version 11 JP1/Base メッセージ | | |
| JP1 Version 12 JP1/Base メッセージ | | |
| JP1 Version 10 JP1/Base 関数リファレンス | JP1/Base 関数リファレンス | |
| JP1 Version 11 JP1/Base 関数リファレンス | | |
| JP1 Version 12 JP1/Base 関数リファレンス | | |
| JP1 Version 10 JP1/Automatic Job Management System 3 構築ガイド 1 | JP1/Automatic Job Management System 構築ガイド | |
| JP1 Version 10 JP1/Automatic Job Management System 3 構築ガイド 2 | | |
| JP1 Version 11 JP1/Automatic Job Management System 3 構築ガイド | | |
| JP1 Version 12 JP1/Automatic Job Management System 3 構築ガイド | | |
| JP1 Version 10 JP1/Automatic Job Management System 3 操作ガイド | JP1/Automatic Job Management System 操作ガイド | |
| JP1 Version 11 JP1/Automatic Job Management System 3 操作ガイド | | |
| JP1 Version 12 JP1/Automatic Job Management System 3 操作ガイド | | |
| JP1 Version 8 JP1/Automatic Job Management System 2 - Scenario Operation | JP1/Automatic Job Management System 2 - Scenario Operation | |
| JP1 Version 10 JP1/Audit Management - Manager 構築・運用ガイド | JP1/NETM/Audit | |
| JP1 Version 11 JP1/Audit Management - Manager 構築・運用ガイド | | |

| 正式名称 | このマニュアルでの表記 |
|---|---|
| JP1 Version 8 JP1/ServerConductor/Deployment Manager | JP1/ServerConductor/Deployment Manager |
| JP1 Version 9 JP1/ServerConductor/Deployment Manager | |
| HiRDB Version 9 システム導入・設計ガイド（UNIX(R)用） | HiRDB システム導入・設計ガイド |
| HiRDB Version 10 システム導入・設計ガイド（UNIX(R)用） | |
| HiRDB Version 9 システム導入・設計ガイド（Windows(R)用） | |
| HiRDB Version 10 システム導入・設計ガイド（Windows(R)用） | |
| HiRDB Version 9 UAP 開発ガイド | HiRDB UAP 開発ガイド |
| HiRDB Version 10 UAP 開発ガイド | |

## ■ 関連マニュアルの読書手順

OpenTP1 の一連の関連マニュアルは，利用目的に合わせて，選択して読むことができます。次の案内に従ってお読みいただくことをお勧めします。

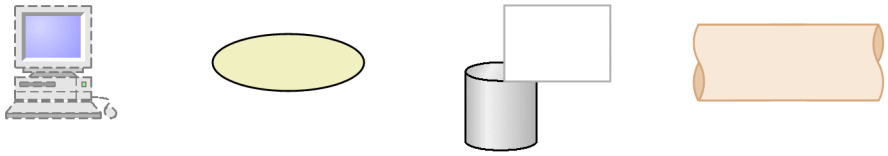



（凡例）▼ ：必ず読むマニュアル
▽ ：必要に応じて読むマニュアル

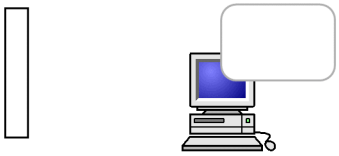
## ■ 図中で使用する記号

このマニュアルの図中で使用する記号を，次のように定義します。

●論理端末　●ネットワーク　●ファイルの内容　●コネクション



●プログラムの流れ　●画面の表示



## ■ このマニュアルでの表記

### (1)製品名

このマニュアルでは，製品の名称を省略して表記しています。製品の名称と，このマニュアルでの表記を次に示します。

| 製品名称 | 略称 | |
|---|---|---|
| AIX V6.1 | AIX | |
| AIX V7.1 | | |
| AIX V7.2 | | |
| AIX V7.3 | | |
| COBOL2002 Net Developer | COBOL | |
| COBOL2002 Net Server Runtime | | |
| COBOL2002 Net Server Runtime(64) | | |
| COBOL2002 Net Server Suite | | |
| COBOL2002 Net Server Suite(64) | | |
| COBOL85 | | |
| JP1/ServerConductor/Deployment Manager | DPM | |
| HiRDB Server Version 9 | HiRDB | |
| HiRDB Server Version 10 | | |
| HP-UX 11i V2 (IPF) | HP-UX (IPF) | HP-UX |
| HP-UX 11i V3 (IPF) | | |
| Itanium Processor Family | IPF | |

| 製品名称 | 略称 | | |
|---|---|---|---|
| Java Virtual Machine | Java VM | | |
| JP1/Automatic Job Management System 3 - Agent | JP1/AJS - Agent | JP1/AJS | JP1 |
| JP1/Automatic Job Management System 3 - Manager | JP1/AJS - Manager | | |
| JP1/Automatic Job Management System 3 - View | JP1/AJS - View | | |
| JP1/Automatic Job Management System 2 - Scenario Operation Manager | JP1/AJS2 - Scenario Operation Manager | JP1/AJS2 - Scenario Operation | |
| JP1/Automatic Job Management System 2 - Scenario Operation View | JP1/AJS2 - Scenario Operation View | | |
| JP1/Audit Management - Manager | JP1/NETM/Audit | | |
| Red Hat Enterprise Linux Server 6 (32-bit x86) | Linux (AMD64／Intel EM64T／x86) | Linux | |
| Red Hat Enterprise Linux Server 6 (64-bit x86_64) | | | |
| Red Hat Enterprise Linux Server 7 (64-bit x86_64) | | | |
| Red Hat Enterprise Linux Server 8 (64-bit x86_64) | | | |
| JP1/NETM/DM Client | NETM/DM | | |
| JP1/NETM/DM Manager | | | |
| JP1/NETM/DM SubManager | | | |
| Oracle 9i Database | Oracle | | |
| Oracle Database 10g | | | |
| Oracle Database 11g | | | |
| Oracle Database 12c | | | |
| Solaris 8 | Solaris | | |
| Solaris 9 | | | |
| Solaris 10 | | | |
| uCosminexus TP1/Client/J | TP1/Client/J | TP1/Client | |
| uCosminexus TP1/Client/P | TP1/Client/P | | |
| uCosminexus TP1/Client/P(64) | | | |
| uCosminexus TP1/Client/W | TP1/Client/W | | |
| uCosminexus TP1/Client/W(64) | | | |
| uCosminexus TP1/Client for .NET Framework | TP1/Client for .NET Framework | Client .NET | |

| 製品名称 | 略称 | |
|---|---|---|
| uCosminexus TP1/Connector for .NET Framework | TP1/Connector for .NET Framework | Connector .NET |
| uCosminexus TP1/Extension for .NET Framework | TP1/Extension for .NET Framework | Extension.NET |
| uCosminexus TP1/Server Base Enterprise Option | TP1/EE | |
| uCosminexus TP1/Server Base Enterprise Option(64) | | |
| uCosminexus TP1/Extension 1 | TP1/Extension 1 | |
| uCosminexus TP1/Extension 1(64) | | |
| uCosminexus TP1/FS/Direct Access | TP1/FS/Direct Access | |
| uCosminexus TP1/FS/Direct Access(64) | | |
| uCosminexus TP1/FS/Table Access | TP1/FS/Table Access | |
| uCosminexus TP1/FS/Table Access(64) | | |
| uCosminexus TP1/High Availability | TP1/High Availability | |
| uCosminexus TP1/High Availability(64) | | |
| uCosminexus TP1/LiNK | TP1/LiNK | |
| uCosminexus TP1/Message Control | TP1/Message Control | |
| uCosminexus TP1/Message Control(64) | | |
| uCosminexus TP1/Message Control/Tester | TP1/Message Control/Tester | |
| uCosminexus TP1/Message Control - Extension 1 | TP1/Message Control - Extension 1 | |
| uCosminexus TP1/Message Control - Extension 1(64) | | |
| uCosminexus TP1/Message Queue | TP1/Message Queue | |
| uCosminexus TP1/Message Queue(64) | | |
| uCosminexus TP1/Message Queue Access | TP1/Message Queue Access | |
| uCosminexus TP1/Message Queue Access(64) | | |
| uCosminexus TP1/Messaging | TP1/Messaging | |
| uCosminexus TP1/Multi | TP1/Multi | |
| uCosminexus TP1/NET/High Availability | TP1/NET/High Availability | |
| uCosminexus TP1/NET/High Availability(64) | | |
| uCosminexus TP1/NET/Library | TP1/NET/Library | |
| uCosminexus TP1/NET/Library(64) | | |

| 製品名称 | 略称 | |
|---|---|---|
| uCosminexus TP1/NET/OSAS-NIF | TP1/NET/OSAS-NIF | |
| uCosminexus TP1/NET/OSI-TP | TP1/NET/OSI-TP | |
| uCosminexus TP1/NET/Secondary Logical Unit - TypeP2 | TP1/NET/SLU - TypeP2 | |
| uCosminexus TP1/NET/TCP/IP | TP1/NET/TCP/IP | |
| uCosminexus TP1/NET/TCP/IP(64) | | |
| uCosminexus TP1/NET/User Datagram Protocol | TP1/NET/UDP | |
| uCosminexus TP1/NET/User Datagram Protocol(64) | | |
| uCosminexus TP1/NET/User Agent | TP1/NET/User Agent | |
| uCosminexus TP1/NET/XMAP3 | TP1/NET/XMAP3 | |
| uCosminexus TP1/Offline Tester | TP1/Offline Tester | |
| uCosminexus TP1/Online Tester | TP1/Online Tester | |
| uCosminexus TP1/Resource Manager Monitor | TP1/Resource Manager Monitor | |
| uCosminexus TP1/Server Base | TP1/Server Base | |
| uCosminexus TP1/Server Base(64) | | |
| uCosminexus TP1/Shared Table Access | TP1/Shared Table Access | |
| uCosminexus TP1/Web | TP1/Web | |
| uCosminexus TP1/Web(64) | | |
| Windows 7 Enterprise (x86) | Windows 7 | Windows 7 |
| Windows 7 Professional (x86) | | |
| Windows 7 Ultimate (x86) | | |
| Windows 7 Enterprise (x64) | Windows 7 x64 Edition | |
| Windows 7 Professional (x64) | | |
| Windows 7 Ultimate (x64) | | |
| Windows 8 Enterprise (x86) | Windows 8 | Windows 8 |
| Windows 8 Pro (x86) | | |
| Windows 8 Enterprise (x64) | Windows 8 x64 Edition | |
| Windows 8 Pro (x64) | | |
| Windows 8.1 Enterprise (x86) | Windows 8.1 | Windows 8.1 |
| Windows 8.1 Pro (x86) | | |

| 製品名称 | 略称 | |
|---|---|---|
| Windows 8.1 Enterprise (x64) | Windows 8.1 x64 Edition | Windows 8.1 |
| Windows 8.1 Pro (x64) | | |
| Windows 10 Enterprise (x86) | Windows 10 | Windows 10 |
| Windows 10 Pro (x86) | | |
| Windows 10 Enterprise (x64) | Windows 10 x64 Edition | |
| Windows 10 Pro (x64) | | |
| Windows 11 Enterprise | Windows 11 | |
| Windows 11 Pro | | |
| Windows Server 2008 R2 Datacenter | Windows Server 2008 R2 | Windows Server 2008 |
| Windows Server 2008 R2 Enterprise | | |
| Windows Server 2008 R2 Standard | | |
| Windows Server 2012 Datacenter | Windows Server 2012 | |
| Windows Server 2012 Standard | | |
| Windows Server 2012 R2 Datacenter | Windows Server 2012 R2 | |
| Windows Server 2012 R2 Standard | | |
| Windows Server 2016 Datacenter | Windows Server 2016 | |
| Windows Server 2016 Standard | | |
| Windows Server 2019 Datacenter | Windows Server 2019 | |
| Windows Server 2019 Standard | | |
| Windows Server 2022 Datacenter | Windows Server 2022 | |
| Windows Server 2022 Standard | | |

- Windows 7，Windows 8，Windows 8.1，Windows 10，Windows 11，Windows Server 2008，Windows Server 2012，Windows Server 2012 R2，Windows Server 2016，Windows Server 2019，および Windows Server 2022 で機能差がない場合，Windows と表記しています。
- AIX，HP-UX，Linux，および Solaris を総称して UNIX と表記しています。

### (2)適用 OS による違いについて

Windows 版の製品をご使用になる場合，マニュアルの記述を次のように読み替えてください。

| 項目 | マニュアルの表記 | 読み替え |
|---|---|---|
| 環境変数の表記 | $aaaaaa<br>例　$DCDIR | %aaaaaa%<br>例　%DCDIR% |
| 複数のパス名を列挙するときの区切り文字 | : | ; |
| ディレクトリの区切り文字 | / | ¥ |
| 完全パス名 | ルートディレクトリから指定します。<br>例　/tmp | 先頭にドライブ文字を付加して，ルートディレクトリから指定します。<br>例　C:¥tmp |
| 実行形式ファイル名 | ファイル名だけを指定します。<br>例　mcfmngrd | ファイル名に拡張子を付加して指定します。<br>例　mcfmngrd.exe |
| make コマンド | make | nmake |

### (3)インストールディレクトリのパスの違いについて

このマニュアルでは，OpenTP1 のインストールディレクトリを「/BeTRAN」と表記しています。インストールディレクトリは OS によって異なります。ご利用の OS に応じて，次の表のとおりに読み替えてください。

| このマニュアルでの表記 | 適用 OS ごとの読み替え | | |
|---|---|---|---|
| | AIX，HP-UX または Solaris | Linux | Windows |
| /BeTRAN | /BeTRAN | /opt/OpenTP1 | OpenTP1 をインストールしたディレクトリ |

### (4)JIS コード配列のキーボードと ASCII コード配列のキーボードとの違いについて

JIS コード配列と ASCII コード配列では，次に示すコードで入力文字の違いがあります。このマニュアルの文字入力例（コーディング例）の表記は，**JIS コード配列**（日本語のキーボード）に従った文字に統一しています。

| コード | JIS コード配列 | ASCII コード配列 |
|---|---|---|
| $(5c)_{16}$ | '¥'（円記号） | '＼'（バックスラッシュ） |
| $(7e)_{16}$ | '‾'（オーバライン） | '~'（チルド） |

## ■ 略語一覧

このマニュアルで使用する英略語の一覧を次に示します。

| 英略語 | 英字での表記 |
|---|---|
| ACL | Access Control List |
| ANSI | American National Standards Institute |
| AP | Application Program |
| API | Application Programming Interface |
| C/S | Client/Server |
| CGI | Common Gateway Interface |
| CPU | Central Processing Unit |
| CRM | Communication Resource Manager |
| CUP | Client User Program |
| DAM | Direct Access Method |
| DAM FRC | DAM File Recovery |
| DB | Database |
| DBMS | Database Management System |
| DCE | Distributed Computing Environment |
| DHCP | Dynamic Host Configuration Protocol |
| DID | Distributed Identifier |
| DML | Data Manipulation Language |
| DNS | Domain Name System |
| DPM | ServerConductor/DeploymentManager |
| DTP | Distributed Transaction Processing |
| EX | Exclusive |
| FDDI | Fiber Distributed Data Interface |
| FEP | Front End Processor |
| FIFO | First-In-First-Out |
| FRC | File Recovery |
| GUI | Graphical User Interface |
| HA | High Availability |
| HTML | Hyper Text Markup Language |
| I/O | Input/Output |

| 英略語 | 英字での表記 |
|---|---|
| ID | Identifier |
| IDL | Interface Definition Language |
| IP | Internet Protocol |
| ISAM | Indexed Sequential Access Method |
| IST | Internode Shared Table |
| J2EE | Java 2 Enterprise Edition |
| JCA | J2EE Connector Architecture |
| JDBC | Java DataBase Connectivity |
| LAN | Local Area Network |
| MCF | Message Control Facility |
| MHP | Message Handling Program |
| MQ | Message Queue |
| MQA | Message Queue Access |
| MQI | Message Queue Interface |
| MQT | Message Queue Transfer |
| MSDTC | Microsoft Distributed Transaction Coordinator |
| NIF/OSI | Network Interface Feature/OSI |
| NIS | Network Information Service |
| OLTP | Online Transaction Processing |
| OS | Operating System |
| OSI | Open Systems Interconnection |
| OSI TP | Open Systems Interconnection Transaction Processing |
| PC | Personal Computer |
| PR | Protected Retrieve |
| PRF | Performance |
| RI | Recovery Information |
| RM | Resource Manager |
| RMM | Resource Manager Monitor |
| RPC | Remote Procedure Call |

| 英略語 | 英字での表記 |
|---|---|
| RTS | Real Time Statistic |
| SCSI | Small Computer Systems Interface |
| SNA | Systems Network Architecture |
| SPP | Service Providing Program |
| SRF | Server Recovery Journal File |
| SUP | Service Using Program |
| TAM | Table Access Method |
| TAM FRC | TAM File Recovery |
| TCO | Total Cost of Ownership |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TP | Transaction Processing |
| TRF | Transaction Recovery journal File |
| UAP | User Application Program |
| UID/GID | User Identifier/Group Identifier |
| UOC | User Own Coding |
| UTC | Coordinated Universal Time |
| VM | Virtual Machine |
| WAN | Wide Area Network |
| WS | Workstation |
| WWW | World Wide Web |
| XA | Extended Architecture |
| XAR | Extended Architecture Resource |

## ■ KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

# 目次

# 1 概要

この章では，OpenTP1 の特長，システム形態，ソフトウェア構成，およびサービスについて説明します。

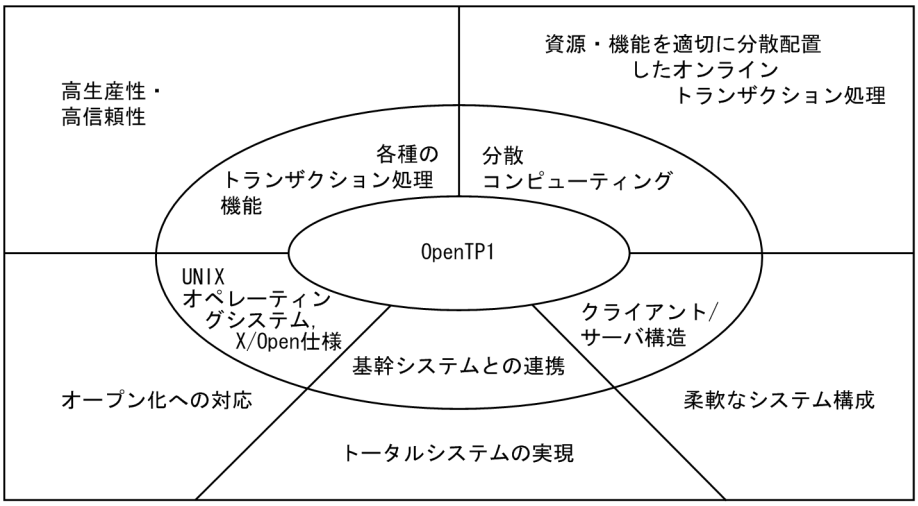
## 1.1　OpenTP1 の特長

分散トランザクション処理機能 OpenTP1 は，オープンシステム上で**オンライントランザクション処理**※（**OLTP**　Online Transaction Processing）をできるようにするソフトウェア（**TP モニタ**）です。

近年の情報処理システムの形態は，分散システム化，オープンシステム化の傾向にあります。さらに，業務の拡大に柔軟に対応できる拡張性が求められています。OpenTP1 を使用すると，現存する資産を生かしながら，分散システムやオープン環境に対応できる OLTP を実現できます。

OpenTP1 の特長を次の図に示します。

図 1-1　OpenTP1 の特長




注※

データ通信業務では，業務処理ごとの単位に区切って，それぞれの処理の結果を有効にするか無効にするかを明確に決める必要があります。有効にするか無効にするかどちらかに決定する処理の単位を**トランザクション**といいます。

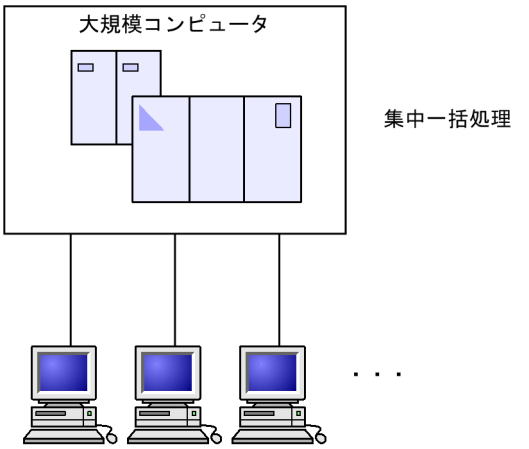
### 1.1.1　トランザクション処理ができる分散コンピューティング環境の実現

従来の OLTP は，大規模コンピュータで一括して処理する，集中型の OLTP でした。OpenTP1 を使うと，資源・機能を適切に分散配置した，**分散コンピューティング環境**での OLTP を実現できます。そのため，システム規模や業務量など，それぞれのシステム固有の状況に適した構成を柔軟に選択できます。

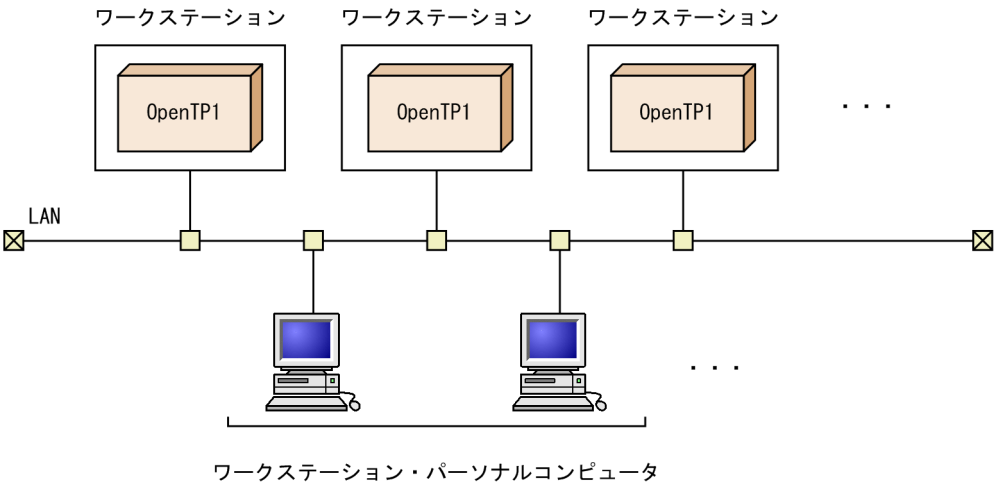
従来のシステムと，分散コンピューティング環境の OLTP の比較を次の図に示します。

図 1-2　従来のシステムと分散コンピューティング環境の OLTP の比較

●従来のシステム（大規模コンピュータでの一括処理）




●OpenTP1を使用した分散コンピューティング環境（全体で一つのオンラインシステム）

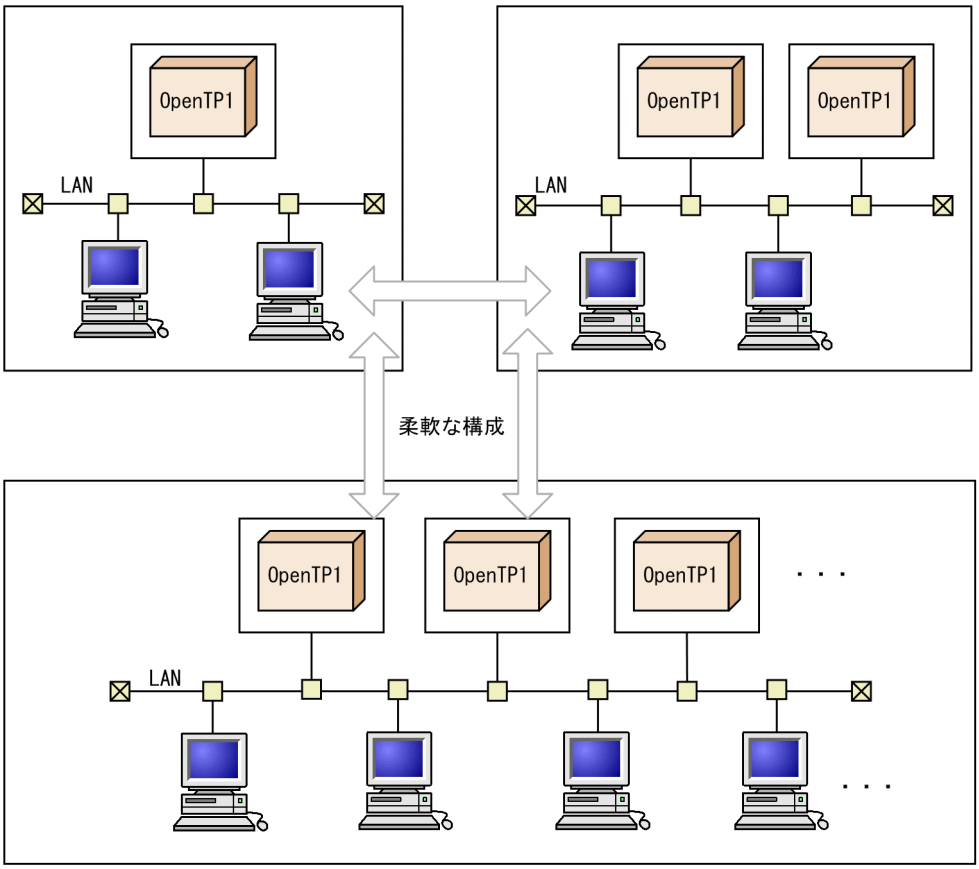



## 1.1.2　柔軟なシステム構成

OpenTP1 では，分散コンピューティング環境での OLTP を実現するために，**ユーザアプリケーションプログラム**（UAP）を含めたシステム構成が，**クライアント/サーバ構造**をしています。このため，プログラムやデータベースなどのソフトウェア資源を，ハードウェアの構成から独立させることができます。さらに，業務量の増加などでハードウェア構成を変更しても，ソフトウェア資源にはほとんど影響がありません。

柔軟なシステム構成を次の図に示します。

図 1-3　柔軟なシステム構成



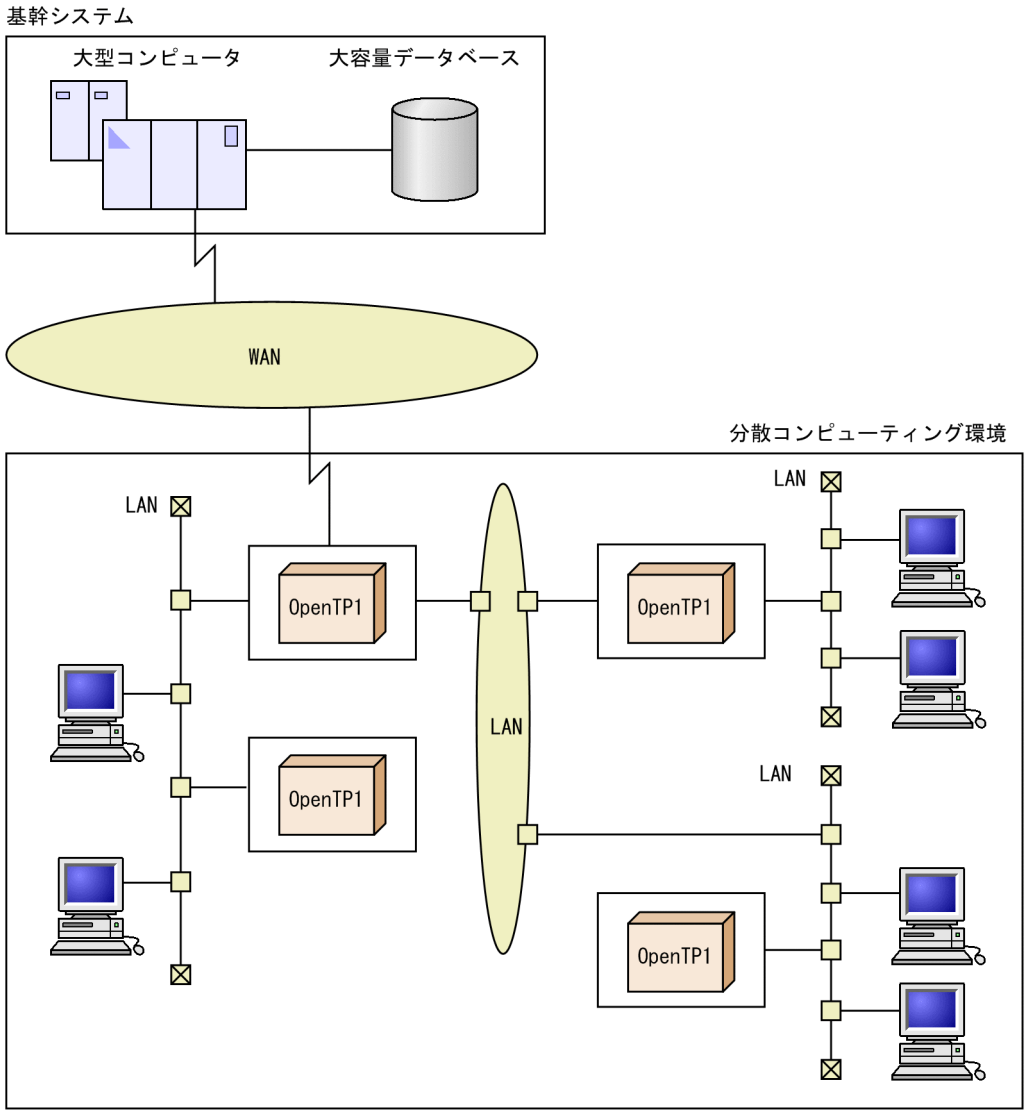

## 1.1.3　基幹システムと連携した大規模システムの実現

OpenTP1 を使用したシステムは，VOS3 システムなどの基幹システムと接続できます。このため，基幹システムの大容量データベース管理やネットワーク管理の業務に，OpenTP1 の分散コンピューティング環境を統合できます。

その結果，基幹システムからワークステーションまでの広範囲にわたる大規模システムを実現できます。

基幹システムと連携した大規模システムを次の図に示します。

図 1-4　基幹システムと連携した大規模システム




## 1.1.4　オープンシステム化推進の支援

OpenTP1 の構造は，オープンシステムの標準化団体である X/Open で規定している DTP モデルに準拠しています。そのため，OpenTP1 で構築した OLTP システムを使用すると，オープンシステムの利点を生かしたデータ通信システムを構築できます。主な特長を次に示します。

- OpenTP1 の UAP では，X/Open に準拠した API（TX インタフェースでのトランザクション制御，XATMI インタフェースでの通信，TxRPC インタフェースでの通信）を使用できます。
- OpenTP1 は，X/Open の XA インタフェースに準拠した，DBMS など各種リソースマネジャと接続できます。

### 1.1.5　システム構築の高生産性と高信頼性の実現

分散コンピューティング環境でのビジネスアプリケーションのトランザクション処理には，高い生産性と信頼性が必要です。OpenTP1 では，生産性と信頼性を上げるための各種機能を提供しています。これらの機能を使用することで，分散処理システムを構築する上でのユーザの負担を軽減させ，目的業務に応じたシステム設計ができます。

UAP をコーディングするときには，**C 言語**，**C++言語**，および **COBOL 言語**を使えます。C 言語でコーディングする場合には，**ANSI C 形式**，または ANSI 準拠前の **K&R 形式**のどちらでも使えます。COBOL 言語でコーディングする場合には，**COBOL85** または **COBOL2002** のどちらでも使えます。そのため，プログラマの技術環境に合った言語で開発できます。

COBOL 言語では**データ操作言語**（**DML**）も使用できるので，従来のホストコンピュータでの UAP 作成環境を，OpenTP1 で実現できます。

高い信頼性を要求される 24 時間運転業務に対応するため，OpenTP1 ではシステムの二重化を支援する機能（**系切り替え機能**）を備えています。系切り替え機能を使用すると，システムの障害による停止時間を最小限に抑えて，システム全体の稼働率を向上できます。

## 1.2　OpenTP1 のシステム形態

OpenTP1 を使用したシステム（OpenTP1 システム）は，次のような形態へ適用できます。

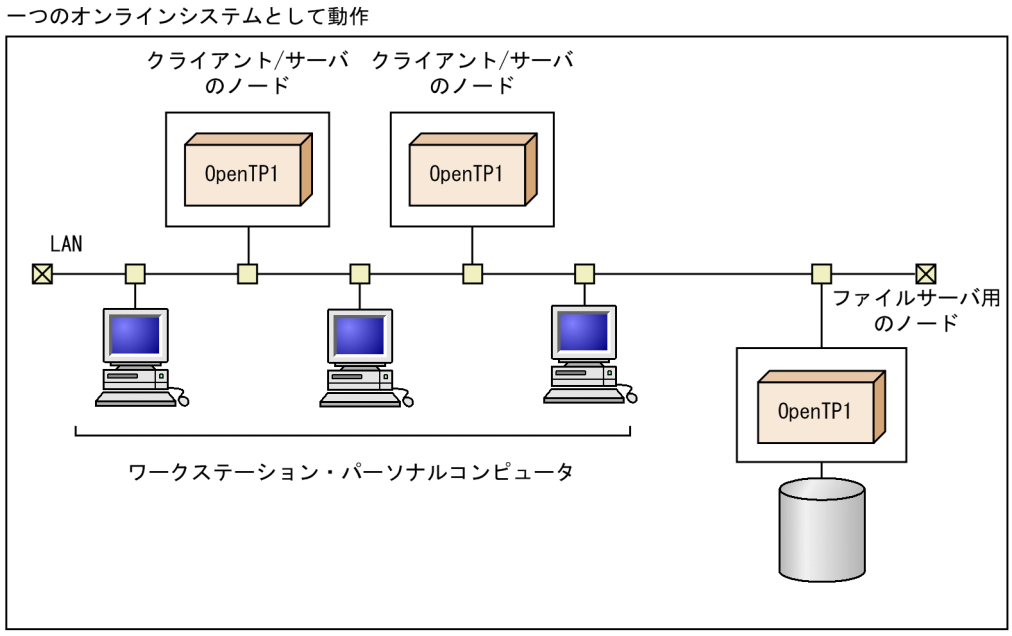
### 1.2.1　クライアント/サーバシステム（C/S システム）の形態

クライアント/サーバ方式で構築する，水平分散のシステム形態です。複数の OpenTP1 システムが，互いに協調して一つのオンラインシステムとして動作します。

業務を拡大するときは，必要なハードウェアを追加してシステムを拡張できます。ソフトウェア資源を煩雑に再編成する必要はありません。そのため，必要な規模に合わせたシステムを柔軟に構築できます。また，水平分散のトランザクション処理によるサーバ間の連携や，GUI のアプリケーションを使用したオンラインシステムにも対応しています。

クライアント/サーバシステム（C/S システム）の形態の例を次の図に示します。

図 1-5　クライアント/サーバシステム（C/S システム）の形態の例




### 1.2.2　フロントエンドプロセサ（FEP）の形態

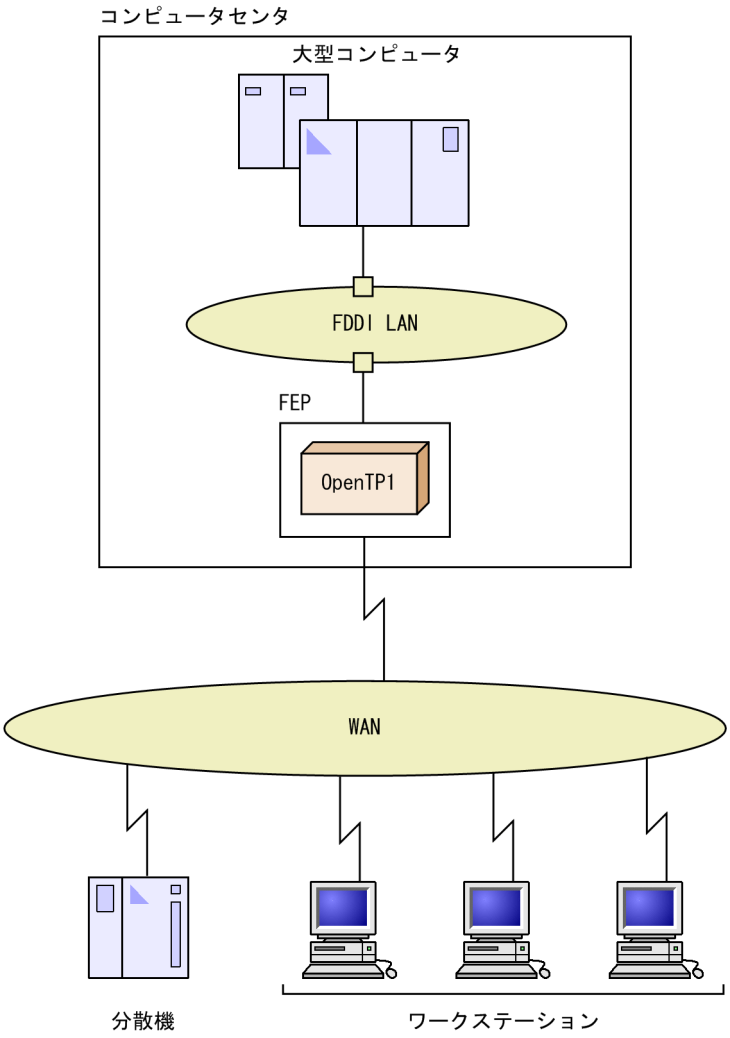
大型コンピュータのセンタ業務を分担する，フロントエンドプロセサ（FEP）に OpenTP1 を適用するシステム形態です。大型コンピュータと FEP は，高速 LAN（FDDI　LAN，またはチャネル接続）で接続させます。

FEP 上の OpenTP1 では，通信プロトコル処理や業務処理の一部を実行します。大型コンピュータの負荷を軽減したり，大型コンピュータで障害が起こった場合にセンタ業務を継続したりできます。

OpenTP1 システムを使うと，多種の通信プロトコルで，多回線接続ができます。そのため，広域網を介した各種通信に対応した FEP として使用できます。

フロントエンドプロセサ（FEP）の形態の例を次の図に示します。

図 1-6　フロントエンドプロセサ（FEP）の形態の例

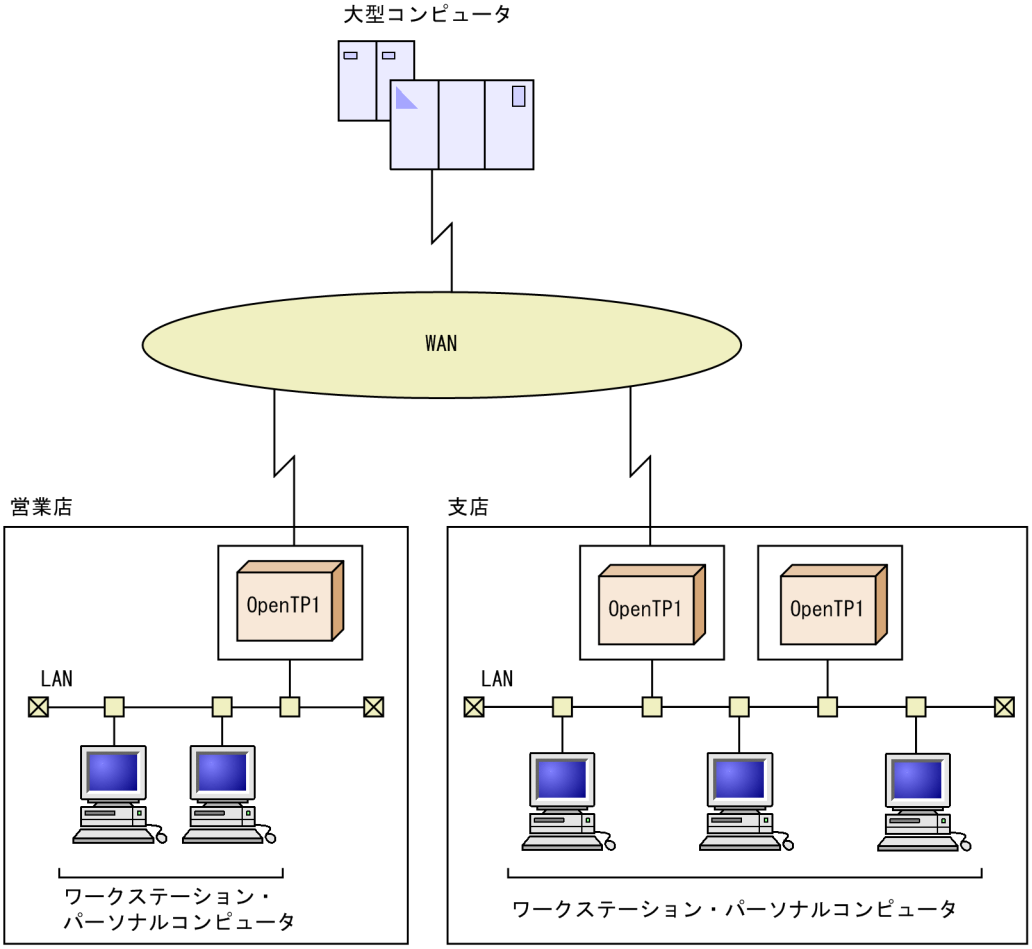



## 1.2.3　分散処理システムの形態

大型コンピュータと接続された営業店や支店（分散システム）に OpenTP1 を適用する，垂直分散のシステム形態です。大型コンピュータの業務処理を分散システムで分担して実行できます。また，大型コンピュータと分散システム内のワークステーション，パーソナルコンピュータとの中継ができます。

分散処理システムの形態の例を次の図に示します。

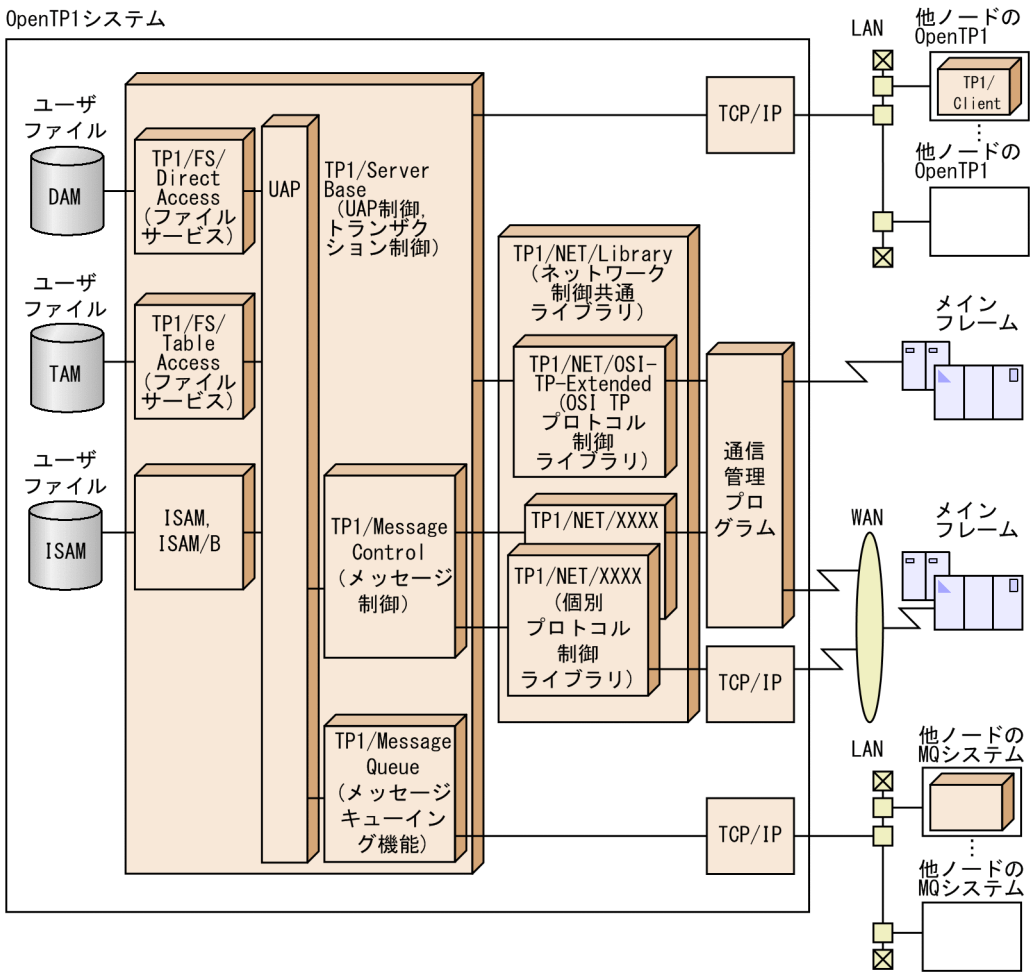
図 1-7　分散処理システムの形態の例



# 1.3　OpenTP1 のソフトウェア構成

OpenTP1 のソフトウェア構成の概要を次の図に示します。

図 1-8　OpenTP1 のソフトウェア構成の概要




## 1.3.1　OpenTP1 のソフトウェア製品

OpenTP1 のソフトウェア製品と，関連するソフトウェア製品を次に示します。

## (1)　OpenTP1 の基本部分を構成する製品

OpenTP1 の基本部分を構成する製品を次に示します。

- TP1/Server Base

  分散トランザクション処理の基本制御をする製品です。トランザクションを制御したり，UAP をスケジュールしたりします。

## (2) ユーザデータを管理する製品

ユーザデータを管理する OpenTP1 の製品を次に示します。

- TP1/FS/Direct Access

  OpenTP1 専用のユーザファイルを，直接編成ファイルとして使えるようにする製品です。TP1/FS/Direct Access を使うと，ユーザファイル（**DAM ファイル**）をトランザクション処理と同期して管理できます。

- TP1/FS/Table Access

  OpenTP1 専用のユーザファイルを，テーブルアクセス法で使えるようにする製品です。テーブルアクセス法とは，共用メモリ上にあるテーブルで回復可能なファイルとしてアクセスする方法です。
  TP1/FS/Table Access を使うと，ユーザファイル（**TAM ファイル**）をトランザクション処理と同期して管理できます。

- TP1/Shared Table Access

  共用メモリ上にあるテーブル（**IST テーブル**）を，複数の OpenTP1 システムで共用できるようにする製品です。テーブルの実体がどのノードにあるかを意識しないで，UAP からデータを参照，更新できるようにします。TP1/Shared Table Access の場合，ユーザファイルの実体はなく，メモリ上の IST テーブルにデータが格納されます。そのため，オンラインが異常終了した場合はデータを回復できません。

## (3) メッセージ制御機能を使うときに必要な製品

メッセージで通信できるようにする OpenTP1 の製品を次に示します。

- TP1/Message Control

  OpenTP1 システムがほかのシステムと，ネットワークを通して通信できるようにする製品です。TP1/Message Control を使った通信では，メッセージを使って通信します。

- TP1/NET/Library

  ネットワークを制御するのに必要な運用，構成管理，スケジュールなどの役割をするライブラリの製品です。

  TP1/Message Control と TP1/NET/Library は，メッセージ送受信機能を実現します。メッセージで通信する場合は，メッセージ送受信機能の製品と，通信プロトコル対応製品（**TP1/NET/××××**）が必要です。通信プロトコル対応製品については，「付録 A　TP1/Message Control で使える通信プロトコル対応製品」を参照してください。

- TP1/Messaging

  TP1/Message Control と TP1/NET/Library に相当する機能を持っています。また，TCP/IP 通信機能（TP1/NET/TCP/IP の機能）も持っています。

  TP1/Messaging については，マニュアル「TP1/Messaging 使用の手引」を参照してください。

## (4) メッセージキューイング機能を使うときに必要な製品

OpenTP1 のメッセージキューイング機能を使うときに必要な製品を次に示します。

- TP1/Message Queue

  OpenTP1 システムの UAP 間，およびネットワークを介した他システムの UAP との間で，非同期にメッセージを転送する場合に使用します。通信相手の UAP の状態に関係なくメッセージを転送でき，通信相手からの応答を待たないで処理できます。

## (5) OSI TP 通信を使うときに必要な製品

クライアント/サーバ形態の通信で，通信プロトコルに OSI TP を使う場合に必要となる製品を次に示します。

- TP1/NET/OSI-TP-Extended

  OSI TP 通信のプロトコル管理をする製品です。TP1/NET/OSI-TP-Extended に関する定義およびセットアップ手順については，マニュアル「OpenTP1 プロトコル TP1/NET/OSI-TP-Extended 編」を参照してください。

## (6) 系切り替え機能を使うときに必要な製品

OpenTP1 システムを二重化する機能（**系切り替え機能**）を使えるようにする製品を次に示します。

- TP1/High Availability

  OpenTP1 を系切り替え機能を使ったシステム形態に使うときに必要な製品です。系切り替え機能では，HA モニタが前提となります。

- TP1/NET/High Availability

  系切り替え機能を使った OpenTP1 のシステム形態で，メッセージ制御機能を使うときに必要な製品です。TP1/NET/High Availability には，TP1/High Availability と HA モニタが前提となります。また，二重化しているシステムとメッセージ送受信をする場合に，**コネクション切り替え機能**を使うときにも必要となります。コネクション切り替え機能については，マニュアル「OpenTP1 プロトコル TP1/NET/TCP/IP 編」を参照してください。

## (7) OpenTP1 の拡張機能を使うときに必要な製品

OpenTP1 の性能の確保，運用を容易にするための拡張機能を使うときに必要な製品を次に示します。

- TP1/Extension1

  次の拡張機能を使用する場合に必要な製品です。

  - 性能検証用トレース

    負荷テストや本番環境で，サービス呼び出しのどこに時間が掛かっているのかを詳細に分析することができます。

  - マルチスケジューラ機能

多数のクライアントから長大データを受け取る場合に複数のスケジューラを起動できるようにし，スケジューラのボトルネックを解消できます。

- ノード直接指定 RPC

  あて先ノードの IP アドレスを直接指定して RPC を発行できます。これによって，ネームサーバのドメイン外への RPC，およびネットワーク経路が二重化されている場合の経路の選択ができるようになります。

- オンライン中サーバプロセス多重度変更コマンド

  オンライン中のサービスの処理プロセス数をコマンドで変更し，処理能力の拡張や縮退ができます。

- プロセスごとのサービス処理時刻表示コマンド

  コマンドで，プロセスごとにサービスを受付けた時刻と終了した時刻を表示することができます。

- ユーザサーバのプロセスのリフレッシュ機能

  オンラインを停止することなく，ユーザサーバのロードモジュールを入れ替えられます。

- TP1/Message Control - Extension1

  メッセージ制御機能に関する，次の拡張機能を使用する場合に必要な製品です。

  - MCF 構成変更再開始機能

    前回のオンライン停止時に，入出力キュー上に残っていた未処理メッセージを次回オンラインに引き継ぎつつ，OpenTP1 ファイルシステムやキューグループの構成変更，および論理端末（コネクション）やアプリケーションの構成を変更できます。

## (8) OpenTP1 の UAP をテストするときに必要な製品

OpenTP1 システムで使う UAP をテストするときに必要な製品を次に示します。

- TP1/Offline Tester

  **オフラインテスタ機能**を使うときに必要な製品です。システムで OpenTP1 が稼働していない環境でも，UAP をテストできます。

- TP1/Online Tester

  **オンラインテスタ機能**を使うときに必要な製品です。OpenTP1 が稼働している環境で，UAP をテストできます。

- TP1/Message Control/Tester

  **MCF オンラインテスタ機能**を使うときに必要な製品です。メッセージ制御機能を使う UAP をテストできます。

## (9) OpenTP1 のクライアント用マシンに必要な製品

OpenTP1 のサーバへサービスを要求する，クライアント用マシンに必要な製品を次に示します。

- TP1/Client/W，TP1/Client/P

WS や PC をクライアント用マシンとして，OpenTP1 のサーバへサービスを要求できるようにする製品です。

- TP1/Client/J

  Java アプレット，Java アプリケーション，または Java サーブレットから，OpenTP1 のサーバへサービスを要求できるようにする製品です。

TP1/Client/W，TP1/Client/P，および TP1/Client/J については，「2.2.2　OpenTP1 クライアント機能の概要」を参照してください。

## (10)　分散アプリケーションサーバ機能で必要な製品

- TP1/LiNK

  分散システム環境で，UAP のスケジュールなどの基本制御をする製品です。TP1/Server Base に比べ，小規模な部門に適用できます。製品の組み込みからセットアップまでを，簡単に操作できるようにしています。

## (11)　クラスタ/並列システムで OpenTP1 を使うときに必要な製品

OpenTP1 をクラスタ/並列システムに使うときに必要な製品を次に示します。

- TP1/Multi

  クラスタ/並列システム環境で OpenTP1 を使う場合に必要な製品です。一つのノードから複数のノードを操作できる**マルチノード機能**を使えるようにします。

## (12)　OpenTP1 提供以外のリソースマネジャを制御するときに必要な製品

OpenTP1 提供以外のリソースマネジャを制御できるようにする必要な製品を次に示します。

- TP1/Resource Manager Monitor

  OpenTP1 提供以外のリソースマネジャを制御できるようにする製品です。TP1/Resource Manager Monitor を使うと，OpenTP1 で，リソースマネジャの開始と終了を制御できます。

## (13)　OpenTP1 システムに関連する製品

OpenTP1 に関連する製品について説明します。

- ISAM，ISAM/B（HP-UX だけ）

  ユーザファイルを X/Open の ISAM モデルに準拠した索引順編成ファイルとして使えるようにする製品です。ユーザファイル（ISAM ファイル）をトランザクション処理と同期して管理する場合は，ISAM に加えて，ISAM/B が必要になります。

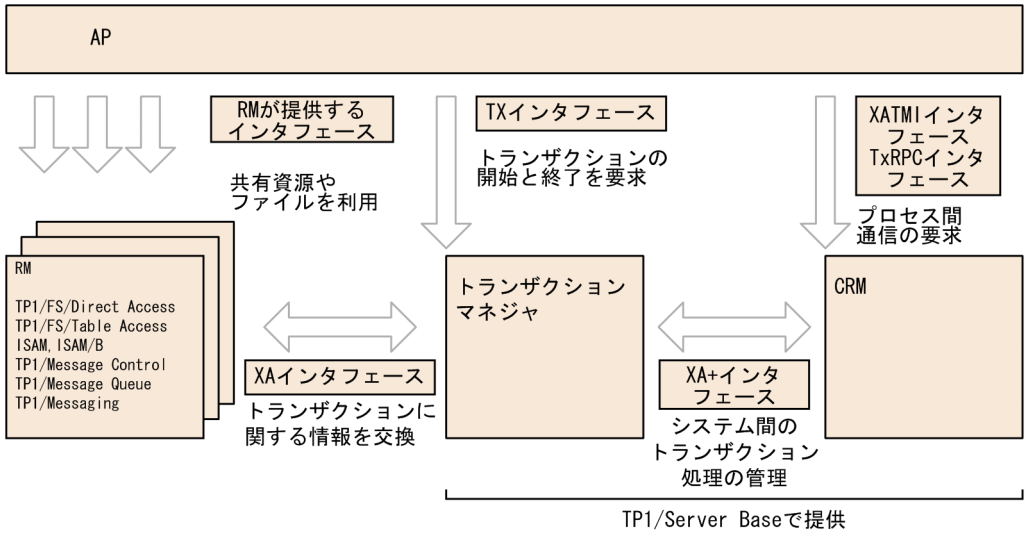
- HA モニタ

  OpenTP1 システムの信頼性を上げるために，システムを二重化するときに必要になる製品です。HA モニタを使った OpenTP1 システムについては，「6.1　系切り替え機能」を参照してください。

## 1.3.2　X/Open の DTP モデルと OpenTP1 システムの関係

OpenTP1 を使った分散トランザクション処理システムは，X/Open の DTP モデルに準拠しています。X/Open の DTP モデルの構成を次の図に示します。

図 1-9　X/Open の DTP モデルの構成




### (1)　DTP モデルの構成

DTP モデルは，次に示す要素から構成されます。

- AP（アプリケーションプログラム）

  システム利用者が高級言語で作成するアプリケーションプログラムです。OpenTP1 の UAP のことです。
- トランザクションマネジャ

  システムのトランザクションを管理して，資源の更新情報を基に一貫性を保証します。
- RM（リソースマネジャ）

  ユーザデータなどシステムの資源を管理します。
- CRM（コミュニケーションリソースマネジャ）

  システム間の通信に関する資源を管理します。

### (2)　各要素間のインタフェース

DTP モデルを構成する各要素は，次に示すインタフェースで連携できます。

- TX インタフェース

  AP からトランザクションマネジャへトランザクションの開始と終了を指示します。
- XATMI インタフェース，TxRPC インタフェース

  AP から CRM へ通信を指示します。

- XA インタフェース
  トランザクションマネジャと RM で資源の更新情報に基づいて同期を取ります。
- XA+インタフェース
  CRM を使ってほかのシステムと通信する場合に，トランザクションマネジャで管理するトランザクションをほかのシステムの処理まで拡張します。
- RM が提供するインタフェース
  AP から RM へ資源の更新を指示します。RM への API には，SQL などがあります。

## (3) OpenTP1 が提供するリソースマネジャ

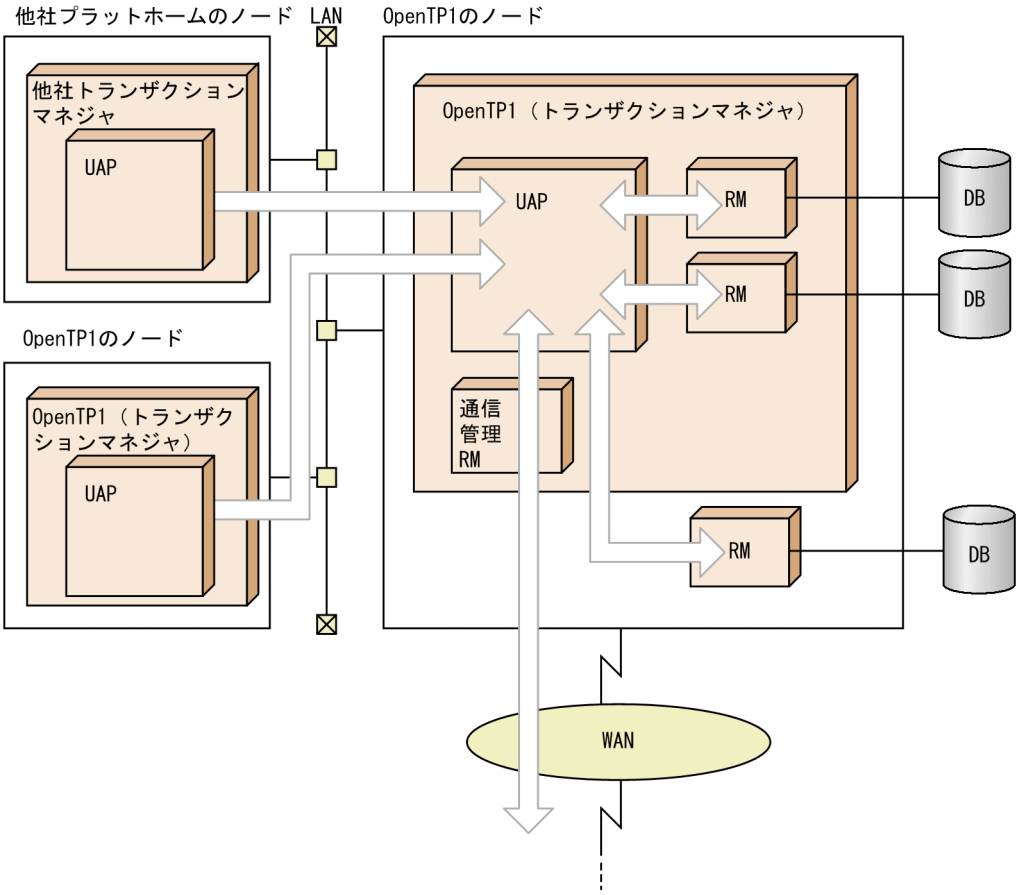
OpenTP1 で使える RM を次に示します。

- ファイルなど，ユーザ資源を管理する RM
  TP1/FS/Direct Access，TP1/FS/Table Access，ISAM（ISAM/B），および X/Open の仕様に準拠した RM
- メッセージを使った通信を管理する RM
  TP1/Message Control，TP1/NET/Library，TP1/Message Queue，TP1/Messaging

OpenTP1 で OSI TP 通信をする場合は，XA+インタフェースによって，他システムへトランザクション処理を拡張できます。

X/Open の DTP モデルと OpenTP1 で構築するシステムの関係を次の図に示します。

図 1-10　X/Open の DTP モデルと OpenTP1 で構築するシステムの関係



# 1.4 OpenTP1 システムのサービス

## 1.4.1 OpenTP1 のサービスの種類

クライアント/サーバ構造に基づいたオンライントランザクション処理システムを構築するため，OpenTP1 システムでは，各種のサービス機能を使えます。ほかのプロセスから呼び出して利用できるプログラムの単位を**サービス**，サービスを提供するプロセスを**サーバ**といいます。

OpenTP1 のサービスを**システムサービス**といいます。これに対して，ユーザで作成した UAP によって，業務処理としてサービスを提供するサーバを**ユーザサーバ**といいます。

## 1.4.2 OpenTP1 のシステムサービス

OpenTP1 で提供するシステムサービスを次に示します。（　）は，サービスを識別する名称です。

### (1) OpenTP1 システムの前提になるシステムサービス

- トランザクションサービス（trn）
  トランザクション処理の開始，終了，およびトランザクションの同期点取得による資源の更新処理を管理します。
- ネームサービス（nam）
  UAP 間の通信（リモートプロシジャコール）を実現するために，LAN のネットワークアドレスとユーザサーバの対応を管理します。
- スケジュールサービス（scd）
  RPC のスケジュールや，メッセージ制御機能のスケジュールを管理します。
- プロセスサービス（prc）
  ノード内のプロセスを管理します。
- ステータスサービス（sts）
  主に，トランザクションとは関係ない履歴情報を，専用のファイルに格納して，管理します。
- インタバルサービス（itv）
  各システムサービスでの内部処理の監視時間を管理します。
- ジャーナルサービス（jnl）
  トランザクション処理に関係する履歴情報（ジャーナル）を，システムジャーナルファイルに格納して，管理します。
- チェックポイントダンプサービス（cpd）
  ジャーナルの読み込み時間を短縮するため，システムジャーナルファイルにチェックポイントを設定して管理します。

- ロックサービス（lck）
  トランザクション処理に伴う各種の排他制御を管理します。
- ログサービス（log）
  OpenTP1 システムからオペレータに内部情報を知らせるためのメッセージログを管理します。
- タイマサービス（tim）
  トランザクション経過時間の監視など，各種の時間監視機能を提供します。

## (2) ユーザデータを管理するシステムサービス

- DAM サービス（dam）
  業務処理に使用する，OpenTP1 専用のユーザファイル（DAM ファイル）を管理します。
- TAM サービス（tam）
  業務処理に使用する，OpenTP1 専用のユーザファイル（TAM ファイル）を管理します。
- IST サービス（ist）
  複数の OpenTP1 システムで共用する，メモリ上のテーブル（IST テーブル）を管理します。

## (3) メッセージ送受信関連のシステムサービス

- MCF サービス（MCF マネジャサービス，MCF 通信サービス，アプリケーション起動サービス）
  メッセージを使用した通信をする場合に，各種のメッセージ制御機能を実現します。
- メッセージキューサービス（que）
  メッセージ制御機能を使用する場合に，メッセージを格納する待ち行列（入出力キュー）を管理します。
- マッピングサービス（map）
  GUI 対応の拡張プレゼンテーションシステムを OpenTP1 で使う場合に，その機能を支援します。
  これらのサービスは，メッセージ制御用の RM である TP1/Message Control が前提となります。

## (4) メッセージキューイング関連のシステムサービス

- MQA サービス（mqa）
  UAP 間で非同期にメッセージを転送する場合に，メッセージキューイング機能を実現します。メッセージキューイング制御用の RM である TP1/Message Queue が前提となります。

## (5) OSI TP を使ったクライアント/サーバ形態関連のシステムサービス

- XATMI 通信サービス（xat）
  OSI TP を使ったクライアント/サーバ形態の通信を管理します。
  この通信をする場合には，TP1/NET/OSI-TP-Extended が前提となります。

## (6) リソースマネジャを管理するシステムサービス

- リソースマネジャモニタサービス（rmm）
  リソースマネジャの開始や終了を，OpenTP1 で制御します。

## (7) OpenTP1 クライアント機能（TP1/Client）で使うプロセスを管理するシステムサービス

- クライアントサービス（clt）
  TP1/Client の AP（CUP）からトランザクションを開始する場合に，TP1/Server Base に必要なプロセスを管理します。

## (8) オンラインテスタ（TP1/Online Tester）を管理するシステムサービス

- テスタサービス（uto）
  オンラインテスタの実行環境を管理します。

## (9) 補助的なシステムサービス

- トランザクションジャーナルサービス（tjl）
  長時間掛かるトランザクションの処理の履歴情報（ジャーナル）を，専用のジャーナル（トランザクションリカバリジャーナル）として格納して，管理します。
- リアルタイム統計情報サービス（rts）
  OpenTP1 システムの稼働状況をリアルタイムに把握するための，リアルタイム統計情報を管理します。

## (10) マルチノード環境で，アーカイブジャーナルファイルを管理するシステムサービス

- グローバルアーカイブジャーナルサービス（jnl）
  クラスタ/並列システムで，各 OpenTP1 ノードのシステムジャーナルファイルをアーカイブして管理します。

## (11) X/Open 準拠の索引順編成ファイルを管理するシステムサービス

- ISAM サービス（ism）
  業務処理に使う，X/Open に準拠した索引順編成のユーザファイル（ISAM ファイル）を管理します。

## (12) OpenTP1 を監視するシステムサービス

- OpenTP1 監視サービス（dcmon）
  OpenTP1 システムの正常稼働を維持するために，プロセスサービスの稼働状態を監視します。

## 1.4.3 OpenTP1 のシステム定義

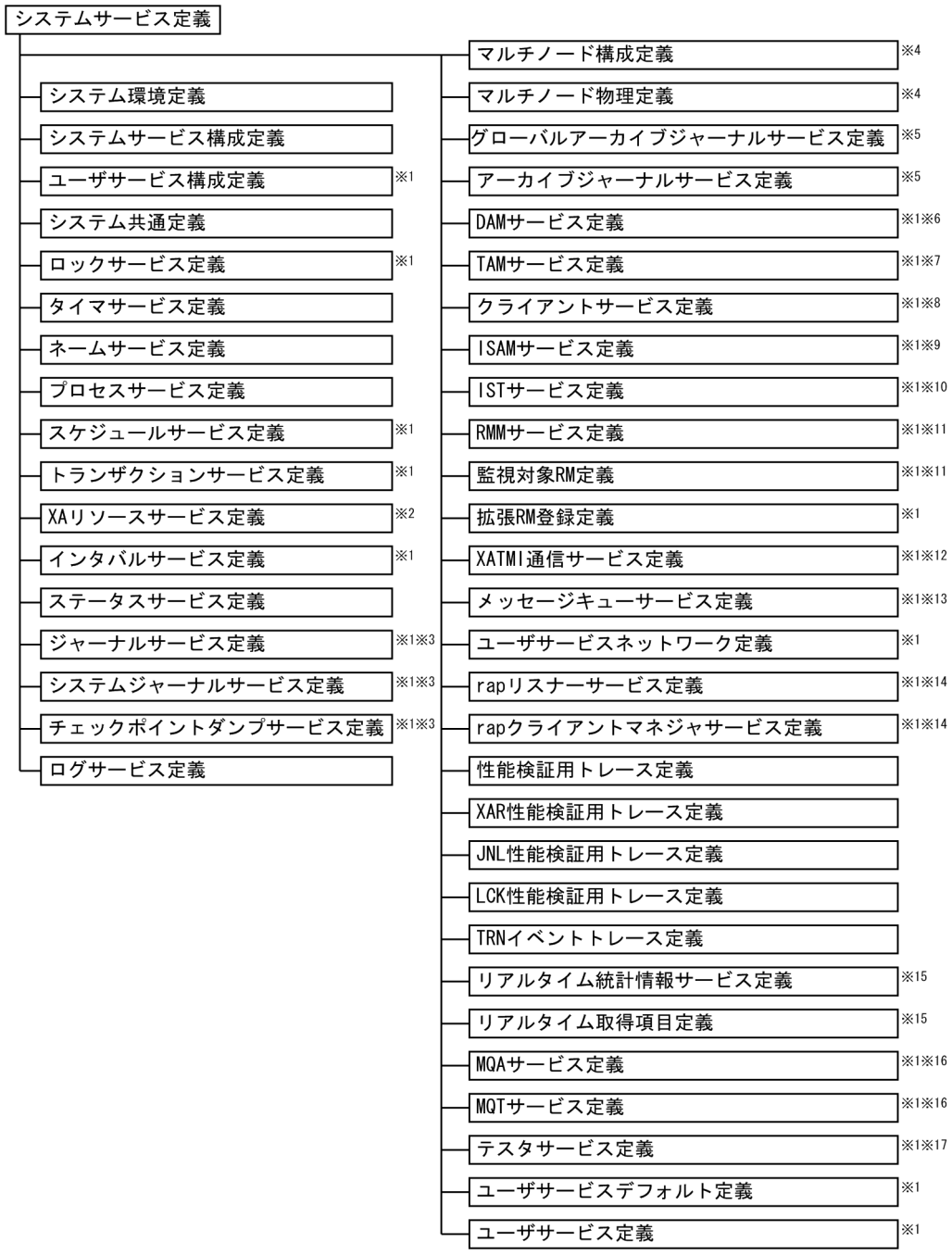
OpenTP1 の動作環境や OpenTP1 が使う各種の資源は，**システム定義**に指定します。OpenTP1 のシステム定義は，次のように分けられます。

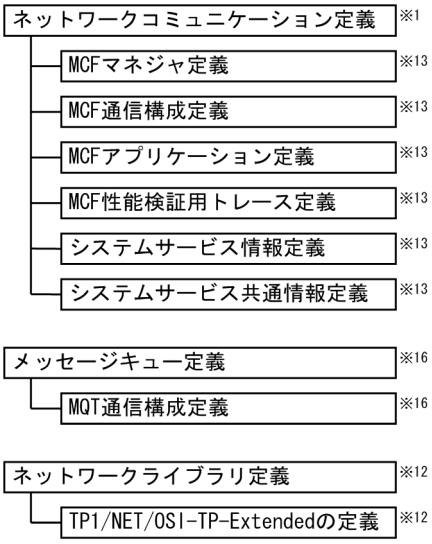
- **システムサービス定義**
  OpenTP1 の基本機能（TP1/Server Base），各種ファイルサービス，マルチノード機能，およびメッセージキューイング機能に関連する項目について定義します。
- **ネットワークコミュニケーション定義**
  メッセージ送受信機能（TP1/Message Control）を使う場合に定義します。
- **メッセージキュー定義**
  メッセージキューイング機能（TP1/Message Queue）を使う場合に定義します。
- **ネットワークライブラリ定義**
  TP1/NET/OSI-TP-Extended を使った OSI TP 通信をする場合に定義します。

定義を作成するときは，テキストエディタでファイルを作成します。システム定義については，マニュアル「OpenTP1 システム定義」を参照してください。

OpenTP1 のシステム定義の体系を，次の図に示します。

## 図 1-11　OpenTP1 システム定義の体系






注※1　グローバルアーカイブジャーナルサービスを使う OpenTP1 ノードでは，定義しません。

注※2　XA リソースサービス機能を使う場合に定義します。

注※3　ジャーナルファイルレス機能を使用する OpenTP1 ノードでは，定義しません。

注※4　TP1/Multi を使う場合に定義します。

注※5　グローバルアーカイブジャーナルサービスを使う場合に定義します。

注※6　TP1/FS/Direct Access を使う場合に定義します。

注※7　TP1/FS/Table Access を使う場合に定義します。

注※8　TP1/Client/W，または TP1/Client/P を使う場合に定義します。

注※9　ISAM ファイルをトランザクション処理する機能（ISAM/B）を使う場合に定義します。定義の内容については，マニュアル「索引順編成ファイル管理 ISAM」を参照してください。

注※10　TP1/Shared Table Access を使う場合に定義します。

注※11　TP1/Resource Manager Monitor を使う場合に定義します。

注※12　TP1/NET/OSI-TP-Extended を使った OSI TP 通信をする場合に定義します。定義の内容については，マニュアル「OpenTP1 プロトコル TP1/NET/OSI-TP-Extended 編」を参照してください。

注※13 メッセージ送受信機能（TP1/Message Control）を使う場合に定義します。

注※14　リモート API 機能を使う場合に定義します。

注※15　リアルタイム統計情報サービスを使う場合に定義します。

注※16　メッセージキューイング機能（TP1/Message Queue）を使う場合に定義します。定義の内容については，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

注※17　TP1/Online Tester を使う場合に定義します。定義の内容については，マニュアル「OpenTP1 テスタ・UAP トレース使用の手引」を参照してください。

# 2 業務処理の形態

この章では，OpenTP1 でできる業務処理の形態，関連する製品，およびアプリケーションプログラムについて説明します。

## 2.1 OpenTP1 の通信形態

分散コンピューティング環境の業務をするため，OpenTP1 では，次に示す形態で通信できます。

- **クライアント/サーバ形態の通信**

  TCP/IP プロトコルで接続した LAN を介して OpenTP1 システム間で通信する形態です。OpenTP1 の UAP から**リモートプロシジャコール**を発行して通信します。また，X/Open に準拠した API で通信したり，OpenTP1 以外のオープンシステムと通信したりすることもできます。

  また TP1/Web を使えば，インターネット上の WWW サーバと WWW ブラウザの情報のやり取りを OpenTP1 のリモートプロシジャコールに変換して，OpenTP1 の UAP を実行できます。この機能によって WWW ブラウザを端末とする業務システムを構築できます。

- **メッセージ送受信形態の通信**

  ネットワークを介して，他システムとメッセージを使って通信する形態です。メッセージ送受信形態では，各種の通信プロトコルに準拠した手順で通信します。この形態の通信では，メッセージ送受信機能の製品（TP1/Message Control，TP1/NET/Library），および通信プロトコル対応製品が必要です。

  このマニュアルでは，OpenTP1 のメッセージ送受信機能の製品と通信プロトコル対応製品を総称した機能を，以降 **MCF**（Message Control Facility）または **MCF サービス**と表記します。

- **メッセージキューイング機能を使った形態の通信**

  TCP/IP プロトコルで接続したネットワークを介して，キューマネジャを使って通信する形態です。この形態で通信する OpenTP1 システムには，キューマネジャとして TP1/Message Queue が必要です。

上記のほかにも，複数の OpenTP1 を使うシステム形態（系切り替え形態，クラスタ/並列システム形態）があります。詳細は，「6. 複数の OpenTP1 を使用する場合の機能」を参照してください。

## 2.2　クライアント/サーバ形態の処理概要

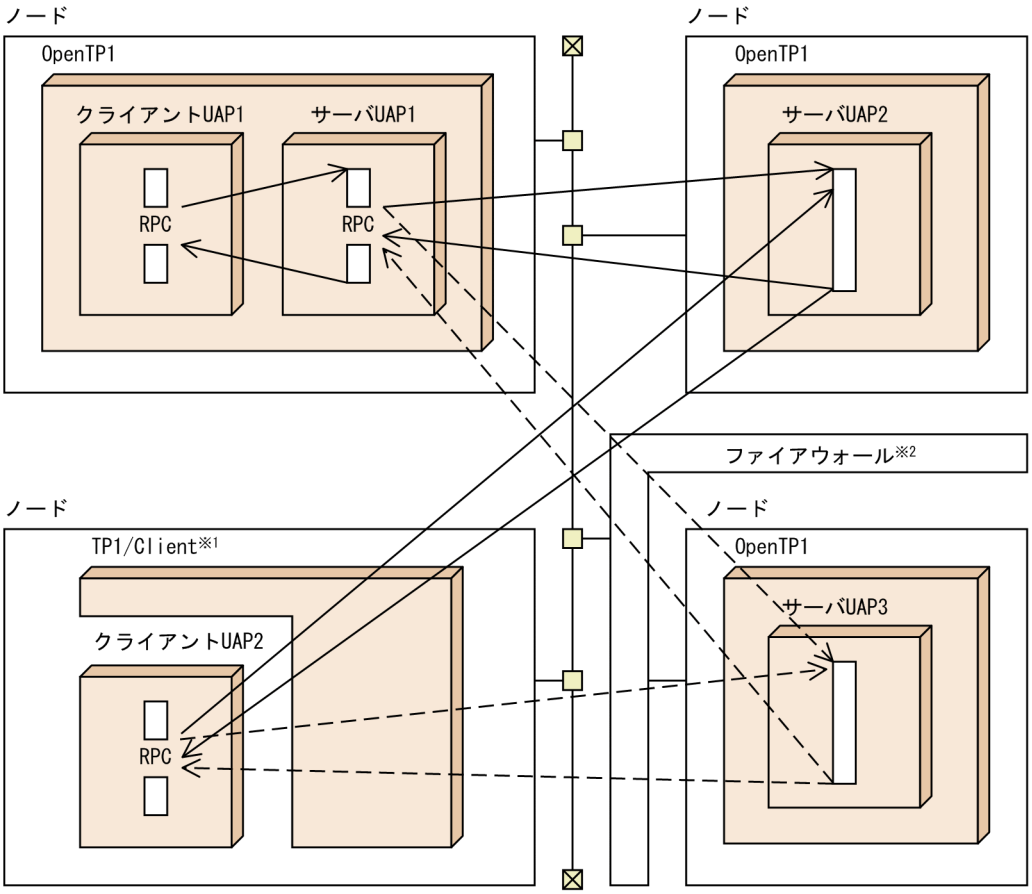
クライアント/サーバ形態の通信とは，使用する通信プロトコル手順を意識しないで，ほかの UAP プロセスと通信する方法です。クライアント/サーバ形態の通信の概要について説明します。

### 2.2.1　リモートプロシジャコールを使った通信

OpenTP1 の UAP では，ほかの UAP と**リモートプロシジャコール（RPC）**を使用して通信できます。リモートプロシジャコールとは，UAP プロセスからほかの UAP プロセスへサービスを要求し，サービスを要求された UAP プロセスは要求元の UAP へ処理結果を返す通信です。

クライアント/サーバ形態の通信の処理概要を次の図に示します。

図 2-1　クライアント/サーバ形態の通信の処理概要




注※1

TP1/Client は，OpenTP1 のサーバへサービスを要求する，クライアント専用の製品です。

TP1/Client については，「2.2.2　OpenTP1 クライアント機能の概要」および「3.5　OpenTP1 クライアント機能（TP1/Client）」を参照してください。

注※2

OpenTP1 独自のインタフェースを使うと，ファイアウォールの内側にあるサーバ UAP にもリモートプロシジャコールを使って通信できます。ただし，ファイアウォールを通過した場合，通過先のサーバ UAP をトランザクション処理に含めることはできません。ファイアウォールを通過する場合については，「3.7.1　リモート API 機能の使用例」を参照してください。

## (1)　クライアント UAP とサーバ UAP

サービスを要求する UAP とサービスを提供する UAP は，クライアントとサーバの関係になります。サービスを要求する側の UAP を**クライアント UAP**，サービスを提供する側の UAP を**サーバ UAP** といいます。

UAP については，「2.6　OpenTP1 のアプリケーションプログラム」を参照してください。

## (2)　トランザクショナル RPC

リモートプロシジャコールでは，サービス要求をしてから結果が返ってくるまでの処理をトランザクションとするかどうかを選べます。トランザクションとする RPC を，**トランザクショナル RPC** といいます。トランザクショナル RPC の場合は，複数のノードにわたって処理している RPC でも，一つのトランザクションとして処理できます。

## (3)　OpenTP1 で使えるリモートプロシジャコール

OpenTP1 でクライアント/サーバ形態の通信をする場合は，次に示すインタフェースを使えます。

### (a)　OpenTP1 独自のインタフェース

OpenTP1 独自のライブラリ関数を使ってサービスを要求する通信方法です。

### (b)　X/Open に準拠したインタフェース

X/Open に準拠した通信方法を次に示します。

- XATMI インタフェース
  X/Open で規定するライブラリ関数を使ってサービスを要求する通信方法です。
- TxRPC インタフェース
  X/Open で規定した手順で作成した UAP の関数を直接呼び出して通信する方法です。

これらのほかにも，OpenTP1 以外のオープンシステムと通信することもできます。クライアント/サーバ形態の通信については，「3.2　クライアント/サーバ形態の通信」を参照してください。

## (4)　クライアント/サーバ形態の通信プロトコル

OpenTP1 のクライアント/サーバ形態の通信プロトコルには，TCP/IP と OSI TP を使えます。どちらを使う場合でも，UAP の処理で通信プロトコルを意識する必要はありません。

通信プロトコルに OSI TP を使う場合には，通信プロトコルを管理する製品として TP1/NET/OSI-TP-Extended が必要です。OSI TP を使ったクライアント/サーバ形態の通信については，「3.6　OSI TP を使ったクライアント/サーバ形態の通信」を参照してください。

### (5)　TCP_NODELAY

TCP_NODELAY は，送信済みデータの応答待ちの状態でも遅延させることなくデータを送信できるようにする機能です※。この機能では，OpenTP1 がノード間で使用する通信ソケット（INET ドメイン）に，TCP_NODELAY オプションを使用します。

データ送信の際に TCP_NODELAY オプションを使用するかどうかは，スケジュールサービス定義，ユーザサービス定義，またはユーザサービスデフォルト定義の ipc_tcpnodelay オペランドで指定します。なお，TCP_NODELAY オプションを使用すると，INET ドメイン通信時の送信効率が低下し，ネットワークの負荷が大きくなる場合があります。TCP_NODELAY オプションを使用する場合は，ipc_sendbuf_size オペランド，ipc_recvbuf_size オペランド，ネットワークの帯域などを考慮し，この機能の必要性を十分に検討してください。

注※

TCP/IP の Nagle アルゴリズムを無効にすることで，この機能を実現させています。

## 2.2.2　OpenTP1 クライアント機能の概要

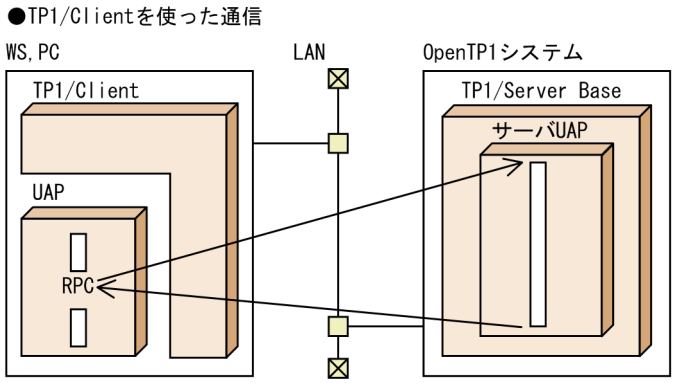
OpenTP1 で構築したサーバシステムへ，クライアント用の WS または PC からサービスを要求できます。この通信をするための製品を **OpenTP1 クライアント機能**（TP1/Client）といいます。

OpenTP1 クライアント機能は，UAP を作成して，OpenTP1 のサーバ UAP と通信できます。クライアント側に期待する性能，接続数，および OS に応じて，クライアント側の製品を選択できます。OpenTP1 クライアント側の製品には，次の製品があります。

- **TP1/Client/W**（WS 向け）
- **TP1/Client/P**（PC 向け）
- **TP1/Client/J**（Java 動作環境向け）

OpenTP1 クライアント機能でできる通信の概要を次の図に示します。

## 図 2-2　OpenTP1 クライアント機能でできる通信の概要




TP1/Client の機能の詳細については，「3.5　OpenTP1 クライアント機能（TP1/Client)」を参照してください。

## 2.3　メッセージ送受信形態の処理概要

**メッセージ送受信**とは，OpenTP1 以外のシステムと通信規約（通信プロトコル）に定められた手順でメッセージをやり取りする通信方法です。メッセージ送受信の概要について説明します。

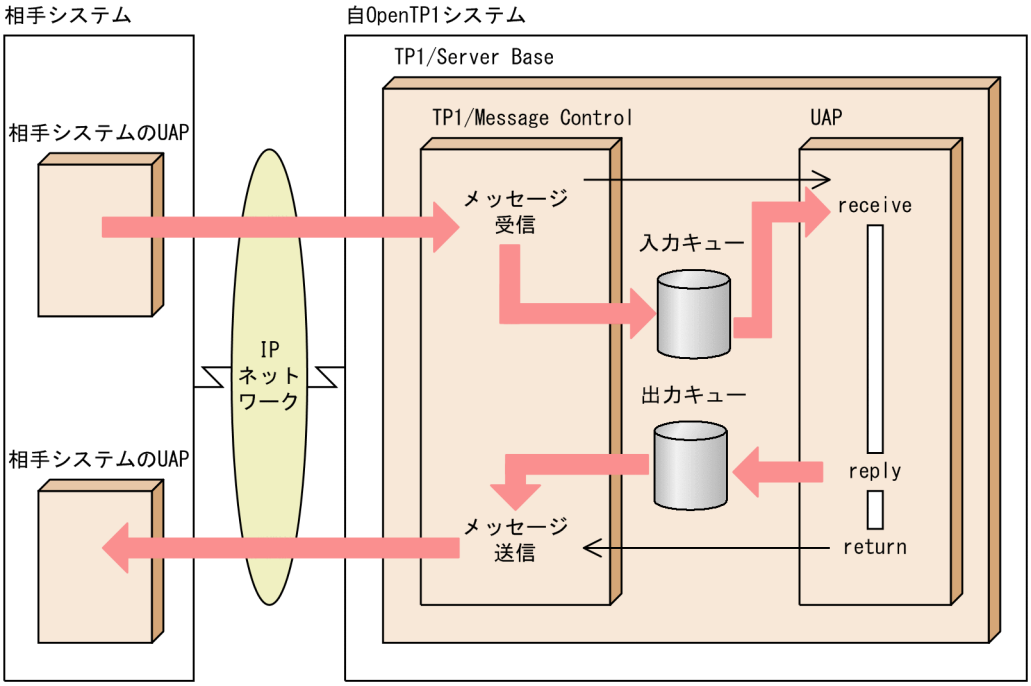
### 2.3.1　メッセージ送受信の概要

メッセージ送受信では，メッセージを格納する待ち行列（**メッセージキュー**）を使用します。

OpenTP1 システムで，相手システムからメッセージを受信すると，MCF は受信したメッセージを**入力キュー**（受信メッセージの待ち行列）に格納して，メッセージを処理する UAP を起動します。UAP ではメッセージを受信して，処理します。メッセージを処理したあとで，送信するメッセージを**出力キュー**（送信メッセージの待ち行列）に格納します。出力キューに格納したメッセージは，MCF で相手システムへ送信します。

メッセージ送受信の処理概要を次の図に示します。

図 2-3　メッセージ送受信の処理概要




### 2.3.2　使用できるネットワーク

メッセージを送受信するには，ネットワークの各種プロトコルに対応した OpenTP1 の通信プロトコル対応製品を使用します。主に IP ネットワーク上でメッセージを送受信できます。

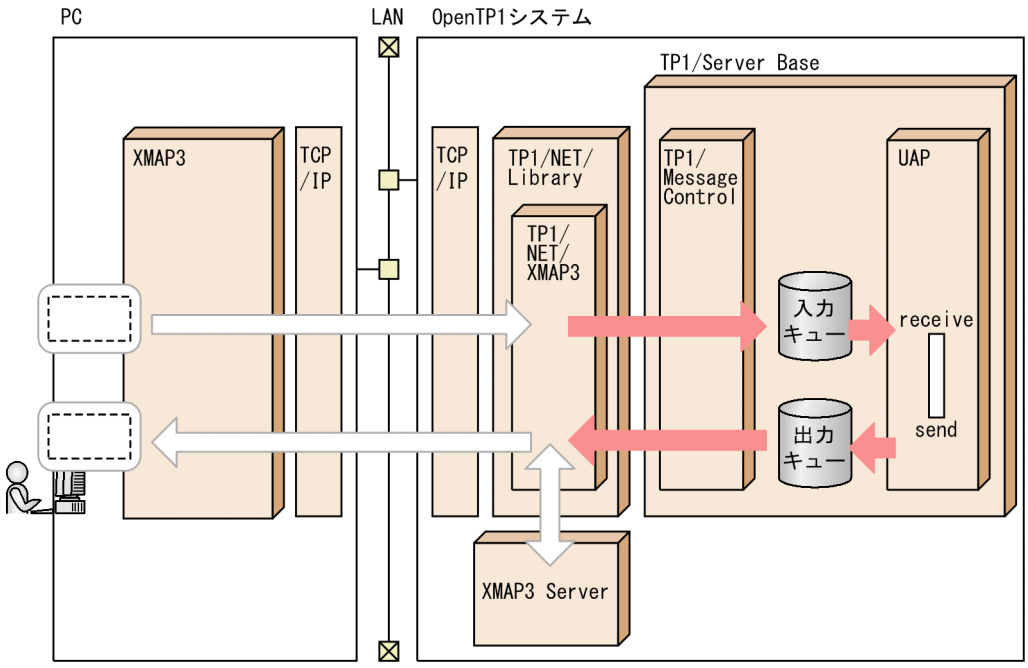
## 2.3.3　XMAP3 を使ったメッセージ送受信の形態

メッセージ送受信の業務処理形態で，XMAP3 を使用した**通信**を実現できます。XMAP3 を使用すると，OpenTP1 システムの UAP の処理で，PC に画面を表示したり，帳票を印刷したりできます。

XMAP3 を使用した通信では，**TP1/NET/XMAP3** を使用します。OpenTP1 システムの通信相手となる PC には XMAP3 が必要です。PC で動作する XMAP3 は，GUI 機能を提供します。OpenTP1 システムにサービス要求するクライアント UAP として，PC 側でプログラムを作成する必要はありません。

XMAP3 を使ったメッセージ送受信形態の例を次の図に示します。

図 2-4　XMAP3 を使ったメッセージ送受信形態




プレゼンテーション機能の特長を次に示します。

### (1)　メッセージ送受信による画面データの操作

XMAP3 を使う場合，TP1/NET/XMAP3 は，XMAP3 Server と連携して物理マップと論理マップ間のマッピング処理を実行しています。

UAP を作成するときは，画面上のマッピングについて意識する必要はありません。また，画面表示のために特別なメッセージを作成する必要はありません。通常のメッセージ送受信で論理メッセージを扱うのと同様に，UAP を作成できます。

### (2)　PC の GUI 環境での操作の実現

XMAP3 を使うと，PC の画面で操作できます。さらに，GUI を使ったインタフェース環境で，各種のデータ操作を実現できます。

## (3) 業務形態に適合したシステムを構築

XMAP3 は，クライアント/サーバシステムを採用しています。そのため，OpenTP1 システムと PC の間で，最適な業務形態に合わせたシステムを構築できます。

TP1/NET/XMAP3 については，マニュアル「OpenTP1 プロトコル TP1/NET/XMAP3 編」を，XMAP3 については，マニュアル「XMAP3 Version 5 画面・帳票サポートシステム XMAP3 実行ガイド」を参照してください。

## 2.4　メッセージキューイング機能を使った形態の処理概要

OpenTP1 では，米国 IBM 社で開発した通信仕様（**メッセージキューイング機能**）を使って通信できます。

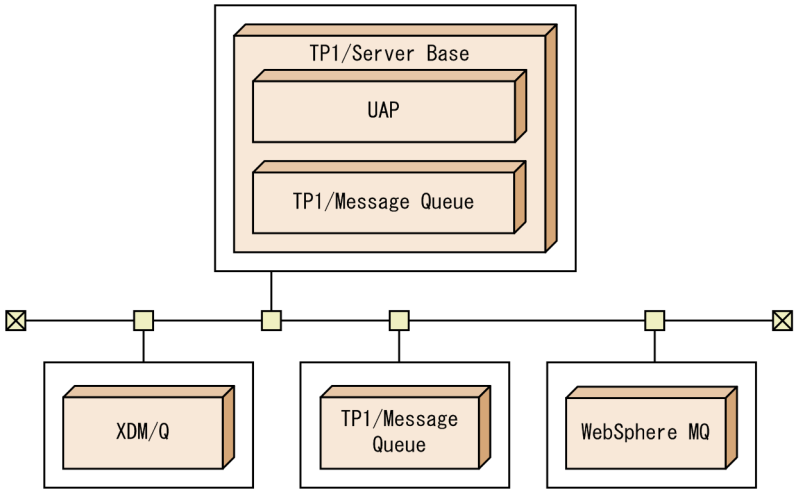
メッセージキューイング機能を使う場合は，メッセージキューイング機能を管理する製品（**キューマネジャ**）が必要です。メッセージキューイング機能では，アプリケーション間通信をキューマネジャが管理するため，UAP で通信手順を意識する必要はありません。

OpenTP1 システムのキューマネジャとなる製品は，TP1/Message Queue です。メッセージキューイング機能を使う場合には，OpenTP1 システムに TP1/Message Queue が必要です。

OpenTP1 のメッセージキューイング機能については，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

メッセージキューイング機能を使った OpenTP1 システムの構成の例を次の図に示します。

図 2-5　メッセージキューイング機能を使った OpenTP1 システムの構成の例




### 2.4.1　メッセージキューイング機能の特長

メッセージキューイング機能の通信では，UAP がキューマネジャへメッセージを登録したり，キューマネジャからメッセージを取り出したりして通信します。UAP が通信先のあて先をメッセージに設定して登録すれば，キューマネジャが通信相手のキューマネジャへメッセージを送信します。

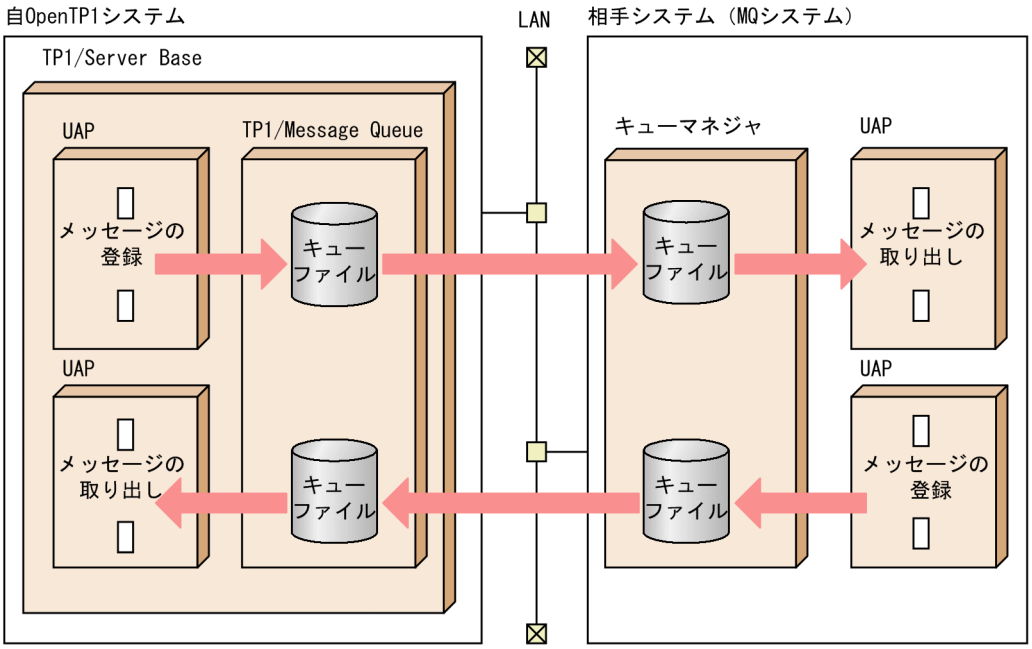
キューにメッセージを登録する場合もメッセージを取り出す場合も，通信する相手の UAP と同期を取らなくて済みます。そのため，UAP では任意のタイミングでメッセージを扱え，電子メールのような通信ができます。

UAP からキューへアクセスするときには，MQI という API を使います。MQI はキューマネジャで共通であるため，コーディングした処理をほかのキューマネジャのシステムで使えるようにしています。

UAP から登録されたメッセージを格納するキューを**メッセージキュー**といいます。メッセージキューには，キューマネジャの目的別に各種のキューを格納できます。メッセージキューを格納するファイルを，**MQA キューファイル**といいます。キューの種類については，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

メッセージキューイング機能を使った処理の概要を次の図に示します。

図 2-6　メッセージキューイング機能を使った処理の概要




## 2.4.2　メッセージキューイング機能を使った通信の概要

メッセージキューイング機能を使った通信の概要について説明します。

### (1)　メッセージの登録

通信相手にメッセージを渡したい場合は，UAP はキューマネジャへメッセージを登録します。メッセージを登録するときには，MQPUT を使います。MQPUT が正常に終了すると，メッセージは登録され，UAP はほかの処理を実行できます。

通信相手の UAP からの応答を待つ（会話型の通信）場合は，MQPUT に応答を待つように設定します。

### (2)　メッセージの受け取り

メッセージを受け取るときは，MQGET を使います。メッセージを受け取るタイミングは，メッセージを受け取る UAP が稼働しているかどうかで，次に示す 2 とおりに分かれます。

- **UAP のプロセスが稼働している場合**
  UAP が MQGET を実行した時点で，その UAP あてのメッセージをキューマネジャから受け取れます。

- UAP のプロセスが稼働していない場合

  キューマネジャがメッセージ受け取りを知らせる通知をします。この機能を**トリガ**といい，通知される情報をトリガイベントといいます。トリガイベントは，専用の UAP で受信するようにします。トリガイベントを受ける UAP を，**トリガモニタアプリケーション**といいます。トリガモニタアプリケーションは，該当する UAP を起動します。

### (3)　メッセージキューイング機能のトランザクション処理

OpenTP1 では，メッセージキューイング機能の処理をトランザクションとして扱えます。UAP でデータを更新してメッセージをキューマネジャへ登録したあとにトランザクションをコミットすると，登録が有効になります。トランザクションがロールバックされると，登録処理を無効にできます。トランザクションについては，「3.1　トランザクション制御」を参照してください。

## 2.4.3　メッセージキューイング機能を使う場合の注意

メッセージキューイング機能の特長は，UAP を常に起動させておかなくてもよいため，UAP が任意のタイミングでメッセージを受け取れることです。メッセージキューイング機能で会話型の通信もできますが，RPC に比べると性能が落ちることがあります。システム間の通信にメッセージキューイング機能を使うかどうかは，業務目的に従ってください。

## 2.5　OpenTP1 に関連するソフトウェア製品

OpenTP1 システムを支援するソフトウェア製品について説明します。OpenTP1 に関連する製品を次に示します。

- 統合システム運用管理機能　JP1
- アプリケーション開発支援ツール　SEWB+

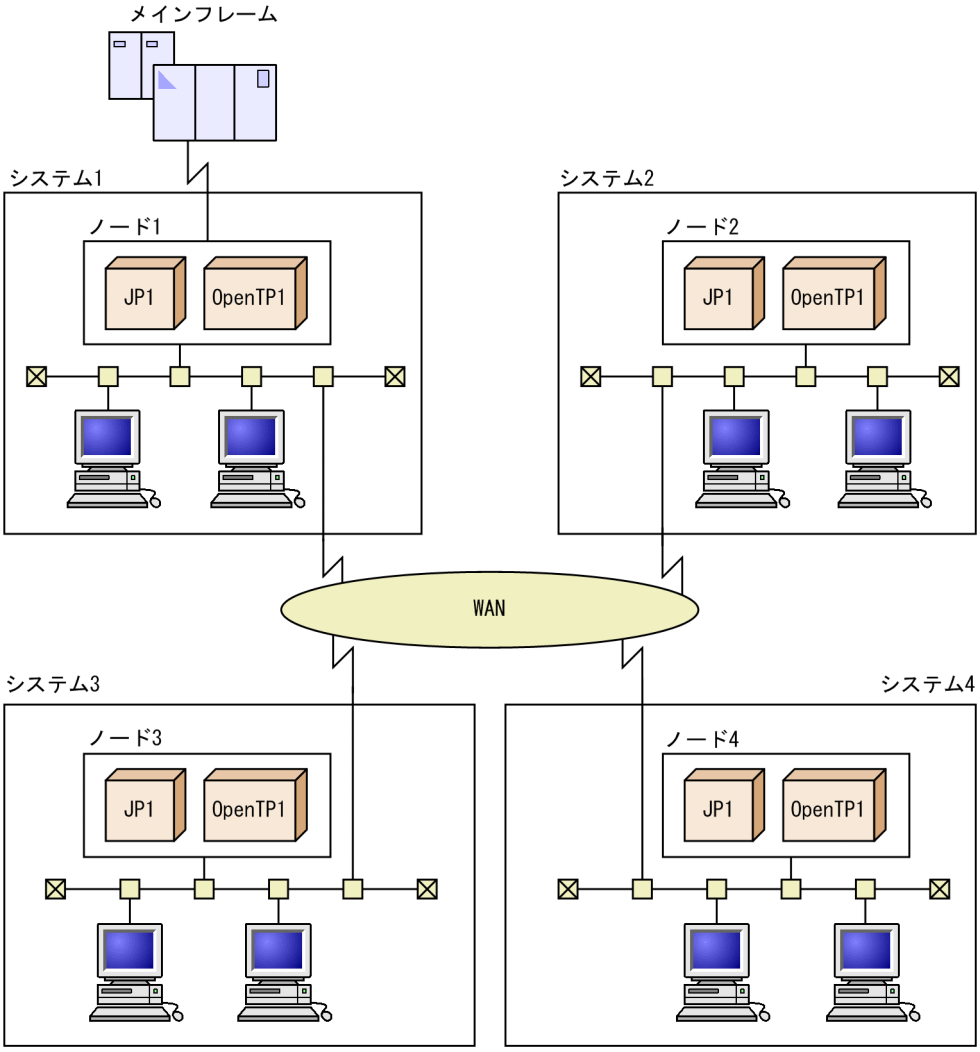
これらの製品と連携して OpenTP1 システムを構築すると，OpenTP1 の分散トランザクション処理に加えて，JP1 による効率的なバッチジョブの運用，ネットワーク管理，およびソフトウェア配布ができるようになります。

### 2.5.1　統合システム運用管理機能　JP1

統合システム運用管理機能 JP1 とは，ネットワークコンピューティング環境の中で，システム全体の運用の自動化，省力化を統合的に支援し，エンタープライズ TCO（企業情報システム全体の運用管理コスト）を最適化する製品です。

JP1 を使った OpenTP1 システムの構成を次の図に示します。

図 2-7　JP1 を使った OpenTP1 システムの構成




ジョブ管理

OpenTP1 システムに JP1 を使うと，メインフレームの業務であるバッチ処理を UNIX システムでできるようになります。そのため，OpenTP1 の業務をオンライン中の業務とバッチ処理業務に分けることができ，システムを効率良く運用できます。さらに，OpenTP1 の開始，終了などのイベントを JP1 イベントサービス機能（JP1/Base）に登録すると，JP1 ジョブ管理機能（JP1/AJS）と連携した OpenTP1 システムの自動運転ができるようになります。

配布管理・資産管理

JP1/NETM/DM を使うと，プログラムを一括してオンライン配布できます。

シナリオテンプレートを利用したシステムの運用

JP1/AJS2 - Scenario Operation は，JP1/AJS で管理していたジョブ，およびジョブネットを，統合的に管理する製品です。

システム構成の変化に対応した運用手順をシナリオテンプレートとして定義したり，運用環境に応じたシナリオを実行したりできます。また，OpenTP1 で提供するシナリオテンプレートを組み合わせてシナリオを作成したり，実行したりすることができます。シナリオテンプレートによって，システムを自動的に運用できます。

JP1/AJS2 - Scenario Operation は，次の製品で構成されます。

- JP1/AJS2 - Scenario Operation Manager
  シナリオテンプレートの定義を保存し，シナリオテンプレートの実行を管理します。JP1/AJS - Manager と接続して，保存したシナリオテンプレートを JP1/AJS のジョブネットとして登録できます。
- JP1/AJS2 - Scenario Operation View
  GUI を使用して JP1/AJS2 - Scenario Operation の操作または監視をします。

JP1/AJS2 - Scenario Operation を使用して，シナリオテンプレートをシステム運用に利用する場合の詳細については，「3.10　シナリオテンプレートを利用したシステムの運用」またはマニュアル「OpenTP1 運用と操作」を参照してください。

システム運用機能に応じた JP1 製品の使用方法については，使用する JP1 製品のマニュアルを参照してください。

## 2.5.2　アプリケーション開発支援ツール　SEWB+

**アプリケーション開発支援ツール　SEWB+**とは，ワークステーション（WS）でのプログラミングを支援するアプリケーション開発支援ツールです。SEWB+には，次のような機能があります。

- プログラム開発に必要な情報を統合して管理できます。
- 設計仕様やプログラムの論理構造を図形で表現できるので，プログラムを効率良く開発できます。

OpenTP1 システムに SEWB+を使うと，UAP や UOC を効率良く開発できます。さらに，C 言語，C++言語，COBOL85，および COBOL2002 を使うこともできます。SEWB+に関連する製品については，マニュアル「SEWB+ クライアントサーバシステム開発ガイド」を参照してください。

## 2.6 OpenTP1 のアプリケーションプログラム

OpenTP1 の UAP について説明します。

### 2.6.1 業務形態と，使用するアプリケーションプログラム

OpenTP1 の UAP には，次に示す種類があります。

#### (1) クライアント/サーバ形態の通信で使用する UAP

- サービス利用プログラム（SUP）
  クライアント専用の UAP です。
- サービス提供プログラム（SPP）
  クライアント UAP からの要求に対して，サービスを提供する UAP（サーバ UAP）です。

#### (2) メッセージ送受信形態の通信で使用する UAP

- メッセージ処理プログラム（MHP）
  通信回線を経由して送られたメッセージを受信して処理する UAP です。

#### (3) ユーザファイルを初期化する UAP

- オフラインの業務をする UAP
  ユーザ任意の処理をする UAP です。

#### (4) OpenTP1 クライアント機能（TP1/Client）で使用する UAP

- クライアントユーザプログラム（CUP）
  OpenTP1 クライアント機能（TP1/Client）で使用する，クライアント専用の UAP です。
  なお，TP1/Client/J を使用すると，SPP のサービスを要求する Java アプレット，Java アプリケーション，および Java サーブレットを作成できます。

### 2.6.2 アプリケーションプログラムの概要

OpenTP1 の UAP の概要について説明します。UAP については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。OpenTP1 の UAP を次に示します。

- SUP
- SPP

- MHP
- オフラインの業務をする UAP
- CUP

CUP については，マニュアル「OpenTP1 クライアント使用の手引」を参照してください。

## (1) サービスを利用する UAP（SUP）

クライアント/サーバ型の通信で，UAP の業務処理を開始するクライアント専用の UAP を**サービス利用プログラム**（SUP）といいます。クライアント/サーバ型の通信では，すべての業務処理は SUP から開始します。

SUP では，トランザクションの開始/終了の制御や，サーバ UAP へのサービス要求ができます。

SUP は，OpenTP1 の開始と同時に開始させることも，オンライン中に OpenTP1 のコマンド（dcsvstart コマンド）で，任意の時期に開始させることもできます。このように，SUP は業務に応じた運用ができます。OpenTP1 の開始と同時に開始させるかどうかは，ユーザサービス構成定義に指定します。

SUP の終了は，OpenTP1 では制御していないので，業務終了後に SUP 自身で終了するように作成します。

SUP はクライアント専用の UAP なので，サーバ UAP としてほかの UAP にサービスを提供できません。

## (2) サービスを提供する UAP（SPP）

クライアント/サーバ型の通信で，ユーザサーバとしてサービスを提供するサーバ UAP を**サービス提供プログラム**（SPP）といいます。SPP では，クライアント UAP からのサービス要求を処理します。また，トランザクションの開始，メッセージの送信もできます。

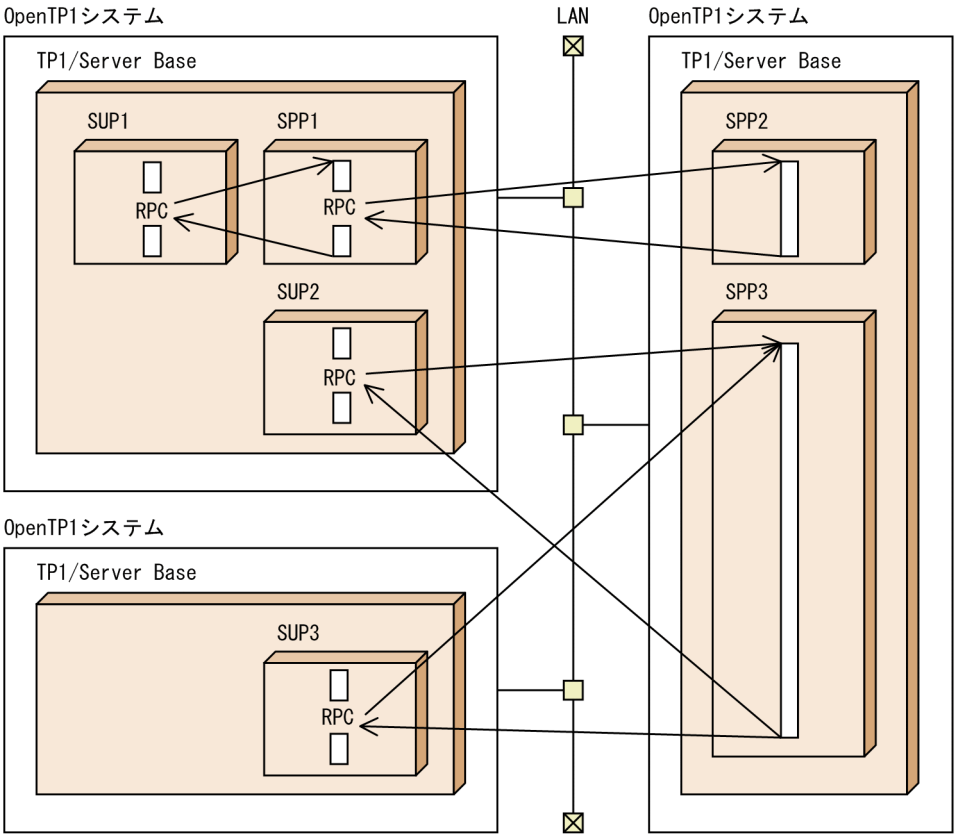
SPP は，OpenTP1 の開始と同時にサービス要求を受け付けられる状態にすることも，オンライン中に OpenTP1 のコマンド（dcsvstart コマンド）で，任意の時期に開始させることもできます。OpenTP1 の開始と同時に開始させるかどうかは，ユーザサービス構成定義に指定します。

SPP を業務処理ごとに作成しておくことで，必要な業務をサービスとして提供できるので，拡張しやすいシステムを構築できます。

SPP から SPP へサービスを要求できるので，業務処理のネストを実現できます。

SUP と SPP の位置を次の図に示します。

図 2-8　SUP と SPP の位置




## (a)　SPP の構造

要求される SPP のサービスに対応する処理を，関数として作成します。この一つ一つの関数を，C 言語の場合は**サービス関数**，COBOL 言語の場合は**サービスプログラム**といいます。個々に作成したサービス関数をまとめる関数を，C 言語の場合は**メイン関数**，COBOL 言語の場合は**メインプログラム**といいます。

## (b)　翻訳，結合

UAP のソースプログラムをコンパイル/リンケージして，実行形式ファイルを作成します。リンケージの際に，各サービス関数の入り口点を定義した**スタブ**を結合させます。

ただし，すべてのサービス関数を UAP 共用ライブラリ化※してサービス関数動的ローディング機能を使う場合，スタブは不要です。サービス関数動的ローディング機能の詳細については，「3.8　サービス関数動的ローディング機能」を参照してください。

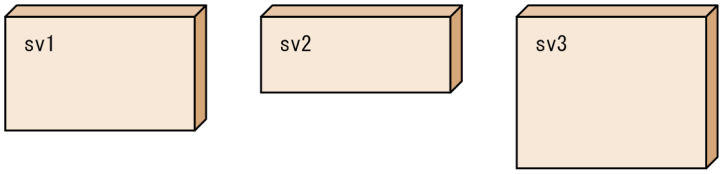
注※

UAP 共用ライブラリ化とは，UAP のソースファイルを翻訳（コンパイル）して作成した UAP オブジェクトファイルを結合（リンケージ）して，共用ライブラリとしてまとめることです。

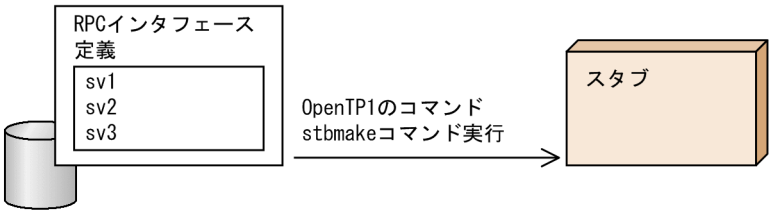
SPP の構造を，スタブを使う場合とサービス関数動的ローディング機能を使う場合に分けて，それぞれを以降の図に示します。UAP の作成方法については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 図 2-9　SPP の構造（スタブを使う場合）

1. クライアントUAPに提供するサービスを，サービス関数として作成します。




2. 各サービスの入り口点を定義したスタブを作成します。




3. メイン関数を作成して，メイン関数，サービス関数，スタブを翻訳，結合すると，SPPの実行形式ファイルとなります。

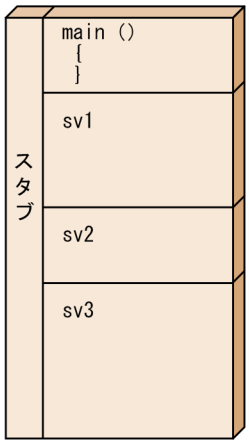
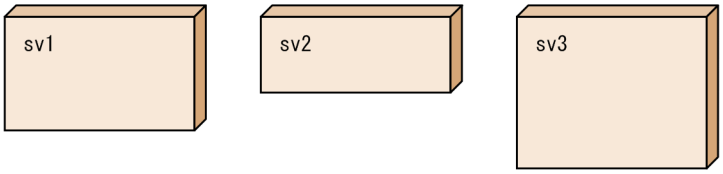
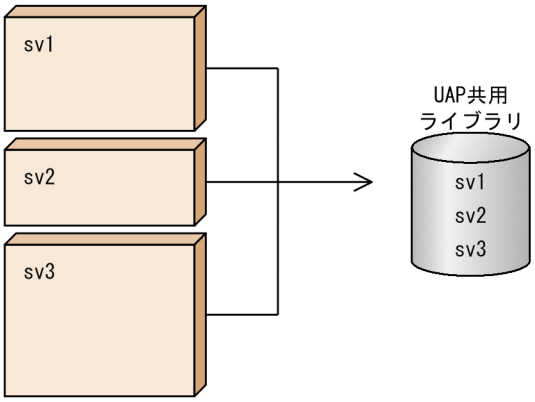


図 2-10　SPP の構造（サービス関数動的ローディング機能を使う場合）

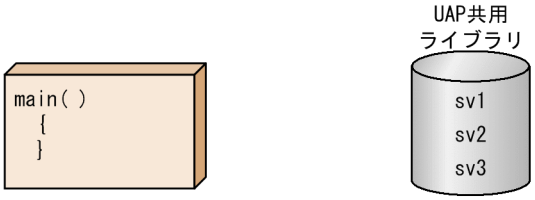
1. クライアントUAPに提供するサービスを，サービス関数として作成します。




2. サービス関数をUAP共用ライブラリとしてまとめます。




3. メイン関数を作成してから，翻訳してSPPの実行形式ファイルを作成します。
SPP実行時は，2. で作成したUAP共用ライブラリからSPP起動時にサービス関数を取得して
実行します。




### (c)　UAP と RPC との関係

SPP の実行形式ファイルを**サービスグループ名**として，それぞれのサービス関数を**サービス名**として，ユーザサービス定義で指定します。サービス名は RPC インタフェース定義で指定した入り口点と対応づけます。

クライアント UAP からサービスを要求するときは，**SPP のサービスグループ名**と**サービス名**を設定します。

一つの SPP を別々のサービスグループとして開始することも，異なるサービスグループのサービス処理を同じサービス名で開始することもできます。

## (3)　メッセージを処理する UAP（MHP）

メッセージ送受信をする場合に使う，メッセージ処理専用の UAP を**メッセージ処理プログラム（MHP）**といいます。MHP では，他システムからのメッセージを受信して，処理をします。

MHP では，メッセージ送信/受信を始めとする，MCF で提供する各種サービスを使用できます。また，MHP から SPP へのサービス要求もできます。

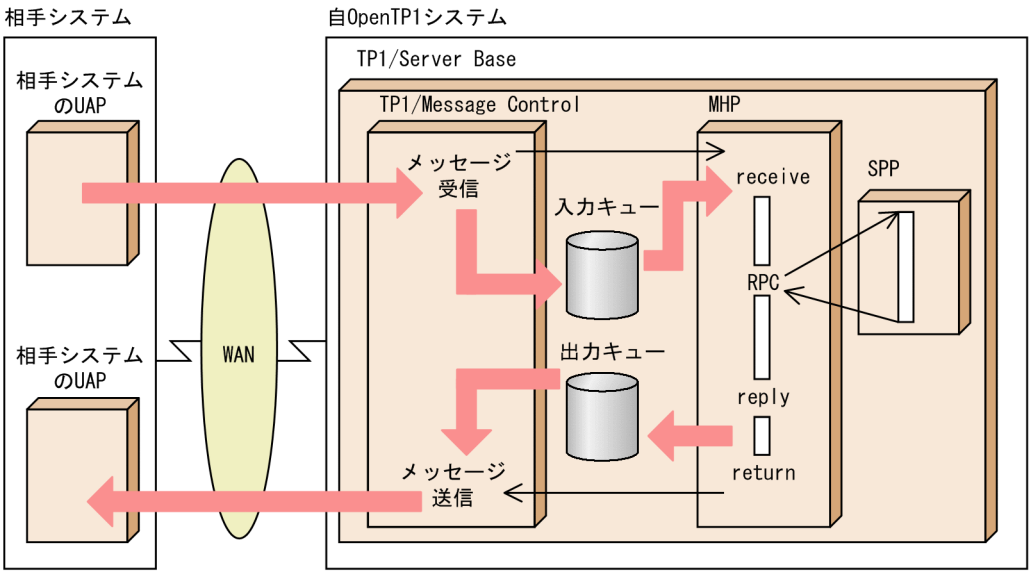
MHP は，メッセージを受信したことを契機に業務処理を開始します。

MHP を OpenTP1 の開始時から開始するかどうかは，ユーザサービス構成定義に指定します。MHP を OpenTP1 の開始時に開始させる指定をすれば，オンライン開始からメッセージを受信できる状態になります。また，開始する指定でない MHP は，OpenTP1 のコマンド（dcsvstart コマンド）で，任意の時期に開始できます。

MHP が稼働するノードは，TP1/Message Control が前提です。

MHP の位置を次の図に示します。

図 2-11　MHP の位置




## (a)　MHP の構造

メッセージ送受信処理では，メッセージを処理する UAP の単位を**アプリケーション**といいます。アプリケーションに対応する MHP の処理を，関数として作成します。この一つ一つの関数を，C 言語の場合は**サービス関数**，COBOL 言語の場合は**サービスプログラム**といいます。個々に作成したサービス関数をまとめる関数を，C 言語の場合は**メイン関数**，COBOL 言語の場合は**メインプログラム**といいます。

## (b)　翻訳，結合

MHP のソースプログラムをコンパイル/リンケージして，実行形式ファイルを作成します。リンケージの際に，各サービス関数の入り口点を定義した**スタブ**を結合させます。スタブは，RPC インタフェース定義を作成したあと，stbmake コマンドを実行して作成します。

ただし，すべてのサービス関数を UAP 共用ライブラリ化※してサービス関数動的ローディング機能を使う場合，スタブは不要です。サービス関数動的ローディング機能の詳細については，「3.8　サービス関数動的ローディング機能」を参照してください。

注※

UAP 共用ライブラリ化とは，UAP のソースファイルを翻訳（コンパイル）して作成した UAP オブジェクトファイルを結合（リンケージ）して，共用ライブラリとしてまとめることです。

MHP の構造を，スタブを使う場合とサービス関数動的ローディング機能を使う場合に分けて，それぞれを以降の図に示します。UAP の作成方法については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 図 2-12　MHP の構造（スタブを使う場合）

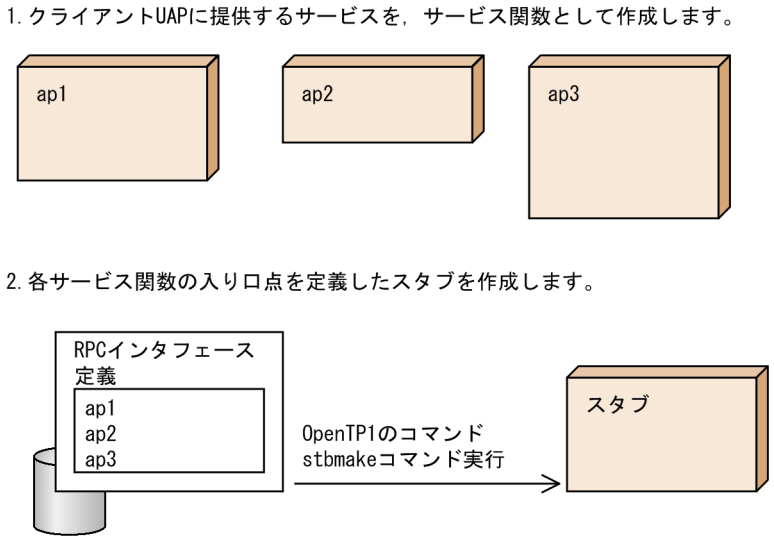
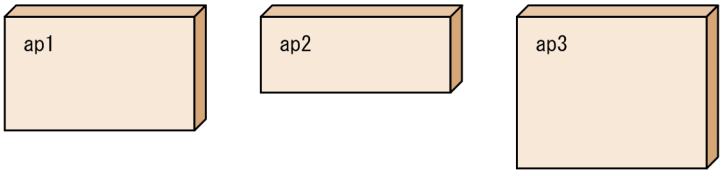
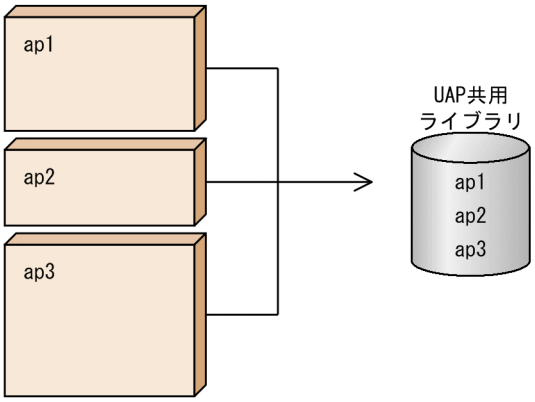


図 2-13　MHP の構造（サービス関数動的ローディング機能を使う場合）

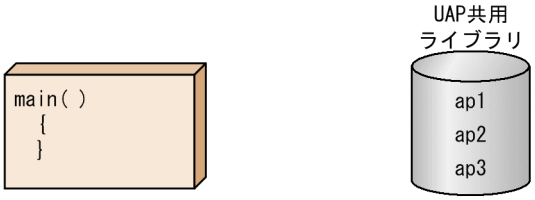
1. クライアントUAPに提供するサービスを，サービス関数として作成します。




2. サービス関数をUAP共用ライブラリとしてまとめます。




3. メイン関数を作成してから，翻訳してMHPの実行形式ファイルを作成します。
MHP実行時は，2. で作成したUAP共用ライブラリからMHP起動時にサービス関数を取得して
実行します。




## (c)　MHP のサービス名とアプリケーションの関係

MHP の実行形式ファイルを**サービスグループ名**として，それぞれのサービス関数を**サービス名**として，ユーザサービス定義で指定します。サービス名は RPC インタフェース定義で指定した入り口点と対応づけます。

ユーザサービス定義で指定した**サービス名**は，MCF アプリケーション定義，アプリケーション属性定義で，**アプリケーション名**と対応づけます。このアプリケーション名を基に，受信したメッセージを処理します。

## (d)　アプリケーションの型

MHP のサービス（アプリケーション）には，メッセージを処理する形態別にアプリケーションの型を決めておきます。アプリケーションの型には次の三つがあります。

- 応答型（ans 型）

  通信相手システムに応答を返す MHP

- **非応答型（noans 型）**
  通信相手システムに応答を返さない MHP
- **継続問い合わせ応答型（cont 型）**
  通信相手システムとのメッセージの受信/送信を連続させる MHP

アプリケーションの型は，MCF アプリケーション定義，アプリケーション属性定義で指定します。

### (4) オフラインの業務をする UAP

オフライン環境で DAM ファイルにアクセスする UAP を作成できます。オフラインの業務をする UAP では，OpenTP1 開始前に DAM ファイルを初期化する処理をします。

オフラインの業務をする UAP からは，DAM ファイルの初期化処理以外の OpenTP1 の機能（RPC やメッセージの処理）は使えません。

オフラインの業務をする UAP は，シェルから開始します。オフラインの業務をする UAP の開始，終了はユーザで管理します。

## 2.6.3 SPP，MHP とユーザプロセスの関連

OpenTP1 では，サービスグループを実行するためのプロセスを複数起動できる**マルチサーバ**を実現できます。同じサービスグループでも，クライアント UAP が異なれば，決まったプロセスで実行されるとは限りません。また，あるサービスから同じサービスグループに属するサービスを要求する場合でも，そのサービスグループを実行するための新しいプロセスが必要になります。

OpenTP1 では，サービスグループごとに実行できる最大プロセス数を定義できます。また，サービス要求は同じサービスグループを処理する待機中のプロセスに渡すため，それぞれのプロセスに掛かる負荷を平均化できます。

UAP のプロセス制御については，「3.4.3 プロセスの制御」を参照してください。

## 2.6.4 UAP のテスト，デバッグ機能

作成した UAP を実際に OpenTP1 の業務に使う前に，UAP の処理動作をテストできます。UAP のテスト機能には，**オフラインテスタ**，**オンラインテスタ**，および **MCF オンラインテスタ**があります。UAP のテスト機能については，マニュアル「OpenTP1 テスタ・UAP トレース使用の手引」を参照してください。

### (1) オフラインテスタ

オンライン業務用の UAP を，オフライン環境でテストするときに使用します。OpenTP1 システムで UAP を使う前にテストする場合に使います。

オフラインテスタでは，SPP，MHP の動作をテストできます。

オフラインテスタを実行するマシンには，TP1/Offline Tester が組み込まれていることが前提です。

## (2) オンラインテスタ

オンライン環境で，OpenTP1 と連携して，UAP をテストするときに使います。UAP に実際の業務をさせる前に，OpenTP1 のサービスを使って動作をテストできます。

オンラインテスタでは SUP，SPP の動作をテストできます。さらに，MHP を SPP としてテストできます。

オンラインテスタを実行するマシンには，TP1/Online Tester が組み込まれていることが前提です。

## (3) MCF オンラインテスタ

オンライン環境で，TP1/Message Control と連携して，MHP をテストするときに使います。オンラインテスタと同様に，オンライン中に OpenTP1 のサービスを使って MHP の動作をテストできます。

MCF オンラインテスタを実行するマシンには，TP1/Message Control/Tester が組み込まれていることが前提です。

## 2.7　インターネットを使った形態の処理概要

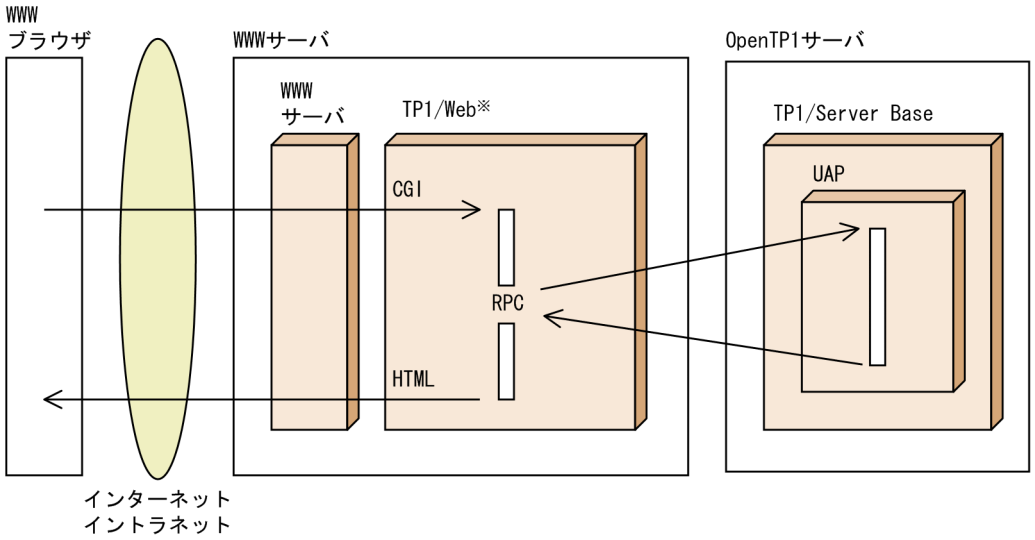
TP1/Web は，インターネット上の WWW サーバと WWW ブラウザの情報のやり取りを OpenTP1 の RPC に変換し，OpenTP1 サーバ上の UAP を実行します。この機能によって WWW ブラウザを端末とする業務システムを構築できます。

TP1/Web は，OpenTP1 と連携して次のような業務システムに適用できます。

- インターネットによる通信販売業務・オンラインショッピング
- インターネットによる乗り物，イベントなどの座席予約やホテルの客室予約のシステム
- 国内外の拠点，出張先からインターネット経由で利用する企業オンラインシステム

また，TP1/Web は，インターネットに接続した構成だけでなく企業内の LAN に接続した構成にも適用できます。次の図に TP1/Web を使った構成を示します。

図 2-14　TP1/Web を使った構成




注※　TP1/Web を使用する場合，TP1/Client が必要です。

# 3 機能

この章では，OpenTP1 の機能について説明します。

## 3.1　トランザクション制御

OpenTP1 を使うと，分散コンピューティング環境で高い信頼性を要求される業務処理プログラムに必要な，各種のトランザクション制御機能を実現できます。OpenTP1 を運用しているときにこれらの機能を使うことで，分散システムに特有な障害対策の負担を軽減できます。

### 3.1.1　分散トランザクション

トランザクションとは，関連する複数の処理を一つのまとまった処理として扱うための論理的な単位です。一つのトランザクション内で実行した処理は，すべてが有効になるか，すべてが無効になるかのどちらかです。

これまでのオンラインシステムは，トランザクションを単体の大型コンピュータで実行していました。このため，一つのコンピュータ内でしかトランザクションの整合性を保証していませんでした。

分散コンピューティング環境で動作する OpenTP1 では，トランザクションはほかのコンピュータに分散して実行されます。このようなトランザクション処理を**分散トランザクション**といいます。OpenTP1 は，分散して処理する一連のトランザクション全体の整合性を保証しています。

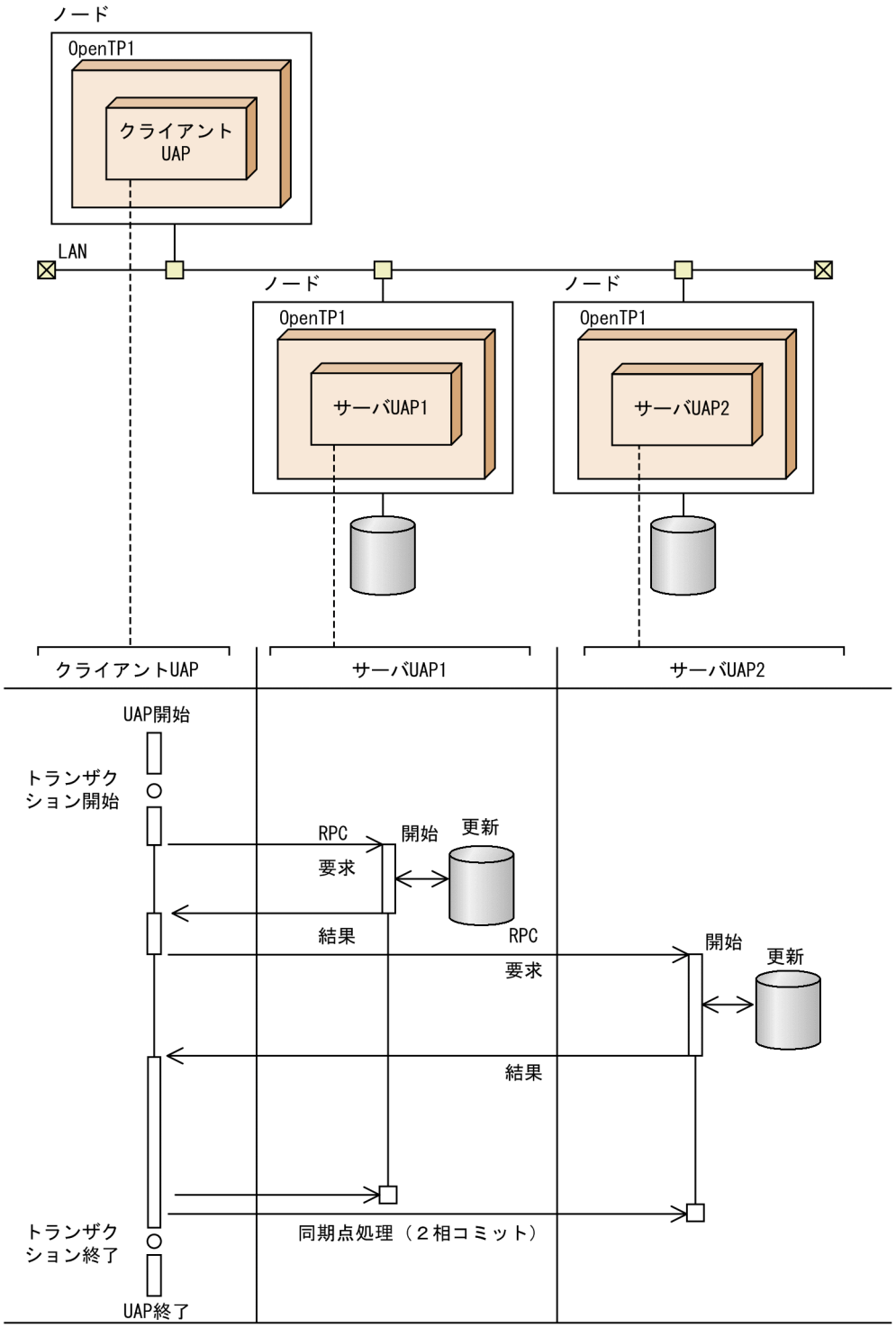
トランザクションは，リソースマネジャの排他制御によって，複数のトランザクションが同時に，同じ資源を更新することはありません。資源は常にトランザクションごとに更新されます。

トランザクションが正常に終了して，トランザクションを有効にすると決定した場合，その後の UAP の処理に関係なく資源を更新することをリソースマネジャに指示します。このため，資源間の不整合が起きません。これは，障害が発生しても資源間で不整合が生じないように，トランザクションに関する履歴情報を，システムジャーナルとして取得しているためです。トランザクションの処理途中に障害が発生して，トランザクションを有効にしないと決定した場合には，トランザクションを開始する前の状態に資源を戻すことをリソースマネジャに指示します。

このように，業務をトランザクションとして処理することで，資源の整合性を保てます。

分散トランザクションを次の図に示します。

図 3-1　分散トランザクション



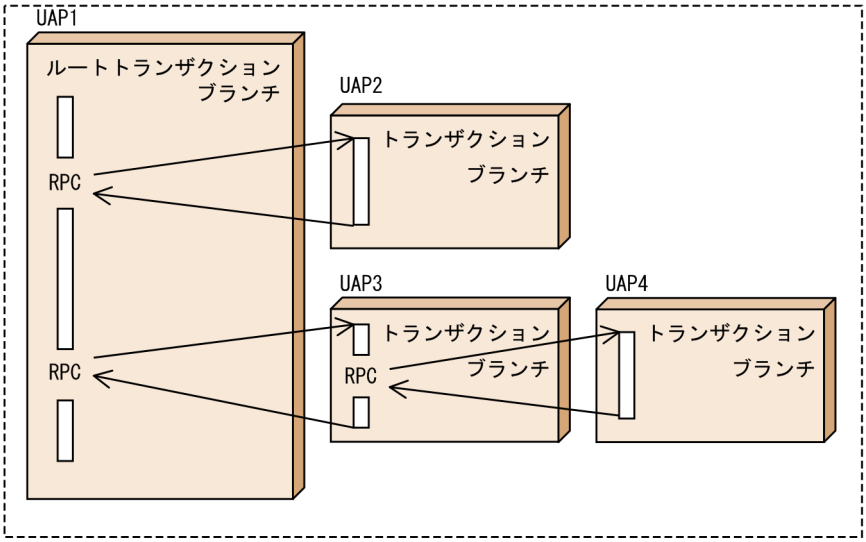

## 3.1.2　グローバルトランザクション

トランザクション処理をする UAP プロセスを**トランザクションブランチ**といいます。RPC を使用した場合，複数の UAP プロセスで構成されるトランザクション処理ができます。複数の UAP プロセスで構成さ

れるトランザクションブランチの集合を**グローバルトランザクション**といいます。特に，トランザクションの開始を宣言したプロセスを**ルートトランザクションブランチ**といいます。

グローバルトランザクションを次の図に示します。

図 3-2　グローバルトランザクション




注　破線の内側がグローバルトランザクション

## 3.1.3　トランザクションのコミットとロールバック

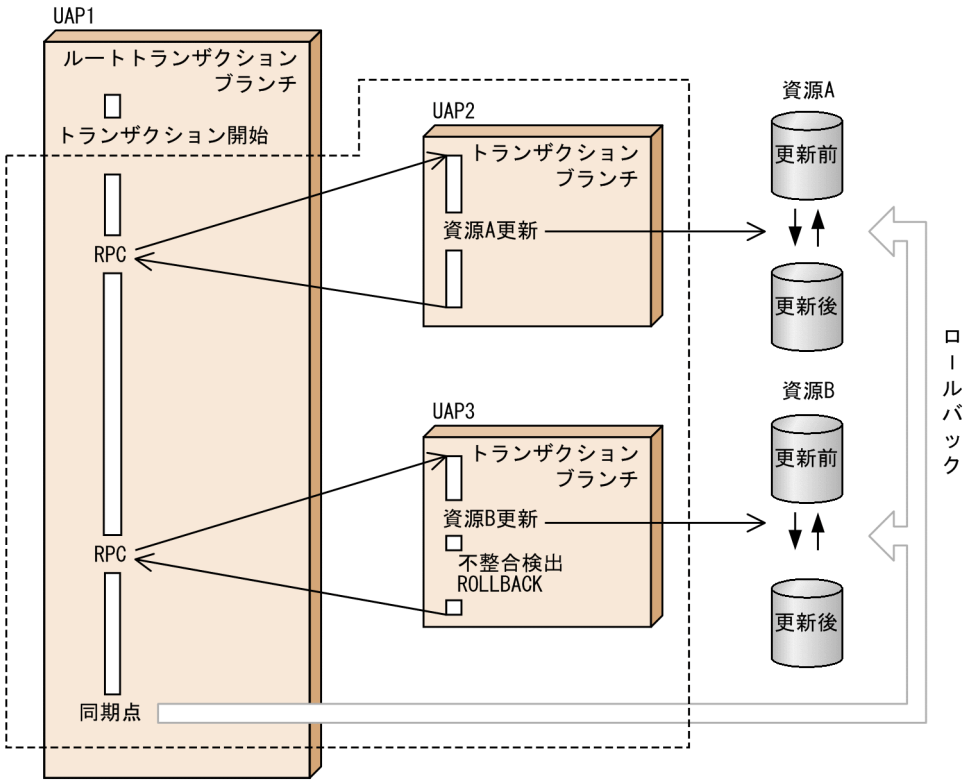
トランザクションは，成功または失敗することによって終了します。トランザクションが成功すること，つまり整合性を保って資源を更新することを，トランザクションの**コミット**といいます。トランザクションがコミットしたときに初めて，グローバルトランザクション内のすべてのトランザクションブランチでの資源の更新が有効になります。トランザクションをコミットするかどうか最終的に決定するトランザクション処理の区切りを，**同期点**といいます。資源は，同期点で更新します。

トランザクションが失敗して，トランザクションで更新するはずの資源の状態を，トランザクション開始直前の状態に戻すことを，トランザクションの**ロールバック**といいます。トランザクションをコミットできなかった場合や，処理の不整合を検出した場合には，これまでの処理をロールバックで取り消して，データの整合性を保ちます。トランザクションがロールバックした要因はログに取得できます。ロールバック要因をログに取得するかどうか，トランザクションサービス定義の trn_rollback_information_put オペランドで指定してください。

OpenTP1 はトランザクションブランチごとに経過時間を監視しています。経過時間を過ぎてもトランザクションが終了しないときには，トランザクションをロールバックします。経過時間の監視については，マニュアル「OpenTP1 システム定義」を参照してください。

トランザクションのロールバックを次の図に示します。

図 3-3　トランザクションのロールバック




注　破線の内側がグローバルトランザクション

## 3.1.4　2 相コミット

### (1)　2 相コミットの概要

複数の資源を更新するとき，トランザクションは同期点で，**2 相コミット**と呼ばれる方法で最終的に資源を更新するかどうか決定します。2 相コミットとは，同期点処理を資源の更新準備処理（**プリペア処理**）と，資源の更新処理（**コミット処理**または**ロールバック処理**）の二段階に分ける方法です。同期点処理を二つの相に分けることで，複数の資源を矛盾なく更新できます。

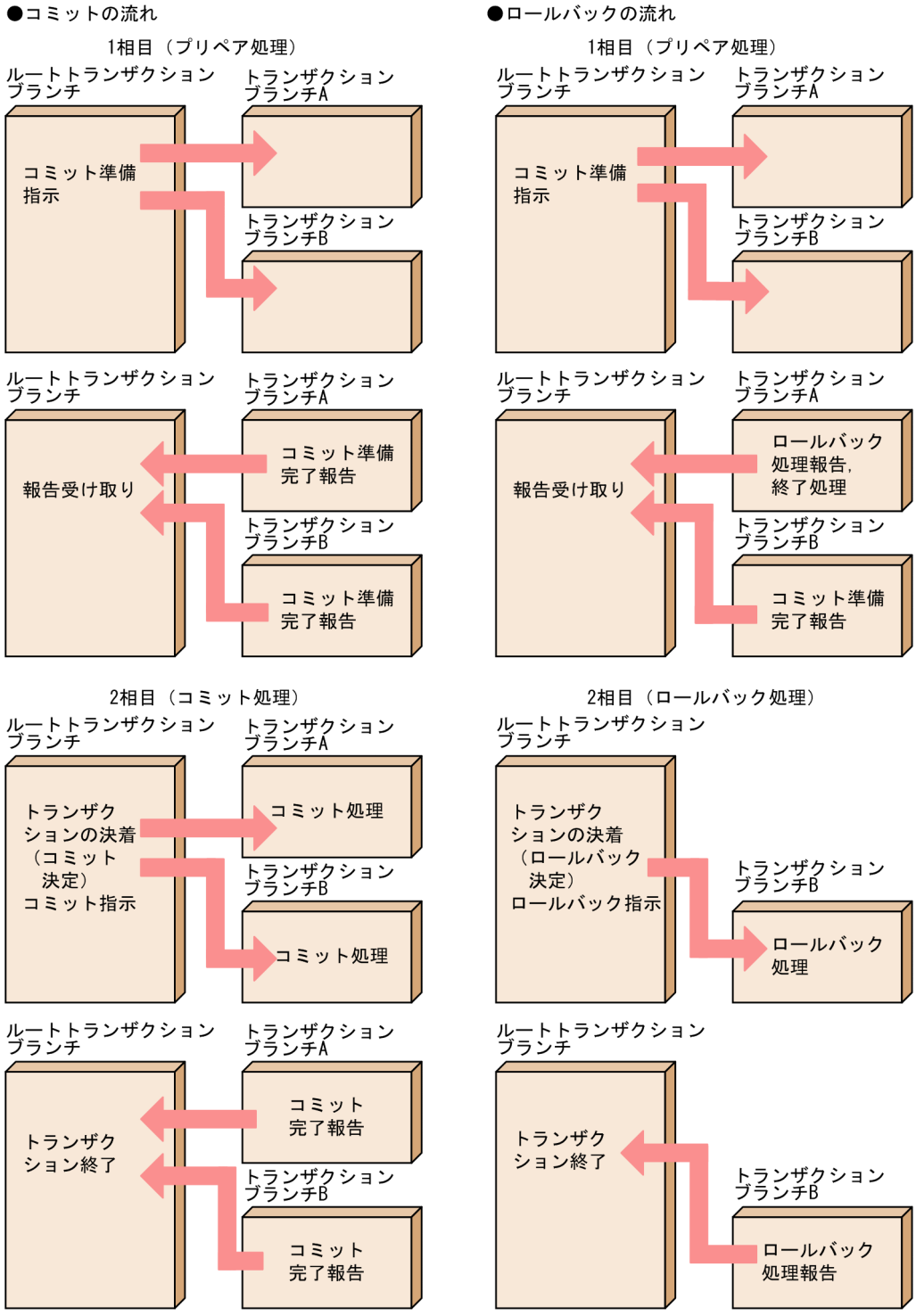
同期点の**第 1 相**で，ルートトランザクションブランチがグローバルトランザクション内のすべてのトランザクションブランチに，コミットの準備を要求します。トランザクションブランチは，コミットの準備が完了したか，ロールバックしたかを，ルートトランザクションブランチに報告します。

同期点の**第 2 相**で，グローバルトランザクションをロールバックするかコミットするか決定します。すべてのトランザクションブランチでコミットの準備が完了すると，ルートトランザクションブランチはグローバルトランザクション全体のコミットを決定します。トランザクションブランチのどれかがロールバックしたときには，ルートトランザクションブランチはグローバルトランザクション全体のロールバックを決定します。グローバルトランザクション内のトランザクションブランチはすべて，ルートトランザクションブランチでの決定に従います。

トランザクションの決着（コミット決定，またはロールバック決定）以降にシステムに障害が発生した場合に備えて，2 相コミット処理の過程でトランザクションに関する履歴情報を，システムジャーナルに取得します。

2 相コミットの流れを次の図に示します。

図 3-4　2 相コミットの流れ



## (2) ヒューリスティック決定

ルートトランザクションブランチがトランザクションブランチにコミットの準備を指示したあとで通信障害が発生して，ルートトランザクションブランチでの決定がトランザクションブランチに届かなくなることを，**ヒューリスティックハザード**といいます。ヒューリスティックハザードが起こった場合には，コマンドを使って，トランザクションブランチごとにトランザクションを強制的に決着できます。ルートトランザクションブランチの同期点処理を待たないで，コマンドでトランザクションブランチのトランザクションを強制的に決着することを，**ヒューリスティック決定**といいます。

ヒューリスティック決定には，次に示す種類があります。

- **ヒューリスティックコミット**
  トランザクションブランチを独自にコミットすることです。
- **ヒューリスティックロールバック**
  トランザクションブランチを独自にロールバックすることです。

ヒューリスティックコミットにするときには trncmt コマンドを，ヒューリスティックロールバックにするときには trnrbk コマンドを使います。コミットまたはロールバックしたトランザクションブランチを強制終了する場合は，trnfgt コマンドを使います。

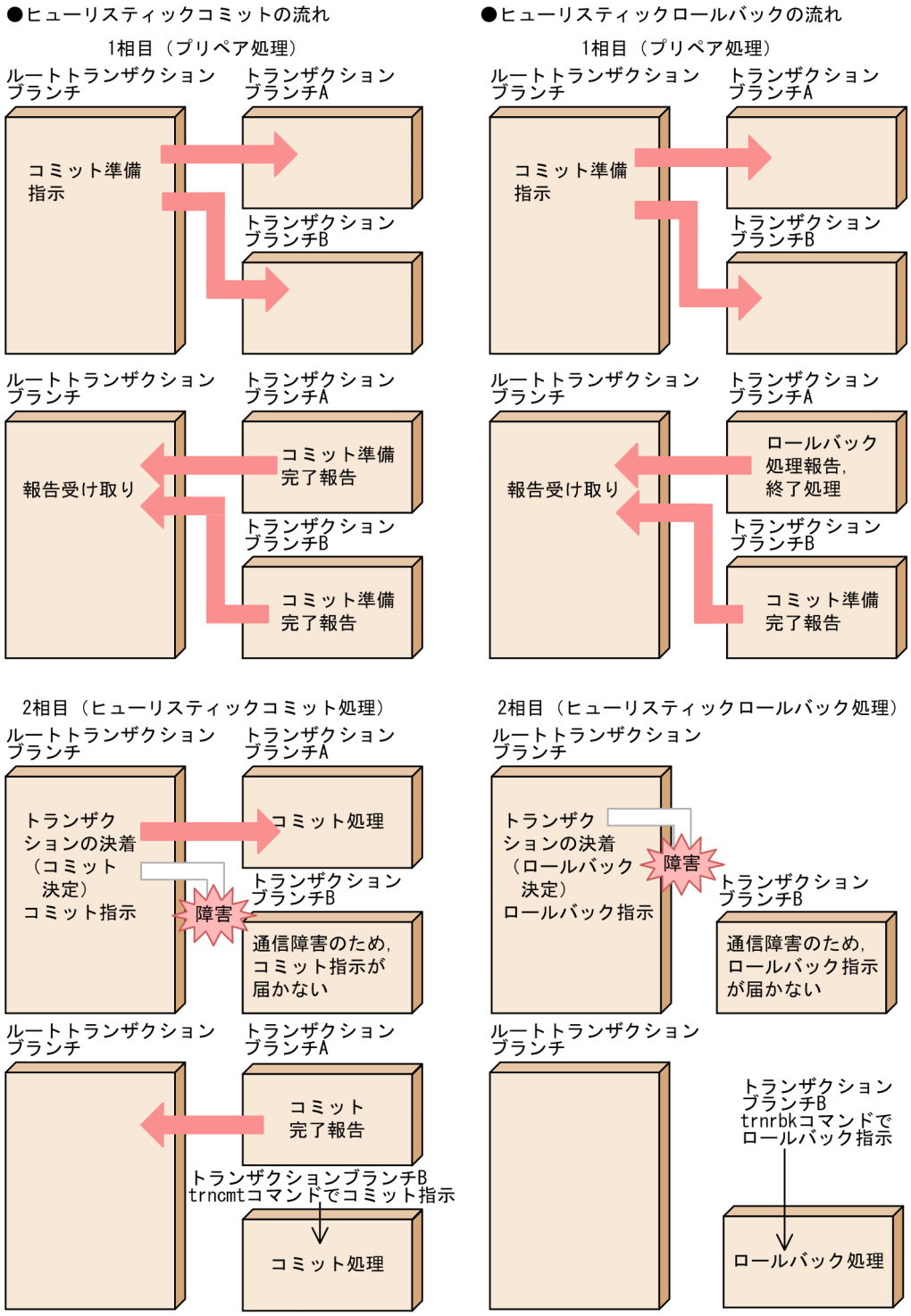
コミットかロールバックかは，ルートトランザクションブランチとトランザクションブランチとで一致させてください。一致させるためには，まずルートトランザクションブランチがコミットしたかロールバックしたかを trnls コマンドを使って調べます。trnls コマンドで表示した状態を参照した上で，trncmt コマンドまたは trnrbk コマンドを実行します。

コマンドでトランザクションを決着させた場合，トランザクションの扱いについては十分考慮してください。ルートトランザクションブランチの決着とトランザクションブランチの決着とが矛盾（**ヒューリスティックミックス**）しないように十分注意してから trncmt コマンド，または trnrbk コマンドを実行してください。

ヒューリスティック決定の操作方法については，マニュアル「OpenTP1 運用と操作」のトランザクションに関する運用の説明を参照してください。

ヒューリスティック決定の流れを次の図に示します。

図 3-5　ヒューリスティック決定の流れ




## 3.1.5　UAP とトランザクションの関係

UAP の処理をトランザクションにするかどうかを，事前に決められます。トランザクションとして処理する UAP を，**トランザクション属性の UAP** といいます。グローバルトランザクションのトランザクションブランチとなる UAP には，必ずトランザクション属性を指定しておいてください。トランザクション属

性は，ユーザサービス定義の atomic_update オペランドで指定します。ファイルの情報を更新する UAP をトランザクション処理として，演算処理だけの UAP はトランザクション処理としない指定をしておくと，トランザクション処理に掛かるシステムの負担を軽くできます。

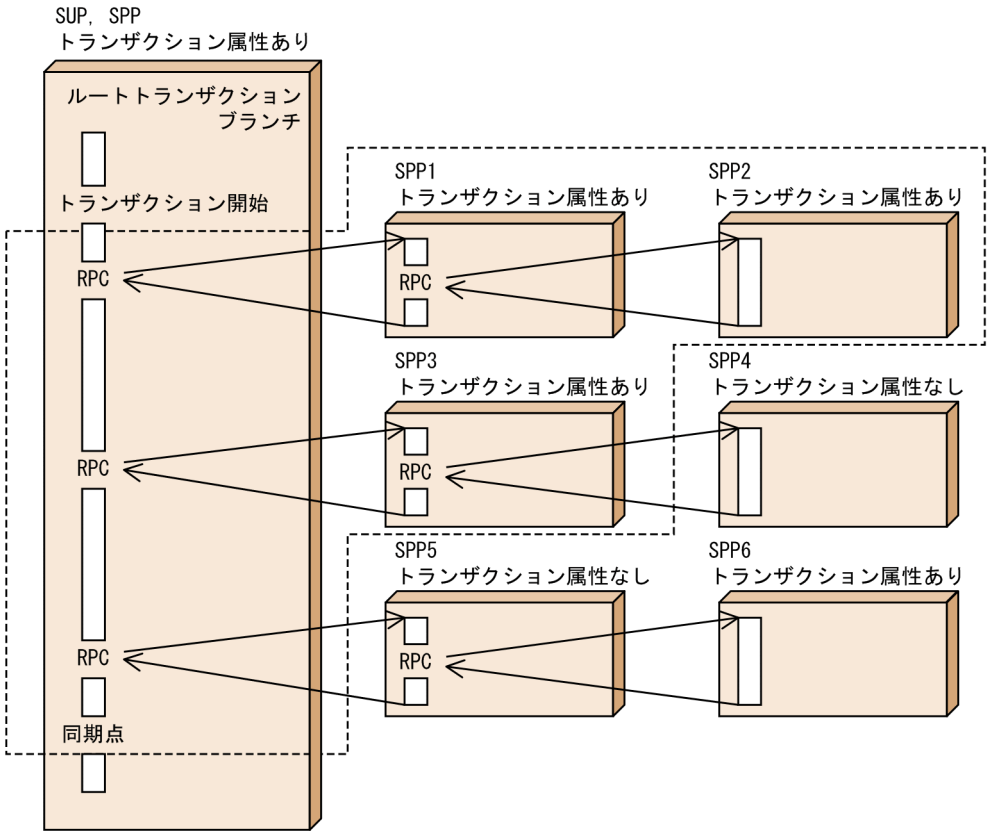
## (1) SUP，SPP とトランザクションの関係

トランザクション属性の SPP は，トランザクションとして処理しているプロセスから RPC で呼び出されると，自動的にトランザクションブランチになります。

ルートトランザクションブランチやトランザクションブランチから呼び出されても，トランザクション属性でない SPP は，グローバルトランザクションに含まれません。このとき，トランザクション属性でない SPP がトランザクション属性の SPP を呼び出したとしても，呼び出された SPP はトランザクションブランチにはなりません。

SUP，SPP のグローバルトランザクションを次の図に示します。

図 3-6　SUP，SPP のグローバルトランザクション




注　破線の内側がグローバルトランザクション

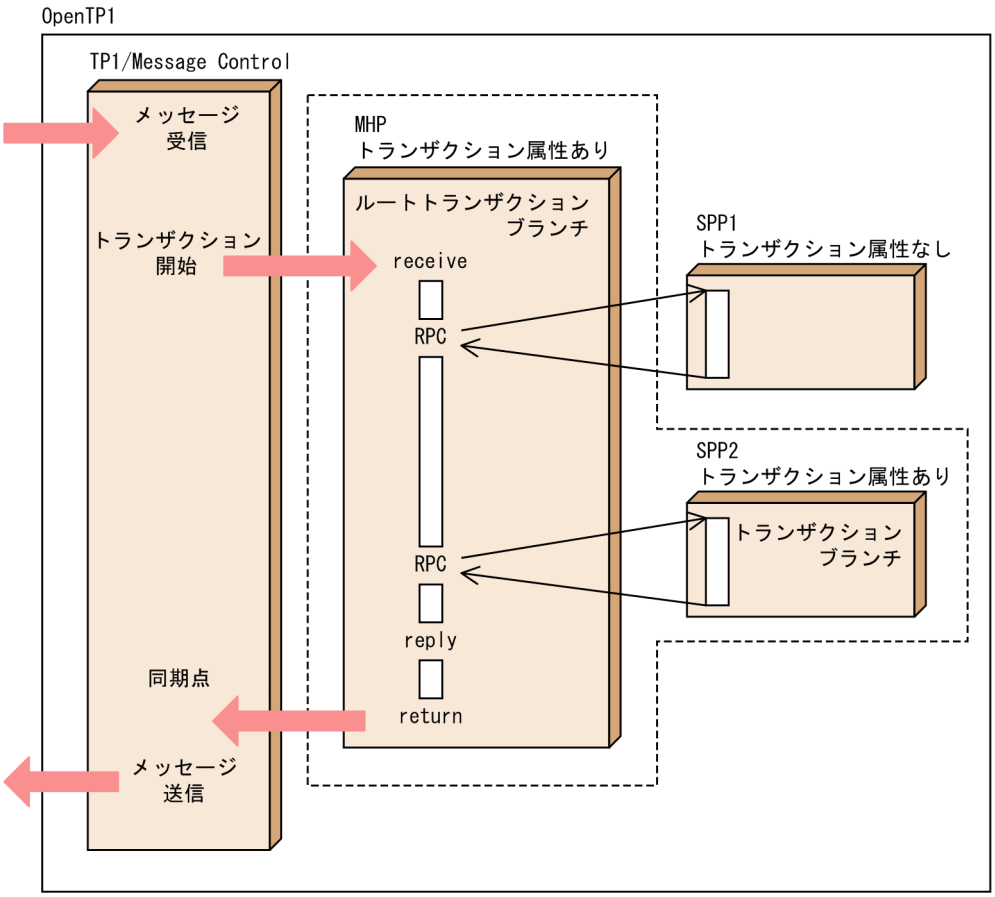
SUP または SPP の処理をトランザクションにする場合は，dc_trn_begin 関数を呼び出して，トランザクションの開始を宣言します。同期点を取得する場合も，SUP または SPP から dc_trn_××××_commit 関数または dc_trn_××××_rollback 関数を呼び出します（××××は，同期点を取得したあとに，それまでとは別の新しいトランザクションを開始するかどうかで異なります）。トランザクションの開始と同期点を取得する関数は，ルートトランザクションブランチからだけ呼び出せます。

### (2)　MHP とトランザクションの関係

MHP のサービスを呼び出した時点で，トランザクションの処理が開始します。MHP では，dc_trn_begin 関数は使えません。MHP から SPP へサービスを要求する場合，MHP がルートトランザクションブランチとなります。MHP のトランザクションは，MHP が終了するかまたは同期点を取得する関数（dc_mcf_commit 関数，dc_mcf_rollback 関数）を呼び出すと決着します。MHP の処理の中で同期点を取得した場合は，同期点取得以降の処理は，それまでとは別の，新しいトランザクションの処理となります。なお，非トランザクション属性の MHP も作成できます。この場合，MHP で同期点は取得できません。

MHP のグローバルトランザクションを次の図に示します。

図 3-7　MHP のグローバルトランザクション




注　破線の内側がグローバルトランザクション

## 3.1.6　TX インタフェースによるトランザクション制御

OpenTP1 では，X/Open で規定する DTP モデルに準拠したトランザクション制御の関数（TX_関数）を使用できます。関数の使い方については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 3.1.7　XA リソースサービスによるトランザクション制御

**XA リソースサービス**とは，OpenTP1 と次に示す連携先との間で 2 相コミットによるトランザクション連携を行うための機能です。

- J2EE で動作するアプリケーションサーバ
- .NET Framework アプリケーション（.NET Framework 上で動作するアプリケーション）

OpenTP1 と.NET Framework アプリケーションとの間で連携する機能を，**MSDTC 連携機能**といいます。MSDTC 連携機能を使用すると，MSDTC と連携できるリソースと OpenTP1 上のリソースとの間でトランザクション連携ができます。MSDTC 連携機能を使用するためには，XA リソースサービス定義の xar_msdtc_use オペランドに Y を指定する必要があります。

ここでは，XA リソースサービスを使ったトランザクション制御の概要について説明します。操作方法については，マニュアル「OpenTP1 運用と操作」の XA リソースサービスの記述を参照してください。また，XA リソースサービスのトレース情報の詳細については，「5.3.6(5)　XAR 性能検証用トレース」，および「5.3.6(9)　XAR イベントトレース」を参照してください。

なお，J2EE で動作するアプリケーションサーバと.NET Framework アプリケーションで機能差がない場合，XA リソースサービスで連携するアプリケーションサーバと呼びます。

### (1)　XA リソースサービスの概要

XA リソースサービスの主な機能は，XA リソースサービスで連携するアプリケーションサーバから渡された XID（トランザクションの識別子）のステータスを管理し，OpenTP1 の XID とマッピングすることです。MSDTC 連携機能を使用する場合は，RI（トランザクション回復情報）も管理します。RI とは，トランザクション決着処理で障害が発生した場合にトランザクションを回復するため情報で，トランザクション決着処理時に MSDTC が作成します。これらのトランザクションの状態は，必要に応じて OpenTP1 ファイルシステムに記録されます。

注

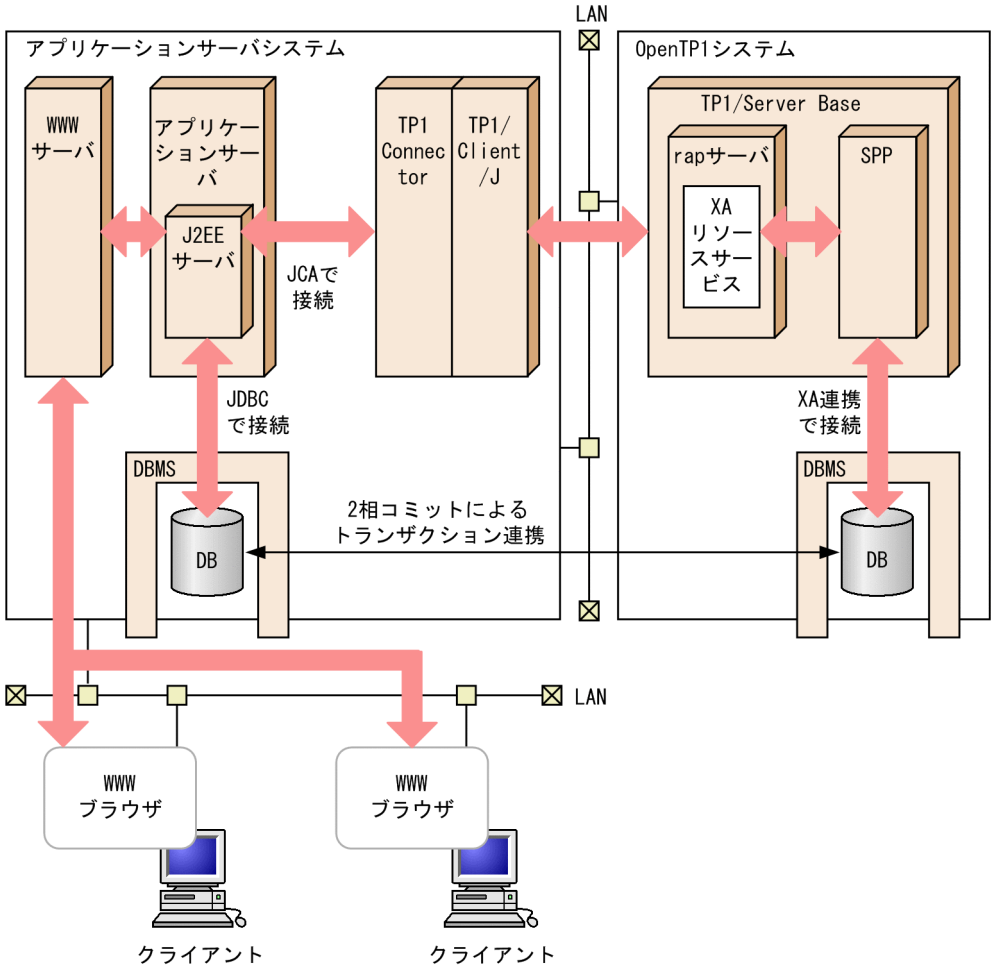
XA リソースサービスの「XA」は，OpenTP1 が DBMS に対してトランザクション決着の指示を行う XA 連携機能の「XA」とは異なります。

#### (a)　J2EE で動作するアプリケーションサーバと連携する場合

J2EE で動作するアプリケーションサーバは，JCA に準拠した uCosminexus TP1 Connector または Cosminexus TP1 Connector を経由してトランザクションを制御します。OpenTP1 では，J2EE で動作するアプリケーションサーバからのトランザクション要求を rap サーバで受信し，XA リソースサービスを使用してトランザクション処理を行います。

J2EE で動作するアプリケーションサーバと連携する場合のトランザクション制御の流れを次の図に示します。

図 3-8　J2EE で動作するアプリケーションサーバと連携する場合のトランザクション制御

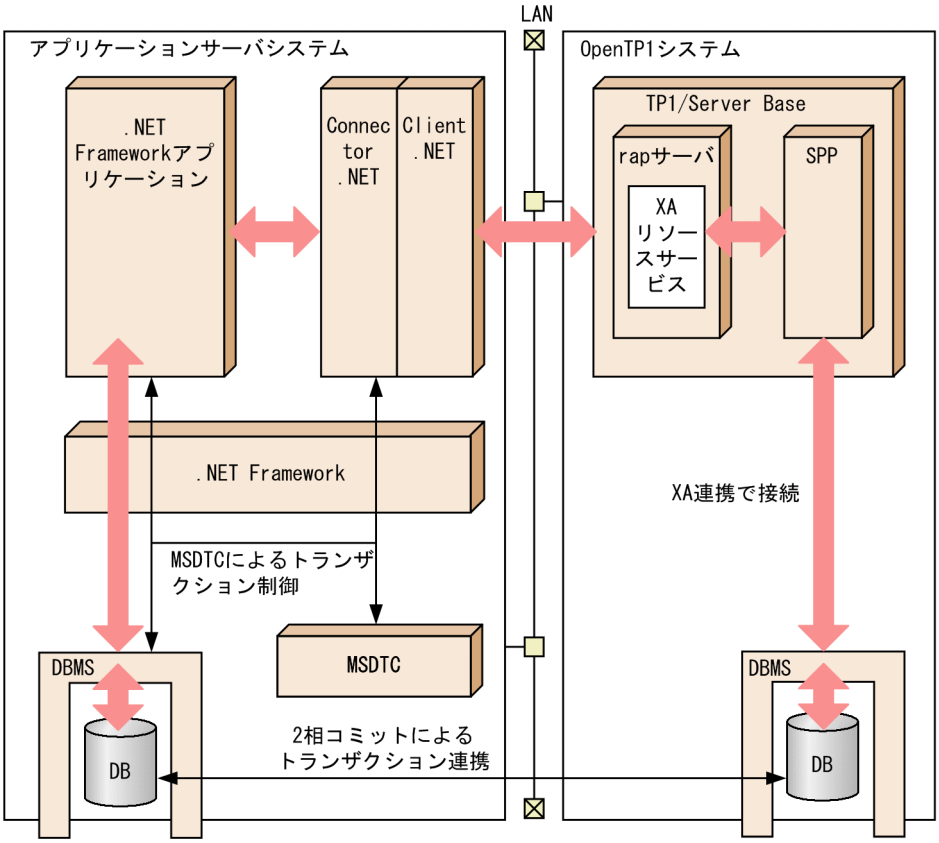



## (b)　.NET Framework アプリケーションと連携する場合

.NET Framework アプリケーションは，MSDTC を使用してトランザクションを制御し，Connector .NET を経由して OpenTP1 へのトランザクション指示をします。OpenTP1 では，.NET Framework アプリケーションからのトランザクション指示を rap サーバで受信し，XA リソースサービスを使用してトランザクション処理を行います。

.NET Framework アプリケーションと連携する場合のトランザクション制御の流れを次の図に示します。

図 3-9　.NET Framework アプリケーションと連携する場合のトランザクション制御




## (2)　JCA の XA リソースインタフェース

J2EE で動作するアプリケーションサーバから OpenTP1 に対するトランザクション制御は，JCA に従って行われます。JCA とは，アプリケーションサーバとリソースアダプタとの間を接続するための標準仕様のことです。

JCA に従うことで，アプリケーションサーバは，DBMS やほかのリソースアダプタを扱うのと同様に，OpenTP1 のトランザクションを制御できます。また，OpenTP1 は，さまざまなアプリケーションサーバ上で標準リソースアダプタとして動作できます。

JCA に従ったトランザクション制御のインタフェースには，次の二つがあります。

- XA リソースインタフェース
- ローカルトランザクションインタフェース

各インタフェースの詳細については，JCA のドキュメントを参照してください。

XA リソースサービスは，OpenTP1 上で，XA リソースインタフェースを使用したトランザクション制御を受け付けるための機能です。

## (3)　XA リソースサービスの前提機能

XA リソースサービスを使用するためには，リモート API 機能と，次に示す製品を使用する必要があります。

J2EE で動作するアプリケーションサーバと連携する場合

- TP1/Client/J
- uCosminexus TP1 Connector または Cosminexus TP1 Connector

.NET Framework アプリケーションと連携する場合

- Client .NET
- Connector .NET

なお，.NET Framework アプリケーションと連携する場合，TP1/Client for .NET Framework 構成定義の<extendLevel 要素>の value 属性には 00000001 を指定することをお勧めします。<extendLevel 要素>の value 属性に 00000001 を指定すると，XAR ファイルのレコード長が不足したことによってトランザクションの決着処理に失敗した場合に，KFCA32045-E メッセージにトランザクション要求元の IP アドレスを表示させることができます。

## (a) リモート API 機能

**リモート API 機能**は，クライアント側が発行した API を，OpenTP1 がサーバ側に転送してサーバ側のプロセスで代理実行する機能です。また，API を代理実行するサーバを rap サーバといいます。

XA リソースサービスは，リモート API 機能を使用した rap サーバ上で動作します。XA リソースサービスで連携するアプリケーションサーバからのトランザクション要求は，すべて rap サーバ上で管理します。そのため，XA リソースサービスを使用する場合は，rap サーバをあらかじめ起動しておく必要があります。rap サーバが起動していない場合，XA リソースサービスで連携するアプリケーションサーバからのトランザクション要求はすべてエラーとなります。

OpenTP1 は，リモート API を要求した UAP と rap サーバとの間に，常設コネクションという論理的な通信路を設定します。常設コネクションのスケジュール方式には，スタティックコネクションスケジュールモードとダイナミックコネクションスケジュールモードの 2 種類があります。XA リソースサービスは，どちらのモードを使用しても動作します。

リモート API 機能の詳細については，「3.7　リモート API 機能」を参照してください。

## (b) TP1/Client/J または Client .NET

**TP1/Client/J** は，J2EE で動作するアプリケーションサーバからのトランザクション要求を OpenTP1 に伝える際に経由する製品です。

**Client .NET** は，.NET Framework アプリケーションからのトランザクション要求を OpenTP1 に伝える際に経由する製品です。

XA リソースサービスを使用する場合は，リモート API 機能が前提となります。したがって，TP1/Client/J または Client .NET の RPC によるサービス要求方式は，リモート API 機能に限られます。また，リモート API 機能を使用するときのコネクトモードは，オートコネクトモードに限られます。

TP1/Client/J または Client .NET の RPC のサービス呼び出し形態には，同期応答型，非応答型，連鎖型およびトランザクションを引き継がない形態の 4 種類があります。XA リソースサービスは，どの形態も使用できます。

通常の RPC では，トランザクション要求に対する窓口となる OpenTP1 のホスト名を複数指定できますが，XA リソースサービスを使用する場合，XA リソースサービスで連携するアプリケーションサーバからのトランザクション要求に対する窓口となる OpenTP1 のホスト名は，一つだけ指定できます。

XA リソースサービスで使用できる TP1/Client/J または Client .NET の機能を次の表に示します。

### 表 3-1　XA リソースサービスで使用できる TP1/Client/J または Client .NET の機能一覧

| TP1/Client/J または Client .NET の機能 | 方式 | 使用の可否 |
|---|---|---|
| RPC のサービス要求方式 | リモート API 機能 | ○ |
| | スケジューラダイレクト機能 | × |
| | ネームサービス機能 | × |
| コネクトモード | オートコネクトモード | ○ |
| | 非オートコネクトモード | × |
| RPC のサービス呼び出し形態 | 同期応答型 | ○ |
| | 非応答型 | ○ |
| | 連鎖型 | ○ |
| | トランザクションを引き継がない形態 | ○ |
| 窓口となる OpenTP1 ホスト名の指定（dchost オペランドで指定） | 一つだけ指定できる | ○ |
| ユーザデータの圧縮 | データ圧縮機能 | ○ |

（凡例）

○：使用できます。

×：使用できません。

TP1/Client/J の各機能の詳細については，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を，Client .NET の各機能の詳細については，マニュアル「TP1/Client for .NET Framework 使用の手引」を参照してください。

## (c)　uCosminexus TP1 Connector または Cosminexus TP1 Connector

uCosminexus TP1 Connector または Cosminexus TP1 Connector は，OpenTP1 の通信およびトランザクションを，JCA に従ったリソースアダプタとして制御するための製品です。J2EE で動作するアプリケーションサーバは，uCosminexus TP1 Connector または Cosminexus TP1 Connector に対するトランザクション制御を行うことで，OpenTP1 に 2 相コミットを伝えます。

XA リソースサービスを使用して OpenTP1 と J2EE で動作するアプリケーションサーバと連携するときの注意事項については，uCosminexus TP1 Connector または Cosminexus TP1 Connector のドキュメントを参照してください。

### (d) Connector .NET

Connector .NET は，.NET Framework 環境から OpenTP1 への通信およびトランザクションを制御する製品です。MSDTC 連携機能では，Connector .NET は MSDTC のトランザクションに参加するリソースマネジャとして機能します。MSDTC は，Connector .NET に対するトランザクション制御を行うことで，OpenTP1 に 2 相コミットを伝えます。

XA リソースサービスを使用して OpenTP1 と.NET Framework アプリケーションと連携するときの注意事項については，マニュアル「TP1/Connector for .NET Framework 使用の手引」を参照してください。

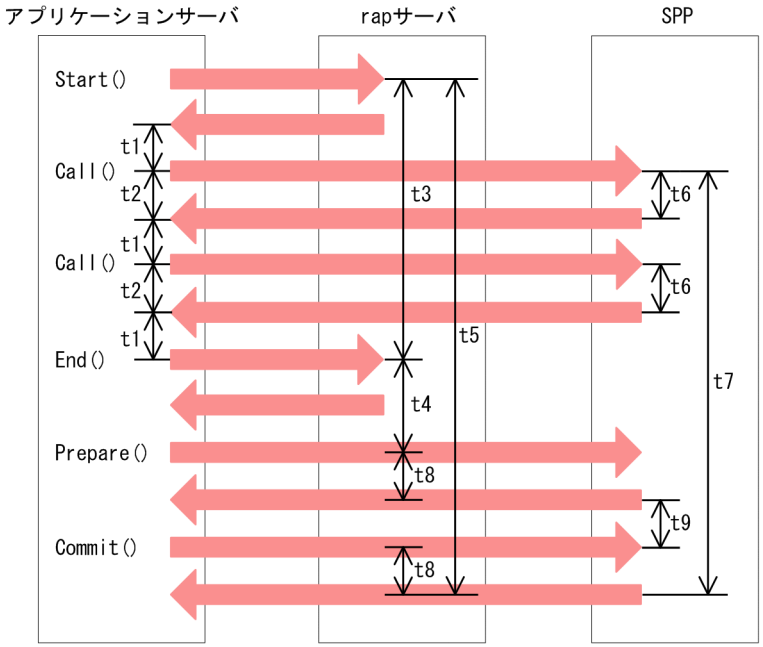
## (4) タイマ監視機能

XA リソースサービスで連携するアプリケーションサーバからの要求が停止した場合や，業務プログラムの処理が滞留した場合，各種タイマ監視機能を使用してトランザクション処理を打ち切ることができます。

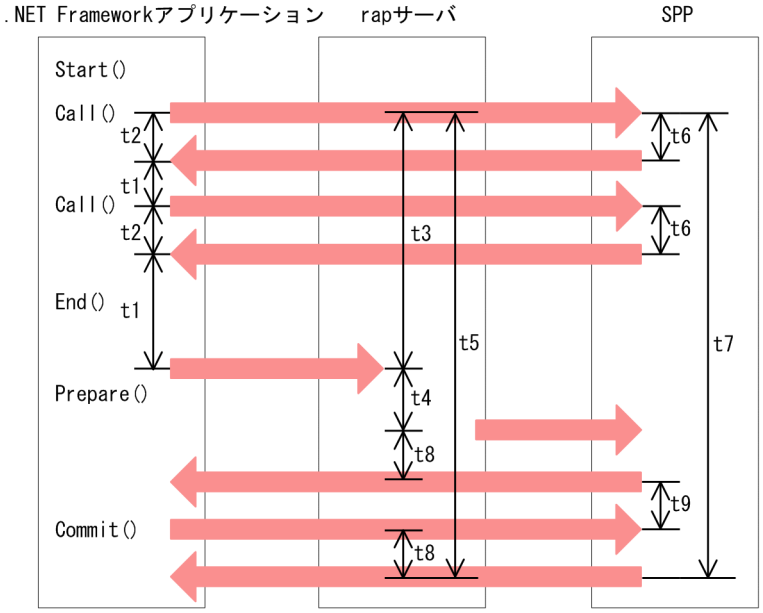
XA リソースサービスで連携するアプリケーションサーバ，rap サーバ，および SPP のタイマ監視機能の範囲を次の図に示します。

## 図 3-10　各種タイマ監視機能の範囲

●J2EEで動作するアプリケーションサーバと連携する場合




●.NET Frameworkアプリケーションで動作するアプリケーションサーバと連携する場合




注 アプリケーションサーバ側のStart()～Commit()は，OpenTP1の内部関数です。

図中のタイマ監視 t1～t9 の対象と，指定するオペランドを次の表に示します。

## 表 3-2　タイマ監視の対象とオペランドの一覧

| 区間 | タイマ監視の対象 | J2EE で動作するアプリケーションサーバ※ | .NET Framework アプリケーション※ | OpenTP1 |
|---|---|---|---|---|
| t1 | 問い合わせ間隔の監視 | TP1/Client/J 環境定義の dccltinquiretime オペランド | TP1/Client for .NET Framework 構成定義の | rap リスナーサービス定義の rap_inquire_time オペランド |

| 区間 | タイマ監視の対象 | J2EE で動作するアプリケーションサーバ※ | .NET Framework アプリケーション※ | OpenTP1 |
|---|---|---|---|---|
| t1 | 問い合わせ間隔の監視 | TP1/Client/J 環境定義の dccltinquiretime オペランド | <rapService>要素の inquireTime 属性 | rap リスナーサービス定義の rap_inquire_time オペランド |
| t2 | メッセージ送受信の監視 | TP1/Client/J 環境定義の dcwatchtim オペランド | TP1/Client for .NET Framework 構成定義の <rpc>要素の watchTime 属性 | rap リスナーサービス定義，ユーザサービスデフォルト定義，またはシステム共通定義の watch_time オペランド |
| t3 | トランザクションブランチの処理の監視 | TP1/Client/J 環境定義の dcclttrexptm オペランド | TP1/Client for .NET Framework 構成定義の <xarTransaction>要素の expireTime 属性 | rap リスナーサービス定義，ユーザサービスデフォルト定義，またはトランザクションサービス定義の trn_expiration_time オペランド |
| t4 | アイドル状態のトランザクションブランチの監視 | − | − | XA リソースサービス定義の xar_session_time オペランド |
| t5 | トランザクションブランチの処理の完了監視 | − | − | rap リスナーサービス定義，ユーザサービスデフォルト定義，またはトランザクションサービス定義の trn_completion_limit_time オペランド |
| t6 | トランザクションブランチの処理の監視 | − | − | ユーザサービス定義，ユーザサービスデフォルト定義，またはトランザクションサービス定義の trn_expiration_time オペランド |
| | サービス関数の処理の監視 | − | − | ユーザサービス定義，ユーザサービスデフォルト定義の service_expiration_time オペランド |
| t7 | トランザクションブランチの処理の完了監視 | − | − | ユーザサービス定義，ユーザサービスデフォルト定義，またはトランザクションサービス定義の trn_completion_limit_time オペランド |
| t8 | トランザクション同期点処理時の最大通信待ち時間 | TP1/Client/J 環境定義の dcclttrwatchtime オペランド | TP1/Client for .NET Framework 構成定義の <transaction>要素の internalWatchTime 属性 | rap リスナーサービス定義の trn_watch_time オペランド |

| 区間 | タイマ監視の対象 | J2EE で動作するアプリケーションサーバ※ | .NET Framework アプリケーション※ | OpenTP1 |
|---|---|---|---|---|
| t9 | トランザクション同期点処理時の最大通信待ち時間 | － | － | ユーザサービス定義，ユーザサービスデフォルト定義，またはトランザクションサービス定義の trn_watch_time オペランド |

（凡例）

－：該当しません。

注※

定義の詳細については，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」，「TP1/Client for .NET Framework 使用の手引」を参照してください。

## 3.1.8 XA インタフェースについて

XA インタフェースは，X/Open が規定した DTP モデルの一部です。

XA インタフェースとは，トランザクションマネジャとリソースマネジャとの間での指示のやり取りを規定したものです。したがって，UAP が XA インタフェースを直接使用することはありません。OpenTP1 とリソースマネジャが使用する XA ライブラリ・サブルーチンを次の表に示します。

### 表 3-3 XA ライブラリ・サブルーチン

| 関数名 | XA ライブラリ・サブルーチン |
|---|---|
| ax_reg | トランザクションマネジャに，リソースマネジャを追加します。 |
| ax_unreg | トランザクションマネジャでの，リソースマネジャの登録を削除します。 |
| xa_close | リソースマネジャをクローズします。 |
| xa_commit | すべてのリソースマネジャがコミット可能（prepare 状態）であることがわかったので，リソースマネジャにコミットすることを通知します。 |
| xa_end | リソースマネジャに，トランザクションの終了を通知します。 |
| xa_forget | リソースマネジャの判断で，終了していたトランザクションの情報を破棄してよいということを通知します。 |
| xa_open | リソースマネジャをオープンします。 |
| xa_prepare | トランザクションがコミットしようとしていることを，リソースマネジャに通知します。 |
| xa_recover | リソースマネジャ内の，未決着トランザクションのリストを取得します。 |
| xa_rollback | トランザクションをロールバックすることを，リソースマネジャに通知します。 |
| xa_start | トランザクションが開始，再開始されたことを，リソースマネジャに通知します。 |

## 3.2　クライアント/サーバ形態の通信

OpenTP1 では，次に示す三つのクライアント/サーバ形態の通信ができます。

- **OpenTP1 のリモートプロシジャコール（RPC）通信**

  OpenTP1 のライブラリ関数を使った，UAP プロセス間の通信方法です。UAP から関数を使って，通信相手にサービスを要求します。通常の RPC では，各ノードのネームサービスが管理しているサーバ UAP のサービス情報を使って通信しています。各ノードのネームサービスは，目的のサーバ UAP がネットワーク上のどのノードにあるかを，サービス情報を交換することで管理します。また，**サービス情報検索の付加機能**を使用することで，運用形態に合わせたサービスの要求方法を選択できます。さらに，**ノード管理**を行うことで，ネットワーク障害によって OpenTP1 同士の通信ができなくなった場合，そのあとに実行する RPC に障害が発生するのを防止します。

  それぞれについては，次を参照してください。

  - RPC の概要：3.2.1　OpenTP1 のリモートプロシジャコール通信
  - サービス情報検索の付加機能：3.2.2　サービス情報検索の付加機能
  - ノード管理：3.2.3　OpenTP1 のノード管理
  - RPC の処理の流れ：付録 D　リモートプロシジャコールの処理の概要

- **XATMI インタフェースの通信**

  X/Open で規定する DTP モデルに準拠した通信方法です。

- **TxRPC インタフェースの通信**

  X/Open で規定する，トランザクション機能付きの DCE RPC ができる通信方法です。TxRPC の通信では，ユーザが作った関数を直接呼び出す形式で通信します。

通信プロトコルに TCP/IP を使っている場合は，上記のクライアント/サーバ形態の通信はすべてできます。通信プロトコルに OSI TP を使っている場合は，**XATMI インタフェースの通信**を使います。OSI TP 通信をする場合は，OpenTP1 システムに TP1/NET/OSI-TP-Extended が必要です。OSI TP 通信については，「3.6　OSI TP を使ったクライアント/サーバ形態の通信」を参照してください。

### 3.2.1　OpenTP1 のリモートプロシジャコール通信

OpenTP1 のライブラリ関数を使った RPC について説明します。

#### (1)　サービスの要求方法

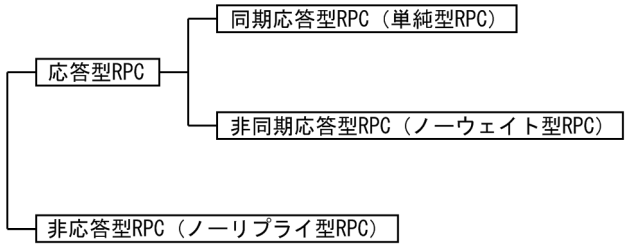
クライアント UAP（SUP，MHP，SPP）では，サービス要求に必要な項目を dc_rpc_call 関数に設定してサーバ UAP（SPP）にサービスを要求します。サーバ UAP は，複数のサービスを実行できます。**サービス**とは，RPC で呼び出す処理単位であり，各サーバ UAP がクライアント UAP に提供する機能です。また，一つのサーバ UAP が提供するサービスの集まりを**サービスグループ**といいます。

クライアント UAP は，dc_rpc_call 関数に**サービスグループ名**および**サービス名**を指定して，サーバ UAP にサービスを要求します。**ネームサービス**でサーバ UAP があるノードのネットワークアドレスとサービス名を管理しているので，ネットワーク上のどのノードにサーバ UAP があるかをクライアント UAP で意識する必要はありません。

## (2) リモートプロシジャコールの形態

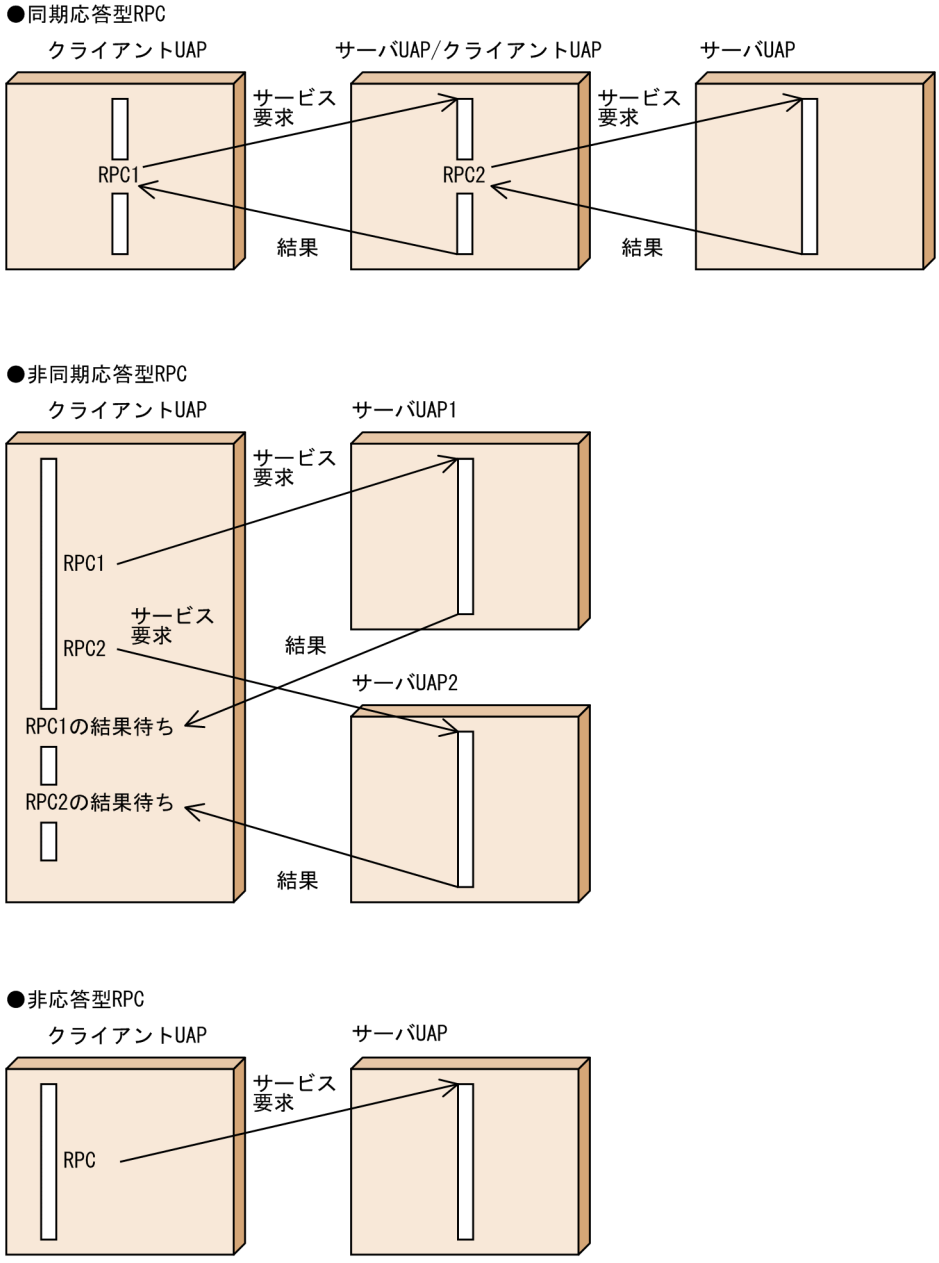
RPC は，応答を受け取るかどうかで，応答型 RPC と非応答型 RPC に分類されます。RPC の種類を次の図に示します。

図 3–11 RPC の種類




RPC の形態別の処理概要を次の図に示します。

図 3-12　RPC の形態別の処理概要




## (a)　同期応答型 RPC

サービスを要求してから，応答が返ってくるのを待つ RPC です。応答が返るまでの待ち時間を監視します。最大応答待ち時間を過ぎた場合，サービス要求はエラーリターンします。最大応答待ち時間は，システム共通定義，またはユーザサービス定義で指定します。

同期応答型 RPC は，**単純型 RPC** ともいいます。

## (b)　非同期応答型 RPC

サービスを要求してから，応答が返ってくるのを待たないで，処理を続ける RPC です。応答を受信する関数（dc_rpc_poll_any_replies 関数）を使って，応答を受信します。応答を受信する関数は，応答を受信するまで待ちます。この関数に設定した応答待ち時間を過ぎた場合，関数はエラーリターンします。

非同期応答型 RPC は，**ノーウェイト型 RPC** ともいいます。

### (c) 非応答型 RPC

サービス要求の処理結果が戻らない RPC です。非応答型 RPC でサービス要求をした場合，応答は受け取れません。サービスを要求した UAP は，処理を続けます。

非応答型 RPC は，**ノーリプライ型 RPC** ともいいます。

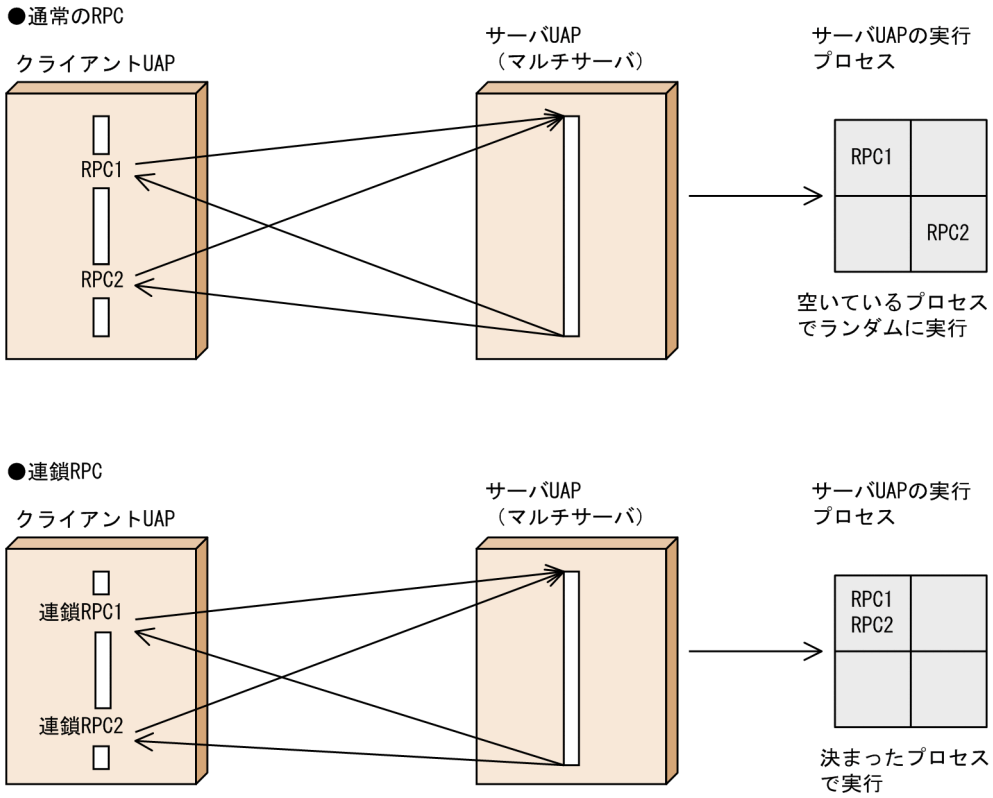
サービス要求を受け取ったサーバ UAP から，さらに別のサービスを要求することもできます（**RPC のネスト**）。

## (3) リモートプロシジャコールの連鎖（連鎖 RPC）

サーバ UAP の実行プロセスは，マルチサーバ（同じサーバ UAP を複数のプロセスで同時に起動する機能）の場合，サービスが要求されるたびに起動されます。一つのクライアント UAP から同じサービスグループを 2 回以上呼び出したとき，そのサービスグループのサーバ UAP が以前と同じプロセスで実行されるとは限りません。ただし，同期応答型 RPC で，かつ同じサービスグループに属するサービスを 2 回以上要求する場合に限り，そのサービスを以前と同じプロセスで実行させることができます。これを**連鎖 RPC** といいます。連鎖 RPC でサービスを要求すると，マルチサーバのサーバ UAP でも前回の RPC と同じプロセスで実行されるため，トランザクション処理に必要なプロセスを最小限にできます。UAP のプロセスはサービスグループごとに確保されるため，同じサービスグループに属していれば，異なるサービスに対しても一つのプロセスでサービスを実行できます。

通常の RPC と連鎖 RPC の比較を次の図に示します。

図 3-13　通常の RPC と連鎖 RPC の比較




連鎖 RPC の処理は，トランザクションとしてもトランザクションとしなくても実行できます。トランザクションとして連鎖 RPC を実行する場合は，一つのグローバルトランザクションとして処理されます。

連鎖 RPC は，クライアント UAP のプロセス単位で保証されます。同じグローバルトランザクション内でも，クライアント UAP が異なれば，複数回呼び出されたサービスが同じプロセスで起動されることは保証されません。

### (a)　連鎖 RPC の時間監視

連鎖 RPC の処理中に通信障害などで次のサービス要求を受け取れない場合，SPP がプロセスを確保したままになってしまうことがあります。これを防ぐため，連鎖 RPC で実行している SPP では，応答を返してから次のサービス要求，または連鎖 RPC の終了要求が来るまでの時間（最大時間間隔）を監視しています。連鎖 RPC 間隔監視時間は，ユーザサービス定義の watch_next_chain_time オペランドに指定します。この監視時間を過ぎても次のサービス要求，または連鎖 RPC の終了要求が来ない場合は，OpenTP1 はクライアント UAP で障害が起こったものと見なして，該当する SPP のプロセスを異常終了させます。

## (4)　リモートプロシジャコールの送信データの圧縮

LAN のネットワークの負荷を軽減するため，RPC でやり取りするデータを圧縮できます。データを圧縮する場合は，システム定義に次に示す値を指定します。

- TP1/Server Base の場合
  システム共通定義の rpc_datacomp オペランドに Y を指定します。

- TP1/Client の場合
  クライアント環境定義にデータの圧縮を指定します。

クライアント側のシステムがデータの圧縮を指定している場合は，rpc_datacomp オペランドの指定に関係なく，サーバの OpenTP1 が自動的に圧縮データを復元して処理します。その後再び圧縮してクライアントへ応答を返します。

送信データの圧縮でネットワークの負荷が軽減できても，ノード内のデータ圧縮と復元の処理が原因で通信時間が長くなってしまう場合があります。送信データの圧縮を指定するかどうかは，業務内容や通信形態に応じて判断してください。

## (5) 大規模な分散システムでのサービスの要求方法

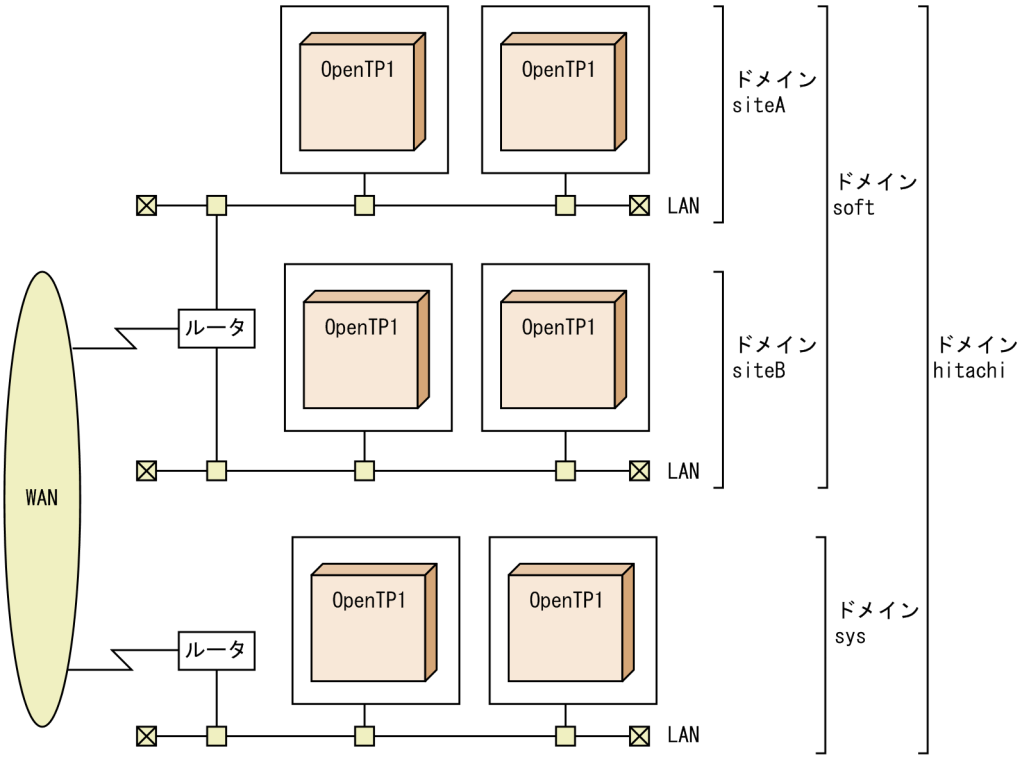
### (a) ドメインネームシステムの概要

WAN を介したネットワークシステムや，部門間を接続する全社システムなどの大規模な分散システムでは，名前の検索に時間が掛かり，システム名の管理に手間が掛かります。大規模な分散システムでリモートプロシジャコールを使う場合，ドメインネームシステムを構築することで，これらの問題を改善できます。

ドメインネームシステムを構築すると，システム全体を管理しないで，ドメインとして構成したグループごとに管理できます。ドメインは，小規模システムや，各部門のシステムに該当します。ドメインは，自ドメインだけを意識して管理するため，システム全体のサービスの名前などを管理する手間が省けます。

ドメインの概要を次の図に示します。

図 3-14 ドメインの概要



（説明）

ほかのドメインにあるサービスを利用するには，dc_rpc_call 関数の引数にドメイン名を付けます。

ドメイン sys にあるクライアントから，ドメイン名（siteA.soft.hitachi）を付けてサービスを要求すると,ドメイン hitachi，soft，siteA の順で問い合わせます。

## (b)　サービスの要求方法

サービスの要求方法を次に示します。

### 「サービスグループ名とサービス名」の指定

通常，サービスを要求するには，dc_rpc_call 関数で「サービスグループ名とサービス名」を指定します。

大規模な分散システムの場合，dc_rpc_call 関数での「サービスグループ名とサービス名」の指定では，ドメイン間のサービスを利用できません。

### 「サービスグループ名+ドメイン名とサービス名」の指定

大規模な分散システムでドメイン間のサービスを利用する場合，dc_rpc_call 関数で「**サービスグループ名+ドメイン名とサービス名**」を指定します。

dc_rpc_call 関数のサービスグループ名にこのドメイン名を付けると，ドメインの窓口のスケジュールサービスにサービス要求が渡ります。ドメインの窓口となるスケジュールサービスを，**ドメイン代表スケジュールサービス**と呼びます。ドメイン代表スケジュールサービスがドメインを構成するサーバのサーバ UAP にスケジュールします。

このため，指定したドメインにあるサーバ UAP にスケジュールされ，ほかのドメインにあるサーバ UAP にスケジュールされることはありません。また，ドメイン代表スケジュールサービスだけ意識するため，ネームサービスですべてのノードのネットワークアドレスを管理する必要がありません。

なお，サーバ UAP をソケット受信型サーバとした場合は，dc_rpc_call 関数のサービスグループ名にドメイン名は付けられません。ソケット受信型サーバについては，「3.4.1　SPP のスケジュール」を参照してください。

### サービス要求先の位置の指定

dc_rpc_call 関数が位置透過であるのに対して，サービスを要求するサーバの位置をユーザが認識して，dc_rpc_call_to 関数で特定のサービスを利用できます。サービス要求先の位置を指定する方法について次に示します。

- 「ホスト名称」指定

  ネットワーク管理上の/etc/hosts ファイル，または DNS などで IP アドレスとマッピングできるホスト名を指定することによって，利用するサービスが存在しているマシンを指定する方法です。

  指定したホスト内の OpenTP1 のシステム共通定義の name_port の指定値は，dc_rpc_call_to 関数を発行した OpenTP1 のシステム共通定義の name_port の指定値と同じであることが前提です。

- 「ノード識別子」指定

システム共通定義の node_id で指定したノード識別子を指定することによって，OpenTP1 システムを指定する方法です。

指定したノード識別子に対応するサービス要求先の OpenTP1 ノードのホスト名がグローバルドメイン内にあることが前提です。

ここでのグローバルドメインとは，次のノード名の集合を指します。

- システム共通定義の name_domain_file_use オペランドに N を指定している場合
  システム共通定義の all_node オペランド，all_node_ex オペランドで指定したノード名の集合です。
- システム共通定義の name_domain_file_use オペランドに Y を指定している場合
  ドメイン定義ファイルに指定したノード名の集合です。なお，ドメイン定義ファイルの格納場所は，次のとおりです。

  all_node のドメイン定義ファイル
  $DCCONFPATH/dcnamnd ディレクトリ下

  all_node_ex のドメイン定義ファイル
  $DCCONFPATH/dcnamndex ディレクトリ下

- 「ホスト名称＋ポート番号」指定
  次の値を指定することでサービス要求先を特定します。
  - /etc/hosts にファイル，または DNS などで IP アドレスとマッピングできるホスト名
  - 上記で指定したホストに存在する OpenTP1 システムの，システム共通定義の name_port に指定したネームサービスのポート番号

  このとき，サービス要求先の name_port オペランドに指定した値と，サービス要求元の name_port オペランドに指定した値は同じでなくてもかまいません。

なお，この機能は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できませんので，ご了承ください。

### (c) 大規模な分散システムでの環境設定

ドメインを構成するノードは，システム共通定義の all_node オペランドに指定したノードです。

ドメイン代表スケジュールサービスは，ドメイン代表スケジュールサービスがあるホスト名を namdomainsetup コマンドで登録します。そして，「/etc/services」に，ドメイン代表スケジュールサービスのポート番号と，サービス名「OpenTP1scd」を指定します。

ドメインネームシステムを使う場合，サーバシステムの起動時に各ノードのアドレスを問い合わせるため，システムの起動完了までに時間が掛かる場合があります。このような場合には，あらかじめシステム共通定義の domain_masters_addr オペランドにドメイン名とホスト名を，domain_masters_port オペランドにドメイン代表スケジュールサービスのポート番号を指定しておきます。この指定をしておくと，サーバ起動時に各ノードのアドレスを問い合わせる時間を削減できます。

### (6) ネームサービスを使わないサービスの要求方法

通常の RPC では，各ノードのネームサービスが管理している，サーバ UAP のサービス情報を使って通信しています。各ノードのネームサービスは，目的のサーバ UAP がネットワーク上のどのノードにあるかを，サービス情報を交換することで管理しています。多数のノードを接続した大規模システムでは，ネームサービスによるサービス情報のやり取りが，ネットワークに負荷をかける場合があります。

定義ファイルにサービス情報を定義しておくことで，ネームサービスを使わずに RPC ができます。ネームサービスへサービス情報を問い合わせないため，ネットワークに掛かる負荷を軽減できます。

ユーザは**ユーザサービスネットワーク定義ファイル**にサーバ UAP のサービスグループ名とサービス情報（ホスト名とポート番号）を定義します。また，dc_rpc_call 関数発行時にネームサービスを使うか，定義ファイルのサービス情報を使うかを，ユーザサービス定義の rpc_destination_mode オペランドで指定します。ユーザサービス定義で定義ファイルのサービス情報を使うと指定された UAP から dc_rpc_call 関数が呼び出されると，OpenTP1 はユーザサービスネットワーク定義ファイルを参照します。サーバ UAP の情報が定義されていれば定義情報を基に，定義されていなければネームサービスを使ってサーバ UAP を呼び出します。なお，ユーザサービスネットワーク定義に指定するサーバ UAP は，ユーザサービス定義で receive_from=queue が指定された UAP（キュー受信型サーバ）でなければいけません。

ユーザサービスネットワーク定義ファイルのサービス情報（ホスト名）を複数指定した場合，指定したサービス情報からランダムにサービス要求先が選択され，サービス要求が送信されます。サービス要求が成功すると，UAP 内で以降に呼び出す同じサービスグループ名への dc_rpc_call は，障害が発生するまで同じホストにサービス要求の送信を継続します。

なお，dcsvgdef 定義コマンドの-t オプションであて先再選択間隔に値を指定し，次に示す時間があて先再選択間隔に指定した時間を超えていた場合，再度ランダムにサービス要求先を選択しサービス要求を送信します。

- UAP 内で同じサービスグループ名への dc_rpc_call を呼び出し時に，サービス要求の送信を継続しているホストとの，初回から今回のサービス要求までの経過時間

## 3.2.2 サービス情報検索の付加機能

サービス情報検索の付加機能には，グローバル検索機能と，サービス情報優先度指定機能があります。これらの機能を使用すると，OpenTP1 システム内で水平分散を利用したサービス要求をしたり，特定のノードのサービス情報を優先的に使用したりできます。

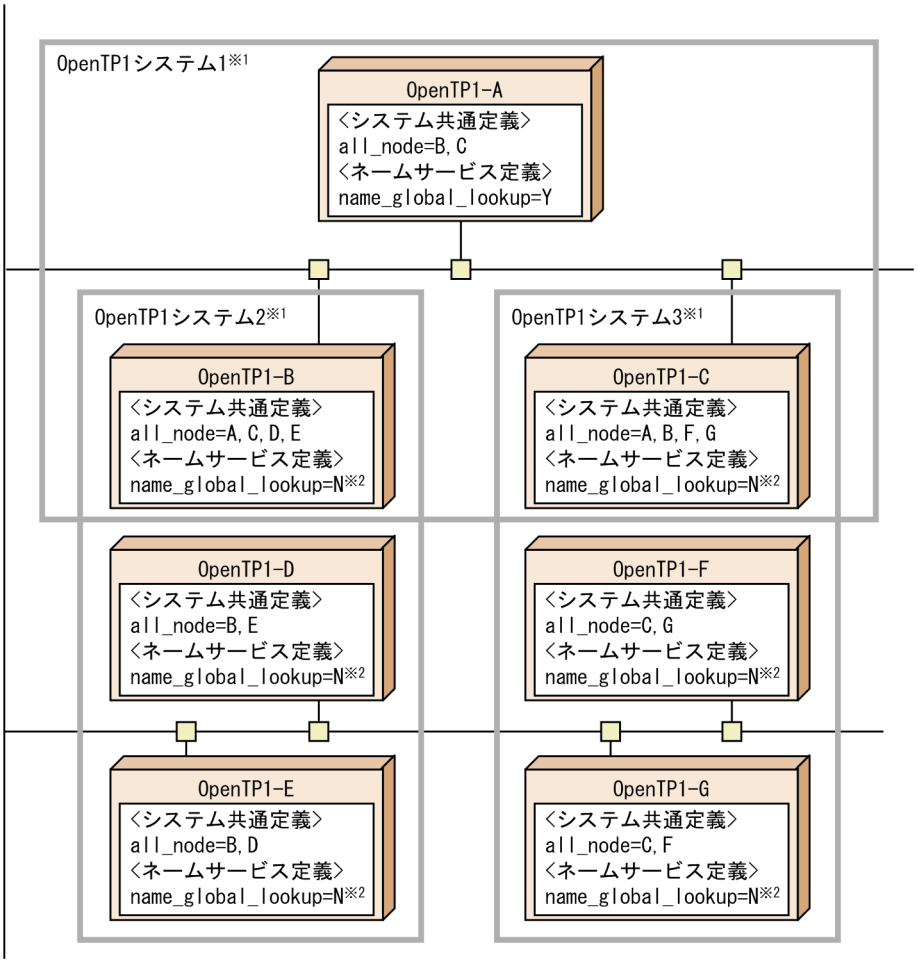
### (1) グローバル検索機能

**グローバル検索機能**を使用すると，次に示すサービス情報を取得できます。

1. all_node オペランドで指定された各ノードで起動するサービス情報
2. 1.のノード上のネームサービスが管理する，all_node オペランドで起動するサービス情報

グローバル検索機能使用時のシステム構成例を次の図に示します。各 OpenTP1 ノードのネームサービス定義の name_global_lookup オペランドを次の図のように指定してください。name_global_lookup オペランドの指定方法の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

図 3-15　グローバル検索機能使用時のシステム構成




注　　OpenTP1-B～OpenTP1-Gのバージョンは，03-02以降でなければなりません。
注※1　OpenTP1システムとは，各OpenTP1ノードのall_nodeオペランドで指定されているOpenTP1ノード群を指します。
注※2　name_global_lookupオペランドでNを指定したOpenTP1ノード，またはname_global_lookupオペランドを未サポートのOpenTP1ノードが有効です。

OpenTP1-A からの検索範囲は，OpenTP1-B および C の all_node オペランドで指定した OpenTP1 ノードまでとなります。つまり，OpenTP1 システム 1 だけでなく，OpenTP1 システム 2 および 3 までのサービスと通信できるようになります。なお，OpenTP1-D，E，F および G の all_node オペランドで指定したノードは検索対象となりません。

グローバル検索機能を使用した環境では，DCRPC_BINDTBL_SET 関数の引数 portno にネームサービスのポート番号を指定した，dc_rpc_call_to 関数は使用できません。

DCRPC_BINDTBL_SET 関数の引数 nid にノード識別子を指定した，dc_rpc_call_to 関数を使用する場合は，必ずグローバル検索機能の検索範囲（図の OpenTP1 システム 1～3）内のすべての OpenTP1 ノードのノード識別子を一意に定義してください。

サービス情報（閉塞，負荷状態など）は，検索元 OpenTP1（図の OpenTP1-A）まで通知されないため，OpenTP1 システム 2 内の OpenTP1-D と E の間，および OpenTP1 システム 3 内の OpenTP1-F と G の間のように，OpenTP1 システム内で相互に all_node オペランドを指定し合い，水平分散を利用したサービス要求を行うことをお勧めします。

ネームサービス定義の name_cache_size オペランドの指定値は，自ノードから検索を要求するサービス情報の数に，all_node オペランドで指定されたノードにキャッシュされるサービス情報の数まで含めて算出する必要があります。

グローバル検索機能を使用した OpenTP1（図の OpenTP1-A）に対し，TP1/Client/P，TP1/Client/W または TP1/Client/J からサービス要求が送信された場合，OpenTP1 システム 1 だけでなく，OpenTP1 システム 2 および 3 のサービス情報まで取得します。

## (2) サービス情報優先度指定機能

**サービス情報優先度指定機能**とは，ネームサービスがサービス要求元のクライアント UAP にサービス情報を返すときに，特定のノードのサービス情報を優先的に返す機能です。サービス情報を優先的に返すノードを，**優先選択ノード**と呼びます。

通常の RPC を実行する場合，高負荷などの状態を除き，クライアント UAP のサービス要求を処理するサーバ UAP が複数ある場合には，それらを同じ優先度で扱います。このため，システム共通定義の all_node オペランドに指定したすべてのノードのネームサービスに対してサービス情報の検索要求を送信し，応答によって取得したすべてのサービス情報をクライアント UAP に返します。クライアント UAP は，返ってきたサービス情報の中から RPC のあて先となるサーバ UAP をランダムに選び出し，RPC を実行します。

これに対して，サービス情報優先度指定機能を使用すると，クライアント UAP のサービス要求を処理するサーバ UAP が複数ある場合には，優先選択ノードのサービス情報を優先してクライアント UAP に返します。クライアント UAP は，優先選択ノードのサーバ UAP に対して RPC を実行します。また，優先選択ノードでサーバ UAP の閉塞などの障害が発生した場合には，優先選択ノード以外のノードのサービス情報をクライアント UAP に返します。このように，サービス情報優先度指定機能を使用すると，通常は優先選択ノードのサーバ UAP を使用し，障害が発生した場合だけ優先選択ノード以外のノードのサーバ UAP を使用できるため，サーバ UAP を実行系と待機系に分けて扱うことができます。

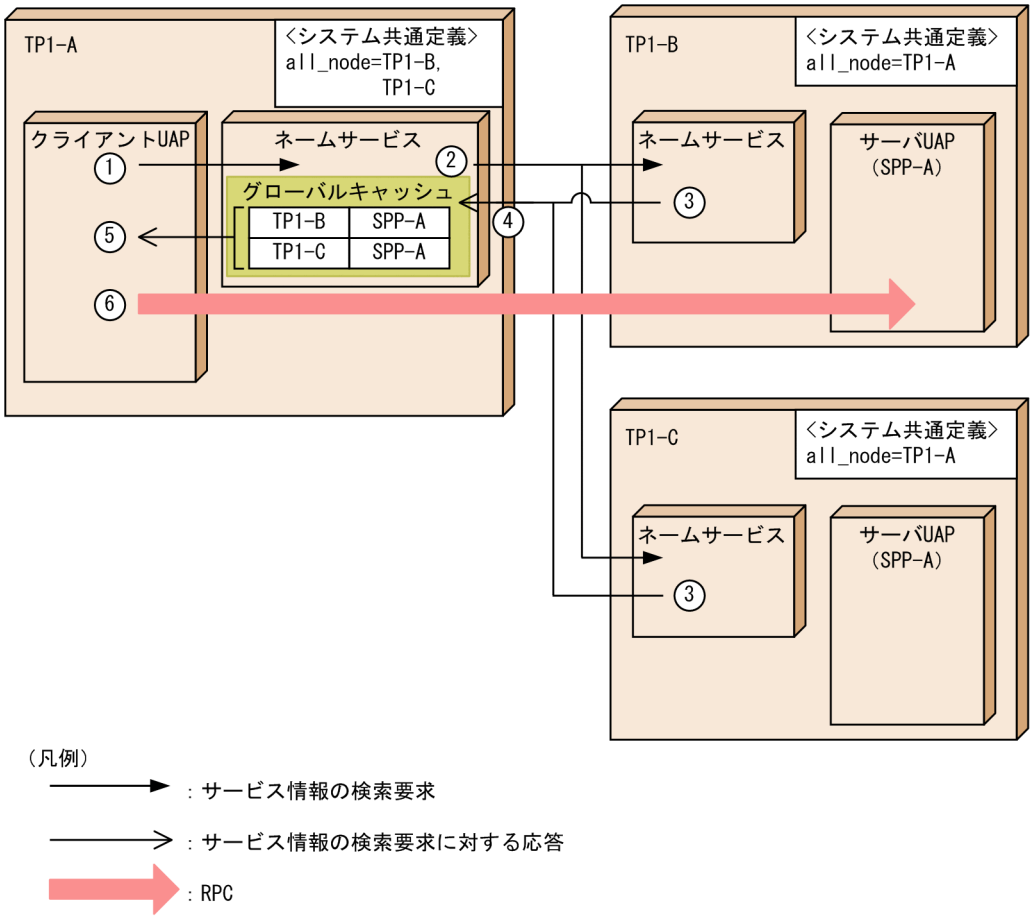
優先選択ノードは，システム共通定義の all_node オペランドで指定します。優先選択ノードの定義ファイルの指定方法は，ドメイン定義ファイルの指定と同じです。優先選択ノードの指定方法については，マニュアル「OpenTP1 システム定義」を参照してください。

以降，通常の RPC の流れと，サービス情報優先度指定機能を使用した場合の RPC の流れの違いを説明します。なお，以降の説明のグローバルキャッシュとは，ネームサービスが，他ノードで起動しているサーバのサービス情報を管理する領域を示します。

- **通常の RPC の流れ**

  通常の RPC の流れを次の図に示します。

図 3-16　通常の RPC の流れ




図で示した RPC の流れについて説明します。番号は図中の番号と対応しています。

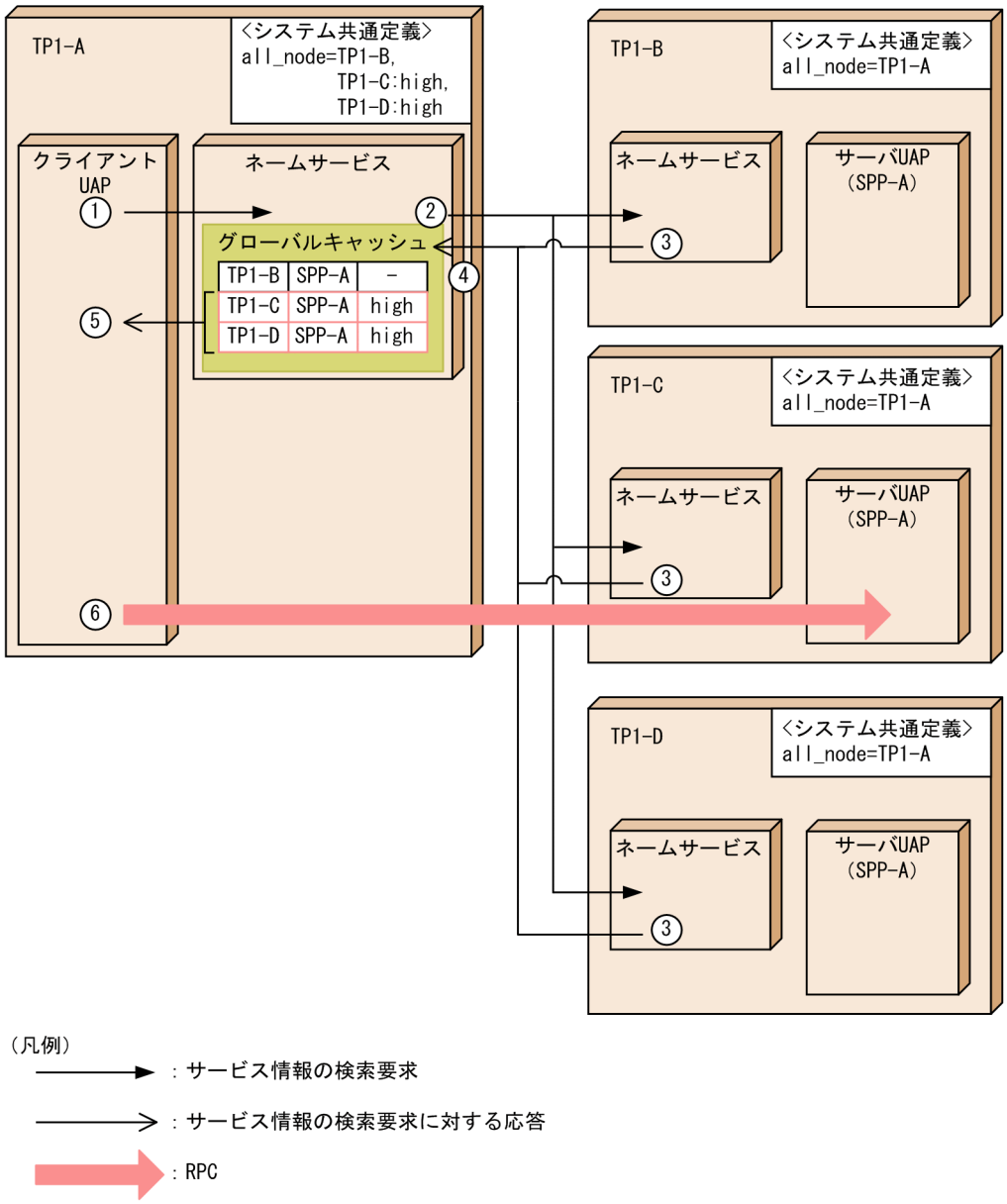
1. RPC を実行するクライアント UAP が，TP1-A のネームサービスへサービス情報の検索要求をします。
2. ネームサービスは，システム共通定義の all_node オペランドで指定された TP1-B，および TP1-C のネームサービスにサービス情報の検索要求を送信します。
3. 検索要求を受けた TP1-B，および TP1-C のネームサービスは，自ノードで稼働している SPP-A のサービス情報を要求元の TP1-A へ送信します。
4. TP1-A のネームサービスは，受信したサービス情報をグローバルキャッシュに登録します。
5. ネームサービスは，グローバルキャッシュに登録されたすべてのサービス情報をクライアント UAP に返します。

   クライアント UAP は，RPC のあて先をランダムに選びます。ここでは，TP1-B を RPC のあて先に決定したものとします。
6. クライアント UAP は，TP1-B に RPC を実行します。

- **サービス情報優先度指定機能を使用した場合の RPC の流れ**

  サービス情報優先度指定機能を使用した場合の RPC の流れを次の図に示します。ここでは，TP1-C および TP1-D を優先選択ノードに指定しています。

図 3-17　サービス情報優先度指定機能を使用した場合の RPC の流れ




図で示した RPC の流れについて説明します。番号は図中の番号と対応しています。

1. RPC を実行するクライアント UAP が，TP1-A のネームサービスへサービス情報の検索要求を送信します。
2. ネームサービスは，システム共通定義の all_node オペランドで指定された TP1-B，TP1-C，および TP1-D のネームサービスにサービス情報の検索要求を送信します。
3. 検索要求を受けた TP1-B，TP1-C，および TP1-D のネームサービスは，自ノードで稼働している SPP-A のサービス情報を要求元の TP1-A へ送信します。
4. TP1-A のネームサービスは，受信したサービス情報をグローバルキャッシュに登録します。
5. TP1-C および TP1-D を優先選択ノードに指定しているため，ネームサービスは，TP1-C および TP1-D のサービス情報だけをクライアント UAP に返します。

ここでは，クライアント UAP は返されたサービス情報の中から TP1-C を RPC のあて先に決定したものとします。

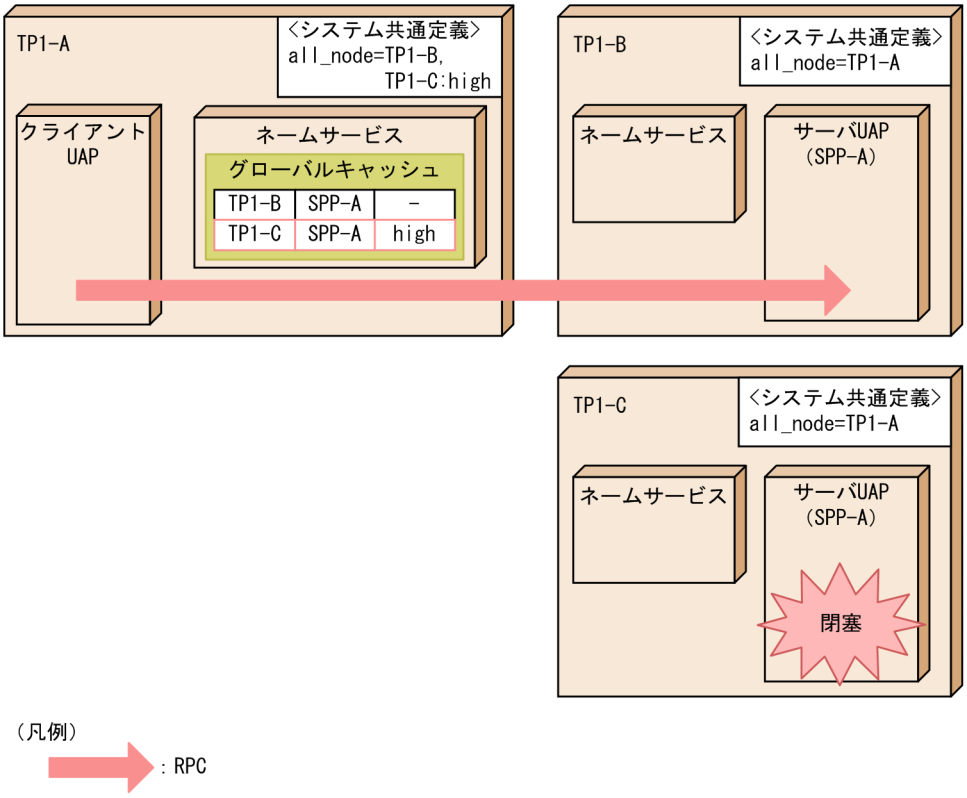
6. クライアント UAP は，TP1-C に RPC を実行します。

優先選択ノードのサーバ UAP が閉塞していたり，グローバル検索機能を使用したりする場合は，上の図で示したサービス情報優先度指定機能を使用した場合の RPC の流れとは異なります。RPC の流れが異なる場合について，(a)～(c)で説明します。なお，以降の説明では，ネームサービスの処理を省略しています。

## (a)　サーバ UAP が閉塞している場合の RPC の流れ

優先選択ノードのサーバ UAP が閉塞している場合の RPC の流れを次の図に示します。

図 3-18　サーバ UAP が閉塞している場合の RPC の流れ




TP1-A は，TP1-C を優先選択ノードとして指定していますが，TP1-C のサーバ UAP は閉塞しています。TP1-C のサーバ UAP の閉塞が TP1-A に通知されると，TP1-A は TP1-C を RPC のあて先の選択候補から除きます。そのため，TP1-A は，優先選択ノードに指定していない TP1-B のサーバ UAP に RPC を実行します。

### 注意事項

TP1-C のサーバ UAP の閉塞情報が TP1-A に通知されていない間は，TP1-A は TP1-C のサーバ UAP へ RPC を実行します。このとき，TP1-C のシステム共通定義の all_node オペランドに TP1-B を指定していると，TP1-C から TP1-B へ RPC が転送されることがあります。一方，TP1-C の

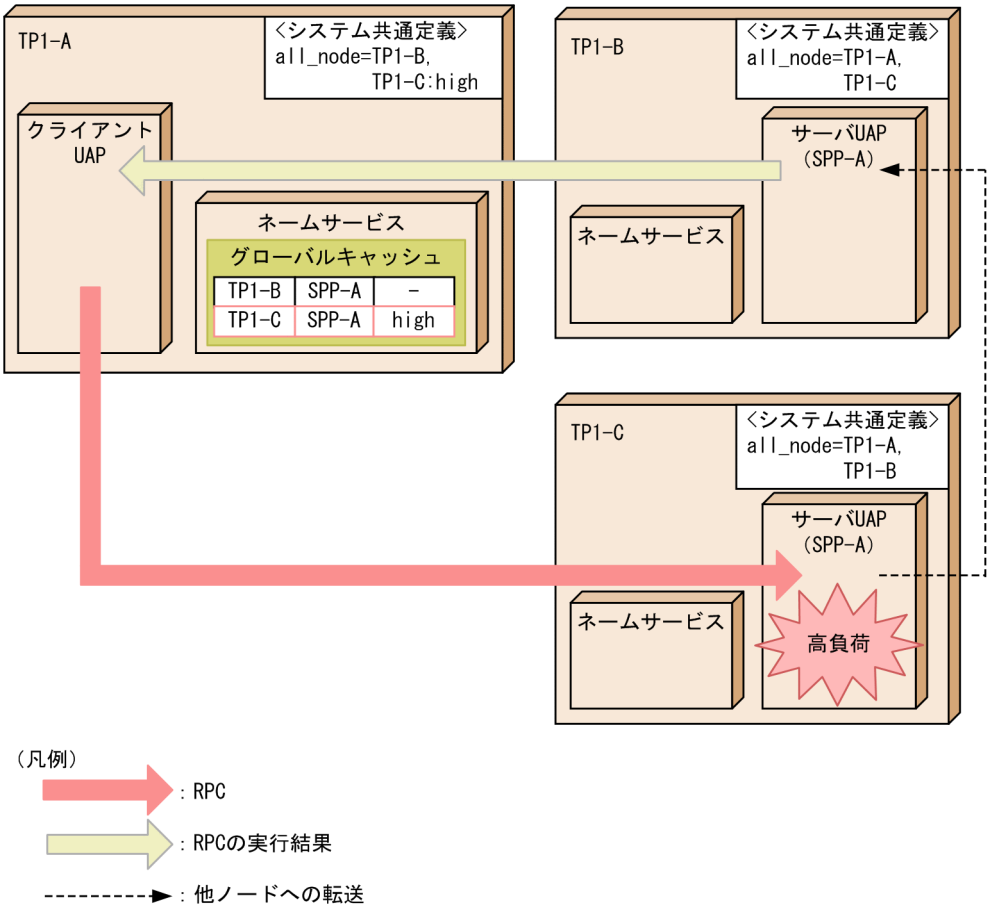
システム共通定義の all_node オペランドに TP1-B を指定していないときは，RPC はエラーリターンします。

また，閉塞情報はサーバ単位で通知されます。そのため，TP1-C のサーバ UAP がサービス単位で閉塞した場合，閉塞情報は TP1-A に通知されません。したがって，TP1-A は TP1-C のサーバ UAP に RPC を実行します。この要求先のサービスが閉塞していた場合，RPC はエラーリターンします。

### (b)　サーバ UAP が高負荷状態になっている場合の RPC の流れ

優先選択ノードのサーバ UAP が高負荷状態になっている場合の RPC の流れを次の図に示します。

図 3-19　サーバ UAP が高負荷状態になっている場合の RPC の流れ




TP1-A は優先選択ノードに TP1-C を指定しているため，TP1-C で稼働しているサーバ UAP が RPC のあて先の選択候補になります。しかし，TP1-C のサーバ UAP が高負荷状態であるため，TP1-C のシステム共通定義の all_node オペランドに指定したノードのサーバ UAP に RPC を転送します。ここでは，TP1-C から TP1-B のサーバ UAP へ RPC が転送され，TP1-B のサーバ UAP が TP1-A の RPC を実行します。

### (c)　グローバル検索機能とサービス情報優先度指定機能を併用した場合の RPC の流れ

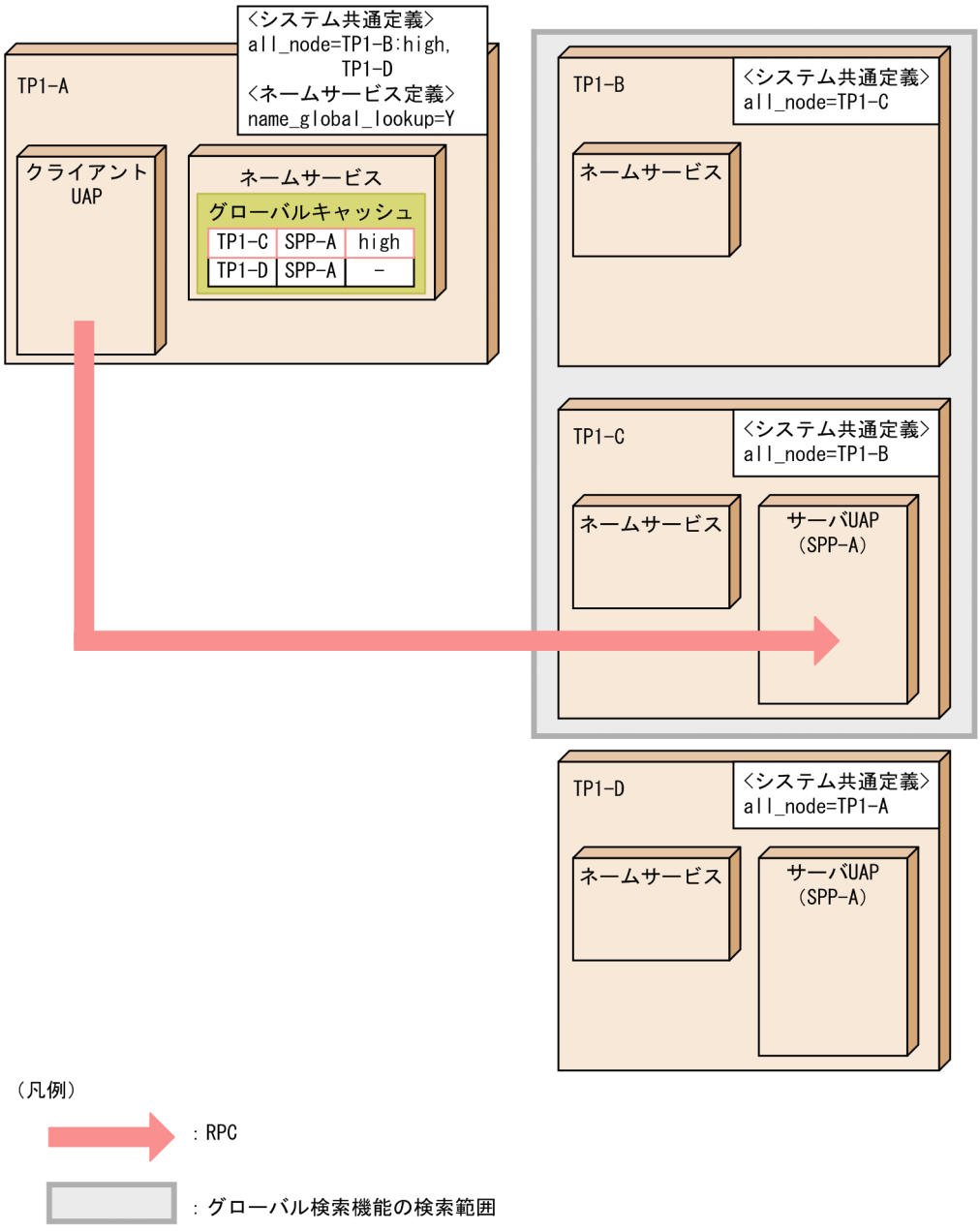
グローバル検索機能とサービス情報優先度指定機能は，併用できます。この二つの機能を併用した場合，グローバル検索中継ノードを優先選択ノードに指定しているかどうかで，RPC の流れが異なります。

グローバル検索中継ノードを優先選択ノードに指定している場合は，グローバル検索機能の検索範囲の中で指定された優先選択ノードが RPC のあて先になります。グローバル検索中継ノードを優先選択ノードに指定していない場合は，要求元で指定した優先選択ノードが RPC のあて先になります。それぞれの場合について説明します。

- グローバル検索中継ノードを優先選択ノードに指定した場合の RPC の流れ

  グローバル検索中継ノードを優先選択ノードに指定した場合の RPC の流れを次の図に示します。

図 3-20　グローバル検索中継ノードを優先選択ノードに指定した場合の RPC の流れ



グローバル検索機能を使用しているため，TP1-A は，TP1-B を経由して，TP1-B のシステム共通定義の all_node オペランドに指定されている TP1-C のサービス情報を取得します。

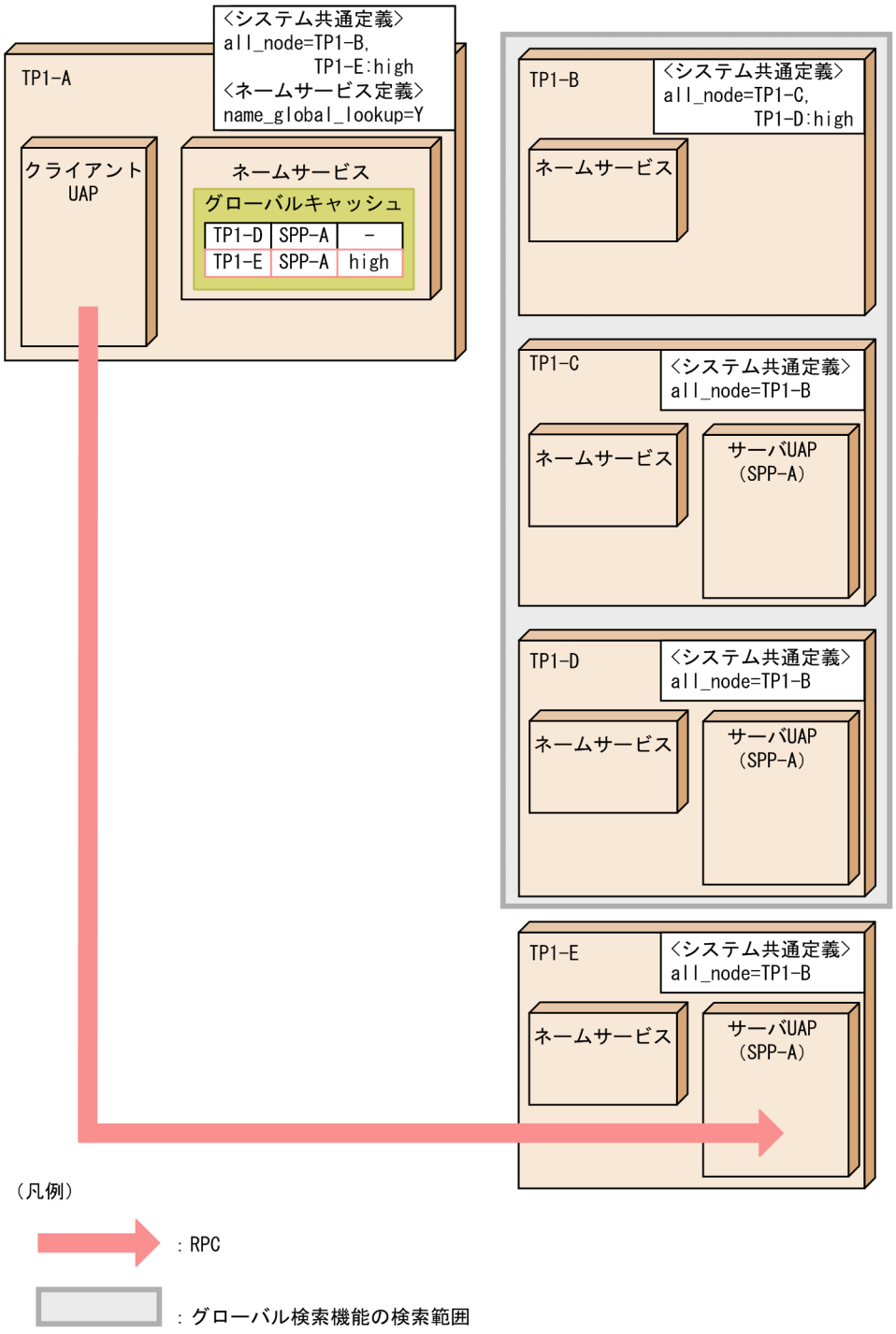
TP1-A は優先選択ノードに TP1-B を指定しているため，TP1-B の all_node オペランドに指定されている TP1-C のサービス情報が，優先選択ノードで稼働しているものとして扱われます。この結果，TP1-A は，TP1-C のサービス情報を優先的に選択します。

つまり，サービス要求元のノード（TP1-A）での優先度に関係なく，優先選択ノードに指定したグローバル検索中継ノード（TP1-B）を経由して取得したサービス情報（TP1-C のサービス情報）が，優先選択ノードで稼働しているものとして扱われて，優先的に選択されます。

- **グローバル検索中継ノードを優先選択ノードに指定していない場合の RPC の流れ**

  グローバル検索中継ノードを優先選択ノードに指定していない場合の RPC の流れを次の図に示します。

図 3-21　グローバル検索中継ノードを優先選択ノードに指定していない場合の RPC の流れ




グローバル検索機能を使用しているため，TP1-A は，TP1-B を経由して，TP1-B のシステム共通定義の all_node オペランドに指定されている TP1-C および TP1-D のサービス情報を取得しようとします。このとき，TP1-B は，TP1-B の優先選択ノードである TP1-D で稼働しているサービス情報を選択して，TP1-A に返します。

この結果，TP1-A は，優先選択ノードである TP1-E を選択します。

つまり，サービス要求元のノード（TP1-A）での優先度が有効となります。グローバル検索中継ノード（TP1-B）での優先度に関係なく，優先選択ノードではないグローバル検索中継ノード（TP1-B）を経由して取得したサービス情報（TP1-D のサービス情報）は，選択されません。

### (d)　サービス情報優先度指定機能を使用する場合の注意事項

サービス情報優先度指定機能を使用する場合の注意事項を次に示します。

- 優先選択ノードに一つ以上のノードを指定し，かつ RPC を実行するノードでも RPC 先となるサーバ UAP が稼働している場合は，RPC を実行するノードも優先選択ノードになります。
- dc_rpc_call_to 関数を実行した場合，サービス情報優先度指定機能は無効になります。そのため，指定されたホスト名称が示すノードで稼働しているサーバ UAP へ RPC を実行します。
- 優先選択ノードで稼働しているサーバ UAP がすべて閉塞していた場合，そのノードは優先選択ノードから除きます。そのため，優先選択ノードに指定していないノードで稼働しているサーバ UAP へ RPC を実行します。
- 優先選択ノードで稼働しているサーバ UAP が高負荷状態の場合，優先選択ノード以外のノードで稼働している同一サービスグループに RPC が転送されてしまうことがあります。そのため，優先選択ノードに指定していないノードで稼働しているサーバ UAP で RPC が実行されることがあります。

  高負荷状態の詳細については，「3.4.3(6)　ノード間負荷バランス機能」を参照してください。

## 3.2.3　OpenTP1 のノード管理

OpenTP1 同士の通信は TCP/IP によって実装した RPC を使用します。TCP/IP では，通信するサーバ間でコネクションを確立した状態で通信します。

ネットワーク障害によって OpenTP1 との通信ができなくなった場合，コネクション障害を検知できません。このため，そのあとに実行した RPC が障害となることがあります。このような状態を防止するための OpenTP1 の機能について説明します。

### (1)　起動通知機能

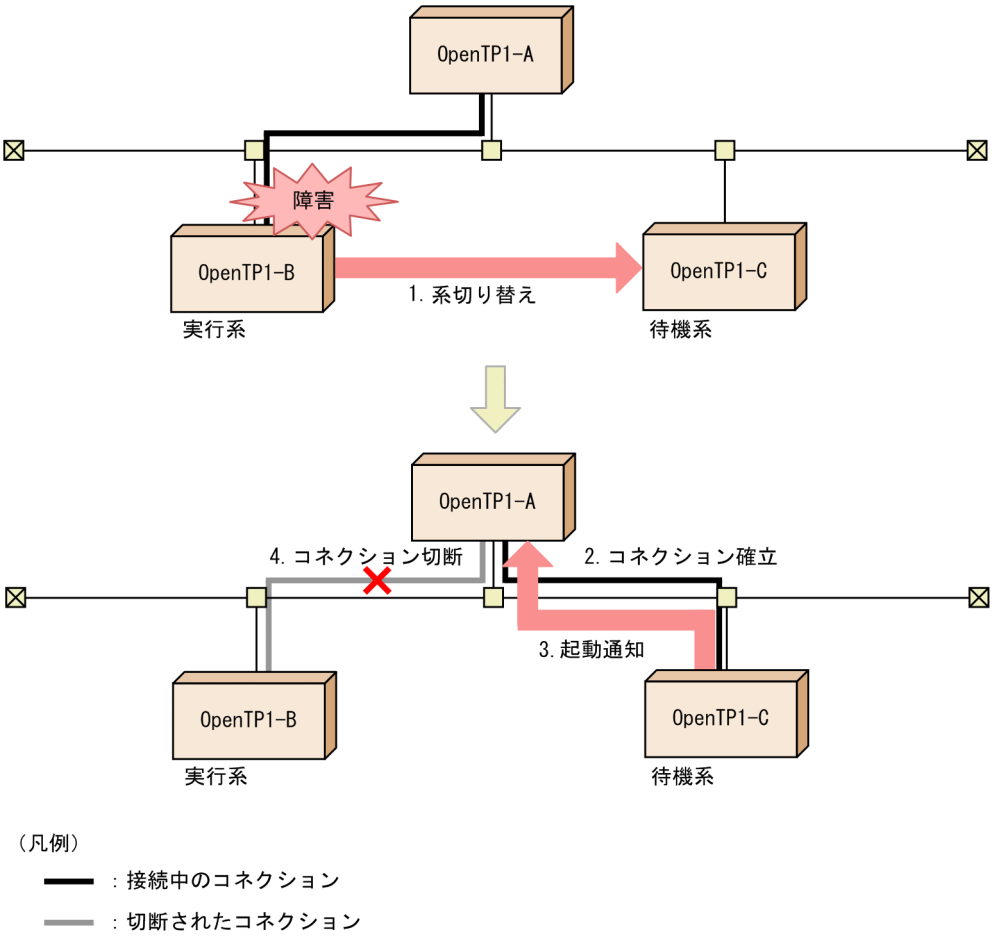
OpenTP1 の起動時に，他ノードで起動する OpenTP1 のネームサービスに対して，自ノードが起動したことを通知し，それまで接続されていたコネクションを強制的に切断します。この機能は，系切り替えをする場合などに使用します。

OpenTP1 の起動を通知する場合，送信側および受信側の両方のノードでシステム共通定義の name_notify オペランドに Y を指定してください。

この機能を使用するには，送信側および受信側の両方のノードで OpenTP1 のバージョンを 05-02 以降にする必要があります。

起動通知機能を系切り替え時に使用した場合の構成例を次の図に示します。

図 3-22　起動通知機能を系切り替え時に使用した場合




1. OpenTP1-B がサーバ障害などでダウンすると系切り替えが発生します。OpenTP1-A では，OpenTP1-B からの障害を検知できなくてコネクションは残留します。
2. 系切り替えが行われ，待機系で OpenTP1-C が開始されます。
3. 起動通知機能を使用した場合，OpenTP1-C では起動通知を OpenTP1-A に通知します。
4. OpenTP1-A では，OpenTP1-B へのコネクションを強制切断します。

このように，OpenTP1-A，OpenTP1-B，OpenTP1-C との通信はコネクション確立から再開するため，通信障害を発生させることなく処理できます。

何らかの要因で OpenTP1-A に起動通知できなかった場合，OpenTP1-C で KFCA00642-W メッセージが出力されます。この場合，OpenTP1-A で namunavl コマンドを実行する必要があります。OpenTP1-C で起動通知を通知できなかったノードは，namunavl コマンドに-l オプションを指定して実行することで確認できます。

**注意事項**

この機能は，ノードごとにユニークな IP アドレスが割り振られていることを前提としているため，同一の IP アドレスで複数の OpenTP1 が起動する（1LAN ボードだけ使用時など）システムでは，起動通知機能を使用できません。

## (2)　ノード監視機能

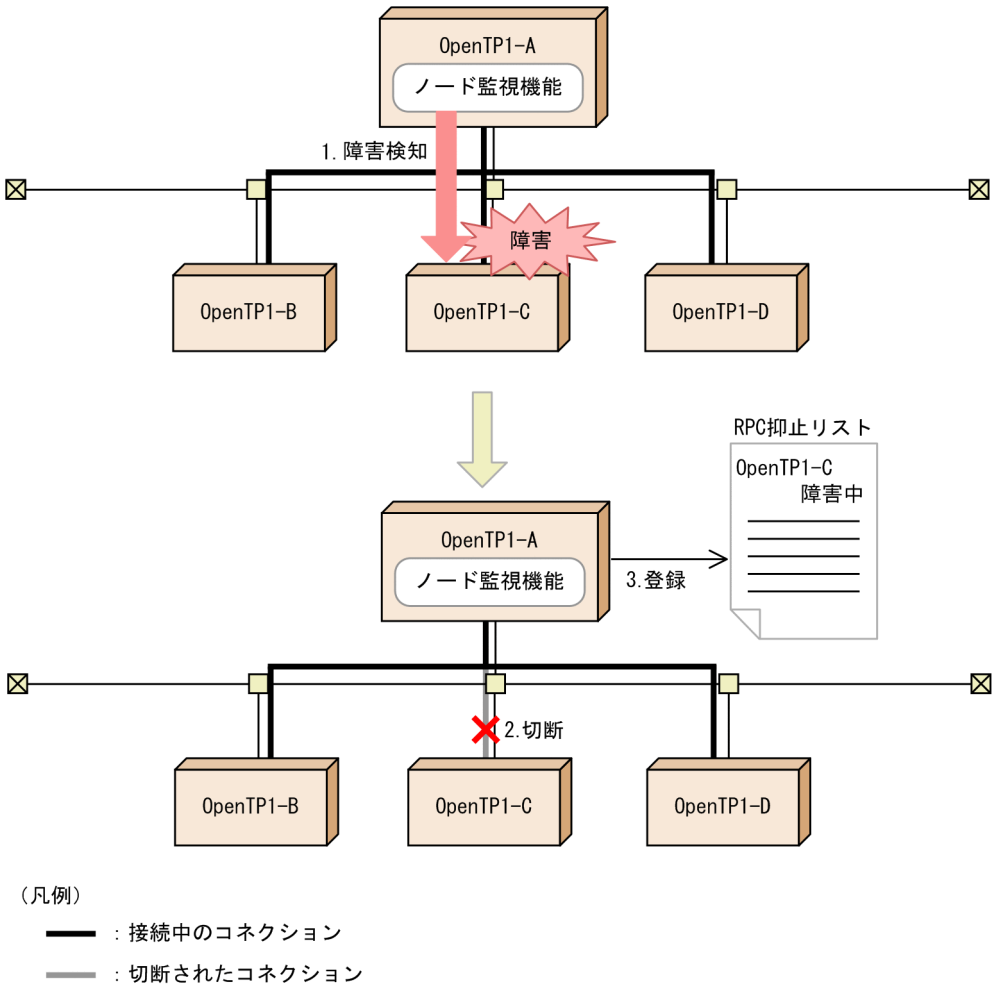
定期的にノードとの通信を行い，障害を検知します。

ノード監視機能を使用すると，システム共通定義の all_node オペランドおよび all_node_ex オペランドに指定されたノードの起動状況を監視できます。この機能では，起動を検出できなかった OpenTP1 ノードのすべてのサービス情報をキャッシュから削除し，さらに，そのノードとのコネクションを切断します。

障害を検知し，障害ノードと接続されているコネクションを強制切断できるため，障害を最小にできます。

ノード監視機能で，他ノードを監視する例を次の図に示します。

図 3-23　ノード監視機能を使用した他ノードの監視



ノード監視機能を使用すると，OpenTP1-A から OpenTP1-B，OpenTP1-C，OpenTP1-D を定期的に監視します。

1. OpenTP1-C でノードダウンが発生すると，ノード監視機能が OpenTP1-C との通信障害を検知します。
2. OpenTP1-C のコネクションを切断し，KFCA00650-I を出力します。
3. RPC 抑止リスト※に登録します。キャッシュに登録しているサービス情報のうち，未起動を検知したノードのサービス情報を削除します。

注※

RPC 抑止リストは，OpenTP1 システムが未起動の OpenTP1 ノードの情報を保持しているリストです。

ノード監視機能は，ネームサービス定義の name_audit_interval オペランドに指定された時間ごとに，各ノードの起動状況を確認します。ノード監視機能を使用する場合は，ネームサービス定義の name_audit_conf オペランドに 1，または 2 を指定してください。

name_audit_conf オペランドの指定値ごとにノード監視機能の動作を次に示します。

- name_audit_conf に 1 を指定した場合

  一方送信型ノード監視を行います。送信処理が終了した時点で次の動作をします。

  - 稼働状態だったノードとの送信処理が失敗したとき

    送信処理に失敗したノードが停止状態になったと判断し，KFCA00650-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストに登録します。

  - 停止状態だったノードとの送信処理が成功したとき

    停止状態だったノードが稼働状態になったと判断し，KFCA00651-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストから削除します。

- name_audit_conf に 2 を指定した場合

  送受信型ノード監視を行います。送受信処理が終了した時点で次の動作をします。

  - 稼働状態だったノードとの送信処理が失敗，または，稼働状態だったノードからの受信処理に失敗※したとき

    送受信処理に失敗したノードが停止状態になったと判断し，KFCA00650-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストに登録します。

  - 停止状態だったノードとの送受信処理が成功したとき

    停止状態だったノードが稼働状態になったと判断し，KFCA00651-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストから削除します。

  注※

  受信処理で受信タイムアウトが発生した場合も，失敗したと認識します。タイムアウト監視時間は，ネームサービス定義の name_audit_watch_time オペランドに指定した値です。

ノード監視機能を使用した場合のノードの起動状況によって次の動作をします。

- 稼働状態だったノードとの通信が失敗したとき

  通信に失敗したノードが停止状態になったと判断し，KFCA00650-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストへ登録します。

- 停止状態だったノードとの通信が成功したとき

  停止状態だったノードが稼働状態になったと判断し，KFCA00651-I メッセージを標準出力へ出力したあと，該当するノードの情報を RPC 抑止リストから削除します。

ノードの起動状況は，namalivechk コマンドでも確認できます。ノード監視機能を使用した場合と namalivechk コマンドを使用した場合のノード監視の違いについて，次の表に示します。

表 3-4　ノード監視機能使用時と namalivechk コマンド使用時のノード監視の比較

| 比較項目 | ノード監視機能使用時 | namalivechk コマンド使用時 |
|---|---|---|
| 監視対象 | • システム共通定義の all_node オペランドに指定されたすべてのノード（起動／未起動にかかわらずすべてのノード）<br>• システム共通定義の all_node_ex オペランドに指定されたすべてのノード（起動／未起動にかかわらずすべてのノード） | • システム共通定義の all_node オペランドに指定されたノードのうち，OpenTP1 が未起動を検知していないノード<br>• システム共通定義の all_node_ex オペランドに指定されたすべてのノード（起動／未起動にかかわらずすべてのノード） |
| ノードの未起動を検知した場合の動作 | • all_node オペランドに指定されたノードで，かつ RPC 抑止リストに登録されていないノードの場合，ノードの情報を RPC 抑止リストへ登録します。すでに RPC 抑止リストに登録されているノードの場合，何もしません。<br>• 未起動を検知したノードとのコネクションを切断します。<br>• キャッシュに登録しているサービス情報のうち，未起動を検知したノードのサービス情報を削除します。 | • all_node オペランドに指定されたノードのうち，起動を検出できなかったノードの情報を RPC 抑止リストへ登録します。<br>• 未起動を検知したノードとのコネクションを切断します。<br>• キャッシュに登録しているサービス情報のうち，未起動を検知したノードのサービス情報を削除します。 |
| ノードの起動を検知した場合の動作 | all_node オペランドに指定されたノードで，かつ RPC 抑止リストに登録されているノードの場合，RPC 抑止リストからノードの情報を削除します。 | 何もしません。 |

### 注意事項

- この機能は，ノードごとにユニークな IP アドレスが割り振られていることを前提としているため，同じ IP アドレスを使用するノードをシステム共通定義の all_node オペランドまたは all_node_ex オペランドに複数指定しているノードや，同一の IP アドレスで複数の OpenTP1 が起動する（1LAN ボードだけ使用時など）ノードに対しては，ノード監視機能を使用できません。
- ノード監視機能の監視用通信処理で，ノードダウン検知の感度をチューニングする場合，次のオペランドを変更してください。

  name_audit_conf オペランドに 1 を指定した場合

システム共通定義の ipc_conn_interval オペランドを変更してください。

**name_audit_conf オペランドに 2 を指定した場合**

ネームサービス定義の name_audit_watch_time オペランドを変更してください。

- ノード監視機能で同時に監視できるノード数は 60 ノードまでです。システム共通定義の all_node オペランド，および all_node_ex オペランドに指定したノード数が 60 を超える場合，60 ノード単位で監視を繰り返します。
- システム共通定義の all_node オペランド，および all_node_ex オペランドに多くのノードを指定している場合，ノード監視機能を使用すると UAP で実行する RPC に影響を及ぼすおそれがあります。この場合，name_audit_interval オペランドに小さな値を指定しないでください。また，namalivechk コマンドを繰り返し実行する場合，その間隔を短くしないでください。

## (3) RPC 抑止リストに登録されたノードの監視機能

ネームサービスでは，ノード監視機能とは別に 180 秒ごとに RPC 抑止リストに登録されたノードの起動状況を確認できます。この機能の使用有無は，ネームサービス定義の name_rpc_control_list オペランドで指定します。

ノードの監視機能の設定を考慮して，この機能を使用してください。例えば，次に示す状態の場合，RPC 抑止リストに登録されたノードの監視機能を無効にする必要があります。

- name_audit_interval オペランドに指定した時間が経過していなくても，障害から復旧したノードが RPC 抑止リストから削除されることがあります。この場合，KFCA00651-I メッセージは出力されません。
- name_audit_conf オペランドに 2 を指定している場合，KFCA00650-I メッセージが定期的に出力されることがあります。

RPC 抑止リストに登録されたノードの監視機能を無効にした場合，name_audit_interval オペランドに指定した時間が 180 秒以上のとき，障害から復旧したノードが RPC 抑止リストから削除されるまでの時間が従来よりも長くなります。

ノード監視機能と RPC 抑止リストに登録されたノードの監視機能について，推奨する設定を次に示します。

- name_audit_conf オペランドに 1 または 2 を指定し，かつ name_audit_interval オペランドに 180 以下を指定した場合

  name_rpc_control_list オペランドに N を指定することをお勧めします。
- name_audit_conf オペランドを省略するか，または 0 を指定した場合

  name_rpc_control_list オペランドを省略するか，Y を指定することをお勧めします。

  なお，name_audit_conf オペランドを省略するか，または 0 を指定した場合，name_rpc_control_list オペランドに N を指定したときは，ノード監視機能および RPC 抑止リストへの監視機能は無効になるので次に示す状態になります。

- RPC 抑止リストに登録したノードから自ノードに対して通信が発生しないかぎり，RPC 抑止リストから該当ノードは削除されません。
- RPC 抑止リストに登録したノードの all_node オペランドに自ノードを指定していない場合，自ノードの OpenTP1 を再起動しないかぎり，RPC 抑止リストから該当ノードが削除されません。

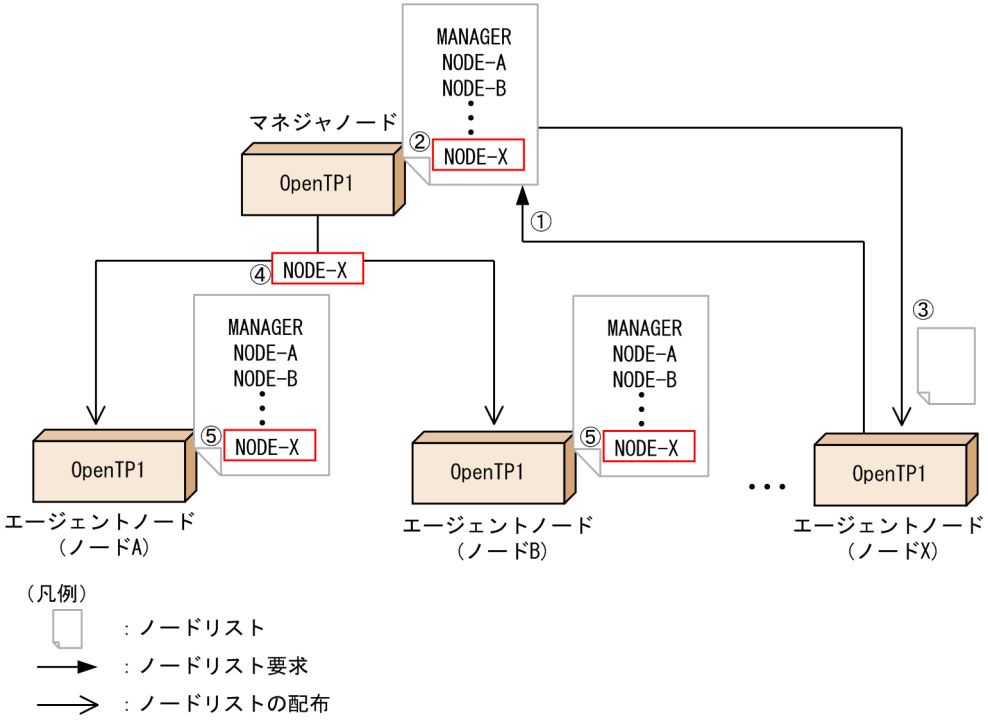
## (4) ノード自動追加機能

**ノード自動追加機能**とは，OpenTP1 システムにノードを追加する手順を容易にするための機能です。ノード自動追加機能では，ノード情報を管理するノードを定義します。このノードのことを**マネジャノード**といいます。逆にマネジャノードに管理されるノードのことを**エージェントノード**といいます。

マネジャノードが管理するノードの情報のリストを**ノードリスト**といいます。ノードリストは，マネジャノードからエージェントノードへ配布され，OpenTP1 システム内でノードリストを共有します。

ノード自動追加機能を使用してノードを追加する場合のノードリストの流れを次の図に示します。

図 3-24　ノード自動追加機能を使用してノードを追加する場合のノードリストの流れ




図で示したノードリストの流れについて説明します。番号は図中の番号と対応しています。

1. 追加するノード（ノード X）からマネジャノードに対して，ノードリスト要求を送信します。
2. マネジャノードは，ノード X のノード情報をノードリストに追加します。
3. マネジャノードからノード X にノードリストを返します。
4. マネジャノードから各ノードへノード X のノード情報を配布します。
5. 各ノードで，ノード X をノードリストに追加します。

### (a) ノード自動追加機能を使用するメリット

ノード自動追加機能を使用すると，次のメリットがあります。

- ノード追加時の作業手順の軽減

  ノード自動追加機能を使用すれば，ノード追加時に修正する定義は，追加ノード分だけです。また，そのほかのノードを再起動する必要もありません。

- 大規模構成への対応

  ノード追加時の定義の修正作業が緩和されることによって，大規模システム構成への対応が容易になります。マネジャノードは最大 511 のノードを管理できます。

### (b) ノードの追加手順

ノード自動追加機能を使用する場合と，使用しない場合のノードの追加手順の違いを次に示します。

●ノード自動追加機能を使用した場合のノードの追加手順

追加するノードの作業

1. システム共通定義の次のオペランドを指定します。
   name_service_mode オペランドに agent を指定
   name_manager_node オペランドにマネジャノードのノード名を指定

OpenTP1 システム内のそのほかのノードの作業は必要ありません。

●ノード自動追加機能を使用しない場合のノードの追加手順

追加するノードの作業

1. システム共通定義の all_node オペランドに OpenTP1 システム内のすべてのノードを指定します。
2. 次のどちらかの方法で手順 1 で指定した定義を有効にします。
   namndchg コマンドを実行
   OpenTP1 を停止し，再起動

OpenTP1 システム内のそのほかのノードの作業

1. システム共通定義の all_node オペランドに追加するノードを指定します。
2. 次のどちらかの方法で手順 1 で指定した定義を有効にします。
   namndchg コマンドを実行
   OpenTP1 を停止し，再起動
3. OpenTP1 システム内のすべてのノードに対して，手順 1～2 を繰り返します。

ノード自動追加機能を使用する場合の運用方法については，マニュアル「OpenTP1 運用と操作」を参照してください。

### (5) ノードの情報表示

namsvinf コマンドを実行すると，OpenTP1 ノードの起動結果，IP アドレス，およびネームサービスのポート番号を表示できます。情報を表示できる OpenTP1 ノードは，システム共通定義の all_node オペランドおよび all_node_ex オペランドに指定されたノードです。

## 3.2.4 XATMI インタフェースの通信

**XATMI インタフェース**とは，X/Open で規定する DTP モデルに準拠した，クライアント/サーバ形態の通信をする API です。OpenTP1 では，XATMI インタフェースを使って，UAP プロセス間で通信できます。

XATMI インタフェースの通信は，通信プロトコルに OSI TP だけが使えます。なお，OSI TP は，ノード間通信でだけ使えます。自ノード内通信では使えません。

XATMI インタフェースの通信を使用できるのは，**SUP** と **SPP** です。SUP と SPP の両方に，XATMI インタフェース定義ファイルから作成した**スタブ**を結合しておいてください。

MHP を作成するときには，XATMI インタフェースの関数は使えません。

XATMI インタフェースの通信については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

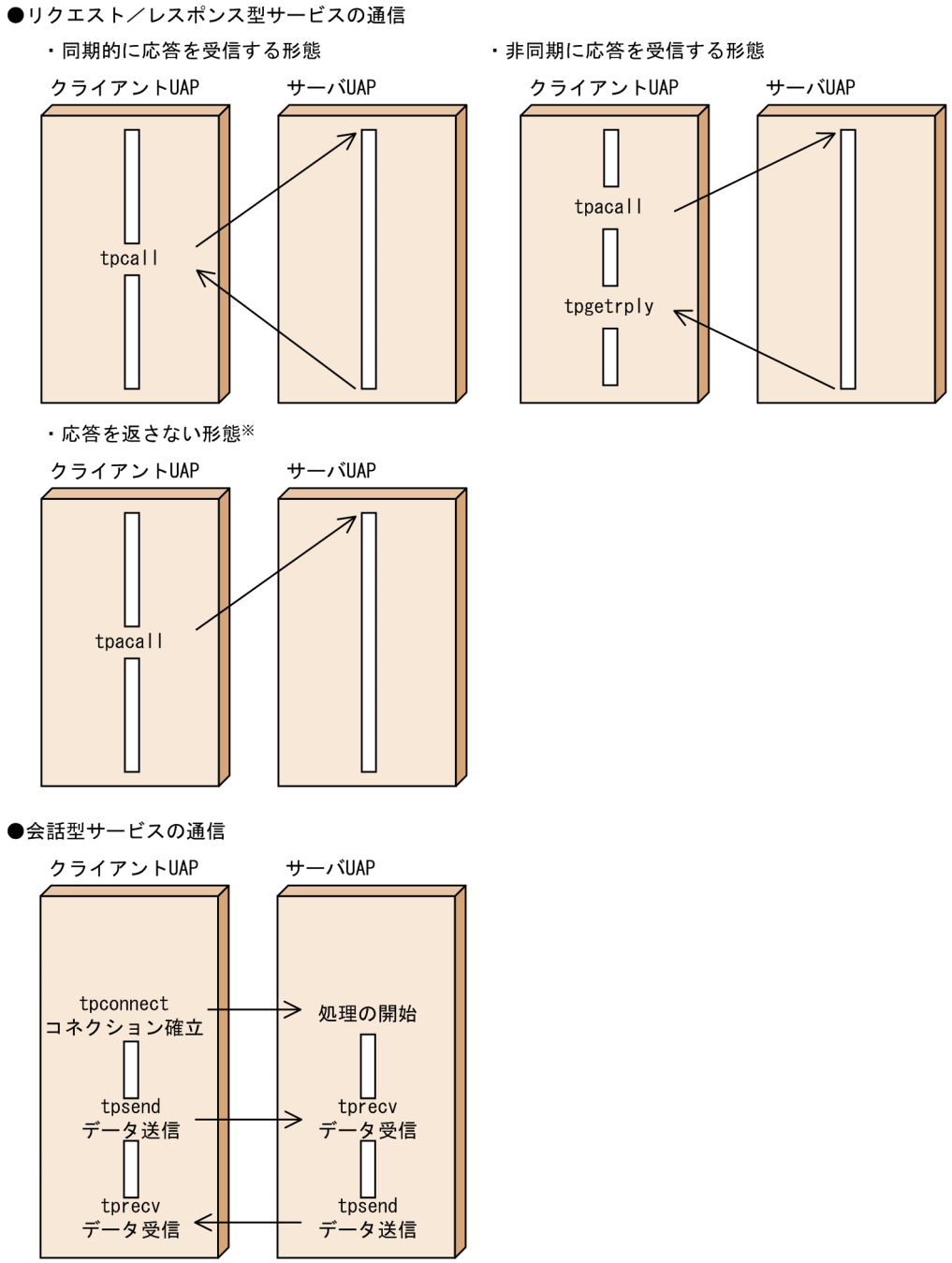
### (1) XATMI インタフェースの通信形態

XATMI インタフェースでは，次に示す通信形態を提供します。

- **リクエスト/レスポンス型サービスの通信**
  サービス関数に対して一つの要求を送信し，一つの応答を受信する形態の通信です。OpenTP1 のリモートプロシジャコールと同様の，サービスを要求して結果を受信する形態の通信です。
- **会話型サービスの通信**
  通信相手のサービス関数を起動して確立したコネクションを介して，サービス関数と互いにデータを送受信する通信です。通信プロトコルに OSI TP を使っている場合は，会話型サービスの通信は使えません。

XATMI インタフェースの通信の概要を次の図に示します。

図 3-25　XATMI インタフェースの通信の概要




注※　リクエスト／レスポンス型サービスの通信の，応答を返さない形態は，サービス要求をトランザクションにできません。

## (2)　サービスの要求方法

サービスを要求するときは，サーバ UAP のサービス関数を識別する名称（**サービス名**）を引数に設定して，通信する関数を呼び出します。

### (3) 通信に使用するデータ

XATMI インタフェースの通信では，プロセス間の通信で使用するデータの型を設定します。XATMI インタフェースの通信で使用する，型付きのデータを格納するバッファを**型付きバッファ（タイプトバッファ）**または**型付きレコード（タイプトレコード）**といいます。

## 3.2.5 TxRPC インタフェースの通信

**TxRPC インタフェース**とは，X/Open で規定するクライアント/サーバ形態の通信方法です。OpenTP1 の UAP プロセス間で，TxRPC インタフェースの方式で通信できます。

ほかのクライアント/サーバ形態の通信と異なり，TxRPC の通信では**ユーザが作成した関数**をクライアントから呼び出して通信します。

TxRPC の通信をする UAP では，OpenTP1 の RPC（dc_rpc_call 関数）を使って，ほかの UAP と通信することもできます。

### (1) TxRPC 通信の種類

TxRPC の通信方法には，DCE RPC を使うかどうかで次に示す 2 種類があります。

- IDL-only TxRPC
  IDL コンパイラから生成されるファイルだけを使って UAP を作成する方式です。IDL-only TxRPC の場合は，DCE を前提としません。
- RPC TxRPC
  通信プロトコルに DCE RPC を使う方式です。このバージョンでは RPC TxRPC をサポートしていません。

TxRPC 通信については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

### (2) 作成できるアプリケーションプログラム

TxRPC で通信する UAP として，次のものが作成できます。

- クライアント UAP（SUP）
- サーバ UAP（SPP）

TxRPC 通信では，作成する UAP の種類によって，前提となるライブラリが異なります。

#### (a) SUP または SPP を作成する場合

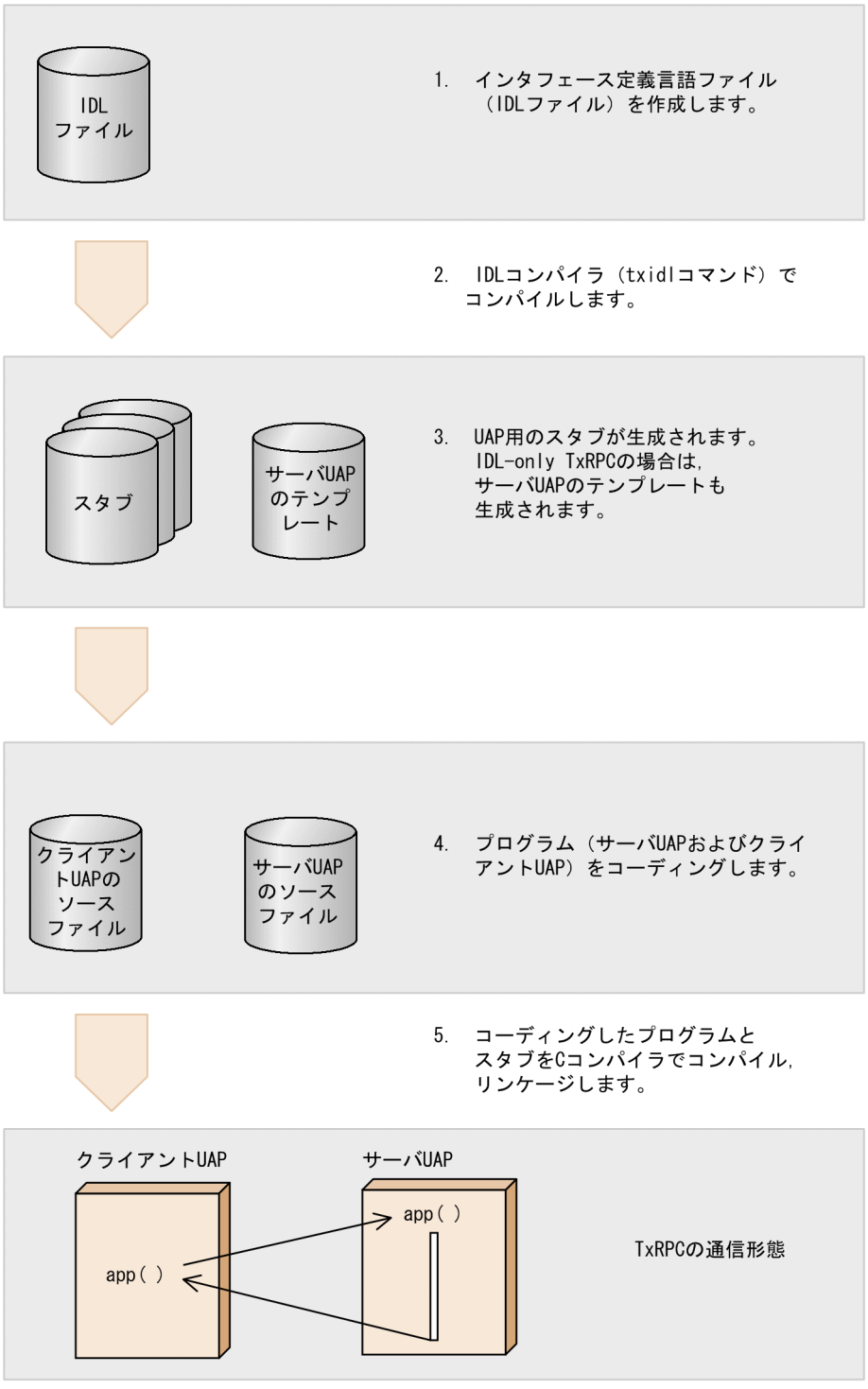
次に示す製品がシステムに組み込んであることが前提です。

- TP1/Server Base

## (3)　TxRPC 通信の UAP を作成する手順

TxRPC の通信をする UAP を作成するときの手順を，次の図に示します。

図 3-26　TxRPC 通信の UAP 作成手順




TxRPC の通信の UAP を作成する手順については，マニュアル「OpenTP1 プログラム作成の手引」，および「OpenTP1 プログラム作成リファレンス C 言語編」を参照してください。

## 3.3　メッセージ制御

OpenTP1 システムでは，OpenTP1 を使っていないシステムと通信するために，各種の通信プロトコルに準拠した形式の通信ができます（**メッセージ制御機能**）。この通信形態を使うと，OpenTP1 を使っていないシステムと水平，または垂直分散型のネットワークシステムを構築できます。

OpenTP1 のメッセージ制御機能を使うときは，次に示す製品が前提となります。

- TP1/Message Control（メッセージ制御機能を管理する製品）
- TP1/NET/Library（ネットワークを制御するための製品）
- TP1/NET/××××（通信プロトコル別のインタフェースを制御する製品）

メッセージ制御機能で使う UAP は，**MHP** です。さらに，一部のメッセージの送受信は SPP でもできます。

なお，OS が Windows の場合は，一部の機能が使用できません。Windows でサポートしていないメッセージ制御機能については，マニュアル「OpenTP1 使用の手引 Windows(R)編」を参照してください。

### 3.3.1　メッセージ送受信

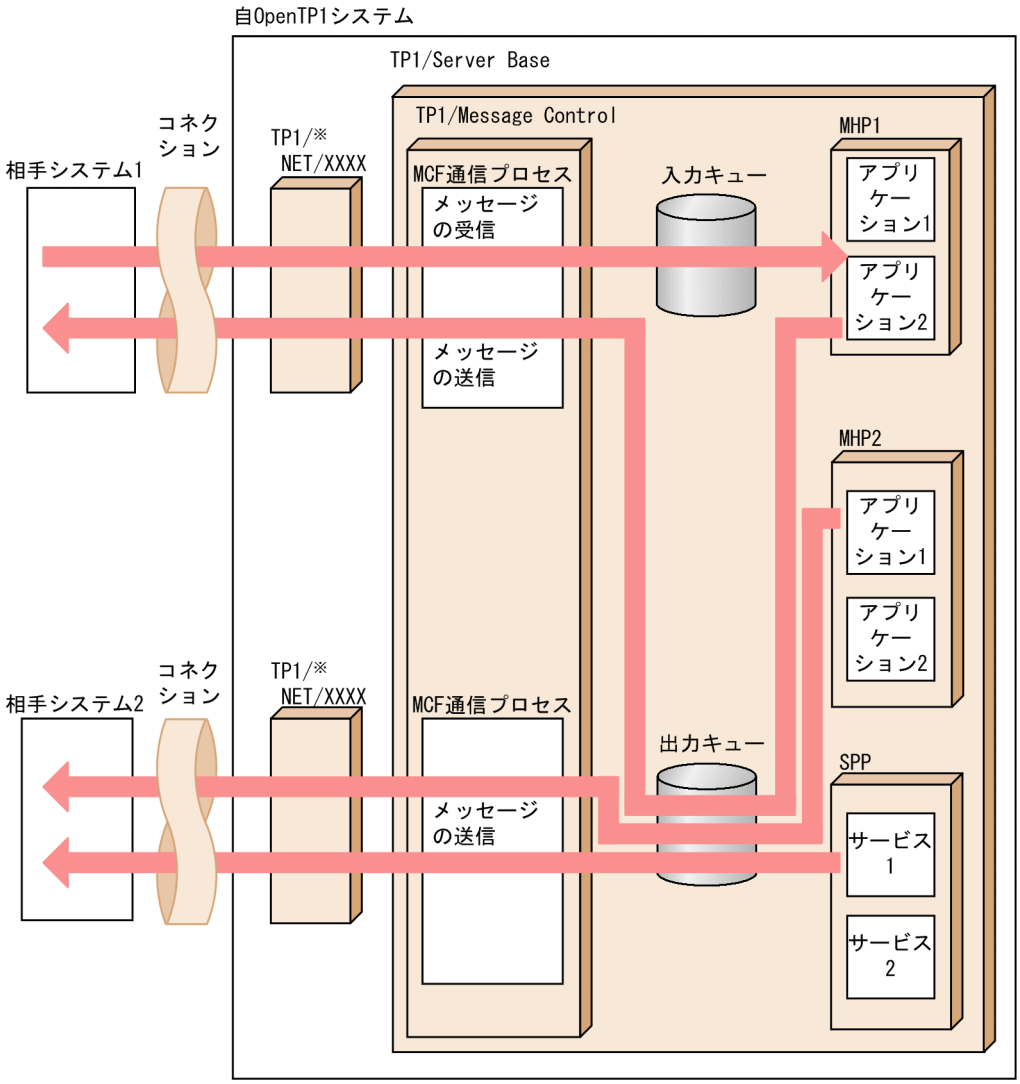
OpenTP1 は，相手システムからメッセージを受信すると，入力キューにメッセージを書き込み，該当する MHP にメッセージを渡します。MHP は，送信先を指定してメッセージを出力キューに書き込みます。出力キューに書き込まれたメッセージは，OpenTP1 が優先度に従って相手システムに送信します。

MHP がメッセージを受け取ってからリターンするまでの間が，MHP のグローバルトランザクションです。また，トランザクション属性を持つ SPP から他システムへメッセージを送信することもできます。

メッセージの送受信は，通信プロトコルごとに **MCF 通信プロセス**が管理します。MCF 通信プロセスを起動するため，通信プロトコルごとに MCF 通信構成定義を作成してください。

メッセージの送受信の流れを次の図に示します。

図 3-27　メッセージの送受信の流れ




注※　通信プロトコル対応の製品です。

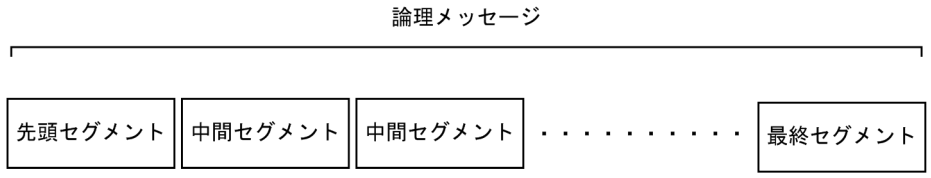
## 3.3.2　メッセージの構造

通信で意味を持つ単位を**論理メッセージ**といいます。そして，UAP の関数で扱うメッセージの単位を**セグメント**といいます。論理メッセージは，一つのセグメントから構成される場合と，複数のセグメントから構成される場合があります。

論理メッセージとセグメントの関係を次の図に示します。

図 3-28　論理メッセージとセグメントの関係

●論理メッセージの形式（複数のセグメントで構成される場合）

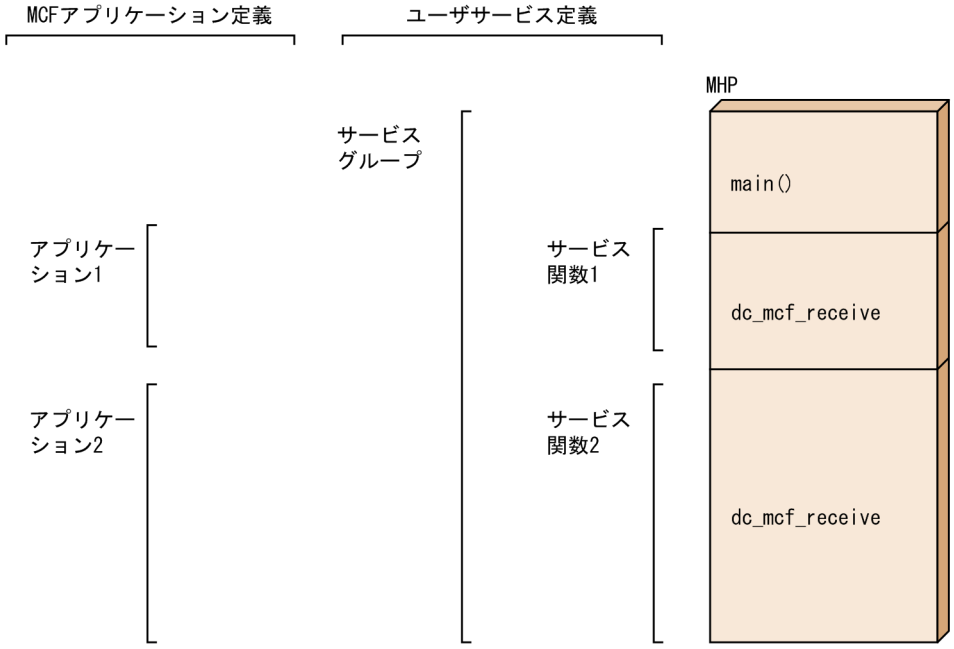



## 3.3.3　アプリケーションプログラムの構造とアプリケーション名の関連

MHP はメイン関数とサービス関数から構成されます。この MHP の実行形式ファイルを TP1/Server Base に登録するため，**ユーザサービス定義**を作成します。そして，MCF で使えるように MHP を **MCF アプリケーション定義**に指定します。ここでは，MHP のサービス関数とアプリケーション名を対応づけます。これで，MHP のサービス関数が MCF のアプリケーションとして使えるようになります。

アプリケーションプログラムの構造とアプリケーション名の関連を次の図に示します。

図 3-29　アプリケーションプログラムの構造とアプリケーション名の関連



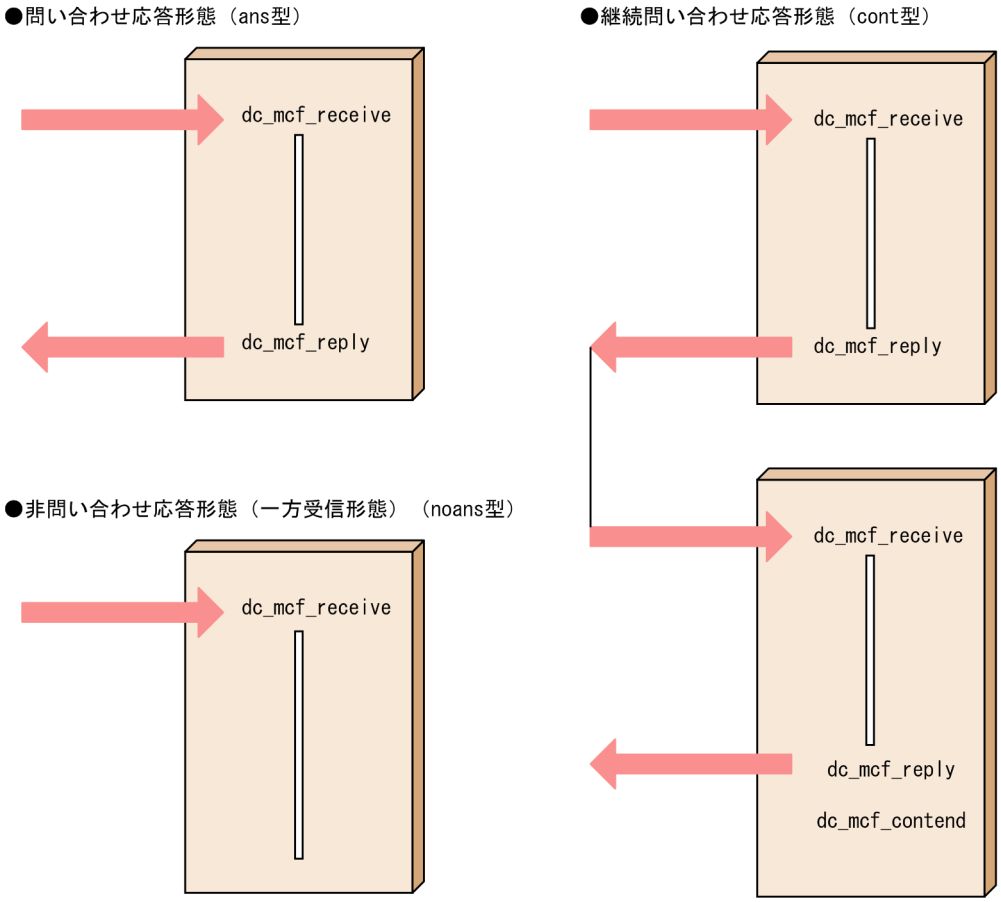

## 3.3.4　メッセージの通信形態

MHP で使えるメッセージの通信形態を次に示します。通信プロトコルによって使える形態は異なります。

ここで説明するメッセージの通信形態は，MHP でだけ使えます。

メッセージの通信形態とアプリケーションの型を次の図に示します。

図 3-30　メッセージの通信形態とアプリケーションの型




## (1)　問い合わせ応答形態

相手システムから送られてきたメッセージを受信して，応答のメッセージを送信する形態です。メッセージを受信するときは，dc_mcf_receive 関数を使い，応答メッセージを送信するときは，dc_mcf_reply 関数を使います。

問い合わせ応答形態の通信をする MHP のアプリケーションの型には，**応答型（ans 型）**を指定します。応答型を指定した MHP で dc_mcf_reply 関数を呼び出さないで終了すると，エラーが起こったと見なして，MHP は異常終了します。

## (2)　非問い合わせ応答形態（一方受信形態）

相手システムから送られてきたメッセージを受信するだけで，応答のメッセージを送信しない形態です。メッセージを受信するときは，dc_mcf_receive 関数を使います。

非問い合わせ応答形態（一方受信形態）の通信をする MHP のアプリケーションの型には，**非応答型（noans 型）**を指定します。

### (3)　継続問い合わせ応答形態

問い合わせ応答形態を連続させる形態です。dc_mcf_receive 関数でメッセージを受信して dc_mcf_reply 関数で応答メッセージを送信したあとに，続けてメッセージを受信します。続けてメッセージを受信する MHP は，同じ MHP でも異なった MHP でもかまいません。継続問い合わせ応答形態の通信を終了するときは，dc_mcf_contend 関数を使います。

継続問い合わせ応答形態の通信をする MHP のアプリケーションの型には，**継続問い合わせ応答型（cont 型）**を指定します。

## 3.3.5　通信形態に依存しないメッセージ

MHP または SPP から送受信できるメッセージについて説明します。

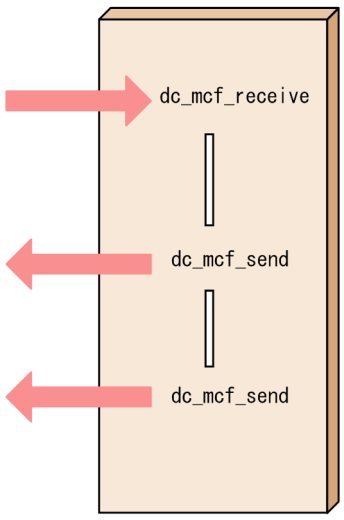
### (1)　一方送信メッセージ

OpenTP1 の UAP から送信するメッセージです。一方送信メッセージを送信するときは，dc_mcf_send 関数を使います。dc_mcf_send 関数を呼び出した UAP のトランザクションがコミットしてから，実際に相手システムにメッセージが送信されます。

一方送信メッセージの概要を次の図に示します。

図 3-31　一方送信メッセージの概要

●一方送信メッセージ




### (2)　メッセージの再送

OpenTP1 の UAP からいったん送信したメッセージを，再び送り直すことをメッセージの再送といいます。メッセージを再送するときは，dc_mcf_resend 関数を使います。

### (3) 同期型のメッセージ

同期型のメッセージとは，UAP からの関数呼び出しと同期してメッセージを送受信する形式のメッセージです。同期型のメッセージには，次に示す 3 種類があります。

#### (a) 同期送信

同期送信のメッセージを使うときは，dc_mcf_sendsync 関数を呼び出します。UAP から dc_mcf_sendsync 関数を呼び出すと，UAP の処理をいったん止めて，出力キューにメッセージが書き込まれます。そして，相手システムにメッセージを送信し終わってから，関数がリターンします。

#### (b) 同期受信

同期受信のメッセージを使うときは，dc_mcf_recvsync 関数を呼び出します。UAP から dc_mcf_recvsync 関数を呼び出すと，UAP の処理をいったん止めて，関数に指定したアプリケーション名へのメッセージを入力キューから探します。メッセージがある場合は，セグメントを受信して関数がリターンします。該当するメッセージがない場合は，送信されてくるまで待ちます。

#### (c) 同期送受信

同期送受信のメッセージを使うときは，dc_mcf_sendrecv 関数を呼び出します。UAP から dc_mcf_sendrecv 関数を呼び出すと，UAP の処理をいったん止めて，出力キューにメッセージが書き込まれます。そして，相手システムにメッセージを送信し終わってから，応答を自システムで受信し，その後，関数がリターンします。

## 3.3.6 メッセージ制御のトランザクション

MCF のトランザクション処理について説明します。

### (1) トランザクションとして処理する場合

相手システムから MCF でメッセージを受信したあとで，MHP のサービスを呼び出した時点でトランザクションが開始します。メッセージを処理した MHP が正常終了すると，トランザクションがコミットしたことになり，該当する MHP の処理が有効になります。

MHP の処理では，トランザクションをコミットすることもロールバックすることもできます。コミットするときは，dc_mcf_commit 関数を，ロールバックするときは dc_mcf_rollback 関数を使います。

### (2) トランザクションとして処理しない場合

受信したメッセージの処理をトランザクションとして処理しないこともできます。トランザクションとして処理しないと，UAP に障害が起こった場合に回復できませんが，処理効率は良くなります。トランザクションとして処理しない MHP を**非トランザクション属性の MHP** といいます。非トランザクション属性の MHP を利用する場合，アプリケーション属性定義の trnmode オペランドで nontrn を指定します。

## 3.3.7　MHP の起動

MCF は，次に示す場合に MHP を起動させます。

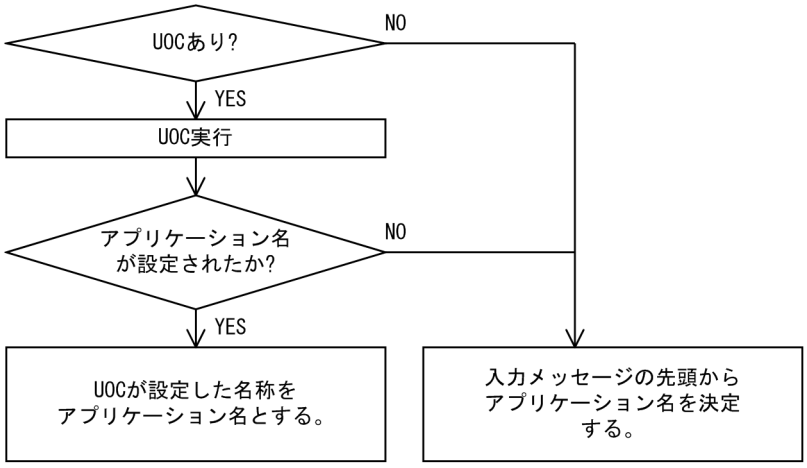
- MCF がメッセージを受信した場合
- UAP がアプリケーションを起動する関数（dc_mcf_execap 関数）を呼び出した場合
- MCF イベントが通知された場合
- MHP を起動させるコマンド（mcfuevt コマンド）を実行した場合

### (1)　メッセージ受信による MHP の起動

MCF は，入力メッセージを基に，MHP を起動する名前（**アプリケーション名**）を決めます。

アプリケーション名の決定方法を次の図に示します。

図 3-32　アプリケーション名の決定方法




アプリケーション名を決定する方法には，**アプリケーション名決定ユーザオウンコーディング（UOC）**を組み込まない方法と，組み込む方法の二つがあります。

#### (a)　アプリケーション名決定 UOC を組み込んでいない場合

MCF は，入力メッセージの先頭から最初に空白が現れるまでの 8 文字以内の文字列をアプリケーション名と見なします。OpenTP1 はアプリケーション名を MHP のサービスグループ名とサービス名に変換し，該当する MHP を起動します。先頭から 9 文字目まで空白の場合は，アプリケーション名不正としてエラーとなります。

アプリケーション名と，サービスグループ名およびサービス名の対応は，MCF アプリケーション定義のアプリケーション属性定義で指定します。

#### (b)　アプリケーション名決定 UOC を組み込んでいる場合

MCF は，UOC を呼び，その UOC が設定した名称をアプリケーション名と見なします。OpenTP1 はアプリケーション名を MHP のサービスグループ名とサービス名に変換し，該当する MHP を起動します。

アプリケーション名と，サービスグループ名・サービス名の対応は，MCF アプリケーション定義のアプリケーション属性定義で指定します。なお，UOC を組み込んでいても UOC でアプリケーション名を設定しなければ，入力メッセージの先頭から最初に空白が現れるまでの 8 文字以内の文字列をアプリケーション名と見なします。先頭から 9 文字目まで空白の場合は，アプリケーション名不正としてエラーとなります。

## (2) UAP からの関数発行による MHP の起動

MHP や SPP から関数（dc_mcf_execap 関数）を呼び出して，ほかの MHP を起動できます。MHP や SPP から MHP を起動する機能を，**アプリケーション起動機能**といいます。

起動される MHP は，起動要求元の MHP や SPP が正常終了したあと，または指定された時間が経過したあとに，**アプリケーション起動プロセス**が起動します。

MHP から関数（dc_mcf_rollback 関数）を呼び出してトランザクションをロールバックするとき，関数の引数にリトライを指定すると，ロールバックした MHP を再び起動して処理を再実行します。この場合もアプリケーション起動プロセスが起動します。

このように，UAP から関数を呼び出してアプリケーションを起動，または再起動するときには，アプリケーション起動プロセスを起動する必要があります。MCF 通信プロセスとは別に，アプリケーション起動プロセスの MCF 通信構成定義を作成してください。アプリケーション起動プロセスでプロトコルに依存しない処理をすることで，MCF 通信プロセスの負荷を軽減します。

起動要求元の MHP や SPP が MCF にデータを渡す，MCF 内部の論理的な通信路を**内部通信路**といい，アプリケーション起動環境定義で指定します。MCF が，起動されるアプリケーションにデータを渡すときにも，内部通信路を使用します。

## (3) MCF イベントによる MHP の起動

エラーや障害の発生，コネクションの確立や解放などを通知するためのメッセージを **MCF イベント**といいます。この MCF イベントに対応して起動させる MHP を作成できます。MCF イベントに対応して起動させる MHP を，**MCF イベント処理用 MHP** といいます。MCF イベント処理用 MHP を作成しておくと，オンライン中に発生した現象を特定の端末に通知するなど，ユーザ独自の対応がとれます。

### (a) MCF イベントの種類

MCF イベントには，エラーや障害発生などの**エラーイベント**と，コネクションの確立や解放などプロトコルに依存する**通信イベント**の 2 種類があります。

2 種類のイベントには，発生した現象の内容に応じて **MCF イベントコード**があります。MCF イベントコードと，MCF イベント処理用 MHP のサービスグループ名とサービス名は，アプリケーション属性定義で対応づけます。各 MCF イベントに対応した MHP のサービスを，アプリケーション起動プロセス，または MCF 通信プロセスが起動します。エラーイベント用の MCF イベント処理用 MHP はアプリケーション起動プロセスまたは MCF 通信プロセスで，通信イベント用の MCF イベント処理用 MHP は MCF 通信プロセスで起動します。エラーイベント用の MCF イベント処理用 MHP を使用する場合は，アプリケーション起動プロセス用の MCF 通信構成定義も作成してください。

通信イベントが障害となった場合に，エラーイベントを通知できます。この通知の有無はアプリケーション属性定義の errevt オペランドで指定できます。ただし，エラーイベントの障害によるエラーイベントは通知できません。

MCF イベントの一覧を次の表に示します。MCF イベントについては，マニュアル「OpenTP1 プログラム作成の手引」および「OpenTP1 プロトコル」の該当するプロトコル編を参照してください。

### 表 3-5　MCF イベントの一覧

| MCF イベント名 | MCF イベントコード | MCF イベントが通知された原因 | MCF イベント処理用 MHP で実行する処理の例 |
|---|---|---|---|
| 不正アプリケーション名検出通知イベント | ERREVT1 | メッセージのアプリケーション名が MCF アプリケーション定義にありません。 | 該当するアプリケーション名がなかったことを知らせます。<br>受信したメッセージが問い合わせメッセージの場合は，応答メッセージを送信できます。 |
| メッセージ廃棄通知イベント | ERREVT2 | 次に示す理由で，MCF で受信した入力キューのメッセージ，または即時指定のアプリケーション起動機能によって入力キューに入力されたメッセージを廃棄しました。<br>• 入力キューに障害が発生しました。<br>• 入力メッセージ最大格納数を超過しました。<br>• 動的共用メモリが不足しました。<br>• キューファイルが満杯になりました。<br>• MHP のサービス，サービスグループ，またはアプリケーションが閉塞しています。<br>• スケジュール閉塞されているサービスグループの入力キューに未処理受信メッセージが残った状態で，OpenTP1 を正常終了または計画停止 A で終了しました。<br>• MHP のサービスグループ，またはアプリケーションがセキュア状態です。<br>• MHP で呼び出す dc_mcf_receive 関数にセグメントを渡す前に，MHP が異常終了しました。<br>• アプリケーション名に相当する MHP のサービスがありません。<br>• ユーザサーバ未起動などによって，MHP の起動に失敗しました。<br>• DBMS の障害などによって，トランザクションの開始に失敗しました。 | メッセージを廃棄したことを知らせます。<br>受信したメッセージが問い合わせメッセージの場合は，応答メッセージを送信できます。 |

| MCF イベント名 | MCF イベントコード | MCF イベントが通知された原因 | MCF イベント処理用 MHP で実行する処理の例 |
|---|---|---|---|
| UAP 異常終了通知イベント | ERREVT3 | MHP で呼び出す dc_mcf_receive 関数にセグメントを渡したあとで，MHP が異常終了，またはロールバック※しました。 | UAP が異常終了，またはロールバックしたことを知らせます。<br>受信したメッセージが問い合わせメッセージの場合は，応答メッセージを送信できます。 |
| タイマ起動メッセージ廃棄通知イベント | ERREVT4 | タイマ指定のアプリケーション起動機能によって入力キューに入力されたメッセージを ERREVT2 に示す理由で破棄しました。 | メッセージを廃棄したことを知らせます。<br>受信したメッセージが問い合わせメッセージの場合は，応答メッセージを送信できます。 |
| 未処理送信メッセージ廃棄通知イベント | ERREVTA | 次に示す理由で，UAP から送信した未処理送信メッセージを廃棄しました。<br>• MCF の正常終了処理時に，未処理送信メッセージの滞留時間監視の時間切れ（タイムアウト）が起こりました。<br>• mcftdlqle コマンドまたは dc_mcf_tdlqle 関数で，出力キューが削除されました。<br>• タイマ起動要求や閉塞されている論理端末の出力キューに未処理送信メッセージが残った状態で，dcstop コマンドが実行されました。 | 未処理送信メッセージを廃棄したことを知らせます。<br>受信した未処理送信メッセージは，任意のファイルへ退避します。 |
| 送信障害通知イベント | SERREVT | メッセージを送信する途中で，通信プロトコルの障害が起こりました。 | 通信プロトコルの障害でメッセージを送信できなかったことを知らせます。 |
| 送信完了通知イベント | SCMPEVT | 相手システムへ，メッセージを正常に送信できました。 | 相手システムまでメッセージを正常に送信できたことを知らせます。 |
| 障害通知イベント | CERREVT（VERREVT） | 通信管理プログラムで，コネクション障害，または論理端末障害が起こりました。コネクション確立再試行を定義している場合は，通知されません。 | コネクション，または論理端末に障害が起こったことを知らせます。 |
| 状態通知イベント | COPNEVT（VOPNEVT） | コネクションが確立しました。 | メッセージを送受信できることを知らせます。 |
| | CCLSEVT（VCLSEVT） | コネクションが正常に解放されました。 | メッセージを送受信できないことを知らせます。 |

注

ERREVT1，ERREVT2，ERREVT3，ERREVT4，ERREVTA はエラーイベントを示します。

SERREVT，SCMPEVT，CERREVT，COPNEVT，CCLSEVT は通信イベントを示します。

注※

MCF アプリケーション定義（mcfaalcap -g）の recvmsg オペランドに r を指定した場合，または dc_mcf_rollback の action に DCMCFRTRY もしくは DCMCFRRTN を指定した場合は除きます。

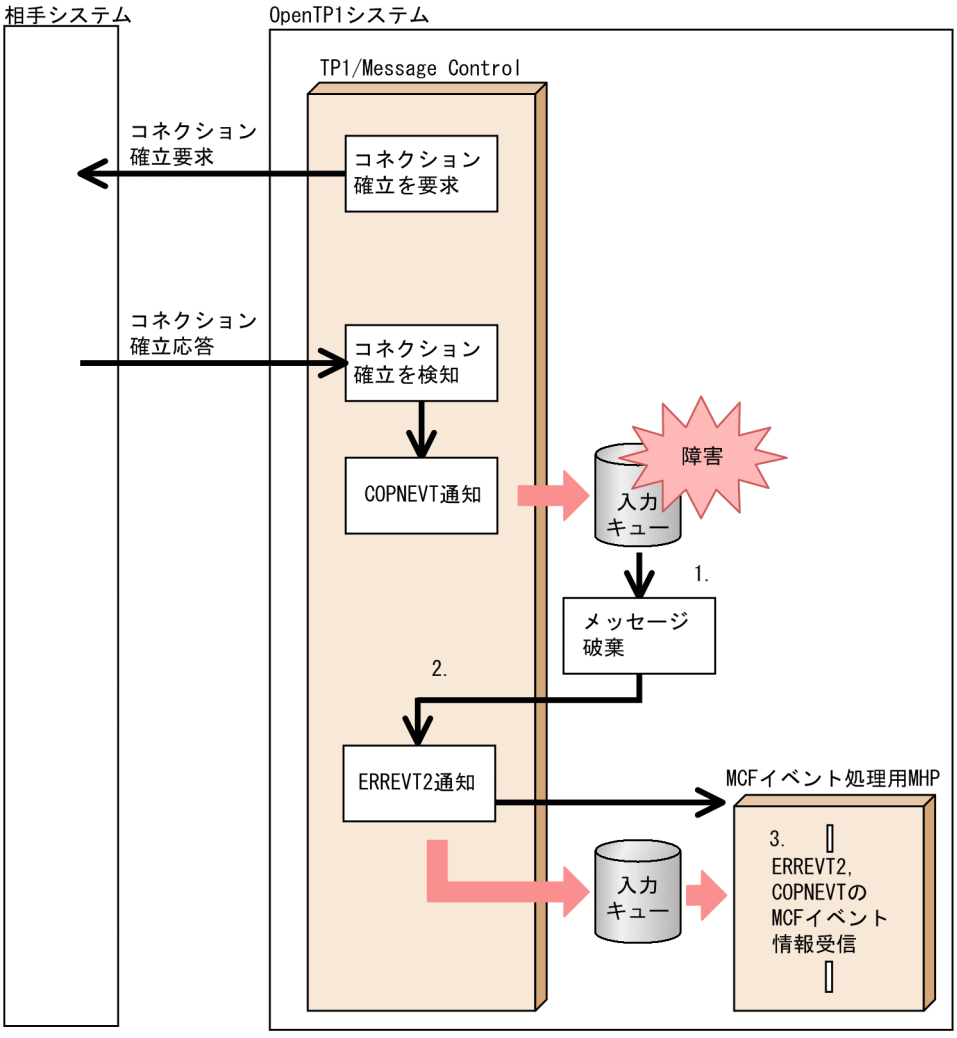
### (b)　通信イベント障害時のエラーイベント通知

通信イベント障害時のエラーイベント通知は，通信イベントの障害をエラーイベントによってユーザに通知する機能です。通信イベント障害時のエラーイベント通知を使用することによって通信イベントの障害の回復手段に MCF イベント処理用 MHP を使用でき，通信イベントのメッセージは第 2 セグメントで受信できます。

通信イベント障害時のエラーイベント通知の機能を使用するには，アプリケーション属性定義の-n オプションの errevt オペランドに yes を指定します。

次の図に COPNEVT 障害時の ERREVT2 通知の概要を示します。

図 3-33　COPNEVT 障害時の ERREVT2 通知の概要




1. COPNEVT が入力キューの障害によって破棄される。

2. 制御が MCF に戻り，ERREVT2 が通知されて，ERREVT2 の MCF イベント処理用 MHP がスケジュールされる。
3. ERREVT2 の MCF イベント情報を先頭セグメントで受信できる。障害となった COPNEVT の MCF イベント情報を第 2 セグメントで受信できる。受信した MCF イベント情報を参照し，COPNEVT の MCF イベント処理用 MHP で実行する予定であった処理を，ERREVT2 の MCF イベント処理用 MHP で実行することによって，障害に対処できる。

## (4) コマンドを使用した MHP の起動

コマンドを入力して，MHP を起動できます。メッセージ受信を契機に起動する MHP でも，コマンドで直接 MHP を起動することで他システムへのメッセージ送信ができるようになります。

起動できる MHP は，非応答型（noans 型）だけです。コマンドで起動する MHP には，MCF アプリケーション定義のアプリケーション属性定義で noans 型を指定してください。

### (a) コマンドで起動する MHP の定義

コマンドで起動する MHP のアプリケーション名は，UCMDEVT とします。MCF アプリケーション定義のアプリケーション属性定義 mcfaalcap コマンドの-n オプションに，次の値を指定しておきます。

- name オペランド：UCMDEVT
- kind オペランド：user（または省略）
- type オペランド：noans（または省略）

### (b) MHP の起動方法

MHP を起動するときは，mcfuevt コマンドを実行します。mcfuevt コマンドの引数として，MCF 通信プロセス識別子と MHP に渡す入力メッセージを指定します。

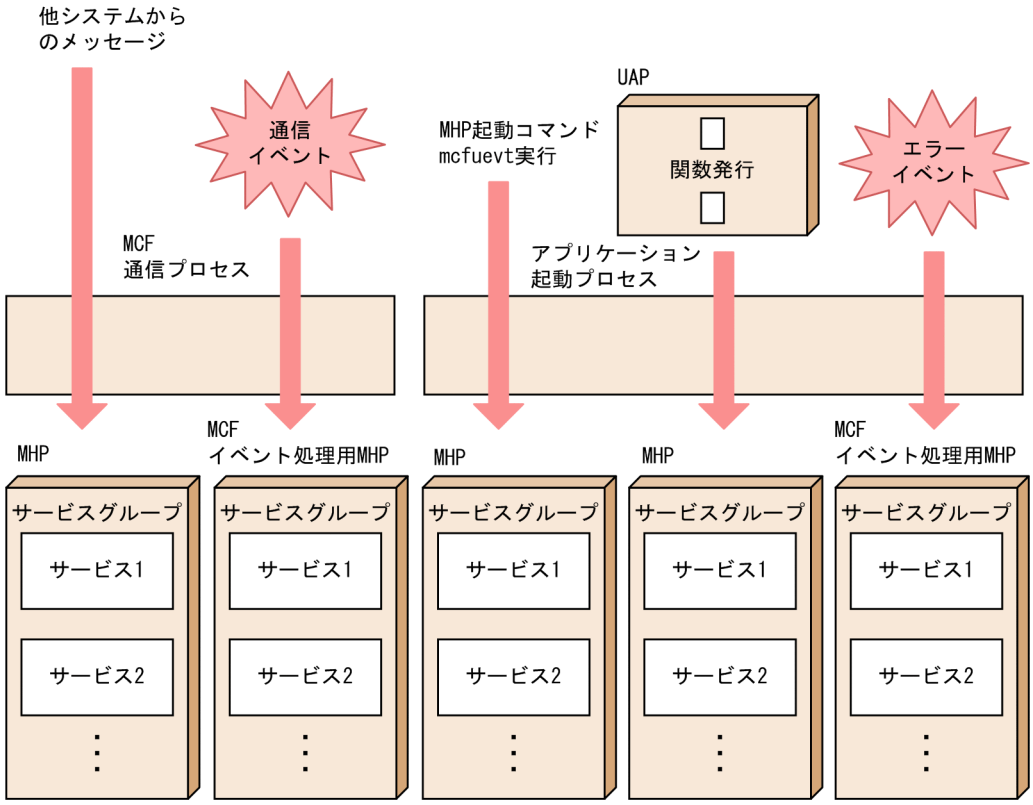
UCMDEVT を定義していない場合に mcfuevt コマンドを実行したときは，mcfuevt コマンドはエラーリターンします。このとき，ERREVT1 は通知されません。

コマンドで起動する MHP は通信プロトコルに依存しないので，mcfuevt コマンドに指定する MCF 通信プロセスには，アプリケーション起動プロセスを指定することをお勧めします。

## (5) MCF が MHP を起動させる要因

MCF が MHP を起動させる要因を次の図に示します。

図 3-34　MCF が MHP を起動させる要因




## 3.3.8　メッセージキュー

### (1)　入力キューと出力キュー

OpenTP1 は，送受信するメッセージをキューに蓄えます。相手システムから受信した入力メッセージを管理する待ち行列を**入力キュー（ITQ）**といいます。また，相手システムへ送信する出力メッセージを管理する待ち行列を**出力キュー（OTQ）**といいます。

入力キューや出力キューの割り当て先として，**ディスクキュー**と**メモリキュー**を選択できます。

割り当て先としてディスクキューを使用する場合，入力キューおよび出力キューに蓄えられたメッセージは，オンラインシステムが異常終了した場合などの再開始時に，引き継ぐことができます。また，メッセージの再送を使用する場合は，出力キューにディスクキューを割り当てる必要があります。

一方，メモリキューを使用する場合，再開始時にメッセージを引き継ぐことはできませんが，ディスクへの入出力が発生しないため，ディスクキューと比較して処理性能が向上します。再開始時に情報を引き継ぐ必要のないメッセージは，応答時間の短いメモリキューの使用をお勧めします。

OpenTP1 は，ディスクキューとして，メッセージキューファイルを使用します。ディスクキューを使用する指定をした場合には，メッセージキューファイルを作成してください。メッセージキューファイルについては，「4.3.1　メッセージキューファイル」を参照してください。

入力キュー用にディスクキューを使用するのか，またはメモリキューを使用するのかは，アプリケーションごとにアプリケーション属性定義（mcfaalcap）の quekind オペランドで指定します。

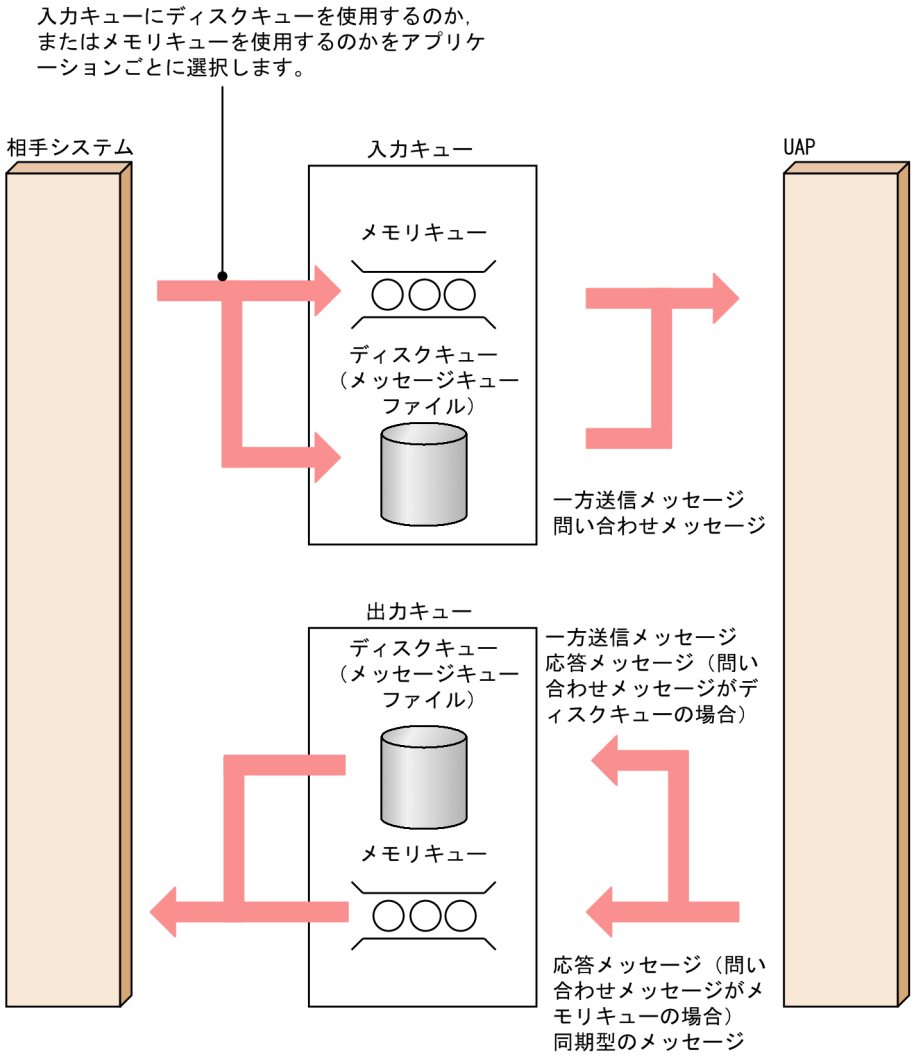
出力キュー用にディスクキューを使用するのか，またはメモリキューを使用するのかは，論理端末ごとに論理端末定義（mcftalcle）の quekind オペランドで指定します。

ただし，出力キューにディスクキューを使用する指定をしていても，次のメッセージの場合はメモリキューを使用します。

- 問い合わせ応答型や継続問い合わせ応答型のアプリケーションで，問い合わせメッセージが使用した入力キューがメモリキューのときの応答メッセージの送信
- 同期型のメッセージの送信

入力キューまたは出力キューにディスクキューを使用したメッセージ送受信を次の図に示します。

### 図 3-35　入力キューまたは出力キューにディスクキューを使用したメッセージ送受信



## (2) OpenTP1 再開始時のメッセージの扱い

OpenTP1 の再開始時にメッセージを引き継ぐかどうかは，メッセージの種類によって異なります。

OpenTP1 の再開始時のメッセージの扱いを次の表に示します。なお，メッセージが格納されているキューの割り当て先がメモリキューの場合，表の内容に関係なく OpenTP1 の再開始時にメッセージを破棄します。

### 表 3-6 OpenTP1 の再開始時のメッセージの扱い

| キュー種別 | メッセージの種類 | | OpenTP1 再開始時のメッセージの扱い |
|---|---|---|---|
| 入力キュー | 一方送信メッセージ<br>問い合わせメッセージ | | 通信プロトコル対応製品によって，動作が異なります。表 3-7～表 3-8 を参照してください。 |
| | MCF イベント | ERREVT1 | 通信プロトコル対応製品によって，動作が異なります。表 3-7～表 3-8 を参照してください。※1 |
| | | ERREVT2<br>ERREVT3<br>ERREVT4<br>ERREVTA | エラーイベントが発生したメッセージの種類によって，動作が異なります。<br>OpenTP1 の再開始時に引き継がれるメッセージの場合，エラーイベントを引き継ぎます。<br>OpenTP1 の再開始時に破棄されるメッセージの場合，エラーイベントを破棄します。※1 |
| | | SERREVT<br>SCMPEVT | 破棄します。 |
| | | 通信イベント（SERREVT，SCMPEVT を除く） | 通信プロトコル対応製品によって，動作が異なります。表 3-7～表 3-8 を参照してください。 |
| | UAP からの関数発行によるアプリケーション起動メッセージ※2 | | ＜非応答型のアプリケーションを起動する場合＞<br>引き継ぎます。 |
| | | | ＜応答型，継続問い合わせ応答型のアプリケーションを起動する場合＞<br>起動元のメッセージの種類によって，動作が異なります。<br>OpenTP1 の再開始時に引き継がれるメッセージの場合，アプリケーション起動メッセージを引き継ぎます。OpenTP1 の再開始時に破棄されるメッセージの場合，アプリケーション起動メッセージを破棄します。 |
| | 運用コマンド（mcfuevt）によるアプリケーション起動メッセージ | | 引き継ぎます。 |
| | ユーザタイマ監視機能によるアプリケーション起動メッセージ | | 破棄します。 |
| 出力キュー | 一方送信メッセージ | | 引き継ぎます。 |

| キュー種別 | メッセージの種類 | OpenTP1 再開始時のメッセージの扱い |
|---|---|---|
| 出力キュー | 応答メッセージ | 通信プロトコル対応製品によって，動作が異なります。表 3-7～表 3-8 を参照してください。 |
| | 同期型のメッセージ | 破棄します。 |

注※1

エラーイベントの場合，OpenTP1 の再開始時にメッセージを引き継ぐかどうかを選択できます。詳細については，マニュアル「OpenTP1 システム定義」の UAP 共通定義（mcfmuap）を参照してください。

注※2

アプリケーション起動メッセージを再開始時に引き継ぐ場合，入力キューだけでなくアプリケーション起動用論理端末が使用する出力キューにも，ディスクキューを割り当ててください。アプリケーション起動用論理端末が使用する出力キューにメモリキューを割り当てた場合，再開始時にメッセージが引き継がれないことがあります。

また，指定した時間が経過したあとに起動する（タイマ起動）アプリケーションを再開始時に引き継ぐ場合，さらにアプリケーション起動環境定義（mcftpsvr）の reruntm オペランドで yes を指定します。再開始時のタイマ起動の扱いについては，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 表 3-7　通信プロトコル対応製品と OpenTP1 再開始時のメッセージの扱い 1

| メッセージの種類 | 通信プロトコル対応製品 | | | |
|---|---|---|---|---|
| | TP1/NET/OSAS-NIF | TP1/NET/OSI-TP | TP1/NET/SLU - TypeP2 | TP1/NET/TCP/IP |
| 一方送信メッセージ | ○ | × | ○ | ○ |
| 問い合わせメッセージ | ○ | － | － | × |
| 応答メッセージ | ○ | － | － | × |
| ERREVT1 | ○ | × | ○ | ○ |
| 通信イベント（SERREVT，SCMPEVT を除く） | × | × | × | ○ |

（凡例）

○：OpenTP1 の再開始時に引き継がれます。

×：OpenTP1 の再開始時に破棄されます。

－：該当する通信プロトコル対応製品では使用できないメッセージです。

## 表 3-8　通信プロトコル対応製品と OpenTP1 再開始時のメッセージの扱い 2

| メッセージの種類 | 通信プロトコル対応製品 | | |
|---|---|---|---|
| | TP1/NET/User Agent | TP1/NET/UDP | TP1/NET/XMAP3 |
| 一方送信メッセージ | ○ | ○ | ○ |
| 問い合わせメッセージ | × | － | × |
| 応答メッセージ | × | － | × |
| ERREVT1 | △ | ○ | × |

| メッセージの種類 | 通信プロトコル対応製品 | | |
|---|---|---|---|
| | TP1/NET/User Agent | TP1/NET/UDP | TP1/NET/XMAP3 |
| 通信イベント（SERREVT，SCMPEVT を除く） | × | ○ | × |

（凡例）

○：OpenTP1 の再開始時に引き継がれます。

×：OpenTP1 の再開始時に破棄されます。

△：問い合わせ型（request）または受信型（receive）の論理端末の場合，OpenTP1 の再開始時に引き継がれます。応答型（reply）の論理端末の場合，OpenTP1 の再開始時に破棄されます。

－：該当する通信プロトコル対応製品では使用できないメッセージです。

## (3) メモリキューでの縮退運転

ディスクキューを使用する場合には，OpenTP1 の開始時に何らかの理由でディスクキューが使えないとき，メモリキューを代わりに使って開始できます。これをメモリキューでの**縮退運転**といいます。メモリキューで縮退運転するかどうかを，入力キューと出力キューそれぞれの拡張予約定義で指定します。メモリキューで縮退運転しない場合は，ディスクキューに格納されるメッセージが廃棄されます。このとき，メッセージ廃棄通知イベント（ERREVT2）に対応する MCF イベント処理用 MHP があれば，起動されます。

メモリキューで縮退運転する場合は，メッセージ（KFCA11065-W または KFCA11066-W）が出力されます。

メモリキューで縮退して OpenTP1 を開始させる場合，メモリキューはオンライン停止後の全面回復時にメッセージを回復できない点に注意してください。

オンライン中にディスクキューが使用できなくなったときは，メモリキューで縮退運転できません。

縮退運転の原因と復旧方法については，マニュアル「OpenTP1 運用と操作」を参照してください。

## (4) ディスクキューのメッセージを保持

ディスクキューを使用した場合には，読み出されたあとの再読み出しに備えて，メッセージを残しておけます。再読み出しに備えて残しておくメッセージを，**保持メッセージ**といいます。保持メッセージ数は，メッセージキューサービス定義で指定します。

メモリキューを使用した場合には，メッセージを残しておけません。

ディスクキューごとに，取り出し待ちメッセージの数と保持メッセージの数が，ユーザが定義した一定の割合を超えたとき，ディスクキューの使用率を警告するメッセージを出力できます。警告メッセージを出力するディスクキューの使用率（使用容量警告率　%で表示）は，メッセージキューサービス定義で指定します。

## (5) ディスクキュー満杯時の処理

ディスク入力キューに空きがなくなった場合は，到着したメッセージを廃棄します。メッセージ廃棄通知イベント（ERREVT2）に対応する MCF イベント処理用 MHP があれば，これを起動します。ディスク出力キューに空きがなくなった場合は，メッセージを送信する関数がエラーリターンします。

## (6) 入力キューと出力キューの保留

入力キューおよび出力キューへの書き込みや読み込みは，それぞれ mcfthldiq コマンド，mcfthldoq コマンドで一時的に停止できます。これを**保留**といいます 。

保留の目的は次のとおりです。

- オンライン中にユーザサーバの入れ替えができます。
- 入力キューまたは出力キューの内容を複写できます。
- 入力キューまたは出力キューに滞留しているメッセージを削除できます。

保留には入力キューまたは出力キューへの書き込みの保留（入力保留）と入力キューまたは出力キューからの読み込みの保留（スケジュール保留）があります。

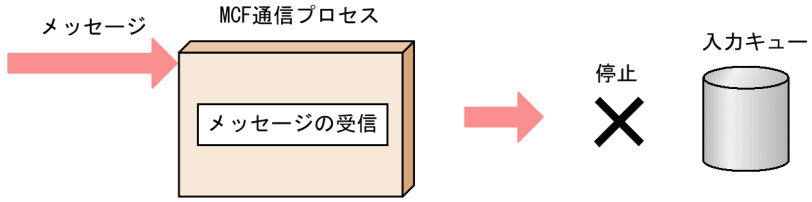
保留状態は，入力キューは mcftrlsiq コマンド，出力キューは mcftrlsoq コマンドで解除できます。また，MCF マネジャ定義の状態引き継ぎ定義（mcfmsts）や MCF 通信構成定義の状態引き継ぎ定義（mcftsts）を指定すれば，OpenTP1 が異常終了した場合などの再開始時にも引き継ぐことができます。

保留中に OpenTP1 を終了した場合，OpenTP1 を強制停止します。OpenTP1 を終了する前に必ず保留を解除してください。

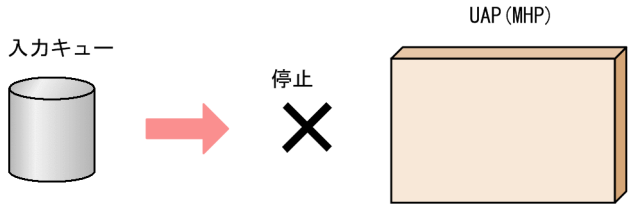
入力保留とスケジュール保留を次の図に示します。

## 図 3-36　入力保留とスケジュール保留

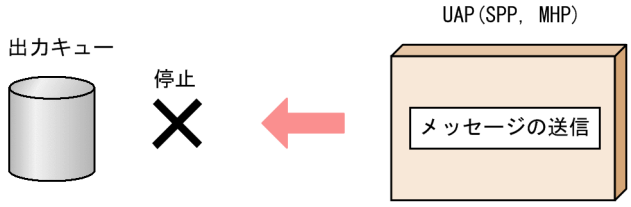
●入力キューの入力保留



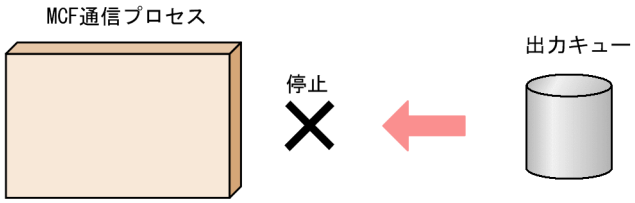

●入力キューのスケジュール保留




●出力キューの入力保留




●出力キューのスケジュール保留




保留中に受信または送信したメッセージに対する処理を次の表に示します。

## 表 3-9　保留中に受信または送信したメッセージに対する処理

| キュー種別 | 保留種別 | 受信または送信したメッセージに対する処理 | |
|---|---|---|---|
| | | 保留したあとに受信または送信したメッセージ | 保留する前にキューに格納されていたメッセージ |
| 入力キュー | 入力保留 | 入力保留が解除されるまで MCF 通信プロセスまたはアプリケーション起動プロセスの動作を停止します。 | 該当する MHP を正常に起動します。 |
| | スケジュール保留 | 入力キューにメッセージを格納し，スケジュール保留が解除されるまで MHP がメッセージを取り出しません。 | スケジュール保留が解除されるまで MHP がメッセージを取り出しません。 |
| | 入力保留とスケジュール保留 | 入力保留が解除されるまで MCF 通信プロセスまたはアプリケーション起動プロセスの動作を停止します。 | スケジュール保留が解除されるまで MHP がメッセージを取り出しません。 |

| キュー種別 | 保留種別 | 受信または送信したメッセージに対する処理 | |
|---|---|---|---|
| | | 保留したあとに受信または送信したメッセージ | 保留する前にキューに格納されていたメッセージ |
| 出力キュー※1 | 入力保留 | 入力保留が解除されるまで UAP の動作を停止します。※2 | MCF 通信プロセスが正常にメッセージを処理します。 |
| | スケジュール保留 | 出力キューにメッセージを格納し，スケジュール保留が解除されるまで MCF 通信プロセスがメッセージを取り出しません。 | スケジュール保留が解除されるまで MCF 通信プロセスがメッセージを取り出しません。 |
| | 入力保留とスケジュール保留 | 入力保留が解除されるまで UAP の動作を停止します。※2 | スケジュール保留が解除されるまで MCF 通信プロセスがメッセージを取り出しません。 |

注※1

同期型メッセージの場合，出力キューへの書き込みや読み込みを保留できません。保留状態に関わらず出力キューにメッセージを格納し，MCF 通信プロセスが正常にメッセージを処理します。

注※2

出力キューの入力保留の場合，UAP の動作を停止するタイミングを同期点処理で待ち合わせるか関数発行時に待ち合わせるかを選択できます。詳細については，マニュアル「OpenTP1 システム定義」の UAP 共通定義（mcfmuap）を参照してください。

**注意事項**

- 入力キューを保留して，オンライン中にユーザサーバを入れ替える場合，スケジュール保留だけを行ってください。入力保留を行った場合，MCF 通信プロセスまたはアプリケーション起動プロセスの動作が停止することで，他のユーザサーバでのメッセージ送受信ができなくなることがあります。
- 保留状態を再開始時に引き継ぐ指定をした場合，次のことをするときは，mcfthldiq コマンドや mcfthldoq コマンドに-r オプションを指定することを推奨します。

  入力キューまたは出力キューの内容の複写

  滞留しているメッセージの削除

  オンライン中のユーザサーバの入れ替え

## 3.3.9　アプリケーションプログラムのメッセージ送受信

### (1)　UAP のメッセージ受信

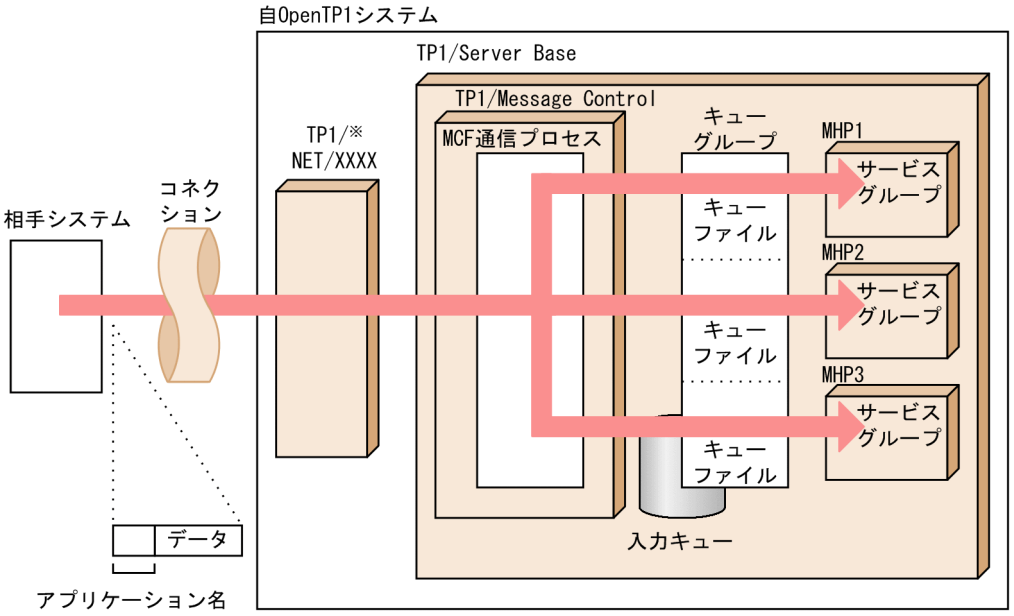
OpenTP1 は，相手システムからコネクションを通してメッセージを受信します。**コネクション**とは，相手システムと MCF の論理的な通信路のことです。OpenTP1 が受信したメッセージは，**MCF 通信プロセス**が管理します。MCF は，メッセージに組み込まれたアプリケーション名をサービスグループ名とサービス名に変換し，受信メッセージを入力キューに書き込みます。

入力キューは，サービス要求としての入力を管理する待ち行列です。ディスクキューに書き込むメッセージの場合，サービスグループ単位にキューを割り当てます。ディスクキューのうち，個々のサービスグループに割り当てた部分を**キューファイル**といいます。対応するサービスグループの名称が**キューファイル名**になります。メモリキューに書き込むメッセージの場合も同様に，サービスグループ単位でキューを管理します。

MCF は，入力キューに受信メッセージの最終セグメントを書き込んだ時点で，サービス要求を該当する MHP のスケジュールキューに登録します。スケジュールされた MHP は，受信メッセージをセグメント単位に dc_mcf_receive 関数で受信します。

UAP のメッセージ受信の概要を次の図に示します。

図 3-37　UAP のメッセージ受信の概要




注※　通信プロトコル対応の製品です。

## (2)　UAP のメッセージ送信

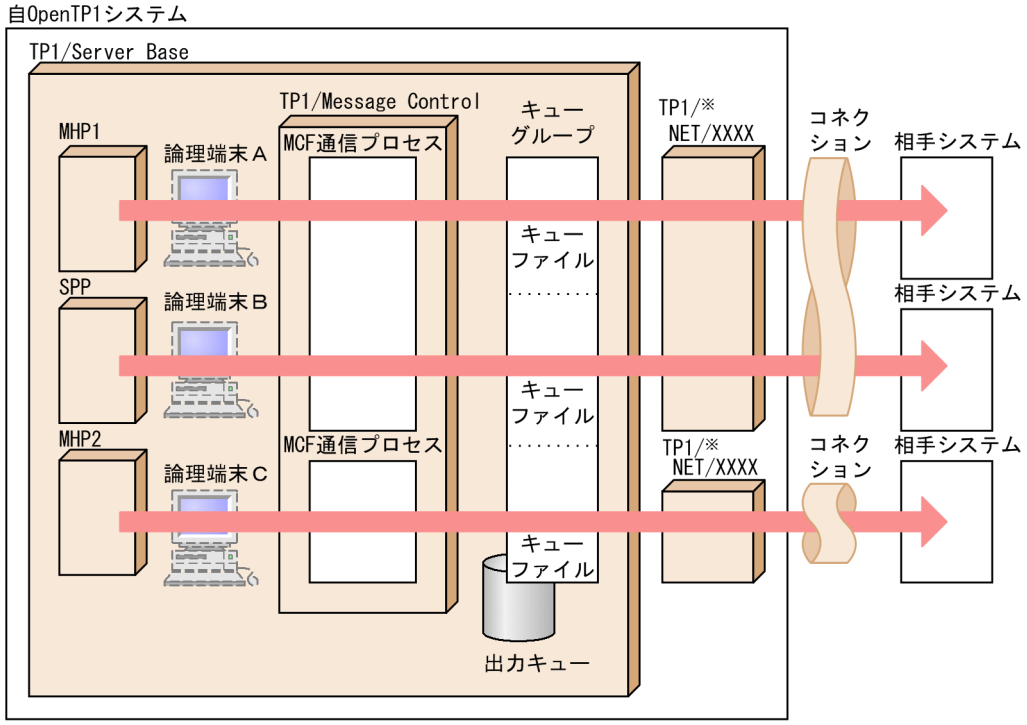
MCF は，メッセージに組み込まれた**論理端末名称**を基に，送信メッセージを出力キューに書き込みます。論理端末名称とは，マルチウィンドウを持つワークステーションの一つのウィンドウや，プリンタなどの端末の実体ごとに，ユーザが任意に付ける名称です。OpenTP1 は，端末の実体を論理端末名称を持つ**論理端末**として識別します。論理端末名称は，論理端末定義で指定します。

ディスクキューを使用する送信メッセージの場合，論理端末ごとにキューを割り当てます。ディスクキューのうち，個々の論理端末に割り当てた部分を**キューファイル**といいます。対応する論理端末の名称が**キューファイル名**になります。メモリキューを使用する送信メッセージの場合も同様に，論理端末ごとにキューを管理します。

プロトコル対応に作成した MCF 通信プロセスが，セグメント単位で他システムにメッセージを送信します。

UAP のメッセージ送信の概要を次の図に示します。

図 3-38　UAP のメッセージ送信の概要




注※　通信プロトコル対応の製品です。

## (a)　一方送信メッセージの送信

MHP と SPP は，dc_mcf_send 関数で相手システムへ一方送信メッセージを送信できます。送信先として論理端末名称を指定します。

## (b)　問い合わせに対する応答メッセージの送信

MHP は，相手システムからの問い合わせメッセージを dc_mcf_receive 関数で受信したあと，応答メッセージを dc_mcf_reply 関数で問い合わせ発生元論理端末へ送信できます。

## (c)　問い合わせメッセージの送信

MHP と SPP は，相手システムへ問い合わせメッセージを dc_mcf_send 関数で送信できます。

## (d)　同期型のメッセージの送信

MHP と SPP は，相手システムへ同期型のメッセージを dc_mcf_sendsync 関数，または dc_mcf_sendrecv 関数で送信できます。

dc_mcf_sendsync 関数は，相手システムにメッセージを送信し終わるまで UAP の処理を止めるため，相手システムの UAP 処理と同期がとれます。

dc_mcf_sendrecv 関数は，相手システムにメッセージを送信し終わってから，応答を自システムで受信するまで UAP の処理を止めるため，相手システムの UAP 処理と同期がとれます。

## (3) UAP からのアプリケーションの起動

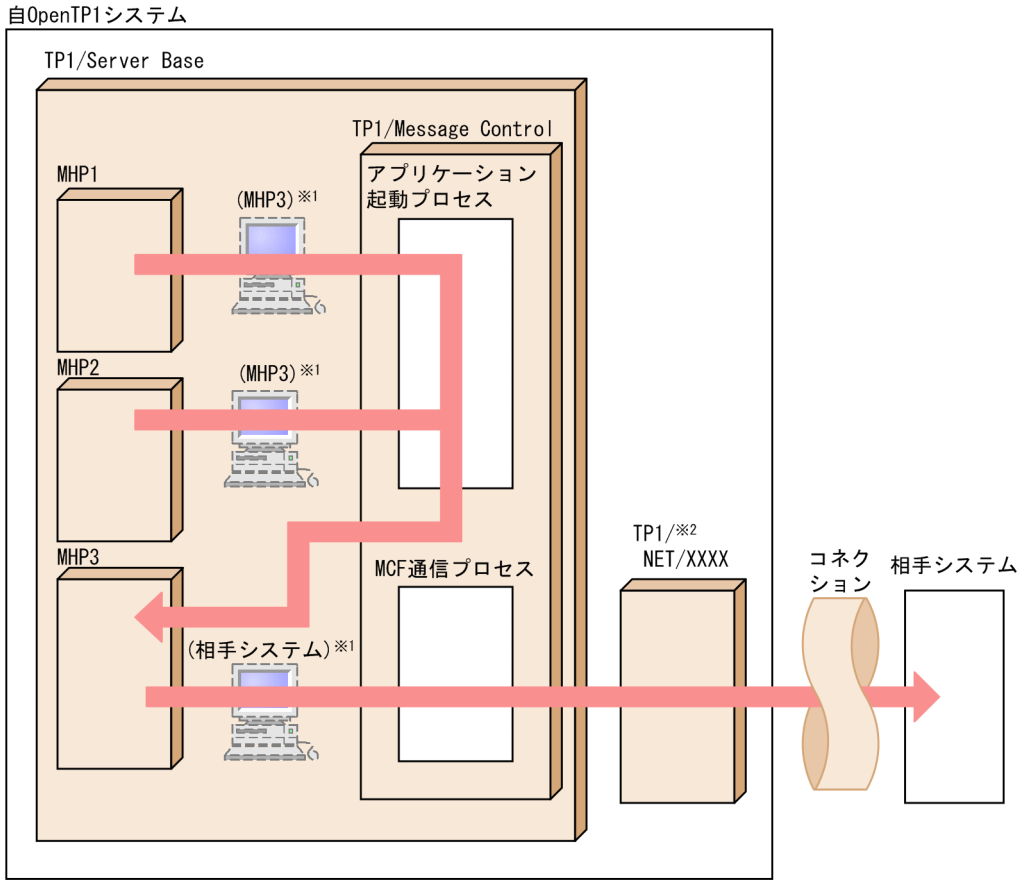
アプリケーション起動機能を使えば，メッセージの編集・送信用の MHP を作成することで，メッセージ送信を**一元化**できます。

応答メッセージを編集して送信する MHP を作成すれば，問い合わせメッセージを受信した MHP に代わって，問い合わせ発生元論理端末への応答を任せることができます。また，一方送信メッセージを編集して送信する MHP を作成すれば，MHP または SPP からこの MHP を起動することで，メッセージ送信を任せることもできます。どちらの場合も，送信要求元の UAP から **dc_mcf_execap 関数**を呼び出すことでメッセージの送信を一元化します。

dc_mcf_execap 関数は非同期型の通信関数です。MCF は，メッセージを編集して送信する MHP を，関数を呼び出した MHP や SPP のサービス正常終了直後，または指定された時間が経過したあとに起動（タイマ起動）します。このため，送信要求元の MHP や SPP と，メッセージを編集して送信する MHP とは，別のグローバルトランザクションになります。送信要求元の MHP や SPP は，送信先の論理端末名称などの必要なデータを MCF 経由で，メッセージを編集して送信する MHP に渡します。なお，タイマ起動の場合には，mcfadltap コマンドまたは dc_mcf_adltap 関数で起動要求をキャンセルできます。

メッセージ送信の一元化を次の図に示します。

図 3-39　メッセージ送信の一元化




注※1　（　）内は，論理端末名称を示します。
注※2　通信プロトコル対応の製品です。

## (4) OpenTP1 終了時のメッセージの扱い

### (a) 未処理メッセージの監視

MCF は，終了処理の開始から出力キューのメッセージがなくなるまでの時間を監視します。出力キューのメッセージが処理されないで，終了処理に時間が掛かることを防止するためです。出力キューのメッセージがなくなるまでの監視時間を，**未処理送信メッセージ滞留時間**といいます。監視時間を超えても出力キューに残っているメッセージは廃棄し，未処理送信メッセージ廃棄イベント（ERREVTA）に対応する MCF イベント処理用 MHP が作成してあれば，これを起動します。ただし，応答メッセージ，閉塞されている論理端末のメッセージおよびタイマ起動によるアプリケーション起動のメッセージの場合，未処理送信メッセージ滞留時間の対象にはなりません。システム終了時に滞留しているメッセージは無条件に削除されます。このとき，閉塞されている論理端末のメッセージおよびタイマ起動によるアプリケーション起動のメッセージの場合は ERREVTA で通知しますが，応答メッセージの場合は ERREVTA で通知されません。

また，MCF は終了処理の開始から入力キューのメッセージがなくなるまでの時間を監視します。MHP で異常な処理をしたために入力キューのメッセージが処理されないで，終了処理が終わらないことを防止するためです。また，正常終了時にサービスグループの閉塞によって滞留しているメッセージは破棄され ERREVT2 で通知されます。入力キューのメッセージがなくなるまでの監視時間を，**未処理受信メッセージ滞留時間**といいます。未処理受信メッセージ滞留時間を過ぎても，入力キューにメッセージが残っている場合には，MHP に何らかの異常があるものと見なして，MCF を異常終了させます。

未処理送信メッセージ滞留時間と未処理受信メッセージ滞留時間は，MCF 通信構成定義で指定します。

### (b) OpenTP1 の終了モードと未処理メッセージの監視との関係

OpenTP1 の終了モードには，**正常終了**，**強制正常終了**，**計画停止 A**，**計画停止 B**，および**強制停止**があります。

**正常終了**および**強制正常終了**のときには，新たなメッセージの受け付けを禁止したあと，**未処理受信メッセージ滞留時間**と**未処理送信メッセージ滞留時間**を監視して，入力キューと出力キューのメッセージがなくなるまで処理を続けます。

**計画停止 A** のときには，新たなメッセージの受け付けを禁止したあと，入力キューのメッセージがなくなるまで処理を続けます。出力キューにメッセージが残っていても終了します。出力キューに残ったメッセージは，次回の再開始時に処理します。ただし，メモリ出力キューのメッセージ，および MCF 通信構成定義の mcftpsvr コマンドの-o オプションに reruntm=no を指定している場合のタイマ起動要求のメッセージは回復できません。計画停止 A のときには，**未処理受信メッセージ滞留時間**だけを監視します。

**計画停止 B** のときには，OpenTP1 が現在実行しているサービスだけを完了させて終了します。入力キューや出力キューのメッセージは処理されないままになるため，未処理メッセージの滞留時間を監視しません。入出力キューに残ったメッセージは，次回の再開始時に処理します。ただし，メモリキューのメッセージ，および MCF 通信構成定義の mcftpsvr コマンドの-o オプションに reruntm=no を指定している場合のタイマ起動要求のメッセージは回復できません。

**強制停止**のときには，OpenTP1 が現在実行しているサービスの完了を待たないで終了します。未処理メッセージの滞留時間も監視しません。入出力キューに残ったメッセージは，次回の再開始時に処理します。ただし，メモリキューのメッセージ，および MCF 通信構成定義の mcftpsvr コマンドの-o オプションに reruntm=no を指定している場合のタイマ起動要求のメッセージは回復できません。

相手システムにメッセージを送信したあとに，自システムがオンラインダウン，または強制停止し，送信済みのメッセージが送信完了扱いとならない場合，再開始後に再度，メッセージが送信されることがあります。そのため，相手システムとの間で，二重送信の扱いについて考慮する必要があります。

OpenTP1 の終了と入出力キューのメッセージに対する処理を次の表に示します。

表 3-10　OpenTP1 の終了と入出力キューのメッセージに対する処理

| 終了モード | 入力キュー中のメッセージ | 出力キュー中のメッセージ |
|---|---|---|
| 正常終了，強制正常終了 | すべて処理する | すべて処理する |
| 計画停止 A | すべて処理する | 残す |
| 計画停止 B | 残す | 残す |
| 強制停止 | 残す | 残す |

## (5)　メッセージ出力通番

OpenTP1 は，UAP からの送信メッセージに対して，メッセージ種別（問合せ応答メッセージ，一般分岐メッセージ，および優先分岐メッセージ）ごとに一連の番号を付けられます。この一連の番号を**メッセージ出力通番（出力通番）**と呼びます。

出力通番の目的は次のとおりです。

- メッセージログやジャーナルの出力通番を確認して，システム障害時または通信障害時のメッセージの送信抜けや二重の送信を知ることができます。
- メッセージを再送するときのキーにできます。

dc_mcf_send 関数または dc_mcf_reply 関数を呼び出すときに，出力通番を付けるかどうかを指定できます。出力通番を付ける指定をした dc_mcf_send 関数または dc_mcf_reply 関数を呼び出した場合，OpenTP1 はメッセージに出力通番を付けたあと，メッセージを出力キューに書き込みます。

また，UAP に送信メッセージの通番編集 UOC を組み込んだ場合，出力通番を付ける指定をした dc_mcf_send 関数または dc_mcf_reply 関数の先頭セグメントの送信時に，OpenTP1 は送信メッセージの通番編集 UOC を呼び出します。この UOC では，先頭セグメントの任意の位置に出力通番を設定するなどの処理ができます。

出力通番は，OpenTP1 の再開始時に引き継がれます。

出力キュー内の先頭と最後の未送信メッセージに付けられている出力通番は，mcftlsle コマンドで表示できます。

## 3.3.10　メッセージを送信する順序とアプリケーションを起動する順序

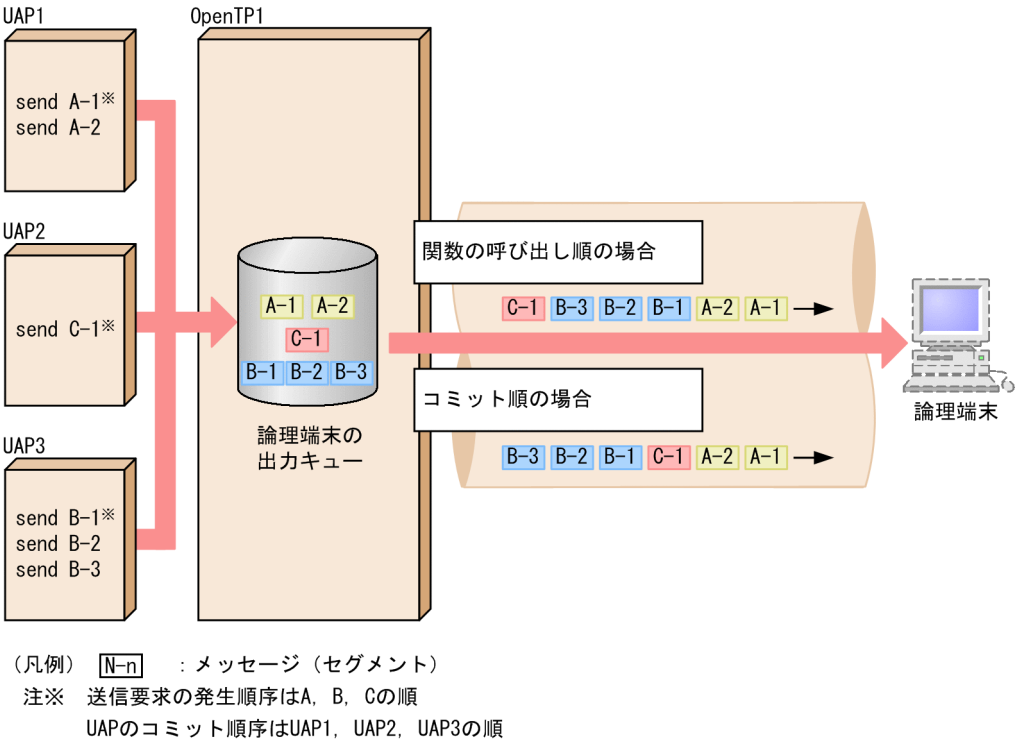
OpenTP1 では，相手システムへのメッセージの送信と dc_mcf_execap 関数によるアプリケーションの起動を次のようにスケジュールします。

### (1)　論理端末にメッセージを送信する順序

OpenTP1 では，コネクションで接続されている論理端末間で，**先入れ先出し（FIFO）**とメッセージの優先度に従って送信しています。

一つのメッセージが複数のセグメントから構成される場合，各セグメントを連続して送信します。一つのメッセージを送信後，次のメッセージを論理端末に送信します。複数の UAP からの送信要求に対する処理順序を関数の呼び出し順とするかコミット順とするかは，UAP 共通定義（mcfmuap -c）の order オペランドで指定します。論理端末への送信順序を次の図に示します。

図 3-40　論理端末への送信順序




### (2)　メッセージの種類と送信優先度の関係

UAP が送信するメッセージの種類によって，優先度が異なります。一つの論理端末に対して異なる種類のメッセージが送信待ちの場合，優先度の高いものから先に送信します。同じ種類のメッセージの場合は，FIFO で順次送信します。

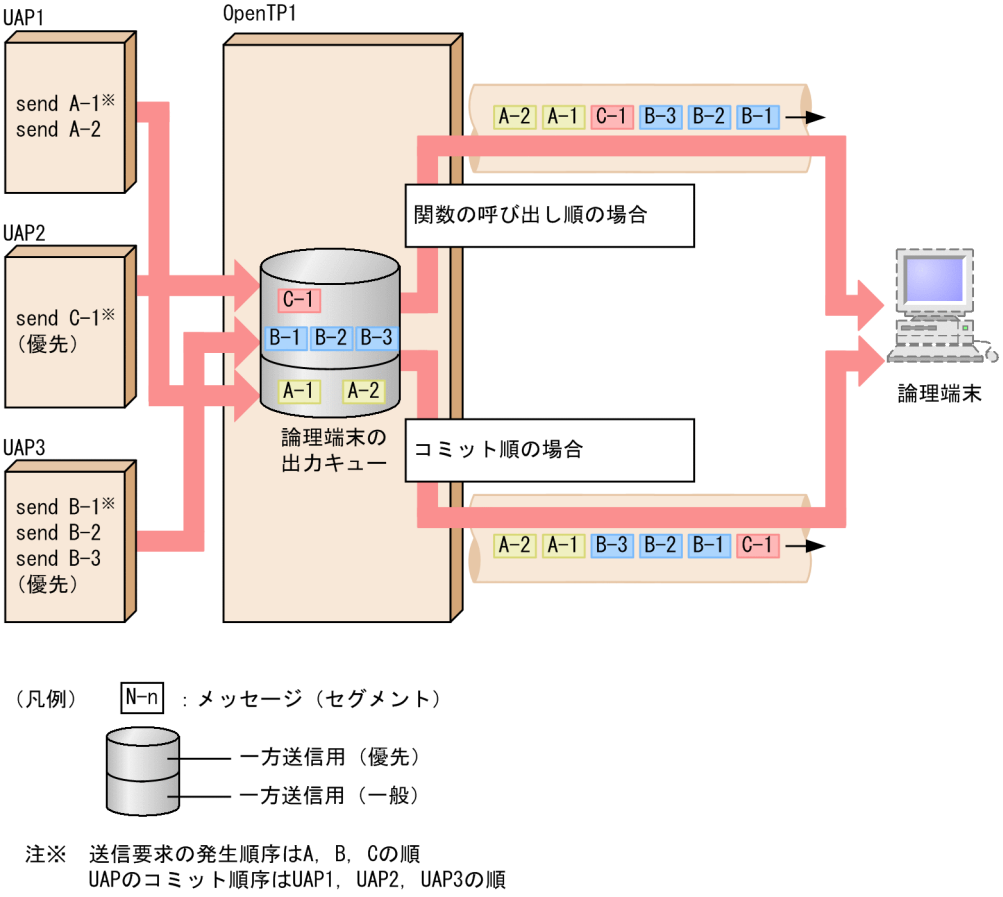
メッセージ種別ごとの**送信優先度**を次の表に示します。

表 3-11　メッセージ種別ごとの送信優先度

| メッセージ種別 | 送信優先度 |
|---|---|
| 応答メッセージ | 高 |
| 一方送信メッセージ（優先） | 中 |
| 一方送信メッセージ（一般）／問い合わせメッセージ | 低 |

OpenTP1 では，FIFO による送信順序と，送信優先度による送信順序とを組み合わせて送信しています。FIFO と送信優先度の組み合わせ送信順序を次の図に示します。

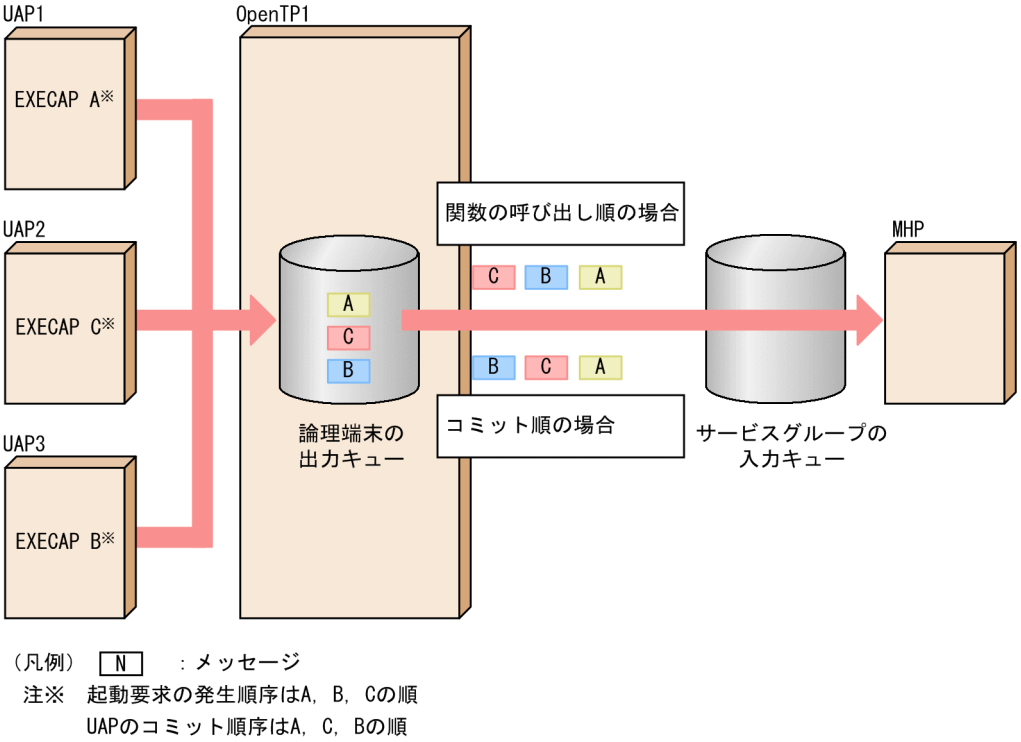
図 3-41　FIFO と送信優先度の組み合わせ送信順序




## (3)　dc_mcf_execap 関数でアプリケーションを起動する順序

複数の UAP からの dc_mcf_execap 関数に対する処理順序を関数の呼び出し順とするかコミット順とするかは，UAP 共通定義（mcfmuap -c）の order オペランドで指定します。アプリケーションの起動順序を次の図に示します。

図 3-42　アプリケーションの起動順序




（凡例）　N　：メッセージ
注※　起動要求の発生順序はA，B，Cの順
　　UAPのコミット順序はA，C，Bの順

## 3.3.11　オンライン中の MCF 通信サービスの部分入れ替え

次の通信プロトコル対応製品を使用する場合，OpenTP1 を停止しないで MCF 通信サービスのメイン関数，UOC，およびライブラリを変更できます。

- TP1/NET/TCP/IP
- TP1/NET/SLU - TypeP2
- TP1/NET/OSAS-NIF

OpenTP1 を停止しないで MCF 通信サービスのメイン関数，UOC，およびライブラリを変更するには，変更対象の MCF 通信サービスだけを部分的に停止し，該当ファイルを入れ替えます。この方法を利用すると，変更対象の MCF 通信サービス以外を停止させることなく変更できるので，システムの連続運転が可能になります。入れ替えられるファイルは次のファイルです。

- MCF 通信サービスのロードモジュール（メイン関数）
- MCF 通信サービスが使用しているユーザ作成のライブラリ（UOC）
- 対象の通信プロトコル製品が提供しているライブラリ

MCF 通信サービスを部分的に入れ替えるには，MCF 通信サービス管理の運用コマンド（mcftlscom，mcftstart，mcftstop）を使用します。これらの運用コマンドの詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

## 3.3.12 MCF 構成変更再開始機能

OpenTP1 の再開始は，前回のオンライン時のシステムの構成要素と再開始時のシステムの構成要素とが同一であることを前提としています。このため，オフライン中に OpenTP1 のシステム定義や OpenTP1 ファイルシステムを変更してオンラインを再開始した場合，OpenTP1 が正常に開始できないことがあります。

MCF 構成変更再開始機能は，前回のオンライン停止時に入力キュー（ディスクキュー）上に残っていた未処理受信メッセージおよび出力キュー（ディスクキュー）上に残っていた未送信メッセージを次回のオンラインに引き継ぎつつ，OpenTP1 ファイルシステムやキューグループの構成変更，および論理端末やアプリケーションの構成変更をできるようにする機能です。

MCF 構成変更再開始機能を使用するためには，TP1/Message Control - Extension 1 が必要です。

また，通信プロトコル製品には次のどちらかを使用できます。

- TP1/NET/TCP/IP
- TP1/NET/XMAP3

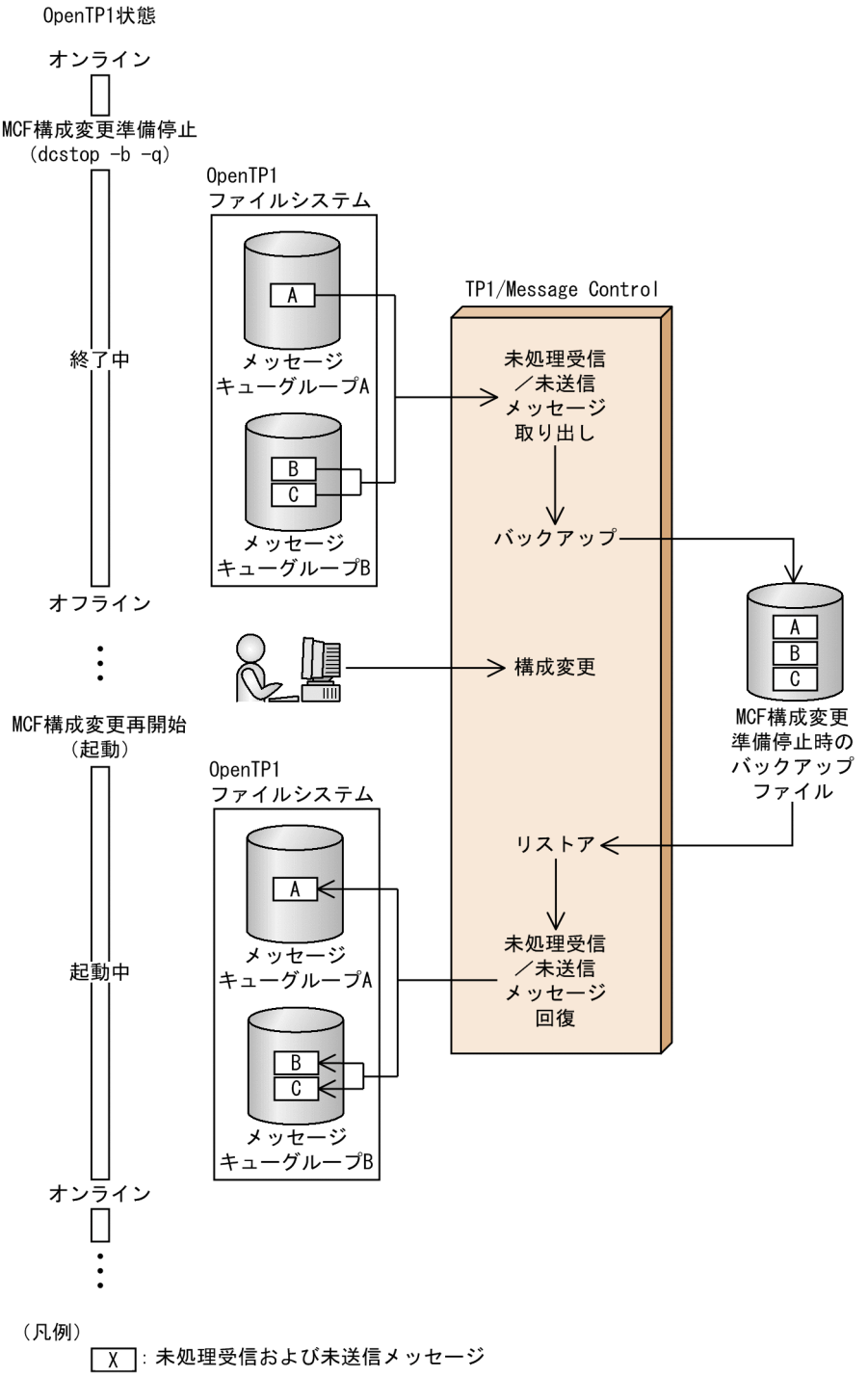
MCF 構成変更再開始機能を使用する場合，システム構成を変更するモード（MCF 構成変更準備停止）で OpenTP1 を停止します。システム停止後，システムの構成変更を実施し，構成変更が反映されるモード（MCF 構成変更再開始）でオンラインシステムを再開始します。

MCF 構成変更準備停止によるオンライン終了時に，未処理受信メッセージおよび未送信メッセージがディスクキューの入出力キュー上に残っている場合，未処理受信メッセージおよび未送信メッセージをキューから取り出し，MCF 構成変更準備停止時のバックアップファイルに格納します。

MCF 構成変更再開始が行われたときに，自動的に MCF 構成変更準備停止時のバックアップファイルから元々格納されていたキューに未処理受信メッセージおよび未送信メッセージをリストアします。

MCF 構成変更再開始機能使用時の流れを次の図に示します。

## 図 3-43　MCF 構成変更再開始機能使用時の流れ




MCF 構成変更再開始機能の詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

## 3.4　アプリケーションプログラムのスケジュール

SPP，または MHP へのサービス要求を効率良く処理するため，スケジュールとプロセスを制御する機能を**スケジュール機能**といいます。

### 3.4.1　SPP のスケジュール

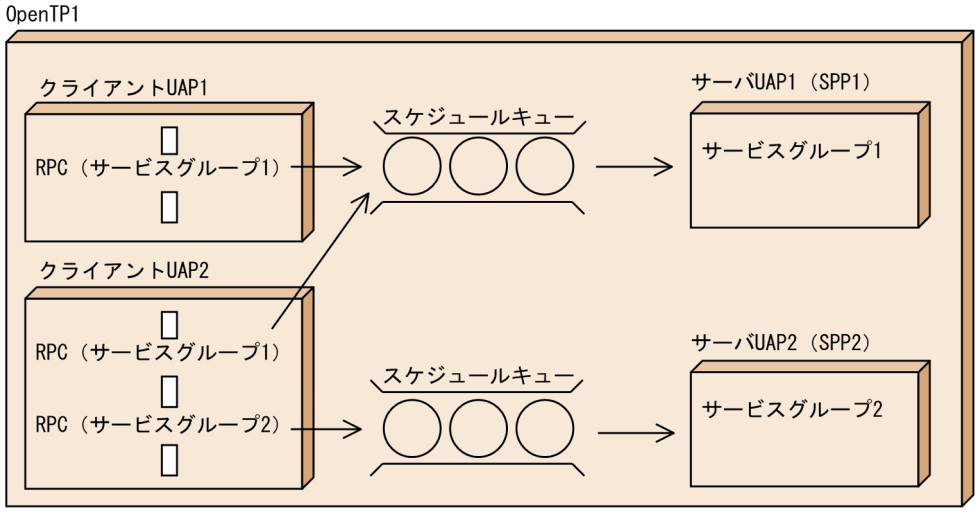
#### (1)　SPP のスケジュールの方法

OpenTP1 は，SPP のサービスグループごとに**スケジュールキュー**を作成してスケジュールします。クライアント UAP が RPC で指定したサービスグループ名とサービス名を基に，SPP へのサービス要求をスケジュールキューに登録します。登録したサービス要求は，**先入れ先出し**でスケジュールキューから取り出します。

クライアント UAP が指定した SPP のサービスグループが，ほかのノードにある場合でも，スケジュールできます。

SPP のスケジュールを次の図に示します。

図 3-44　SPP のスケジュール




#### (2)　サービス要求のスケジュールプライオリティの設定

クライアント UAP からのサービス要求に，優先順位（スケジュールプライオリティ）を付けられます。サービス要求のスケジュールプライオリティは，クライアント UAP で RPC を使う直前に，dc_rpc_set_service_prio 関数で宣言します。ここで宣言したスケジュールプライオリティに従って，サーバ UAP 側のスケジュールサービスでスケジュールします。

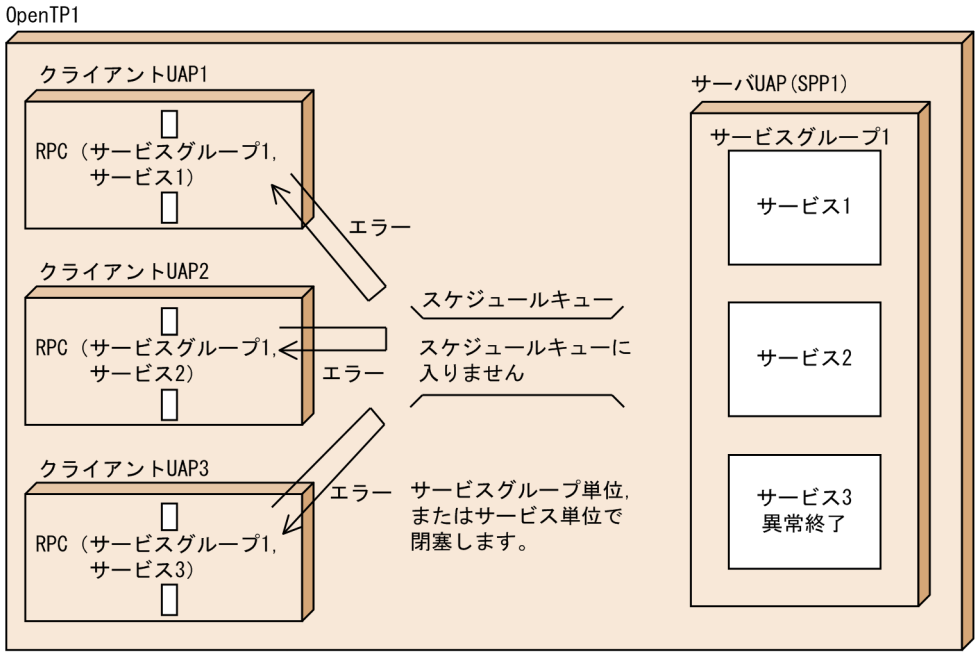
クライアント UAP のスケジュールプライオリティの指定に従うかどうかは，サーバ UAP 側のユーザサービス定義で指定します。

## (3) SPP のスケジュールの閉塞

サービス実行中の SPP が異常終了したときに，SPP のスケジュールを閉塞するかどうかをユーザサービス定義に指定します。

SPP のスケジュールの閉塞を次の図に示します。

図 3-45 SPP のスケジュールの閉塞




### (a) スケジュールを閉塞する単位

スケジュールを閉塞する単位として，**サービスグループ単位**，**サービス単位**のどちらかを指定します。サービスグループ単位で閉塞すると指定した場合は，異常終了したサービスがある SPP をすぐに閉塞します。サービス単位で閉塞すると指定した場合は，該当のサービスのスケジュールだけを閉塞して，同じサービスグループに属するほかのサービスのスケジュールは閉塞しません。

### (b) SPP の異常終了時の処理

SPP の UAP プロセスが異常終了したときに閉塞すると指定した場合は，そのサービスのスケジュールを閉塞します。閉塞されたサービスにサービス要求が来た場合，サービス要求元のクライアント UAP にエラーリターンします。また，サービスを閉塞した時点ですでにスケジュールキューに入っていたサービス要求も，サービス要求元のクライアント UAP にエラーリターンします。

SPP の UAP プロセスが異常終了したときに閉塞しないと指定した場合は，閉塞しないでスケジュールを続けます。ただし，ユーザサービス定義に指定した時間内に 3 回以上異常終了した場合は，その SPP をサービスグループ単位またはサービス単位で閉塞します。

### (c) SPP の閉塞を解除する方法

SPP の閉塞を解除して再開始するときは，scdrles コマンドを実行します。

### (d) 再開始時の閉塞状態の扱い

オンライン停止後の全面回復時に，SPP の閉塞状態を引き継ぐかどうかを，SPP ごとにユーザサービス定義で指定します。オンライン停止時に閉塞状態だった SPP のうち，閉塞状態を引き継ぐ指定をした SPP は，全面回復後も閉塞状態で再開始されます。閉塞している SPP にサービスを要求しても，サービス要求元にエラーリターンされます。

閉塞状態を引き継がないと指定した SPP は，閉塞が解除された状態で再開始されて，要求されたサービスを実行します。

### (e) コマンドによる閉塞

scdhold コマンドと scdrles コマンドで，SPP をサービスグループ単位，またはサービス単位で閉塞/閉塞解除できます。

SPP の実行形式ファイルをオンライン中に入れ替える場合は，サービスグループ単位で閉塞します。そして閉塞済みの SPP を新しい SPP と入れ替えます。入れ替えが完了してから，scdrles コマンドで閉塞を解除すれば，新しい SPP を実行できます。

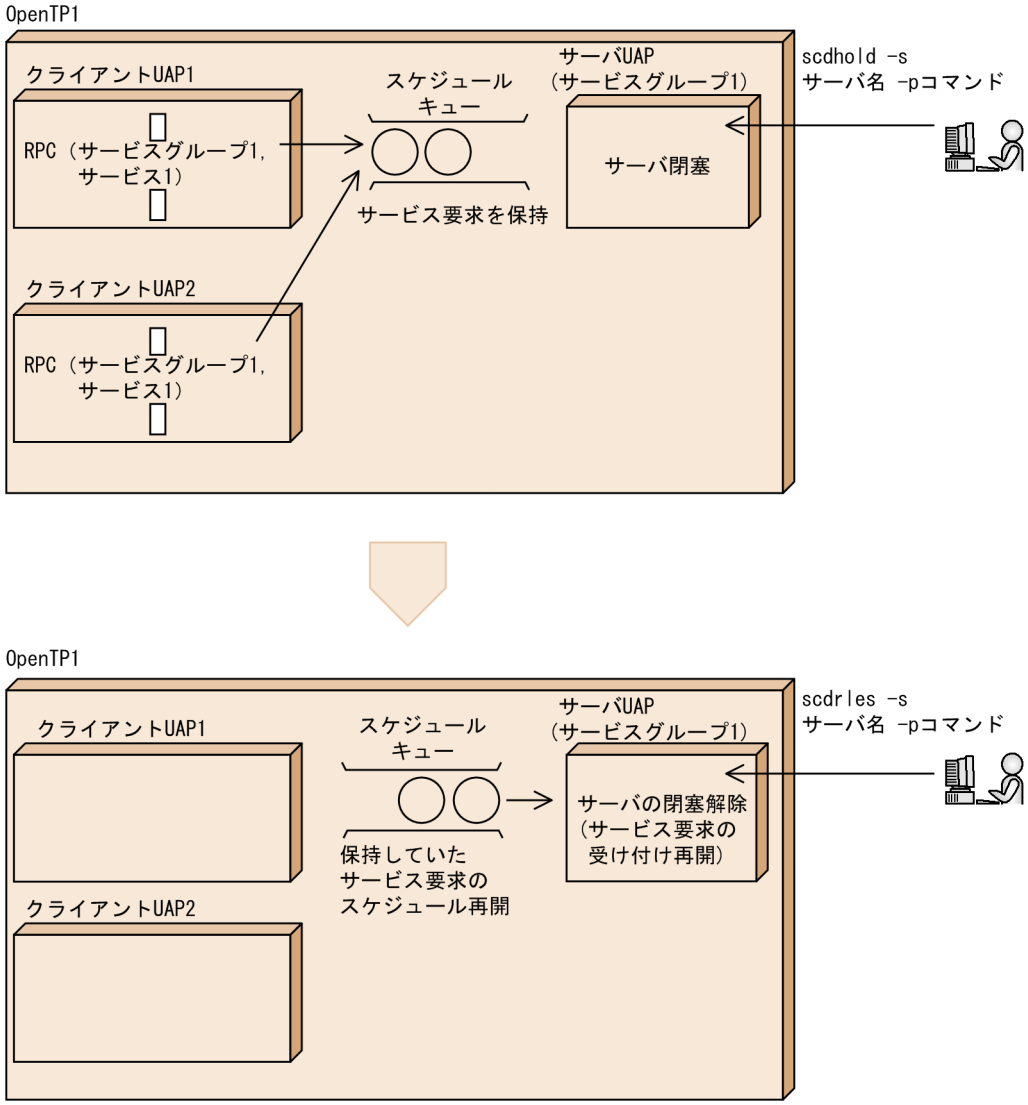
このように，コマンドで閉塞/閉塞解除することで，オンライン中に SPP を変更したり修正したりできます。

scdhold コマンドに指定するサーバ名に，-p オプションを付けて指定する（scdhold -s サーバ名 -p）と，SPP を閉塞したあとでも，該当する SPP へのサービス要求を受け付けられます。受け付けたサービス要求は，スケジュールキューに保持されます。該当する SPP がサービスを処理できる状態になったら，保持していたサービス要求のスケジュールを再開します。

サービス要求を保持できるのは，UAP のサービスグループ単位で閉塞する場合だけです。サービス単位で閉塞する場合は，コマンドの引数にサービス要求を保持する指定はできません。

スケジュールキューにサービス要求を保持する閉塞を次の図に示します。

図 3-46　スケジュールキューにサービス要求を保持する閉塞




## (4)　ソケット受信型サーバ

この機能は廃止したため使用できません。廃止した製品のバージョンについては，「はじめに」を参照してください。

## (5)　サービス単位のスケジュール制御

ユーザサービス定義またはユーザサービスデフォルト定義に scdsvcdef 定義コマンドを指定することで，SPP のスケジュールキューに対してサービス単位のスケジュール制御ができます。指定できるスケジュール制御は次のとおりです。なお，scdsvcdef 定義コマンドの詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

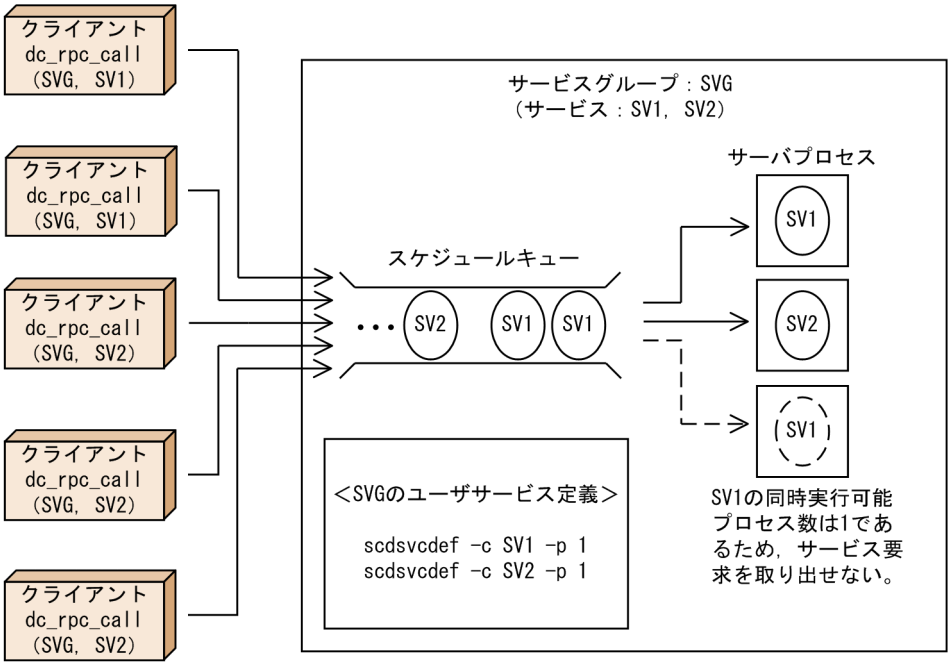
- サービスごとに同時実行可能なサービス数を指定する。
- サービスごとにキューイング可能なサービス要求数を指定する。
- サービスごとにキューイング可能なメッセージ格納バッファプール長を指定する。

### (a)　サービスごとに同時実行可能なサービス数を指定する

該当サービスを処理しているサーバプロセス数が，scdsvcdef 定義コマンドで指定した同時実行可能なサービス数に達している場合，同時実行可能なサービス数に達していないサービスへのサービス要求を，登録順およびスケジュールプライオリティに関係なくスケジュールします。

スケジュールキューに登録されているすべてのサービス要求が，同時実行可能なサービス数に達しているサービスへのサービス要求であった場合，実行中のサービス処理が終了するのを待ち合わせます。サービスごとの同時実行可能なサービス数を次の図に示します。

図 3-47　サービスごとの同時実行可能なサービス数




（凡例）

・・・　：スケジュールキューに登録されるサービス要求の省略を示します。

→　：キューイング，またはサービス要求の取り出しができたことを示します。

⇢　：キューイング，またはサービス要求の取り出しができなかったことを示します。

1. scdsvcdef 定義コマンドの-c オプション，および-p オプションに次のように指定して，サービス SV1 およびサービス SV2 の同時実行可能なサービス数を 1 に指定します。

```
scdsvcdef -c SV1 -p 1
scdsvcdef -c SV2 -p 1
```

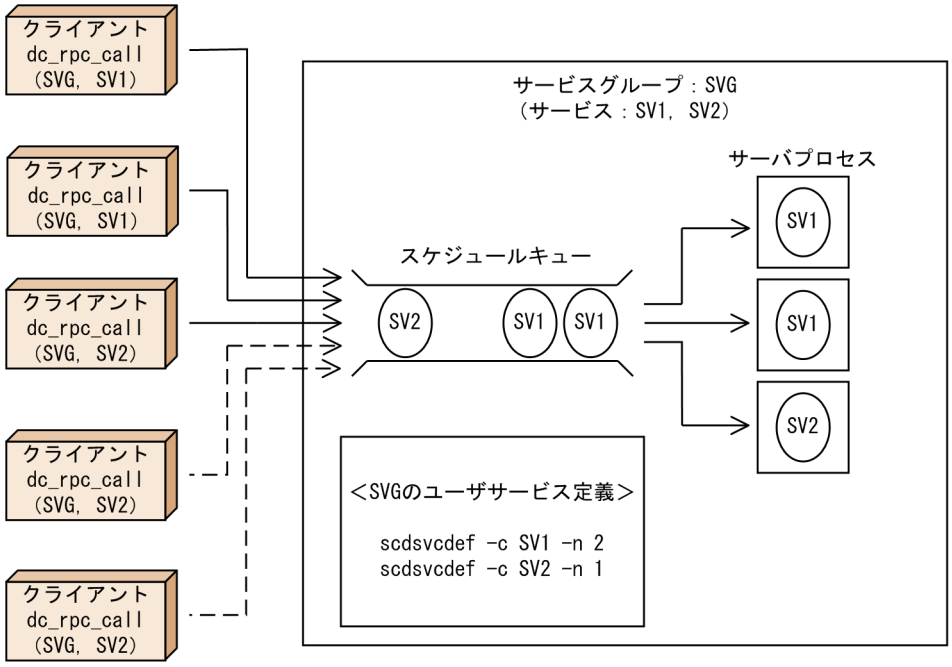
2. 1.でサービスの同時実行数を指定しているため，指定を超えたサービス要求は，スケジュールキューで待ち合わせます。

### (b)　サービスごとにキューイング可能なサービス要求数を指定する

該当サービスへのサービス要求数が，scdsvcdef 定義コマンドで指定したキューイング可能なサービス要求数に達している場合，スケジュールキューへは登録しないで，ほかの OpenTP1 ノードへの再スケジュールを試みます。

再スケジュールできない場合には，サービス要求元にメッセージ格納バッファ不足としてエラーリターンします。サービスごとのキューイング可能なサービス要求数を次の図に示します。

図 3-48　サービスごとのキューイング可能なサービス要求数




（凡例）

⟶ ：キューイング，またはサービス要求の取り出しができたことを示します。

⤏ ：キューイング，またはサービス要求の取り出しができなかったことを示します。

1. scdsvcdef 定義コマンドの-c オプション，および-n オプションに次のように指定して，サービス SV1 のキューイング可能なサービス要求数を 2 に，サービス SV2 のキューイング可能なサービス要求数を 1 に指定します。

```
scdsvcdef -c SV1 -n 2
scdsvcdef -c SV2 -n 1
```

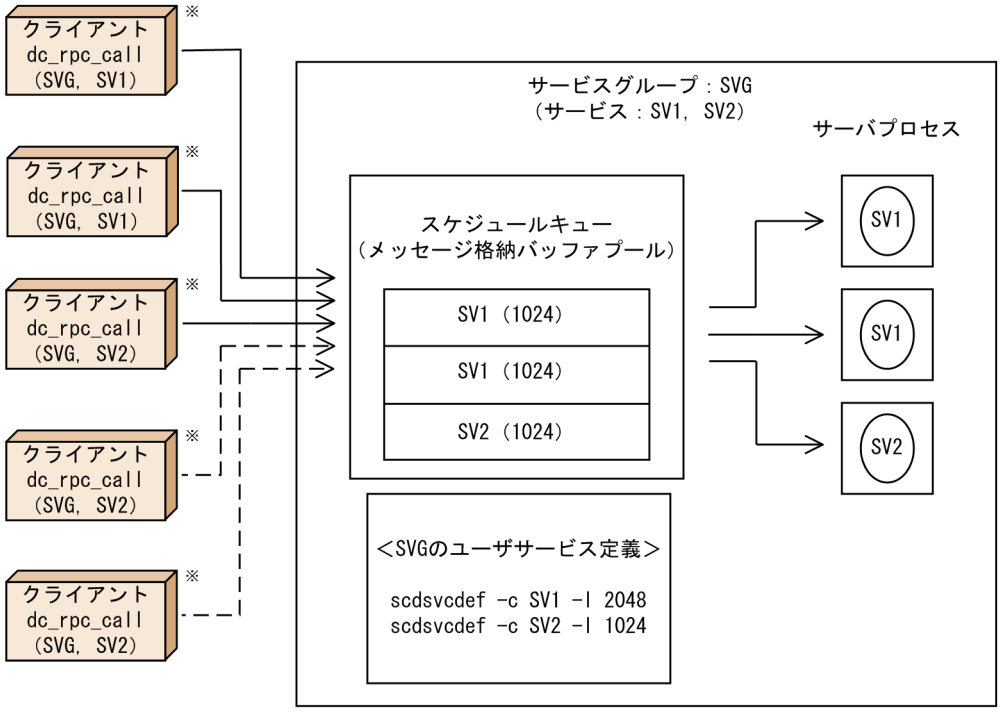
2. 1.でサービスのキューイング可能なサービス要求数を指定しているため，指定を超えたサービス要求はキューイングできません。

### (c)　サービスごとにキューイング可能なメッセージ格納バッファプール長を指定する

該当サービスへのサービス要求のメッセージ格納バッファプール長が，scdsvcdef 定義コマンドで指定したキューイング可能なメッセージ格納バッファプール長に達している場合，スケジュールキューへは登録しないで，ほかの OpenTP1 ノードへの再スケジュールを試みます。

再スケジュールできない場合には，サービス要求元にメッセージ格納バッファ不足としてエラーリターンします。サービスごとのキューイング可能なメッセージ格納バッファプール長を次の図に示します。

図 3-49　サービスごとのキューイング可能なメッセージ格納バッファプール長




（凡例）

→：キューイング，またはサービス要求の取り出しができたことを示します。

－＞：キューイング，またはサービス要求の取り出しができなかったことを示します。

注※

RP電文サイズは，1024バイトとします。

1. scdsvcdef 定義コマンドの-c オプション，および-l オプションに次のように指定して，サービス SV1 のキューイング可能なメッセージ格納バッファプール長を 2048 に，サービス SV2 のキューイング可能なメッセージ格納バッファプール長を 1024 に指定します。

```
scdsvcdef -c SV1 -l 2048
scdsvcdef -c SV2 -l 1024
```

2. 1.でサービスのキューイング可能なメッセージ格納バッファプール長を指定しているため，指定を超えたサービス要求はキューイングできません。

## (6)　スケジュール対象の選択

ノード間負荷バランス機能が使用できる環境で，クライアントからのサービス要求をどのノードのサーバ UAP にスケジューリングするかは，各ノードのサーバ UAP の状態によって判断します。次に示した状態のサーバ UAP はスケジューリングの対象外となります。ただし，通信先を指定（ノード識別子指定）した RPC の場合は該当しません。

- 正常停止した（または未起動）
- 閉塞している
- 負荷レベルが LEVEL2※

注※

負荷レベルが LEVEL2 のサーバ UAP でも，そのほかのサーバ UAP の状態が停止や閉塞の状態，またはすべてのサーバ UAP の負荷レベルが LEVEL2 の場合は，負荷レベルが LEVEL2 のサーバ UAP もスケジュール対象となります。

## 3.4.2　MHP のスケジュール

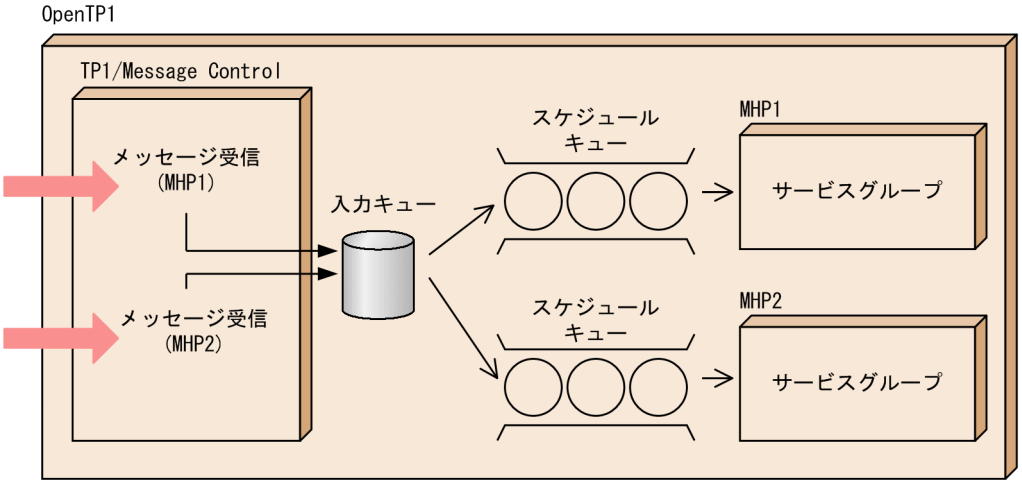
### (1)　MHP のスケジュールの方法

OpenTP1 は，MHP のサービスグループ単位に**スケジュールキュー**を作成してスケジュールします。まず，MCF はメッセージに組み込まれたアプリケーション名をサービスグループ名とサービス名に変換し，受信メッセージを入力キューに登録します。アプリケーション名と，サービスグループ名・サービス名の対応は，アプリケーション属性定義で指定します。MCF は，入力キューに受信メッセージの最終セグメントを格納した時点で，変換されたサービスグループ名とサービス名を基に，MHP へのサービス要求をスケジュールキューに登録します。登録したサービス要求は，スケジュールキューから**先入れ先出し**で取り出します。

MHP のサービスグループが，メッセージを受信した MCF と同じノードにないとスケジュールできません。

MHP のスケジュールを次の図に示します。

図 3-50　MHP のスケジュール




### (2)　MHP のスケジュールの閉塞

#### (a)　OpenTP1 のコマンドによる閉塞

コマンドで MHP のアプリケーション名を指定して，対応するアプリケーションを閉塞できます。MHP を閉塞，閉塞を解除するコマンドを次に示します。

- 閉塞するコマンド

mcfadctap コマンド：アプリケーション単位の閉塞
mcftdctsg コマンド：サービスグループ単位の閉塞
mcftdctsv コマンド：サービス単位の閉塞

- **閉塞を解除するコマンド**
  mcfaactap コマンド：アプリケーション単位の閉塞の解除
  mcftactsg コマンド：サービスグループ単位の閉塞の解除
  mcftactsv コマンド：サービス単位の閉塞の解除

コマンドを入力して MHP を閉塞した場合の MHP の閉塞状態は，オンラインシステム停止後の全面回復時に引き継げます。閉塞状態を引き継ぐかどうかを，状態引き継ぎ定義で指定します。なお，mcftdctsg コマンドでサービスグループを閉塞するとき，閉塞状態を引き継がない指定ができます。この場合，全面回復時に mcftactsg コマンドで閉塞を解除する必要はありません。

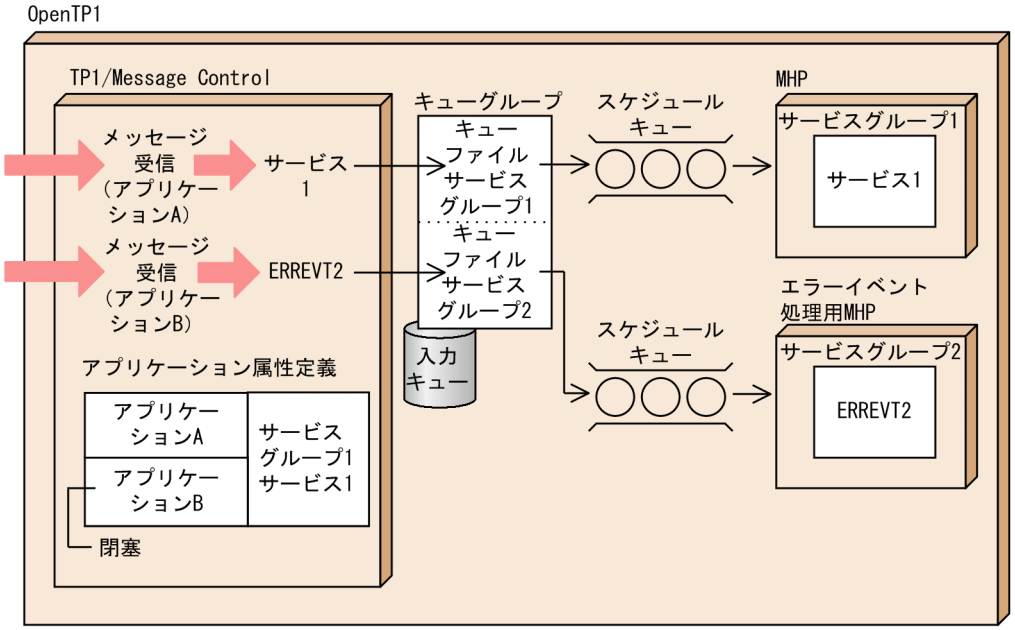
- **アプリケーション単位の閉塞**
  MHP をアプリケーション名単位で閉塞するときは，**mcfadctap コマンド**を使います。コマンドの引数によって，次に示す 3 種類の閉塞を選べます。
  MHP のアプリケーション名による閉塞を次の表に示します。

**表 3-12　MHP のアプリケーション名による閉塞**

| 閉塞種別 | タイミング | |
|---|---|---|
| | 閉塞したあとに到着したメッセージに対する処理 | 閉塞する前に到着してすでに入力キューに登録されているメッセージに対する処理 |
| 入力とスケジュールを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |
| 入力だけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | 該当する MHP を正常に起動します。 |
| スケジュールだけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |

サービス名に対応するアプリケーション名を，プロトコルごとに付けることができます。プロトコルごとにアプリケーション名を変えておくと，アプリケーション名によって閉塞することで，特定のプロトコルからのサービス要求だけを拒否できます。
アプリケーション名による閉塞を次の図に示します。

図 3-51　アプリケーション名による閉塞




- **サービス単位の閉塞**

  MHP をサービス単位で閉塞するときは，**mcftdctsv コマンド**を使います。コマンドの引数によって，次に示す 3 種類の閉塞を選べます。あるサービスを閉塞しても，同じ MHP のサービスグループにあるほかのサービスは使えます。

  MHP のサービスを閉塞する方法を次の表に示します。

表 3-13　MHP のサービスの閉塞

| 閉塞種別 | タイミング | |
|---|---|---|
| | 閉塞したあとに到着したメッセージに対する処理 | 閉塞する前に到着してすでに入力キューに登録したメッセージに対する処理 |
| 入力とスケジュールを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |
| 入力だけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | 該当する MHP を正常に起動します。 |
| スケジュールだけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |

- **サービスグループ単位の閉塞**

  MHP をサービスグループ単位で閉塞するときは，**mcftdctsg コマンド**を使います。コマンドの引数によって，次に示す 3 種類の閉塞を選べます。さらに，入力キューがディスクキューかメモリキューかによって，一部の処理が異なります。

  MHP のサービスグループの閉塞を次の表に示します。

表 3-14　MHP のサービスグループの閉塞

| 閉塞種別 | | タイミング | |
|---|---|---|---|
| | | 閉塞したあとに到着したメッセージに対する処理 | 閉塞する前に到着してすでに入力キューに登録されているメッセージに対する処理 |
| ディスク入力キュー | 入力とスケジュールを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | スケジュール待ち状態にし，閉塞解除後に該当する MHP を正常に起動します。 |
| | 入力だけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | 該当する MHP を正常に起動します。 |
| | スケジュールだけを閉塞 | 入力キューに登録し，スケジュール待ち状態にします。※ | スケジュール待ち状態にし，閉塞解除後に該当する MHP を正常に起動します。 |
| メモリ入力キュー | 入力とスケジュールを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |
| | 入力だけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | 該当する MHP を正常に起動します。 |
| | スケジュールだけを閉塞 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。 | MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録し直します。 |

注※

ディスク入力キューに登録されたメッセージの数が最大格納数を超えたときには，格納数を超えた分のメッセージは，MCF イベント処理用 MHP（ERREVT2）対応の入力キューに登録します。

### (b)　異常終了による自動閉塞

MHP がサービスを実行中に異常終了したときの閉塞単位をアプリケーション属性定義で指定すると，異常終了したときに，定義した単位で自動的に閉塞できます。

アプリケーション単位で閉塞するには，アプリケーション属性定義の aplihold に**アプリケーション単位で閉塞する**指定をします。

サービス単位で閉塞するには，アプリケーション属性定義の servhold に**サービス単位で閉塞する**指定をします。

サービスグループ単位で閉塞するには，アプリケーション属性定義の srvghold に**サービスグループ単位で閉塞する**指定をします。

また，異常終了した回数によって，アプリケーション，サービスまたはサービスグループを閉塞するかどうかを，アプリケーション属性定義で指定します。このとき，連続して異常終了した回数を数えるか，または異常終了した回数の合計を数えるかのどちらかを選択します。異常終了した回数には，MHP から dc_mcf_rollback 関数（ノーリターン指定）でロールバックした回数も含めます。ロールバックの関数については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

サービスグループ単位でスケジュールだけを閉塞する場合，MHP がサービスを実行中に異常終了したときの受信メッセージを，エラーイベントに切り替えるか，スケジュールキューの先頭に再スケジュールするかを指定できます。スケジュールキューの先頭に再スケジュールするには，アプリケーション属性定義の recvmsg に r を指定します。

異常終了による自動閉塞での MHP の閉塞状態は，オンラインシステム停止後の全面回復時に引き継ぎません。

アプリケーション属性定義の aplihold，servhold または srvghold に，異常終了した場合に**閉塞しない**指定をした場合には，アプリケーション，サービスまたはサービスグループを閉塞しません。この場合，異常終了が続くときには，コマンドを使って，アプリケーション，サービスまたはサービスグループを閉塞してください。

### (c)　MHP の再スケジュール

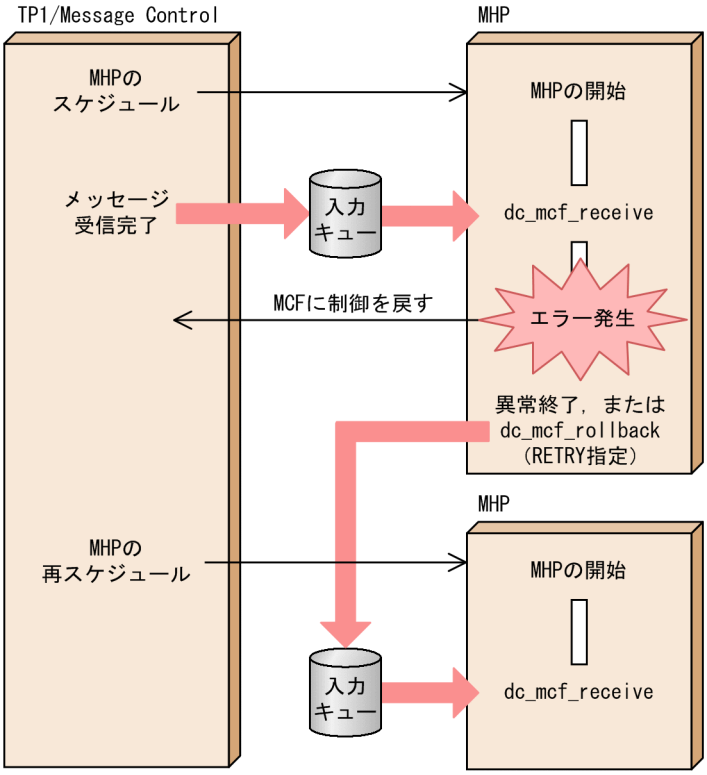
トランザクション処理を実行している MHP は，障害発生時に再スケジュールできます。

MCF は，次に示す場合に MHP を再スケジュールします。

- MHP から dc_mcf_rollback 関数（RETRY 指定）でロールバックした場合
- アプリケーション属性定義（mcfaalcap）または UAP 共通定義（mcfmuap）の reschedulecnt オペランドの指定値が 1 以上の場合に，MHP がサービス実行中に異常終了してロールバックしたとき

MHP の再スケジュール時の処理の流れを，次の図に示します。

図 3-52　MHP の再スケジュールの処理の流れ



異常終了した MHP を再スケジュールする場合，アプリケーション属性定義（mcfaalcap）または UAP 共通定義（mcfmuap）の次のオペランドで，再スケジュールの回数や間隔などを指定できます。

- reschedulecnt オペランド
- rescheduleint オペランド
- reschedulelog オペランド

指定した回数を超えて MHP が異常終了すると，再スケジュールされません。KFCA11163-E メッセージが出力され，エラーイベントが通知されます。

なお，MHP から dc_mcf_rollback 関数（RETRY 指定）でロールバックした場合や MHP が異常終了した場合に，MHP を再スケジュールすると，アプリケーション，サービス，またはサービスグループの自動閉塞で使用する，MHP が連続して異常終了した回数には含まれません。

MHP を再スケジュールする場合，受信メッセージを入力キューに再度格納する際に，先頭に格納するか最後に格納するかを選択できます。

先頭に格納する場合の注意事項を次に示します。

- システム共通定義の watch_time オペランドを省略するか，180 以上を指定してください。
- プロセスサービス定義の prc_recovery_resident オペランドを省略するか，Y を指定してください。
- MHP のユーザサービス定義またはユーザサービスデフォルト定義の scd_process_ctl_opt オペランドに 1 を指定してください。
- アプリケーション起動プロセスがなくても，MHP の再スケジュールができます。
  ただし，アプリケーション異常終了時の再スケジュール回数（アプリケーション属性定義（mcfaalcap -d）の reschedulecnt オペランドまたは UAP 共通定義（mcfmuap -r）の reschedulecnt オペランド）の指定値に 1 以上を指定していて，指定値を超えて異常終了したあとにエラーイベントを起動したい場合には，アプリケーション起動プロセスが必要となります。
- MHP の多重度が 1 の場合，MHP の再スケジュールをしたときにあとから受信したメッセージが前に受信したメッセージを追い越すことはありません。このため，繰り返し MHP の再スケジュールをする場合，エラーが解消されて正常に処理されるなどにより再スケジュールをしなくなるまで後続のメッセージを処理できませんので注意してください。

最後に格納する場合の注意事項を次に示します。

- MHP を再スケジュールするためには，アプリケーション起動プロセスが必要です。このため，アプリケーション起動プロセス用の MCF 通信構成定義を作成する必要があります。
- 受信メッセージを，入力キューに再度格納してから再スケジュールします。このため，MHP の多重度を 1 にしていたとしても，再スケジュールするメッセージよりもあとに受信したメッセージが，再スケジュールするメッセージを追い越すことがあります。

何らかの理由によって MHP の再スケジュールに失敗した場合，受信メッセージを破棄します。このとき，エラーイベントは通知されません。

MHP のサービス実行中にエラーが発生して繰り返し MHP の再スケジュールをしているときに，OpenTP1 を正常終了，強制正常終了または計画停止 A で終了した場合，エラーが解消されて正常に処理されるなどにより再スケジュールをしなくなるまで OpenTP1 の終了処理を待ち合わせます。

このため，MHP を停止するまで回復の見込みがない障害に対して無条件に RETRY 指定のロールバックをしないでください。

また，MHP のサービス実行中にエラーが発生して繰り返し MHP の再スケジュールをしているときに，OpenTP1 を正常終了，強制正常終了または計画停止 A で終了してしまい，MHP を停止するまで回復の見込みがない場合，OpenTP1 を強制停止してください。

#### (d) アプリケーション名と MHP の不整合による自動閉塞

MCF は MHP のスケジュール時に，アプリケーション属性定義で指定したアプリケーション名に対応するサービスグループがあるかどうかをチェックします。アプリケーション名に対応するサービスグループがない場合，該当するサービスグループが存在しないというメッセージログを出力して，サービスグループへのスケジュールだけを閉塞します。

アプリケーション名と MHP の不整合による自動閉塞での MHP の閉塞状態は，オンラインシステム停止後の全面回復時に引き継ぎません。

### 3.4.3 プロセスの制御

OpenTP1 のシステムサービス，またはユーザサーバ（UAP）を実行するときには，OS の作業領域を使います。この作業領域の処理を**プロセス**といいます。ユーザサーバの実行によって生成されるプロセスを特に**ユーザサーバプロセス**，**UAP プロセス**，または単に**プロセス**といいます。

OpenTP1 では，システムサービスとユーザサーバのプロセスが必要以上に増えたり減ったりしないように，プロセスサービス定義の指定に従って，使うプロセスの総数を制御しています。

ユーザサーバプロセスを制御するには，**ユーザサーバを事前に開始していること**が前提です。OpenTP1 と一緒に開始させておくか，dcsvstart -u コマンドで開始させておきます。

#### (1) マルチサーバ

実行中のユーザサーバに対して，さらにサービス要求が来た場合でも，ユーザサーバの処理を新しい別のプロセスで実行できます。このように，一つのユーザサーバの処理を，別のプロセスで並行して実行する機能を**マルチサーバ**といいます。

マルチサーバを使えるのは，スケジュールキューを使う SPP（キュー受信型サーバ），および MHP です。ソケット受信型サーバの場合はマルチサーバを使えません。ソケット受信型サーバには，使うプロセスは一つだけと，ユーザサービス定義で指定してください。

## (2) 常駐プロセスと非常駐プロセス

ユーザサービス定義でマルチサーバを指定したUAPのプロセスを，OpenTP1の稼働中にいつも確保しておくことも，動的に確保することもできます。いつも確保されているプロセスを**常駐プロセス**といいます。また，稼働中には確保されていなくて，必要に応じて起動されるプロセスを**非常駐プロセス**といいます。

マルチサーバを使う場合，使うプロセスの最大値をユーザサービス定義で設定しておきます。常駐プロセスを複数個指定していれば，指定した数だけ並行してプロセスが起動されます。また，非常駐プロセスを複数個指定していれば，指定した数だけ動的にプロセスが起動されます。

プロセスを非常駐プロセスと設定しておくと，OpenTP1システム内のメモリ領域を効率良く使えます。また，プロセスを常駐プロセスと設定しておくと，そのユーザサーバの処理は非常駐プロセスに比べて速くなります。

システムのメモリに空きがない場合，非常駐プロセスは稼働中の非常駐プロセスが終了してから実行されます。

プロセスを常駐とするか非常駐とするかは，ユーザサービス定義でユーザサーバの起動プロセス数を事前に指定しておきます。

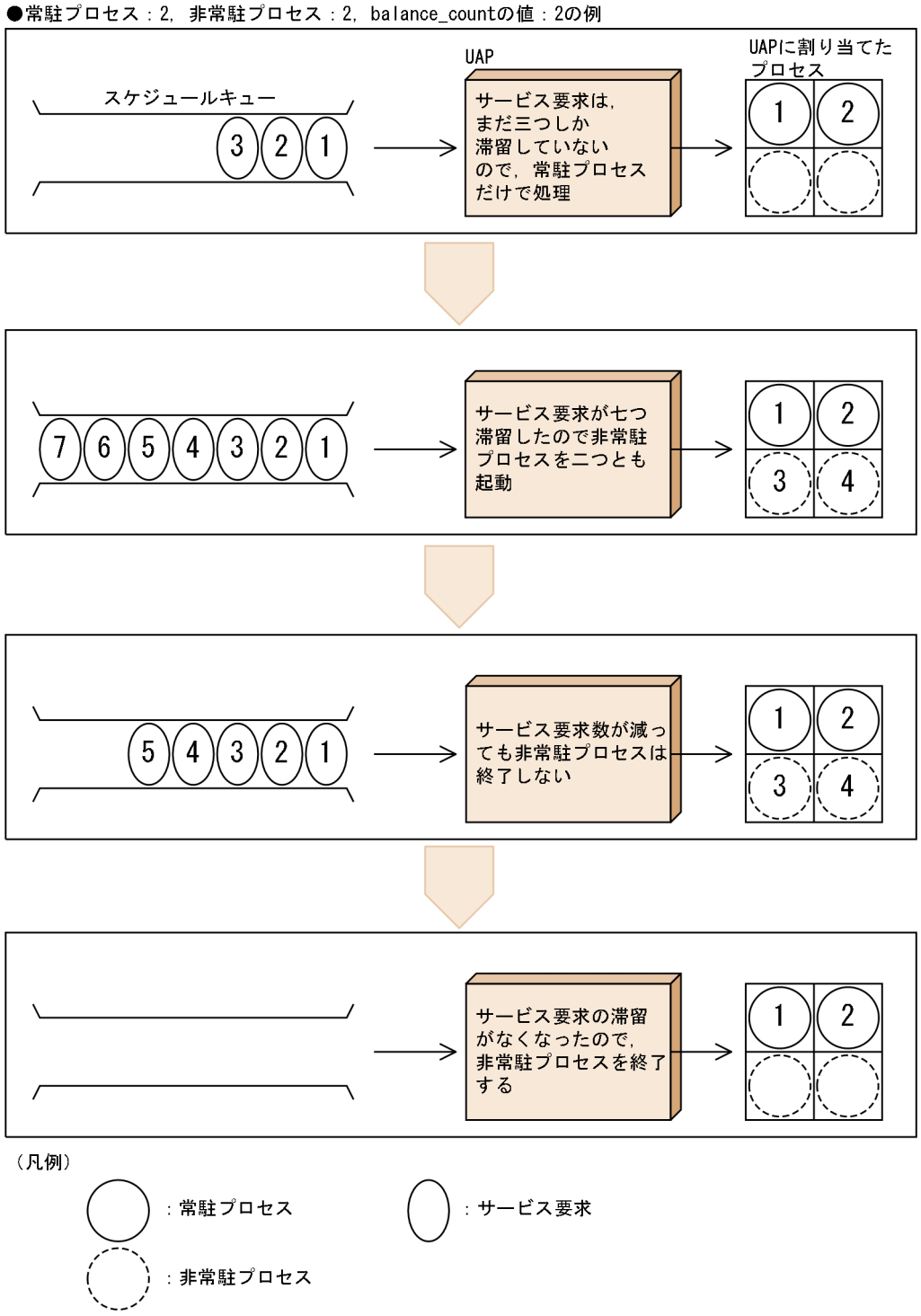
## (3) マルチサーバ負荷バランス機能

スケジュールキューにあるサービスの要求数に応じて，非常駐プロセスの数を増やしたり減らしたりする機能を**マルチサーバ負荷バランス機能**といいます。

非常駐プロセスをいつ起動するかは，ユーザサービス定義のbalance_countオペランドに指定する値で決まります。（balance_countオペランドに指定した値　×　起動中のプロセス）の数を超えてサービス要求が滞留したときに，OpenTP1は非常駐プロセスを起動します。スケジュールキューに滞留しているサービス要求の数が0になると，OpenTP1は非常駐プロセス数分のプロセスをランダムに終了させます。

常駐/非常駐プロセスによるマルチサーバの概要を次の図に示します。

図 3-53　常駐/非常駐プロセスによるマルチサーバの概要

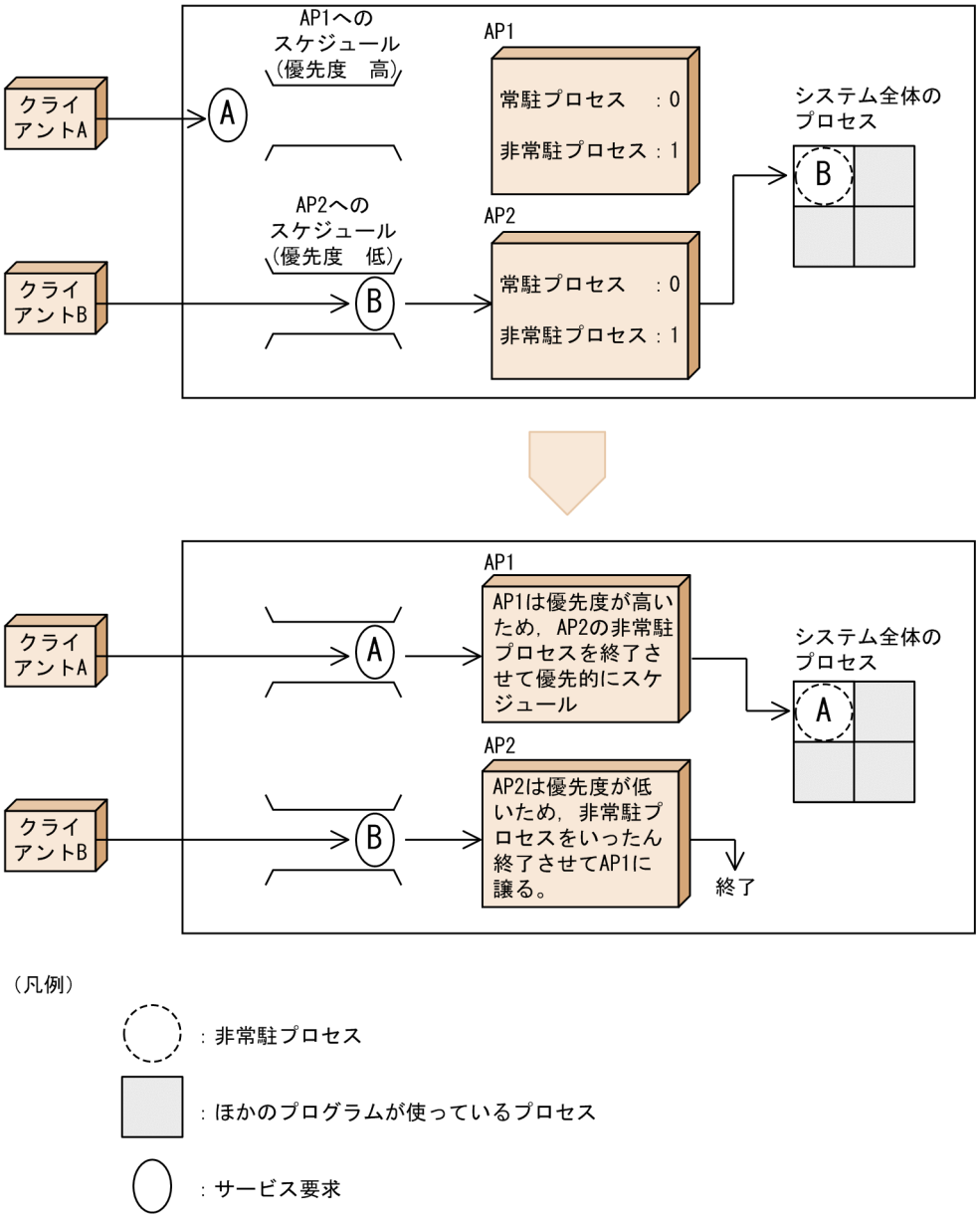



## (4)　スケジュールの優先度

一つ一つのユーザサーバには，ユーザサービス定義で**スケジュールの優先度**（スケジュールプライオリティ）を付けることができます。優先度が高いユーザサーバの非常駐プロセスは，ほかの非常駐プロセスに比べて優先的にスケジュールされます。

スケジュールの優先度の概要を次の図に示します。

図 3-54　スケジュールの優先度の概要




## (5)　非常駐 UAP プロセスのリフレッシュ機能

非常駐プロセスを使用する場合，一つのサービス要求ごとに実行プロセスを起動し直すことができます。この機能を，**非常駐 UAP プロセスのリフレッシュ機能**といいます。この機能を使用することで，リエントラント構造ではない UAP の実行もできるようになります。この機能は，非常駐プロセスだけで構成される UAP の場合に使用できます。

この機能を使用するためには，ユーザサービス定義またはユーザサービスデフォルト定義に，scd_refresh_process オペランドを指定します。また，この機能は 1 プロセスが処理するサービス要求数（ユーザサービス定義の balance_count オペランドの指定値）が 0 の場合にだけ使用できます。

非常駐 UAP プロセスのリフレッシュ機能の詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## (6) ノード間負荷バランス機能

SPP に要求されたサービスの処理に多くのプロセスが必要な場合，他ノードにある同じサービスグループ名の SPP に処理を分担できます。この機能を**ノード間負荷バランス機能**といいます。ノード間負荷バランス機能を使用するためには，負荷分散の前提として次の条件を満たしている必要があります。

- 一つの LAN の中で，複数のノードに同一のサービスを提供するユーザサーバが起動されていること。
- 各 OpenTP1 ノードが，システム共通定義の all_node オペランドに自分以外のノードを定義することによって，お互いの OpenTP1 ノードで起動されているユーザサーバの情報（ネーム情報）をやり取りしていること。

サービス要求は，任意に選ばれたノードのユーザサーバに渡されます。さらに，OpenTP1 では，スケジュールするノードのサーバ情報を参照して，スケジュール性能が低くなっているノードは選ばれにくいように制御しています。そのため，自ノードにユーザサーバがあり，スケジュールできる状態でも，自ノードのユーザサーバに渡されるとは限りません。自ノードのユーザサーバを優先するには，スケジュールサービス定義の scd_this_node_first オペランドに Y を指定します。これによって，自ノードのユーザサーバのスケジュール性能が低いときだけ，ほかのノードのユーザサーバにスケジュールできます。

ノード間負荷バランス機能は，ノード間でユーザサーバの動作条件が同じであることを前提としています。選択されるノードによって次に示す条件が大きく異なる場合は，ノード間負荷バランス機能に不適当な環境ですので，同じ名前のサービスグループを複数のノードに配置しないでください。

- 公衆回線の回線料金などの運用コスト
- 回線速度
- 回線品質
- ノードの単体性能

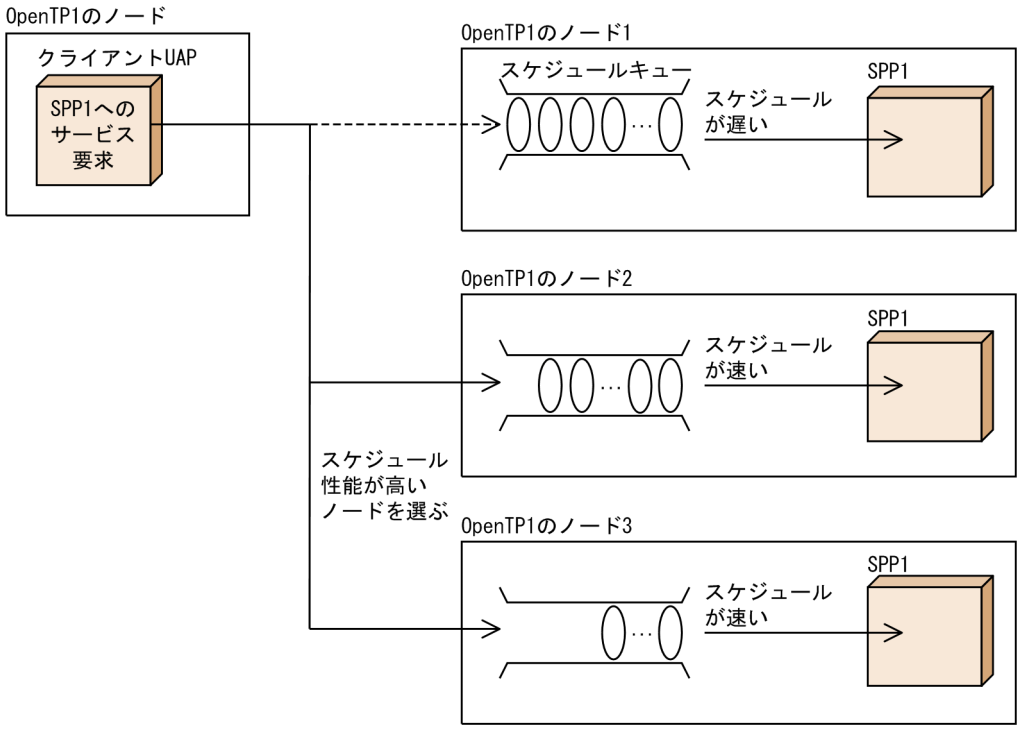
ユーザサーバにスケジュールするときに参照するサーバ情報は，各ノードから通知されます。同じサービスグループ名の SPP が複数のノードにないシステムでは，サーバ情報を通知する必要がなく，むだな通信となります。特に，公衆回線を使用する場合，不要な回線料金が課金されます。したがって，同じサービスグループ名が複数のノードにないシステムでは，すべてのノードのスケジュールサービス定義の scd_announce_server_status オペランドに N を指定して，サーバ情報の通知を抑止してください。

ノード間で負荷分散する SPP は，キュー受信型サーバでもソケット受信型サーバでもかまいません。ただし，ソケット受信型サーバの場合は，OpenTP1 のスケジュール性能を参照する制御はしていません。

ノード間負荷バランス機能で負荷を分散できるノードの数は，最大 128 です。

ノード間負荷バランス機能の概要を次の図に示します。

図 3-55　ノード間負荷バランス機能の概要




サービス要求は各ノードの負荷レベルに応じて，スケジュールされます。次に示す負荷レベルがあります。

- LEVEL0

  小さい負荷です。通常，サービス要求は LEVEL0 または LEVEL1 のノードにスケジュールされます。

- LEVEL1

  やや大きい負荷です。障害などによる再スケジュール時には，サービス要求は LEVEL1 のノードにスケジュールされにくくなります。ただし，再スケジュール時に LEVEL1 または LEVEL2 のノードしかない場合は，それらのノードにスケジュールされます。

- LEVEL2

  大きい負荷です。通常，サービス要求は LEVEL2 のノードにスケジュールされません。ただし，LEVEL2 のノードしかない場合は，LEVEL2 のノードにスケジュールされます。

各ノードの負荷レベルは，負荷監視インタバルごとに監視されます。その際，今回の負荷レベルは，前回の負荷レベル，キューイング数，サービス要求滞留数，およびサーバ処理率によって決定されます。

負荷レベルの決定条件について次の表に示します。

表 3-15　負荷レベルの決定条件

| 前回の負荷レベル | キューイング数：Q | サービス要求滞留数：q | サーバ処理率：X | 今回の負荷レベル |
|---|---|---|---|---|
| LEVEL0 | Q ≧ 1 | − | X ＜ 50 | LEVEL1 |
| LEVEL1 | Q ≧ 1 | − | 75 ≦ X | LEVEL0 |
| | | | 50 ≦ X ＜ 75 | LEVEL1 |

| 前回の負荷レベル | キューイング数：Q | サービス要求滞留数：q | サーバ処理率：X | 今回の負荷レベル |
|---|---|---|---|---|
| LEVEL1 | Q ≧ 1 | － | X ＜ 50 | LEVEL2 |
| LEVEL2 | － | q ＝ 0 | － | LEVEL0 |
| | | q ≧ 1 | | LEVEL2 |

（凡例）

－：無視します。

キューイング数

負荷監視インタバル間にスケジュールキューにキューイングされたサービス要求数です。

サービス要求滞留数

負荷監視時にスケジュールキューに滞留しているサービス要求数です。

サーバ処理率

次に示す計算式から算出される処理率です。

サーバ処理率＝（サービス処理数／（キューイング数＋サービス要求滞留数））×100

サービス処理数は，負荷監視インタバル間で処理したサービス要求数です。

負荷レベルに変更があった場合は，各ノードのネームサービスにサーバ情報が通知され，サーバ情報が更新されます。また，ユーザサービス定義の loadlevel_message オペランドを使用すると，負荷レベルの変更を通知するメッセージを出力できます。

## (7) ノード間負荷バランス機能使用時の定義

ノード間負荷バランス機能を使用する場合の TP1/Server Base 側，TP1/Client 側の関連する定義と処理，および RPC の処理を説明します。

### (a) サーバ側の判断で負荷分散を行う場合

TP1/Server Base のスケジュールサービスが，ノードのスケジュール状態に応じて，より効率的に処理できるノードへ負荷を分散させます。

#### サーバ（TP1/Server Base）側の定義

TP1/Server Base の定義では，次のどちらかの設定をする必要があります。

- スケジュールサービス定義のオペランドに次の設定をする。
  set scd_this_node_first ＝ N （デフォルト）
  set scd_announce_server_status ＝ Y （デフォルト）
- スケジュールサービス定義を省略する。

#### クライアント（TP1/Client）側の定義

クライアント環境定義に「dcscddirect=Y（TP1/Client/P の場合）」を定義します。

これによって，TP1/Client は，TP1/Server Base のスケジュールサービスに負荷分散を依頼するため，TP1/Client の定義でどの OpenTP1 ノードのスケジュールサービスに判断を依頼するかを指定します。

この場合，スケジュールを依頼する OpenTP1 ノードは，dchost オペランドに指定された順番にスケジュールを依頼します。dchost オペランドに書かれた順番ではなく，スケジュールを依頼する OpenTP1 ノードをランダムに指定したい場合は，さらに「dchostselect=Y（TP1/Client/P の場合）」を追加して定義する必要があります。

## (b) サーバからの負荷情報からクライアント側で判断する場合

### ■ クライアントが TP1/Client のとき

#### サーバ（TP1/Server Base）側の定義

TP1/Server Base の定義では，次のどちらかの設定をする必要があります。

- スケジュールサービス定義のオペランドに次の設定をする。
  set scd_this_node_first = N （デフォルト）
  set scd_announce_server_status = Y （デフォルト）
- スケジュールサービス定義を省略する。

#### クライアント（TP1/Client）側の定義

クライアント環境定義で「dccltloadbalance = Y（TP1/Client/P の場合）」を定義します。

これによって，TP1/Server Base から得たサーバの負荷レベルを基に，TP1/Client でサービス要求を行う OpenTP1 ノードを決めてから RPC を行います。

この場合，dccltcachetim で指定された時間（秒）だけ，サーバの負荷レベルを含む情報を一時的に dccache オペランドで指定された大きさの領域に保持します。つまり，dccltcachetim オペランドで指定した値を短くすることで，より新しいサーバの負荷レベルに基づいて RPC の要求先が決まるということになります。

その反面，負荷レベルの取得のために，頻繁に TP1/Server Base のネームサービスとの通信が発生することを考慮する必要があります。

### ■ サーバ，クライアントとも TP1/Server Base のとき

サーバ側，およびクライアント側の両方に，次のどちらかの設定をする必要があります。

- スケジュールサービス定義のオペランドに次の設定をする。
  set scd_this_node_first = N （デフォルト）
  set scd_announce_server_status = Y （デフォルト）
- スケジュールサービス定義を省略する。
  この形態では，TP1/Server Base はこれから要求を出そうとしているサーバの負荷レベルをすでに知っているので，最初から負荷レベルが低いノードに対して RPC を行います。この要求を受け取った時点で，スケジュールサービスは，負荷レベルの評価による転送はしないで，自ノードで要求を処理できる状態であれば，自ノードで処理します。サーバが閉塞している場合，または自ノードのサーバの負荷レベルが LEVEL2 で，他ノードに自ノードより負荷レベルの低いサーバがある場合だけ，ほかのノードに対して処理要求を転送します。

### (c) ノード間負荷バランス機能と別の機能を組み合わせた場合の動作

ノード間負荷バランス機能と別の機能を組み合わせた場合の動作を，次の表に示します。

表 3-16 ノード間負荷バランス機能と別機能を組み合わせた場合の動作

| 組み合わせる機能 | 動作 |
|---|---|
| TP1/Client で常設コネクションを使用した場合 | TP1/Server Base の CUP 実行プロセスが，常設コネクションを張ったノードで RPC を行う。<br>(b)の「サーバ，クライアントとも TP1/Server Base のとき」と同じ動作になる。 |
| TP1/Client でトランザクション制御 API を使用した場合 | TP1/Server Base のトランザクショナル RPC 実行プロセスが RPC を行う。<br>(b)の「サーバ，クライアントとも TP1/Server Base のとき」と同じ動作になる。 |
| リモート API 機能を使用した場合 | TP1/Server Base の rap サーバが RPC を行う。<br>(b)の「サーバ，クライアントとも TP1/Server Base のとき」と同じ動作になる。 |

## (8) ノード間負荷バランス拡張機能

ユーザは次に示す指定ができます。

- LEVEL0 のノードへのスケジュール比率の指定

  スケジュールサービス定義の schedule_rate オペランドを指定することによって，LEVEL0 のノードにスケジュールする比率（%）を指定できます。

  TP1/Client のクライアント環境定義の DCSCDDIRECT オペランドに Y を指定してサービス要求をする場合だけ schedule_rate オペランドは有効です。

  ただし，スケジュールサービス定義の scd_this_node_first に Y を指定している場合には，自ノードに優先的にスケジュールされます。

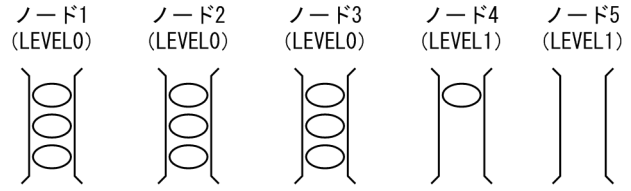
  次に schedule_rate オペランドに 80 を指定した場合のスケジュールについて説明します。サービス要求数は 10 とします。

  1. スケジューラは，ネームサービスから全ノードの負荷情報を取得し，LEVEL0 と LEVEL1 のノード数をカウントします。
  2. 1.でカウントしたノード数を schedule_rate オペランドの指定数で重みづけして，LEVEL0 のノードにスケジュールする比率を求めます。

     LEVEL0：LEVEL1 ＝ 80×3：20×2 ≒ 85：15
  3. 2.で求めた比率で LEVEL0 のノードの一つを選択して，サービス要求をスケジュールします。

  LEVEL0 のノードへのスケジュールについて次の図に示します。

図 3-56 LEVEL0 のノードへのスケジュール



- 負荷監視インタバル時間の指定

  ユーザサービス定義およびユーザサービスデフォルト定義の loadcheck_interval オペランドを指定することによって，サービスグループごとに負荷監視インタバル時間を指定できます。負荷監視時に負荷レベルの変更があった場合は，各ノードのネームサービスにサーバ情報が通知されます。そのため，負荷監視インタバルごとにサーバ情報がネットワーク上に送信される可能性がありますので，必要以上に小さい値を指定しないでください。

  loadcheck_interval オペランドを指定しない場合の負荷監視インタバルは，30 秒です。また，負荷監視の要否のチェックなどについて 10 秒のインタバルで実行されます。つまり，負荷監視の要否のチェックなどの 3 回目には，負荷監視が実行されます。

  しかし，loadcheck_interval オペランドを指定する場合，負荷監視インタバルはオペランドの指定値となり，負荷監視の要否のチェックなどについては，10 と各ユーザサーバの loadcheck_interval オペランドの指定値との最大公約数から求めたインタバルで実行されます。例えば，SPP1 の loadcheck_interval オペランドに 3 を，SPP2 の loadcheck_interval オペランドに 5 を指定する場合，10 と 3 と 5 の最大公約数の 1（秒）が負荷監視の要否のチェックなどを実行するインタバルとなります。負荷監視の要否のチェックなどの 3 回目には，SPP1 の負荷監視が実行されます。5 回目には，SPP2 の負荷監視が実行されます。

  そこで，システムに与える影響を少なくするために，loadcheck_interval オペランドに指定する値は，5 の倍数にすることをお勧めします。

  loadcheck_interval オペランドに 0 を指定することによって，負荷監視をサービスグループ単位で抑止できます。

- 負荷レベルのしきい値の指定

  ユーザサービス定義およびユーザサービスデフォルト定義の levelup_queue_count オペランドおよび leveldown_queue_count オペランドを次に示すとおりに指定することによって，サービスグループごとにサービス要求滞留数によって負荷レベルを決定するしきい値を指定できます。

```
set levelup_queue_count = U1,U2
set leveldown_queue_count = D0,D1
```

  U1：サーバの負荷レベルが LEVEL1 に上がったと判断するサービス要求滞留数

  U2：サーバの負荷レベルが LEVEL2 に上がったと判断するサービス要求滞留数

  D0：サーバの負荷レベルが LEVEL0 に下がったと判断するサービス要求滞留数

  D1：サーバの負荷レベルが LEVEL1 に下がったと判断するサービス要求滞留数

  今回の負荷レベルは，前回の負荷レベル，およびサービス要求滞留数によって決定されます。

  負荷レベルとサービス要求滞留数について次の表に示します。

**表 3-17　負荷レベルとサービス要求滞留数**

| 前回の負荷レベル | サービス要求滞留数：q | 今回の負荷レベル |
|---|---|---|
| LEVEL0 | q ＜ U1 | LEVEL0 |
| | U1 ≦ q ＜ U2 | LEVEL1 |
| | U2 ≦ q | LEVEL2 |

| 前回の負荷レベル | サービス要求滞留数：q | 今回の負荷レベル |
|---|---|---|
| LEVEL1 | q ≦ D0 | LEVEL0 |
| | D0 ＜ q ＜ U2 | LEVEL1 |
| | U2 ≦ q | LEVEL2 |
| LEVEL2 | q ≦ D0 | LEVEL0 |
| | D0 ＜ q ≦ D1 | LEVEL1 |
| | D1 ＜ q | LEVEL2 |

- 通信障害時のリトライ回数の指定

  通常，サービス要求のスケジュール時に通信障害が発生すると，再スケジュールしないでエラーリターンします。

  スケジュールサービス定義の scd_retry_of_comm_error オペランドを指定することによって，通信障害が発生したノード以外へスケジュールするリトライ回数を指定できます。

  ただし，scd_retry_of_comm_error オペランドの指定値が，サービス要求の対象となるサービスグループが起動しているノード数を上回っている場合は，サービス要求の対象となるサービスグループが起動しているノード数をリトライ回数の上限値とします。

  0 を指定した場合には，リトライしません。

  なお，この機能は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できませんので，ご了承ください。

## (9) マルチスケジューラ機能

従来のスケジューラデーモン（これ以降**マスタスケジューラデーモン**といいます）とは別に，サービス要求受信専用デーモン（これ以降**マルチスケジューラデーモン**といいます）を複数プロセス起動し，サービス要求メッセージ受信処理を並行動作させることによって，受信処理の競合によるスケジューリング遅延を回避できます。この機能を**マルチスケジューラ機能**といいます。

マルチスケジューラ機能を使用するには，次の定義を指定する必要があります。

RPC 受信側

スケジュールサービス定義　scdmulti

ユーザサービス定義　scdmulti

RPC 送信側

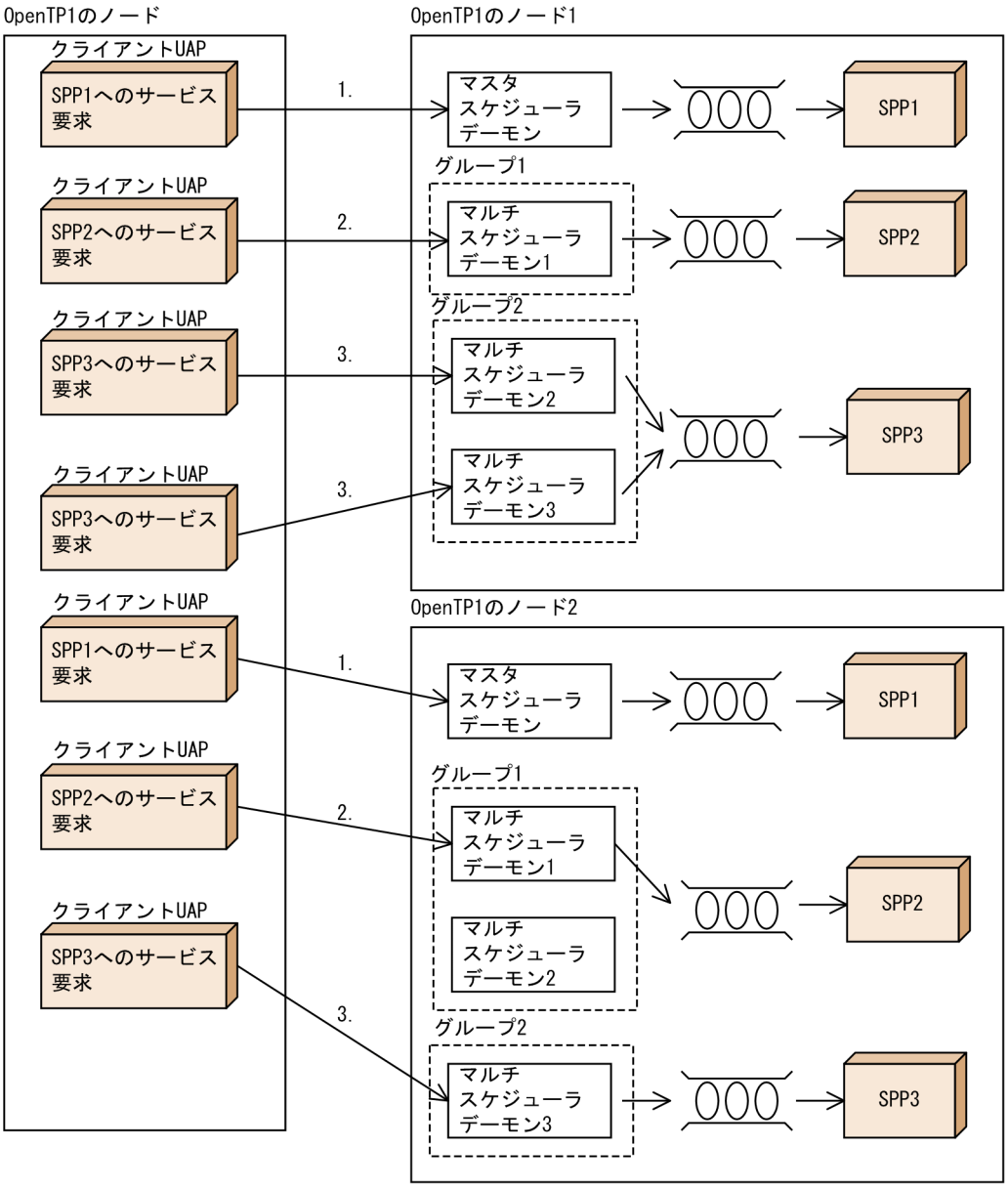
ユーザサービス定義　multi_schedule

また，幾つかのマルチスケジューラデーモンをキュー受信型サーバごとにグループ化できます。これによって，異なるサーバ間でサービス要求メッセージの受信処理が競合するのを回避できます。マルチスケジューラデーモンをグループ化している場合，サーバ側でユーザサービス定義 scdmulti を指定する必要があります。

なお，この機能は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できませんので，ご了承ください。

マルチスケジューラ機能の概要を次の図に示します。

図 3-57　マルチスケジューラ機能の概要




・SPP1は，短いサービス要求メッセージを扱うため，マルチスケジューラ機能を使用しないで，マスタスケジューラデーモンにスケジューリングさせます。（図の1.）
・SPP2は，長大サービス要求メッセージを扱うため，OpenTP1のノード1とノード2に分散させ，ノード1にあるグループ1のマルチスケジューラデーモン1，またはノード2にあるグループ1のマルチスケジューラデーモン1，2にスケジューリングさせます。（図の2.）
・SPP3は，短いサービス要求メッセージを扱いますが，サービス要求数が多いため，OpenTP1のノード1とノード2に分散させ，ノード1にあるグループ2のマルチスケジューラデーモン2，3，またはノード2にあるグループ2のマルチスケジューラデーモン3にスケジューリングさせます。（図の3.）

### 3.4.4　バッファ領域の共用による共用メモリの節約

共用メモリ上にサービス要求を格納する領域（**メッセージ格納バッファプール**）は，ユーザサーバごとに確保されます。この領域を複数のユーザサーバ間で共用すると，ユーザサーバのスケジュールに使う共用メモリを節約できます。

メッセージ格納バッファプールを共用する場合は，該当するノードに次に示す定義を指定します。

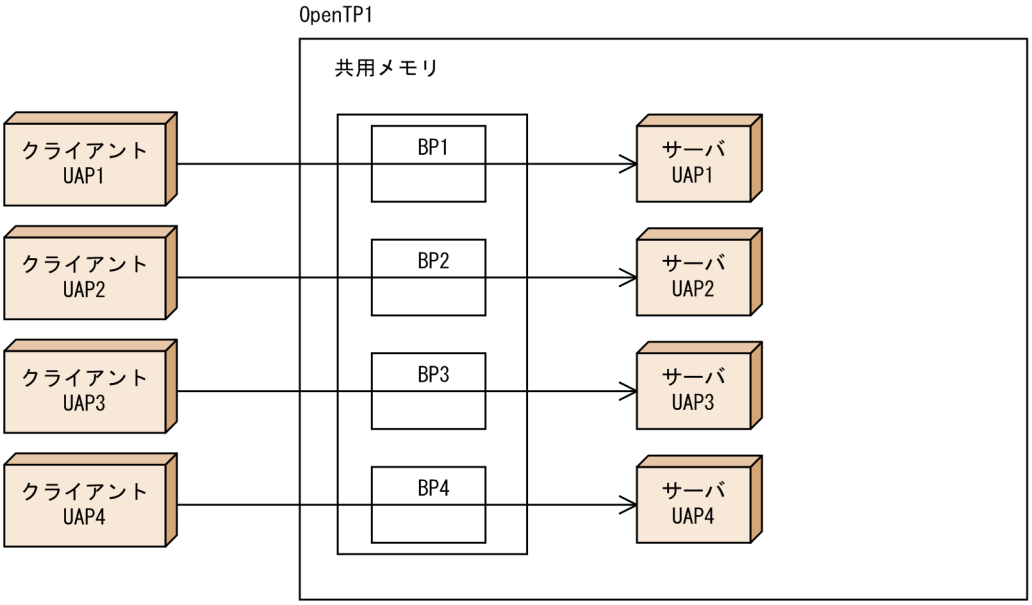
- スケジュールサービス定義の scdbufgrp オペランドに，バッファ領域を共用化するユーザサーバのグループの名称（**スケジュールバッファグループ名**）を指定します。
- 共用化するそれぞれのユーザサーバのユーザサービス定義に，スケジュールバッファグループ名を scdbufgrp オペランドで指定します。

ユーザサービス定義に指定したスケジュールバッファグループ名に該当する名称がスケジュールサービス定義にない場合は，該当するユーザサーバの開始時にエラーになります。

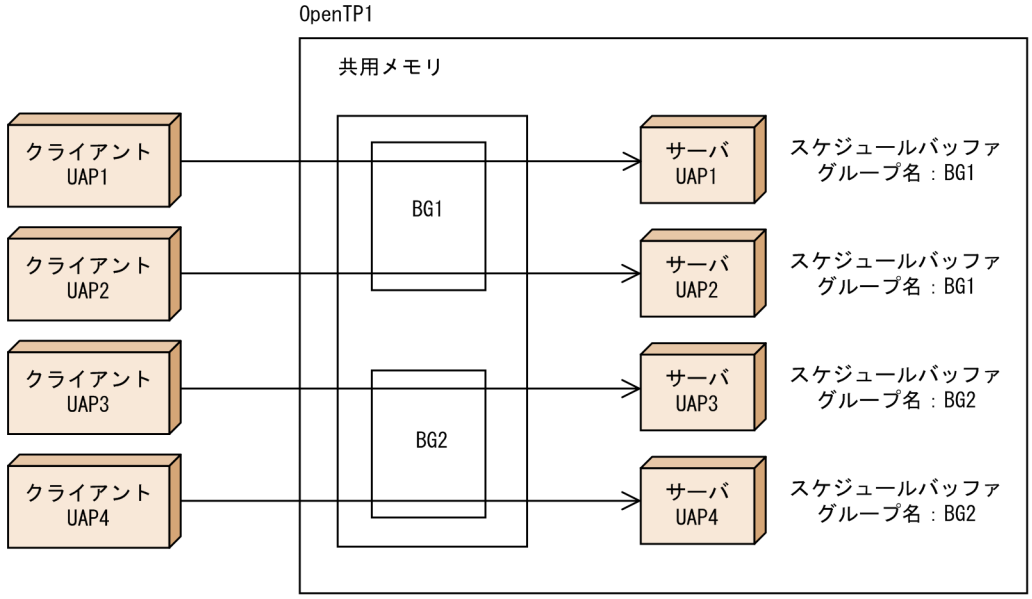
バッファ領域の共用を次の図に示します。

図 3-58　バッファ領域の共用

●バッファ領域を共用化しない場合




●バッファ領域を共用化する場合




（凡例）
BP：メッセージ格納バッファプール
BG：スケジュールバッファグループ（スケジュールサービス定義で指定）

## (1)　バッファ領域の共用を指定できるユーザサーバ

バッファ領域の共用化の指定は，キュー受信型サーバの場合にだけ有効です。ソケット受信型サーバにバッファ領域を共用する指定をしても無視されて，共用しない状態で実行されます。

## (2)　バッファ領域を共用化する場合の注意

共用化を指定した複数のユーザサーバのプロセスが同時に起動されると，バッファ領域の競合が起こってスケジュール性能が落ちる場合があります。

## (3)　共有化したバッファの使用サイズの制限

バッファ領域を共用化する場合，共用している一部のユーザサーバの処理が遅延したり，大きなデータのサービス要求が格納されたりすると，バッファ領域が圧迫され，ほかのユーザサーバに対するサービス要求がバッファ不足となることがあります。これを防止するために，ユーザサーバごとにバッファの使用サイズを制限することができます。

バッファの使用サイズを制限するには，ユーザサービス定義の scdbufgrp 定義コマンドに-s オプション，または-p オプションを指定します。-s オプションは共用するバッファに対し，そのユーザサーバに対するサービス要求を何バイトまで格納できるかを指定します。また，-p オプションは共用するバッファに対し，そのユーザサーバに対するサービス要求を何％まで格納できるかを指定します。

バッファの使用サイズの制限は，キュー受信型 SPP の場合にだけ使用できます。MHP では使用できません。指定した場合は無視されます。

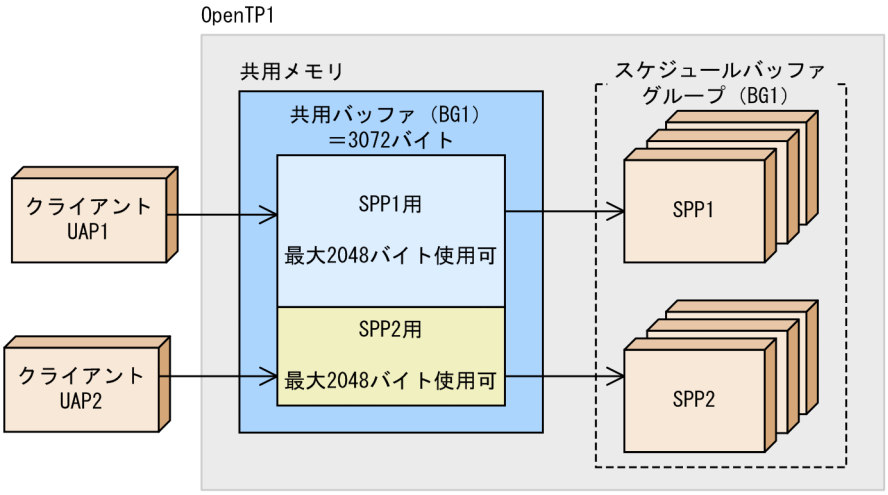
バッファの使用サイズの制限について，scdbufgrp 定義コマンドに指定するオプション別に以降の図に示します。

図 3-59　バッファの使用サイズの制限（scdbufgrp 定義コマンドに-s オプションを指定）

```
〈スケジュールサービス定義〉
scdbufgrp -g BG1
          -n 6 -l 512
```

```
〈SPP1のユーザサービス定義〉
scdbufgrp -g BG1 -s 2048
```

```
〈SPP2のユーザサービス定義〉
scdbufgrp -g BG1 -s 2048
```



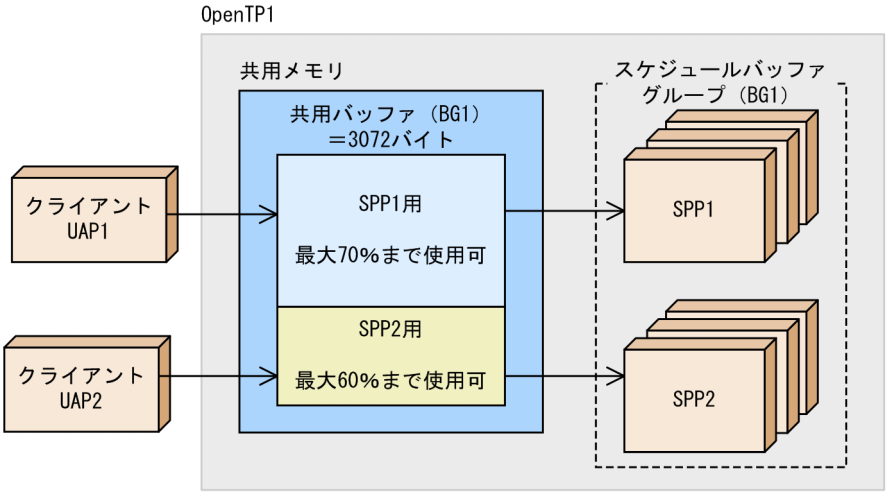

この図では，スケジュールサービス定義を指定して，バッファグループ BG1 に 3072 バイトを確保しています。また，ユーザサービス定義を指定して，SPP1，および SPP2 が，それぞれ最大 2048 バイトまでバッファを使用できるようにしています。これによって，SPP1 向けのサービス要求がバッファに滞留しても最大 2048 バイトまでしか使用できないため，SPP2 は残り 1024 バイトのバッファで動作できます。

図 3-60　バッファの使用サイズ制限（scdbufgrp 定義コマンドに-p オプションを指定）

```
〈スケジュールサービス定義〉
scdbufgrp -g BG1
          -n 6 -l 512
```

```
〈SPP1のユーザサービス定義〉
scdbufgrp -g BG1 -p 70
```

```
〈SPP2のユーザサービス定義〉
scdbufgrp -g BG1 -p 60
```




この図では，スケジュールサービス定義を指定して，バッファグループ BG1 に 3072 バイトを確保しています。また，ユーザサービス定義を指定して，SPP1 は最大 70%まで，SPP2 は最大 60%までバッファを使用できるようにしています。これによって，SPP1 向けのサービス要求がバッファに滞留しても約 70%（2048 バイト※）までしか使用できないため，SPP2 は残り 30%（1024 バイト）のバッファで動作できます。

注※

指定した割合によって，使用できるサイズはセルサイズの倍数に切り下げられます。

バッファグループに属するサーバ UAP で一律に使用サイズを制限する場合，スケジュールサービス定義の scdbufgrp 定義コマンドに-s オプション，または-p オプションを指定することもできます。この場合は，ユーザサービス定義の scdbufgrp 定義コマンドには-s オプション，または-p オプションを指定する必要はありません。

スケジュールサービス定義で一律に使用サイズを制限する場合の例を次に示します。図中の番号と以降の説明の番号は対応しています。

図 3-61　スケジュールサービス定義で一律に使用サイズを制限する場合の例

```
〈スケジュールサービス定義〉
scdbufgrp -g BG1 -n 6 -l 512 -s 1024  ……1.
scdbufgrp -g BG2 -n 8 -l 512 -p 25    ……2.
```

SPP1
```
〈ユーザサービス定義〉
scdbufgrp -g BG1  ……3.
```

SPP2
```
〈ユーザサービス定義〉
scdbufgrp -g BG2  ……4.
```

SPP3
```
〈ユーザサービス定義〉
scdbufgrp -g BG1  ……3.
```

SPP4
```
〈ユーザサービス定義〉
scdbufgrp -g BG2  ……4.
```

SPP5
```
〈ユーザサービス定義〉
scdbufgrp -g BG1 -s 2048  ……5.
```

SPP6
```
〈ユーザサービス定義〉
scdbufgrp -g BG2 -p 50  ……6.
```

SPP7
```
〈ユーザサービス定義〉
scdbufgrp -g BG2 -s 2048  ……7.
```

1. スケジュールサービス定義を指定して，バッファグループ BG1 に 3072 バイト確保し，一律で制限する使用サイズとして 1024 バイトを指定します。
2. スケジュールサービス定義を指定して，バッファグループ BG2 に 4096 バイト確保し，一律で制限する使用サイズとして 25%（1024 バイト）を指定します。
3. SPP1 および SPP3 はバッファグループ BG1 を使用し，使用サイズはスケジュールサービス定義で一律に指定された 1024 バイトに制限されます。
4. SPP2 および SPP4 はバッファグループ BG2 を使用し，使用サイズはスケジュールサービス定義で一律に指定された 25%（1024 バイト）に制限されます。
5. SPP5 はバッファグループ BG1 を使用しますが，使用サイズはスケジュールサービス定義で一律に指定された 1024 バイトではなく，ユーザサービス定義で指定された 2048 バイトになります。
6. SPP6 はバッファグループ BG2 を使用しますが，使用サイズはスケジュールサービス定義で一律に指定された 25%（1024 バイト）ではなく，ユーザサービス定義で指定された 50%（2048 バイト）になります。
7. SPP7 はバッファグループ BG2 を使用しますが，使用サイズはスケジュールサービス定義で一律に指定された 25%（1024 バイト）ではなく，ユーザサービス定義で指定された 2048 バイトになります。

## (4)　バッファ領域を共有化する場合の注意

共用化したバッファの使用サイズを制限する，スケジュールサービス定義の scdbufgrp 定義コマンドの-s オプション，または-p オプションは，ユーザサービスデフォルト定義にも指定できます。しかし，ユーザサービスデフォルト定義に指定した場合，共用バッファではない通常のバッファプール（ユーザサービス定義の message_store_buflen オペランドを指定）を使用するユーザサーバが使用できなくなります。

### 3.4.5　マルチサーバのプロセス制御の例

マルチサーバを指定した UAP（SPP，または MHP）のプロセス制御を，例を使って示します。ユーザサービス定義でプロセス制御に関して，次の表のように指定したとします。

表 3-18　ユーザサービス定義での指定

| 定義項目 | サービスグループ名 | | | 合計 |
|---|---|---|---|---|
| | G1 | G2 | G3 | |
| 常駐プロセスの数 | 2 | 1 | 1 | 4 |
| サービスグループ内のプロセスの最大数 | 4 | 2 | 2 | 8 |
| スケジュールプライオリティ | 1 | 2 | 3 | 該当しない |

このとき同時に起動できる SPP と MHP のプロセスの最大数を 6 とします。サービスグループ G1，G2，G3 の常駐プロセス数の合計が 4 なので，動的に起動できるプロセスはあと二つになります。その二つを，サービスグループ G1，G2，G3 がスケジュールプライオリティに従って利用します。プロセス制御の流れを次の表に示し，時間の経過とともにプロセスの制御の流れを説明します。

表 3-19　プロセス制御の流れ

| 時間の経過 | サービス要求 | サービスグループ名 | | |
|---|---|---|---|---|
| | | G1 | G2 | G3 |
| 1 | オンライン開始 | ○○ | ○ | ○ |
| 2 | G1，G3 にサービス要求 | ●● | ○ | ● |
| 3 | G1，G3 のサービス要求増加 | ●●▲ | ○ | ●▲ |
| 4 | G1 のサービス要求さらに増加 | ●●▲▲ | ○ | ● |
| 5 | G2 にサービス要求 | ●●▲▲ | ● | ● |
| 6 | G2 のサービス要求増加 | ●●▲▲ | ● | ● |
| 7 | G1 のサービス要求低下 | ●●▲ | ●▲ | ● |

（凡例）

○：待機している常駐プロセス（起動済み）

●：サービスを実行中の常駐プロセス

▲：サービスを実行中の非常駐プロセス

経過 1

オンライン開始時にサービスグループ G1，G2，G3 の常駐プロセスを起動します。常駐プロセスの合計が 4，起動できるプロセスの最大数が 6 なので，動的に起動できるプロセス数は 2 になります。

経過 2

G1，G3 にサービス要求が来たので，G1，G3 の常駐プロセスでサービスを実行します。

経過 3

G1，G3 のサービス要求が増加したので，G1，G3 の非常駐プロセスをサービスグループ内の最大数の範囲内で起動して，サービスを実行します。このとき，プロセス数の合計が 6 になったので，これ以上動的にプロセスを増やすことはできません。

経過 4

G1 のサービス要求がさらに増加しても，これ以上動的にプロセスを増やすことができません。そこで，G3 よりも G1 の方がスケジュールプライオリティが高いので，G3 の非常駐プロセスを，実行中のサービスが終了した時点で終了させ，G1 の非常駐プロセスを起動します。

経過 5

G2 にサービス要求が来ると，待機していた G2 の常駐プロセスでサービスを実行します。

経過 6

G2 のサービス要求が増加しても，これ以上動的にプロセスを起動できません。また，G2 よりも G1 の方がスケジュールプライオリティが高いので，G1 の非常駐プロセスを終了させて，G2 の非常駐プロセスを起動することはしません。

経過 7

G1 のサービス要求が低下したので，G1 の非常駐プロセスのうちの一つを，実行中のサービスが終了した時点で終了させ，G2 の非常駐プロセスを起動します。

以上のように OpenTP1 では，SPP と MHP に対するサービス要求を，プロセスを制御することで効率的に処理しています。

## 3.5 OpenTP1 クライアント機能（TP1/Client）

OpenTP1 で構築したサーバシステムへ，クライアント用の WS または PC からサービスを要求できる機能（**OpenTP1 クライアント機能　TP1/Client**）について説明します。

なお，TP1/Client/W および TP1/Client/P の詳細については，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/W，TP1/Client/P 編」を参照してください。TP1/Client/J の詳細については，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。

- **TP1/Client の機能概要**

  TP1/Client の機能を次に示します。

  - **TP1/Client のリモートプロシジャコール**

    TP1/Client の UAP（CUP），Java アプレット，Java アプリケーション，または Java サーブレットから OpenTP1 の SPP へサービスを要求します。TP1/Client/W または TP1/Client/P を使用する場合は，CUP からトランザクションを開始して，SPP へサービスを要求することもできます。

  - **TCP/IP プロトコルを使ったメッセージ送受信**

    （TP1/Client/W および TP1/Client/P でサポートします）

    CUP と OpenTP1 の MHP で，メッセージを送受信できます。

  - **XDM/DCCM3 との通信**

    CUP，Java アプレット，Java アプリケーション，または Java サーブレットから OpenTP1 へリモートプロシジャコールを発行するのと同様の方法で，XDM などの従来型システムと通信できます。

- **TP1/Client から OpenTP1 へサービスを要求する準備**

  OpenTP1 が TP1/Client からサービスを要求されたときに，その要求を受け付けるかどうかを判定する機能を**ユーザ認証機能**といいます。

  ユーザ認証機能を使用するためには，認証を受けるログイン名とパスワードを，OpenTP1 が動作する OS にユーザ ID（ユーザアカウント）としてあらかじめ登録しておく必要があります。なお，パスワードは，TP1/Client のユーザ認証要求関数（dc_clt_cltin 関数または dc_clt_cltin_s 関数）の引数 logname，および引数 passwd に指定する値です。

  ユーザ認証機能の使い方は，製品によって異なります。TP1/Client/W，および Windows 用の TP1/Client/P の場合，CUP のコーディング時に，ユーザ認証機能の開始と終了を示す関数（dc_clt_cltin 関数，dc_clt_cltout 関数）を呼び出します。

  なお，TP1/Client/J を使用する場合は上記の準備は不要です。

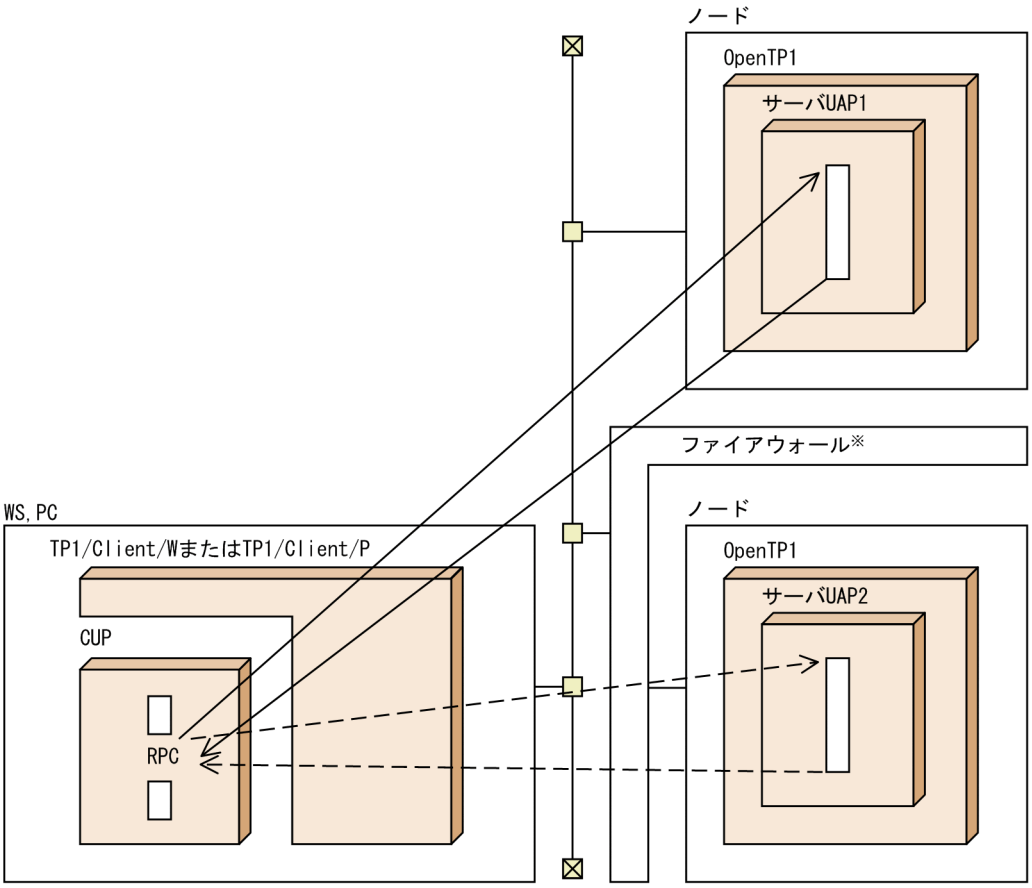
### 3.5.1 TP1/Client のリモートプロシジャコール

CUP から，TP1/Client/W または TP1/Client/P のライブラリ関数を使って，OpenTP1 のサーバ UAP（SPP）へサービスを要求します。

TP1/Client/W または TP1/Client/P を使用する場合は，CUP からトランザクションを開始することもできます。

TP1/Client/W または TP1/Client/P と OpenTP1 との通信を次の図に示します。

### 図 3-62　TP1/Client/W または TP1/Client/P と OpenTP1 との通信

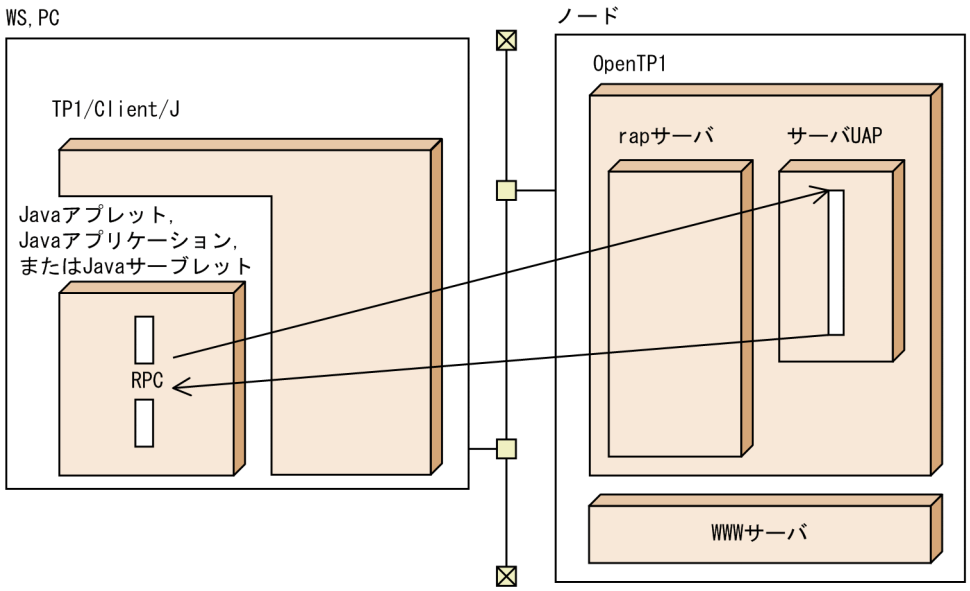



注※　OpenTP1独自のインタフェースを使うと，ファイアウォールの内側にあるサーバUAPにもリモートプロシジャコールを使って通信できます。ただし，ファイアウォールを通過した場合，通過先のサーバUAPをトランザクション処理に含めることはできません。
ファイアウォールを通過する場合についてはマニュアル「OpenTP1　クライアント使用の手引」を参照してください。

Java アプレット，Java アプリケーション，または Java サーブレットからは，TP1/Client/J のクラスライブラリを使って，OpenTP1 のサーバ UAP（SPP）へサービスを要求します。

TP1/Client/J と OpenTP1 との通信を次の図に示します。

図 3-63　TP1/Client/J と OpenTP1 との通信




## (1)　OpenTP1 の定義が必要な TP1/Client の機能

TP1/Client/W または TP1/Client/P で次に示す機能を使う場合，サーバとなる OpenTP1 には**クライアントサービス定義**が必要です。

- CUP からトランザクションを開始する場合

  クライアントサービス定義を作成した OpenTP1 では，TP1/Client/W または TP1/Client/P から開始したトランザクションを管理するシステムサービスが起動します。このシステムサービスを，**クライアントサービス**といいます。起動したクライアントサービスは，TP1/Client の CUP が開始したトランザクションを管理します。

- 常設コネクションを使用する場合

  クライアントサービス定義を作成した OpenTP1 では，TP1/Client/W または TP1/Client/P との間に常設コネクションを接続するためのシステムサービスが起動します。このシステムサービスを，**クライアント拡張サービス**といいます。起動したクライアント拡張サービスは，TP1/Client の CUP との間に常設コネクションを確立します。

また，TP1/Client/W または TP1/Client/P でリモート API 機能を使用する場合や，TP1/Client/J で OpenTP1 の SPP へサービスを要求する場合には，OpenTP1 側に rap リスナーサービス定義が必要です。

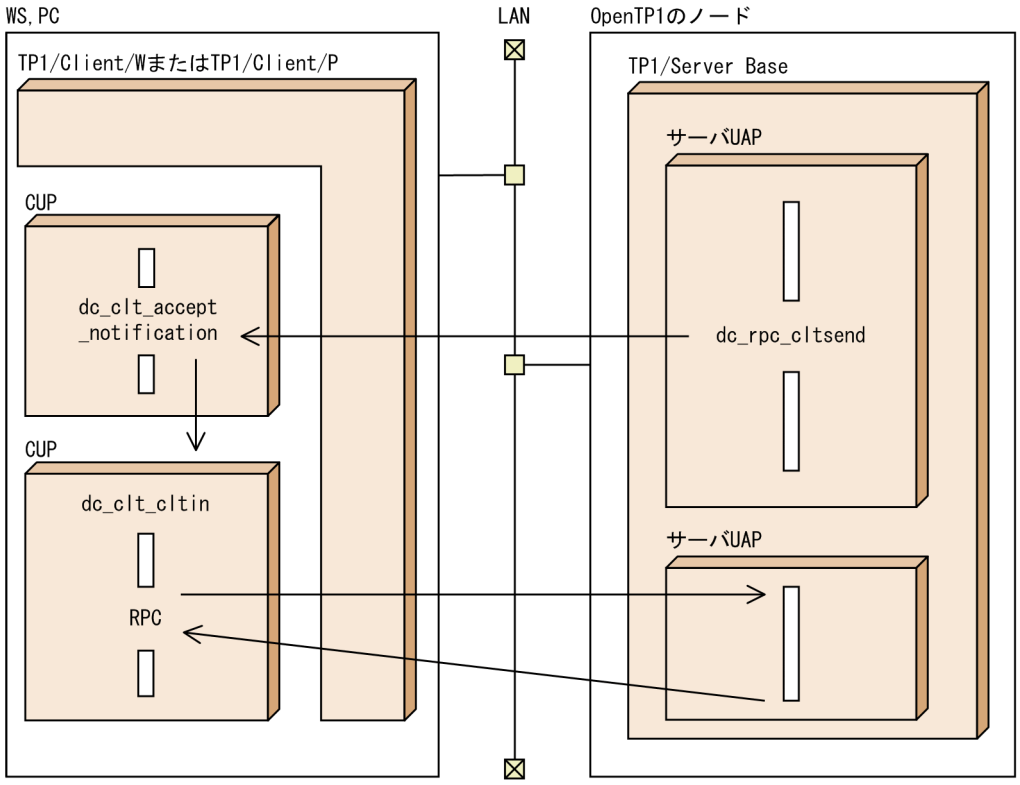
## (2)　サーバの開始を TP1/Client へ通知

OpenTP1 のサーバ UAP から，TP1/Client/W または TP1/Client/P のアプリケーションプログラム（CUP）へ UAP が開始したことを通知できます。CUP へは，dc_rpc_cltsend 関数でデータを送って，サーバ UAP の開始を通知します。この機能を使って，サーバの起動完了を一斉にクライアントへ知らせることができます。

dc_rpc_cltsend 関数で通知したデータは，CUP の dc_clt_chained_accept_notification 関数，または dc_clt_accept_notification 関数で受け取ります。CUP がデータを受け取ることで，TP1/Client/W または TP1/Client/P はサーバが稼働中であることがわかります。その後，CUP からサーバへサービスを要求します。dc_clt_chained_accept_notification 関数，および dc_clt_accept_notification 関数については，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/W，TP1/Client/P 編」を参照してください。

OpenTP1 のサーバ UAP から TP1/Client/W または TP1/Client/P の CUP への通信を次の図に示します。

図 3-64　OpenTP1 のサーバ UAP から TP1/Client/W または TP1/Client/P の CUP への通信




## (3)　ソケット用ファイル記述子の見積もり計算式

トランザクショナル RPC 実行プロセス（clttrnd）のソケット用ファイル記述子の最大数は，ユーザサービスデフォルト定義の max_socket_descriptors オペランドに指定します。

トランザクショナル RPC 実行プロセスのソケット用ファイル記述子の最大数の計算式を次に示します。

```
↑（トランザクショナルRPC実行プロセスが通信するUAPプロセス＋1
＋システムサービスプロセス数）／0.8↑
```

（凡例）

↑↑：小数点以下を切り上げます。

なお，ここに指定した値は，すべての UAP および一部のシステムサーバにも有効になりますので注意してください。

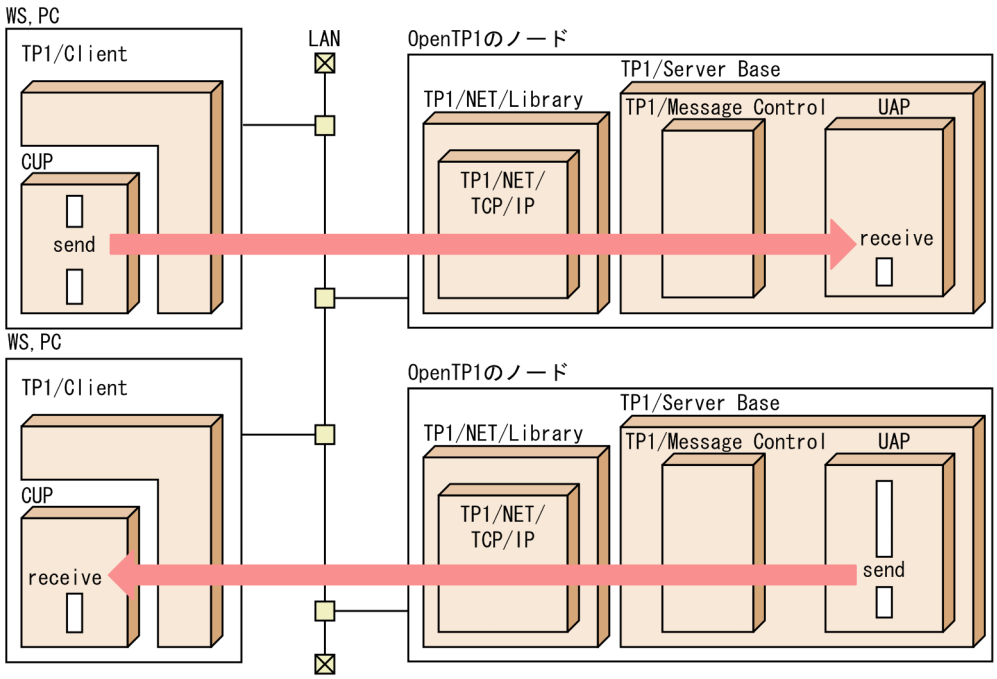
## 3.5.2　TCP/IP プロトコルを使ったメッセージ送受信

TCP/IP プロトコルで接続した LAN を使って，OpenTP1 の UAP とメッセージ送受信形態で通信する方法です。CUP から MHP へメッセージを送信することも，MHP から送られたメッセージを CUP で受信することもできます。

TCP/IP 通信をする場合は，サービス要求先の OpenTP1 システムで TP1/Message Control，TP1/NET/Library，および TCP/IP 用の通信プロトコル対応製品（TP1/NET/TCP/IP）を使っていることが前提となります。

TCP/IP プロトコルを使ったメッセージ送受信の概要を次の図に示します。

図 3-65　TCP/IP プロトコルを使ったメッセージ送受信の概要




## 3.5.3　XDM/DCCM3 との通信

TCP/IP プロトコルで接続した LAN を使って，OpenTP1 以外のサーバと通信する方法です。OpenTP1 へサービスを要求するのと同様の方法で，XDM/DCCM3 と通信できます。

次の OpenTP1 クライアント製品と XDM/DCCM3 が通信する場合，XDM/DCCM3 のホストに DCCM3/SERVER/TP1，または DCCM3/Internet 03-07 以降が組み込まれていることが必要です。

- TP1/Client/W

- TP1/Client/P
- TP1/Client for .NET Framework

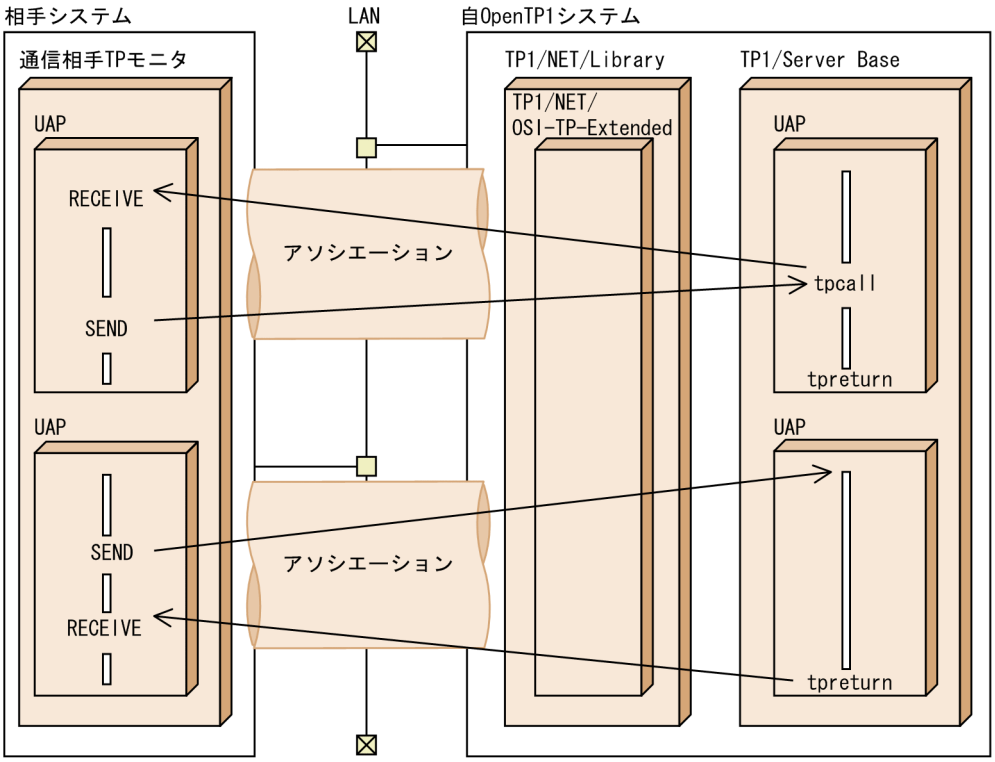
TP1/Client/J と XDM/DCCM3 が通信する場合は，ホストに DCCM3/Internet が組み込まれていることが必要です。

## 3.6　OSI TP を使ったクライアント/サーバ形態の通信

OpenTP1 のクライアント/サーバ形態の通信では，TCP/IP と OSI TP を通信プロトコルに使えます。ここでは，通信プロトコルに OSI TP を使う場合の概要について説明します。通信プロトコルに OSI TP を使う場合には，TP1/NET/Library，TP1/NET/OSI-TP-Extended および OSI TP の通信管理をする製品が必要です。さらに，OpenTP1 のシステムサービス（XATMI 通信サービス）が必要です。

OSI TP を使ったクライアント/サーバ形態の通信の形態を次の図に示します。

図 3-66　OSI TP を使ったクライアント/サーバ形態の通信の形態




## 3.6.1　OpenTP1 の通信相手のシステム

OpenTP1 の通信相手となるシステムは，OpenTP1，または XDM（XDM/DF/TP）です。OSI TP 通信の場合は，通信相手が OpenTP1 以外のシステムの場合でも，自システムの OpenTP1 で開始したトランザクションを相手システムに拡張できます。

OSI TP を使ったクライアント/サーバ形態の通信では，XATMI インタフェースの API を使います。XATMI インタフェースについては，「3.2.4　XATMI インタフェースの通信」を参照してください。

## 3.6.2　通信に使う経路

自システムの OpenTP1 と相手システムの間に論理的な通信路を確立します。この通信路を**アソシエーション**※といいます。アソシエーションが確立すると，クライアント/サーバ形態で通信できるようになります。

アソシエーションは，システムとシステムの接点を示すアドレス（PSAP アドレス）の間に確立します。

注※

このマニュアルでは，通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合にだけ，論理的な通信路を**アソシエーション**と表記します。

### (1)　発呼側と着呼側

OSI TP 通信では，どちらのシステムが確立を要求するかで，確立したアソシエーションの属性が決まります。アソシエーションの確立を要求することを**発呼**，要求されることを**着呼**といいます。アソシエーションの確立を要求した方のシステムを**発呼側**といい，要求された方のシステムを**着呼側**といいます。

OpenTP1 から相手システムにサービスを要求する場合は，OpenTP1 システムが発呼側として確立したアソシエーションを使います。OpenTP1 システムが着呼側として確立したアソシエーションは，主に障害回復の連絡のために使います。

### (2)　アソシエーションの確立

TP1/NET/OSI-TP-Extended の定義にシステム開始時に確立を要求する指定をしておくと，システム開始時にアソシエーションの確立が実行されます。自システムが発呼側となるか着呼側となるかは，TP1/NET/OSI-TP-Extended の定義のプロトコル固有定義 nettalccn に指定します。

TP1/NET/OSI-TP-Extended の定義では，システム開始時に確立を要求する指定をしておいてください。

### (3)　アソシエーションの解放

OpenTP1 システムが正常に終了すると，アソシエーションは正常に解放されます。OpenTP1 システムまたは相手システムが異常終了すると，アソシエーションは異常解放されます。

### (4)　システム定義の指定

OSI TP を使ったクライアント/サーバ形態の通信をするために必要になる OpenTP1 の定義を次に示します。

- XATMI 通信サービス定義
- ネットワークライブラリ定義（TP1/NET/Library と TP1/NET/OSI-TP-Extended の定義）

XATMI 通信サービスと TP1/NET/OSI-TP-Extended を上記の定義で対応づけることで，OSI TP 通信ができるようになります。

## 3.6.3　通信に使うアプリケーションプログラム

OpenTP1 の UAP は，相手システムとの通信に XATMI インタフェースを使います。また，相手システムへトランザクション処理を拡張できます。OSI TP を使ったクライアント/サーバ形態の通信で使える OpenTP1 の UAP は，SUP と SPP です。ほかの OpenTP1 の UAP（MHP）は使えません。

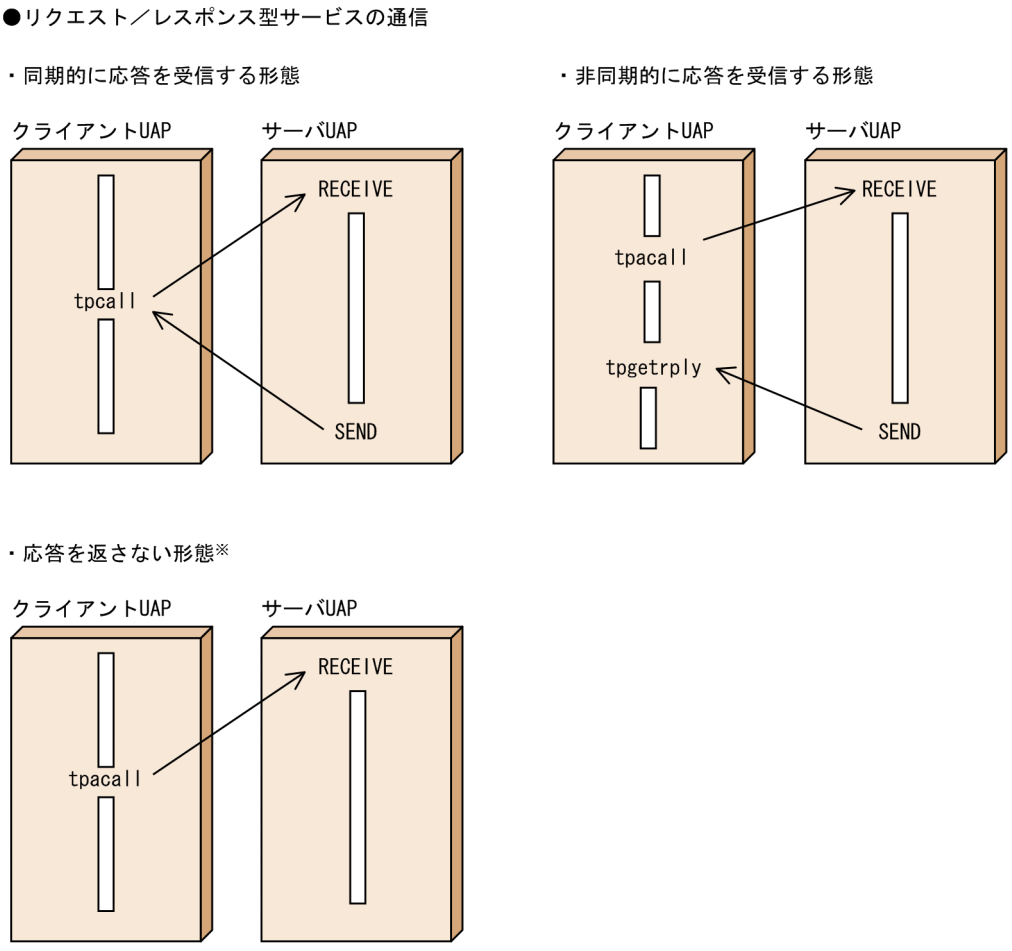
OSI TP を使ったクライアント/サーバ形態の通信の場合に使える XATMI インタフェースの通信形態を次に示します。

- **リクエスト/レスポンス型サービスの通信**（同期的に応答を受信する形態，非同期に応答を受信する形態，応答を返さない形態）

会話型サービスの通信は，OSI TP を使ったクライアント/サーバ形態の通信では使えません。

OpenTP1 がクライアント，XDM/DF/TP がサーバの場合のアプリケーションプログラムの通信の形態を次の図に示します。

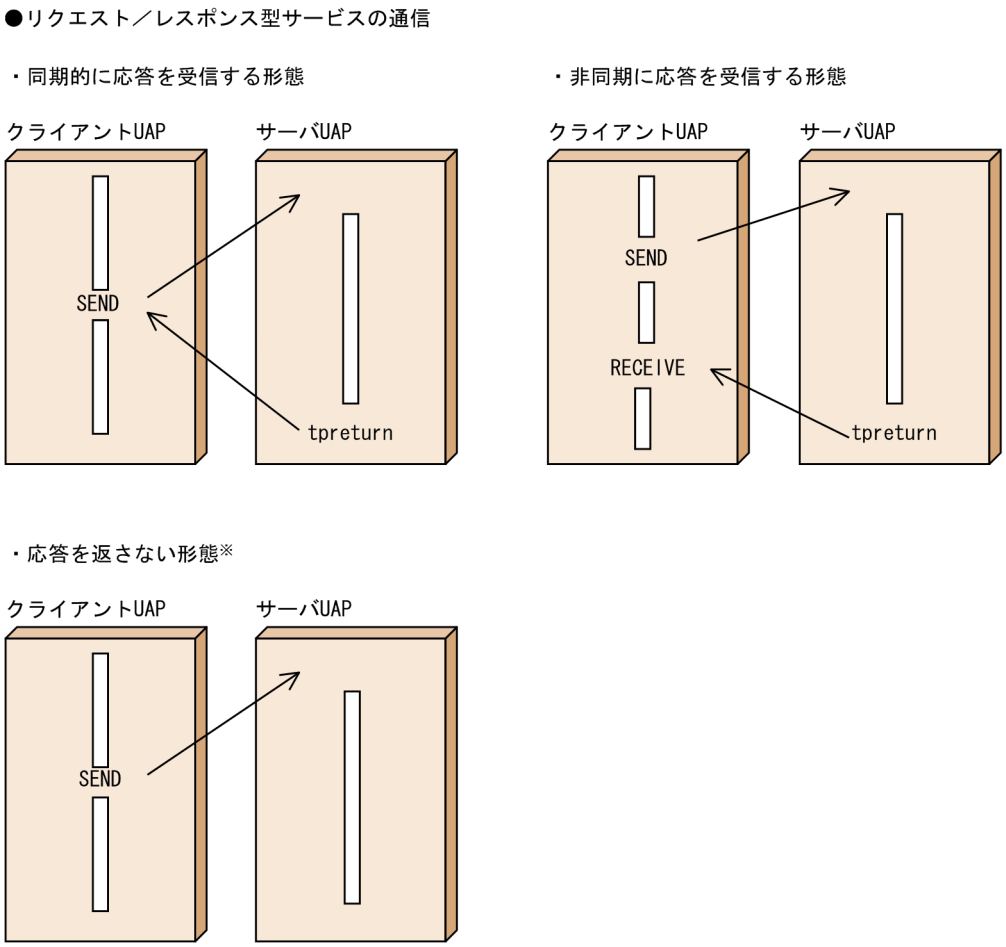
図 3-67　アプリケーションプログラムの通信の形態（OpenTP1 がクライアント，XDM/DF/TP がサーバの場合）




注※　応答を返さない形態では，サービス要求をトランザクションにできません。

XDM/DF/TP がクライアント，OpenTP1 がサーバの場合のアプリケーションプログラムの通信の形態を次の図に示します。

図 3-68　アプリケーションプログラムの通信の形態（XDM/DF/TP がクライアント，OpenTP1 がサーバの場合）




注※　応答を返さない形態では，サービス要求をトランザクションにできません。

## (1)　トランザクション処理との関係

OpenTP1 システムと OpenTP1 システムでは，トランザクション処理を相手システム間で拡張できます。OpenTP1 と OpenTP1 以外のシステムでは，OSI TP を使ってトランザクション処理を相手システム間で拡張できます。

## (2)　アソシエーションの状態を認識する SPP（通信イベント処理用 SPP）

OSI TP を使ったクライアント/サーバ形態の通信をする場合，アソシエーションの確立と解放を知るための SPP を作成する必要があります。この SPP を**通信イベント処理用 SPP** といいます。通信イベント処理用 SPP を作成すると，アソシエーションの解放の通知イベントを SPP で受信できます。この通信イベントを受信することで，アソシエーションの再確立のきっかけを知ることができます。アソシエーションを再確立するときは，**dc_xat_connect 関数**を使います。

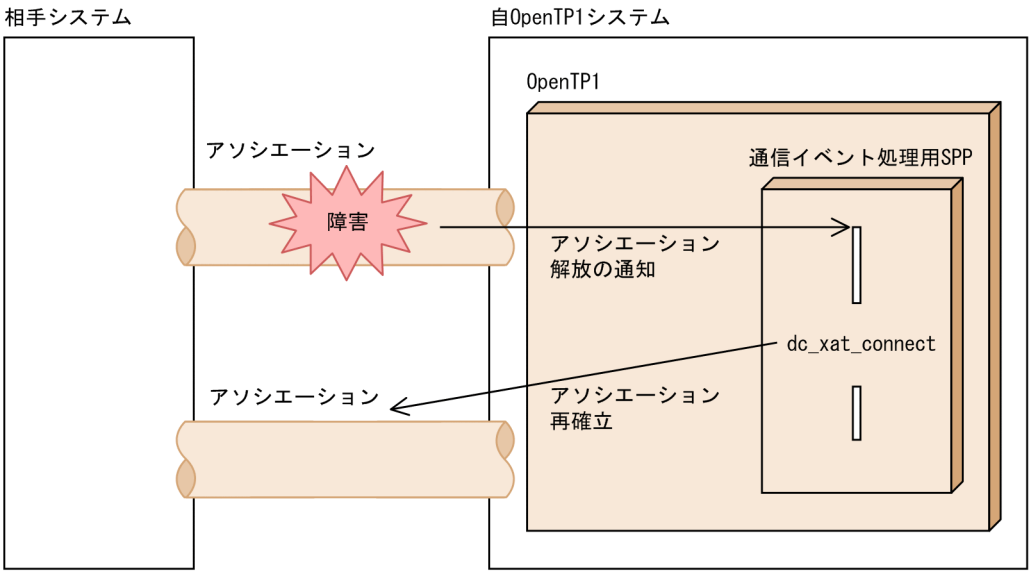
通信イベントは，自システムが発呼側か着呼側かどうかに関係なく通知されます。通信イベント処理用SPP では，通知イベントの詳細情報からアソシエーションの属性や状態について知ることができます。

通信イベント処理用 SPP のサービスグループ名とサービス名は，あらかじめ XATMI 通信サービス定義に指定しておきます。この指定に従って，通信イベントが通知されます。

通信イベント処理用 SPP と受信するイベントについては，マニュアル「OpenTP1 プログラム作成の手引」および「OpenTP1 プログラム作成リファレンス」の該当する言語編を参照してください。

通信イベント処理用 SPP の概要を次の図に示します。

図 3-69　通信イベント処理用 SPP の概要




## 3.6.4　環境設定の概要

クライアント/サーバ形態の通信プロトコルに OSI TP を使う場合の，OpenTP1 システムの環境設定手順を次に示します。(2)と(3)の手順は，通常の OpenTP1 の作業と同じです。

### (1)　OpenTP1 の製品の組み込み

OSI TP 通信に必要な，次に示す製品を組み込みます。

TP1/Server Base，TP1/NET/Library，TP1/NET/OSI-TP-Extended

### (2)　TP1/Server Base の定義の作成

スーパユーザが OpenTP1 管理者を決定したあとで，OpenTP1 管理者は TP1/Server Base の定義を作成します。

### (3) OpenTP1 の登録

OpenTP1 を OS に登録するため，dcsetup コマンドを実行します。

### (4) TP1/NET/OSI-TP-Extended の定義の作成

TP1/NET/OSI-TP-Extended の定義を作成します。定義ファイルは，定義を解析するユティリティでオブジェクトファイルにしたあと，OpenTP1 のシステム定義を格納するディレクトリの下に次のファイル名で格納します。

- $DCCONFPATH/xatcex

OpenTP1 で TP1/NET/OSI-TP-Extended を使う場合，TP1/NET/Library を登録するコマンド（netsetup コマンド）を実行する必要はありません。

OpenTP1 のシステム設定手順については，「5.1 OpenTP1 システムの環境設定手順」を参照してください。また，OpenTP1 以外の前提プログラムについては，マニュアル「OpenTP1 プロトコル TP1/NET/OSI-TP-Extended 編」を参照してください。

## 3.6.5 障害が起こった場合

OSI TP を使ったクライアント/サーバ形態の通信で障害が起こった場合，サービスを要求した XATMI インタフェースの関数はエラーリターンします。どのリターン値が返るかについては，「OpenTP1 プログラム作成リファレンス」の該当する言語編にある XATMI インタフェースの各関数の注意事項を参照してください。

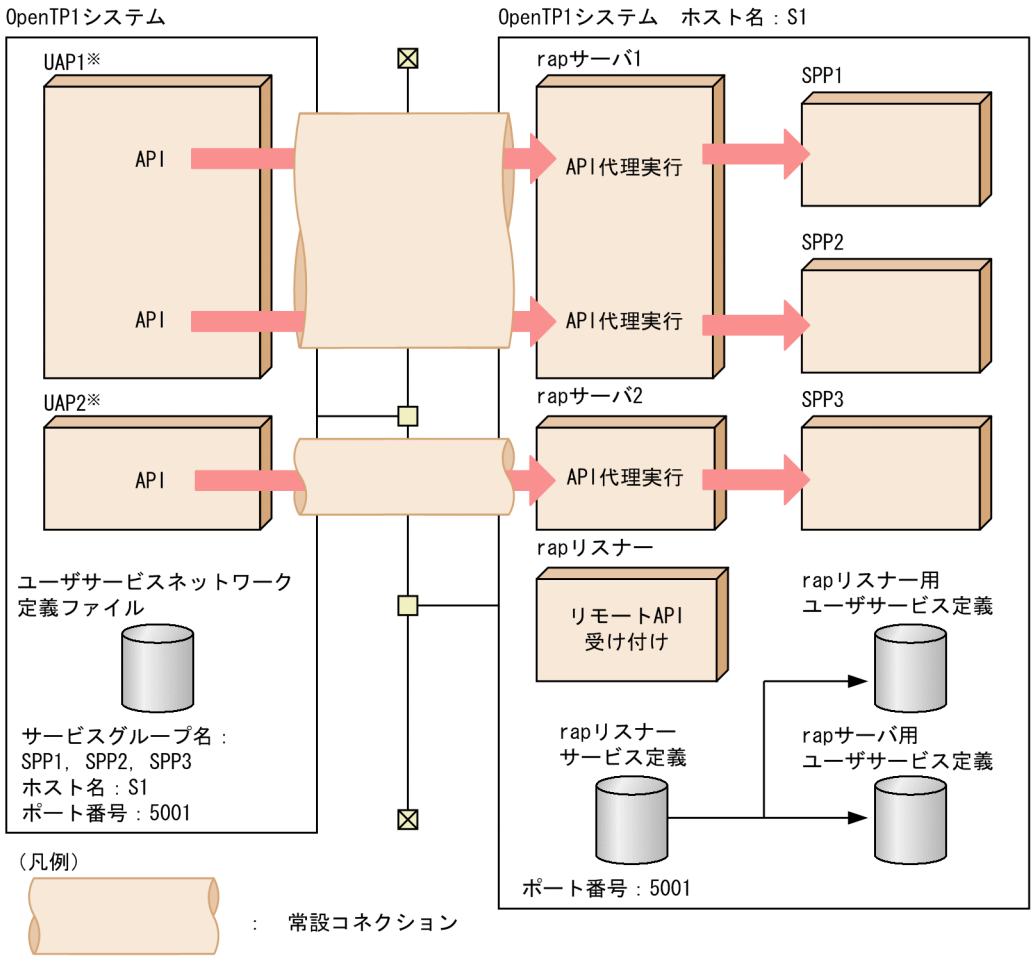
通信プロトコルで障害が起こった場合の処置については，マニュアル「OpenTP1 プロトコル TP1/NET/OSI-TP-Extended 編」の障害対策の説明を参照してください。

## 3.7　リモート API 機能

**リモート API 機能**とは，クライアント側のノードにある UAP が発行した API を，OpenTP1 がサーバ側に転送してサーバ側のプロセスで代理実行する機能です。リモート API 機能を要求するクライアント側のノードにある UAP を **rap クライアント**といいます。rap クライアントが発行した API を，OpenTP1 の **rap リスナー**が受け付け，**rap サーバ**がサーバ側のノードで実行します。rap リスナー，rap サーバは OpenTP1 のユーザサービスとして動作します。

リモート API 機能を次の図に示します。

### 図 3-70　リモート API 機能




注　　リモートAPI機能では，XATMIインタフェースを使用してはいけません。
　　XATMIインタフェースを使用した場合，OpenTP1の動作は保証されません。
注※　リモートAPI機能を使用するUAPはユーザサービス定義のrpc_destination_modeオペランドで
　　definitionを指定します。

rap クライアントとなる UAP を次の表に示します。

### 表 3-20　rap クライアントとなる UAP

| プログラムプロダクト | rap クライアントとなる UAP |
|---|---|
| TP1/Server Base，TP1/LiNK | SUP，SPP，MHP |

| プログラムプロダクト | rap クライアントとなる UAP |
|---|---|
| TP1/Client | CUP |

次に，rap クライアントから代理実行できる API を rap クライアントの種類ごとに示します。

- TP1/Server Base，TP1/LiNK が rap クライアントとなる場合

| C 言語ライブラリ | COBOL-UAP 作成用プログラム |
|---|---|
| dc_rpc_call | CBLDCRPC（'CALL '） |

- TP1/Client/P，TP1/Client/W が rap クライアントとなる場合

| C 言語ライブラリ | COBOL-UAP 作成用プログラム |
|---|---|
| dc_rpc_call_s | CBLDCRPS（'CALL '） |
| dc_trn_begin_s | CBLDCTRS（'BEGIN '） |
| dc_trn_chained_commit_s | CBLDCTRS（'C-COMMIT'） |
| dc_trn_chained_rollback_s | CBLDCTRS（'C-ROLL '） |
| dc_trn_unchained_commit_s | CBLDCTRS（'U-COMMIT'） |
| dc_trn_unchained_rollback_s | CBLDCTRS（'U-ROLL '） |

- TP1/Client/J が rap クライアントとなる場合

| メソッド |
|---|
| rpcCall |
| trnBegin |
| trnChainedCommit |
| trnChainedRollback |
| TrnUnchainedCommit |
| trnUnchainedRollback |

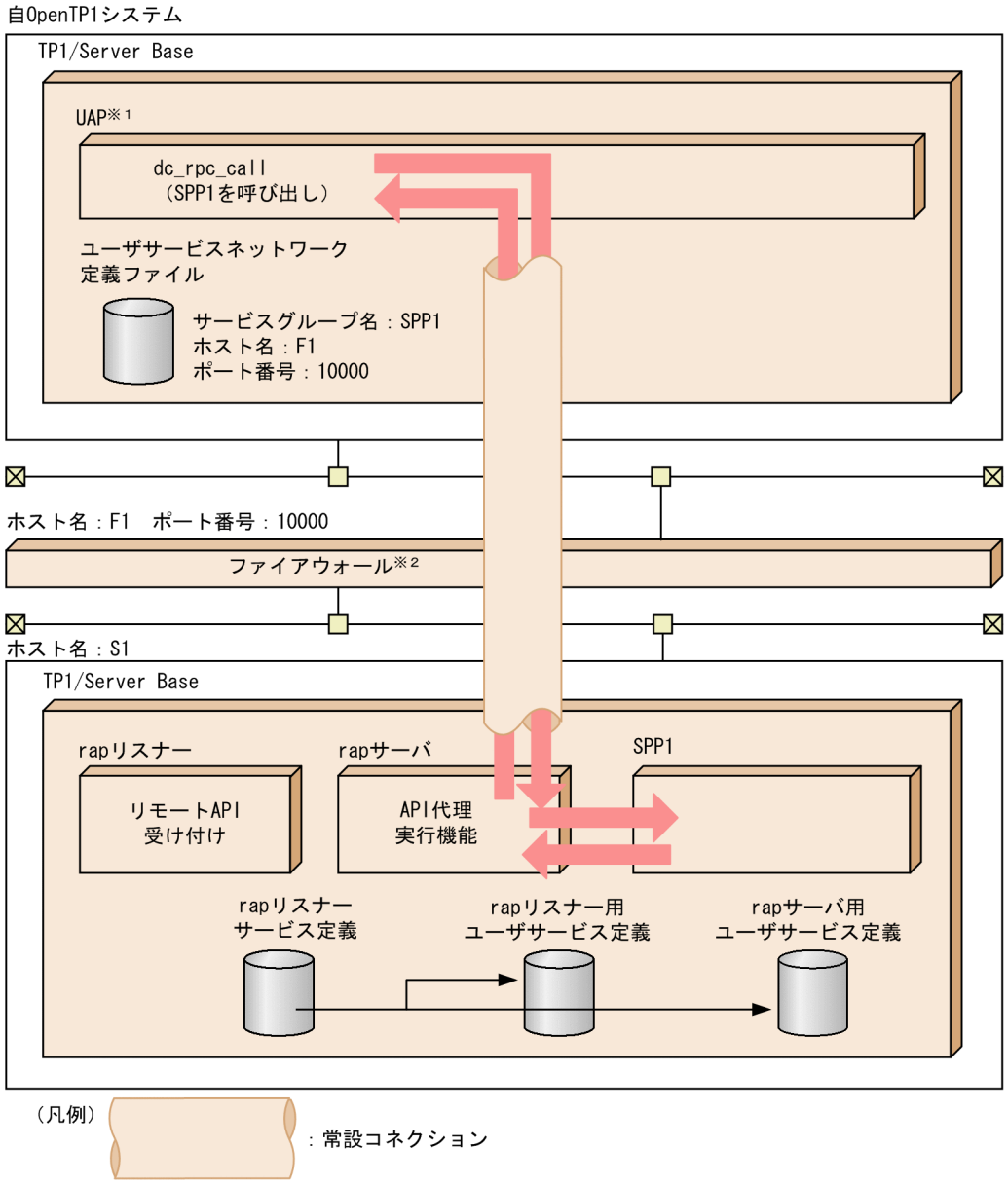
- TP1/Client for .NET Framework が rap クライアントとなる場合

| メソッド |
|---|
| Call |
| Begin |
| CommitChained |
| RollbackChained |
| Commit |
| Rollback |

## 3.7.1　リモート API 機能の使用例

リモート API 機能を使うと，**ファイアウォール**の内側にある UAP に対しても，サービスを要求できます。ファイアウォールの内側へのリモートプロシジャコールを次の図に示します。なお，TP1/Client を使用する場合については，マニュアル「OpenTP1 クライアント使用の手引」を参照してください。

図 3-71　ファイアウォールの内側にある UAP へのリモートプロシジャコール




注1　ファイアウォールを通過する場合は，次の点に注意してください。

・非同期応答型RPCを使用してはいけません。
・ファイアウォールを通過した場合，通過先のUAP（この図の場合　SPP1）は，トランザクションブランチになりません。

注2　リモートAPI機能では，XATMIインタウェースを使用してはいけません。
XATMIインタフェースを使用した場合，OpenTP1の動作は保証されません。

注※1　UAPはユーザサービス定義のrpc_destination_modeオペランドで，definitionを指定します。
注※2　ファイアウォールにはGauntletを想定しています。

## 3.7.2　常設コネクション

OpenTP1 は，リモート API を要求した UAP（rap クライアント）と rap サーバとの間に，論理的な通信路（**常設コネクション**）を設定します。

常設コネクションのスケジュール方法には，スタティックコネクションスケジュールモードとダイナミックコネクションスケジュールモードがあります。どちらを使用するかは，rap リスナーサービス定義の rap_connection_assign_type オペランドで指定できます。

### (1)　スタティックコネクションスケジュールモード

スタティックコネクションスケジュールモードを使用するには，rap_connection_assign_type オペランドに static を指定します。または指定を省略します。

スタティックコネクションスケジュールモードは，rap クライアントがコネクションの確立を要求すると同時に rap サーバを割り当て，rap クライアントと rap サーバを 1 対 1 で関係づける常設コネクションのスケジュール方法です。同時に使用する rap クライアント数が，開始する rap サーバ数以内の場合に適しています。なお，OpenTP1 の制限のため，rap サーバの最大数は 1024 です。

スタティックコネクションスケジュールモードを使用する場合，rap リスナーが管理できる常設コネクションの最大数は 1024 です。

rap リスナーサービス定義の rap_max_client オペランドには，rap リスナーに同時接続する最大クライアント数を指定できます。ただし，OpenTP1 の制限のため，rap_max_client オペランドおよび max_socket_descriptors オペランドの和が 1993 以下になるように調整してください。

また，同時に常設コネクションが設定され，実行できる rap クライアント数は rap サーバ数が上限です。残りの常設コネクションは，現在実行中の rap クライアントが常設コネクションを解放するまで常設コネクションの確立はできません。そのため，rap クライアントが rap サーバ数以上の場合には，次に示すどちらかの運用をしてください。

- API 代理実行を要求する前後で，rap クライアントで常設コネクションの確立および解放を実行してください。
- rap リスナーを複数開始して，rap クライアントが常設コネクションを確立する rap サーバ数を増やしてください。

### (2)　ダイナミックコネクションスケジュールモード

ダイナミックコネクションスケジュールモードを使用するには，rap_connection_assign_type オペランドに dynamic を指定します。

ダイナミックコネクションスケジュールモードは，rap クライアントがコネクションの確立を要求するときに rap サーバを割り当てないで，API 代理実行の要求が発生したときに動的に未使用の rap サーバを割り当てるコネクションスケジュール方法です。同時に使用する rap クライアント数が rap サーバ数以上あるが，rap サーバに必要な資源を増加させたくない場合に適しています。

ダイナミックコネクションスケジュールモードを使用する場合，rap リスナーが管理できる常設コネクションの最大数は 1024 です。rap リスナーサービス定義の rap_max_client オペランドには rap リスナーに同時接続する最大クライアント数を指定できます。ただし，OpenTP1 の制限のため，rap_parallel_server オペランド，rap_max_client オペランドおよび max_socket_descriptors オペランドの和が 1993 以下になるように調整してください。

ダイナミックコネクションスケジュールモードでは，rap クライアントからの API 代理実行の要求を受け取ったときにだけ rap サーバを割り当て，API 代理実行の終了後には rap サーバを解放します。このため，スタティックコネクションスケジュールモードで rap クライアントが rap サーバ数以上となった場合に行う次の対策は必要ありません。そのため，rap リスナーを複数開始する必要がないのでファイアウォールを通過させるための通信ポートを増やす必要もありません。

- API 代理実行の要求をするたびに常設コネクションの確立と解放を実行する
- rap リスナーを複数開始して，rap クライアントが常設コネクションを確立する rap サーバ数を増やす

ただし，ダイナミックコネクションスケジュールモードは，rap リスナーへの負荷が高くなるとスタティックコネクションスケジュールモードに比べてレスポンス性能が悪くなります。

## 3.7.3 コネクトモード

常設コネクションの管理方法は，コネクションの確立・解放方法によって二つに分けられます。コネクションの確立，解放を OpenTP1 が管理する形態を**オートコネクトモード**，ユーザが管理する形態を**非オートコネクトモード**といいます。ユーザはコネクションをオートコネクトモードで管理するか，非オートコネクトモードで管理するかを，rap クライアントのユーザサービス定義で定義します。

### (1) オートコネクトモード

OpenTP1 が常設コネクションの確立・解放を管理する形態です。rap クライアントが，ユーザサービスネットワーク定義に-w オプション付きで定義されているサービスグループ名を指定して dc_rpc_call を発行した場合に，自動的に常設コネクションを確立します。

ユーザサービスネットワーク定義に定義されているサービスグループに dc_rpc_call を発行した時点から，dc_rpc_close を発行してリモートプロシジャコールを終了するまで常設コネクションを確保します。

### (2) 非オートコネクトモード

ユーザが常設コネクションの確立・解放を管理する形態です。ユーザは rap クライアントでコネクションを確立・解放するための API（dc_rap_connect・dc_rap_disconnect）を発行します。rap クライアントが，ユーザサービスネットワーク定義に-w オプション付きで定義されているサービスグループ名を指定して dc_rpc_call を発行したときに，ユーザが常設コネクションを確立していないと，dc_rpc_call がエラーになります。

### 3.7.4　rap クライアントマネジャ

rap クライアントが常設コネクションを確立していると認識していても，そのコネクションが有効でない場合（rap サーバ側マシンの電源の切断など）があります。この場合，rap クライアントが，dc_rpc_call を発行して rap サーバにサービス要求を送信しても，サービス要求は rap サーバで受信されません。さらに，rap クライアントには，タイムアウトになるまで dc_rpc_call の応答が返ってきません。この現象を回避するために，rap クライアントマネジャサービス定義を定義し，rap クライアントマネジャを起動します。

rap クライアント側で動作する rap クライアントマネジャは，rap リスナーからの起動通知を受信したとき，通知元の rap リスナーを識別する情報と起動通知を受信した時刻を rap クライアントマネジャの管理テーブルに保持します。rap クライアントは，rap リスナーとコネクションを確立するときにこの管理テーブルを参照して起動通知を受信した時刻を保存しておきます。rap クライアントは，dc_rpc_call 関数を発行するとき，再度この管理テーブルの起動通知を受信した時刻を参照して，コネクション確立時に保存しておいた値と比較します。比較した結果，起動通知時刻が一致していれば，現在のコネクションが有効であると判断し，不一致であれば現在のコネクションが無効であると判断します。

コネクションが無効と判断した場合の動作は，コネクトモードによって異なります。

- オートコネクトモード

  不要なコネクションを解放し，再度コネクションを確立したあとで，dc_rpc_call 関数を発行してサービス要求を送信します。
- 非オートコネクトモード

  dc_rpc_call 関数は DCRPCER_NET_DOWN でエラーリターンします。

rap クライアントマネジャの動作環境を設定するには，rapsetup コマンドを実行してください。rap クライアントマネジャを起動するには，rapdfgen コマンドを実行して rap クライアントマネジャ用ユーザサービス定義を生成してください。

注

rap クライアントマネジャが異常終了したあとに再起動，または正常終了した場合，rap クライアントは現在のコネクションを無効と判断し，強制的にコネクションを切断することがあります。強制的に切断したコネクションが有効であった場合，rap リスナー側で，KFCA26965-E メッセージ（スタティックコネクションスケジュール機能使用時），または KFCA26956-W メッセージ（ダイナミックコネクションスケジュール機能使用時）が出力されます。

KFCA26965-E メッセージ，KFCA26956-W メッセージの詳細については，マニュアル「OpenTP1 メッセージ」を参照してください。

### 3.7.5　リモート API 機能を使用する場合に設定する必要がある定義

TP1/Server Base 間でリモート API 機能を使用する場合に設定する必要がある定義は次のとおりです。

- rap クライアント側

ユーザサービス定義
ユーザサービスネットワーク定義
rap クライアントマネジャサービス定義（任意）

- rap サーバ側
  rap リスナーサービス定義

### 3.7.6　XA リソースサービスを使用する場合

XA リソースサービスを使用するためには，リモート API 機能と，次に示す製品が必要です。

J2EE で動作するアプリケーションサーバと連携する場合

- TP1/Client/J
- uCosminexus TP1 Connector または Cosminexus TP1 Connector

.NET Framework アプリケーションと連携する場合

- Client .NET
- Connector .NET

XA リソースサービスは，常設コネクションのスケジュール方式の，スタティックコネクションスケジュールモードとダイナミックコネクションスケジュールモードのどちらのモードを使用しても動作します。

XA リソースサービスの詳細については，「3.1.7　XA リソースサービスによるトランザクション制御」を参照してください。

## 3.8　サービス関数動的ローディング機能

サービス関数動的ローディング機能とは，UAP 共用ライブラリ化したサービス関数を，動的にローディングする（読み込む）機能です。定義を変更するだけでサービスを追加または削除できます。なお，UAP 共用ライブラリ化とは，UAP のソースファイルを翻訳（コンパイル）して作成した UAP オブジェクトファイルを結合（リンケージ）して，共用ライブラリとしてまとめることです。

サービス関数動的ローディング機能を使うと，サービス関数を追加または削除する場合に，スタブの変更，および UAP の実行形式ファイルの再生成をしなくても，ユーザサービス定義の service オペランドを変更するだけでサービス関数を追加または削除できます。UAP 起動時にサービス関数をローディングするため，UAP の実行形式ファイル作成時には，スタブおよびサービス関数は不要です。また，RPC インタフェース定義を使わないため，既存の UAP に新たにサービスを追加または削除する場合，サービスの追加または削除に伴う UAP の実行形式ファイルの再生成は不要です。

このように作業を省略できるため，特に頻繁にサービスを追加または削除するシステムで有効に適用できます。

**注意事項**

サービス関数動的ローディング機能は，SPP または MHP の場合にだけ使用できます。ただし，次の場合は，サービス関数動的ローディング機能を使用できません。

- XATMI インタフェースを使ったユーザサーバの場合
- TP1/Offline Tester を使用している場合

### 3.8.1　サービス関数動的ローディング機能の使用例

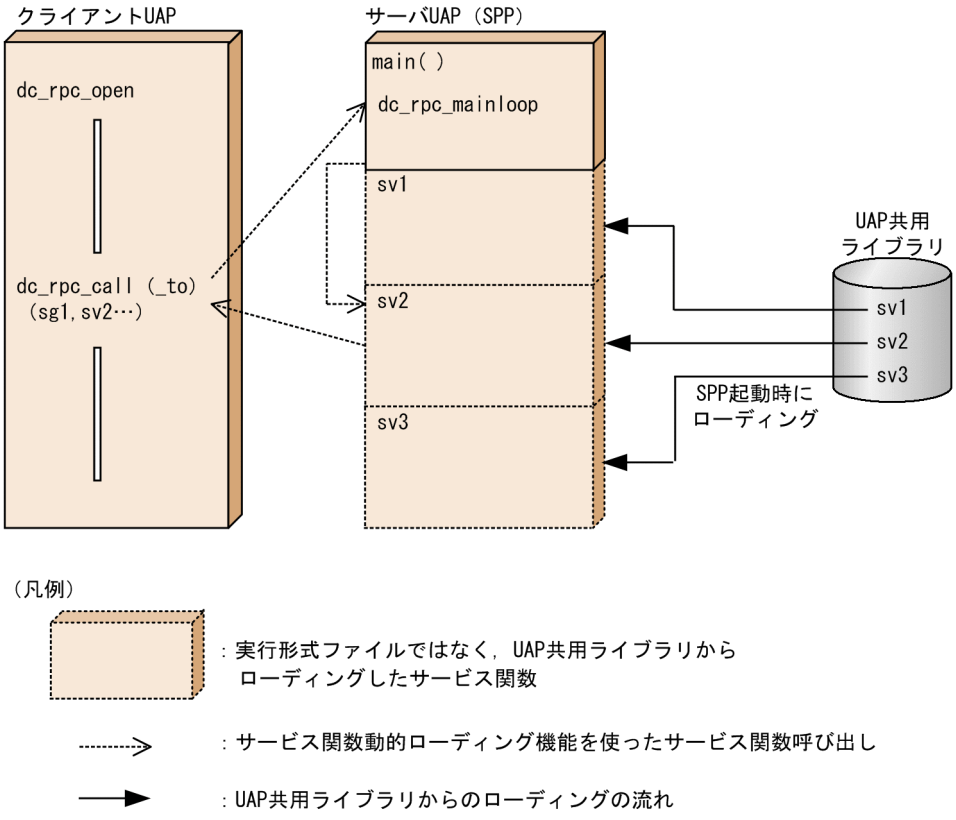
サービス関数動的ローディング機能を使う場合，サービス関数動的ローディング機能だけを使用したり，サービス関数動的ローディング機能とスタブを併用したりできます。

#### (1)　サービス関数動的ローディング機能だけを使用する例

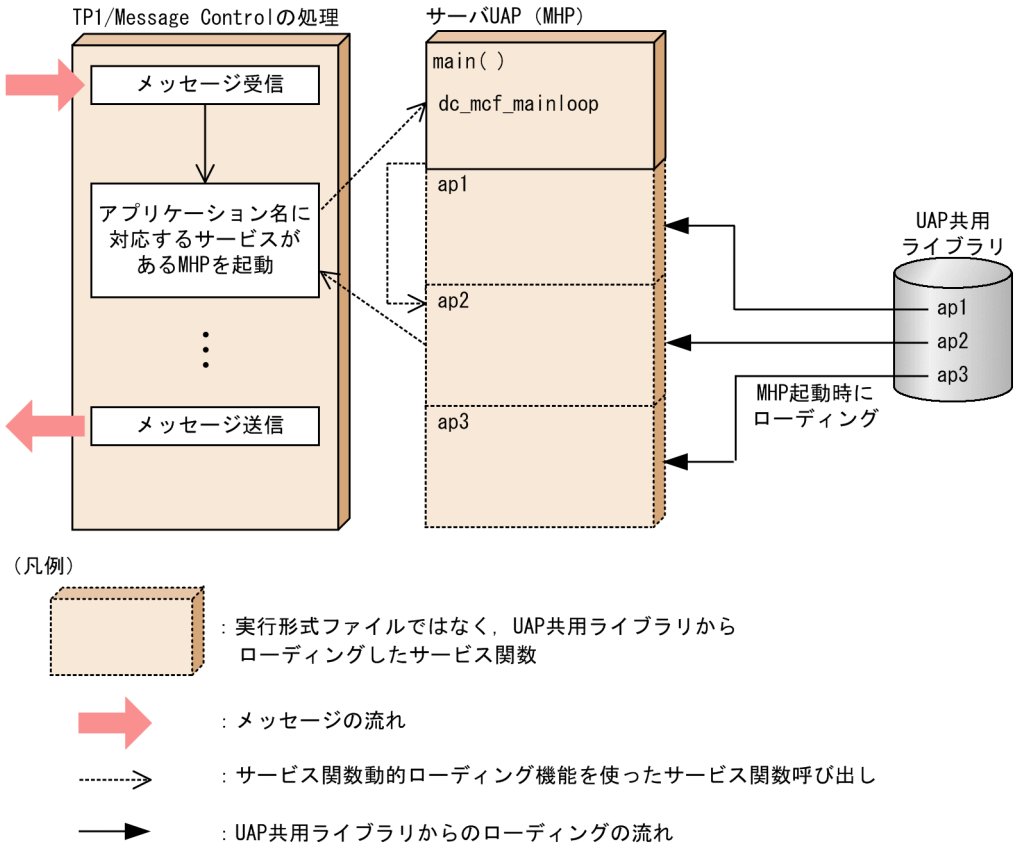
サーバ側でクライアントからの要求を受信すると，UAP 共用ライブラリから取得したサービス関数を実行します。

サービス関数動的ローディング機能を使った UAP の例を，SPP の場合と MHP の場合に分けてそれぞれを以降の図に示します。

## 図 3-72　サービス関数動的ローディング機能を使った UAP（SPP）の例




## 図 3-73　サービス関数動的ローディング機能を使った UAP（MHP）の例



## (2)　サービス関数動的ローディング機能とスタブを併用する例

サービス関数動的ローディング機能は，スタブとサービス関数を使った SPP と併用できます。併用する場合，サービスを追加または削除するときに，スタブとサービス関数を使った UAP の実行形式ファイルを再生成する必要はありません。

サービス関数動的ローディングとスタブを併用した UAP の例を，SPP の場合と MHP の場合に分けてそれぞれを以降の図に示します。

図 3-74　サービス関数動的ローディング機能とスタブを併用した UAP（SPP）の例

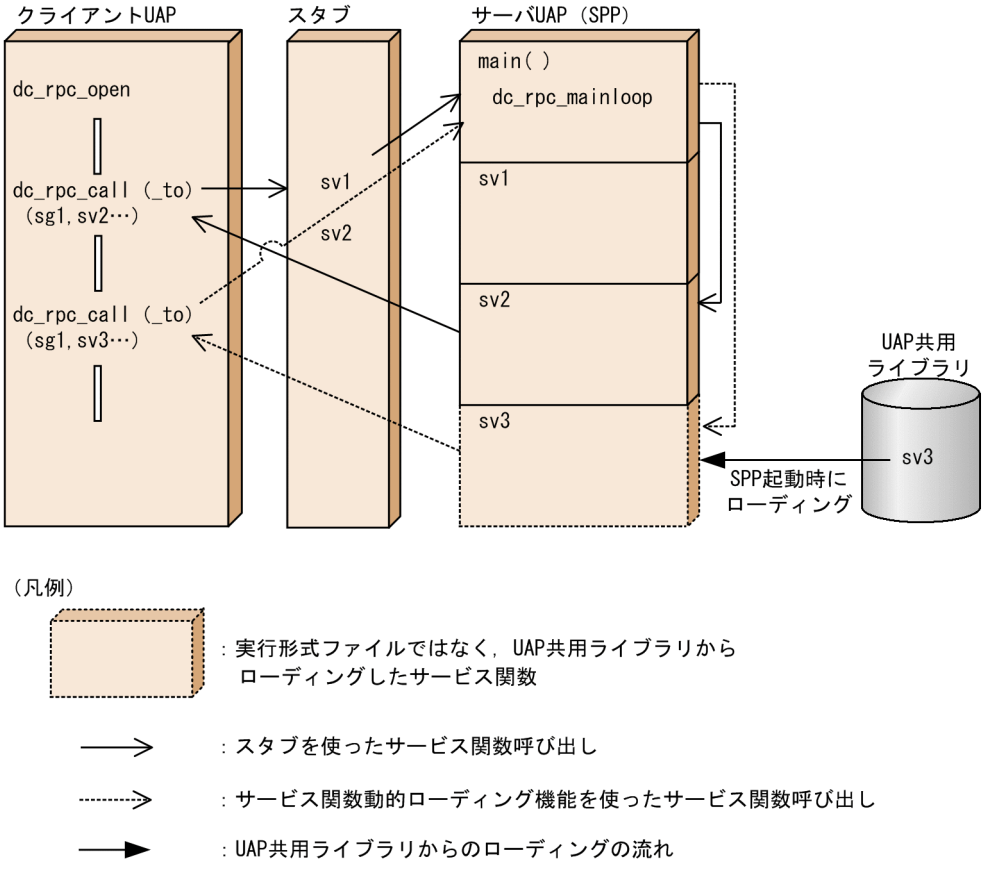
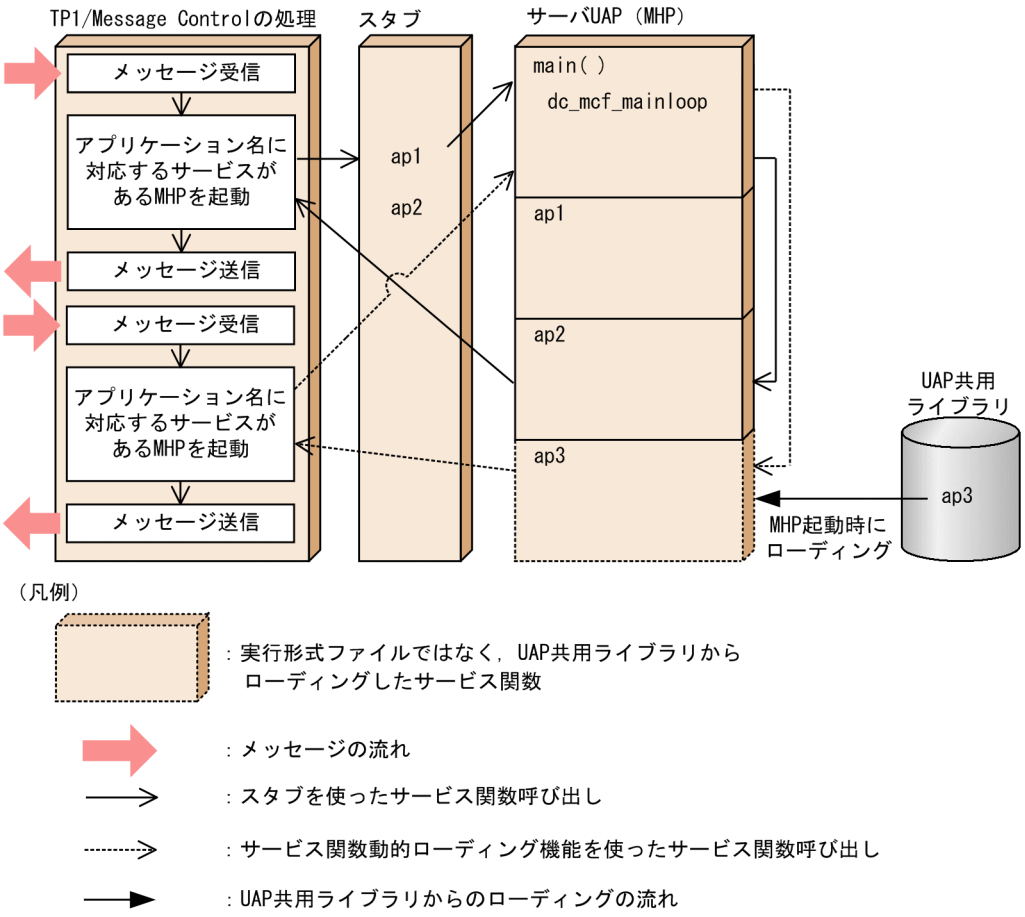


図 3-75　サービス関数動的ローディング機能とスタブを併用した UAP（MHP）の例




## 3.8.2　サービス関数動的ローディング機能を使用する場合に必要な準備

サービス関数動的ローディング機能を使う場合に必要な事前準備について説明します。

### (1)　設定の手順

サービス関数動的ローディング機能を使う場合に必要な作業を次に示します。

1. サービス関数から，UAP 共用ライブラリを作成します。
2. ユーザサービス定義の service オペランドに，サービス名，エントリポイント名，および UAP 共用ライブラリ名を設定します。

service オペランドの詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

### (2)　再リンケージの実行条件

サービス関数を追加する場合に，UAP オブジェクトの再リンケージが必要かどうかを，次の表に示します。

表 3-21　再リンケージの実行条件

| サービスの追加または削除のベースとなる UAP オブジェクト | サービスの追加方法 | 再リンケージ |
|---|---|---|
| RPC インタフェース定義だけを使った UAP オブジェクト | RPC インタフェース定義を使う場合。 | ○ |
| | サービス関数動的ローディング機能を使う場合。 | × |
| サービス関数動的ローディング機能だけを使った UAP オブジェクト | RPC インタフェース定義を使う場合。 | ○ |
| | サービス関数動的ローディング機能を使う場合。 | × |
| RPC インタフェース定義およびサービス関数動的ローディング機能を併用した UAP オブジェクト | RPC インタフェース定義を使う場合。 | ○ |
| | サービス関数動的ローディング機能を使う場合。 | × |

（凡例）

○：再リンケージが必要です。

×：再リンケージは不要です。

## 3.9　OpenTP1 の運用を補助する機能

ここまでに説明したシステムサービス以外で，OpenTP1 の運用を補助するための機能について説明します。

OpenTP1 提供以外のリソースマネジャを制御する場合は，OpenTP1 システムに TP1/Resource Manager Monitor を組み込んでおく必要があります。それ以外は，TP1/Server Base で標準で提供する機能です。

### 3.9.1　資源の排他制御

OpenTP1 では，複数のユーザが DAM ファイルや TAM ファイル以外の資源も共用できるようにしています。資源を管理して整合性を保つことができる機能を**排他制御**といいます。排他資源の占有，および待ち状態は lckls コマンドで表示できます。

#### (1)　排他制御の管理単位と有効範囲

OpenTP1 は，資源の排他要求時から排他資源の解放要求時まで排他を管理します。排他制御の管理単位は，**グローバルトランザクション**です。ただし，DAM サービスが使用する排他制御機能の管理単位は，トランザクションブランチです。

排他制御の有効範囲は，一つの OpenTP1 システム内だけです。ほかの OpenTP1 システム上の UAP との排他制御はできません。また，複数のノードから構成される一つの OpenTP1 システム内でも，ノード間の排他制御はできません。

#### (2)　資源の排他要求

資源を確保するときは，UAP から dc_lck_get 関数を呼び出します。このとき，引数に資源名称と**排他制御モード**を指定します。排他制御モードとは，同じ資源にアクセスするほかの UAP に対する排他の方式です。

排他制御モードには，資源を参照するだけの UAP とは資源を共用し，資源を更新する UAP だけを排他する **PR モード**と，一つの UAP が資源を占有し，ほかの UAP をすべて排他する **EX モード**があります。

排他制御モードを次の表に示します。

表 3-22　排他制御モード

| 排他制御モード | 資源の処理 | 排他の内容 |
|---|---|---|
| PR | 参照 | ほかの UAP に対して，参照だけを許可します。 |
| EX | 更新 | ほかの UAP に対しては，参照，更新を許可しません。 |

排他制御モードの組み合わせによる資源の共用を次の表に示します。

表 3-23　排他制御モードの組み合わせによる資源の共用

| 使用中の排他制御モード | 要求元の排他制御モード | |
|---|---|---|
| | PR | EX |
| PR | ○ | × |
| EX | × | × |

（凡例）

○：共用できます。

×：共用できません。

## (3)　資源の解放

ユーザは，次に示す方法で資源を解放します。

1. 資源名称ごとの**排他解除要求**を UAP が出す
2. 全資源の**一括排他解除要求**を UAP が出す

また，トランザクションの終了時に，システムが全資源を自動的に解放します。

これによって，UAP での解放漏れや，UAP 異常終了が発生した場合でも，資源が不当に占有されることがなくなります。

## (4)　排他待ち

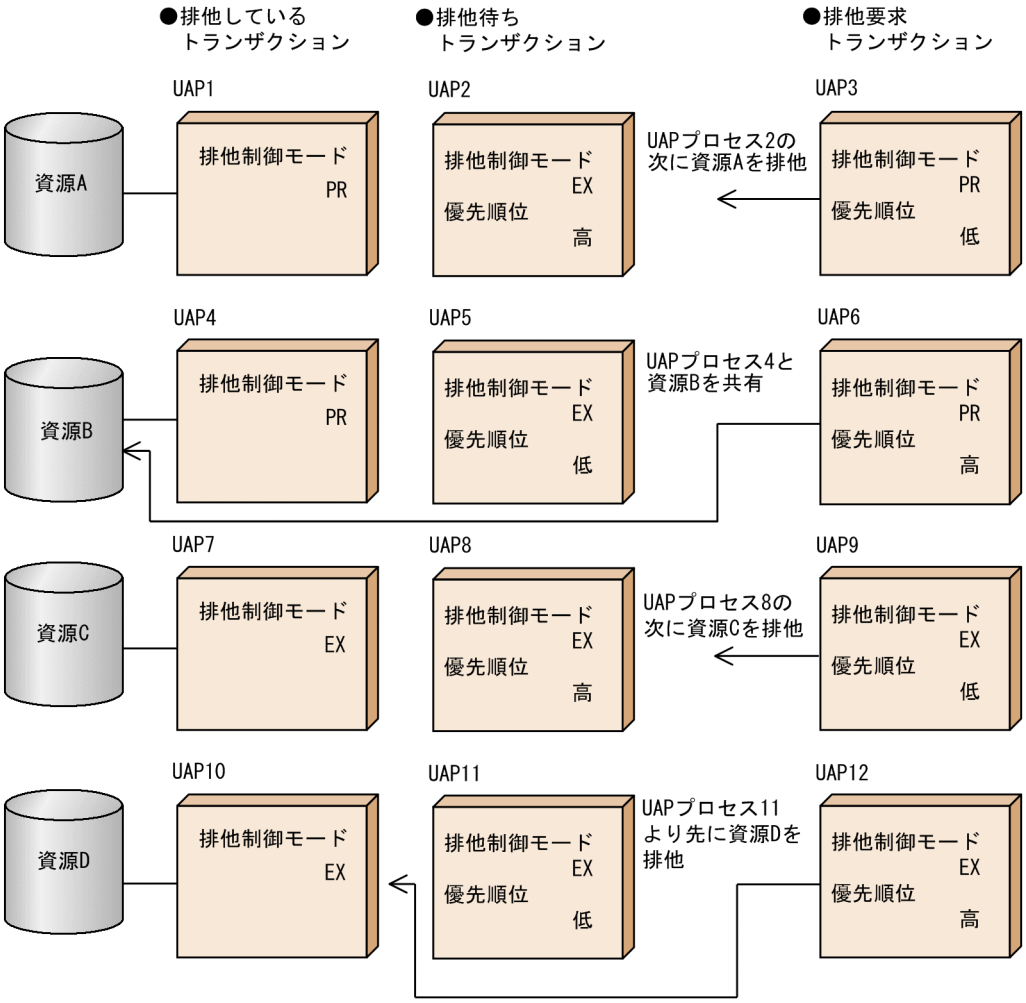
資源を排他できない場合，資源を排他できるまで待つかどうか，dc_lck_get 関数の引数で指定します。排他待ちするのは，次の場合です。

- 排他要求したときに，ほかの UAP がすでに EX モードで資源に排他をかけている場合
- 排他要求したときに，ほかの UAP がすでに PR モードで資源に排他をかけていて，かつ自 UAP で EX モードの排他をかける場合

資源が解放されるまで待つと指定した場合には，**排他待ちの優先順位**に従って，ほかの UAP が資源の排他を解除するまで待ちます。待たないと指定した場合は，すぐにエラーリターンします。排他待ちの優先順位は，ユーザサービス定義で UAP のサービスグループごとに定義します。優先順位を高く設定すれば，排他待ちしている，優先順位の低いほかの UAP よりも先に資源を排他できます。逆に，優先順位を低く設定した UAP は，あとから排他要求した優先順位の高い UAP に追い越されることがあるので，排他待ちの優先順位の設定には注意が必要です。排他待ちの優先順位と排他順序を次の図に示します。

ロックサービス定義で**排他待ち限界経過時間**を指定すると，排他待ち状態がこの時間を超えたとき，その排他要求をエラーリターンします。

図 3-76　排他待ちの優先順位と排他順序




## (5)　デッドロックの対処方法

複数の UAP が複数の資源に対して異なる順番で排他要求を行った場合，それぞれの UAP が相手の占有する資源の排他解除を互いに待ち続けて，処理が止まってしまう場合があります（**デッドロック**）。

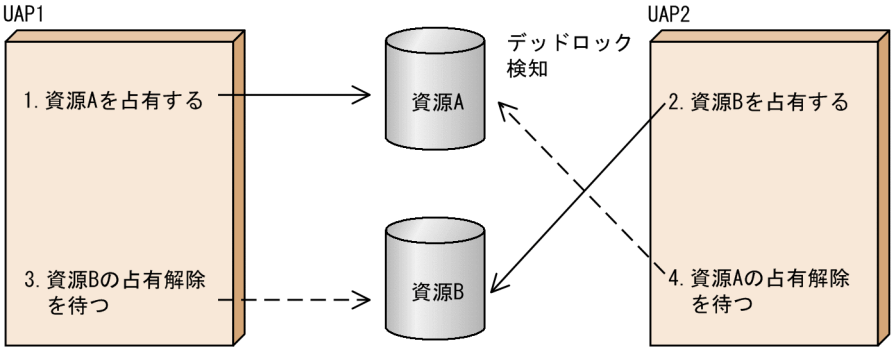
OpenTP1 は，デッドロック状態を検出して速やかに解消するため，定期的にデッドロック状態になっているかどうかを監視しています。

同一ノード内の UAP 間では，定期的に**排他待ち状態**を監視してデッドロックを検知します。

異なるノードの UAP に RPC でサービス要求して，そのサービスとの間でデッドロックが起こっても，検知できません。そのため**排他待ち時間監視**をする必要があります。排他待ち限界経過時間は，ロックサービス定義で指定します。

デッドロックの例を次の図に示します。

図 3-77　デッドロックの例




（凡例）——→　：排他を要求して，資源を占有したことを示します。
-----→　：排他を要求して，ほかのUAPが資源を解放するのを待っていることを示します。
1. ～4.　：排他を要求した順番を示します。

## (a)　デッドロック時の OpenTP1 の処置

デッドロックが起こった場合，OpenTP1 は UAP の排他待ち優先順位に従って，優先順位の低い UAP プロセスからの排他要求をエラーリターンさせます。UAP の排他待ち優先順位は，**ユーザサービス定義の** deadlock_priority オペランドに指定します。

デッドロックによって，資源を確保しようとした関数がエラーリターンした場合，UAP では次のように対処してください。

- SUP，SPP のデッドロック時の処置

  SUP，または SPP の処理でデッドロックが起こった場合は，dc_trn_unchained_rollback 関数，dc_trn_chained_rollback 関数などロールバックする関数で，トランザクションをロールバックさせてください。デッドロックでロールバックした SUP，または SPP は再実行（リトライ）しません。もう一度該当するサービスをクライアント UAP から要求し直してください。

- MHP のデッドロック時の処置

  MHP の処理でデッドロックが起こった場合は，dc_mcf_rollback 関数でロールバックしてください。再実行（リトライ）するかどうかは，dc_mcf_rollback 関数の引数に指定します。

## (b)　デッドロック情報，タイムアウト情報の出力

デッドロックが起こった場合，デッドロックの原因となった UAP の詳細情報を，ロックサービスがあるノードのディレクトリに出力できます。この情報を**デッドロック情報**といいます。

資源の解放を待っている UAP が，ロックサービス定義の lck_wait_timeout オペランドに指定した時間を超えた場合，UAP から呼び出した関数はエラーリターンします。このとき，確保しようとした資源に関する詳細情報を，ロックサービスがあるノードのディレクトリに出力できます。この情報を**タイムアウト情報**といいます。

デッドロック情報，タイムアウト情報を出力するかどうかは，ロックサービス定義の lck_deadlock_info オペランドに指定します。

デッドロック情報，タイムアウト情報の出力形式については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

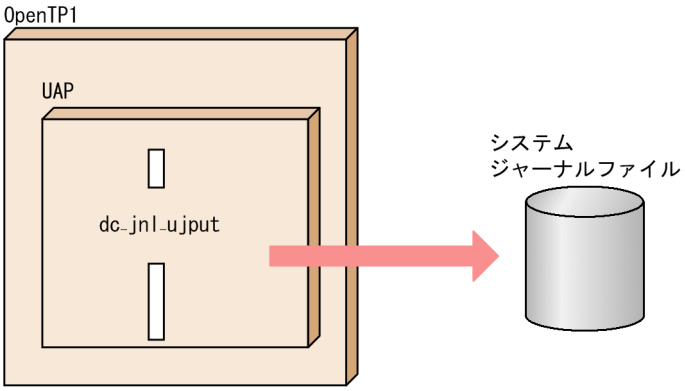
取得したデッドロック情報，タイムアウト情報は削除できます。情報を削除する方法を次に示します。

- **コマンドで削除する方法**
  lckrminf コマンドを実行します。
- **OpenTP1 の開始時に，前回までのオンラインで作成した情報を削除する方法**
  ロックサービス定義の lck_deadlock_info_remove_level オペランドに，削除する条件を指定しておきます。

## 3.9.2　ユーザジャーナルの取得

UAP の中で UAP 履歴情報取得機能を使用して任意の情報をシステムジャーナルファイルに取得できます。**ユーザジャーナル（UJ）**の取得方法を次の図に示します。

図 3-78　ユーザジャーナルの取得方法




UJ の取得方法については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 3.9.3　ジャーナル維持機能

**ジャーナル維持機能**とは，アンロードジャーナルファイルにコピーしたジャーナルデータを加工して利用するための機能です。ジャーナル維持機能を使って，複数のアンロードジャーナルファイルをマージしたりします。

ジャーナル維持機能には，**ジャーナルの複写**，**ファイル回復用ジャーナルの集積**，**ジャーナルの編集**，**アンロードジャーナルファイルのレコード出力**の四つがあります。

### (1)　ジャーナルの複写（jnlcopy コマンド）

ジャーナルを複写する機能には，**レコードを複写する機能**と**マージする機能**があります。

レコードを複写するときは，アンロードジャーナルファイルの情報を，範囲を区切って標準出力に出力できます。jnlcopy コマンドの引数に次のように範囲を指定して，レコード単位で出力します。

- ジャーナルを取得した日時

  ジャーナル種別（PJ，HJ，BJ，TJ，DJ，FJ，CJ，MJ，IJ，GJ，OJ，AJ，SJ，UJ）を jnlcopy コマンドの-j オプションに指定して，ジャーナル種別ごとのアンロードジャーナルファイルを作成することもできます。

複数のアンロードジャーナルファイルを一つのファイルにマージして出力することもできます。マージするアンロードジャーナルファイルは，1 回のオンライン中に出力した連続する世代番号を持つファイルに限ります。

## (2) ファイル回復用ジャーナルの集積（jnlcolc コマンド）

ファイル回復用ジャーナルを集積するときは，アンロードジャーナルファイルから DAM FRC，または TAM FRC に必要なデータだけを，標準出力に出力します。複数のアンロードジャーナルファイルを一度に集積して，ファイル回復時の入力用ファイルを作成できます。このファイル回復時の入力用ファイルを，**集積ジャーナルファイル**といいます。

障害が発生した DAM ファイルや TAM ファイルは，バックアップファイルとアンロードジャーナルファイルを基に回復します。このとき，OpenTP1 はアンロードジャーナルファイルを集積して，回復処理をします。回復処理をするとき，ユーザが作成した最新の集積ジャーナルファイルがあれば，これを利用して回復時間を短縮できます。

アンロードジャーナルファイル内の未決着トランザクションに関する回復用の履歴情報を，次回のジャーナル集積時に引き継ぐため，集積ジャーナルファイルとは別のファイルに格納します。このファイルを**引き継ぎファイル**といいます。次回のジャーナル集積時には，引き継ぎファイルとアンロードジャーナルファイルから，ファイル回復に必要なデータを集積します。

## (3) ジャーナルの編集（jnledit コマンド）

ジャーナルを編集するときは，アンロードジャーナルファイルごとにファイルの内容をリスト形式にして標準出力に出力します。また，指定したファイルの属性に関する情報も出力できます。ジャーナルレコード，またはジャーナルブロック単位に範囲を限定することで，全ジャーナルのうち必要なジャーナルだけを出力できます。

ジャーナルを編集すると，ジャーナルレコードを 16 進数，または 16 進数と文字の形式で確認できます。

アンロードジャーナルファイル内のユーザジャーナルのレコード（UJ レコード）を，編集しないでそのまま標準出力に出力できます。

ジャーナルを編集した出力形式については，マニュアル「OpenTP1 運用と操作」を参照してください。

### (4) アンロードジャーナルファイルのレコード出力（jnlrput コマンド）

アンロードジャーナルファイルのレコードは，アンロードジャーナルファイル内のユーザジャーナルレコード情報，またはトランザクションブランチの CPU 使用時間情報を編集しないで，そのまま標準出力に出力できます。

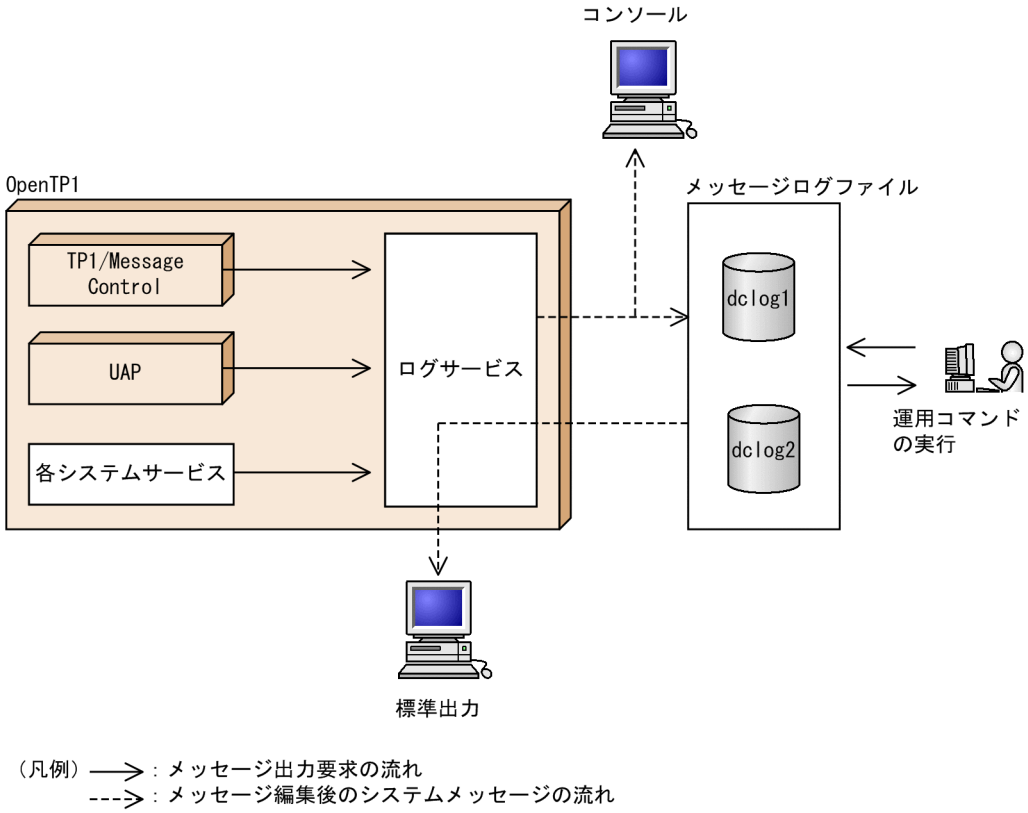
出力するとき，出力範囲，ジャーナル種別，およびジャーナル取得モードなどを指定することで，必要な情報を選択できます。

## 3.9.4 メッセージログの操作

OpenTP1 では，システムメッセージを編集し，**メッセージログ**をファイルに取得して，オンラインシステムの監視ができます。メッセージログは，ログサービスで管理します。

メッセージログの取得方法を次の図に示します。

図 3-79 メッセージログの取得方法




メッセージログの運用方法については，マニュアル「OpenTP1 運用と操作」を参照してください。

### (1) システムメッセージを格納するファイル

ログサービスは，各システムサービス，MCF，および UAP からのメッセージ出力要求を受け付け，メッセージを編集して**メッセージログファイル**に出力します。メッセージログファイルには，dclog1 と dclog2

の二つのファイルがあります。二つのファイルは，ラウンドロビン方式で使用され，1 世代前のメッセージ情報は保証されます。また，ログサービスからメッセージログファイルに出力されるメッセージを，同時にコンソールへ出力できます。

メッセージログファイルに出力されたシステムメッセージは，logcat コマンドで**標準出力**へ出力されます。出力するとき，dclog1 と dclog2 の最終更新時刻を比較して古い情報から順に出力します。出力するときに付ける情報を logcat コマンドのオプションで選択できます。

OpenTP1 のシステムメッセージは，重要なメッセージ（エラーメッセージ，警告メッセージ）に限り，**syslog ファイル**にも出力されます。さらに，OpenTP1 のシステムメッセージすべてを強制的に syslog ファイルに出力することもできます。ログサービス定義の log_syslog_××××オペランドを指定すると，メッセージすべてを syslog ファイルへ出力できます。

### (2) メッセージ通番

付加情報として**メッセージ通番**が取得できます。メッセージ通番は，メッセージログ全体の通番です。障害などでメッセージが欠落した場合でも，メッセージ通番を基に，メッセージに抜けがあったことを検知できます。

### (3) メッセージログの抑止

排他エラーが発生すると，OpenTP1 はメッセージログを出力します。エラーリターンした UAP のリトライを繰り返していると，メッセージログ量が多くなってしまいます。このため，DAM ファイルの場合，排他によるエラーリターンのメッセージを出力するかどうかを，DAM サービス定義の dam_message_level オペランドで指定できます。指定しなければ，メッセージを出力しなくて済みます。

### (4) 出力するメッセージログの言語種別の指定

メッセージログを，英語で出力するか日本語で出力するかを選べます。システム共通定義の putenv 形式で LANG 変数に，出力したい言語種別を指定します。指定を省略すると，メッセージログは英語で出力されます。

## 3.9.5 メッセージログの通知

OpenTP1 のメッセージログを，システム内に専用に作成するアプリケーションプログラムへ通知できます。通知を受信したアプリケーションプログラムは，他社のネットワーク管理システムへ OpenTP1 の状態を知らせることができます。

メッセージログを通知する場合は，OpenTP1 のログサービス定義の log_notify_out オペランドに Y を指定しておいてください。

## (1) メッセージログの通知を受信できるアプリケーションプログラム

メッセージログの通知を受信できるのは，**受信用に作成したアプリケーションプログラム**だけです。OpenTP1 の UAP（SUP，SPP，MHP）では，メッセージログ通知を受信できません。

通知を受信するアプリケーションプログラムには，OpenTP1 ホームディレクトリを示す環境変数 DCDIR を設定しておいてください。この値は，メッセージログを通知する OpenTP1 と同じ値にしてください。

OpenTP1 のオンライン業務が開始以降のすべてのメッセージログを取得する場合，通知を受信するアプリケーションプログラムは OpenTP1 よりも先に開始しておいてください。

## (2) メッセージログの通知の受信手順

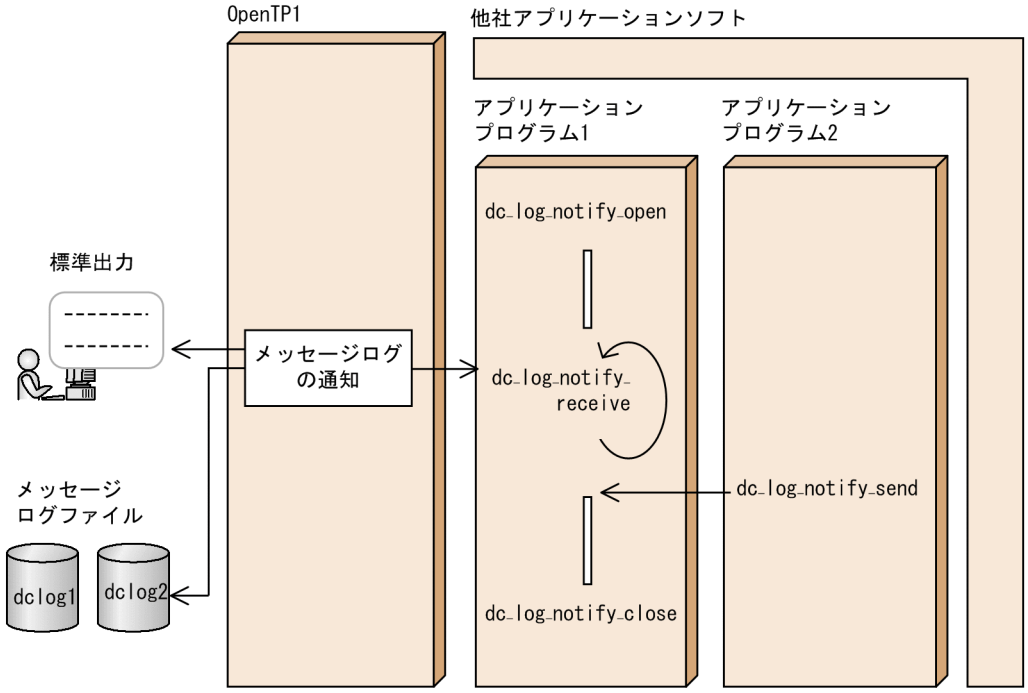
通知を受信するアプリケーションプログラムは，受信の開始を dc_log_notify_open 関数で宣言します。そして，dc_log_notify_receive 関数でメッセージログを受信します。dc_log_notify_receive 関数で受信できるメッセージログは一つです。複数のメッセージログを受信するには，dc_log_notify_receive 関数を繰り返し呼び出します。

メッセージログの通知の受信を終了するときは，dc_log_notify_close 関数を呼び出します。dc_log_notify_close 関数を呼び出したあとで dc_log_notify_open 関数を呼び出せば，再びメッセージログの通知を受信できます。

通知を受信するアプリケーションプログラムは，OpenTP1 が終了したあとでも dc_log_notify_close 関数を呼び出すまで待ち続けます。アプリケーションプログラムに受信処理の終了を知らせる場合には，ほかのアプリケーションプログラムから dc_log_notify_send 関数でデータを送信します。受信処理の終了を知らせるアプリケーションプログラムでは，dc_log_notify_send 関数を呼び出す前に dc_log_notify_open 関数を呼び出せません。

メッセージログの通知の受信を次の図に示します。

図 3-80　メッセージログの通知の受信

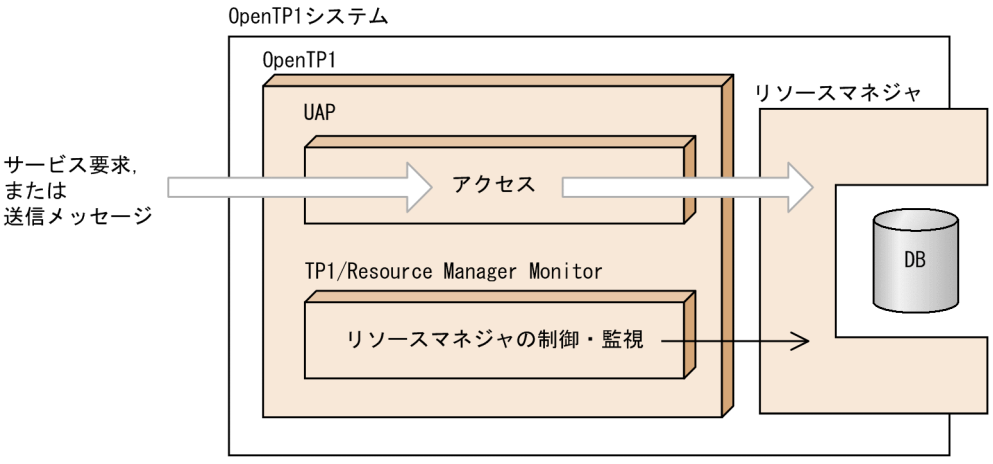



## 3.9.6　OpenTP1 提供以外のリソースマネジャの制御（TP1/Resource Manager Monitor）

TP1/Resource Manager Monitor を使うと，リソースマネジャの開始と終了を制御できます。

リソースマネジャ制御の概要を次の図に示します。

図 3-81　リソースマネジャ制御の概要




### (1)　リソースマネジャを監視する準備

監視する対象のリソースマネジャは，RMM サービス定義の rmm_check_services オペランドに指定しておきます。監視する対象のリソースマネジャへの操作は，ユーザがコマンド（シェルファイル）で作りま

す。そして，作ったコマンドのファイル名を監視対象 RM 定義に指定します。監視対象 RM 定義には，次に示すコマンドを指定します。

- 監視対象 RM 開始コマンド（rmm_start_command オペランド）
- 監視対象 RM 終了コマンド（rmm_stop_command オペランド）
- 監視対象 RM 強制停止コマンド（rmm_abort_command オペランド）
- 監視対象プロセス ID 取得コマンド（rmm_get_pid_command オペランド）

上記のコマンドは，サンプルが提供されています。サンプルを業務に合わせて修正すると，コマンドを最初から作成する手間が省けます。サンプルは，$DCDIR/etc/RMmonitor/ ディレクトリの下にあります。

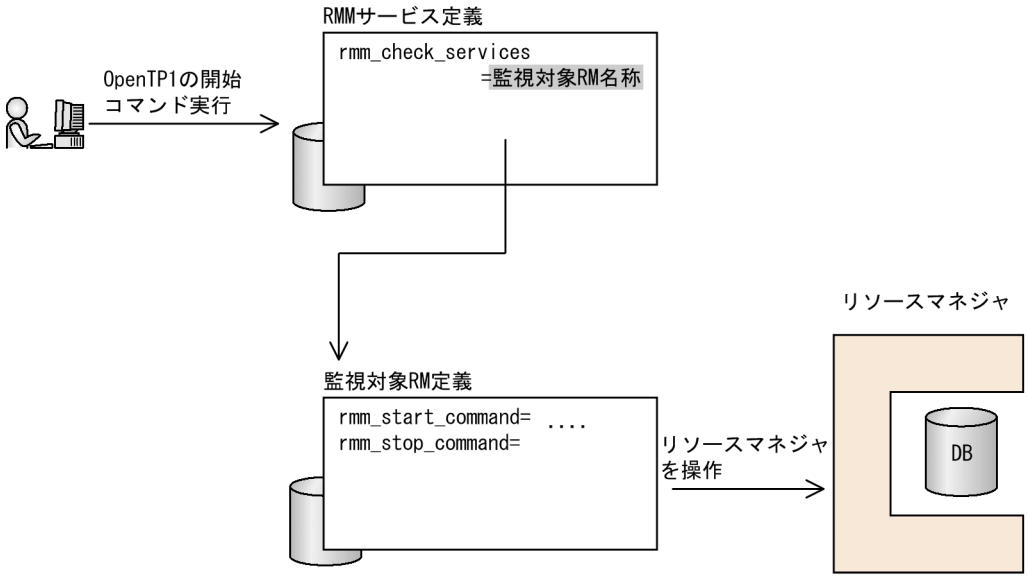
コマンドの作り方については，マニュアル「OpenTP1 運用と操作」を参照してください。定義の方法については，マニュアル「OpenTP1 システム定義」を参照してください。

### (2) 監視を開始する時期

リソースマネジャを監視する定義を作成しておくと，OpenTP1 を開始するのと一緒にリソースマネジャへの操作が自動的に開始されます。

OpenTP1 の開始と参照する定義の関係を次の図に示します。

図 3-82 OpenTP1 の開始と参照する定義の関係




## 3.9.7 稼働統計情報

稼働統計情報は，OpenTP1 の稼働状況を表示します。

## (1) 稼働統計情報の出力（jnlstts コマンド，jnlmcst コマンド）

稼働統計情報を出力するときは，アンロードジャーナルファイルから統計用ジャーナルを抽出し，稼働状況を把握できる形式にデータを加工して，リスト形式で標準出力に出力します。稼働統計情報を出力することで，OpenTP1 の稼働状況がわかります。

稼働統計情報の出力には，システムサービスと UAP の稼働状況に関する**システム統計情報の出力**，トランザクション処理に関する**トランザクション統計情報の出力**（以上，**jnlstts コマンド**），およびメッセージの送受信に関する **MCF 稼働統計情報の出力**（**jnlmcst コマンド**）があります。

稼働統計情報の出力項目については，マニュアル「OpenTP1 運用と操作」を参照してください。

## (2) MCF 稼働統計情報

MCF の稼働統計情報を共用メモリに取得し，統計データとして UNIX ファイルに出力できます。また，出力された統計データを編集して標準出力へ出力できます。

**MCF 稼働統計情報**を使用することによって，MCF の稼働状況を把握できます。

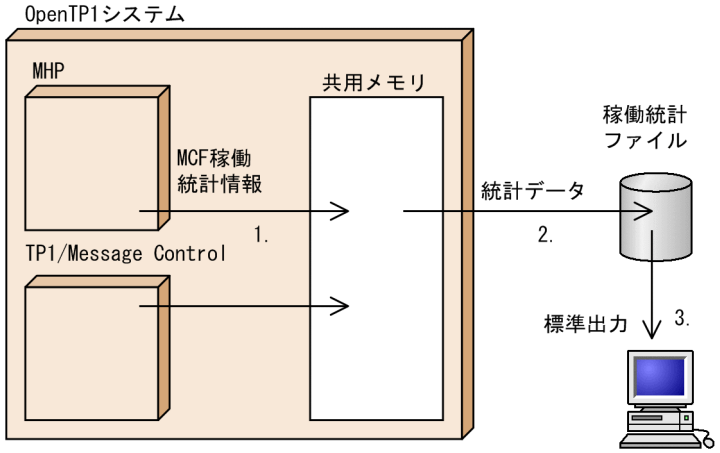
なお，ジャーナルファイルを基に MCF の稼働統計を取得する jnlmcst コマンドと MCF 稼働統計情報の機能とは，それぞれ独立した機能です。jnlmcst コマンドで得られる情報は，指定時間間隔内で処理した件数などです。MCF 稼働統計情報で得られる情報は，処理が待たされている件数などです。

MCF 稼働統計情報の機能は，次の機能から構成されます。番号は下の図中の番号と対応しています。

1. 稼働統計情報取得機能（MCF マネジャ定義の mcfmcomn 定義コマンドの-w オプション stats オペランドに yes を指定）
2. 稼働統計情報出力機能（mcfstats コマンド）
3. 稼働統計情報編集機能（mcfreport コマンド）

MCF 稼働統計情報の機能の概要を次の図に示します。

図 3-83　MCF 稼働統計情報の機能の概要



### (3) システム統計情報のリアルタイム出力（dcreport コマンド）

dcreport コマンドを使用すると，システム統計情報を標準出力にリアルタイムに出力したり，メッセージログに出力したりできます。標準出力に出力するシステム統計情報は CSV 形式にすることもできます。このコマンドは，システム共通定義の statistics オペランドに Y を指定した場合，および dcstats コマンドでシステム統計情報のジャーナル出力要求を指定した場合に使用できます。

dcreport コマンドを使用してシステム統計情報を取得すると，サービスの実行状況，リソースの負荷状況，障害状況などをリアルタイムに参照できます。

## 3.9.8 リアルタイム統計情報サービス

リアルタイム統計情報サービスは，システム全体，サーバおよびサービス単位に統計情報を取得します。このリアルタイム統計情報を標準出力やログファイルに出力すると，OpenTP1 システムの稼働状況をリアルタイムに把握でき，システムの運用管理や障害復旧を迅速に行えます。

ここでは，リアルタイム統計情報サービスの概要について説明します。リアルタイム統計情報サービスを使用する運用，およびリアルタイム統計情報の取得項目については，マニュアル「OpenTP1 運用と操作」を参照してください。

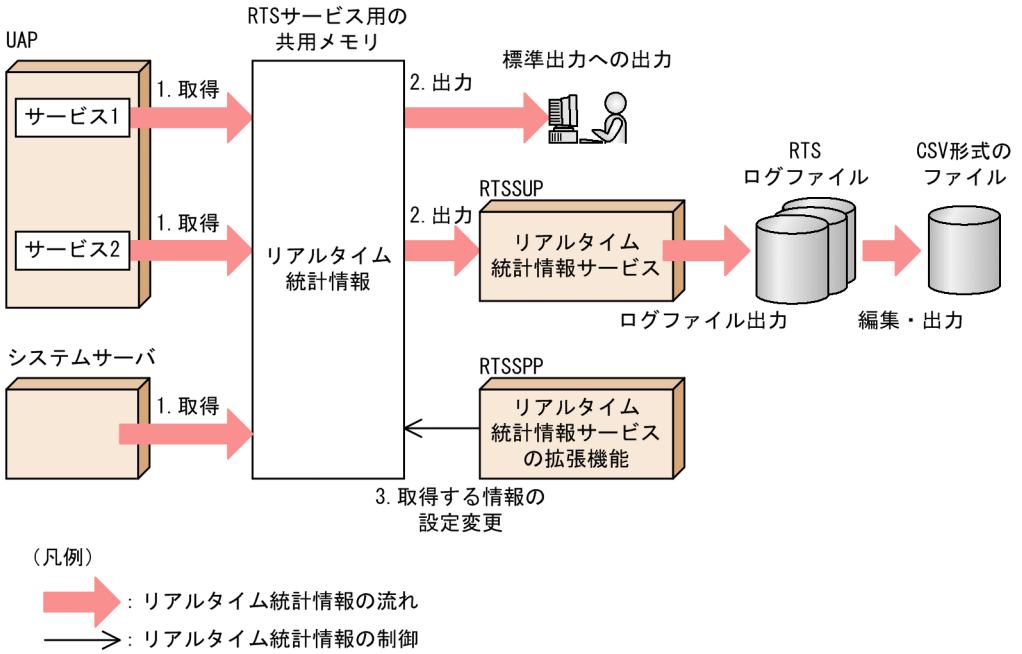
リアルタイム統計情報サービスでは，次に示す UAP を使用します。

RTSSUP
: リアルタイム統計情報サービスを使用する場合に起動させる UAP です。リアルタイム統計情報を取得，出力できます。

RTSSPP
: リアルタイム統計情報サービスの拡張機能を使用する場合に起動させる UAP です。拡張機能では，リアルタイム統計情報サービスのオンライン中に，取得する情報の設定を変更できます。

リアルタイム統計情報サービスの概要を次の図に示します。

図 3-84　リアルタイム統計情報サービスの概要




リアルタイム統計情報サービスの機能を次に示します。番号は図中の番号と対応しています。

1. 定義した項目ごとに，システム全体，サーバおよびサービス単位でリアルタイム統計情報を取得します。

   リアルタイム統計情報を，RTS サービス用の共用メモリに取得します。リアルタイム統計情報サービスで使用する共用メモリについては，「7.2.2(4)　リアルタイム統計情報サービスで使用する共用メモリ」を参照してください。

   UAP 内の任意区間の実行時間を，リアルタイム統計情報として取得することもできます。UAP 内の任意区間でのリアルタイム統計情報の取得については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

2. RTS サービス用の共用メモリに取得したリアルタイム統計情報を出力します。

   - 標準出力にリアルタイムに出力できます。
   - RTS ログファイルに出力できます。
   - RTS ログファイルに出力した統計情報を CSV 形式で編集・出力できます。

3. リアルタイム統計情報サービスのオンライン中に，取得する情報の設定を変更できます。

## 3.9.9　OpenTP1 の状態確認機能

dcstatus コマンドを使用して，任意のタイミングでオンライン，オフラインなどの OpenTP1 の状態を確認できます。

dcstatus コマンドおよび dcstatus コマンドを使用した運用例については，マニュアル「OpenTP1 運用と操作」を参照してください。

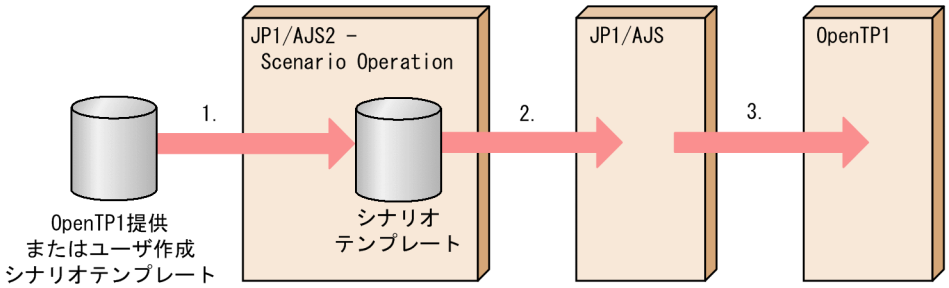
## 3.10　シナリオテンプレートを利用したシステムの運用

OpenTP1 の運用手順を，JP1/AJS2 - Scenario Operation が管理するシナリオテンプレートとしてあらかじめ定義できます。シナリオテンプレートを利用すると，システムを自動的に運用できます。シナリオテンプレートは，JP1/AJS2 - Scenario Operation が JP1/AJS - Manager を介して実行します。
OpenTP1 では，基本的な運用手順を定義したシナリオテンプレートを提供しています。OpenTP1 が提供するシナリオテンプレートを使用すると，より簡単にシステムの運用を自動化できます。

JP1/AJS2 - Scenario Operation の詳細説明については，マニュアル「JP1/Automatic Job Management System 2 - Scenario Operation」を参照してください。

シナリオテンプレートを利用したシステムの運用を次の図に示します。

図 3-85　シナリオテンプレートを利用したシステムの運用




注

シナリオテンプレートの入力シナリオ変数を変更することによって，異なる OpenTP1 環境にシナリオを実行できます。

（説明）

1. OpenTP1 が提供またはユーザが作成したシナリオテンプレートを JP1/AJS2 - Scenario Operation に登録します。
2. JP1/AJS2 - Scenario Operation が，JP1/AJS にシナリオを登録します。
3. JP1/AJS が，OpenTP1 にシナリオを実行させます。

OpenTP1 がシナリオテンプレートを提供しているシナリオを次に示します。

- スケールアウト

  新しい OpenTP1 ノードを構築して，OpenTP1 システムのドメイン構成に新しいノードを追加します。
- スケールイン

  業務単位またはノード単位で，負荷レベルの低いノードのリソースを解放して，他システムに割り当てます。
- ローリングアップデート

  システムを停止させることなく OS や UAP のセキュリティ対策パッチを適用します。

シナリオテンプレートの詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

## 3.11　監査ログによるシステムの監視

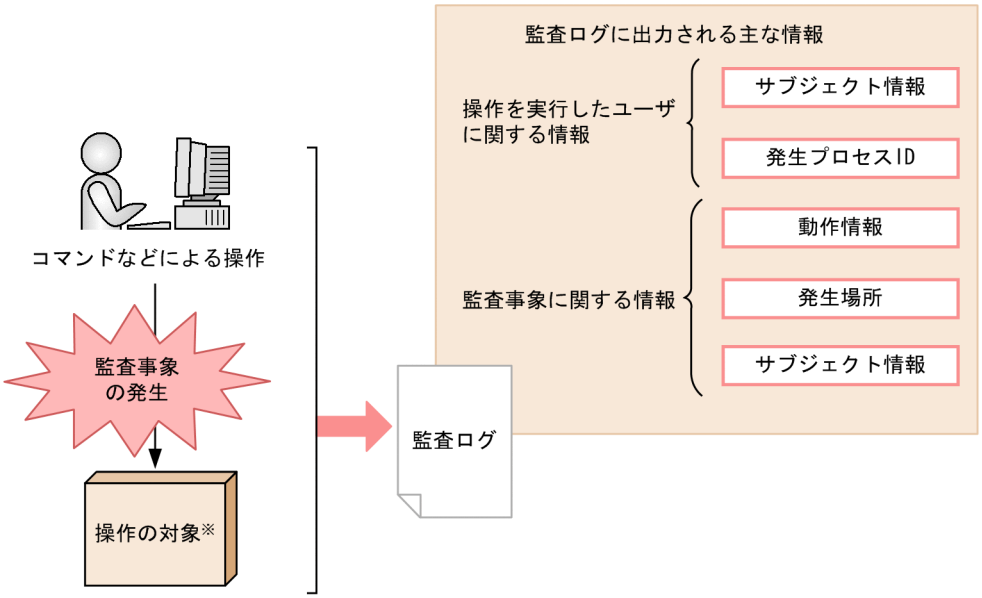
**監査ログ**とは，システム構築者，運用者，および使用者が OpenTP1 のプログラムに対して実行した操作，およびその操作に伴うプログラムの動作の履歴が出力されるファイルです。監査者が監査ログを調査することで，「いつ」，「だれが」，「何をしたか」を知ることができます。そのため，システムの使用状況，不正アクセスなどを監査する資料として使用できます。

監査ログには，コマンドなどによる操作を実行したユーザに関する情報，その操作に伴う処理が成功したか失敗したかなどの監査事象に関する情報，操作や処理の対象に関する情報などが出力されます。これらの情報は，システムの監視に役立ちます。

なお，JP1/NETM/Audit と連携すると，監査ログを自動で収集したり，一括管理をしたりすることができるようになります。

監査ログが出力される流れと主な出力項目を次の図に示します。

### 図 3-86　監査ログが出力される流れと主な出力項目




注※　コマンドなどによる操作の対象を示します。
例えば，サーバを起動するためのコマンドを実行した場合，起動する対象となるサーバを示します。

監査ログが出力される契機になるのは，コマンドの実行など，OpenTP1 のプログラムに対する操作です。この操作は，システム管理者，システム運用者など，その作業に応じたユーザが実行します。また，監査ログは OpenTP1 の各プロセスで**監査事象**が発生したタイミングで出力されます。監査事象とは，OpenTP1 のプログラムに対して実行した操作，およびその操作に伴うプログラム処理のうち，システムの構築や運用，使用が妥当かどうかを記録する必要がある事象のことです。OpenTP1 では，監査事象を次の表に示すとおり分類して定義します。

表 3-24　監査事象の定義

| 監査事象種別 | 内容 | 動作情報 | |
|---|---|---|---|
| StartStop | ソフトウェアの起動と終了を示す事象です。次の操作が該当します。<br>• OpenTP1 の起動・停止<br>• ユーザサーバの起動・停止 | Start | 開始，起動 |
| | | Stop | 終了，停止 |
| Authentication | クライアントユーザが認証を試みて成功・失敗したことを示す事象です。 | Login | ログイン |
| | | Logout | ログアウト |
| | | Logon | ログオン |
| | | Logoff | ログオフ |
| | | Disable | アカウントの無効化 |
| AccessControl | 管理者やユーザが管理リソースへのアクセスを試みて成功・失敗したことを示す事象です。 | Enforce | 実施 |
| ConfigurationAccess | 管理者やユーザが設定情報に対し変更などの操作を実施して成功・失敗したことを示す事象です。 | Refer | 参照 |
| | | Add | 追加 |
| | | Update | 更新 |
| | | Delete | 削除 |
| Failure | ソフトウェアの異常を示す事象です。 | Occur | 発生 |
| LinkStatus | 機器間のリンク状態を示す事象です。 | Up | リンク活性 |
| | | Down | リンク非活性 |
| ExternalService | ソフトウェアと外部サービスとの通信結果を示す事象です。 | Request | 要求 |
| | | Response | 応答 |
| | | Send | 発信 |
| | | Receive | 受信 |
| ContentAccess | 重要なデータへのアクセスを試みて成功・失敗したことを示す事象です。 | Refer | 参照 |
| | | Add | 追加 |
| | | Update | 更新 |
| | | Delete | 削除 |
| Maintenance | 管理者や保守員が保守操作を実行して成功・失敗したことを示す事象です。 | Install | インストール |
| | | Uninstall | アンインストール |

| 監査事象種別 | 内容 | 動作情報 | |
|---|---|---|---|
| Maintenance | 管理者や保守員が保守操作を実行して成功・失敗したことを示す事象です。 | Update | 更新，アップデート |
| | | Backup | バックアップ |
| | | Maintain | 保守作業 |
| AnomalyEvent | 異常な通信の発生を示す事象です。 | Occur | 発生 |
| ManagementAction | プログラムの重要なアクションの実行を示す事象，またはほかの監査カテゴリを契機として実行するアクションを示す事象です。 | Invoke | （管理者の）呼び出し |
| | | Notify | （管理者への）通知 |

監査事象は，監査イベントごとに定義されています。監査イベントの一覧の詳細については，マニュアル「OpenTP1 運用と操作」の監査イベントの出力情報の説明を参照してください。

また，OpenTP1 では，UAP から任意に監査ログを出力する API を提供します（dc_log_audit_print 関数）。この API を使用すると，監査イベントのタイミング以外に，UAP の操作の実行，および UAP による処理のタイミングで監査ログを出力できます。

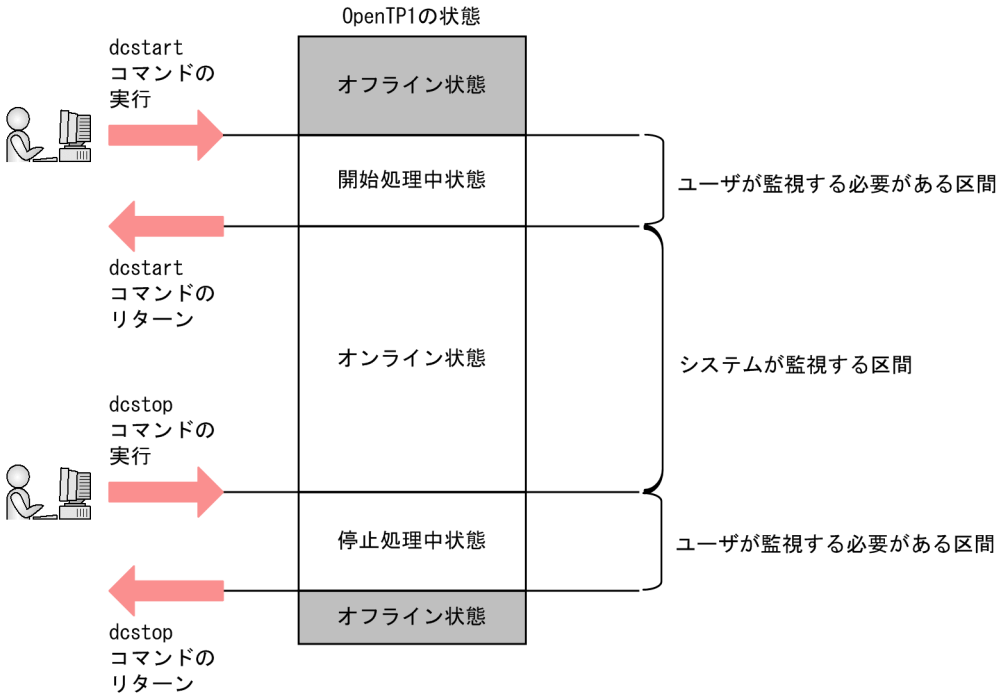
UAP から任意に監査ログを出力する実装方法については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## 3.12　OpenTP1 の監視

ハードウェアの障害に起因する一時的なエラーなどによって，ユーザサーバなどを管理するプロセスサービスが無応答になると，ユーザサーバの起動や OpenTP1 の停止ができなくなります。この状態を無応答状態と呼びます。この場合，プロセスサービスを強制停止し，OpenTP1 を再開始（リラン）する必要があります。

そのため，OpenTP1 システムの運用として，プロセスサービスの稼働状態の監視が必要です。監視区間は，ユーザが監視する必要がある区間とシステムが監視する区間に分けられます。プロセスサービスの監視区間を次の図に示します。

図 3-87　プロセスサービスの監視区間




ここでは，システムが監視する区間で OpenTP1（プロセスサービス）を監視する機能について説明します。

### 3.12.1　HA モニタを使用した OpenTP1 の監視

二重化した OpenTP1 システムでは，HA モニタでプロセスサービスを監視します。HA モニタのサーバモードでのプロセスサービスの監視では，無応答状態を検知すると，系切り替え機能を使用して無応答状態から復帰させます。系切り替え機能については，「6.1　系切り替え機能」を参照してください。

## 3.12.2　OpenTP1 監視機能を使用した OpenTP1 の監視

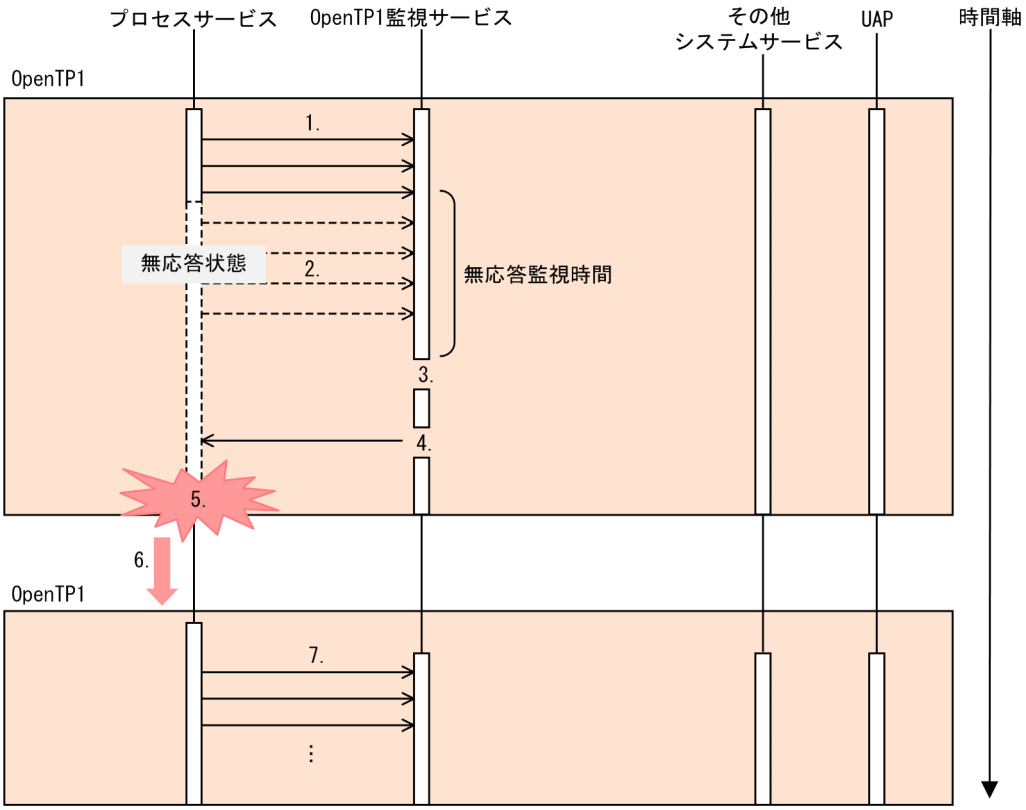
OpenTP1 監視機能とは，OpenTP1 監視サービスによってプロセスサービスを監視し，プロセスサービスを無応答状態から自動で復帰させる機能です。HA モニタを使用しない OpenTP1 システムでも，無応答状態から復帰できます。

OpenTP1 監視機能は，OpenTP1 監視サービスの正常稼働が前提となります。そのため，OS パニックや CPU の高負荷状態など，OpenTP1 が稼働するマシン全体に影響を及ぼす障害に起因するプロセスサービスの無応答については，ユーザの運用で対処する必要があります。

なお，OpenTP1 監視機能は，HA モニタと併用できます。HA モニタ（サーバモード）を使用した系切り替え構成の場合，プロセスサービスの無応答に加え，マシン全体の障害についても監視および系切り替えによって自動復帰できます。そのため，HA モニタを使用した OpenTP1 の監視を優先してください。

OpenTP1 監視機能の動作（プロセスサービスが無応答状態になった場合）を次の図に示します。

図 3-88　OpenTP1 監視機能の動作（プロセスサービスが無応答状態になった場合）




（説明）

1. プロセスサービスは，OpenTP1 監視サービスへ定期的に稼働報告をします。
2. ハードウェアの障害に起因する一時的なエラーなどによって，プロセスサービスは OpenTP1 監視サービスへの稼働報告ができません。
3. OpenTP1 監視サービスは，一定時間（これを無応答監視時間と呼びます）内にプロセスサービスからの稼働報告がないため，無応答状態と判断します。

4. OpenTP1 監視サービスは，プロセスサービスを強制停止します。
5. 再開始（リラン）するため，OpenTP1 を強制停止します。
6. システム環境定義（mode_conf オペランド）の指定に従い，手動または自動で OpenTP1 を再開始（リラン）します。
7. プロセスサービスは，OpenTP1 監視サービスへの定期的な稼働報告を再開します。

OpenTP1 監視機能の運用設計については，マニュアル「OpenTP1 運用と操作」の OpenTP1 の監視に関する運用の説明を参照してください。

# 4 OpenTP1 ファイルシステムとファイル

この章では，OpenTP1 ファイルシステムの概要と，OpenTP1 ファイル（システムで使うファイル，キューを格納するファイル，ユーザデータを格納するファイル），および IST サービスについて説明します。

# 4.1　OpenTP1 ファイルシステムの概要

OpenTP1 は，ファイルの使用効率がよい **OS が提供するファイルシステム**と，信頼性が高い **OpenTP1 が提供するファイルシステム**を使用しています。OpenTP1 で管理するファイルシステムを，**OpenTP1 ファイルシステム**といいます。

## 4.1.1　ファイルシステム

OS が入出力するディスクは連続領域ごとに分割され，それぞれの領域を**パーティション**と呼びます。それぞれのパーティションを，**OS が提供するファイルシステム**または **OpenTP1 ファイルシステム**に使用できます。

### (1)　OpenTP1 ファイルシステムと OpenTP1 ファイル

**OpenTP1 ファイルシステム**は，OS が提供するファイルシステムとは独立した，OpenTP1 専用のファイルシステムです。キャラクタ型スペシャルファイル，または通常ファイル上に OpenTP1 ファイルシステムを作成します。ユーザデータや OpenTP1 の回復に必要なジャーナルなど，システムの信頼性に関係する重要な情報を格納するファイルは，OpenTP1 ファイルシステム上に作成します。OpenTP1 ファイルシステム上に作成するファイルを，**OpenTP1 ファイル**といいます。

系切り替え機能を使用する場合，OpenTP1 ファイルシステムはキャラクタ型スペシャルファイル上に作成してください。系切り替え機能については，「6.1.3　系切り替えの手順」を参照してください。

OpenTP1 ファイルを次の表に示します。

表 4-1　OpenTP1 ファイルの一覧

| ファイル名 | ファイルの使用方法 |
|---|---|
| ステータスファイル | システムサービスの稼働状態やシステム構成情報を格納して，障害が起こった場合に OpenTP1 を回復するときに使います。 |
| システムジャーナルファイル | トランザクション処理の履歴を格納して，障害が起こった場合に OpenTP1 を回復するときに使います。また，UAP の履歴（ユーザジャーナル）も格納します。 |
| チェックポイントダンプファイル | 回復対象のテーブル情報を格納して，障害が起こった場合に OpenTP1 を回復するときに使います。 |
| アーカイブジャーナルファイル | TP1/Multi を使ったクラスタ/並列システムで，各ノードのジャーナルを集積するために使います。 |
| メッセージキューファイル | メッセージ送受信機能（TP1/Message Control）を使ってメッセージをやり取りするときの，待ち行列となるファイルです。TP1/Message Control をシステムに組み込んでいることが前提です。 |
| ノードリストファイル | ノード自動追加機能を使用する場合，ノードリスト情報を引き継ぐときに，ノードリスト情報を格納し，次回の OpenTP1 起動時に使います。 |

| ファイル名 | ファイルの使用方法 |
|---|---|
| MQA キューファイル | メッセージキューイング機能（TP1/Message Queue）を使う UAP がメッセージを登録したり受け取ったりするときに使用します。TP1/Message Queue をシステムに組み込んでいることが前提です。 |
| DAM ファイル | ユーザファイルとして使います。TP1/FS/Direct Access をシステムに組み込んでいることが前提となります。 |
| TAM ファイル | ユーザファイルとして使います。TP1/FS/Table Access をシステムに組み込んでいることが前提となります。 |

注

OpenTP1 ファイルの運用を支援するため，次に示すファイルも作成できます。

- トランザクションリカバリジャーナルファイル（TRF）
  処理が長くなるトランザクションのジャーナルを，少なくするために使います。
- サーバリカバリジャーナルファイル（SRF）
  再開始（リラン）時の回復処理を短縮するために使います。

OpenTP1 ファイルシステムと OS が提供するファイルシステムの関係を図 4-1 に示します。また，OpenTP1 ファイルシステムと OS が提供するファイルシステムとの違いを表 4-2 に，OpenTP1 ファイルシステムを作成するファイルの選択方法を図 4-2 に示します。

## 図 4-1　OpenTP1 ファイルシステムと OS が提供するファイルシステムの関係

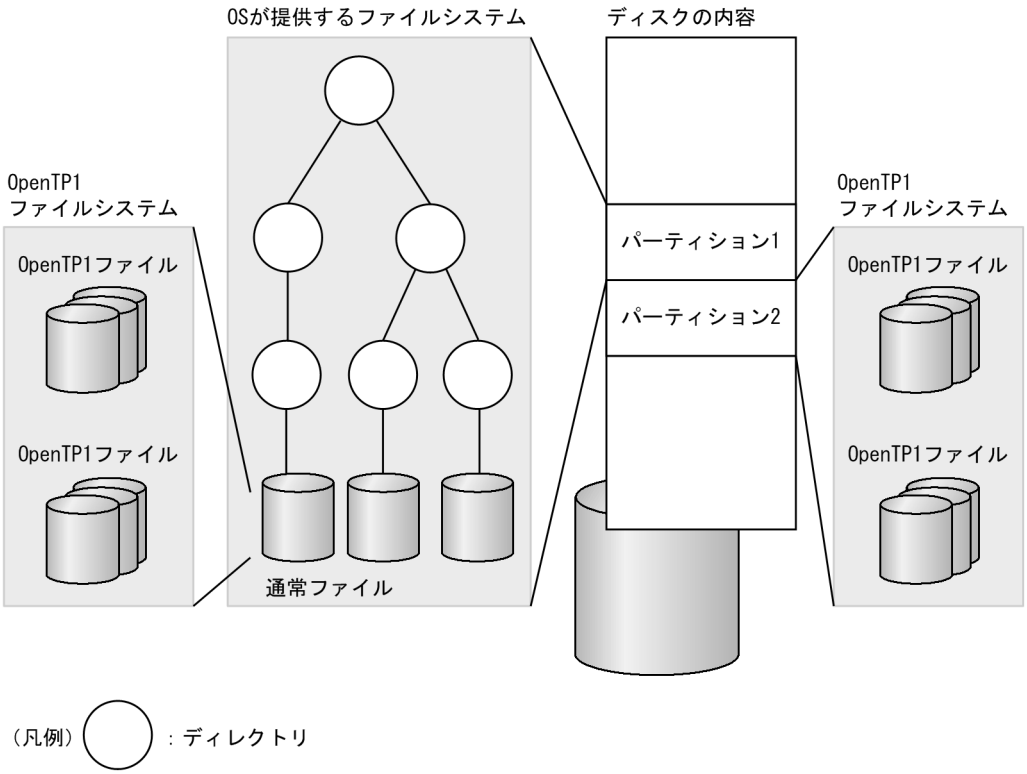
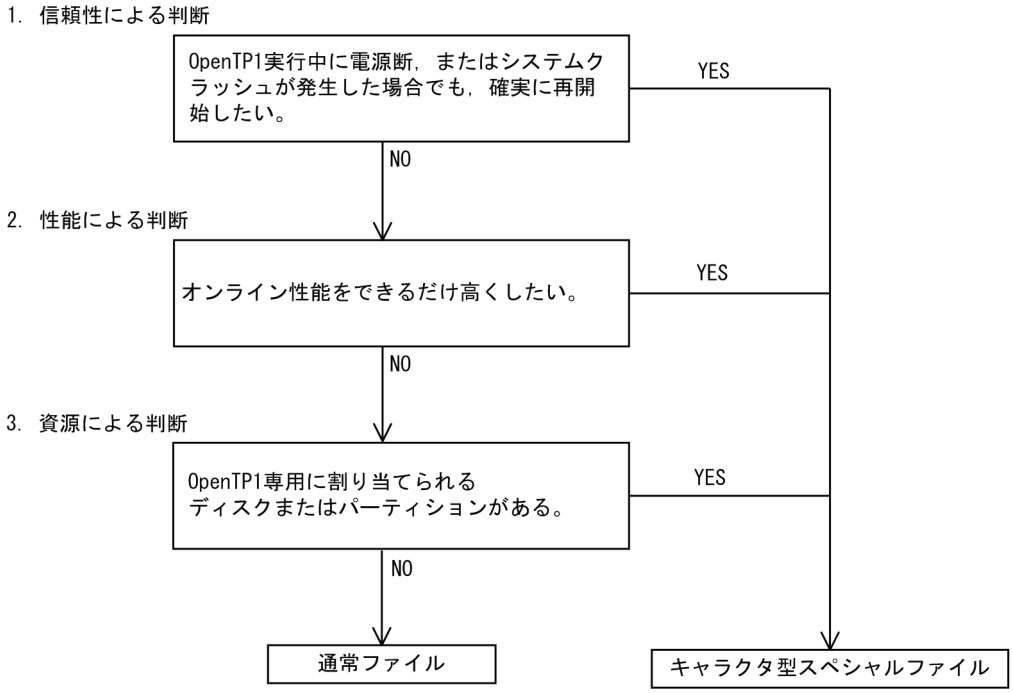


表 4-2　OpenTP1 ファイルシステムと OS が提供するファイルシステムとの違い

| 比較項目 | | OpenTP1 ファイルシステム | | OS が提供するファイルシステム |
|---|---|---|---|---|
| | | キャラクタ型スペシャルファイル上 | 通常ファイル上 | |
| 信頼性 | OpenTP1 が異常終了したとき直前に正常終了した書き込み要求で指定したデータ | 同期 I/O のため，書き込み要求が正常終了したデータは保証される | | データは保証されない |
| | プロセスが異常終了したとき直前に正常終了した書き込み要求で指定したデータ | | | |
| | 電源断などでシステムダウンしたときのディスクファイル | ディスク上の別領域に管理情報を持たないのでファイルの整合性の面で安全性が高い | ディスク上の別領域に管理情報を持つので安全性はキャラクタ型スペシャルファイルよりも低い | ディスク上の別領域に管理情報を持つのでファイルの整合性の面で安全性が低い |
| | ファイル領域の確保 | ファイルシステム作成時に領域を事前に確保するのでオンライン中に容量不足となることはない | | ファイル増加時にブロックを確保するので，書き込みの延長で容量不足となることがある |
| 性能 | ディスク領域の使用効率 | OpenTP1 ファイル中のレコードが書き込まれていない領域も，そのファイルのために確保済みのため，ほかのファイルに割り当てることはできない | | ファイルのサイズが大きくなるときにブロック単位で動的に確保されるので，OpenTP1 ファイルと比べて使用効率が良い |
| | 読み込みの実行時間（関数がリターンするまでの時間） | バッファキャッシュを経由しないので，それによる効果は得られない | 対象データがバッファキャッシュ上でヒットするとディスクから読まないので一般的には速い。ただし，上位でバッファを持って制御するプロセスにとっては二重にバッファリングが行われることになり，逆にオーバヘッドとなるおそれがある | 対象データがバッファキャッシュ上でヒットするとディスクから読まないので，一般的には速い |
| | 書き込みの実行時間（関数がリターンするまでの時間） | 同期 I/O のため，一般的には OS が提供するファイルシステムより遅いが，バッファキャッシュを経由しない分，書き込み処理のオーバヘッドは小さい | 同期 I/O のため，一般的には OS が提供するファイルシステムより遅い | 非同期 I/O のため，一般的には速い |
| | アクセス時間のばらつき | 連続領域が確保されるので，ばらつきは小さい | 必ずしも連続領域とはならないので，ばらつきがある | ばらつきがある |
| 運用 | ファイル容量の見積もり | ファイルの作成時に容量を指定する必要がある | | ファイルの容量を指定する必要はない |

| 比較項目 | | OpenTP1 ファイルシステム | | OS が提供するファイルシステム |
|---|---|---|---|---|
| | | キャラクタ型スペシャルファイル上 | 通常ファイル上 | |
| 運用 | ファイル容量の動的増分 | 増分できない | | 増分できる |
| | ディレクトリによるファイルの分類 | できない | | できる |

図 4-2　OpenTP1 ファイルシステムを作成するファイルの選択方法




## (2)　通常ファイル

**通常ファイル**は，柔軟性があり使用効率がよいという特長があります。このため，OpenTP1 の定義ファイルなどに使用します。

OpenTP1 で使用する通常ファイルを次の表に示します。

表 4-3　OpenTP1 で使用する通常ファイル

| ファイル名 | ファイルの用途 | 備考 |
|---|---|---|
| ユーザプログラムファイル | UAP の実行形式プログラムを格納するファイルです。 | ユーザが作成します。 |
| 各種定義ファイル | OpenTP1 の各種システム定義を格納します。OS のテキストエディタを使用して，テキストファイルとして作成します。 | |
| マップファイル | XMAP3 を使ったクライアントと通信する場合に使用します。物理マップを格納します。 | |
| OpenTP1 プログラムファイル | OpenTP1 のプログラムを格納します。OpenTP1 の実行形式ファイルと UAP の作成に使うファイルがあります。 | インストール時に自動的に作成されます。 |

| ファイル名 | ファイルの用途 | 備考 |
|---|---|---|
| 定義解析用ファイル | OpenTP1 の内部で，定義解析用に使用します。 | インストール時に自動的に作成されます。 |
| メッセージオブジェクトファイル | メッセージテキストを格納します。 | |
| コマンドログファイル | OpenTP1 のコマンドログを格納します。 | |
| メッセージログファイル | OpenTP1 が出力したシステムメッセージを格納します。 | OpenTP1 実行時に作成されます。 |
| MCF トレースファイル | MCF のトレース情報を格納します。 | |
| スケジュールキュー情報ファイル | OpenTP1 の内部で，スケジュールキュー情報を格納します。 | |
| RPC トレースファイル | RPC トレースを格納します。 | |
| トレース情報ダンプファイル | OpenTP1 内部のトレース情報を格納するファイルです。 | |
| 共用メモリダンプファイル | OpenTP1 が出力した共用メモリの内容を格納します。 | |
| 退避コアファイル | 異常終了したプロセスのコアファイルを退避します。 | |
| デッドロック，タイムアウト情報ファイル | デッドロック情報，タイムアウト情報を格納します。 | |
| MCF ダンプファイル | MCF のダンプを格納します。 | |
| 未決着トランザクション情報ファイル | 障害発生時，未決着のトランザクション情報を格納します。 | |
| 不正ジャーナル情報ファイル | ジャーナル読み込み時に検知した不正なジャーナル情報を格納します。 | |
| 入出力キューの内容複写ファイル | 入出力キューの内容複写コマンドを実行したときに入出力キューの内容を格納します。 | |
| 性能検証用トレース情報ファイル | 性能検証用のトレース情報を格納するファイルです。 | |
| XAR 性能検証用トレース情報ファイル | XA リソースサービスを使用したトランザクション連携の各種イベントのトレース情報を格納するファイルです。 | |
| JNL 性能検証用トレース情報ファイル | ジャーナルサービスのトレース情報を格納するファイルです。 | |
| LCK 性能検証用トレース情報ファイル | ロックサービスを使用した排他制御の各種イベントのトレース情報を格納するファイルです。 | |
| MCF 性能検証用トレース情報ファイル | MCF を使用したメッセージ送受信の各種イベントのトレース情報を格納するファイルです。 | |
| TRN イベントトレース情報ファイル | トランザクションブランチで呼び出される XA 関数や，トランザクションサービスの各種イベントのトレース情報を格納するファイルです。 | |
| NAM イベントトレース情報ファイル | ネームサービスで実行される通信処理，キャッシュへのサービス情報の登録，削除などの各種イベントのトレース情報を格納するファイルです。 | |
| プロセスサービスイベントトレース情報ファイル | プロセスサービスのトレース情報を格納するファイルです。 | |

| ファイル名 | ファイルの用途 | 備考 |
|---|---|---|
| FIL イベントトレース情報ファイル | OpenTP1 ファイルへのアクセス要求に対して，システム共通定義の fil_prf_trace_delay_time オペランドの指定値以上の処理時間が掛かった場合に，イベント情報を格納するファイルです。 | OpenTP1 実行時に作成されます。 |
| RTS ログファイル | リアルタイム統計情報を格納するファイルです。 | |
| UAP トレース編集出力ファイル | UAP が異常終了した場合に，UAP トレースを自動的に編集出力して格納したファイルです。 | |
| OpenTP1 デバッグ情報ファイル | UAP が異常終了した場合に，OpenTP1 の情報を格納するファイルです。 | |
| MCF 稼働統計情報ファイル | MCF の稼働統計情報を格納するファイルです。 | |
| マッピングエラー情報ファイル | マッピング時に発生したエラー情報を格納するファイルです。 | |

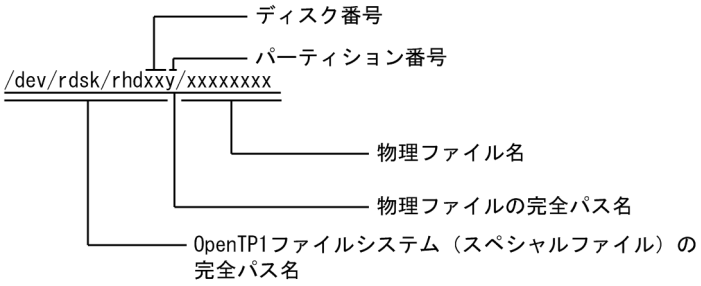
## 4.1.2 OpenTP1 ファイルシステムの作成方法

OpenTP1 管理者は **filmkfs コマンド**を使って，キャラクタ型スペシャルファイルまたは通常ファイル上に OpenTP1 ファイルシステム領域を作成します。

OpenTP1 ファイルシステム領域を作成したあとに，ジャーナルファイルの種類別に**物理ファイル**を割り当てます。物理ファイルを割り当てるコマンドを次に示します。

- ステータスファイル：stsinit コマンド
- システムジャーナルファイル：jnlinit コマンド
- チェックポイントダンプファイル：jnlinit コマンド
- メッセージキューファイル：queinit コマンド
- ノードリストファイル：namnlcre コマンド
- MQA キューファイル：mqainit コマンド
- DAM ファイル：damload コマンド
- TAM ファイル：tamcre コマンド

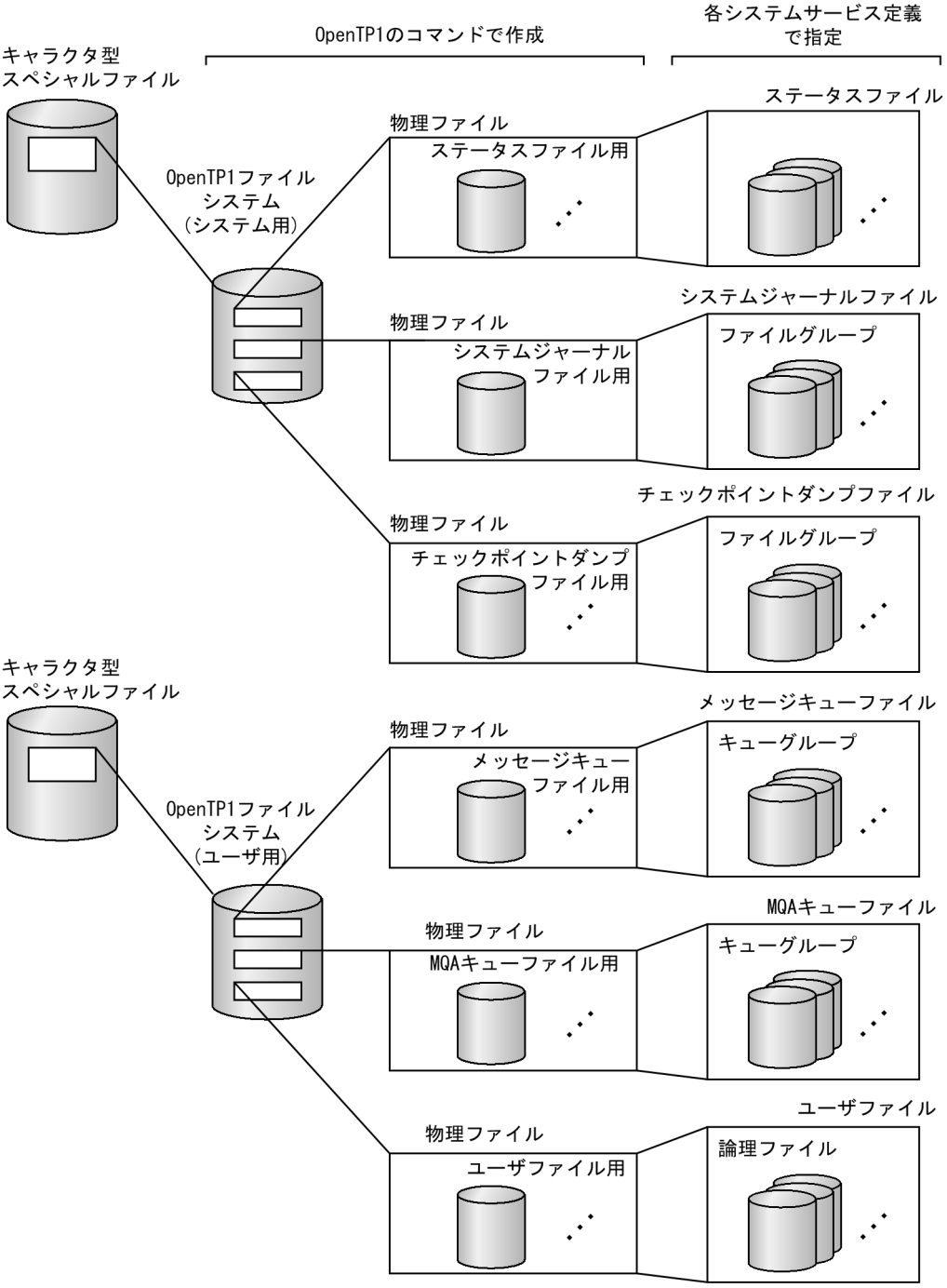
キャラクタ型スペシャルファイル上の OpenTP1 ファイルシステムに作成した物理ファイルの完全パス名の例を次に示します。




次に，OpenTP1 管理者はシステムサービス定義（システムジャーナルサービス定義など）で，物理ファイルが実際に入出力する OpenTP1 ファイル（**論理ファイル**，**ファイルグループ**など）として使えるように設定します。OpenTP1 ファイルの作成手順については，マニュアル「OpenTP1 運用と操作」を参照してください。

OpenTP1 ファイルシステムの構造を次の図に示します。次の図は，キャラクタ型スペシャルファイル上に OpenTP1 ファイルシステムを作成する例です。実際には，システムの信頼性や性能などを考慮して OpenTP1 ファイルシステムを複数作り，OpenTP1 ファイルを分散する必要があります。

図 4-3　キャラクタ型スペシャルファイル上に OpenTP1 ファイルシステムを作成する例

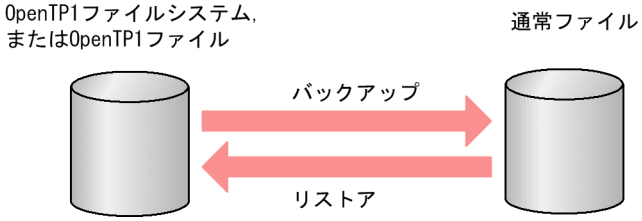



## 4.1.3　OpenTP1 ファイルシステムのバックアップとリストア

OpenTP1 ファイルには，OpenTP1 のプロセスの回復などに使用する重要な情報が入っています。OpenTP1 ファイルを破壊から守るため，定期的（OpenTP1 の開始前など）に filbkup コマンドで OpenTP1 ファイルシステムごと，または OpenTP1 ファイルごとに OpenTP1 ファイルの**バックアップ**を取得しておいてください。バックアップを取得しておけば，OpenTP1 ファイルが破壊された場合でも，filrstr コマンドで OpenTP1 ファイルを**リストア**して回復できます。

OpenTP1 ファイルシステムのバックアップとリストアを次の図に示します。

図 4-4　OpenTP1 ファイルシステムのバックアップとリストア




ユーザファイルを回復する OpenTP1 の機能を次に示します。

- ユーザファイルをバックアップ，リストアするコマンド
  dambkup コマンド，damrstr コマンド
  tambkup コマンド，tamrstr コマンド
- システムジャーナルファイルを複写したファイルとバックアップしたユーザファイルを基に，ユーザファイルを回復する機能
  DAM FRC 機能
  TAM FRC 機能

バックアップとリストアおよびユーザファイルの回復手順については，マニュアル「OpenTP1 運用と操作」を参照してください。

## 4.1.4　OpenTP1 ファイルの保護

操作の誤りでファイルを破壊することは絶対に避けなければなりません。そして，重要なファイル（オンラインを続行するための制御情報，会社の業務情報など）ほど厳重な保護をしなければなりません。ここでは，OpenTP1 ファイルの保護について説明します。

### (1)　OpenTP1 ファイルシステム単位の保護

OpenTP1 ファイルシステム単位にファイルの所有者とアクセス権を管理して，不意な破壊からファイルを保護しておきます。ファイルへのアクセス権は，OS のコマンド（例　chmod コマンド）を使って設定します。

- ファイル所有者の区別：OpenTP1 管理者，OpenTP1 グループ
- ファイルのアクセス権：読み書きできる，読むことができる

システムジャーナルファイル，チェックポイントダンプファイル，ステータスファイルは，OpenTP1 がオンラインを続行するために重要なファイルです。このため，特定の OpenTP1 管理者と OpenTP1 グループのユーザしか操作できない OpenTP1 ファイルシステム（システム用）に割り当てることが必要です。

ユーザファイル，メッセージキューファイル，MQA キューファイルは，上記以外のユーザが操作できた方が使いやすいため，システム用ファイルとは別の OpenTP1 ファイルシステム（ユーザ用）にすることが必要です。

このように，OpenTP1 ファイルの種類や使い勝手を判断して，システム構築前に OpenTP1 ファイルシステムをシステム用とユーザ用に分けておく必要があります。

OpenTP1 ファイルシステムをシステム用とユーザ用に分けた場合の保護（所有者とアクセス権）の例を次の表に示します。

表 4-4　OpenTP1 ファイルシステムの保護（所有者とアクセス権）の指定例

| OpenTP1 ファイルシステム | 所有者 | | アクセス権 | | |
|---|---|---|---|---|---|
| | ユーザ ID | グループ ID | 所有者 | グループ内のユーザ | そのほかのユーザ |
| システム用 | OpenTP1 管理者 | OpenTP1 グループ | rw | r- | r- |
| ユーザ用 | OpenTP1 管理者 | OpenTP1 グループ | rw | rw | r- |

（凡例）

rw：読み書きができる。

r- ：読むことができる。

## (2)　OpenTP1 ファイル単位の保護

OpenTP1 管理者は，オンライン中に filchown コマンドで OpenTP1 ファイル単位に所有者とアクセス権を動的に変更できます。これによって，オンライン中に OpenTP1 ファイルの使用を特定のユーザに認めたり認めなかったりできます。

OpenTP1 ファイルの保護については，マニュアル「OpenTP1 運用と操作」を参照してください。

# 4.1.5　OpenTP1 ファイルシステムの割り当て

ファイルの種類ごとにディスクを分けて OpenTP1 ファイルシステムを作成すれば，性能や信頼性を向上できます。しかし，ファイルの種類の数だけディスクを用意しなければならないため，コストが高くなります。

一つのディスクにすべてのファイルシステムを作成すれば，コストを低く抑えられます。しかし，ディスクが破壊された場合にすべてのファイルが使えなくなるため，信頼性は良くありません。

ファイルの構成を決めるときに考慮する，代表的な項目を次に示します。

- ハードウェア構成上の考慮

キャラクタ型スペシャルファイル上に OpenTP1 ファイルシステムを作成する場合，ディスクの構成が決まっているときには，OpenTP1 ファイルシステムに割り当てられるパーティション数とその容量を知っておく必要があります。

- **システム構成上の考慮**
  - システム構成上の機能から，必要な OpenTP1 ファイルの種類を決めます。例えば，TP1/Message Control を使わない場合は，メッセージキューファイルは必要ありません。
  - 一日のトランザクション発生数などから，システムジャーナルファイルやチェックポイントダンプファイルの容量を決めます。また，メッセージ送受信数やそのメッセージ長などから，メッセージキューファイル容量を決めます。
- **信頼性の考慮**

  OpenTP1 ファイルシステムを重要度別に分けることが必要です。

  例えば，システム用の OpenTP1 ファイルシステムとユーザ用の OpenTP1 ファイルシステムに分けます。
  - システム用：システムジャーナルファイル，チェックポイントダンプファイル，ステータスファイル，ノードリストファイル
  - ユーザ用：メッセージキューファイル，MQA キューファイル，ユーザファイル
- **性能上の考慮**

  アクセスが集中する複数のファイルを一つのディスクに割り当てると，データの入出力時間が掛かり性能が悪くなります。これを防ぐため，アクセスが集中するファイル同士はディスクを分ける必要があります。

最適なファイル構成にするためには，上記の項目などを考慮して性能・信頼性・コスト・操作性などを総合的に判断する必要があります。

## 4.2　システムで使うファイル

障害に備えるため，OpenTP1 で各種の内部情報を取得するファイルを**システムファイル**といいます。OpenTP1 で使うシステムファイルを次に示します。

- ステータスファイル
- システムジャーナルファイル
- チェックポイントダンプファイル
- トランザクションリカバリジャーナルファイル（TRF）
- サーバリカバリジャーナルファイル（SRF）
- アーカイブジャーナルファイル
- ノードリストファイル

### 4.2.1　ステータスファイル

#### (1)　ステータスファイルの目的

前回の OpenTP1 が，dcstop コマンドを入力して終了したか，障害が起こって異常終了したかを判断して，再開始処理を自動化するためのデータを格納します。OpenTP1 を開始するために引き継がなければならない**システム制御情報**を，状態が変化した時点で格納します。システム制御情報には，次のようなものがあります。

- OpenTP1 の終了状態
- 開始処理中，稼働中，終了処理中などシステムサービスや UAP の稼働状況
- システムファイルとユーザファイルのオープン・クローズ状態
- MCF で接続している各種装置の接続状態

#### (2)　ステータスファイルの構成

ステータスファイルは，OpenTP1 を開始するためのきわめて重要なファイルです。ステータスファイルに何らかの障害が起こった場合でも OpenTP1 を開始できるように，二つのステータスファイルを組にして二重化します。組になった二つのステータスファイルを**論理ファイル**といい，個々のステータスファイルをそれぞれ **A 系**，**B 系**と呼んで区別します。ステータスサービス定義で，論理ファイルを 7 個まで指定し，論理ファイルとステータスファイルの対応関係を定義します。

二つのステータスファイルが同時に障害にならないように，A 系のステータスファイルと B 系のステータスファイルは，なるべくディスクを分けてください。組になる A 系と B 系のステータスファイルの容量とレコード長は，同じでなければなりません。

ユーザは，ステータスサービス定義で論理ファイルに任意の名称を指定します。この**論理ファイル名**を使用することで，A 系，B 系のステータスファイルを同時にオープン，クローズするなど，論理ファイル単位に運用できます。

### (a) ステータスファイルの片系運転

オンライン中にステータスファイルに障害が発生して，予備ファイルがない場合でも，異常終了しないで正常な系だけで処理を続けられます。これを**ステータスファイルの片系運転**といいます。片系運転をするかどうかは，ステータスサービス定義に指定します。

片系運転中にシステムで障害が発生した場合，再開始のための情報が失われるため，再開始ができません。片系運転に切り替わったことを知らせるメッセージログが出力された場合は，使用できる予備ファイルを早急に用意して，正常な運転に復帰させてください。

## (3) ステータスファイルの状態

ステータスファイルは次の状態に分けられます。

- **現用**
  現時点でシステム制御情報の出力対象になっているオープン中の状態です。ファイルの実体が必要です。
- **予備**
  現時点でシステム制御情報の出力対象にはなっていないが，現用のステータスファイルが入出力障害などで使用できなくなったときに，現用のステータスファイルと切り替えるためにオープン中の状態です。ファイルの実体が必要です。
- **無効**
  ステータスサービス定義に指定されているが，オープンしないとオンラインでは使用できないクローズ中の状態です。また，削除されて実体のない状態を，実体なしの無効ファイルといいます。OpenTP1 開始時にファイルの実体がない無効ファイルがあると，OpenTP1 は開始できません。
- **閉塞**
  オンライン中に障害が発生し，閉塞しているクローズ中の状態です。

OpenTP1 は，OpenTP1 の開始時に，ステータスサービス定義で指定したすべてのステータスファイルをオープンします。オープンしたステータスファイルのうちステータス情報の出力対象となっているファイルを現用ファイルといい，オープン中のその他のファイルを予備ファイルといいます。ステータスファイルに障害が発生しないかぎり，ステータス情報は常に同じステータスファイルに格納されます。

システム制御情報は，まず A 系のステータスファイルに書き込まれ，その後 B 系のファイルに書き込まれます。このため，A 系ファイル書き込み中に OpenTP1 が異常終了し，A 系ファイルが利用できなくなっても，B 系ファイルは元の状態を保っているので，全面回復時に B 系ファイルを読み込むことで再開始できます。

### (4) ステータスファイルのスワップ

ステータスファイルは論理ファイル単位で現用ファイルを切り替えます。A 系または B 系のステータスファイルに入出力障害が発生すると，OpenTP1 は正常な系のステータスファイルからシステム制御情報を予備の論理ファイルに複写します。システム制御情報は，まず正常な系のファイルを予備の A 系ファイルに複写し，次に予備の B 系ファイルに複写します。複写が終わると，現用と予備を切り替えます。

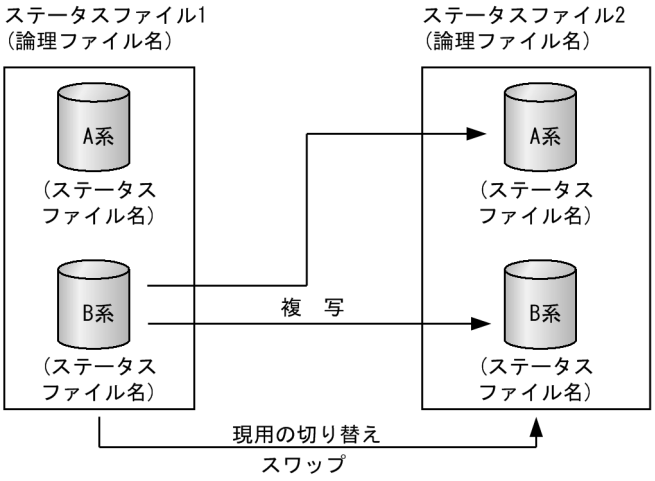
予備の論理ファイルがない場合には，OpenTP1 は異常終了します。OpenTP1 が異常終了した場合の処置は，「5.3　障害対策の概要」を参照してください。ただし，ステータスファイルの片系運転をする指定があれば，片系運転となります。

障害の発生によって閉塞したステータスファイルは，stsrm コマンドで削除し，容量を変更して再作成できます。stsrm コマンドで削除できるのは，閉塞・無効状態のファイルです。現用・予備ファイルは削除できません。

ステータスファイル内のシステム制御情報は，スワップすることでステータスファイルの先頭から再編成されます。OpenTP1 の開始・終了を繰り返すとステータスファイルにフラグメンテーションが発生することがあります。このため，OpenTP1 は必要に応じて自動的にステータスファイルをスワップしてフラグメンテーションを解消します。

ステータスファイルのスワップを次の図に示します。ステータスファイルに障害が発生した場合の処理については，「5.3　障害対策の概要」を参照してください。

図 4-5　ステータスファイルのスワップ




## 4.2.2　システムジャーナルファイル

### (1) システムジャーナルファイルの目的

オンラインシステムが停止した場合の全面回復や，UAP に障害が発生した場合の UAP 部分回復に必要な，回復用の履歴情報を格納します。回復用の履歴情報には，**同期点ジャーナル**と**回復用ジャーナル**があ

ります。全面回復の処理では，最新の世代のチェックポイントダンプと，最新の世代のチェックポイントダンプ取得以降に取得した回復用のジャーナルを基にします。UAP の部分回復ではチェックポイントダンプを使用しないで，回復用のジャーナルだけで処理します。チェックポイントダンプについては，「4.2.3 チェックポイントダンプファイル」を参照してください。

回復用のジャーナルとは別に，システムをチューニングするための統計用ジャーナルや，UAP の任意の履歴情報であるユーザジャーナルも格納します。

## (2) システムジャーナルファイルの構成

OpenTP1 は，システムジャーナルファイルを**ファイルグループ**という論理的な単位で運用します。これに対して，実際にジャーナルを取得するファイル実体を，**物理ファイル**といいます。通常一つのファイルグループは，一つの物理ファイルで構成しますが，ジャーナルファイル二重化機能を使用する場合は，一つのファイルグループを二つの物理ファイルで構成します。この場合，一つのファイルグループ中の，二つの物理ファイルを A 系，B 系と呼んで区別します。

システムジャーナルサービス定義では，ファイルグループを少なくとも二つ以上指定し，ファイルグループと物理ファイルの対応関係を指定してください。さらにシステムジャーナルサービス定義でファイルグループに任意の名称を指定してください。この**ファイルグループ名**を使用することで，A 系，B 系の物理ファイルを同時にオープン，クローズするなど，ファイルグループ単位に運用できます。

## (3) システムジャーナルファイルの二重化

一つのファイルグループを二つの物理ファイルで構成してシステムジャーナルファイルを運用できます。これを**システムジャーナルファイルの二重化**といいます。この場合，OpenTP1 は A 系，B 系に同じジャーナルを取得します。何らかの異常が発生し，その処理のためにジャーナルを読み込むとき，片方の系が障害でも残りの系を読めるため，信頼性を向上できます。

組になる A 系と B 系の物理ファイルは，同時に障害にならないように，なるべくディスクを分けてください。また，A 系と B 系の物理ファイルの容量は，一致していなくてもかまいませんが，容量が一致していない場合には，容量の少ないファイルに合わせてジャーナルを取得します。なるべく容量を一致させてください。

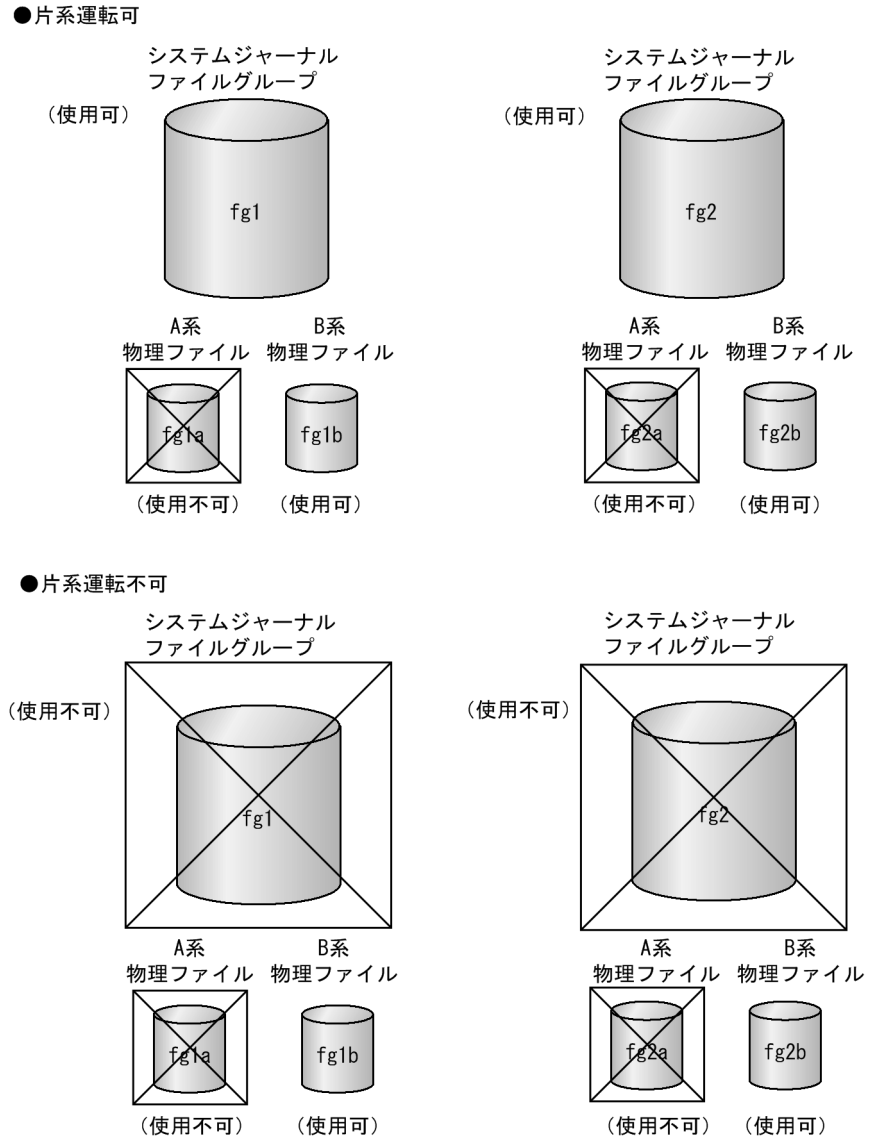
システムジャーナルファイルを二重化した場合，**片系運転可**，**片系運転不可**を選択できます。ファイルグループを構成する二つの物理ファイルのうち，一つが使用できる状態ならファイルグループを使用可能な状態とすることを片系運転可といいます。ファイルグループを構成する二つの物理ファイルが両方使用できる状態の場合にだけ，ファイルグループを使用可能な状態とすることを片系運転不可といいます。

例えば，ファイルグループ fg1，fg2 があり，それぞれが物理ファイル fg1a，fg1b，fg2a，fg2b で構成されているとします。物理ファイルの障害は，ディスクデバイス全体の障害である場合が多く，同一ディスク上の物理ファイルがすべて障害になることが多いのです。A 系の物理ファイル fg1a と fg2a が障害になった場合，片系運転不可では，使用可能なファイルグループがなくなるので，OpenTP1 は異常終了します。このため，障害を対策後，全面回復することになります。片系運転可では，B 系の物理ファイル fg1b，fg2b が正常な限りファイルグループは使用可能なので，OpenTP1 は続行可能です。

このように片系運転可とは，片方の系が障害になっても，その障害が対策されるまでの間に残りの系で運転を続行させるものです。ただし，片系運転可では，一時的に二重化されない期間が発生するため，この部分は信頼性が低下します。片系運転可，片系運転不可はシステムジャーナルサービス定義で指定します。

片系運転可，片系運転不可を次の図に示します。

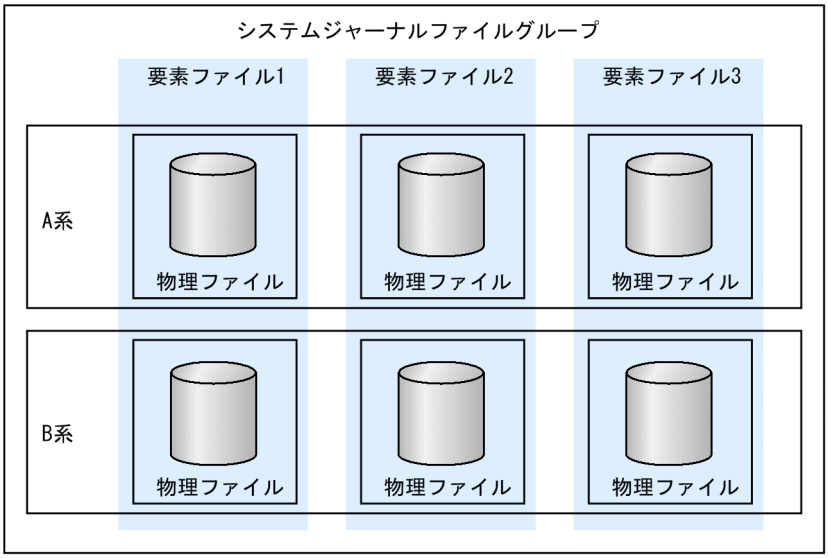
図 4-6　片系運転可，片系運転不可




## (4)　システムジャーナルファイルの並列アクセス機能

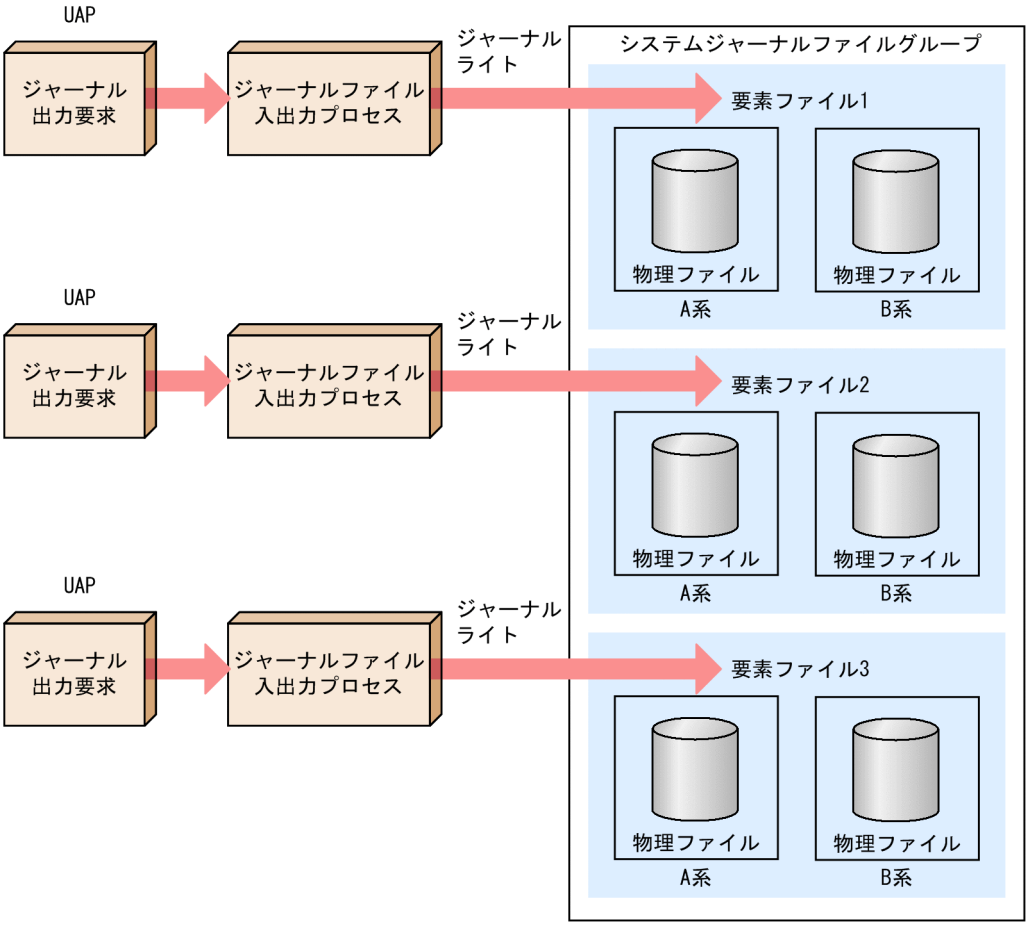
一つのファイルグループを複数の要素ファイルで構成して，システムジャーナルファイルを使えます。この機能を**システムジャーナルファイルの並列アクセス機能**といいます。並列アクセス機能で二重化した場合のファイルグループの構成を次の図に示します。

図 4-7　並列アクセス機能使用時のファイルグループの構成図（jnl_max_file_dispersion=3 の場合）




並列アクセス機能の概要を次の図に示します。

図 4-8　並列アクセス機能の概要（jnl_max_file_dispersion=3 の場合）



ジャーナルの入出力が集中すると，ディスクへの負荷が増大しI/O性能が低下します。このような場合に，ファイルグループを複数の**要素ファイル**で構成して，システムジャーナルファイルへ並行にアクセスできるようにすると，ジャーナルの入出力の性能を上げることができます。

一つのファイルグループを構成する物理ファイルは，SCSIインタフェースやハードディスクが重なるようになっていてもかまいませんが，これらには並行にアクセスできないため，並列アクセス機能の効果を十分に発揮できません。SCSIインタフェースやハードディスクは，重ならないようにしてください。また，複数の物理ファイルの容量は一致していなくてもかまいませんが，一致していない場合，OpenTP1は最も少ない物理ファイルの容量を要素ファイルの容量と見なします。資源を効率良く使えるように，物理ファイルの容量はできるだけ一致させることをお勧めします。

システムジャーナルファイルの並列アクセス機能では，システムジャーナルファイルの障害などで，並行してアクセスできる要素ファイルの数（以降，並列アクセス数といいます）が減ります。これによるジャーナル入出力性能の低下を防ぐため，最低限保証したい並列アクセス数を指定できます。この最低限保証したい並列アクセス数を，**最小分散数**といいます。また，最大でどれだけ並列アクセスするかを示す並列アクセス数を，**最大分散数**といいます。

システムジャーナルサービス定義でファイルグループを指定するときに，要素ファイル名を付けます。さらに，システムジャーナルサービス定義のjnl_max_file_dispersionオペランドに最大分散数を，jnl_min_file_dispersionオペランドに最小分散数を指定しておきます。

## (5) 取得するジャーナルの種類

### (a) 同期点ジャーナル

全面回復時やUAP部分回復時に，未決着のトランザクションを決着して資源の整合性を回復する必要があります。トランザクションが決着しているかいないか，同期点での処理がどこまで進んでいるのかを判断するために，2相コミットの過程で履歴情報を取得します。資源の回復が必要かどうかを判定するこの履歴情報を，**同期点ジャーナル**といいます。同期点ジャーナルには，PJ，HJ，BJ，TJ，DJがあります。同期点ジャーナルを次の表に示します。

表 4-5 同期点ジャーナル

| 種別 | 内容 |
|---|---|
| PJ | ルートトランザクションブランチ，トランザクションブランチがコミット処理を開始したときに出力されます。 |
| HJ | トランザクションブランチがコミット準備完了の状態になったときに出力されます。<br>ルートトランザクションブランチからは出力されません。 |
| BJ | トランザクションをロールバックしたときに出力されます。<br>ルートトランザクションブランチ，トランザクションブランチから出力されます。 |
| TJ | トランザクションの同期点処理を終了したときに出力されます。<br>ルートトランザクションブランチ，トランザクションブランチから出力されます。 |
| DJ | トランザクションをヒューリスティック決定したときに出力されます。<br>ルートトランザクションブランチ，トランザクションブランチから出力されます。 |

### (b) 回復用ジャーナル

全面回復時や部分回復時に，同期点ジャーナルによって資源を回復する必要がある場合，直ちにデータベースや回復対象テーブルのデータを回復します。資源の整合性を回復するため，資源の更新情報を取得します。この更新情報を，**回復用ジャーナル**といいます。回復用ジャーナルには，回復する資源の種類によって，FJ と CJ があります。回復用ジャーナルを次の表に示します。

表 4-6 回復用ジャーナル

| 種別 | 内容 |
|---|---|
| FJ | DAM ファイルの更新情報です。全面回復時と UAP 部分回復時に，このジャーナルを基に DAM ファイルを回復します。 |
| CJ | MCF サービスと TAM サービスが管理する回復対象テーブルの更新情報です。同期点取得時などに取得します。全面回復時にこのジャーナルを基に回復テーブルを回復します。 |
| | トランザクション管理サービスが管理する回復対象テーブルの更新情報です。同期点取得時などに取得します。全面回復時にこのジャーナルを基に回復テーブルを回復します。 |

### (c) 統計用ジャーナル

システムチューニングに必要な統計用の履歴情報です。統計用ジャーナルには，MJ，IJ，GJ，OJ，AJ，SJ があります。

MJ，IJ，GJ，OJ，AJ は，MCF を使用して他システムとメッセージ送受信するときにだけ取得します。MJ は，mcftactmj コマンドを入力して取得を開始し，mcftdctmj コマンドを入力して取得を終了します。IJ，GJ，OJ を取得するかどうかは，MHP のアプリケーションごとにアプリケーション属性定義で指定します。AJ を取得するかどうかは，論理端末ごとに論理端末定義で指定します。

SJ は OpenTP1 システム全体の統計情報です。SJ のうちシステム統計情報は，統計取得用の運用コマンドを入力すると，それ以降ユーザの指定した一定間隔で取得します。トランザクション統計情報は，トランザクションサービス定義で取得するかどうかを指定します。統計用ジャーナルを次の表に示します。

表 4-7 統計用ジャーナル

| 種別 | 内容 |
|---|---|
| MJ | 入力編集前のメッセージ入力情報と，出力編集後のメッセージ出力情報です。 |
| IJ | 受信メッセージを入力キューに格納する直前に取得するメッセージ入力情報です。 |
| GJ | MHP が受信メッセージを受け取る直前に取得する MHP 受信情報です。 |
| OJ | 送信メッセージを出力キューに格納した直後に取得するメッセージ出力情報です。 |
| AJ | メッセージを他システムに送信したあと，他システムからの送信完了報告を受信した直後に取得する送信完了情報です。 |
| SJ | OpenTP1 の稼働状態全般を把握するための稼働統計情報です。システム統計情報とトランザクション統計情報があります。システム統計情報は，システムサービスと UAP に関する統計情報です。一定時間ごとに OpenTP1 の状態を取得します。トランザクション統計情報は，トランザクション処理に関する統計情報です。トランザクションブランチごとに同期点処理が完了した時点で取得します。 |

### (d) ユーザジャーナル（UJ）

UAP から関数を呼び出してシステムジャーナルファイルに出力する，ユーザ任意の情報です。ユーザジャーナルは，システムのチューニングやユーザ業務そのものに利用します。ユーザジャーナルは，トランザクション内，またはトランザクション外で出力できます。

## (6) システムジャーナルファイルの状態

システムジャーナルファイルのファイルグループの状態は，次の 2 種類に分けられます。

- **使用可能状態**
  ファイルグループを構成する要素ファイルのうち，最小分散数以上の要素ファイルがオープンされている状態
- **使用不可能状態**
  ファイルグループを構成する要素ファイルのうち，最小分散数以上の要素ファイルがオープンされていない状態

使用可能状態のファイルグループは，現用，または待機として，使用不可能状態のファイルグループは予約として，状態を管理します。なお，要素ファイルの必要数は，システムジャーナルサービス定義に指定した値で決まります。

- **現用**
  現時点でジャーナルの出力対象になっている使用可能状態のファイルグループです。この状態のファイルグループは，常に一つです。
- **待機**
  現時点でジャーナルの出力対象になっていませんが，現用に変更するために待機している使用可能状態のファイルグループです。
  待機の状態は，次の二つに分けられます。
  - **次回スワップ先にできる状態**
    上書きできる（回復に必要なジャーナルがない）状態で，かつアンロード済み状態の待機ファイルグループです。
  - **次回スワップ先にできない状態**
    上書きできない状態，またはアンロード待ち状態の待機ファイルグループです。
- **予約**
  使用不可能状態のファイルグループです。

システムジャーナルファイルには，予約以外のファイルグループが二つ以上必要です。

OpenTP1 を正常開始で開始すると，システムジャーナルサービス定義で指定したファイルグループのうち，ONL と指定したファイルグループがすべてオープンされます。オープンされたファイルグループのうち，最初に指定したファイルグループが現用となって，そのほかは待機となります。また，構成する物理

ファイルが必要な数に満たなかったファイルグループや，ONL と指定しなかったファイルグループは，予約となります。

システムジャーナルファイルでは，常に現用のファイルグループにジャーナルが出力されます。現用のファイルグループが満杯になると，待機のファイルグループのうち一つを現用に切り替えます。このとき，以前の現用ファイルグループは，待機ファイルグループになります。使用可能状態のすべてのファイルグループが満杯となると，最初のファイルグループに戻ってジャーナルを出力します。

## (7)　システムジャーナルファイルのアンロード

待機のシステムジャーナルファイルのジャーナルは，ファイルに複写しておく必要があります。ファイルにジャーナルを複写することを，**アンロード**といいます。システムジャーナルファイルのアンロードには，jnlunlfg コマンドを使用するか，またはシステムジャーナルサービス定義の jnl_auto_unload オペランドに Y を指定して自動アンロード機能を使用します。ユーザファイルの回復や統計用ジャーナルの編集，ユーザジャーナルの運用は，ファイルにアンロードしたジャーナルを使います。ジャーナルをアンロードしたファイルを，**アンロードジャーナルファイル**といいます。OpenTP1 は，ファイルグループがアンロードされているかどうかチェックして，アンロードされていない待機ファイルグループはスワップ先にしません。ファイルグループがアンロードされているかどうか，チェックすることを**アンロードチェック**といいます。スワップ先がない場合には，OpenTP1 は異常終了します。OpenTP1 が異常終了したときの処置は，「5.3　障害対策の概要」を参照してください。

### (a)　アンロードチェックの抑止

通常はアンロード待ち状態になる待機のファイルグループを，現用のまま使える指定ができます。これをアンロードチェックの抑止といいます。アンロードチェックを抑止する場合は，システムジャーナルサービス定義の jnl_unload_check オペランドに N を指定してください。

アンロードチェックを抑止している場合は，OpenTP1 の稼働中にジャーナルファイルをコマンドでアンロードしないでください。アンロードチェックの抑止を指定した場合にアンロードを実行するときは，いったん jnlclsfg コマンドで該当するファイルグループをクローズしてから，アンロードしてください。

アンロードチェックを抑止するとアンロードジャーナルファイルは作成されないため，アンロードジャーナルファイルを入力とするジャーナル編集コマンド（jnlcolc コマンド，jnlstts コマンドなど）は実行できません。

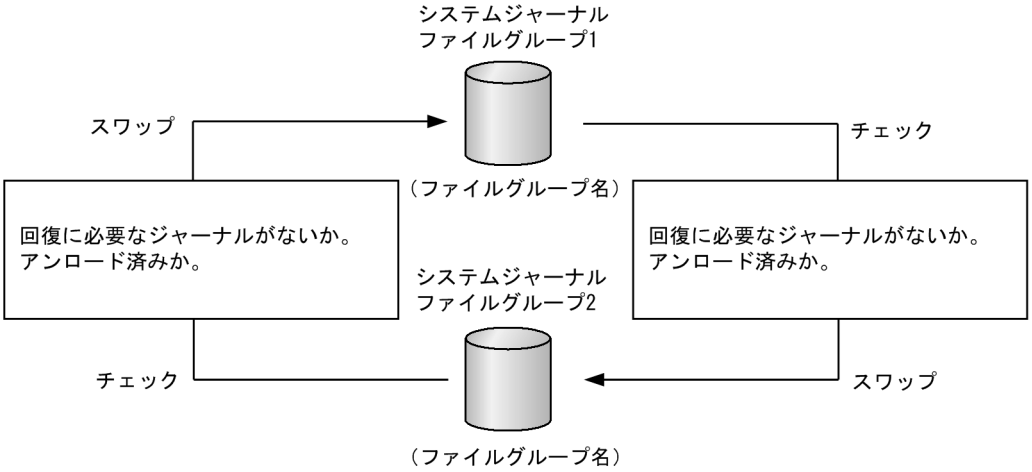
アンロードチェックを抑止した場合は，ファイルグループの状態を jnlls コマンドで確認すると，アンロード待ち状態が出力されることがあります。アンロード待ち状態が出力された場合でも，実際は現用として使われています。

## (8)　システムジャーナルファイルのスワップ

待機状態のシステムジャーナルファイルには，スワップ先にできるものとスワップ先にできないものとがあります。スワップ先にできるシステムジャーナルファイルは，アンロード済みで，回復に必要なジャーナルがないファイルグループです。アンロードしていないファイルグループ，または回復に必要なジャーナルがあるファイルグループは，スワップ先にできません。回復に必要なジャーナルがあるかどうかの

チェックについては，「4.2.3(4)　複数世代保証機能」を参照してください。システムジャーナルファイルのスワップ先がない場合には，OpenTP1 は異常終了します。OpenTP1 が異常終了したときの処置は，「5.3　障害対策の概要」を参照してください。システムジャーナルファイルのスワップを次の図に示します。

図 4-9　システムジャーナルファイルのスワップ




## (9)　全面回復時の予約状態のファイルのオープン

オンライン障害発生後の全面回復時，OpenTP1 はトランザクションを決着するためのジャーナルを出力します。このとき，システムジャーナルファイルにジャーナルを格納しきれなくなった場合には，OpenTP1 は再び異常終了します。全面回復時にジャーナルファイルが不足したとき，予約状態のファイルをオープンして，ジャーナルを格納できます。予約状態のファイルをオープンするかどうか，システムジャーナルサービス定義で指定します。なお，アンロードすればスワップ先としてジャーナルを格納できる，上書きできる状態のシステムジャーナルファイルがある場合は，予約状態のファイルをオープンしません。jnlunlfg コマンドでジャーナルファイルをアンロードしてください。

## (10)　ジャーナル容量に対する考え方

### (a)　理想的なジャーナル容量

理想的なジャーナル容量は，OpenTP1 起動から停止までの間に取得されるジャーナルすべてを格納できるだけの容量です。これが実現できると，OpenTP1 稼働中にジャーナルをアンロードする運用が不要となり，OpenTP1 停止後にジャーナルファイルのメンテナンスができます。

```
理想となる目安：
（トランザクション当たりの平均ジャーナル量）
×（オンライン開始から終了までの総トランザクション数）
＋オンライン開始から終了までにdcstatsコマンドによって取得する統計情報のジャーナル量
＋（UAPのOpenTP1のRPC，1回当たりのジャーナル量）
×（オンライン開始から終了までのUAPから発行するOpenTP1のRPCの回数の総和）
```

ジャーナルの容量の計算式の詳細は，マニュアル「OpenTP1 運用と操作」のシステムジャーナルファイルのサイズの見積もり式の記述を参照してください。

### (b) 現実的なジャーナル容量の見積もり

実際の運用では，24 時間稼働，ディスク容量を確保できないなどの理由で理想どおりのジャーナル容量を用意できないことがあります。

OpenTP1 は，ジャーナルをアンロード，またはステータスを変更することでジャーナルファイルを再利用可能としており，この機能を使用することで理想とする容量より小さいジャーナル容量で運用を行えます。

ジャーナルのアンロードを行う場合，アンロードするタイミングをどのようにするかのシステム運用設計が必要です。例えば，次のような運用が考えられます。

- ジャーナルファイルがスワップした時点のメッセージ（KFCA01222-I）をトリガーにアンロードする。
- OpenTP1 が提供する自動アンロード機能を使用することでアンロードさせる。
- 運用でまとめてアンロードする（例えば，午前と午後の間にアンロードするなど)。

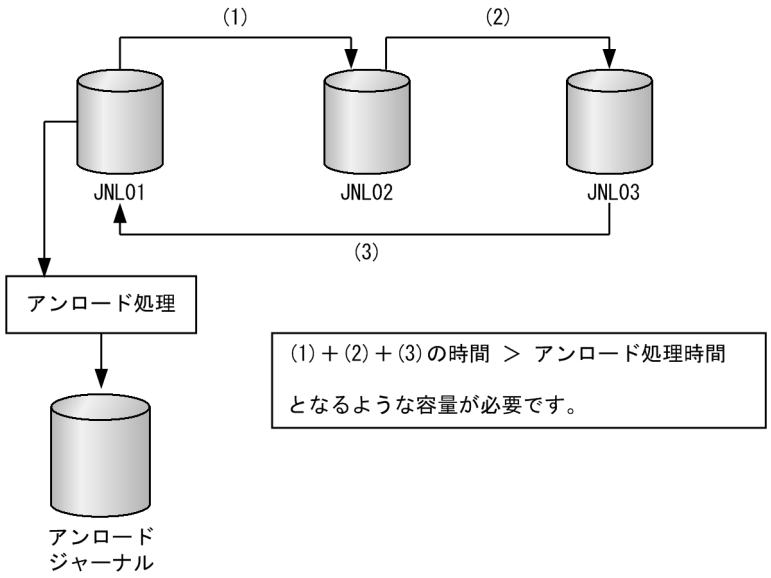
ジャーナルのアンロードを行わないで，ジャーナルを捨てる運用を行う場合，jnlchgfg コマンドで強制的にアンロード済みの状態に変更したり，システムジャーナルサービス定義の jnl_unload_check オペランドに N を指定したりすることで，ジャーナルファイルを再利用可能にできます。

上記運用で，アンロードまたはジャーナルを捨てることで，ジャーナルファイルの再利用が可能となり，ジャーナル容量を減らせます。

### (c) ジャーナル容量見積もりの注意事項

ジャーナル容量を減らす場合，ジャーナルの総容量が小さすぎるとアンロード処理中にジャーナルファイルを使いきり，OpenTP1 ダウンとなるおそれがあります。このため次の点に注意してください。

ジャーナルファイルが一周するまでの時間 ＞ ジャーナルアンロード処理時間
（ジャーナルアンロードする時間に対して，ジャーナルファイル出力に追いつかれないようにすることが必要です）



### (d) ジャーナルとチェックポイントダンプの関係

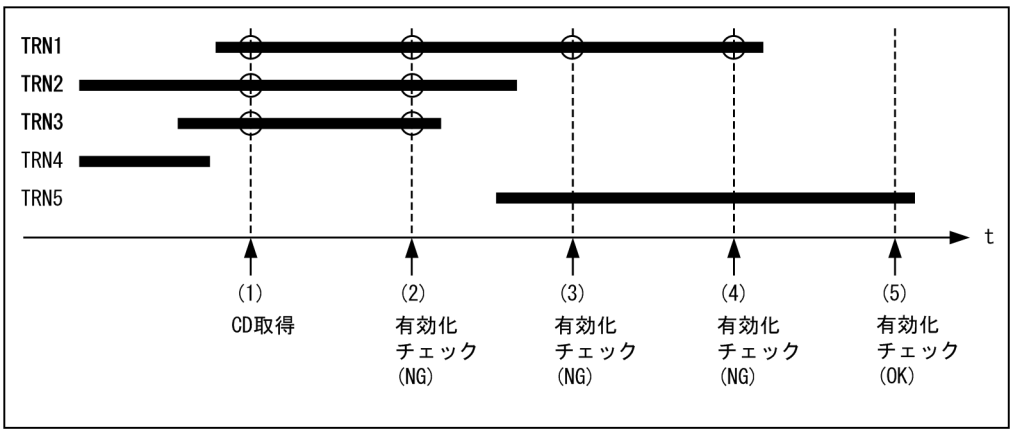
チェックポイントダンプを取得すると，その時点より過去のジャーナルファイルについては再利用できます。

チェックポイントダンプは，次の契機で取得されます。

- システムジャーナルファイルがスワップしたとき。
- システムジャーナルサービス定義で指定した数のジャーナルブロックをシステムジャーナルファイルに格納したとき。
- システムサービスの開始・終了時。
- チェックポイントダンプ取得中にスキップが発生し，その取得が完了したとき。

ただし，チェックポイントダンプは取得してもすぐに有効にはなりません。チェックポイントダンプを取得するタイミングで動作していたトランザクションがすべて完了するまで有効化を待ち合わせています。

次のケースでは，(1)のチェックポイントダンプ取得時点で動作していた TRN1，TRN2，TRN3 がすべて完了するまで有効になりません。(5)のタイミングで有効化できるかチェックした時点で TRN1，TRN2，TRN3 がすべて完了していることが確認できるため，このタイミングで(1)で取得したチェックポイントダンプが有効となります（(1)より後に発生した TRN5 は，(1)のタイミングのチェックポイントダンプ対象とはならないで，次のチェックポイントダンプの対象となります)。




チェックポイントダンプの有効化に時間が掛かるようなケースでは，ジャーナルファイルが再利用可能となるまでの時間も遅れることになるため，OpenTP1 ダウンとなるおそれがあります。

このような事態を想定し，ジャーナルとチェックポイントダンプの関係は次のようになるよう設計してください。

チェックポイント取得時間 ≪ ジャーナルファイルが一周するまでの時間

（チェックポイント取得時間に対して，ジャーナルファイル出力に追いつかれないようにすることが必要です）

チェックポイントの有効化に要した時間は，OpenTP1 のシステム統計情報から求められます。「チェックポイント有効化（取得時間)」が上記の(1)から(5)の時間を示しています。この時間を目安にし，さらに十分な余裕を持った時間を見込んでジャーナルファイルの容量を見積もってください。

## (e) 運用監視

OpenTP1 のシステムダウンを未然防止するため，次のようなシステムメッセージを監視し，マニュアル「OpenTP1　メッセージ」に従った運用対処を行うようシステム設計してください。

### (i) 再利用可能なジャーナル不足の監視

再利用可能なジャーナルがなくなると，KFCA01220-E メッセージを出力し，OpenTP1 は緊急停止（システムダウン）します。この状態になる前に，警告として次のメッセージを出力します。

- KFCA01224-I aaaa(xx....xx)ジャーナルには，次のスワップ要因の発生時に交代先として使用できるファイルグループがありません。

  再利用可能なジャーナルファイルが 1 つしか残っていない状態です。このまま放置すると OpenTP1 ダウンとなる危険性があります。至急ジャーナルファイルを追加するか，再利用可能になっていないファイルに対して対処をしてください。

### (ii) チェックポイントダンプ有効化遅れの監視

チェックポイントダンプ取得契機に，長時間有効化できない状態の場合は，警告として次のメッセージを出力します。

- KFCA02179-I aa....aa サービスのチェックポイントダンプ取得契機をスキップしました。 スキップ回数＝ bb....bb　ジャーナル世代番号 cc....cc　要因コード＝ ddd-ee

  チェックポイント取得契機になりましたが，処理をスキップしています。チェックポイントダンプ取得処理が走行中のため，再利用可能なジャーナルファイルが減少しています。ジャーナルファイルの状態およびトランザクション状態を確認して対処してください。

### (iii) 自動アンロード機能停止の監視

自動アンロード機能使用時に障害が発生した場合，警告として次のメッセージを出力して，自動アンロード機能を停止します。

- KFCA01173-W aaaa(xx....xx)ジャーナルファイルの自動アンロード機能を停止しました。理由コード=bbbb
- KFCA01174-W aaaa(xx....xx)自動アンロード処理中ですが，処理を中断します。理由コード=bbbb
- KFCA01178-E aaaa(xx....xx)自動アンロードの処理中に障害が発生しました。理由コード=bbbb

  自動アンロード機能を停止します。障害要因を取り除いたあと，自動アンロード機能を再開始してください。
- KFCA01179-W aaaa(xx....xx)自動アンロード先ディレクトリを切り替えることができません。切り替え先:bb....bb, 理由コード=cccc

定義されているディレクトリに自動アンロード先ディレクトリを切り替えます。定義されているすべての自動アンロード先ディレクトリが使用できない場合は，自動アンロード機能を停止します。障害要因を取り除いたあと，自動アンロード機能を再開始してください。

## 4.2.3　チェックポイントダンプファイル

### (1)　チェックポイントダンプファイルの目的

オンラインシステムが停止した場合，回復用のジャーナルだけで回復すると，オンライン開始からのすべてのジャーナルが必要となり，回復に膨大な時間が掛かります。そこでオンライン運転中に一定の間隔でポイントを設け，そのポイントで回復する必要のあるテーブルの状態を保存することで，ポイント以前の回復用ジャーナルの必要をなくして，回復時間の短縮を図ります。このポイントを**チェックポイント**といい，チェックポイントで取得するテーブル情報を**チェックポイントダンプ**といいます。オンラインシステムの全面回復には，最新のチェックポイントダンプの取得からオンラインシステム停止までの間で取得した，回復用のジャーナル全部が必要です。

OpenTP1 は，システムサービスごとに回復処理しているため，メモリ上のテーブルを回復する必要があるシステムサービスごとに，チェックポイントダンプを取得しています。チェックポイントダンプを格納するファイルは，システムサービスごとに割り当てます。このファイルを**チェックポイントダンプファイル**といいます。

チェックポイントダンプは，次のタイミングで取得します。

- システムジャーナルファイルがスワップしたとき
- システムジャーナルサービス定義で指定した数のジャーナルをシステムジャーナルファイルに格納したとき
- システムサービスの開始・終了時

### (2)　チェックポイントダンプファイルの構成

チェックポイントダンプファイルは，**ファイルグループ**という論理的な単位で管理されます。これに対して，実際にチェックポイントダンプを取得するファイルの実体を，**物理ファイル**といいます。一つのファイルグループは，一つまたは二つの物理ファイルで構成されます。

一つのファイルグループを二つの物理ファイルで構成する形態を**チェックポイントダンプファイルの二重化**といいます。二重化する場合は，それぞれの物理ファイルを A 系，B 系と区別します。

チェックポイントダンプファイルが必要なシステムサービスは，チェックポイントダンプサービス定義にシステムサービス名，ファイルグループ，および物理ファイルの対応関係を指定します。

チェックポイントダンプサービス定義では，ファイルグループに任意の名称を指定します。この**ファイルグループ名**を使用してチェックポイントダンプファイルを運用します。

チェックポイントダンプファイルのファイルグループは次のような状態に分けられます。

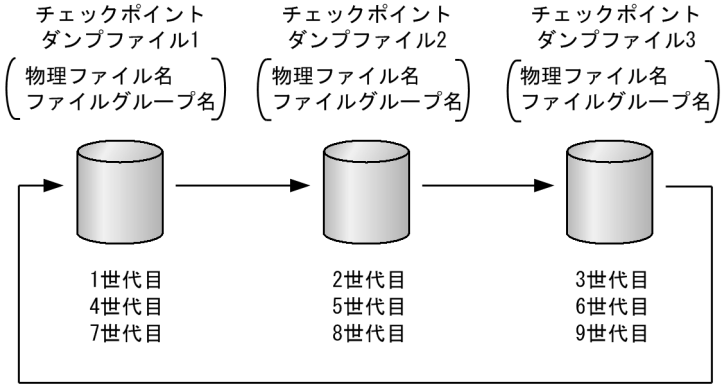
- **上書きできる，または書き込み中の状態：**
  OpenTP1 システムの全面回復時に使用するチェックポイントを含まないので上書きできるオープン中の状態，またはチェックポイントダンプを取得中の状態です。
- **上書きできない状態：**
  OpenTP1 システムの全面回復時に使用するチェックポイントダンプを含んでいるために，上書きを抑止しているオープン中の状態です。
- **予約：**
  オープンしないとオンラインで使用できない，クローズ中の状態です。定義しただけで実体がない場合にもこの状態になります。

チェックポイントダンプサービス定義で，OpenTP1 の開始と同時にオープンするファイルグループを定義します。オープンすると定義しなかったファイルグループは，開始時に予約の状態になります。

1 回に取得するチェックポイントダンプを**世代**といいます。一つの世代を一つのチェックポイントダンプファイルグループに取得するため，ファイルグループを世代ごとに切り替えます。用意したファイルグループすべてにチェックポイントダンプを取得すると，最初のファイルグループに戻ってデータを上書きします。このように，上書きしながら複数のファイルグループにデータを格納していく方式を，**ラウンドロビン方式**といいます。通常，最新の世代を取得したファイルグループを上書きできない状態にし，そのほかのファイルグループを上書きできる状態にしますが，複数世代保証機能を使用する場合は，二世代のファイルグループを上書きできない状態にします。

ラウンドロビン方式によるチェックポイントダンプの取得を次の図に示します。

図 4–10　ラウンドロビン方式によるチェックポイントダンプの取得




## (3)　チェックポイントダンプファイルの物理ファイルを追加

オンライン中にチェックポイントダンプファイルに障害が起こり，運用に必要なファイル数に足りなくなった場合，物理ファイルを動的に追加できます。動的に追加する方法は，事前に予約ファイルを定義しておくかどうかで異なります。

### (a) 定義していない物理ファイルをオンライン中に追加する方法

チェックポイントダンプサービス定義に指定していない物理ファイルでも，運用コマンドでオンライン中に追加できます。まず，jnladdpf コマンドで追加する物理ファイルをファイルグループに割り当てます。そして，jnlopnfg コマンドでファイルグループをオープンします。これでチェックポイントダンプファイルのオンライン中の追加が完了します。

### (b) 予約ファイルを事前に定義しておく方法

チェックポイントダンプサービス定義に事前に予約ファイルを定義してある場合は，jnlopnfg コマンドでオープンする方法と，OpenTP1 が自動的にオープンする方法（**自動オープン機能**）で，ファイルを割り当てられます。

自動オープン機能を使うと，オンライン中に物理ファイルが有効保証世代数にまで減った場合，予約ファイルを現用ファイルとして自動的に割り当てることができます。自動オープン機能を使用する場合は，チェックポイントダンプサービス定義に自動オープンを指定します。

### (c) チェックポイントダンプファイルの物理ファイルを削除

動的に追加した物理ファイルに限り，ファイルグループが予約状態であれば，jnldelpf コマンドで削除できます。

## (4) 複数世代保証機能

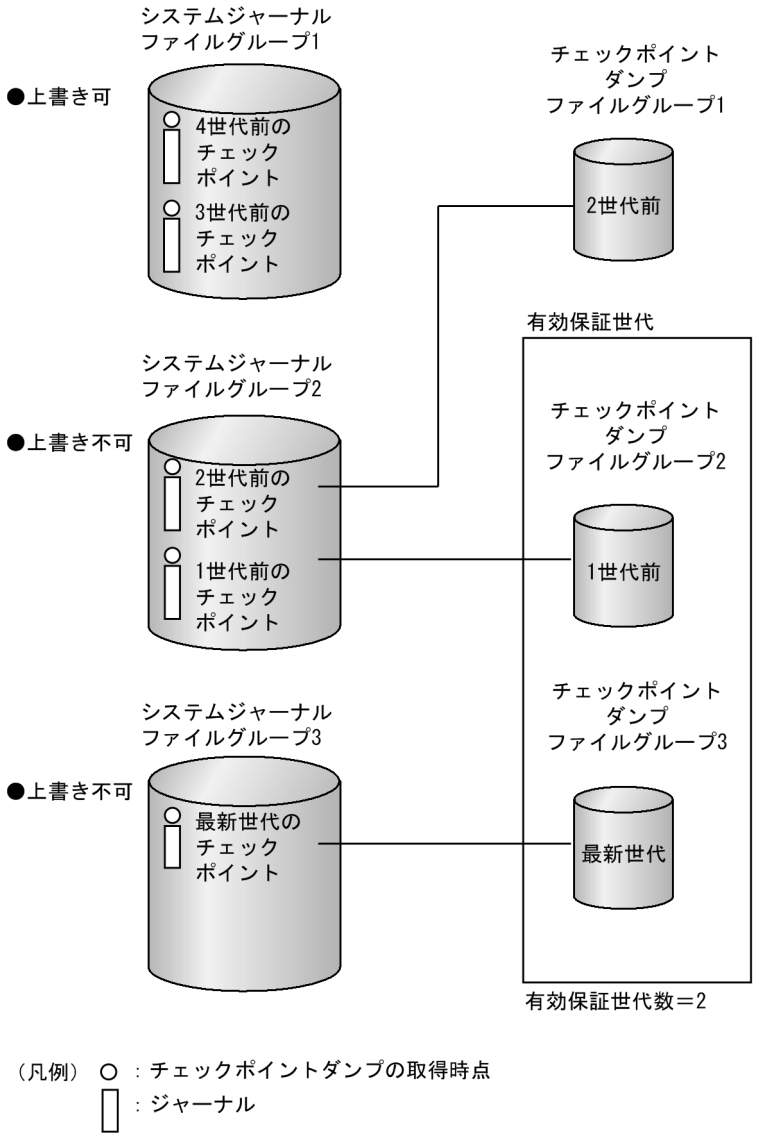
何らかの異常が発生し，最新世代のファイルグループを読み込めない場合でも，一世代前のファイルグループを読み込んで回復処理をすることで，信頼性を向上させることができます。回復処理に備えて，一世代前のファイルグループを上書きできない状態にして保存しておく機能を，**複数世代保証機能**といいます。最新世代も含めて，上書きできない状態にした世代を**有効保証世代**といい，上書きできない状態の世代の数を**有効保証世代数**といいます。複数世代保証機能を使用した場合の有効保証世代数は 2，使用しない場合は 1 になります。

OpenTP1 は，有効保証世代での回復に必要なジャーナルを持つシステムジャーナルファイルのファイルグループを，上書きできない状態にします。有効保証世代に基づくシステムジャーナルファイルの上書きチェックを，**オーバライトチェック**といいます。

### (a) 複数世代保証機能を使用した場合の回復手順

複数世代保証機能を使用した場合，まず最新世代のファイルグループを読み込みます。そして，最新世代のチェックポイント以降に取得した全ジャーナルを使用して回復処理をします。何らかの異常が発生して最新世代のファイルグループが読み込めないときには，一世代前のチェックポイントダンプファイルを読み込みます。このときには，一世代前のチェックポイント以降に取得した全ジャーナルを使用して回復処理をします。有効保証世代のファイルグループがすべて読み込めないときには，さらに世代をさかのぼりますが，有効保証世代以前のジャーナルが上書きされて回復できないことがあります。オンライン中に予備以外の状態のチェックポイントダンプファイルのファイルグループ数は，「有効保証世代数+1」以上，必要です。有効保証世代とシステムジャーナルファイルの関係を次の図に示します。

図 4–11　有効保証世代とシステムジャーナルファイル




## (b)　縮退運転機能を使用した場合の有効保証世代数

オンライン中，または再開始処理中に，運用に必要なファイル数「有効保証世代数+1」を下回った場合でも，最低二つのファイルが使用できれば，処理を続けられる指定もできます。これを**チェックポイントダンプの縮退運転**といいます。縮退運転をするかどうかは，チェックポイントダンプサービス定義で指定します。

縮退運転中に OpenTP1 システムで障害が発生した場合は，再開始のための情報が一つのファイルでしか保証されていないことになります。縮退運転となったときには，通常の運転から縮退運転に切り替わった旨を伝えるメッセージログが出力されます。このメッセージログが出力された場合は，有効保証世代として使用するファイルを早急に準備してください。

有効保証世代として使用できるファイルが確立できた場合，縮退運転の状態から通常の運転に切り替わった旨を伝えるメッセージログが出力されます。

## (5) 有効保証世代とジャーナル容量の関連

システムジャーナルファイルの容量は，チェックポイントダンプファイルの有効保証世代数に比例して，必要になります。システムジャーナルファイル全体でのジャーナルブロック数の目安は，次のとおりです。

チェックポイントダンプを取得する間隔となるジャーナルブロック数×（有効保証世代数＋ 1）

## (6) チェックポイントダンプファイルを二重化したときの運用

チェックポイントダンプファイルを二重化してファイルグループを運用した場合，OpenTP1 は A 系，B 系に同じチェックポイントダンプを取得します。二重化しておくと，チェックポイントダンプの読み込み時に片方の系で障害が起こっても，残った系からデータを読み込めるため，信頼性が上がります。

### (a) チェックポイントダンプサービス定義の指定

チェックポイントダンプファイルを二重化する場合は，チェックポイントダンプサービス定義の jnl_dual オプションに Y を指定します。このとき，一つのファイルグループに二つの物理ファイル（A 系，B 系）を指定しておきます。

A 系と B 系の物理ファイルは，同時に障害にならないようにディスクを分けることをお勧めします。また，A 系と B 系の物理ファイルの容量は，一致していなくてもかまいませんが，一致していない場合 OpenTP1 は少ない容量の方をチェックポイントダンプの容量と見なします。資源を効率良く使えるように，容量はできるだけ一致させておいてください。

### (b) チェックポイントダンプファイルの片系運転

チェックポイントダンプファイルを二重化するとき，一方の物理ファイルに障害が発生した場合，次のどちらかの運用を選択できます。

- 片系運転可

  A 系と B 系のどちらかの系の物理ファイルがオープン状態であれば，ファイルグループをオープン状態とするものです。どちらかの系の物理ファイルに障害が発生しても，正常な系で，処理を続行できます。これによって，正常な系のチェックポイントダンプを有効にできます。

- 片系運転不可

  A 系と B 系の両方の物理ファイルがオープン状態でなければ，ファイルグループを予約状態とするものです。どちらかの系の物理ファイルに障害が発生した場合，ファイルグループも予約状態となります。なお，A 系からの読み込み時に障害が発生した場合，B 系から読み込んだあとにファイルグループを予約状態にします。

### (c) 二重化した場合の運用上の注意

片系運転不可の場合，片方の系だけに対するオープン，クローズなどの操作はできません。

片系運転可の場合，片方の系だけに対する操作はできます。ただし，上書きできないファイルグループに対するクローズは，どちらの場合もできません。

片系運転可と片系運転不可の場合の相違点を次の表に示します。

表 4-8　片系運転可と片系運転不可の場合の相違点

| 操作内容 | 運用方法 | |
|---|---|---|
| | 片系運転可 | 片系運転不可 |
| 一方の系だけをオンライン中に割り当てる | 割り当てられます。 | 割り当てられます。 |
| 一方の系だけをオンライン中に削除する | 削除できます。 | 削除できます。 |
| 一方の系だけをオープンする | 両方の系が割り当てられていれば，オープンできます。 | オープンできません。 |
| 一方の系だけをクローズする（上書きできない状態） | クローズできません。 | クローズできません。 |
| 一方の系だけをクローズする（上書きできる状態） | クローズできます。 | クローズできません。 |

### (7)　チェックポイントダンプ取得契機のスキップ回数の監視

UAP の無限ループなどが発生すると，チェックポイントダンプの取得処理が連続してできない（チェックポイントダンプ取得契機がスキップされる）ことがあります。チェックポイントダンプが取得できないままオンライン処理を続行すると，回復に必要なシステムジャーナルファイルが多くなり，最終的には使用できるシステムジャーナルファイルが不足して，システムダウンすることがあります。

チェックポイントダンプ取得契機のスキップ回数を監視することで，このようなシステムダウンを防止できます。これによって，チェックポイントダンプ取得契機のスキップが一定の回数に達した場合に，トランザクションの実行プロセスを強制的に停止して，トランザクションを決着（コミットまたはロールバック）します。チェックポイントダンプ取得契機のスキップ回数を監視するためには，チェックポイントダンプサービス定義に jnl_cdskip_limit オペランドを指定します。

また，チェックポイントダンプサービス定義に jnl_cdskip_msg オペランドを指定すれば，チェックポイントダンプ取得契機がスキップされた場合に，スキップ要因となっているトランザクションの情報を KFCA32550-I メッセージで表示することもできます。

## 4.2.4　トランザクションリカバリジャーナルファイル

### (1)　トランザクションリカバリジャーナルファイルの目的

**トランザクションリカバリジャーナルファイル（TRF）**とは，システムジャーナルファイルとは別に，トランザクションに関する各種のジャーナル情報を，トランザクションブランチ（UAP プロセス）ごとに取得するファイルです。TRF は$DCDIR/spool/dctjlinf/ディレクトリ下に，trfXXXXXXXX（X は任意の

整数）というファイル名で UNIX ファイルとして作成されます。そのため，系切り替え構成では TRF を使用できません。

トランザクション処理では，一つの処理が長時間掛かることが原因で，トランザクションで更新した資源のジャーナル情報が，複数のシステムジャーナルファイルにわたって取得されることがあります。トランザクション処理の途中でオンラインが停止した場合，この資源を回復するためには複数のシステムジャーナルファイルを読み込む必要があります。このため，回復に時間が掛かります。

また，使用中のシステムジャーナルファイルはスワップ先にできないため，システムジャーナルファイルの容量や数によっては，トランザクション処理が終了するまでの間に，システムジャーナルファイルのスワップ先がなくなってしまうことがあります。システムジャーナルファイルのスワップ先がなくなると，OpenTP1 は異常終了します。

このような事態を避けるため，長時間掛かるトランザクション処理のジャーナルを，システムジャーナルファイルとは別のファイル（TRF）に格納します。TRF を作成することで，回復時に読み込むジャーナルの量を削減できます。また，長時間のトランザクションの実行中でも，使用中のシステムジャーナルファイルをスワップ先にでき，チェックポイントダンプを有効化できます。そのため，再開始時の回復時間を短縮できます。さらに，長時間掛かるトランザクション処理のために，ジャーナル情報のオーバフローによる OpenTP1 システムの異常終了を防げます。処理が長時間となる UAP では，TRF を取得することをお勧めします。ただし，TRF を取得した場合，システムジャーナルファイルと TRF に同じジャーナルを取得するため，トランザクション性能が悪くなります。

TRF を取得するかどうかは，サービスグループごとに，ユーザサービス定義で指定します。

### (2) トランザクションリカバリジャーナルファイルの障害時の回復方法

再開始処理中に TRF に障害が発生した場合には，その旨を伝えるメッセージログが出力されます。このメッセージログが出力された場合は，jnlmkrf コマンドで TRF を回復してください。TRF の回復にはアンロードジャーナルファイルを使用します。

## 4.2.5 サーバリカバリジャーナルファイル

### (1) サーバリカバリジャーナルファイルの目的

**サーバリカバリジャーナルファイル（SRF）**とは，システムジャーナルファイルとは別に，各種のジャーナル情報を，システムサービスごとに取得するファイルです。SRF を使用することで，再開始時のジャーナル読み込みをシステムサービスごとにできます。そのため，再開始時の回復時間を短縮できます。

### (2) サーバリカバリジャーナルファイルの作成方法

SRF は，$DCDIR/spool/dcsjl/ディレクトリの下に，xxx_nn（xxx はシステムサービスの名称，nn は任意の整数）というファイル名で，OpenTP1 で自動的に取得されます。

### (3)　サーバリカバリジャーナルファイルの障害時の回復方法

再開始処理中に SRF に障害が発生した場合には，その旨を伝えるメッセージログが出力されます。このメッセージログが出力された場合は，jnlmkrf コマンドで SRF を回復してください。SRF の回復にはアンロードジャーナルファイルを使用します。

## 4.2.6　アーカイブジャーナルファイル

### (1)　アーカイブジャーナルファイルの目的

クラスタ/並列システム形態で OpenTP1 を使用する場合，システムジャーナルファイルを各ノードにアンロードする必要があります。この負担を軽減するために，各 OpenTP1 ノードのシステムジャーナルファイルを，専用の OpenTP1 ノードにアーカイブできます。アーカイブしたファイルをアンロードするだけで，ほかの OpenTP1 ノードのシステムジャーナルファイルをアンロードする必要がなくなります(**グローバルアーカイブジャーナル機能**)。アーカイブ専用の OpenTP1 ノードで使用するファイルを**アーカイブジャーナルファイル**といいます。グローバルアーカイブジャーナル機能を使用すると，ノードごとにジャーナルをアンロードする必要がなくなります。

グローバルアーカイブジャーナルでは，アーカイブするジャーナルの種類を選択できます。選択するジャーナルの種類は，システムジャーナル定義の jnl_arc_rec_kind に選択するジャーナルレコード種別を指定します。ジャーナルの種類については，「4.2.2　システムジャーナルファイル」を参照してください。

グローバルアーカイブジャーナル機能については，「6.2.3　グローバルアーカイブジャーナル機能」を参照してください。

### (2)　アーカイブジャーナルファイルの種類

アーカイブジャーナルファイルは，**最大 16 種類**を作成できます。複数作成したファイルは，アーカイブするノードの範囲で分けられます。個々のアーカイブジャーナルファイルは，システムジャーナルファイルと同様に，物理ファイルとファイルグループ単位で管理されます。

使用目的別に分けたアーカイブジャーナルファイルのまとまりを**リソースグループ**といいます。グローバルアーカイブジャーナルサービスは，リソースグループ名を使ってアーカイブジャーナルファイルを管理します。一つのリソースグループにアーカイブできるノードは，**20 ノード**までです。

リソースグループは，アーカイブジャーナルサービス定義のファイル名を示します。アーカイブジャーナルノードでリソースグループを幾つ使うかは，グローバルアーカイブジャーナルサービス定義で指定します。

### (3)　アーカイブジャーナルファイルの作成

アーカイブジャーナルファイル作成するときは，jnlinit コマンドで OpenTP1 ファイルシステム上に物理ファイルを作成して，アーカイブジャーナルサービス定義で指定します。

リソースグループを定義するアーカイブジャーナルサービス定義では，ファイルグループ名と要素ファイル名，および物理ファイル名を指定します。

- **物理ファイル名**
  OpenTP1 ファイルの実体を表すパス名
- **要素ファイル名**
  アーカイブジャーナルファイルの並列アクセス機能を使うときに，物理ファイルに対応して任意に付ける名称

要素ファイルは，二重化できます。ファイルグループ名は，一つ以上の要素ファイルのまとまりに対応して任意に付ける名称です。アーカイブジャーナルファイルの並列アクセス機能を使わない（ファイルグループを構成する要素ファイルが一つだけ）場合は，要素ファイル名を省略できます。なお，ファイルグループを構成する要素ファイルの数は，最大８個です。

アーカイブジャーナルファイルを使う前に，ファイルグループごとに OpenTP1 の開始と同時にオープンするかを事前に指定しておきます。また，コマンドの引数にファイルグループ名を指定して，アーカイブジャーナルファイルをファイルグループ単位で操作できます。なお，一つのリソースグループには，二つ以上のファイルグループが必要です。

## (4) アーカイブジャーナルファイルの並列アクセス機能

一つのファイルグループを複数の要素ファイルで構成して，アーカイブジャーナルファイルを使えます。この機能を**アーカイブジャーナルファイルの並列アクセス機能**といいます。

アーカイブジャーナルファイルでは，複数のノードのジャーナルをアーカイブする入出力が集中します。このため，アーカイブジャーナルファイルの入出力がアーカイブするジャーナルの発生に追い付けなくなる場合があります。このような場合に，ファイルグループを複数の**要素ファイル**で構成してアーカイブジャーナルファイルへ並行にアクセスできるようにすると，アーカイブの性能を上げることができます。

一つのファイルグループを構成する物理ファイルは，SCSI インタフェースやハードディスクが重なるようになっていてもかまいませんが，これらには並行にアクセスできないため，並列アクセス機能の効果を十分に発揮できません。SCSI インタフェースやハードディスクは，重ならないようにしてください。また，複数の物理ファイルの容量は，一致していなくてもかまいませんが，一致していない場合 OpenTP1 は少ない容量の方をファイルグループの容量と見なします。資源を効率良く使えるように，容量はできるだけ一致させておいてください。

アーカイブジャーナルファイルの並列アクセス機能では，アーカイブジャーナルファイルの障害などで，並行してアクセスできる要素ファイルの数（以降，並列アクセス数といいます）が減ります。これによって，アーカイブするジャーナルの発生にアーカイブジャーナルファイルの入出力が追い付けなくなることを防ぐため，最低限保証したい並列アクセス数を指定できます。この最低限保証したい並列アクセス数を，**最小分散数**といいます。また，最大でどれだけ並行してアクセスするかを示す値を，**最大分散数**といいます。

アーカイブジャーナルサービス定義でファイルグループを指定するときに，要素ファイル名を付けます。さらに，アーカイブジャーナルサービス定義の jnl_max_file_dispersion オペランドに最大分散数を，jnl_min_file_dispersion オペランドに最小分散数を指定しておきます。

## (5) アーカイブジャーナルファイルの二重化

一つの要素ファイルを二つの物理ファイルから構成して，**アーカイブジャーナルファイルを二重化**して使えます。二重化してファイルグループを運用した場合，OpenTP1 は A 系，B 系に同じジャーナルを取得します。二重化しておくと，障害発生によるジャーナルの読み込み時に片方の系で障害が起こっても，残った系からデータを読み込めるため，信頼性が上がります。

### (a) アーカイブジャーナルファイル定義の指定

アーカイブジャーナルファイルを二重化する場合は，アーカイブジャーナルファイル定義の jnl_dual オプションに Y を指定します。このとき，一つのファイルグループに二つの物理ファイル（A 系，B 系）を指定しておきます。

A 系と B 系の物理ファイルは，同時に障害にならないようにディスクを分けることをお勧めします。また，A 系と B 系の物理ファイルの容量は，一致していなくてもかまいませんが，一致していない場合 OpenTP1 は少ない容量の方をジャーナルの容量と見なします。資源を効率良く使えるように，容量はできるだけ一致させておいてください。

### (b) 片系運転可と片系運転不可

アーカイブジャーナルファイルを二重化した場合には，**片系運転可**または**片系運転不可**のどちらかを選べます。

- **片系運転可**
  要素ファイルを構成する二つの物理ファイルのうち，一つが使用できる状態であれば，要素ファイルを使える状態にします。
- **片系運転不可**
  要素ファイルを構成する二つの物理ファイルが両方とも使える状態のときだけ，要素ファイルを使える状態にします。

片系運転可とするかどうかは，アーカイブジャーナルサービス定義で指定します。

片系運転可では，片方の系が障害になっても，その障害が対策されるまでの間，残りの系で運転を続けます。ただし，この場合，一時的に二重化されない時期が起こるため，この期間は信頼性が下がります。

## (6) アーカイブジャーナルファイルの状態

アーカイブジャーナルファイルの状態は，システムジャーナルファイルと同様に，次の 2 種類に分けられます。

- **使用可能状態**

ファイルグループを構成する要素ファイルのうち，必要な数以上，使用できる要素ファイルがある状態

- **使用不可能状態**
  ファイルグループを構成する要素ファイルのうち，必要な数以上，使用できる要素ファイルがない状態

使用できる要素ファイルとは，その要素ファイルを構成する物理ファイルが必要な数だけオープンされていることを意味します。逆に，要素ファイルを構成する物理ファイルが必要な数だけオープンされていない場合は，その要素ファイルは使用できません。

使用可能状態のファイルグループは，現用または待機として，使用不可能状態のファイルグループは予約として，状態を管理します。なお，要素ファイルの必要数と物理ファイルの必要数は，アーカイブジャーナルサービス定義に指定した値で決まります。

- **現用**
  現時点でジャーナルの出力対象になっている使用可能状態のファイルグループです。この状態のファイルグループは，常に一つです。
- **待機**
  現時点でジャーナルの出力対象になっていませんが，現用に変更するために待機している使用可能状態のファイルグループです。
  待機の状態は，次の二つに分けられます。
  - **次回スワップ先にできる状態**
    上書きできる（回復に必要なジャーナルがない）状態で，かつアンロード済み状態の待機ファイルグループです。
  - **次回スワップ先にできない状態**
    上書きできない状態，またはアンロード待ち状態の待機ファイルグループです。
- **予約**
  使用不可能状態のファイルグループです。

アーカイブジャーナルファイルには，予約以外のファイルグループが二つ以上必要です。

## (7) アーカイブジャーナルファイルのアンロード

アーカイブジャーナルファイルがアンロード待ち状態になった場合は，jnlunlfg コマンドでアンロードしてください。アーカイブジャーナルファイルのファイルグループがアンロードされていない状態でスワップ先がなくなった場合，グローバルアーカイブジャーナルサービスは異常終了します。

アンロードした通常ファイルは，システムジャーナルファイルの場合と同様に，DAM ファイルの回復や稼働統計情報ファイルの編集ができます。また，ユーザジャーナルをオフラインのプログラムへ引き継げます。

グローバルアーカイブアンロードジャーナルファイルには，複数の OpenTP1 ノードのシステムジャーナル情報が入っています。このジャーナルをマージした稼働統計情報の編集，および複数の OpenTP1 ノードのジャーナルを時系列にソートするなど，各種のジャーナル編集ができます。

### (a) アンロードチェックの抑止

通常はアンロード待ち状態になる待機のファイルグループを，現用のまま使える指定ができます。これをアンロードチェックの抑止といいます。アンロードチェックを抑止する場合は，アーカイブジャーナルサービス定義の jnl_unload_check オペランドに N を指定してください。

アンロードチェックを抑止している場合は，OpenTP1 の稼働中にジャーナルファイルをコマンドでアンロードしないでください。アンロードチェックの抑止を指定した場合にアンロードを実行するときは，いったん jnlclsfg コマンドで該当するファイルグループをクローズしてから，アンロードしてください。

アンロードチェックを抑止するとグローバルアーカイブアンロードジャーナルファイルは作成されないため，グローバルアーカイブアンロードジャーナルファイルを入力とするジャーナル編集コマンド（jnlcolc コマンド，jnlstts コマンドなど）は実行できません。

アンロードチェックを抑止した場合は，ファイルグループの状態を jnlls コマンドで確認すると，アンロード待ち状態が出力されることがあります。アンロード待ち状態が出力された場合でも，実際は現用として使われています。

## (8) システムジャーナルファイルの状態

グローバルアーカイブジャーナルサービスを使っている OpenTP1 のシステムジャーナルサービスでは，次に示すジャーナルファイルの状態が追加となります。

- **アーカイブ済み/未アーカイブ**
  アーカイブジャーナルファイルに転送，または出力されたファイルグループかどうかの状態

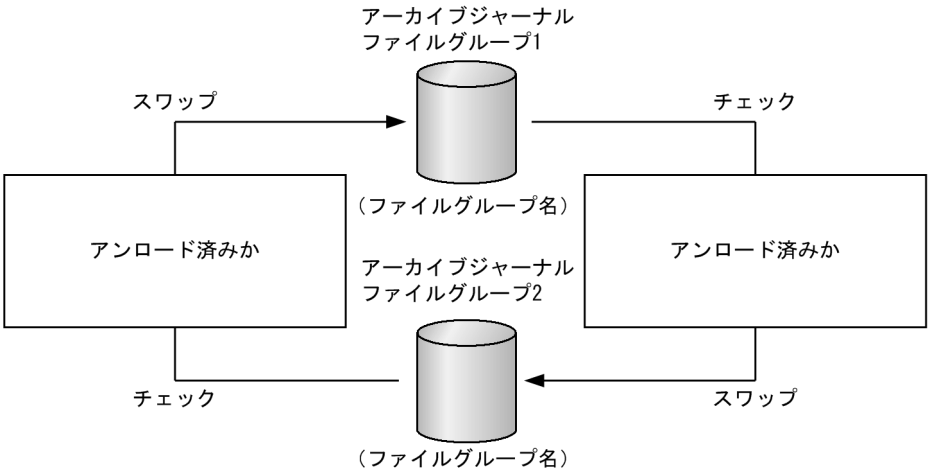
該当するノードでアンロードが済んでいなくても，アーカイブ済みで上書きできる状態のファイルグループであれば，次回のスワップ先にできるファイルグループとなります。

## (9) アーカイブジャーナルファイルのスワップ

待機のアーカイブジャーナルファイルには，スワップ先にできるものとできないものがあります。スワップ先にできるアーカイブジャーナルは，アンロード済みのファイルグループです。アンロードしていないファイルグループは，スワップ先にできません。アーカイブジャーナルファイルのスワップ先がない場合，OpenTP1 は異常終了します。OpenTP1 が異常終了した場合の処置は，「5.3.3　ファイル障害の対策」を参照してください。

アーカイブジャーナルファイルのスワップを次の図に示します。

図 4–12　アーカイブジャーナルファイルのスワップ




## 4.2.7　ノードリストファイル

### (1)　ノードリストファイルの目的

ノード自動追加機能でノードリストを引き継ぐ場合に必要なファイルです。ノードリストを引き継ぐと，オンライン中にノードリストの情報を一定間隔でノードリストファイルへ書き込みます。

次の OpenTP1 の開始時には，ノードリストファイルからノードリストの情報を引き継いで OpenTP1 を開始します。これによりオンライン後すぐに，前回起動時と同等の OpenTP1 システム構成で RPC 要求ができます。

ノードリストファイルがない場合は，ノードリストの整合性が確保されるまでの間，RPC のサービス要求先が制限されます。この制限を解除するために，ノード自動追加機能を使用する場合は，ノードリストファイルを作成し，ノードリストを引き継ぐことをお勧めします。

### (2)　ノードリストファイルに格納する情報

ノードリストファイルには，動作モード，マネジャノード情報，ノードリスト情報などを格納します。nammstr コマンドで変更した動作モードを格納し，次回 OpenTP1 開始時にこれらの情報を引き継げます。

ノードリストファイルに格納する情報について，次の表に示します。

表 4–9　ノードリストファイルに格納する情報

| 情報の種類 | 内容 |
|---|---|
| 動作モード | ノードの動作モード（マネジャノード，またはエージェントノード）を格納します。<br>この情報によって，OpenTP1 起動時の動作モードを決定します。<br>動作モードの取得に失敗した場合は，システム共通定義の name_service_mode の指定に従って，動作モードを決定します。 |

| 情報の種類 | 内容 |
| --- | --- |
| マネジャノード情報 | エージェントノードの場合，マネジャノードに関するノード情報を格納します。<br>マネジャノードの場合は，格納されません。<br>エージェントノードの場合，このマネジャノード情報を基にノードリストの取得要求をします。<br>マネジャノード情報の取得に失敗した場合は，ネームサービス定義の name_start_error オペランドの指定値に従います。 |
| ノードリスト情報 | そのノードがオンライン中に保持しているノード情報の一覧を格納します。 |

## (3) ノードリストファイルの作成

ノードリストファイルは，namnlcre コマンドを実行して作成します。OpenTP1 ファイルシステム上に物理ファイルが作成されます。作成後は，OpenTP1 のオンライン中に一定間隔でノード情報が書き込まれます。

## 4.3　キューを格納するファイル

メッセージ送受信機能およびメッセージキューイング機能で使う OpenTP1 ファイルを，次に示します。

- メッセージキューファイル
  メッセージ送受信機能（TP1/Message Control を使った通信）で使います。
- MQA キューファイル
  メッセージキューイング機能（TP1/Message Queue を使った通信）で使います。

### 4.3.1　メッセージキューファイル

#### (1)　メッセージキューファイルの目的

**メッセージキューファイル**は，MCF を使って相手システムとメッセージを送受信するときに使用する，OpenTP1 ファイルシステム上のファイルです。入出力メッセージ用のディスクキューとして使用します。

入力キューは，入力メッセージを管理する待ち行列です。出力キューは，出力メッセージを管理する待ち行列です。

#### (2)　メッセージキューファイルの構成

OpenTP1 は，メッセージキューファイルを，**キューグループ**という論理的な単位で運用します。これに対して，実際に入出力メッセージを取得するファイル実体を**物理ファイル**といいます。一つのキューグループは，一つの物理ファイルで構成します。メッセージキューファイルは，**入力キュー用**と**出力キュー用**に 2 種類作成します。ネットワークコミュニケーション定義の入出力キュー定義で，キューグループを入力キュー用と出力キュー用に対応づけます。また，入力キュー用と出力キュー用それぞれにメッセージキューサービス定義を作成し，キューグループと物理ファイルの対応関係を指定します。

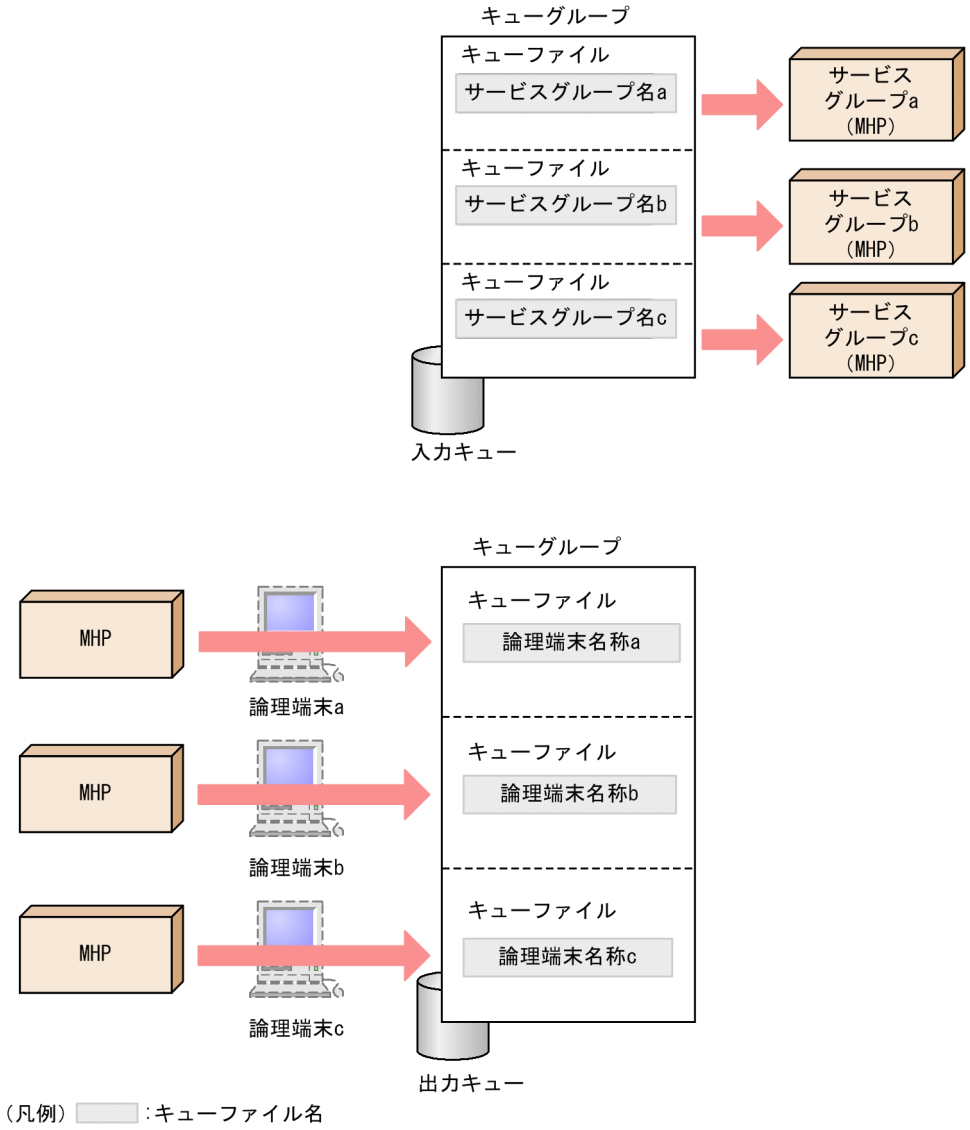
ユーザは，メッセージキューサービス定義でキューグループに任意の名称（ID）を指定します。この**キューグループ ID** を使用して，物理ファイルの使用率を監視するなど，メッセージキューファイルをキューグループ単位で運用します。

一つのキューグループをサービスグループ，または論理端末対応に分割して利用します。キューグループをサービスグループ，または論理端末ごとに分割した単位を，**キューファイル**といいます。対応するサービスグループ名，または論理端末名称が**キューファイル名**になります。

入力メッセージはサービスグループごと，出力メッセージは論理端末ごとにスケジュールされます。キューグループをサービスグループごと，論理端末ごとに分割して使用することで，同一資源にアクセスする場合の性能が向上します。

キューグループとキューファイルの関係を，次の図に示します。

図 4-13　キューグループとキューファイルの関係




メッセージキューファイルを使用したメッセージ送受信については，「3.3.9　アプリケーションプログラムのメッセージ送受信」を参照してください。

## 4.3.2　MQA キューファイル

### (1)　MQA キューファイルの目的

MQA キューファイルとは，メッセージキューイング機能（TP1/Message Queue）を使う場合にキューを格納するファイルです。メッセージキューイング機能のキューとは，UAP から登録されたメッセージを格納する待ち行列です。

MQA キューファイルに登録したメッセージは，キューマネジャ（TP1/Message Queue）が通信相手に送信します。MQA サービス定義，および MQI の設定によって，必要なメッセージを受信した順序に関係

なく取り出すこともできます。さらに，MQA キューファイルにメッセージを格納して，データ処理をほかの UAP に引き継ぐこともできます。

MQA キューファイルの構成，キューの種類，および操作方法については，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

## 4.4　ユーザデータを格納するファイル

OpenTP1 の業務処理で使用する，ユーザデータを格納するファイル（ユーザファイル）について説明します。OpenTP1 で使用できるユーザファイルには，次の 3 種類があります。

- DAM ファイル
  OpenTP1 専用の直接編成ファイルとして使用します。
- TAM ファイル
  テーブルアクセス法で高速にアクセスできる，OpenTP1 専用の直接編成ファイルとして使用します。
- ISAM ファイル
  X/Open の ISAM モデルに準拠した索引順編成ファイルとして使用します。ISAM ファイルについては，マニュアル「索引順編成ファイル管理 ISAM」を参照してください。

OpenTP1 のユーザファイルは，すべて **OpenTP1 ファイル**上に作成します。

上記のほかに，**共用メモリ上のテーブルにアクセスする機能（IST サービス）**や，**データベースマネジメントシステム（DBMS）**で管理するユーザファイルも，OpenTP1 で使えます。

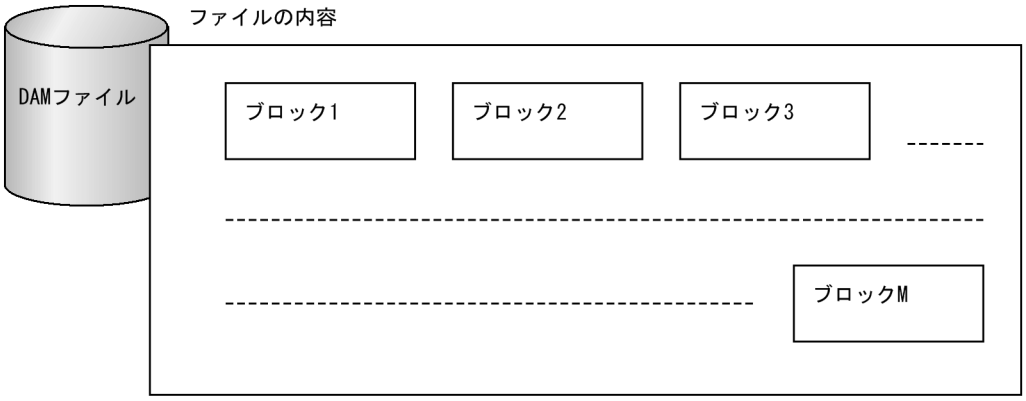
### 4.4.1　DAM ファイル（TP1/FS/Direct Access）

**DAM ファイルサービス**を使うと，OpenTP1 ファイルシステム上の OpenTP1 ファイルに直接編成ファイルとしてアクセスでき，トランザクションと同期して OpenTP1 ファイルの状態を管理できます。OpenTP1 で DAM ファイルを使う場合には，**TP1/FS/Direct Access** が必要です。

DAM ファイルは，DAM サービス定義で，**物理ファイル名**と**論理ファイル名**を指定します。物理ファイル名とは，OpenTP1 ファイルを示す完全パス名のことです。論理ファイル名とは，物理ファイルと対応した論理的な名称です。論理ファイル名は，UAP から API で DAM ファイルにアクセスするときや，コマンドで物理ファイル名と論理ファイル名の対応を変更するときに使います。論理ファイル名を使うことで，物理的な構造を意識しないで DAM ファイルへアクセスできます。論理ファイル名と物理ファイル名は，**1 対 1** で対応づけます。

DAM ファイルの構造を次の図に示します。

図 4-14　DAM ファイルの構造




## (1)　DAM ファイルの特長

DAM ファイルの特長を次の表に示します。

表 4-10　DAM ファイルの特長

| 説明する内容 | 特長 |
|---|---|
| ファイル構成とレコード形式 | DAM ファイルは，直接編成ファイルです。事前に OpenTP1 ファイルとして割り当てたファイルを使います。<br>レコード形式は固定長の非ブロック形式です。 |
| アクセス形態 | 相対ブロック番号を使った直接アクセス法で，ブロックを参照または更新します。<br>連続した複数のブロックを一括して入出力することもできます。 |
| トランザクションとの関係 | トランザクション処理中に更新したブロックの内容は，次に示す場合にファイルに出力されます。<br>• トランザクションがコミットしたとき<br>• DAM サービス定義に指定した最大ブロック数に達したとき<br>トランザクションがロールバックした場合は，トランザクションを開始する以前の状態に回復されます。 |
| ファイルのバックアップとリストア | DAM ファイルのバックアップとリストアには，次のコマンドを使います。<br>• dambkup コマンド<br>• damrstr コマンド<br>バックアップファイルとバックアップ取得時点以降の回復用ジャーナルを基に DAM ファイルの内容を最新の状態に回復できます。DAM ファイルを回復する機能を DAM FRC といいます。 |
| オンライン中のファイルの追加 | オンライン中にファイルを追加する場合は，追加する物理ファイルを作成してから，damadd コマンドを使います。 |
| オンライン中のファイルの削除 | オンラインで使っているファイルを切り離す場合は damrm コマンドを，切り離した DAM ファイルの物理ファイルを削除する場合は damdel コマンドを使います。<br>オンライン中にファイルを削除する場合は，damrm コマンドで論理ファイルを切り離してから，物理ファイルを damdel コマンドで削除します。 |

## (2) DAM ファイルの作成方法

OpenTP1 ファイルシステム上に割り当てられた OpenTP1 ファイルを**物理ファイル**といいます。オフラインでの DAM ファイルへのアクセスは物理ファイルに対して行います。オンライン中の DAM ファイルへのアクセスは，物理ファイルと 1 対 1 に対応した**論理ファイル**に対して行います。

OpenTP1 ファイルシステムを作成したあと，DAM ファイルを作成します。作成方法には，OpenTP1 のコマンドによる方法と，OpenTP1 で提供する関数による方法があります。

1. コマンドを使った DAM ファイルの作成

   damload コマンドで，物理ファイルを割り当てたあと，物理ファイルへ初期データを出力します。

2. UAP を使った DAM ファイルの作成

   オフライン環境で dc_dam_create 関数（ファイルの割り当て，初期データの出力）を UAP から呼び出して，DAM ファイルを作成できます。

## (3) DAM ファイルの入出力関数

OpenTP1 の UAP では，DAM ファイルにアクセスする関数（dc_dam_××××関数）を使用できます。UAP で使用できる DAM ファイルサービスについては，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## (4) DAM ファイルの排他制御

### (a) 排他の単位

DAM ファイルの排他には，ファイル単位で排他をかける**ファイル排他**と，ブロック単位で排他をかける**ブロック排他**があります。DAM ファイルにファイル排他をかけるか，ブロック排他をかけるかを，DAM ファイルをオープンするときに関数（dc_dam_open 関数）の引数に指定します。通常は，ブロック単位で排他します。実際にファイルやブロックに排他をかけるのは，オープンした DAM ファイルのブロックを，更新したり，更新するために参照しようとしたときです。

DAM ファイルをオープンしようとしたとき，そのファイルがすでにほかのトランザクションからファイル排他をかけられていた場合には，DAM ファイルをオープンできないで，エラーリターンします。

### (b) トランザクションと排他の単位

DAM ファイルにアクセスするときは，**トランザクションブランチ単位**で排他制御します。また，**グローバルトランザクション単位**で排他制御することもできます。グローバルトランザクション単位で排他制御すると，同一グローバルトランザクションに属する複数のトランザクションブランチから同一ブロック，または同一ファイルにアクセスしても，排他エラーになりません。排他制御の単位は，DAM サービス定義の指定で変更できます。

グローバルトランザクション単位で排他制御する場合，トランザクションブランチごとに DAM ファイルへアクセスしても，並列にアクセスできないで順次アクセスとなります。そのため，トランザクションの

性能が下がる場合があります。トランザクションブランチごとの DAM ファイルへのアクセスを並列に処理させる場合は，トランザクションブランチ単位で排他制御する指定をしてください。

排他制御の単位については，マニュアル「OpenTP1 プログラム作成の手引」および「OpenTP1 システム定義」を参照してください。

### (c) 資源の排他解除待ちの設定

オープンした DAM ファイルのブロックを，更新したり，更新するために参照したりしようとしたとき，そのブロックがすでに排他をかけられていた場合に，排他が解放されるのを待つかどうかを，関数のパラメタで指定します。排他解除を待つ時間（排他待ち限界経過時間）を，ロックサービス定義で指定します。この排他待ち限界経過時間が経過しても排他が解放されない場合には，エラーリターンします。

## (5) DAM ファイルのディファード更新

トランザクションで更新した DAM ファイルのブロックを，同期点の取得とは非同期に更新できます。この機能を**ディファード更新**といいます。ディファード更新指定の DAM ファイルは，トランザクションの同期点取得時には更新されないで，一定間隔で起動される出力専用プロセスから実更新されます。DAM ファイルにディファード更新をする指定をしておくと，トランザクションの入出力回数を削減できるので，単位時間当たりのトランザクションのスループットを向上できます。

ディファード更新指定の DAM ファイルと通常更新の DAM ファイルを，一つのトランザクションから更新出力した場合には，**どちらの DAM ファイルも同期点で実更新**します。

### (a) ディファード更新する DAM ファイルの指定方法

DAM ファイルをディファード更新するかどうかは，DAM ファイルごとに，DAM サービス定義で指定します。

出力専用プロセスが実行されるまでの間，ブロックを退避しておくバッファ領域と，出力専用プロセスから更新を実行する時間間隔を指定します。この実行間隔時間が長過ぎたり，バッファ領域の値が小さ過ぎる場合は，バッファ領域のオーバフローや，領域の空き待ち時間が発生する場合があります。ディファード更新の指定では，DAM サービス定義に指定するバッファ領域，および時間間隔の値の算出に注意してください。

### (b) ディファード更新指定の DAM ファイルを使用するときの注意

システム障害による OpenTP1 の異常終了が発生した場合，ディファード更新指定の DAM ファイルでは，障害前に完了していたトランザクションのアクセスが DAM ファイル上に反映されていないことがあります。この場合，トランザクションと DAM ファイルの更新内容を保証するために，OpenTP1 の再開始処理（全面回復処理）をしたあと，必ず正常に終了させてください。

また，障害以前に完了したトランザクションでも，出力専用プロセスの処理が障害前に完了していないと，全面回復後にロールバックとなることがあります。障害以前に完了済みのトランザクションをロールバックさせたくない場合には，アクセスする DAM ファイルにディファード更新を指定しないでください。

ディファード更新指定の DAM ファイルにディスク障害が発生した場合は，ファイル回復機能（DAM FRC）を必ず実行してください。

## (6) DAM ファイルのオンラインバックアップ

DAM ファイルのバックアップを，OpenTP1 のオンライン中に取得できます。これを**オンラインバックアップ**といいます。DAM ファイルのバックアップを取得するコマンド（dambkup コマンド）に-o オプション指定して実行すると，オンライン中に DAM ファイルをバックアップできます。

オンラインバックアップしたファイルで DAM ファイルを回復すると，回復に使用するアンロードジャーナルファイルの量が少なくて済みます。そのため，-o オプションを指定しないでバックアップしたファイルを使用して DAM ファイルを回復する場合と比べて，DAM ファイルの回復処理に掛かる時間が少なくて済みます。

なお，DAM ファイルの回復処理は，対象となる DAM ファイルを UAP が使用していると実行できません。その場合，対象となる DAM ファイルをオープンしている UAP をいったん停止するか，または OpenTP1 を停止してください。

-o オプションを指定しない場合は，オフライン状態でバックアップすることになります。この場合，次の手順でバックアップしてください。

1. damhold コマンドを実行して，論理ファイルを論理閉塞します。
2. damrm コマンドを実行して，論理閉塞した論理ファイルをオンラインから切り離します。
3. -o オプションを指定しない dambkup コマンドを実行して，DAM ファイルをバックアップします。

## (7) ユーザデータの抽出

DAM ファイルの管理情報を除いたユーザデータだけを抽出できます。抽出したユーザデータを DAM ファイルの初期データとして，damload コマンドで割り当てることができます。ただし，抽出したユーザデータを damrstr コマンドでリストアすることはできません。ユーザデータを抽出するには，dambkup コマンドに-d オプションを指定して実行します。

## (8) ブロック長拡張機能

dambkup コマンドでバックアップしたファイルをリストアするとき，ブロック長を拡張してリストアできます。これを**ブロック長拡張機能**といいます。ブロック長拡張機能を使うことで，ブロック長の異なる DAM ファイルにデータを移行できます。

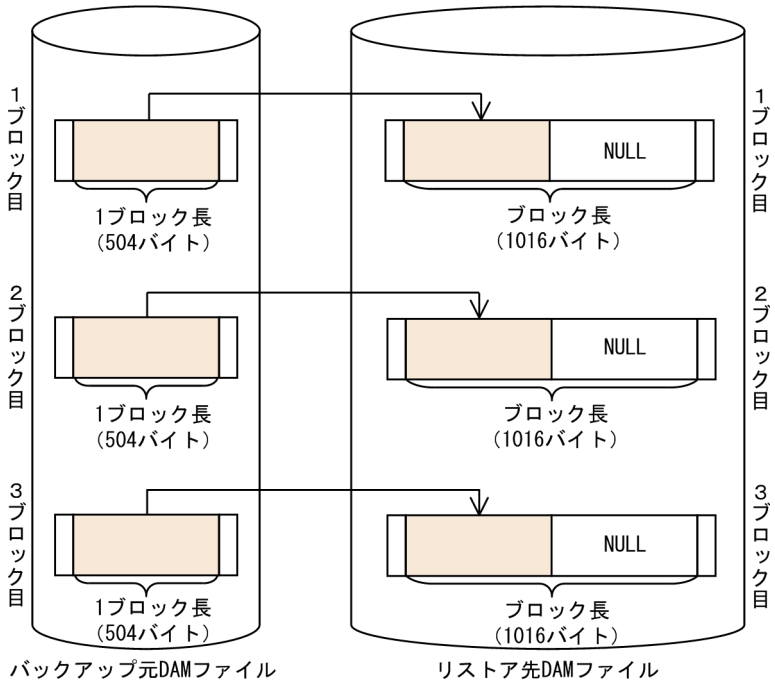
ブロック長拡張機能の方式には，バックアップ元 DAM ファイルのブロック構成を維持する方式とブロック構成を維持しない方式があります。各方式について次に示します。

- ブロック構成を維持する方式

  バックアップ元 DAM ファイルのブロック構成を，リストア先 DAM ファイルでも維持するブロック長拡張方式です。拡張後のブロック長は，damrstr コマンドの-e オプションで指定します。

1 ブロック長 504 バイトの DAM ファイルを，1 ブロック長 1016 バイトの DAM ファイルにリストアする例を，次の図に示します。

図 4–15　ブロック構成を維持するブロック長拡張方式

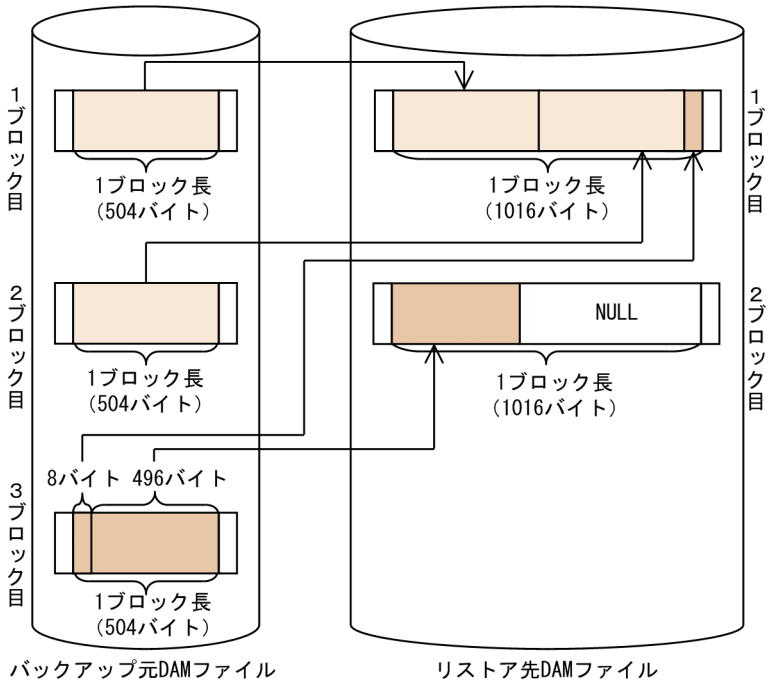



- ブロック構成を維持しない方式

  バックアップ元 DAM ファイルのデータを，リストア先 DAM ファイルの先頭ブロックから詰めて格納する方式です。拡張後のブロック長は，damrstr コマンドの-p オプションで指定します。

  1 ブロック長 504 バイトの DAM ファイルを，1 ブロック長 1016 バイトの DAM ファイルにリストアする例を，次の図に示します。

図 4–16　ブロック構成を維持しないブロック長拡張方式



ブロック長拡張機能を使用する場合は，次の点に注意してください。

- damfrc コマンドで指定する回復対象定義ファイルに，ブロック長を拡張した DAM ファイルを指定できません。
- オンライン中に取得したバックアップファイルを使用して，ブロック長を拡張できません。

## (9) 標準入出力を使った DAM ファイルのバックアップとリストア

DAM ファイルのバックアップ先に標準出力を，リストアの入力ファイルに標準入力を使えます。標準入出力を使うことで，コマンドのリダイレクトができるようになります。

標準入出力を使う場合には，dambkup コマンドおよび damrstr コマンドに-s オプションを指定して実行します。

## (10) 回復対象外の DAM ファイルへのアクセス

トランザクションによる整合性の管理や障害時の回復を保証しない DAM ファイルを作成できます。このような DAM ファイルを**回復対象外の DAM ファイル**といいます。回復対象外の DAM ファイルへは，トランザクションでない処理からブロックを更新したり出力したりできます。

回復対象外にする DAM ファイルには，DAM サービス定義の定義コマンド damfile に-n オプションを付けて指定します。

回復対象外の DAM ファイルへのアクセスでエラーが起こった場合，ファイルデータの障害は回復できません。

## (11) キャッシュブロックの処理

DAM サービスは一度読み込んだ DAM ファイルのブロックデータを DAM サービス専用共用メモリにためます。同じブロックに対して参照要求がある場合，DAM サービス専用共用メモリ上にあるブロックデータを UAP へ返すことによって，ファイル I/O の実行回数を削減しています。

DAM サービス専用共用メモリ内では，DAM ファイルごとにブロックデータをチェイン管理しています。このブロックデータが格納されている DAM サービス専用共用メモリ中の領域を**キャッシュブロック**といいます。

UAP が DAM ファイルにアクセスするとき，DAM サービスは次に示す処理順序でキャッシュブロックを確保します。この処理を**キャッシュブロック確保処理**といいます。

キャッシュブロック確保処理の流れを次に示します。

1. アクセスされた DAM ファイルのキャッシュブロックチェインに，該当するブロックに対応するキャッシュブロックがつながれているか検索します。つながれている場合は，そのキャッシュブロックのデータを UAP へリターンします。

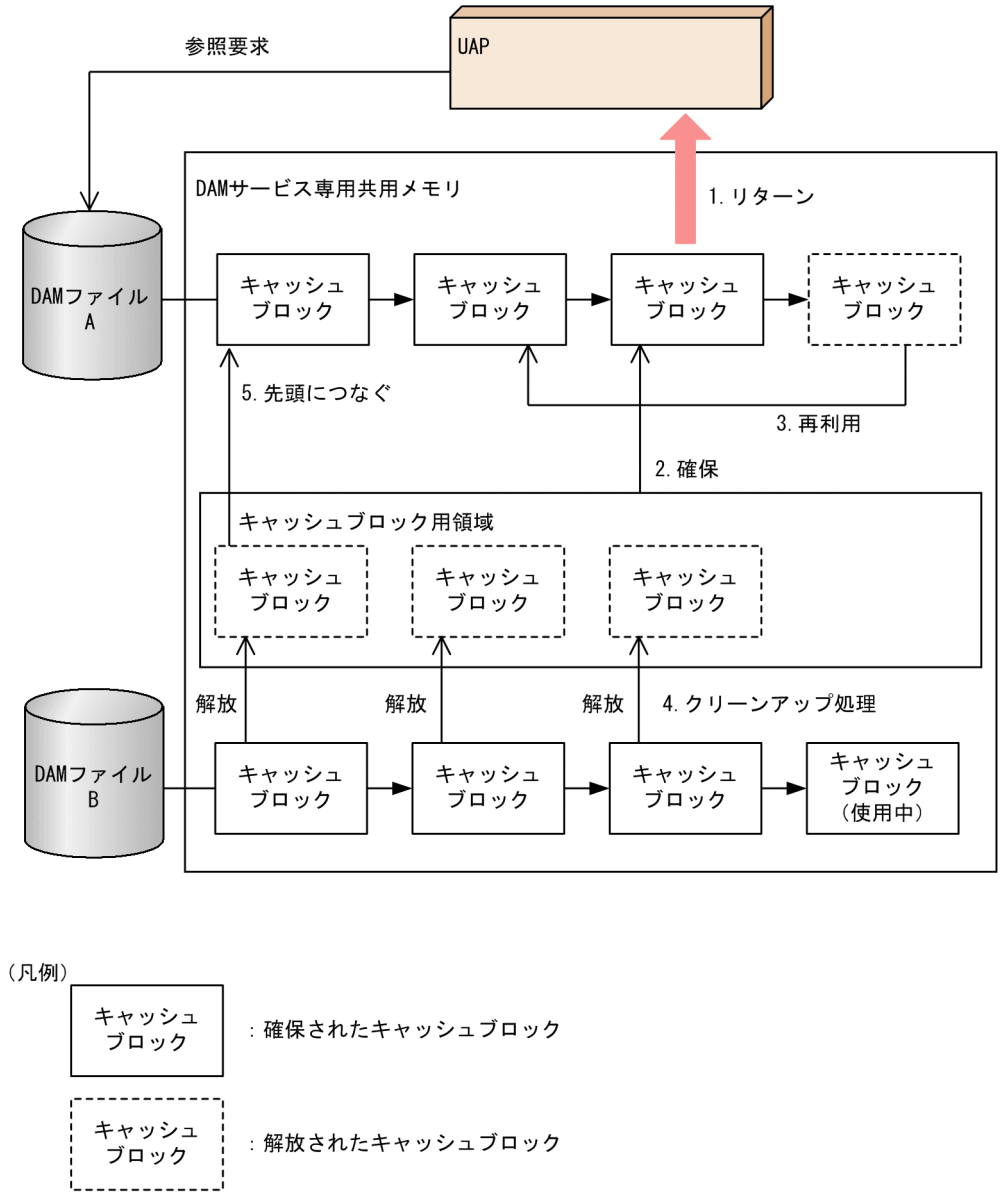
2. キャッシュブロックチェインにつながれていない場合は，キャッシュブロック用領域からキャッシュブロックを確保します。
3. キャッシュブロック用領域に空き領域がない場合は，アクセスした DAM ファイルのキャッシュブロックチェインの中で，実行中トランザクションで参照されていないキャッシュブロックを再利用します。このとき，キャッシュブロックチェインの末尾，つまり最も古くキャッシュブロックチェインにつながれたキャッシュブロックから再利用します。なお，再利用するキャッシュブロックは，DAM サービス定義の dam_cache_reuse_from オペランド指定値によって決まります。デフォルトは，末尾から再利用します。
4. 2.および 3.でキャッシュブロックを確保できない場合は，ほかの DAM ファイルのキャッシュブロックチェインの中で，実行中トランザクションで使用されていないすべてのキャッシュブロックをいったん解放し，再度，キャッシュブロック用領域からキャッシュブロックを確保します。ここでキャッシュブロックを解放する処理を**クリーンアップ処理**といいます。
5. 確保したキャッシュブロックに DAM ファイルのブロックデータをコピーし，キャッシュブロックチェインの先頭につなぎます。

キャッシュブロックチェインにつながれているキャッシュブロック数や，DAM サービス専用共用メモリの全体使用率は，damchinf コマンドで取得できます。

キャッシュブロックチェインの概要を次の図に示します。

図中の番号は上記の番号に対応します。

図 4-17　キャッシュブロックチェインの概要




## (12)　キャッシュブロック数しきい値指定機能

特定の DAM ファイルに対してアクセスが集中する場合，その DAM ファイルが管理するキャッシュブロック数は増大します。その結果，「4.4.1(12)(b)　適用例」に示すとおり，しきい値を適切に指定していないと，該当するキャッシュブロックの検索や解放に時間が掛かるようになり，トランザクション性能が大幅に低下するおそれがあります。

DAM サービス定義の damchlmt 定義コマンドを指定すると，一つの DAM ファイルで管理するキャッシュブロック数に上限を設けることができます。この上限となる値をしきい値といいます。しきい値に達するまではキャッシュブロックが確保され，しきい値に達すると，新たにキャッシュブロックを確保しないで，該当する DAM ファイルの未使用状態のキャッシュブロックが再利用されます。しきい値を指定しない場合は，共用メモリ資源がなくなるまで，キャッシュブロックが確保されます。

DAM サービス定義の damchlmt 定義コマンドの指定値は，オンライン中に damchdef コマンドを使用して動的に変更できます。

### (a) キャッシュブロックの計算式

キャッシュブロック用に使用できる領域のサイズは次に示す計算式で求めます。

```
キャッシュブロック用領域サイズ
＝ M-（M / 576）×34　（単位：バイト）
```

個々のキャッシュブロックのサイズは次に示す計算式で求めます。

```
キャッシュブロックサイズ
＝　（Ab ＋ 8）＋ 64　（単位：バイト）
```

（凡例）

M：DAM サービス専用共用メモリサイズ

Ab：アクセスする DAM ファイルのブロック長

### (b) 適用例

この機能は，特定の DAM ファイルに集中してアクセスするような形態で効果があります。適用例の前提となる条件を次に示します。

- DAM サービスが確保するキャッシュメモリには，100000 個のキャッシュブロックを確保できます。
- オンライン開始後，damfileA だけを集中的にアクセスします。
- damfileA のブロック数は 100000 ブロックです。
- キャッシュブロック一つの検索時間を 0.1 ミリ秒※とします。
- 1 ブロックのファイル I/O 時間を 500 ミリ秒※とします。

注※

ハードウェアやプロセスの状態によって異なります。説明のため，単純な値を使用しています。

- 適用例 1：キャッシュブロック数しきい値を指定しない場合

  キャッシュブロック数しきい値を指定しない場合，damfileA のキャッシュブロックチェインには，キャッシュブロック用領域に空き領域がある範囲で，キャッシュブロックがつながれます。ここでは，100000 個のキャッシュブロックが damfileA のキャッシュブロックチェインにつながれます。

  この状態で，キャッシュブロックチェインの末尾にあるブロックデータを検出するのに必要なチェイン走査時間は 0.1 ミリ秒×100000 ブロック＝ 10 秒となり，直接ファイル I/O した方が相対的なトランザクション性能は良くなります。また，キャッシュメモリ内に確保できる全キャッシュブロックを damfileA が占有していることから，ほかの DAM ファイルへアクセスする場合，damfileA に対してクリーンアップ処理が実行されます。クリーンアップ処理が実行されると，100000 個のキャッシュブロックの解放処理を実行することから，さらにトランザクション性能を低下させてしまいます。

- 適用例 2：キャッシュブロック数しきい値として 4000 を指定する場合

  キャッシュブロック数しきい値に 4000 を指定すると，damfileA のキャッシュブロックチェインには，最大 4000 個のキャッシュブロックがつながれます（DAM サービス定義の各指定値によっては，4000 個以上のキャッシュブロックがつながれることがあります）。

  この状態で，キャッシュブロックチェインの末尾にあるブロックデータを検出するのに必要なチェイン走査時間は，0.1 ミリ秒×4000 ブロック=400 ミリ秒となり，直接ファイル I/O した場合よりもトランザクション性能が良くなります。また，4000 個のキャッシュブロックチェインになるため，残り 96000 個のキャッシュブロックは未確保状態となり，ほかの DAM ファイルへアクセスした場合にクリーンアップ処理が実行されません。

  ただし，各 DAM ファイルのキャッシュブロックチェインにつながれるキャッシュブロック数に制限があることから，キャッシュブロック用領域に対するキャッシュブロック数が減少し，DAM サービス専用共用メモリ内に使われない領域が発生することがあります。

- 適用例 3：キャッシュブロック数しきい値として 0 を指定する場合

  キャッシュブロック数しきい値に 0 を指定する場合，damfileA のキャッシュブロックチェインにつながれるキャッシュブロック数は，トランザクションブランチ内でアクセスするブロック数，および DAM サービス定義の設定値に依存します。

  ここでは DAM サービス定義の設定値を次に示すとおりとします。

  ```
  set dam_update_block = 10
  set dam_tran_process_count = 5
  ```

  - すべてのトランザクションブランチで damfileA だけをアクセスする場合

    アクセスするブロックに重複がないとすると，10×5=50 個のキャッシュブロックが damfileA のキャッシュブロックチェインにつながれます。これは，トランザクション実行中にほかのプロセスから，最新のブロックデータおよびトランザクション決着前の更新後データを参照されないようにするため，キャッシュメモリ中にデータをスタックしているからです。

  - 一つのトランザクションブランチだけが damfileA をアクセスする場合

    アクセスするブロックに重複がないとすると，上記の場合同様に，キャッシュメモリ中にスタックしておくために，10 個のキャッシュブロックが damfileA のキャッシュブロックチェインにつながれます。

  上記のとおり，キャッシュブロック数しきい値として 0 を指定する場合では，キャッシュブロックチェインにつながれるキャッシュブロック数は，UAP でのアクセス形態や DAM サービス定義の設定値に依存します。この現象は，キャッシュブロック数しきい値に小さな値を指定する場合も発生します。小さな値とは，UAP で実行するトランザクション内でアクセスするブロック数よりも小さな値を意味します。

  このときも，適用例 2 の場合と同様に DAM サービス専用共用メモリ内に使われない領域が発生します。

### (c) UAP のアクセス形態に応じた設定

damchinf コマンドを実行した結果，次に示す情報が出力されたとします。

```
CleanUP Count:1  Next CleanUP FILE-No:1  Using Rate:80%
FileNo FileName BlkLen BlkNum CchBlkNum PreservNum LimitNum ReUse
     1 damfile1    504  10000      7900          0       -1 Exist
     0 damfile0    504  10000       100          0       -1 Exist
     2 damfile2    504  10000         0          0       -1  None
```

UAP のアクセス形態ごとに，この状況でキャッシュブロック数しきい値にどのような値を指定するとよいか説明します。

- damfile1 には，以降ほとんどアクセスしない場合

  "Using Rate:80%"と出力されているので，damfile0 および damfile2 に割り当てることのできるキャッシュブロック領域は 20%しかありません。damfile1 には以降ほとんどアクセスしないので，damfile1 のキャッシュブロックチェインにつながれている 7900 個分のキャッシュブロック用領域は，確保されたままの状態になります。そこで，damfile1 のしきい値に小さな値を指定することによって，確保されたままの状態のキャッシュブロック用領域を小さくし，damfile0 および damfile2 に割り当てるキャッシュブロック領域を大きくできます。

  また，damfile1 のキャッシュブロックチェインにキャッシュブロックが多数つながれているため，キャッシュブロック用領域が不足するとクリーンアップ処理が実行されます。ここでは"Next CleanUP FILE-No:1"と出力されていることから，damfile1 のキャッシュブロックチェインがクリーンアップ対象となります。7900 個のキャッシュブロック解放処理が実行されると，急激に性能が劣化します。この現象を抑止するためにも，damfile1 にしきい値を指定することが重要になります。

- damfile1 に頻繁にアクセスする場合

  damfile1 に頻繁にアクセスすると，damfile1 のキャッシュブロックチェインの検索処理が頻繁に実行されます。この場合，しきい値を指定することで，キャッシュブロックチェインを短くでき，その結果，チェイン検索時間が短縮されます。ただし，しきい値が小さ過ぎる場合，キャッシュ効率が悪化するため，かえって性能が劣化します。しきい値設定値を変更しながら最適な値を決定してください。

- すべての DAM ファイルに平均的にアクセスする場合

  すべての DAM ファイルに平均的にアクセスするため，各 DAM ファイルに割り当てられるキャッシュブロック用領域は平均化していることが望まれます。各 DAM ファイルのしきい値に同等の値を設定すると，各 DAM ファイルが使用するキャッシュブロック用領域は平均化されます。

### (d) 注意事項

この機能を使用するときの注意事項を次に示します。

- キャッシュブロック数しきい値を指定しない場合は，DAM サービス専用共用メモリが満杯になるまでキャッシュブロックが確保されます。しかし，しきい値を指定すると途中でキャッシュブロック確保処理は抑止されます。そのため，使用されない共用メモリ領域が発生します。
- 各 DAM ファイルのキャッシュブロックチェインにつながれるキャッシュブロック数は，しきい値に設定した値を超えることがあります。
- DAM サービス定義の damchlmt 定義コマンドに指定範囲外のしきい値を指定した場合，不正な指定のある個所が KFCA00219-E メッセージに出力されます。また，省略値の採用が KFCA01644-I メッ

セージに出力されます。この場合，該当するDAMファイルで管理するしきい値は設定されないで，共用メモリ資源のある限りキャッシュブロックが確保されます。

## (13)　キャッシュブロック再利用時の検索順序指定機能

DAMサービス定義のdam_cache_reuse_fromオペランドを指定すると，DAMファイルをアクセスするトランザクションで新しくキャッシュブロックが必要となった場合に，キャッシュブロックチェインのどのキャッシュブロックから優先的に再利用対象として検索するか指定できます。

lastを指定すると最も古くキャッシュブロックチェインにつながれたキャッシュブロックから再利用します。

firstを指定すると最も新しくキャッシュブロックチェインにつながれたキャッシュブロックから再利用します。

デフォルトはlastです。

## (14)　キャッシュブロック再利用境界値指定機能

DAMサービス定義のdamcache定義コマンドには，キャッシュブロック再利用境界値を指定できます。指定すると，キャッシュブロック確保処理で新しくキャッシュブロックを確保するときに，キャッシュブロック再利用境界値にキャッシュブロック数が達していない間は，アクセスするDAMファイルのキャッシュブロックではなく，ほかのDAMファイルのキャッシュブロックが優先して再利用されます。

なお，キャッシュブロック数がキャッシュブロック再利用境界値を超えている場合は，アクセスするDAMファイルのキャッシュブロックが再利用されます。したがって，キャッシュブロック再利用境界値に0を指定すると，通常の処理と同じように，アクセスするDAMファイルのキャッシュブロックから再利用されます。

DAMサービス定義のdam_default_cache_numオペランドには，damcache定義コマンドを指定しない論理ファイルのキャッシュブロック再利用境界値を指定できます。

この機能は，あるDAMファイルの多数ブロックを集中してアクセスしたあと，別のDAMファイルを頻繁にアクセスする場合，つまり少数のDAMファイルがキャッシュブロックを占有してしまい，ほかの頻繁にアクセスするDAMファイルに割り当てられるキャッシュブロックが少なくなってしまう場合に有効です。

これに対して，複数のDAMファイルをランダムにアクセスする場合，つまり各DAMファイルにキャッシュブロックが均等に使用されている場合には，キャッシュブロック再利用境界値に小さな値を指定するとキャッシュブロックの解放処理が頻繁に実行され，メモリ確保およびファイルからのデータ読み込み処理が従来よりも増加することによって，性能が劣化することがあります。

### (a)　性能向上が見込まれる例

説明のため，使用するDAMファイルは2ファイルとし，同一ブロック長であるとします。また，システム全体で確保可能な最大キャッシュブロック数を100個とし，DAMファイルAを98ブロック参照したあと，DAMファイルBのアクセスが実行されるとします。

DAM ファイル B のアクセスは，次に示すとおり回数が増えるたびに 1 ブロックずつ参照するブロックが増加し，50 ブロックまで参照するとします。

1 回目：ブロック 1 および 2 を参照

2 回目：ブロック 1，2，および 3 を参照

3 回目：ブロック 1，2，3，および 4 を参照

:

49 回目：ブロック 1，2，3，…および 50 を参照

この条件のとき，damcache 定義コマンドを指定しない場合と，指定する場合の処理の違いを次に示します。

- damcache 定義コマンドを指定しない場合

  1. DAM ファイル B を最初にアクセスするとき，共用メモリからキャッシュブロック用領域を確保します。このとき，DAM ファイル A で 98 個のキャッシュブロックが確保されているため，二つだけ確保できます。そのキャッシュブロックに，ブロック 1 および 2 のデータを読み込み，DAM ファイル B のキャッシュブロックチェインにつなぎます。
  2. ブロック 1，2，および 3 を参照するとき，ブロック 1 および 2 はキャッシュヒットするため，ファイル I/O が実行されません。しかし，ブロック 3 は初めてのアクセスであるため，キャッシュブロックを確保し，データを読み込もうとしますが，1.の段階で共用メモリ中の空き領域がなくなっているため，DAM ファイル B のキャッシュブロックの再利用処理が実行されます。再利用処理では，ブロック 1 のデータが設定されているキャッシュブロックをキャッシュブロックチェインから切り離し，ブロック 3 のデータを読み込み，キャッシュブロックチェインの先頭につなぎます。
  3. ブロック 1，2，3，および 4 を参照するとき，2.と同様にキャッシュブロックの再利用処理が実行され，ブロック 2 のキャッシュブロックを解放し，ブロック 1 を読み込み，キャッシュブロックチェインの先頭につなぎます。さらに，ブロック 2 のデータを読み込むため，ブロック 3 のキャッシュブロックを再利用し，ブロック 2 を読み込みます。この処理をブロック 3 および 4 のときにも繰り返します。
  4. 3.を繰り返すと，DAM ファイル B のキャッシュブロックの再利用処理，つまり，チェインからの解放，およびデータ読み込みのためのファイル I/O が頻繁に実行されます。チェインからの解放処理，およびファイル I/O の回数は，約 1275 回ずつ実行されます。

- damcache 定義コマンドを指定する場合

  DAM ファイル A のキャッシュブロック再利用境界値を 50 とし，DAM ファイル B のキャッシュブロック再利用境界値を 90 とします。

  1. damcache 定義コマンドを指定しない場合の 1.と同様に，余っているキャッシュブロックを二つ確保し，ブロック 1 および 2 のデータを読み込み，キャッシュブロックチェインにつなぎます。
  2. ブロック 1，2，および 3 を参照するとき，ブロック 1 および 2 はキャッシュヒットするため，ファイル I/O が実行されません。また，ブロック 3 のデータを読み込むためのキャッシュブロックは，

DAM ファイル B のキャッシュブロック数がキャッシュブロック再利用境界値を超えていないため，ほかの DAM ファイルの内，キャッシュブロック数がキャッシュブロック再利用境界値を超えている DAM ファイル（この場合は DAM ファイル A）のキャッシュブロックを解放し確保します。確保したキャッシュブロックにブロック 3 のデータを読み込みキャッシュブロックチェインにつなげます。これによって，DAM ファイル B のキャッシュブロックチェインには，三つのキャッシュブロックがつながることになります。

3. ブロック 1，2，3，および 4 を参照するとき，ブロック 1，2，および 3 はキャッシュヒットするため，ファイル I/O は実行されません。また，ブロック 4 のデータ読み込み時には，DAM ファイル A の未使用キャッシュブロックを再利用するため，ファイル I/O が 1 回だけ実行されます。
4. 3.を DAM ファイル A のキャッシュブロック数がキャッシュブロック再利用境界値を超えるまで繰り返します。キャッシュブロックチェインからの解放処理，およびファイル I/O の回数は，約 50 回ずつとなり性能が向上します。

### (b) 性能が劣化する例

DAM ファイルのブロック長などの条件は上記と同じとします。ただし，各ファイルのアクセス形態を次に示すとおりとします。

1 回目：DAM ファイル A の 1～50 ブロックを参照

2 回目：DAM ファイル B の 1～50 ブロックを参照

3 回目：DAM ファイル A の 51～100 ブロックを参照

4 回目：DAM ファイル B の 1～50 ブロックを参照

：

以降，DAM ファイル A のブロックを 50 ブロックずつずらしながら，DAM ファイル A と B を交互に参照します。

キャッシュブロックの最大が 100 個であるため，2 回目で DAM ファイル A と B ともに，50 個のキャッシュブロックがつながります。3 回目以降の処理で，damcache 定義コマンドを指定しない場合と，指定する場合の処理の違いを次に示します。

- damcache 定義コマンドを指定しない場合

  3 回目では，ブロック 1～50 のキャッシュブロックを解放し，ブロック 51～100 のデータに置き換えたあとチェインにつなげます。

  4 回目では，すべてのブロックがキャッシュヒットするため，キャッシュブロックの解放処理，およびファイル I/O は実行されません。

  以降，DAM ファイル A では，3 回目のときと同様にすべてのキャッシュブロックの解放とファイル I/O が実行されますが，DAM ファイル B については，キャッシュブロックの解放，およびファイル I/O は実行されません。DAM ファイル A の参照ブロックが 1000 ブロックまでとすると，キャッシュ

ブロックの解放処理回数は 950 回，ファイル I/O の回数は DAM ファイル A と B 合わせて 1050 回です。

- damcache 定義コマンドを指定する場合

  DAM ファイル A のキャッシュブロック再利用境界値を 100，DAM ファイル B のキャッシュブロック再利用境界値を 50 とします。

  3 回目では，DAM ファイル A のブロック 51～100 を参照するとき，50 個分のキャッシュブロックを確保する必要があります。このとき，DAM ファイル A のキャッシュブロック数がキャッシュブロック再利用境界値を超えていないため，ほかの DAM ファイルのキャッシュブロックを再利用します。このとき，DAM ファイル B のキャッシュブロック数はキャッシュブロック再利用境界値を超えていなくても，共用メモリ内の空き領域がないため，強制的に再利用します。そのため，DAM ファイル B のブロック 1～50 のデータが設定されているキャッシュブロックが解放され，DAM ファイル A のブロック 51～100 までのデータに置き換えられ，DAM ファイル A のキャッシュブロックチェインの先頭につながれます。このため，3 回目が終了した時点では，DAM ファイル A のキャッシュブロックチェインには，ブロック 1～100 までのキャッシュブロックがつながれ，DAM ファイル B のキャッシュブロックはなくなります。

  4 回目では，DAM ファイル B のブロック 1～50 を参照するとき，50 個分のキャッシュブロックを確保する必要があります。3 回目と同様に DAM ファイル A のキャッシュブロックを強制的に再利用し，DAM ファイル A のブロック 1～50 のデータが設定されているキャッシュブロックを解放し，DAM ファイル B のブロック 1～50 のデータを設定したあと，DAM ファイル B のキャッシュブロックチェインにつなぎます。

  以降，3 回目および 4 回目と同様の処理を繰り返し，DAM ファイル A の参照ブロックが 1000 ブロックまでとすると，キャッシュブロックの解放処理回数は 1900 回，ファイル I/O の回数は DAM ファイル A と B 合わせて 2000 回となり，damcache 定義コマンドを指定しない場合と比較して性能が劣化します。

### (c) 注意事項

この機能を使用するときの注意事項を次に示します。

- アクセスする DAM ファイルのキャッシュブロック数がキャッシュブロック再利用境界値を超えていない場合，ほかの DAM ファイルのキャッシュブロックを優先して再利用します。このとき，ほかの DAM ファイルの全キャッシュブロックがアクセス中の場合，アクセスする DAM ファイルのキャッシュブロックに対して再利用処理を実行します。そのため，キャッシュブロック再利用境界値を超えていなくても，再利用されることがあります。
- damadd コマンドに-l オプションを指定しないで実行すると，その DAM ファイルのキャッシュブロック再利用境界値は，DAM サービス定義の dam_default_cache_num オペランドの指定値に依存しないで 0 となります。
- DAM サービス定義の各キャッシュブロック再利用境界値に指定範囲外の値を指定した場合，次に示すとおり動作します。
  - dam_default_cache_num オペランドに指定範囲外の値を指定するとき

KFCA00216-E メッセージを出力して，指定の不正な個所を表示します。さらに，KFCA01644-I メッセージを出力して省略値の採用を表示します。この場合，damcache 定義コマンドでキャッシュブロック再利用境界値を指定していない DAM ファイルのキャッシュブロック再利用境界値として 0 を使用します。

- damcache 定義コマンドに指定範囲外の値を指定するとき

  KFCA02529-E メッセージを出力して，指定の不正な個所を表示します。さらに，KFCA01644-I メッセージを出力して省略値の採用を表示します。この場合，該当する DAM ファイルのキャッシュブロック再利用境界値として，dam_default_cache_num オペランドで指定した値を使用します。dam_default_cache_num オペランドの指定がない場合，または指定範囲外の値を指定した場合は，0 を使用します。

## (15)　回復対象外 DAM ファイルのキャッシュレスアクセス

「4.4.1(11)　キャッシュブロックの処理」で説明したように，DAM サービスは，一度読み込んだ DAM ファイルのブロックデータを DAM サービス専用共用メモリにため，ファイル I/O の実行回数を削減しています。

回復対象外の DAM ファイルに限り，直接ファイル I/O を実行してディスク上のブロックデータを UAP へ返せます。この方式を**キャッシュレスアクセス**といいます。回復対象外の DAM ファイルをキャッシュレスアクセスにする場合は，DAM サービス定義の damfile 定義コマンドの-f オプションで指定します。

キャッシュレスアクセスを指定すると，参照，更新ごとにファイル I/O が実行されます。

- キャッシュレスアクセスで性能向上が見込まれる例

  複数のブロックデータに 1 回だけアクセスする処理
- キャッシュレスアクセスで性能が劣化する例

  同じブロックデータに複数回アクセスする処理

キャッシュレスアクセス指定の DAM ファイルを使用する場合，最適な DAM サービス専用共用メモリサイズを算出し，DAM サービス定義の dam_cache_size_fix オペランドに指定すると，不要なメモリを削減できます。詳細については，マニュアル「OpenTP1 システム定義」の DAM サービス定義の dam_cache_size_fix オペランドの説明を参照してください。

# 4.4.2　TAM ファイル（TP1/FS/Table Access）

**TAM ファイルサービス**を使うと，OpenTP1 ファイルシステム上の OpenTP1 ファイル（直接編成ファイル）のデータに，メモリ上のテーブルからアクセスできます。そのため，データへのアクセス処理を高速にできます。また，トランザクションと同期して，OpenTP1 ファイルの状態を管理できます。OpenTP1 で TAM ファイルを使う場合には，TP1/FS/Table Access が必要です。

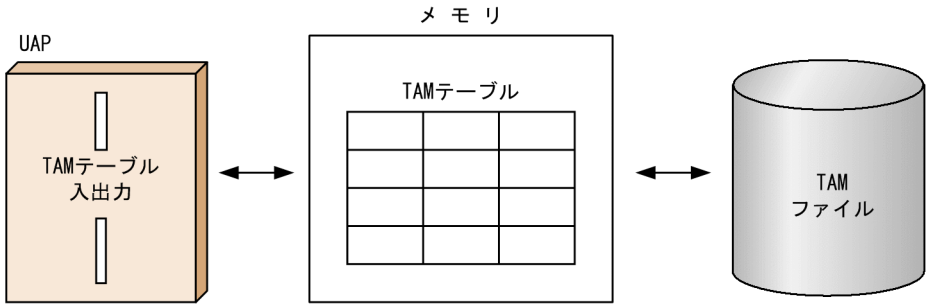
TAM ファイルは，ユーザデータを格納するファイルです。DAM ファイルがディスクのデータに直接アクセスするのに対して，TAM ファイルはデータを**単純構造テーブル**としてメモリにロードして，メモリ上

のテーブルにアクセスします。TAM サービスが提供するこの単純構造テーブルを TAM テーブルといい，TAM テーブルを格納するファイルの実体を TAM ファイルといいます。

TAM ファイルを使うときは，TAM サービス定義を作成し，TAM テーブルと TAM ファイルを対応づけます。TAM テーブルと TAM ファイルは 1 対 1 に対応します。さらに，TAM サービス定義で TAM テーブルに任意の名称を指定します。この TAM テーブル名を使用して，TAM テーブルを運用します。

TAM テーブルと TAM ファイルの関係を次の図に示します。

図 4-18　TAM テーブルと TAM ファイルの関係




## (1)　TAM テーブルの特長

TAM テーブルは，レコードのアドレスを管理する方式の違いで，ツリー形式とハッシュ形式に分けられます。TAM テーブルはレコードごとに，検索用のキー値で管理しています。ツリー形式とは，レコードごとに割り当てられたキー値の大小を基にアドレスを管理する形式です。ハッシュ形式とは，レコードごとに割り当てられたキー値を TAM サービス内部のハッシュ関数を用いて変換し，変換後の値を基にアドレスを管理する形式です。この変換後の値をハッシュ値といいます。

ハッシュ形式はメモリ効率が悪くなりますが，ツリー形式よりも高速にアクセスできます。メモリ量に余裕がある場合には，ハッシュ形式，余裕がない場合にはツリー形式で TAM テーブルを作成してください。ツリー形式とハッシュ形式については，「4.4.2(2)　TAM テーブルの形式」を参照してください。

TAM テーブルおよび TAM ファイルの概要を次の表に示します。

表 4-11　TAM テーブルおよび TAM ファイルの特長

| 説明する内容 | 特長 |
|---|---|
| ファイル構成とレコード形式 | TAM ファイルは，直接編成ファイルです。事前に OpenTP1 ファイルとして割り当てたファイルを使います。<br>レコード形式は固定長の非ブロック形式です。 |
| アクセス形態 | キー値を使ったテーブルアクセス法で，ブロックを参照または更新します。TAM ファイルには，次に示す 3 種類があります。<br>• 参照するだけの参照型ファイル<br>• レコードを追加・削除できない更新型ファイル<br>• レコードを追加・削除できる更新型ファイル<br>連続した複数のレコードを一括して入出力することもできます。 |

| 説明する内容 | 特長 |
|---|---|
| トランザクションとの関係 | トランザクション処理中に仮更新したレコードの内容は，トランザクションがコミットした時点で，メモリ上で実更新されます。トランザクションがロールバックした場合は，仮更新した内容は廃棄されます。<br>メモリ上で実更新されたレコードは，TAM ファイルに反映されます。TAM サービスは，メモリ上で実更新されたレコードだけを，一定の間隔で TAM ファイルへ書き込みます。 |
| ファイルのバックアップとリストア | TAM ファイルのバックアップとリストアには，次のコマンドを使います。<br>• tambkup コマンド<br>• tamrstr コマンド<br>バックアップファイルとバックアップ取得時点以降の回復用ジャーナルを基に TAM ファイルの内容を最新の状態に回復できます。TAM ファイルを回復する機能を **TAM FRC**（Table Access Method File Recovery）といいます。<br>tambkup コマンドでは，オンライン中に TAM ファイルをバックアップすることもできます。 |
| オンライン中のテーブルの追加 | オンライン中に TAM テーブルを追加する場合は，追加する物理ファイルを作成してから，tamadd コマンドを使います。tamadd コマンドでファイルを追加したあとで，tamrles コマンドを実行すると，オンラインで使えるようになります。 |
| オンライン中のファイルの削除 | オンラインで使っているファイルを切り離す場合は tamrm コマンドを，切り離した TAM ファイルの物理ファイルを削除する場合は tamdel コマンドを使います。オンライン中にファイルを削除する場合は，tamrm コマンドで論理ファイルを切り離してから，物理ファイルを tamdel コマンドで削除します。 |
| ユーザデータの抽出と TAM ファイルの再作成 | TAM ファイルからのユーザデータの抽出と TAM ファイルの再作成には，次のコマンドを使います。<br>• tambkup コマンド（-d オプション指定）<br>• tamcre コマンド<br>TAM ファイルの再作成は，ファイル容量の拡張やデータ移行を行う場合にも有効な手段となります。 |

メモリが十分にある場合には，TAM ファイルを使用すれば性能面で優れたシステムを構築できます。利用できるメモリ量が限られている場合には，DAM ファイルを使用してください。

DAM ファイルに入出力する関数で，TAM テーブルにアクセスできます。このため，使用できるメモリを増やした場合は，それまで使用していた UAP を変更しないで，ユーザデータを DAM ファイルから TAM ファイルに移せます。ユーザデータの一部を TAM ファイルに移し，一部を DAM ファイルに残すような場合にも有効です。

## (2) TAM テーブルの形式

TAM サービスは，TAM テーブルのレコードとキー値の対応関係を基に，レコードのアドレスを管理します。通常，キー値はレコードの中にも含まれています。レコードにキー値を含めたくない場合は，tamcre コマンドで，レコードからキー値を削除できます。

ツリー形式では，中間の値を持つキー値をツリーのルートとして，それよりも大きな値を持つか小さな値を持つかによって，レコードをツリー構造で管理しています。ツリー形式は，ツリー構造に管理されたキー値の大小を順々に比較していくことでアドレスを求めるため，検索に時間が掛かります。

ハッシュ形式では，キー値をハッシュ値に変換し，ハッシュ値とレコードアドレスを対応づけることによって，レコードを管理しています。TAM テーブルの全レコードのうち，ハッシュ値とレコードアドレスを対応づけた領域を**ハッシュ域**といいます。同じハッシュ値を持つレコードは，リスト構造で管理されます。リストで管理されたキー値を順々に比較していくことで，同じハッシュ値を持つレコードのアドレスを求めます。

ツリー形式とハッシュ形式で，アクセス時のキー値の指定方法が異なります。

TAM テーブルアクセス時のキー値の指定方法を次の表に示します。

### 表 4-12　TAM テーブルアクセス時のキー値の指定方法

| 検索種別 | ツリー形式 | ハッシュ形式 |
|---|---|---|
| キー値=n | 指定されたキー値を持つレコードを検索します。指定されたキー値を持つレコードがない場合，エラーリターンします。 | |
| キー値<n<br>キー値>n<br>キー値≦n<br>キー値≧n | 指定されたキー値に最も近いキー値を持つ最初のレコードを検索します。 | エラーリターンします。 |
| 先頭検索※ | エラーリターンします。 | キー値を無視して，TAM ファイルの先頭レコードを検索します。 |
| NEXT 検索※ | エラーリターンします。 | 指定されたキー値を持つレコードの，次のレコードを検索します。 |

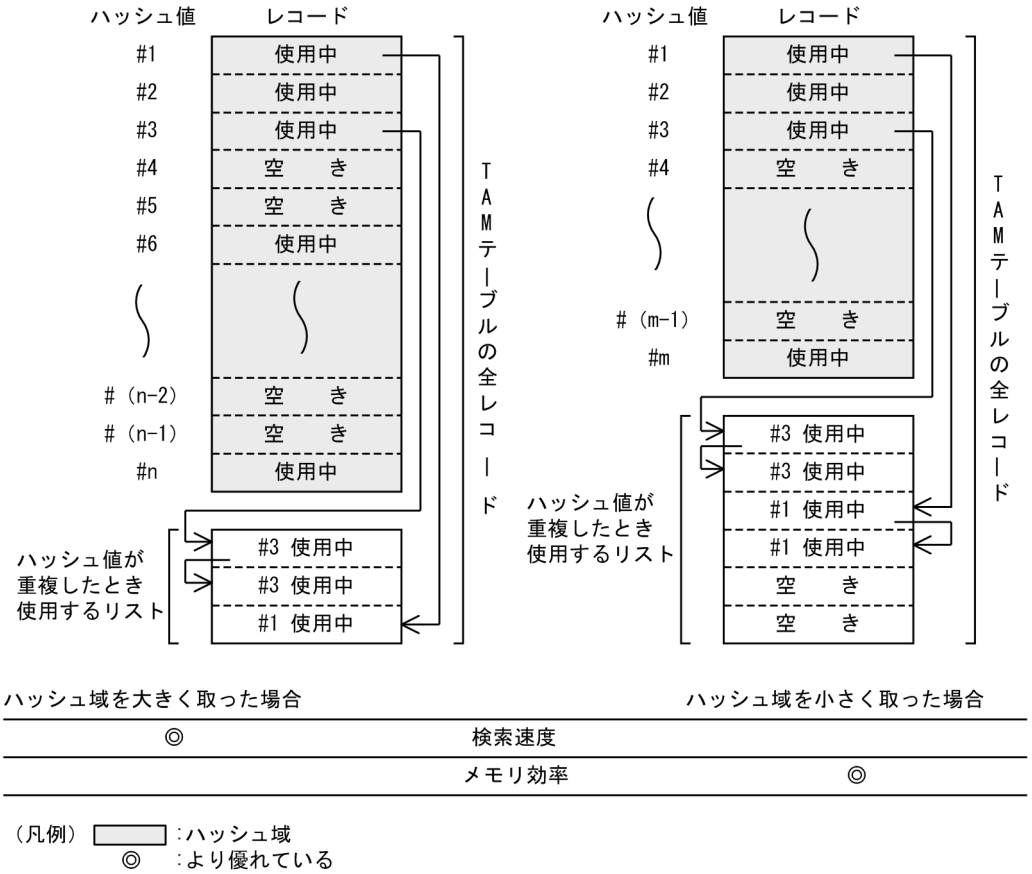
注※

ハッシュ形式の場合，ユーザは先頭検索と NEXT 検索を組み合わせて，TAM テーブル上の全レコードを検索できます。ただし，レコードからキー値を削除した場合には，NEXT 検索時に指定するキー値がユーザにわからないため，全レコードの検索ができません。

ハッシュ形式の場合，全レコードのうちどれだけハッシュ域として使用するかを，TAM ファイル作成時，tamcre コマンドの引数で指定します。ハッシュ域を大きく取った場合，ハッシュ値の重複が発生する確率が小さくなり，リストの検索が不要になる割合が増すため，検索が速くなります。ただし，リストの領域が小さくなり，リストが満杯になったとき，それ以上ハッシュ値の重複が発生すると，ハッシュ域に空きがあってもレコードを追加できないので，メモリ効率が悪くなります。ハッシュ域を小さく取った場合，ハッシュ値の重複が起きる確率が高まります。このため，リストを検索する割合が増して，検索が遅くなります。その代わり，メモリ効率は良くなります。

ハッシュ域を大きく取った場合と，小さく取った場合を次の図に示します。

図 4-19　ハッシュ域を大きく取った場合と小さく取った場合




## (3)　TAM ファイルの作成方法

OpenTP1 ファイルシステムを作成したあと，tamcre コマンドで TAM ファイルを初期作成します。TAM ファイルの作成と同時に初期データを TAM ファイルに格納する場合には，まずファイルに初期データを格納しておきます。TAM ファイルの初期作成時に tamcre コマンドの引数でファイルを指定すれば，OpenTP1 が初期データをファイルから TAM ファイルに複写します。TAM ファイルの初期データを格納するこのファイルを，**TAM データファイル**といいます。

## (4)　TAM テーブルのアクセス形態

TAM テーブルは，UAP から TAM テーブルのレコードにどのようにアクセスするかによって，**参照型**，**追加・削除できない更新型**，**追加・削除できる更新型**に分類できます。UAP からのレコードに対するアクセスを制限するために設定する，TAM テーブルの属性を**アクセス形態**といいます。アクセス形態を設定することで，TAM ファイルを目的に応じて使い分けることができ，TAM テーブルのレコードを管理できます。

**参照型**は，レコードを参照することだけができる TAM テーブルです。オンライン中にレコードを追加したり，削除したり，更新したりすることはできません。

**追加・削除できない更新型**は，レコードを更新できますが，オンライン中にレコードを追加したり，レコードを削除したりすることはできません。**追加・削除できる更新型**は，オンライン中にレコードを追加したり，削除したり，更新したりできます。

OpenTP1 の開始と同時にオンラインに組み込む TAM テーブルのアクセス形態は，TAM サービス定義で指定します。また，OpenTP1 稼働中にオンラインに追加する TAM テーブルのアクセス形態は，tamadd コマンドの-a オプションで指定します。

## (5) TAM テーブルのロードとアンロード

TAM サービスは，ユーザが指定したタイミングで，TAM ファイルに格納されている TAM テーブルをメモリにロードします。TAM テーブルをメモリにロードするタイミングを，**ローディング契機**といいます。OpenTP1 の開始と同時にオンラインに組み込む TAM テーブルは，ローディング契機を TAM サービス定義で指定します。また，OpenTP1 の稼働中にオンラインに追加する TAM テーブルの場合には，ローディング契機を tamadd コマンドの-o オプションで指定します。

tamload コマンドでローディングした TAM テーブルは，tamunload コマンドでアンロードします。OpenTP1 の開始と同時，またはオンラインへの追加と同時にローディングした TAM ファイルは，OpenTP1 の終了時点で自動的にアンロードします。また，UAP から dc_tam_open 関数を使ってローディングした TAM ファイルは，dc_tam_close 関数を使った時点で自動的にアンロードします。

TAM テーブルのローディング契機とアンロードの方法を次の表に示します。

### 表 4-13 TAM テーブルのローディング契機とアンロードの方法

| TAM テーブルをオンラインに登録する時期 | ローディング契機（三つのローディング契機のうち一つを選択） | アンロードの方法※1 |
|---|---|---|
| OpenTP1 の開始と同時※2 | OpenTP1 の開始に伴って TAM サービスが開始したとき | TAM 終了時に自動 |
| | tamload コマンドを入力したとき | コマンド入力 |
| | dc_tam_open 関数を使って TAM テーブルをオープンしたとき | クローズ時に自動 |
| OpenTP1 稼働中※3 | オンラインに登録されたとき | TAM 終了時に自動 |
| | tamload コマンドを入力したとき | コマンド入力 |
| | dc_tam_open 関数で TAM テーブルをオープンしたとき | クローズ時に自動 |

注※1

TAM 終了時はローディング契機にかかわらずアンロードします。

注※2

TAM サービス定義でローディング契機を指定します。

注※3

TAM テーブルを追加する TAM テーブル管理コマンドの引数でローディング契機を指定します。

## (6) TAM テーブルの排他制御

TAM テーブルの排他には，テーブル単位で排他をかける**テーブル排他**と，レコード単位で排他をかける**レコード排他**があります。

TAM テーブルやレコードがすでにほかの UAP から排他をかけられていた場合に，排他が解放されるのを待つかどうかを，dc_tam_open 関数，dc_tam_read 関数，dc_tam_write 関数，dc_tam_delete 関数の引数で指定します。排他の解除を待つ時間は，ロックサービス定義で指定します。この排他待ち限界経過時間が経過しても排他が解放されない場合には，エラーリターンします。

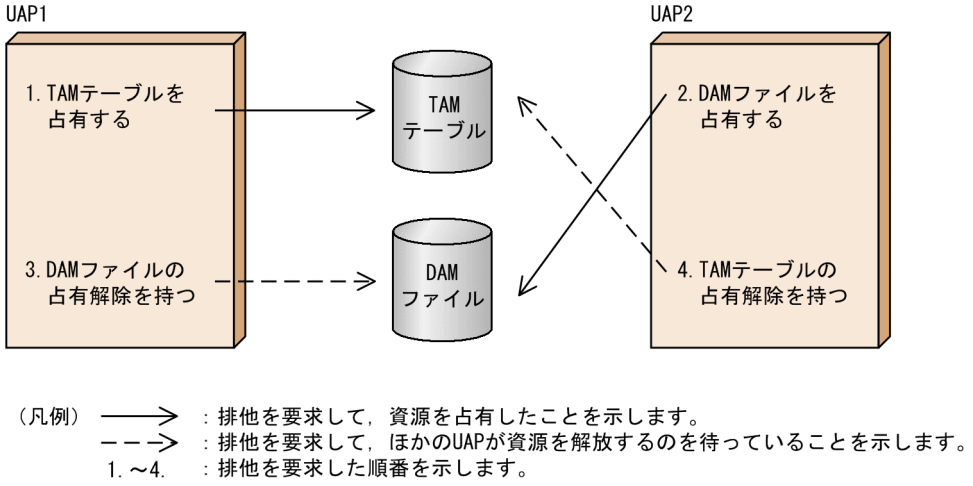
TAM テーブルは，グローバルトランザクション単位に排他をかけます。

## (7) TAM テーブルと DAM ファイルのデッドロックの対処方法

TAM テーブルと DAM ファイルの両方にアクセスする UAP を複数作成した場合に，UAP 間でデッドロックが起こることがあります。デッドロックが起こった場合には，ユーザサービス定義で指定した，デッドロックを解消するための優先順位に従ってデッドロックを解消します。

TAM テーブルと DAM ファイルの両方にアクセスした場合のデッドロックの例を，次の図に示します。

図 4-20　TAM テーブルと DAM ファイルの両方にアクセスした場合のデッドロックの例




## (8) TAM ファイルのオンラインバックアップ

TAM ファイルのバックアップを，OpenTP1 の業務中に取得できます。これを**オンラインバックアップ**といいます。TAM ファイルのバックアップを取得するコマンド（tambkup コマンド）に-o オプション指定して実行すると，オンライン中に TAM ファイルをバックアップできます。

オンラインバックアップしたファイルで TAM ファイルを回復すると，回復に使うアンロードジャーナルファイルの量が少なくて済みます。そのため，-o オプションを指定しないでバックアップしたファイルを使って TAM ファイルを回復する場合と比べて，TAM ファイルの回復処理に掛かる時間が少なくて済みます。

-o オプションを指定しない場合は，オフライン状態でバックアップすることになります。この場合，次の手順でバックアップしてください。

1. tamhold コマンドを実行して，論理ファイルを論理閉塞します。
2. tamrm コマンドを実行して，論理閉塞した論理ファイルをオンラインから切り離します。
3. -o オプションを指定しない tambkup コマンドを実行して，TAM ファイルをバックアップします。

### (9) 標準入出力を使った TAM ファイルのバックアップとリストア

TAM ファイルのバックアップ先に標準出力を，リストアの入力ファイルに標準入力を使うことができます。標準入出力を使うことで，コマンドのリダイレクトができるようになります。

標準入出力を使う場合には，tambkup コマンドおよび tamrstr コマンドに-s オプションを指定して実行します。

### (10) ユーザデータの抽出と TAM ファイルの再作成

TAM ファイルからユーザデータを抽出できます。抽出したユーザデータを TAM データファイルとして tamcre コマンドに再度入力することで，TAM ファイルを再作成できます。

TAM ファイルの再作成は，ファイル容量を拡張したり，ほかのディスク媒体やほかのシステムにデータを移行する場合にも必要な手段となります。抽出したデータに対してデータの追加，削除や更新を行った上で TAM ファイルを再作成することもできます。

ユーザデータを抽出するには，tambkup コマンドに-d オプションを指定して実行します。

なお，-d オプションを指定して抽出したユーザデータを tamrstr コマンドでリストアすることはできません。

## 4.4.3 IST サービス（TP1/Shared Table Access）

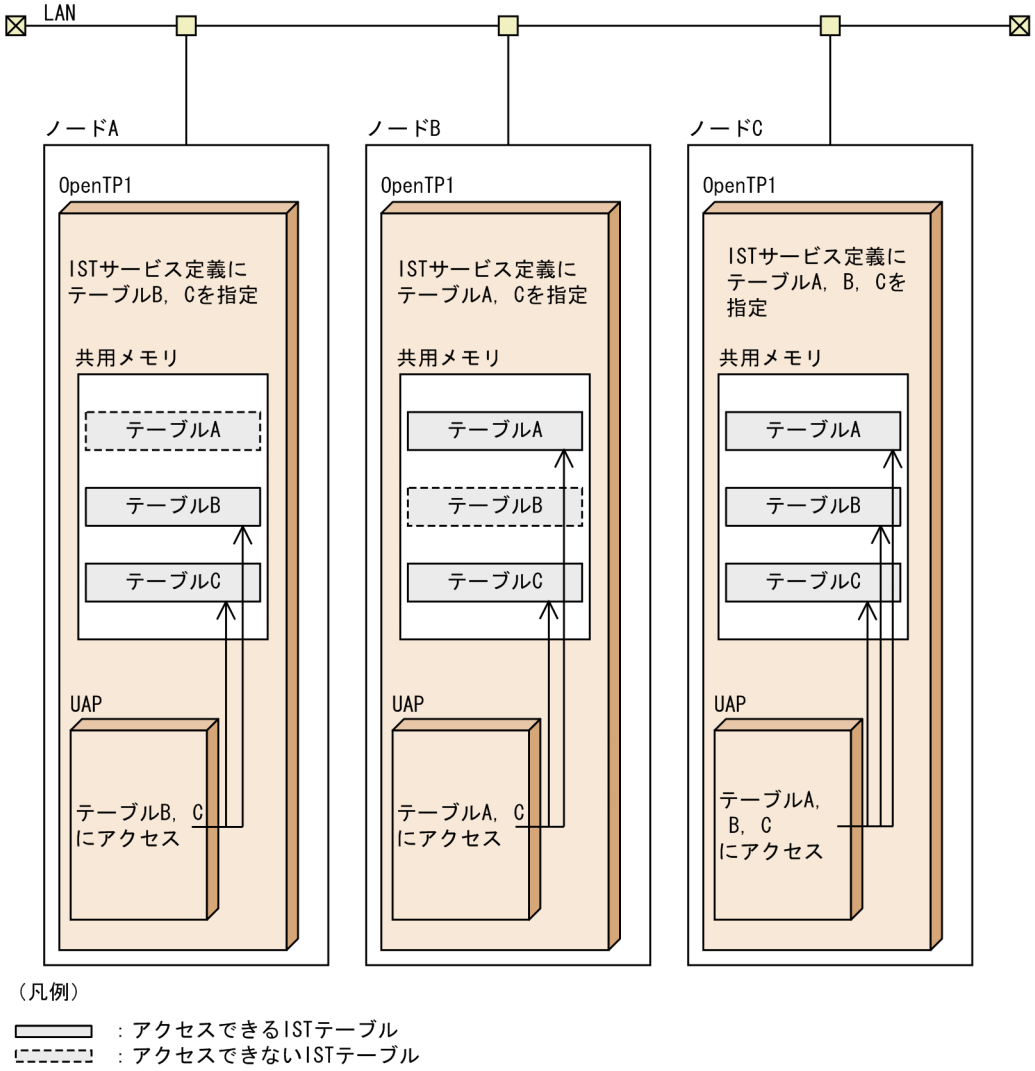
複数の OpenTP1 システムが，ノードをわたってテーブルを共用できる機能を **IST サービス**といいます。IST サービスで使えるテーブルを **IST テーブル**といいます。IST サービスを使うと，テーブルの実体がどのノードにあるかを意識しないで，UAP からデータを参照，更新できます。また，各ノードの業務状態を管理するために，メールとして IST テーブルを使うこともできます。ただし，複数のノードにわたってデータを配布させる場合，次に示す条件の業務には IST サービスはお勧めできません。

- データを即時に配布する必要がある業務
- 大量のデータを扱う必要がある業務
- 頻繁にデータを更新する業務

IST テーブルを使う場合，各ノードのシステムに TP1/Shared Table Access が組み込まれていることが前提となります。

IST サービスの構成を次の図に示します。

図 4-21　IST サービスの構成




IST サービスを使用するには，ノード間の IST テーブル定義の指定を合わせてください。この図の場合，ノード A，ノード B，およびノード C で IST テーブル定義に指定したテーブル名が合っていないため，予期しないテーブル情報を受信したとして，ノード A およびノード B で，OpenTP1 が終了するまで定期的に KFCA25533-W メッセージを出力し続けます。

## (1)　IST テーブルへのアクセス環境

IST テーブルは，各ノードの共用メモリ上にあるテーブルです。テーブルの実体にあたるファイルはありません。そのため，UAP から IST テーブルへアクセスできるのは，**オンライン中**だけです。オフライン環境では IST テーブルにはアクセスできません。

## (2) IST テーブルの構造

UAP から IST テーブルのデータを参照，更新するときは，**レコード単位**でアクセスします。IST テーブルは，複数のレコードから構成されます。UAP の処理では，一つのレコードへアクセスすることも，複数のレコードを一括して指定してアクセスすることもできます。

## (3) IST テーブルへのアクセス

UAP から IST テーブルへアクセスする手順については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

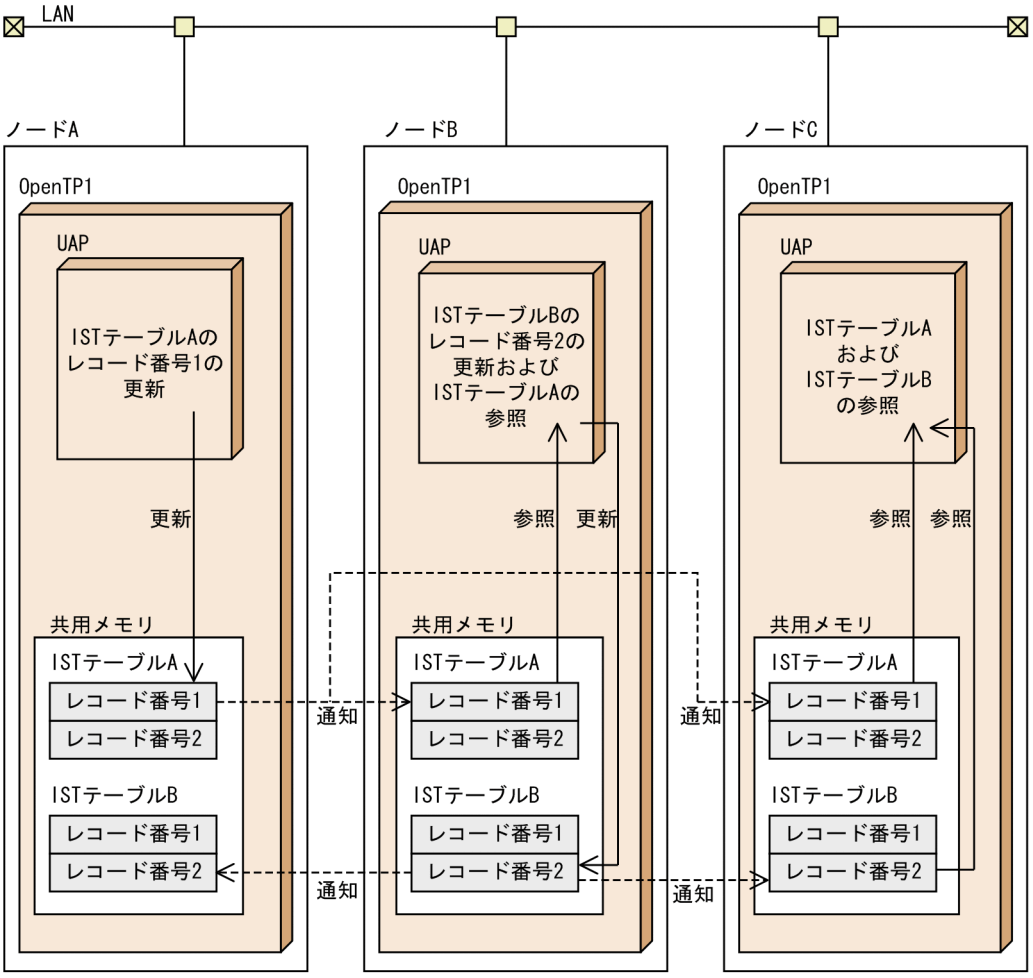
IST テーブルへのアクセスは，トランザクションの関数でコミット，ロールバックできません。

IST テーブルでは，UAP から呼び出した関数ごとに排他制御しています。データの入力から更新まで，IST テーブルを占有する制御はしていません。そのため，一つのテーブルに複数の UAP からアクセスした場合でも，デッドロックが起こることはありません。

## (4) IST サービスの運用例

IST サービスの効果的な運用例を次の図に示します。

図 4-22　IST サービスの効果的な運用例



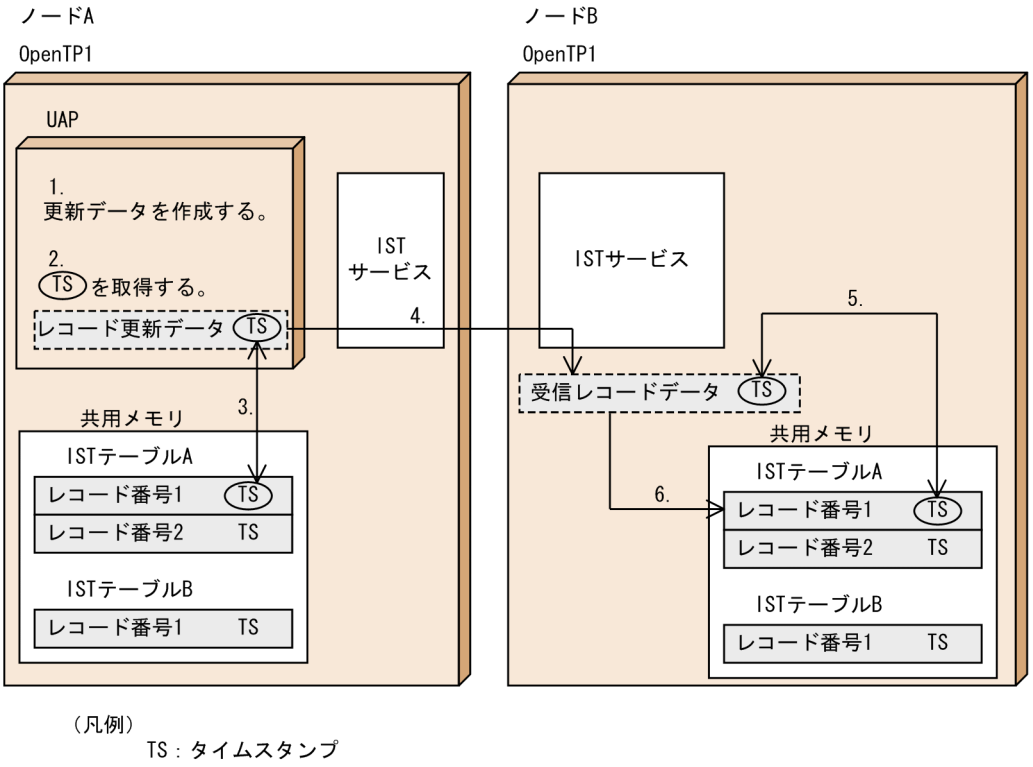

この図では，IST テーブル A はノード A だけが更新し，IST テーブル B はノード B だけが更新します。それぞれのノードで更新したデータは，IST サービスが各ノードに通知および反映するので，更新された IST テーブルは，どのノードでも参照できます。このように，IST テーブルを更新するノードを一つに限定した運用が効果的です。

例えば，IST テーブル A には，ノード A のステータス情報を書き込み，IST テーブル B には，ノード B のステータス情報を書き込みます。ノード C が更新できる IST テーブル C を作成し，IST テーブル C にノード C のステータス情報を書き込めば，各ノードで，全ノードのステータス情報を参照できます。ただし，自ノードで更新した IST テーブルの他ノードへの通知が完了するまでに必要な通信時間があります。この通信時間内は最新の情報が未反映のため，古い情報を参照することがあります。

## (5)　IST サービスの利用時の注意事項

複数のノードで IST サービスを使う場合には，各ノードの時刻を合わせておく必要があります。ノード間で時刻が一致していないと，あるノードで更新したデータに対して，別のノードからの更新が反映されないことがあります。IST サービスで複数のノードの IST レコード（IST テーブル中のレコード）を更新する処理の流れを次の図に示します。

## 図 4-23　IST レコードの更新処理




1. ノード A の IST テーブル A の IST レコード（レコード番号 1）を更新するレコード更新データを作成します。
2. 現在の時刻（マシン時刻：マイクロ秒精度）を取得し，レコード更新データにタイムスタンプとして付与します。
3. ノード A の共用メモリ上の該当する IST レコードに設定されているタイムスタンプとレコード更新データに付与したタイムスタンプとを比較します。

   レコード更新データの方が新しい場合は，共用メモリ上の IST レコードを更新します。レコード更新データの方が古い場合は，共用メモリ上の IST レコードを更新しません。なお，IST レコードを更新しない場合も，dc_ist_write 関数は正常にリターンします。
4. 共用メモリ上の IST レコードを更新した場合，ノード A で IST レコードを更新したことをノード B の IST サービスへ通知します。このとき，IST レコードと IST レコードに付与したタイムスタンプも通知します。
5. 更新された IST レコードを受信したノード B の IST サービスは，ノード内の該当する IST レコードに設定されているタイムスタンプと受信した IST レコードのタイムスタンプとを比較します。
6. 5.の結果，受信した IST レコードのタイムスタンプの方が新しいと判断した場合だけ，ノード B の該当する IST レコードを，受信した IST レコードの情報に更新します。

上記のように，IST サービスでは，IST レコードを更新するか，またはそのままとするかを，タイムスタンプを基に判断しています。次のような場合は，最新の更新データが IST レコードに反映されないことがあります。

- ノード A のマシン時刻がノード B のマシン時刻よりも進んでいる場合

ノードAでISTレコードを更新したあとに，ノードBから同一のISTレコードの更新を通知されても，ノードAのISTレコードに設定されたタイムスタンプの方が新しいと判断します。そのため，ノードBで更新されたISTレコードの情報がノードAのISTレコードに反映されません。

また，ノードAで更新したISTレコードをノードBへ通知したときに，通知したISTレコードのタイムスタンプの方が新しいと判断されるため，ノードBの該当するISTレコードは，実際には最新の情報であっても，通知したISTレコードの情報に更新されてしまいます。

- ノードAのマシン時刻がノードBのマシン時刻よりも遅れている場合
  - ノードBがISTレコードを更新して，そのISTレコードの情報がすでにノードAに通知されているとき

    ノードBでISTレコードを更新したあとに，ノードAで同一のISTレコードを更新しても，更新処理をしないでdc_ist_write関数が正常リターンします。
  - ノードBがISTレコードを更新したが，そのISTレコードの情報がまだノードAに通知されていないとき

    ノードBでISTレコードを更新したあとに，ノードAで同一のISTレコードを更新すると，ノードAの更新情報で，いったんISTレコードを更新しますが，そのあとにノードBから通知されたISTレコードのタイムスタンプの方が新しいと判断するため，ノードBから通知されたISTレコードの情報を，ノードAのISTレコードに反映してしまいます。

## 4.4.4 ISAMファイル（ISAM，ISAM/B）

索引順編成ファイルを管理する，ISAMファイルサービスについて説明します。機能については，マニュアル「索引順編成ファイル管理 ISAM」を参照してください。

### (1) ISAMファイルの概要

索引順編成ファイルは，キーを管理する**インデクス部**と，データを格納する**データファイル部**から構成されます。キーを使用して，**順処理（シーケンシャルアクセス）**や**乱処理（ランダムアクセス）**ができます。

ISAMファイルの操作には，**ライブラリ関数をUAPから呼び出す方法**と，**ユティリティコマンドを実行して管理する方法**があります。

### (2) ISAMサービスの種類

OpenTP1のUAPから処理できるISAMを次に示します。

#### (a) ISAM

ISAMファイルを，通常のファイルとして使います。OpenTP1のトランザクション処理とは同期しません。

### (b) ISAM/B

ISAM ファイルを OpenTP1 トランザクション処理と同期して使う機能です。ISAM/B を使用すると，トランザクション処理のコミット，またはロールバックで，ファイルの整合性が保てるようになります。

- **ISAM/B の前提となる製品**

  ISAM ファイルを ISAM/B として使用する場合は，ISAM に加えて，ISAM トランザクション機能（ISAM/B）が前提となります。

- **ファイルを作成する領域**

  ISAM/B で使用する ISAM ファイルは，OpenTP1 ファイルシステムとして割り当てた領域に作成します。

- **OpenTP1 のファイルサービス（TP1/FS/xxx）との違い**

  ISAM/B では，**ロックサービスを使用しません。**そのため，デッドロックが起こっても，OpenTP1 のロックサービス機能（優先順位による縮退やデッドロック情報の出力）は使用できません。

## 4.4.5 データベースにアクセスする場合

OpenTP1 の UAP で使うユーザファイルに，DBMS を使用した場合について説明します。

### (1) OpenTP1 のトランザクション処理との関係

DBMS を使用する場合，X/Open で規定した DTP モデルの XA インタフェースをサポートした DBMS かどうかで，OpenTP1 のトランザクションと連携できるかどうか決まります。

#### (a) XA インタフェースをサポートした DBMS の場合

XA インタフェースをサポートした DBMS の場合は，OpenTP1 のトランザクションと同期を取って更新できます。同期を取る場合は，OpenTP1 の同期点を制御する関数（dc_trn_begin 関数，dc_trn_unchained_commit 関数，tx_begin()，tx_commit()など）を使います。DBMS が提供するトランザクションを制御する機能などは使えません。OpenTP1 のトランザクション処理で制御できる DBMS は，XA インタフェースをサポートした製品（例　Oracle7 など）に限ります。

XA インタフェースをサポートしている複数のデータベースへアクセスする UAP では，整合性を保ちながら複数のデータベースを更新できます。次に示す OpenTP1 のリソースマネジャは，XA インタフェースをサポートしています。

- TP1/FS/Direct Access（DAM ファイルサービス）
- TP1/FS/Table Access（TAM ファイルサービス）
- ISAM，ISAM/B（ISAM ファイルサービス）
- TP1/Message Control（メッセージ送受信機能）
- TP1/Message Queue（メッセージキューイング機能）

XA インタフェースに準拠した DBMS と，OpenTP1 のリソースマネジャの両方にアクセスする UAP でも，OpenTP1 のトランザクションとして処理できます。障害が原因で UAP が異常終了した場合や，OpenTP1 を再開始（リラン）した場合でも，DBMS と OpenTP1 のリソースマネジャの両方のトランザクションを，OpenTP1 で決着します。

### (b) XA インタフェースをサポートしていない，または XA インタフェースで OpenTP1 と連携していない DBMS の場合

XA インタフェースをサポートしていない DBMS の場合，DBMS へのアクセスはできますが，OpenTP1 のトランザクションとは同期を取れません。

XA インタフェースで OpenTP1 と連携していないため，DBMS へのアクセス中に，障害が原因で UAP が異常終了した場合や，OpenTP1 を再開始（リラン）した場合には，OpenTP1 から DBMS へトランザクションの決着を指示しません。そのため，DBMS 独自の機能でトランザクションを回復する必要があります。

## (2) XA インタフェースでデータベースへアクセスする場合の準備

XA インタフェースをサポートした DBMS を，OpenTP1 と XA インタフェースで連携して使う場合に準備する項目を次に示します。この準備は，OpenTP1 提供以外のリソースマネジャを使う場合に必要です。

### (a) OpenTP1 への登録

OpenTP1 提供以外のリソースマネジャの各種名称を登録します。OpenTP1 へは，次に示すどちらかの方法で登録します。

1. dcsetup コマンドで OpenTP1 をセットアップ後，trnlnkrm コマンドを実行する。
2. 拡張 RM 登録定義を作成する。

拡張 RM 登録定義を作成しておくと，dcsetup コマンドで OpenTP1 をセットアップしたあとに trnlnkrm コマンドを実行しなくても良くなります。trnlnkrm コマンドの使い方については，マニュアル「OpenTP1 運用と操作」を参照してください。拡張 RM 登録定義の指定方法については，マニュアル「OpenTP1 システム定義」を参照してください。

### (b) UAP のリンケージ

UAP の実行形式ファイルを作成するときに，トランザクション制御用オブジェクトファイル，および DBMS のライブラリとオブジェクトモジュールをリンケージする必要があります。

トランザクション制御用オブジェクトファイルは，**trnmkobj コマンド**を実行して作成します。trnmkobj コマンドの使い方については，マニュアル「OpenTP1 運用と操作」を参照してください。

### (c) システム定義

DBMS を使用する場合，トランザクションサービス定義に **trnstring 形式の定義**を，必要に応じて，ユーザサービス定義，またはユーザサービスデフォルト定義に，**trnrmid 形式の定義**をする必要があります。

指定する内容には，DBMS 固有の項目もあります。このような項目は，使用する DBMS のマニュアルを参照してください。

trnstring，trnrmid 形式の定義を指定すると，一つのリソースマネジャを複数の制御単位に分け，接続するユーザ名称などを変更してリソースマネジャに接続することもできます（リソースマネジャ接続先選択機能）。リソースマネジャ接続先選択機能については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

trnstring，trnrmid 形式の定義については，マニュアル「OpenTP1 システム定義」を参照してください。

### (d) 環境変数

DBMS を使用する場合，特定の環境変数が必要になる場合があります。その場合，トランザクションサービス定義，ユーザサービス定義，またはユーザサービスデフォルト定義に，putenv 形式の定義をする必要があります。

putenv 形式の定義については，マニュアル「OpenTP1 システム定義」を参照してください。

### (e) OpenTP1 内部スレッドスタック領域の拡張

DBMS を使用する場合，トランザクションサービス定義で，thread_stack_size オペランドに OpenTP1 内部で使用するスレッドスタック領域のサイズを設定する必要があります。

定義の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

# 5 環境設定手順と運用の概要

この章では，OpenTP1 システムを開始するまでの手順と運用方法の概要，および障害対策について説明します。

## 5.1　OpenTP1 システムの環境設定手順

OpenTP1 の環境設定は，OS の管理者である**スーパユーザ**の作業から始まります。その後，スーパユーザが登録した **OpenTP1 管理者**が，OpenTP1 の環境設定を引き継ぎます。

ここでは，OpenTP1 開始直前までの環境設定手順を示します。

次に示す OpenTP1 の機能を使う場合には，OpenTP1 の環境設定手順に加えて，該当する OpenTP1 の機能を実現するための専用の手順が必要になります。

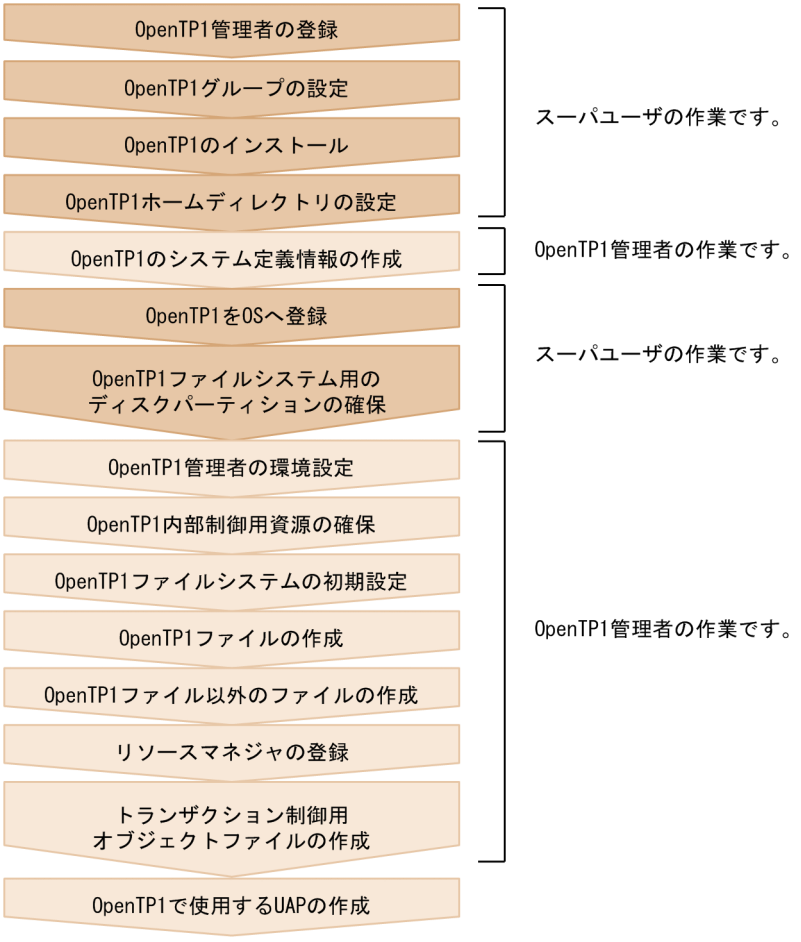
- メッセージ送受信機能を使う場合
- メッセージ制御機能を使う場合
- メッセージキューイング機能を使う場合
- 通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合

### 5.1.1　環境設定手順の概要

#### (1)　OpenTP1 の環境設定手順

OpenTP1 の環境設定手順を次の図に示します。

図 5-1　OpenTP1 の環境設定手順

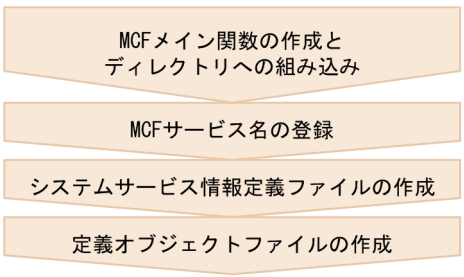



JP1/Base，JP1/AJS，および JP1/AJS2 - Scenario Operation と連携して，スケールアウトのシナリオテンプレートを利用すると，OpenTP1 の環境設定手順の一部を自動化できます。シナリオテンプレートを利用した環境設定については，マニュアル「OpenTP1 運用と操作」のスケールアウトのシナリオテンプレートについての説明を参照してください。

## (2)　メッセージ送受信機能を使う場合の環境設定手順

メッセージ送受信機能を使う場合の環境設定手順を次の図に示します。

図 5-2　メッセージ送受信機能を使う場合の環境設定手順



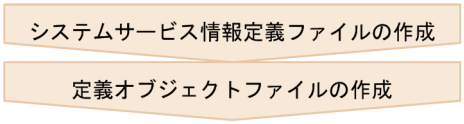
### (3)　メッセージ制御機能を使う場合の環境設定手順

TP1/Messaging を使用する場合の環境設定手順については，マニュアル「TP1/Messaging 使用の手引」を参照してください。

### (4)　メッセージキューイング機能を使う場合の環境設定手順

メッセージキューイング機能を使う場合の環境設定手順を次の図に示します。

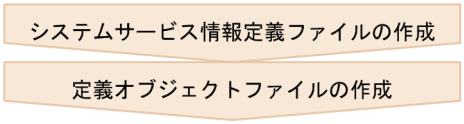
図 5-3　メッセージキューイング機能を使う場合の環境設定手順




### (5)　通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合の環境設定手順

通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合の環境設定手順を次の図に示します。

図 5-4　通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合の環境設定手順




## 5.1.2　環境設定の作業

OpenTP1 の環境設定で実行する作業について説明します。ここに示す作業の詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

### (1)　スーパユーザの環境設定作業

#### (a)　OpenTP1 管理者の登録

OpenTP1 管理者のユーザ ID を設定します。ユーザ ID には必ずパスワードを設定しておいてください。

#### (b)　OpenTP1 グループの設定

OpenTP1 管理者のグループ ID に，OpenTP1 専用の管理者グループを設定します。

### (c) OpenTP1 のインストール

構築する OpenTP1 システムに必要な製品を，インストールします。

### (d) OpenTP1 ホームディレクトリの設定

OpenTP1 をインストールするディレクトリを設定します。このディレクトリの所有者は OpenTP1 管理者を，所有者グループには OpenTP1 専用の管理者グループを設定します。

### (e) OpenTP1 を OS へ登録

OpenTP1 のコマンド（dcsetup コマンド）で，OpenTP1 を OS に登録します。

### (f) OpenTP1 ファイルシステム用のディスクパーティションの確保

OpenTP1 ファイルシステムとして使用する領域を作成します。キャラクタ型スペシャルファイル上に OpenTP1 ファイルシステムを作成するときに，ディスクパーティションを確保します。通常ファイル上に OpenTP1 ファイルシステムを作成する場合は必要ありません。

## (2) OpenTP1 管理者の環境設定作業

### (a) OpenTP1 のシステム定義情報の作成

OpenTP1 のシステムサービス定義の定義ファイルを作成します。作成したシステム定義は，dcdefchk コマンドでチェックできます。

### (b) OpenTP1 管理者の環境設定

OpenTP1 の運用コマンド実行に必要な環境変数を，OpenTP1 管理者のログイン環境に設定します。環境変数 DCDIR のディレクトリ名は，50 バイト以内で設定してください。

### (c) OpenTP1 の内部制御用資源の確保

OpenTP1 で使う資源を確保します。資源を確保するときは dcmakeup コマンドを実行します。dcmakeup コマンドで確保する資源の量は，システムサービス定義の内容から自動的に解析されます。

### (d) OpenTP1 ファイルシステムの初期設定

OpenTP1 ファイルシステムとして使用する領域を，OpenTP1 のコマンド（filmkfs コマンド）で初期化します。OpenTP1 ファイルシステム領域名は，49 文字以内で指定してください。

### (e) OpenTP1 ファイルの作成

OpenTP1 のコマンド（filmkfs コマンド）で初期化済みの領域を，使用する OpenTP1 ファイル固有のコマンド（jnlinit，stsinit など）で初期化します。その後，必要な初期データを作成します。

### (f) OpenTP1 ファイル以外のファイルの作成

UAP の実行形式ファイルや OpenTP1 の各種定義ファイルなど，通常ファイル上で使用するファイルを作成します。

### (g) リソースマネジャの登録

OpenTP1 で提供している製品以外のリソースマネジャを使う場合には，trnlnkrm コマンドで該当する RM を登録します。拡張 RM 登録定義を作成してある場合は，trnlnkrm コマンドを実行する必要はありません。

### (h) トランザクション制御用オブジェクトファイルの作成

trnmkobj コマンドで，トランザクション処理をする UAP に必要な，トランザクション制御用オブジェクトファイルを作成します。

### (i) OpenTP1 で使用する UAP の作成

OpenTP1 で使用する UAP を作成します。メッセージ送受信機能を使う場合には TP1/Message Control の，メッセージキューイング機能を使う場合には TP1/Message Queue の環境設定が完了してから，UAP を作成してください。

## (3) メッセージ送受信機能を使う場合の環境設定手順

メッセージ送受信機能（TP1/Message Control）を使う場合には，次に示す作業をします。

- MCF メイン関数の作成とディレクトリへの組み込み
- MCF サービス名の登録
- システムサービス情報定義ファイルの作成
- 定義オブジェクトファイルの作成

## (4) メッセージキューイング機能を使う場合の環境設定手順

メッセージキューイング機能（TP1/Message Queue）を使う場合には，次に示す作業をします。

- システムサービス情報定義ファイルの作成
- 定義オブジェクトファイルの作成

## (5) 通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合の環境設定手順

TP1/NET/OSI-TP-Extended を使った OSI TP 通信をする場合には，次に示す作業をします。

- システムサービス情報定義ファイルの作成

- 定義オブジェクトファイルの作成

## 5.2 OpenTP1 システムの運用

OpenTP1 には，日常的にしなければならない運用と，システム変更のための運用，そのほかの運用があります。システム運用の詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。OpenTP1 の日常的な運用を表 5-1 に，システム変更のための運用を表 5-2 に，そのほかの運用を表 5-3 に示します。

### 表 5-1 OpenTP1 の日常的な運用

| 運用項目 | 運用の目的 | 運用方法 |
|---|---|---|
| OpenTP1 の開始 | 業務を開始するため。<br>前回の終了状態を引き継ぐ再開始と，引き継がない正常開始があります。 | dcstart コマンドで開始するか，OS 起動時に自動的に開始するかのどちらかを，システム環境定義で指定します。 |
| OpenTP1 の正常終了 | 業務を正常に終了するため。 | dcstop コマンドで終了させます。 |
| OpenTP1 の強制正常終了 | オンラインシステムを正常に終了するため。<br>UAP がオンライン中に異常終了していた場合でも，オンラインシステムを正常終了します。 | dcstop -n コマンドで終了させます。 |
| OpenTP1 の計画停止 A | MCF の管理する端末に障害が発生した場合などオンラインシステムを一時的に中断させるため。<br>出力キューにメッセージを残したまま終了します。 | dcstop -a コマンドで終了させます。 |
| OpenTP1 の計画停止 B | オンラインシステムを直ちに停止させるため。<br>実行中のサービスの完了後，直ちに OpenTP1 を終了します。 | dcstop -b コマンドで終了させます。 |
| OpenTP1 の MCF 構成変更準備停止 | システムの構成を変更した上で，前回の終了状態を引き継ぐ MCF 構成変更再開始をするため。<br>実行中のサービスの完了後，直ちに OpenTP1 を終了します。 | dcstop -b -q コマンドで終了させます。 |
| OpenTP1 の強制停止 | オンラインシステムを緊急に停止させるため。<br>実行中のサービスの完了を待たないで，直ちに OpenTP1 を終了します。 | dcstop -f コマンドで終了させます。 |
| UAP の開始 | UAP 単位で業務を開始するため。 | OpenTP1 開始時に自動的に開始する UAP はユーザサービス構成定義で指定します。 |
| UAP の終了 | UAP 単位で業務を終了するため。<br>正常終了と強制停止があります。 | dcsvstop コマンドで終了させるか，OpenTP1 終了時に自動的に終了させます。 |
| ジャーナルのアンロード | ジャーナル維持機能を利用するため。 | システムジャーナルファイルをファイルにコピーします。 |

| 運用項目 | 運用の目的 | 運用方法 |
|---|---|---|
| ジャーナルのアンロード | DAM ファイル，TAM ファイルの障害発生に備えるため。 | システムジャーナルファイルをファイルにコピーします。 |
| DAM，TAM のバックアップ | DAM ファイル，TAM ファイルの障害発生に備えるため。 | DAM ファイル，TAM ファイルをファイルにコピーします。 |
| メッセージログのファイルへの出力 | OpenTP1 や UAP から出力されたメッセージログを利用するため。 | メッセージログファイルに蓄積されたメッセージログを標準出力に出力します。 |

## 表 5-2　OpenTP1 のシステム変更のための運用

| 運用項目 | 運用の目的 | 運用方法 | OpenTP1 | |
|---|---|---|---|---|
| | | | 停止中 | 稼働中 |
| 定義の変更 | OpenTP1 システムの構成や実行環境を変更するため。 | 定義ファイルの内容を変更します。<br>• **スーパユーザが定義を変更する場合**<br>定義内容を変更して，OpenTP1 を正常終了したあとで，dcsetup -d コマンドで OpenTP1 を OS から削除します。その後 dcsetup コマンドを実行して，OpenTP1 を OS に再登録します。<br>• **OpenTP1 管理者が定義を変更する場合**<br>定義内容を変更して，OpenTP1 を正常終了したあとで，dcreset コマンドを実行します。dcreset コマンドは OpenTP1 稼働中には実行できないので注意してください。 | ○ | × |
| OpenTP1 稼働中の UAP の変更 | UAP を OpenTP1 稼働中に変更するため。（OpenTP1 の終了後，再開したときにも変更したまま） | UAP を終了させたあと，UAP を入れ替えます。必要なら UAP を正常終了させたあとでユーザサービス定義も入れ替えます。入れ替え後に UAP を開始します。<br>正常終了させないで定義を入れ替えた場合，その UAP の動作は保証できません。 | − | ○ |
| OpenTP1 稼働中の UAP の変更（異なるディレクトリ下の UAP に変更） | UAP を OpenTP1 稼働中に一時的に変更するため。（OpenTP1 の終了後再開始したときには，定義されたディレクトリ下の UAP が有効です） | UAP を終了させたあと，運用コマンドでサーチパスを変更します。サーチパスの変更後，UAP を開始します。 | − | ○ |
| ファイルの変更 | ファイルの容量を変更するため。（ファイルの容量は OpenTP1 停止中に変更します。ただし，ステータスファイルは，OpenTP1 稼働中に容量を変更できます） | 物理ファイルを確保してファイルの定義を変更し容量を変えるか，新たなファイルを作成して定義します。<br>ステータスファイルは，運用コマンドで容量を変えられます。 | ○ | △ |

| 運用項目 | 運用の目的 | 運用方法 | OpenTP1 | |
|---|---|---|---|---|
| | | | 停止中 | 稼働中 |
| オンライン中にネットワークの構成を変更 | OpenTP1 システムを構成するネットワークのノードを，オンライン中に追加したり削除したりするため。 | 次の二つの方法があります。<br>**●システム共通定義の内容で変更**<br>1. システム共通定義の all_node オペランドの内容を変更します。<br>2. namndchg コマンドを実行します。<br>現在の all_node オペランドの内容を標準出力に出力するときは，namndchg コマンドに-l オプションを付けて実行します。<br>**●ドメイン定義ファイルの内容で変更**<br>1. all_node および all_node_ex のドメイン定義ファイルを作成します。<br>2. namchgfl コマンドを実行します。 | × | ○ |
| OpenTP1 稼働中の MCF 通信サービスのメイン関数，UOC，ライブラリの変更 | OpenTP1 を停止しないで MCF 通信サービスの運用を変更するため。 | 運用コマンドで，MCF 通信サービスを部分停止させ，該当するメイン関数，UOC，およびライブラリを入れ替えます。 | × | ○ |

（凡例）

○：可能な作業

×：不可能な作業

△：一部可能な作業

－：該当しません

## 表 5-3　そのほかの OpenTP1 の運用

| 運用項目 | 運用の目的 | 運用方法 |
|---|---|---|
| 状態監視 | OpenTP1 の状態を監視するため。 | 各種の状態監視の運用コマンドで OpenTP1 の状態に関する情報を標準出力に出力します。 |
| マルチ OpenTP1 | OpenTP1 と UAP をテストするため。<br>同じノードに二つの OpenTP1 を起動して本番用とテスト用に分けて使用します。 | 二つの OpenTP1 を区別するため，それぞれの OpenTP1 に名称（**OpenTP1 識別子**）を設定します。 |

## 5.3　障害対策の概要

システムの一部に障害が発生しても，OpenTP1 では，部分回復処理をしてシステムが全面停止しないようにしています。また，重大な障害が原因で全面停止した場合でも，全面回復処理をして，システムの状態やユーザの資源を回復できます。ここでは，障害時の回復方法について説明します。障害対策については，マニュアル「OpenTP1 運用と操作」を参照してください。

### 5.3.1　OpenTP1 システム障害の対策

OpenTP1 システムに障害が発生した場合，前回のオンラインの状態を引き継いで，履歴情報を基に OpenTP1 システムを障害発生直前の状態に回復できます。この開始モードを**再開始（リラン）**といいます。OpenTP1 は，再開始（リラン）に備えて，ジャーナルファイルに履歴情報を取得しています。

#### (1)　全面回復による再開始（リラン）の手順

OpenTP1 は，全面回復によって再開始（リラン）します。再開始（リラン）は，システム回復処理，サービス回復処理，トランザクション回復処理の順で実行されます。

1. システム回復処理

   ステータスファイルにはシステム制御情報として，システムサービスや UAP の構成情報，システムファイルの情報などが格納されています。まず，システム制御情報を基に，トランザクションの同期点処理と関係のない，システムの状態を回復します。OpenTP1 は，システム制御情報を基に，回復に使用するチェックポイントダンプファイルやシステムジャーナルファイルを決定します。

2. サービス回復処理

   システム制御情報を基に，システムの状態が回復すると，システムサービスの回復を開始します。システムサービスは，チェックポイントダンプファイルとシステムジャーナルファイルに格納したデータを基に回復します。回復は最新世代のチェックポイントダンプと，最新世代のチェックポイントダンプ取得以降に取得したすべての回復用のジャーナルが必要です。最新世代のチェックポイントダンプが使用できないときには，一世代前のチェックポイントダンプを使用して回復を試みます。この場合，一世代前のチェックポイントダンプ取得以降に取得したすべてのジャーナルが必要です。なお，有効保証世代は二世代までです。有効保証世代のチェックポイントダンプ以降に取得したジャーナルが保証されます。ここまでの処理で，OpenTP1 はオンライン業務を再開できる状態になります。

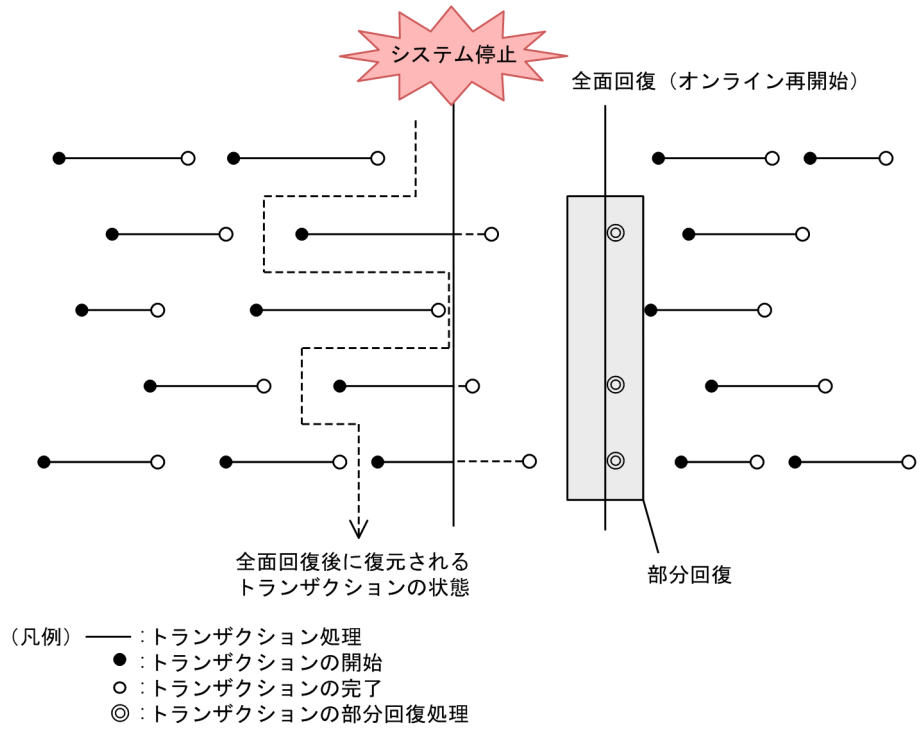
3. トランザクション回復処理

   新たなオンライン処理と並行して，システム障害で OpenTP1 が停止したときに実行中だった，UAP のトランザクション処理を回復してロールバックまたはコミットします。トランザクションをロールバックするか，コミットするかはトランザクション処理がどこまで進んでいたかで決まります。トランザクション処理が同期点の 1 相目以前にあった場合には，OpenTP1 がグローバルトランザクションをロールバックします。2 相目以降にあった場合には，OpenTP1 がルートトランザクションブランチでの決定に従ってグローバルトランザクションをロールバック，またはコミットします。システム停止時

にトランザクション処理がどこまで進んでいたのかは，OpenTP1 がシステムジャーナルファイルの同期点ジャーナルを基に判断します。

全面回復によって復元されるトランザクションの状態を，次の図に示します。

図 5-5　全面回復処理によって復元されるトランザクション

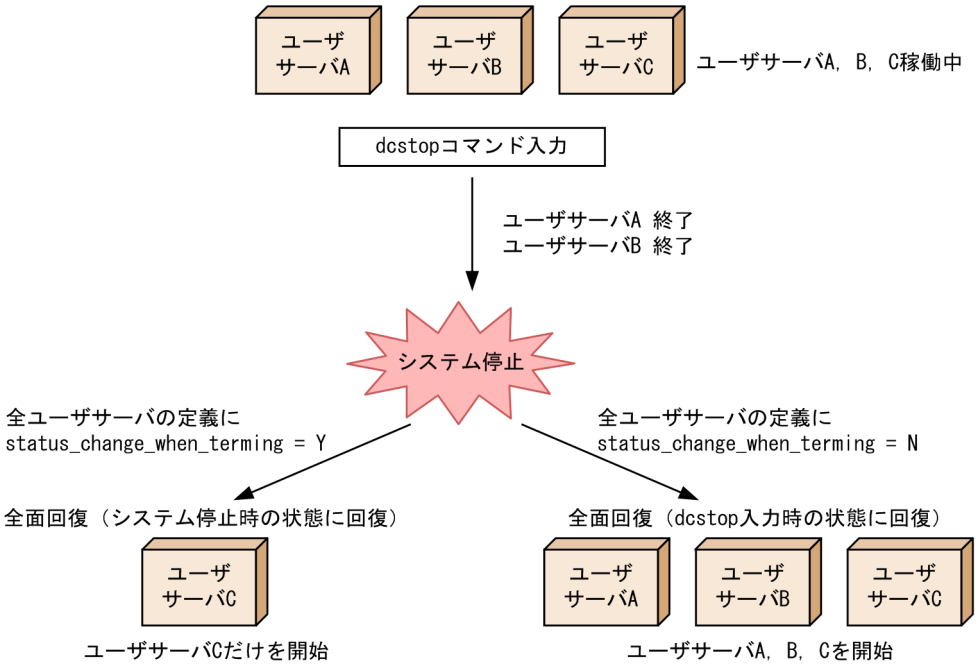



## (2)　OpenTP1 システム終了中にシステム停止した場合の全面回復処理時のユーザサーバの状態

dcstop コマンドを入力して OpenTP1 システムを終了しているときにシステム停止した場合，終了処理中にユーザサーバのステータス変化を反映しないで dcstop コマンド入力時の状態に戻すか，ステータス変化を反映して障害発生時の状態に戻すかを，ユーザサービス定義およびユーザサービスデフォルト定義で選択できます。ユーザサービス定義の status_change_when_terming オペランドに Y を指定したユーザサーバは，最終的なステータスの変化を反映して，システム停止時の状態に戻ります。このオペランドに N を指定したときは，最終的なステータスの変化を反映しないで，dcstop コマンド入力時の状態に戻ります。

OpenTP1 システム終了中にシステム停止した場合の全面回復処理時のユーザサーバの状態を次の図に示します。

図 5-6　システム終了中にシステム停止した場合の全面回復処理時のユーザサーバの状態




## 5.3.2　アプリケーションプログラム障害の対策

UAP に障害が起こった場合，OpenTP1 は UAP 単位の回復処理（**部分回復**）をして，システム全体に影響がないようにします。UAP に障害が起こった場合の処置について説明します。

### (1)　システム定義が間違っている場合の処置

システム定義に適切でない値が設定されている場合は，OpenTP1 はエラーメッセージを出力して該当する UAP を停止させます。この場合は，システム定義を修正して UAP を再び開始させてください。

メモリ容量が不足したことが原因で UAP を開始できない場合は，必要ない UAP のプロセスを停止させるか，または不要なファイルを削除してから OpenTP1 を再開始させてください。

### (2)　プログラムがループした場合の処置

UAP が終了できなかったり，サービス要求の結果が戻ってこなかったりした場合に備えて，OpenTP1 は次に示す時間を監視しています。

- **トランザクション経過時間の監視**

  トランザクションの開始から終了までの経過時間を監視します。トランザクション経過時間を超えると，OpenTP1 は該当する UAP プロセスを強制的に終了させて，トランザクションをロールバックさせます。経過時間を監視するのはトランザクションの処理にだけ有効です。経過時間に 0 を指定した場合は，時間監視をしません。

- **RPC 応答時間監視**

サーバ UAP にサービスを要求してから応答が返るまでの時間を監視します。指定した経過時間を超えた場合，サービスを要求した関数はエラーリターンします。トランザクションの処理の場合は，同期点処理でロールバックします。応答待ち時間に 0 を指定したときは，応答が返るまで待ち続けます。

- CPU 時間の監視

  トランザクション処理を開始してから同期点を取得するまでに，トランザクションブランチが使う CPU 時間を監視します。指定した時間を過ぎてもトランザクションが完了しない場合は，該当するトランザクションブランチを終了させて，トランザクションをロールバックさせます。

## (3) UAP のリンケージの不良で異常終了する場合の処置

OS が HP-UX の場合，リンケージ時のバインドモードには必ず"immediate"を指定してください。UAP を"immediate"以外のバインドモードで作成した場合，UAP が異常終了する場合があります。作成した UAP のバインドモードが"immediate"かどうかは，OS の chatr コマンドで確認してください。

## (4) 異常終了した UAP の部分回復処理の場合の処置

UAP が異常終了した場合，OpenTP1 は UAP の部分回復処理をします。UAP の部分回復には，次のものがあります。

### (a) UAP プロセスの再起動および閉塞処理

- SPP が異常終了した場合

  サービスグループを閉塞するか，またはプロセスを再起動するかをユーザサービス定義に指定します。ただし，ユーザサービス定義の連続異常終了限界経過時間に指定した時間内に 3 回異常終了すると，OpenTP1 は SPP を強制的に閉塞します。閉塞された SPP にサービスを要求しても，エラーリターンします。

- MHP が異常終了した場合

  アプリケーションまたはサービスを閉塞するか，またはプロセスを再起動するかをアプリケーション属性定義に指定します。MHP が閉塞された場合，その MHP に対してメッセージ送信されると，ERREVT2 の MCF イベントが通知されます。

- SUP が異常終了した場合

  異常終了したときにプロセスを再起動するかどうかは，ユーザサービス定義の auto_restart オペランドの指定に従います。

### (b) トランザクションの部分回復処理

異常終了した UAP がトランザクション処理中の場合は，トランザクション部分回復をします。

UAP で障害が起こった場合，トランザクション内の UAP プロセスが異常終了したことを OpenTP1 が検知して，トランザクション回復サービスは該当するトランザクションを回復して，決着させます。このとき，トランザクションのプロセスの回復処理を並行させて，効率を上げることができます。並行させて回復するプロセスの数は，トランザクションサービス定義に指定しておきます。

回復依頼をスケジュールしているときにシステムが使用する領域が不足してトランザクションを回復できない場合があります。これを防ぐため，OpenTP1 は一定間隔で回復していないトランザクションを検索して，再び回復する処理をします。

### (5)　UAP のデッドロックが起こった場合の処置

複数の UAP から同じ資源にアクセスすると，デッドロックが起こる場合があります。OpenTP1 がデッドロックを検知すると，UAP のデッドロックプライオリティを比較して，優先度が低い方の UAP の関数をエラーリターンさせます。

デッドロック時の OpenTP1 の処置については，「3.9.1(5)　デッドロックの対処方法」を参照してください。

## 5.3.3　ファイル障害の対策

次に示す OpenTP1 ファイルに障害が起こったときの対処について説明します。

- システムで使うファイル
- キューを格納するファイル
- ユーザデータを格納するファイル

ファイル障害の対策の詳細については，マニュアル「OpenTP1 運用と操作」の障害対策を参照してください。

### (1)　システムで使うファイルの障害の対策

システムファイルに障害が起こった場合の処置について説明します。

#### (a)　ステータスファイルの障害

オンライン中に障害が起こった場合，使用可能な予備ファイルがあるときは，予備ファイルを現用に切り替えて，OpenTP1 は処理を続けます。予備ファイルがないときは，ステータスサービスが停止して，OpenTP1 は停止します。ただし，ステータスファイルの片系運転可を指定しているときは，予備ファイルがなくても正常な系だけで処理を続けます。停止した場合は，OpenTP1 管理者はエラーの原因に対処したあとで，OpenTP1 を再開始してください。

OpenTP1 の開始時に障害が起こった場合は，ステータスサービス定義の sts_initial_error_switch オペランドの指定によって，OpenTP1 の開始処理を続けるかまたは停止するかが決まります。

#### (b)　システムジャーナルファイルの障害

スワップ先にできるファイルグループがある場合は，OpenTP1 は待機のファイルグループを現用に切り替えます。そして，障害が起こったファイルグループを予約状態にして，OpenTP1 は処理を続けます。

システムジャーナルファイルにスワップ先にできるファイルグループがない場合は，OpenTP1 は停止します。停止した場合は，OpenTP1 管理者はエラーの原因に対処してから，OpenTP1 を再開始してください。

### (c) チェックポイントダンプファイルの障害

上書きできるほかのファイルグループがある場合は，OpenTP1 はそのファイルグループを現用に切り替えます。そして，障害が起こったファイルグループを予約状態にして，OpenTP1 は処理を続けます。

上書きできるほかのファイルグループがない場合は，OpenTP1 は停止します。ただし，チェックポイントダンプファイルを二重化して片系運転可を指定しているときは，予備ファイルがなくても正常なファイルグループだけで処理を続けます。停止した場合は，OpenTP1 管理者はエラーの原因に対処してから，OpenTP1 を再開始してください。

### (d) アーカイブジャーナルファイルの障害

スワップ先にできるファイルグループがある場合は，OpenTP1 はそのファイルグループを現用に切り替えます。そして，障害が起こったファイルグループを予約状態にして，OpenTP1 は処理を続けます。

スワップ先にできるファイルグループがない場合は，OpenTP1 は停止します。停止した場合は，OpenTP1 管理者はエラーの原因に対処してから，OpenTP1 を再開始してください。

### (e) トランザクションリカバリジャーナルファイルまたはサーバリカバリジャーナルファイルの障害

TRF，または SRF で障害が起こると，そのことを知らせるエラーメッセージが出力されます。OpenTP1 管理者は，出力されたエラーメッセージを参照して，リカバリジャーナルファイルを回復するコマンド（jnlmkrf コマンド）を実行してください。

jnlmkrf コマンドを実行してもファイルを回復できない場合は，OpenTP1 管理者はコマンド（damfrc，tamfrc コマンドなど）でリソースマネジャを回復したあとで，OpenTP1 を強制的に正常開始してください。

### (f) メッセージログファイルの障害

メッセージログファイルは二つのファイルを使っています（dclog1，dclog2）。どちらか一つのファイルに障害が起こった場合，OpenTP1 は障害のファイルを切り離して正常なファイルだけでメッセージログを取得します。両方のファイルに障害が起こった場合は，メッセージログをファイルに出力しないで OpenTP1 は処理を続けます。

### (g) ノードリストファイルの障害

オンライン中に障害が起こった場合，OpenTP1 はノードリストファイルへの書き込みをしないで，処理を継続します。したがって，次回オンライン時にノード情報を引き継げません。OpenTP1 管理者はエラーの原因に対処してからノードリストファイルを作成し直し，OpenTP1 を再開始してください。

## (2) キューを格納するファイルの障害の対策

キューを格納するファイルに障害が起こった場合の処置について説明します。

### (a) メッセージキューファイルの障害

メッセージキューファイルをオープンできない場合は，そのことを知らせるエラーメッセージが出力されます。エラーメッセージが出力された場合は，OpenTP1 管理者は出力されたエラーメッセージを参照して，その指示に従って対処してください。

### (b) MQA キューファイルの障害

TP1/Message Queue のキューファイルに障害が起こった場合の処置については，マニュアル「TP1/Message Queue 使用の手引」の障害対策を参照してください。

## (3) ユーザデータを格納するファイルの障害の対策

ユーザファイルに障害が起こった場合の処置について説明します。

### (a) DAM ファイルの障害

DAM サービスは，DAM ファイルにアクセスした UAP にエラーを返します。その後 DAM ファイルは障害閉塞されます。DAM ファイルを回復する必要がある場合は，OpenTP1 管理者は DAM ファイルを OpenTP1 から切り離して，エラーの原因に対処してください。

### (b) TAM ファイルの障害

TAM サービスは，TAM テーブルにアクセスした UAP にエラーを報告します。その後 TAM テーブルは障害閉塞されます。TAM ファイルを回復する必要がある場合は，OpenTP1 管理者は TAM テーブルを OpenTP1 から切り離して，エラーの原因に対処してください。

### (c) ISAM ファイルの障害

ISAM ファイルに障害が起こった場合の処置については，マニュアル「索引順編成ファイル管理 ISAM」の障害対策を参照してください。

### (d) データベースマネジメントシステムの障害

DBMS に障害が起こった場合の処置については，該当する DBMS のマニュアルの障害対策を参照してください。

## 5.3.4　ネットワーク障害の対策

### (1)　通信制御装置，端末，回線の障害

コネクションに障害が起こった場合，エラーメッセージが出力されます。OpenTP1 管理者は，エラーメッセージの内容を参照してエラーの原因に対処してください。対処後，コマンドを実行してコネクションを再び確立してください。

障害時の処置は，通信プロトコル対応製品で異なります。障害が起こったときの処置については，マニュアル「OpenTP1 プロトコル」の該当するプロトコル編を参照してください。

### (2)　LAN の障害

LAN に障害が起こると，ノードをわたる RPC の関数はクライアント UAP にエラーリターンします。各システムの管理者は，OS，およびハードウェアの障害時の処置に従って対処したあとに，システムを再実行してください。

### (3)　マルチノード機能の構成での通信障害

グローバルアーカイブジャーナルサービスと被アーカイブジャーナルノードのジャーナルサービスとの間で通信障害が起こった場合は，エラーメッセージが出力されます。OpenTP1 管理者は，エラーメッセージの内容を参照してエラーの原因に対処してください。

上記の通信障害が起こった場合，アーカイブは実行されません。障害が対処されない状態のときにアンロードする必要がある場合は，被アーカイブジャーナルノードで個別にアンロードしてください。

### (4)　ノードリストファイル障害

ノード自動追加機能使用時に，ノードリストファイル障害が発生した場合，KFCA33615-E メッセージまたは，KFCA33622-E メッセージを出力後，ノードリストファイルへのアクセスを停止し，オンライン処理を続行します。

ノードリストファイルへのアクセスを停止後，OpenTP1 再起動時※に障害が回復し，正常にノードリストファイルを引き継げた場合，障害中に共用メモリ上で更新されたノード情報は，ノードリストファイルへ反映されていないため，前回終了時のノード情報を正しく引き継げないことがあります。

このような場合，OpenTP1 では，オンライン後にノードリストの整合性確保によって，最新のノードリスト情報でノードリストを再構築します。

注※

OpenTP1 管理者はエラーの原因に対処してから，OpenTP1 を再開始してください。

## 5.3.5　OpenTP1 の内部監視

OpenTP1 は，システムの障害を検知するため，次の表に示す項目を監視しています。

### 表 5-4　OpenTP1 監視項目一覧

| 監視項目 | 監視内容 | 関連するシステム定義 |
|---|---|---|
| システム初期化時間 | dcstart コマンドを入力してからシステム初期化処理が終わるまでの待ち時間を監視します。指定時間を過ぎてもシステム初期化処理が終わらない場合には，user_command などの不良と見なしてシステム開始処理を中止します。 | システム環境定義 |
| RPC 応答時間 | RPC のサービス要求に対して，応答が返るまでの待ち時間を監視します。指定時間を過ぎても応答がない場合，サーバ UAP のループなどによる送受信タイムアウトとして要求元にエラーリターンします。 | システム共通定義 |
| 連鎖 RPC 間隔 | サーバ UAP がサービス要求元にリターンしてから，次の RPC かトランザクション決着指示が来るまでの時間を監視します。指定時間を過ぎても RPC や決着指示が来ない場合，サービス要求元の不良と見なして，サーバ UAP のプロセスを異常終了させます。 | ユーザサービス定義<br>ユーザサービス<br>デフォルト定義 |
| デッドロック | 排他要求が待ち状態になってから解除されるまでの待ち時間を監視します。指定時間を過ぎても待ち状態が解除されない場合，デッドロックと見なして，要求元にエラーリターンします。 | ロックサービス定義 |
| トランザクションブランチ限界経過時間 | トランザクションブランチの処理時間を監視します。<br>指定時間を過ぎても処理が終わらない場合，サーバ UAP の不良と見なして，このトランザクションブランチのプロセスを異常終了させて，ロールバックします。 | ユーザサービス定義<br>ユーザサービス<br>デフォルト定義<br>トランザクションサービス定義 |
| CPU 時間 | トランザクション処理を開始してから同期点処理をするまでにトランザクションブランチが使用できる CPU 時間を監視します。指定した時間を過ぎてもトランザクションが完了しない場合は，該当するトランザクションブランチのプロセスを異常終了させてロールバックします。 | ユーザサービス定義<br>ユーザサービスデフォルト定義<br>トランザクションサービス定義 |
| UAP 連続異常終了限界経過時間 | UAP の連続異常終了の時間を監視します。<br>ユーザサービス定義での閉塞指定にかかわらず UAP が指定時間内に 3 回続けて異常終了した場合，該当するサービスグループまたはサービスを閉塞します。 | ユーザサービス定義<br>ユーザサービスデフォルト定義 |
| サービス要求滞留値 | スケジュールキュー内に滞留しているサービス要求数を監視します。（この指定値×起動プロセス数）を超えると，非常駐プロセスを起動してサービスを処理します。 | ユーザサービス定義<br>ユーザサービスデフォルト定義 |
| 入出力キュー使用率 | 入出力キューのキューグループごとに，物理ファイルの使用率を監視します。指定した使用率を超えた場合，警告メッセージを出力します。 | メッセージキューサービス定義 |

| 監視項目 | 監視内容 | 関連するシステム定義 |
|---|---|---|
| 通信関数発行回数 | UAP の暴走監視のためトランザクション内で発行する通信関数の発行回数を監視します。指定値を超えて通信関数を発行した場合，UAP を異常終了させてロールバックします。 | MCF マネジャ定義 |
| 未処理送信メッセージ滞留時間 | 正常終了処理が長時間終了しないのを防止するため，出力キュー中の未処理送信メッセージの滞留時間を監視します。指定値を超えても未処理送信メッセージが残っている場合，ERREVTA を通知したあとで未処理送信メッセージを破棄して，正常終了処理を続行します。 | MCF 通信構成定義 |

## 5.3.6　障害の原因解析

ここでは，OpenTP1 システムで障害の原因を解析するための機能について説明します。

なお，次に示す機能は，TP1/Extension 1 をインストールしていることが前提です。TP1/Extension 1 をインストールしていない場合の動作は保証できません。

- 性能検証用トレース
- XAR 性能検証用トレース
- JNL 性能検証用トレース
- LCK 性能検証用トレース
- MCF 性能検証用トレース
- TRN イベントトレース
- NAM イベントトレース
- プロセスサービスのイベントトレース
- FIL イベントトレース

### (1)　MCF トレース

MCF のプロセスごとに，発生したイベントや送受信データの情報を取得しています。これを**MCF トレース**といいます。MCF トレース情報は，共用メモリ中のトレース領域（トレースバッファ）に取得されます。MCF 通信構成定義で MCF トレースのディスク出力機能を使うことを指定しておくと，トレース領域に空きがなくなった場合に，ディスク上の**MCF トレースファイル**に出力されます。MCF トレースのディスク出力機能を使用しない場合で，一時的に MCF トレース情報が必要になったときは，MCF トレース取得開始コマンド（mcftstrtr），および MCF トレース取得終了コマンド（mcftstptr）を使用すると，MCF トレース情報を MCF トレースファイルに出力できます。コマンドの詳細については，マニュアル「OpenTP1 運用と操作」を参照してください。

MCF トレースは，障害発生までのプロセス内での制御関数と各イベントの制御の流れを解析するために利用します。

## (2) UAP トレース

OpenTP1 の UAP（SUP，SPP，MHP）が異常終了すると，UAP から呼び出した API のトレース情報が出力されます。この機能を **UAP トレース**といい，取得した情報を **UAP トレース情報**といいます。OpenTP1 では，UAP プロセス単位に UAP トレース情報を取得しています。

UAP トレース情報は，$DCDIR/spool/save/ディレクトリの下に，ファイル名**"サーバ名 n.uat"**（n は UAP プロセスの退避コアファイルに付けられる通番を示します）で取得されます。このファイルを，**UAP トレース編集出力ファイル**といいます。

任意の退避コアファイルを基に UAP トレース情報を出力する場合は，UAP が異常終了したときに生成した退避コアファイルを **uatdump コマンド**に指定して実行します。uatdump コマンドの出力先をリダイレクトすると，任意のファイルに UAP トレース情報を取得できます。

UAP トレース情報を格納する件数の最大数は，ユーザサービス定義の uap_trace_max オペランドに指定します。ここに指定した値を超えた場合は，ラップアラウンドして取得します。

UAP トレース情報は，異常終了した **UAP のプロセス単位**に取得されます。RPC でノードをわたるトランザクション処理が異常終了した場合は，UAP トレース情報はノードで実行した UAP プロセスごとに取得されます。

UAP トレースについては，マニュアル「OpenTP1 テスタ・UAP トレース使用の手引」の UAP トレースの説明を参照してください。

## (3) RPC トレース

RPC のサービス要求情報をファイルにトレースとして取得できます。これを **RPC トレース**といいます。RPC トレースは，RPC 管理コマンドでダンプ出力できます。RPC トレースは，次の目的に利用できます。

- どのクライアント UAP とどのサーバ UAP が，RPC でプロセス間通信をしたか調査します。
- 特定のサーバ UAP で，どの順番にサービスを処理したか調査します。
- RPC に障害が発生した場合に，障害発生までのサービス要求の流れを解析します。

RPC トレースファイルを編集出力する場合は，**rpcdump コマンド**を実行します。

RPC トレースファイルは，システムサービスごとに取得できます。該当するシステムサービスの定義に，RPC トレースファイルを取得する指定をします。これらの RPC トレースファイルは，マージして時系列に出力できます。この場合は，**rpcmrg コマンド**を実行します。

## (4)　性能検証用トレース

OpenTP1 で動作する各種サービスの主なイベントで OpenTP1 識別子などのトレース情報を取得しています。これを**性能検証用トレース**といいます。

性能検証用トレースには次の特長があります。

- ノードおよびプロセスにわたる場合でもトレースを追うことができます。
- API 単位ではなく内部のイベント単位でトレースを取得するので，どの処理が性能のボトルネックであるか検証できます。

性能検証用トレースに関係するシステム定義は，次の表に示すとおりです。

### 表 5-5　性能検証用トレースに関係するシステム定義

| 定義名 | 形式 | オペランド | 定義内容 |
|---|---|---|---|
| システム共通定義 | set | prf_trace | 性能検証用トレース情報を取得するかどうかを指定 |
| | set | trn_prf_trace_level | トレースの取得レベル |
| 性能検証用トレース定義 | set | prf_file_size | トレースファイルのサイズ |

それぞれの定義の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

なお，性能検証用トレースは **prfget コマンド**によってバイナリ形式で取り出したあと，**prfed コマンド**，または **dcalzprf コマンド**によってキャラクタ形式で出力できます。

また，ユーザ固有のトレース情報を dc_prf_utrace_put 関数によってトレースファイルに取得したり，直前に取得したトレースのプロセス内取得通番を dc_prf_get_trace_num 関数によって取得したりできます。

**注意事項**

- 通常の OpenTP1 再開始，およびホットスタンバイでは，トレース情報は引き継ぎません。
- 性能検証用トレース取得機能では，オンラインの性能に影響がないように，トレース取得での排他を行いません。そのため，マルチプロセッサ環境でトレース取得の競合が発生した場合，トレース情報が抜けたり，不正なトレース情報が取得されたりすることがあります。不正なトレース情報は，prfed コマンドでトレース情報を編集すると，エラーレコードとして表示されます。

## (5)　XAR 性能検証用トレース

XA リソースサービスを使用したトランザクション連携の各種イベント（アプリケーションサーバからのトランザクション要求，OpenTP1 のトランザクション処理）のトレース情報を取得しています。これを**XAR 性能検証用トレース**といいます。

XAR 性能検証用トレースは，$DCDIR/spool/dcxarinf ディレクトリの下に，"_xr_001"，"_xr_002"，"_xr_003"…というファイル名で取得されます。これらのファイルを**XAR 性能検証用トレース情報ファイル**といいます。XAR 性能検証用トレース情報ファイルの出力先およびファイル名は変更できませんが，取得サイズおよび取得ファイル数は変更できます。詳細については，マニュアル「OpenTP1 システム定義」の XAR 性能検証用トレース定義の説明を参照してください。

XAR 性能検証用トレースを取得するには，次の手順で定義を設定します。

1. システム共通定義の prf_trace オペランドに Y を指定します。
2. XA リソースサービス定義の xar_prf_trace_level オペランドで XAR 性能検証用トレースの取得レベルを指定します。

次の表に xar_prf_trace_level オペランドの指定値と XAR 性能検証用トレースの取得情報の関係を示します。

### 表 5-6　xar_prf_trace_level オペランドの指定値と XAR 性能検証用トレースの取得情報の関係

| xar_prf_trace_level オペランドの指定値 | XAR 性能検証用トレースの取得情報 | イベント ID | トレースデータ長（単位：バイト） | 取得タイミング | |
|---|---|---|---|---|---|
| 00000001 | アプリケーションサーバからのトランザクション要求 | 0x4a00 | 128 | トランザクションブランチの開始要求 | 呼び出し直後 |
| | | 0x4a01 | 128 | | リターン直前 |
| | | 0x4a02 | 128 | トランザクションブランチ内からの RPC 実行要求 | 呼び出し直後 |
| | | 0x4a03 | 128 | | リターン直前 |
| | | 0x4a04 | 128 | トランザクションブランチの終了要求 | 呼び出し直後 |
| | | 0x4a05 | 128 | | リターン直前 |
| | | 0x4a06 | 128 | トランザクションブランチのコミット準備要求 | 呼び出し直後 |
| | | 0x4a07 | 128 | | リターン直前 |
| | | 0x4a08 | 128 | トランザクションブランチのコミット要求 | 呼び出し直後 |
| | | 0x4a09 | 128 | | リターン直前 |
| | | 0x4a0a | 128 | トランザクションブランチのロールバック要求 | 呼び出し直後 |

| xar_prf_trace_level オペランドの指定値 | XAR 性能検証用トレースの取得情報 | イベント ID | トレースデータ長（単位：バイト） | 取得タイミング | |
|---|---|---|---|---|---|
| 00000001 | アプリケーションサーバからのトランザクション要求 | 0x4a0b | 128 | トランザクションブランチのロールバック要求 | リターン直前 |
| | | 0x4a0c | 64 | Prepared 状態，Heuristically Completed 状態のトランザクションブランチ通知要求 | 呼び出し直後 |
| | | 0x4a0d | 64 | | リターン直前 |
| | | 0x4a0e | 128 | Heuristically Completed 状態のトランザクションブランチ破棄要求 | 呼び出し直後 |
| | | 0x4a0f | 128 | | リターン直前 |
| 00000002 | OpenTP1 のトランザクション処理 | 0x4b00 | 64 | トランザクションブランチ開始 | 直前 |
| | | 0x4b01 | 64 | | 直後 |
| | | 0x4b02 | 64 | トランザクションブランチ内からの RPC 実行 | 直前 |
| | | 0x4b03 | 64 | | 直後 |
| | | 0x4b04 | 64 | トランザクションブランチの終了 | 直前 |
| | | 0x4b05 | 64 | | 直後 |
| | | 0x4b06 | 64 | トランザクションブランチのコミット準備 | 直前 |
| | | 0x4b07 | 64 | | 直後 |
| | | 0x4b08 | 64 | トランザクションブランチのコミット | 直前 |
| | | 0x4b09 | 64 | | 直後 |
| | | 0x4b0a | 64 | トランザクションブランチのロールバック | 直前 |
| | | 0x4b0b | 64 | | 直後 |
| | | 0x4b0c | 64 | Prepared，Heuristically Completed 状態のトランザクションブランチの通知 | 直前 |
| | | 0x4b0d | 64 | | 直後 |
| | | 0x4b0e | 64 | Heuristically Completed 状態のトランザクションブランチの破棄 | 直前 |
| | | 0x4b0f | 64 | | 直後 |

xar_prf_trace_level オペランドの詳細については，マニュアル「OpenTP1 システム定義」の XA リソースサービス定義の説明を参照してください。

XAR 性能検証用トレース情報ファイルの取得，編集出力には，prfget コマンド，prfed コマンド，または dcalzprf コマンドを使用します。取得方法，および編集出力方法を次に示します。

XAR 性能検証用トレース情報ファイルの取得方法

最新のラン ID で取得されていないトレース情報だけ取得する場合

```
$DCDIR/bin/prfget -f _xr | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -f _xr | $DCDIR/bin/dcalzprf
```

すべてのトレース情報を取得する場合

```
$DCDIR/bin/prfget -a -f _xr | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _xr | $DCDIR/bin/dcalzprf
```

XAR 性能検証用トレース情報ファイルの編集出力方法

XAR 性能検証用トレース情報ファイルからトレース情報を編集出力する場合，prfed コマンド，または dcalzprf コマンドを実行してください。オプションは必要に応じて指定してください。

XAR 性能検証用トレース情報の出力形式は，性能検証用トレースと同じです。出力形式の詳細については，マニュアル「OpenTP1 運用と操作」の prfed コマンド，または dcalzprf コマンドを参照してください。

## (6) JNL 性能検証用トレース

ジャーナルサービスで実行されるジャーナルバッファリング，およびジャーナル出力の各種イベントトレース情報を取得しています。これを **JNL 性能検証用トレース**といいます。

JNL 性能検証用トレースは，$DCDIR/spool/dcjnlinf/prfinf ディレクトリの下に，"_jl_001"，"_jl_002"，"_jl_003"…というファイル名で取得されます。これらのファイルを **JNL 性能検証用トレース情報ファイル**といいます。JNL 性能検証用トレース情報ファイルの出力先およびファイル名は，変更できません。

JNL 性能検証用トレースを取得するには，次の手順で定義を設定します。

1. システム共通定義の prf_trace オペランドに Y を指定します。
2. システム共通定義の jnl_prf_event_trace_level オペランドで JNL 性能検証用トレースの取得レベルを指定します。

次の表に jnl_prf_event_trace_level オペランドの指定値と取得するトレース情報の関係を示します。各イベントの取得タイミングについては，マニュアル「OpenTP1 運用と操作」の性能検証用トレース情報の取得の説明を参照してください。

表 5-7　jnl_prf_event_trace_level オペランドの指定値と取得するトレース情報の関係

| jnl_prf_event_trace_level オペランドの指定値 | トレース情報のイベント ID | |
|---|---|---|
| | 0xc202，0xc203，0xc401，0xc402 | 0xc001～0xc201，0xc204～0xc400 |
| 00000000 | × | × |
| 00000001 | ○ | × |
| 00000002 | ○ | ○ |
| その他 | ○ | ○ |

（凡例）

○：トレース情報を取得します。

×：トレース情報を取得しません。

JNL 性能検証用トレース情報ファイルの取得，編集出力には，prfget コマンド，prfed コマンド，または dcalzprf コマンドを使用します。取得方法を次に示します。

JNL 性能検証用トレース情報ファイルの取得方法

最新のラン ID で取得されていないトレース情報だけ取得する場合

```
$DCDIR/bin/prfget -f _jl | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -f _jl | $DCDIR/bin/dcalzprf
```

すべてのトレース情報を取得する場合

```
$DCDIR/bin/prfget -a -f _jl | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _jl | $DCDIR/bin/dcalzprf
```

## (7)　LCK 性能検証用トレース

トランザクション処理に伴う各種排他制御のトレース情報を取得しています。これを **LCK 性能検証用トレース**といいます。

LCK 性能検証用トレースは，$DCDIR/spool/dclckinf/prf ディレクトリの下に，"_lk_001"，"_lk_002"，"_lk_003"…というファイル名で取得されます。これらのファイルを **LCK 性能検証用トレース情報ファイル**といいます。LCK 性能検証用トレース情報ファイルの出力先およびファイル名は変更できませんが，取得サイズおよび取得ファイル数は変更できます。詳細については，マニュアル「OpenTP1 システム定義」の LCK 性能検証用トレース定義の説明を参照してください。

LCK 性能検証用トレースを取得するには，次の手順で定義を設定します。

1. システム共通定義の prf_trace オペランドに Y を指定します。
2. ロックサービス定義の lck_prf_trace_level オペランドで LCK 性能検証用トレースの取得レベルを指定します。

次の表に lck_prf_trace_level オペランドの指定値と LCK 性能検証用トレースの取得情報の関係を示します。

### 表 5-8　lck_prf_trace_level オペランドの指定値と LCK 性能検証用トレースの取得情報の関係

| lck_prf_trace_level オペランドの指定値 | LCK 性能検証用トレースの取得情報 | イベント ID | トレースデータ長（単位：バイト） | 取得タイミング | |
|---|---|---|---|---|---|
| 00000000 | 排他制御についてのトレースを取得しない | － | － | － | |
| 00000001 | 排他制御についてのトレースを取得する | 0x6400 | 128 | 資源の排他 | 呼び出し直後 |
| | | 0x6401 | 128 | | リターン直前 |
| | | 0x6410 | 128 | 排他待ち合せ | 開始直前 |
| | | 0x6411 | 128 | | 終了直後 |
| | | 0x6420 | 128 | 全資源の排他の解除 | 呼び出し直後 |
| | | 0x6421 | 128 | | リターン直前 |
| | | 0x6430 | 128 | 資源名称を指定した排他の解除 | 呼び出し直後 |
| | | 0x6431 | 128 | | リターン直前 |

（凡例）

－：該当しません。

lck_prf_trace_level オペランドの詳細については，マニュアル「OpenTP1 システム定義」のロックサービス定義の説明を参照してください。

LCK 性能検証用トレース情報ファイルの取得，編集出力には，prfget コマンド，prfed コマンド，または dcalzprf コマンドを使用します。取得方法，および編集出力方法を次に示します。

LCK 性能検証用トレース情報ファイルの取得方法

最新のラン ID で取得されていないトレース情報だけ取得する場合

```
$DCDIR/bin/prfget -f _lk | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -f _lk | $DCDIR/bin/dcalzprf
```

すべてのトレース情報を取得する場合

```
$DCDIR/bin/prfget -a -f _lk | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _lk | $DCDIR/bin/dcalzprf
```

LCK 性能検証用トレース情報ファイルの編集出力方法

LCK 性能検証用トレース情報ファイルからトレース情報を編集出力する場合，prfed コマンド，または dcalzprf コマンドを実行してください。オプションは必要に応じて指定してください。

LCK 性能検証用トレース情報の出力形式は，性能検証用トレースと同じです。出力形式の詳細については，マニュアル「OpenTP1 運用と操作」の prfed コマンド，または dcalzprf コマンドを参照してください。

## (8) MCF 性能検証用トレース

TP1/Message Control を使用したメッセージ送受信での主なイベントで，MCF 識別子などのトレース情報を取得しています。これを**MCF 性能検証用トレース**といいます。

MCF 性能検証用トレースには次の特長があります。

- スレッド ID や論理端末名称，API 名称など，MCF 固有の詳細情報を出力できます。
- 出力した詳細情報をキーとして MCF 性能検証用トレースを分析することで，一連のメッセージ送受信処理や UAP の性能検証ができます。

MCF 性能検証用トレースに関係するシステム定義は，次の表に示すとおりです。

### 表 5-9 MCF 性能検証用トレースに関係するシステム定義

| 定義名 | 形式 | オペランド | 定義内容 |
|---|---|---|---|
| ユーザサービス定義 | set | mcf_prf_trace | ユーザサーバごとに，MCF 性能検証用トレース情報を取得するかどうかを指定 |
| MCF 性能検証用トレース定義 | set | prf_file_size | MCF 性能検証用トレース情報のトレースファイルサイズ |
| | set | prf_file_count | MCF 性能検証用トレース情報のトレースファイル世代数 |
| システムサービス情報定義 | set | mcf_prf_trace | MCF 通信サービスごとに MCF 性能検証用トレース情報を取得するかどうかを指定 |
| システムサービス共通情報定義 | set | mcf_prf_trace_level | MCF 性能検証用トレース情報の取得レベル |

それぞれの定義の詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

なお，MCF 性能検証用トレースは prfget コマンドによってバイナリ形式で取り出したあと，prfed コマンド，または dcalzprf コマンドによってキャラクタ形式で出力できます。

**注意事項**

- メッセージ送受信時（イベント ID：0xa000，0xa001）に MCF 性能検証用トレースを取得できるのは，次のどれかのプロトコル製品を使用した通信だけです。

  ・TP1/NET/TCP/IP

  ・TP1/NET/XMAP3

  ・TP1/NET/OSAS-NIF

  メッセージ送受信（イベント ID：0xa000，0xa001）以外のイベントについては，通信プロトコル種別に関係なく，MCF 性能検証用トレースを取得できます。

## (9) XAR イベントトレース

XA リソースサービスを使用したアプリケーションサーバからのトランザクション要求種別を，イベントトレース情報として取得します。この機能を**XAR イベントトレース**といい，取得した情報を**XAR イベントトレース情報**といいます。

アプリケーションサーバからのトランザクション要求種別，要求コードおよび要求コードの意味を次の表に示します。

表 5-10　トランザクション要求種別と要求コードの一覧

| 要求種別※ | 要求コード | 要求コードの意味 |
|---|---|---|
| Start() | xar_start | トランザクションブランチの開始処理 |
| Call() | xar_call | トランザクションブランチ内からの RPC の実行 |
| End() | xar_end | トランザクションブランチの終了処理 |
| Prepare() | xar_prepare | トランザクションブランチのコミット準備処理（2 相コミットの 1 相目） |
| Commit() | xar_commit | トランザクションブランチのコミット処理（2 相コミットの 2 相目） |
| Rollback() | xar_rollback | トランザクションブランチのロールバック処理 |
| Recover() | xar_recover | Prepared 状態，Heuristically Completed 状態のトランザクションブランチを通知 |
| Forget() | xar_forget | Heuristically Completed 状態のトランザクションブランチを破棄 |

注※

トランザクション要求種別は，OpenTP1 の内部関数です。

XAR イベントトレース情報は，$DCDIR/spool/dcxarinf/trace/ディレクトリの下に，ファイル名"xarevtr1"および"xarevtr2"で取得されます。これらのファイルを，**XAR イベントトレース情報ファイル**といいます。XAR イベントトレース情報ファイルの出力先およびファイル名は，変更できません。

XA リソースサービス開始時に，XAR イベントトレース情報ファイルがすでに存在する場合は，バックアップファイルが作成され，新規に出力ファイルが作成されます。例えば，xarevtr1 という XAR イベン

トトレース情報ファイルが存在した場合，XA リソースサービス開始時に，xarevtr1 を xarevtr1.bk1 というファイルにリネームします。xarevtr1 という名前のファイルは，リネームされて存在しなくなるため，新規に xarevtr1 という XAR イベントトレース情報ファイルが作成されます。

バックアップファイルは 3 世代まで保存されます。バックアップファイルが作成されるのは，XA リソースサービス開始時に，XAR イベントトレース情報ファイルがすでに存在する場合だけです。オンライン中に，1 ファイルへの書き込みレコード数が，XAR イベントトレース情報ファイルの最大出力レコード数（XA リソースサービス定義の xar_eventtrace_record オペランドで指定）を超えた場合には，バックアップファイルは作成されません。オンライン中に，1 ファイルへの書き込みレコード数が，XAR イベントトレース情報ファイルの最大出力レコード数を超えると，出力先ファイルを切り替え，XAR イベントトレース情報ファイルを順次上書きしていきます。オンライン中は XAR イベントトレース情報ファイルを削除しないでください。

XAR イベントトレース情報の出力レベルは，XA リソースサービス定義の xar_eventtrace_level オペランドで指定できます。デフォルトは，エラーが発生した場合だけ XAR イベントトレースを取得する出力レベルに指定されています。すべての XAR イベントトレース情報を取得することもできますが，その場合オンライン性能に影響を与えるため，デバッグ時以外は，デフォルトの出力レベルで運用することをお勧めします。

出力ファイルに取得された XAR イベントトレース情報は，xarevtr コマンドによって編集，表示できます。

XAR イベントトレース情報の出力レベルおよび出力レコード数の指定方法についてはマニュアル「OpenTP1 システム定義」を，編集・表示方法についてはマニュアル「OpenTP1 運用と操作」を参照してください。

## (10) TRN イベントトレース

トランザクションブランチで発行される XA 関数や，トランザクションサービス（トランザクション管理サービス，トランザクション回復サービス，リソースマネジャ監視サービス）の各種イベントのトレース情報を取得しています。これを **TRN イベントトレース**といいます。

TRN イベントトレースは，$DCDIR/spool/dctrninf/trace/prf/ディレクトリの下に，"_tr_001"，"_tr_002"，"_tr_003"…というファイル名で取得されます。これらのファイルを，**TRN イベントトレース情報ファイル**といいます。TRN イベントトレース情報ファイルの出力先およびファイル名は，変更できません。

TRN イベントトレース情報ファイルは，取得サイズおよび取得ファイル数を変更できます。詳細については，マニュアル「OpenTP1 システム定義」の TRN イベントトレース定義を参照してください。

TRN イベントトレースを取得するには，次の手順で定義を設定します。

1. システム共通定義の prf_trace オペランドに Y を指定します。
2. トランザクションサービス定義の trn_prf_event_trace_level オペランドで TRN イベントトレースの取得レベルを指定します。
3. トランザクションサービス定義の trn_prf_event_trace_condition オペランドで，TRN イベントトレースの取得種別を指定します。

次の表に trn_prf_event_trace_condition オペランドの指定値と TRN イベントトレースの取得情報の関係を示します。

### 表 5-11　trn_prf_event_trace_condition オペランドの指定値と TRN イベントトレースの取得情報の関係

| trn_prf_event_trace_condition オペランドの指定値 | TRN イベントトレースの取得情報 | イベント ID | トレースデータ長（単位：バイト） | 取得タイミング |
|---|---|---|---|---|
| xafunc | XA 関数に関するトレース | 0x4500 | 320（xa_open 関数，xa_close 関数の場合は 192） | トランザクション処理中※ |
| trnservice | トランザクションサービスの動作状況に関するトレース | 0x4501 | 192 | トランザクションサービスの起動，停止または回復処理時 |

注※

2 相コミットのトランザクションの場合，1 トランザクションブランチ当たりに取得するトレース量は「12×リソースマネジャ数」となります。ただし，トレース量は，ユーザサーバにリンケージされている XA インタフェースオブジェクトファイルや，トランザクションの最適化などの条件によって異なります。

このオペランドの詳細については，マニュアル「OpenTP1 システム定義」のシステムサービス定義を参照してください。

TRN イベントトレース情報ファイルの取得，編集出力には，prfget，prfed，または dcalzprf コマンドを使用します。取得方法，および編集出力方法を次に示します。

TRN イベントトレース情報ファイルの取得方法

最新のラン ID で取得されていないトレース情報だけ取得する場合

```
$DCDIR/bin/prfget -f _tr | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -f _tr | $DCDIR/bin/dcalzprf
```

すべてのトレース情報を取得する場合

```
$DCDIR/bin/prfget -a -f _tr | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _tr | $DCDIR/bin/dcalzprf
```

TRN イベントトレース情報ファイルの編集出力方法

TRN イベントトレース情報を出力する場合，prfed コマンドに-d オプションを指定してください。dcalzprf コマンドには-d オプションの指定は必要ありません。そのほかのオプションは，必要に応じて指定してください。

TRN イベントトレース情報の出力形式は，性能検証用トレースと同じです。出力形式の詳細については，マニュアル「OpenTP1 運用と操作」の prfed コマンドを参照してください。

trn_prf_event_trace_condition オペランドに xafunc を指定した場合の prfed コマンドの TRN イベントトレース情報の出力例を次に示します。

```
PRF: Rec Node: trn1 Run-ID: 0x4046b806 Process: 26264      Trace: 10
Event: 0x4500 Time: 2004/01/01 12:34:56 678.123.000 Server-name: Sup
Rc:   0      Client: **** - ********** Server: **** Root: trn1 - ********
Svc-Grp: ************************** Svc: ****************************
Trn: 78d0trn100000001trn1trn100000001
    xa_commit                       (IN)  OpenTP1_TAM
    axid:010000000000000c00000010
         3738643074726e330000000174726e3374726e330000000100000000
  Internal code1:0x2007       Internal code2:   0
  Internal code3:0
```

## (11) NAM イベントトレース

ネームサービスで実行される通信処理，キャッシュへのサービス情報の登録，削除などの各種イベントのトレース情報を取得しています。これを**NAM イベントトレース**といいます。

NAM イベントトレースは，$DCDIR/spool/dcnaminf/ディレクトリの下に，"_nm_001"，"_nm_002"，"_nm_003"…というファイル名で取得されます。これらのファイルを，**NAM イベントトレース情報ファイル**といいます。NAM イベントトレース情報ファイルの出力先およびファイル名は，変更できません。

NAM イベントトレースを取得するには，次の手順で定義を設定します。

1. システム共通定義の prf_trace オペランドに Y を指定します。
2. システム共通定義の nam_prf_trace_level オペランドで NAM イベントトレースの取得レベルを指定します。

次の表に nam_prf_trace_level オペランドの指定値と取得するトレース情報の関係を示します。各イベントの取得タイミングについては，マニュアル「OpenTP1 運用と操作」の性能検証用トレース情報の取得の説明を参照してください。

### 表 5-12　nam_prf_trace_level オペランドの指定値と取得するトレース情報の関係

| nam_prf_trace_level オペランドの指定値 | トレース情報のイベント ID | | |
|---|---|---|---|
| | 0xf000～0xf035 | 0xf100～0xf114 | 0xf200～0xf220 |
| 00000000 | × | × | × |
| 00000001 | × | ○ | × |
| 00000002 | ○ | × | × |
| 00000003 | ○ | ○ | × |
| 00000004 | × | × | ○ |

| nam_prf_trace_level オペランドの指定値 | トレース情報のイベント ID | | |
|---|---|---|---|
| | 0xf000～0xf035 | 0xf100～0xf114 | 0xf200～0xf220 |
| 00000005 | × | ○ | ○ |
| 00000006 | ○ | × | ○ |
| 00000007 | ○ | ○ | ○ |
| その他 | ○ | ○ | × |

（凡例）

○：トレース情報を取得します。

×：トレース情報を取得しません。

NAM イベントトレース情報ファイルの取得，編集出力には，prfget，prfed コマンド，または dcalzprf コマンドを使用します。取得方法を次に示します。

NAM イベントトレース情報ファイルの取得方法

```
$DCDIR/bin/prfget -a -f _nm | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _nm | $DCDIR/bin/dcalzprf
```

## (12) プロセスサービスのイベントトレース

プロセスの生成，消滅，起動，完了などのプロセスイベント情報を取得します。これを**プロセスサービスのイベントトレース**といいます。

プロセスサービスのイベントトレースは，$DCDIR/spool/dcprcinf/ディレクトリの下に，"_pr_001"，"_pr_002"，"_pr_003"…というファイル名で取得されます。これらのファイルを，**プロセスサービスイベントトレース情報ファイル**といいます。プロセスサービスイベントトレース情報ファイルの出力先およびファイル名は，変更できません。

プロセスサービスのイベントトレースを取得するには，プロセスサービス定義の prc_prf_trace オペランドに Y を指定します。

各イベントの取得タイミングについては，マニュアル「OpenTP1 運用と操作」の性能検証用トレース情報の取得の説明を参照してください。

プロセスサービスイベントトレース情報ファイルの取得，編集出力には，prfget，prfed，または dcalzprf コマンドを使用します。取得方法を次に示します。

プロセスサービスイベントトレース情報ファイルの取得方法

```
$DCDIR/bin/prfget -a -f _pr | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _pr | $DCDIR/bin/dcalzprf
```

## (13) FIL イベントトレース

OpenTP1 制御下のプロセスから内部的に発行される OpenTP1 ファイルへのアクセス要求に対して，処理完了までに，システム共通定義の fil_prf_trace_delay_time オペランドの指定値以上の処理時間が掛かった場合，イベント情報を取得します。これを**FIL イベントトレース**といいます。

FIL イベントトレースを分析することで，OpenTP1 ファイルアクセスに関する処理の遅延状況を確認できます。

FIL イベントトレースは，$DCDIR/spool/dcfilinf/ディレクトリの下に，"_fl_001"，"_fl_002"，"_fl_003"というファイル名で取得されます。これらのファイルを**FIL イベントトレース情報ファイル**といいます。FIL イベントトレース情報ファイルの出力先およびファイル名は，変更できません。

FIL イベントトレース情報ファイルの取得，編集出力には，prfget コマンド，prfed コマンド，または dcalzprf コマンドを使用します。取得方法を次に示します。

FIL イベントトレース情報ファイルの取得方法

```
$DCDIR/bin/prfget -a -f _fl | $DCDIR/bin/prfed -d
```

または

```
$DCDIR/bin/prfget -a -f _fl | $DCDIR/bin/dcalzprf
```

## (14) コマンドログ

OpenTP1 の運用コマンドを実行した場合に，コマンド実行時刻や終了時刻などの情報を$DCDIR/spool/cmdlog の cmdlog1，および cmdlog2 に出力します。

cmdlog1，および cmdlog2 は，vi エディタなどで参照できます。コマンドログにはコマンドの開始，終了時刻が出力されるので，コマンドの実行に必要な時間（レスポンスタイム）の測定などができます。

# 6 複数の OpenTP1 を使用する場合の機能

この章では，OpenTP1 を大規模システムに適用するための機能（系切り替え機能，マルチノード機能），マルチ OpenTP1，およびマルチホームドホスト形態を使うときの対応について説明します。

## 6.1　系切り替え機能

OpenTP1 システムの稼働率を上げるため，OpenTP1 システムを二重化できます。

OpenTP1 を二重化するためには，**HA モニタ**または **Hitachi HA Toolkit Extension** が必要です。HA モニタについては，マニュアル「高信頼化システム監視機能 HA モニタ」を，Hitachi HA Toolkit Extension については，マニュアル「Hitachi HA Toolkit」を参照してください。

以降，「HA モニタ」には Hitachi HA Toolkit Extension を含みます。

### 6.1.1　系切り替え機能の概要

OpenTP1（TP1/Server Base），CPU，およびカーネルなどから構成されるサーバシステム（**系**）を二つ準備して，一つの系に障害が発生したときに，準備しておいた系に業務処理を速やかに切り替える機能を**系切り替え機能**といいます。この切り替えには，オペレータの操作を必要としません。

系切り替え後には，障害が発生した系の OpenTP1 を再開始させて，次の切り替えに備えることができます。

オンライン中に業務処理をしている系を**実行系**，待機している系を**待機系**といいます。オンライン中は，系切り替えが発生するたびに，実行系と待機系が入れ替わります。

OpenTP1 でメッセージ制御機能（MCF）またはメッセージキューイング機能（TP1/Message Queue）を組み込んでいる OpenTP1 システムでも，系切り替え構成にできます。

#### (1)　系切り替えの種類

系切り替えには，次の 2 種類があります。※

- **自動系切り替え**
  実行系に障害が発生した場合に，自動的に待機系に処理を切り替えます。
- **計画系切り替え**
  実行系で HA モニタのコマンドを実行して，意図的に待機系に切り替えます。

注※

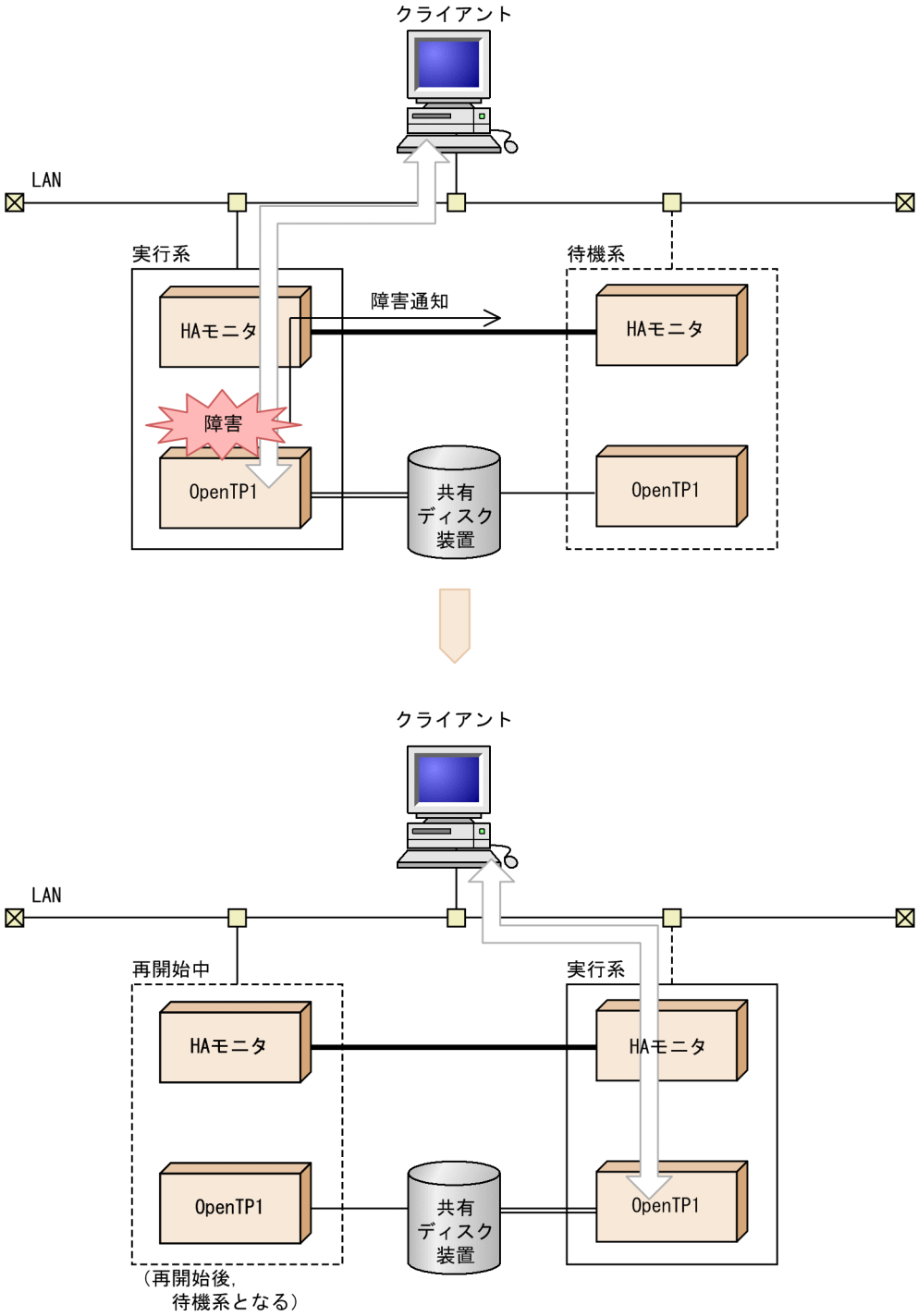
系切り替え機能には，上記以外に**連動系切り替え**という形態があります。連動系切り替えとは，系にある複数の製品をグループ化して，一括して切り替えるようにした系切り替えです。連動系切り替えを使うと，グループ化した製品を常に同じ実行系で稼働させることができます。連動系切り替えでは，系が切り替わるタイミングが自動系切り替えおよび計画系切り替えの場合と異なる場合があります。連動系切り替えを使った場合の系切り替えについては，マニュアル「高信頼化システム監視機能 HA モニタ」を参照してください。

## (2)　系を識別する指定

HA モニタの環境設定では，二つの系を区別するため，オンライン開始時にどちらの系を最初に実行系とするかを決めておきます。最初に業務を開始する系を**現用系**，現用系の障害に備えて待機する系を**予備系**といいます。現用系と予備系はオンライン開始時の属性を示す用語です。この定義は，系切り替えが起こっても変わりません。

系切り替えの機能の概要を次の図に示します。

図 6-1　系切り替え機能の概要



## 6.1.2　系切り替え機能を使う OpenTP1 のシステム構成

系切り替え機能を使う場合，現用系と予備系の OpenTP1 システム（TP1/Server Base）に必要なソフトウェアを次に示します。

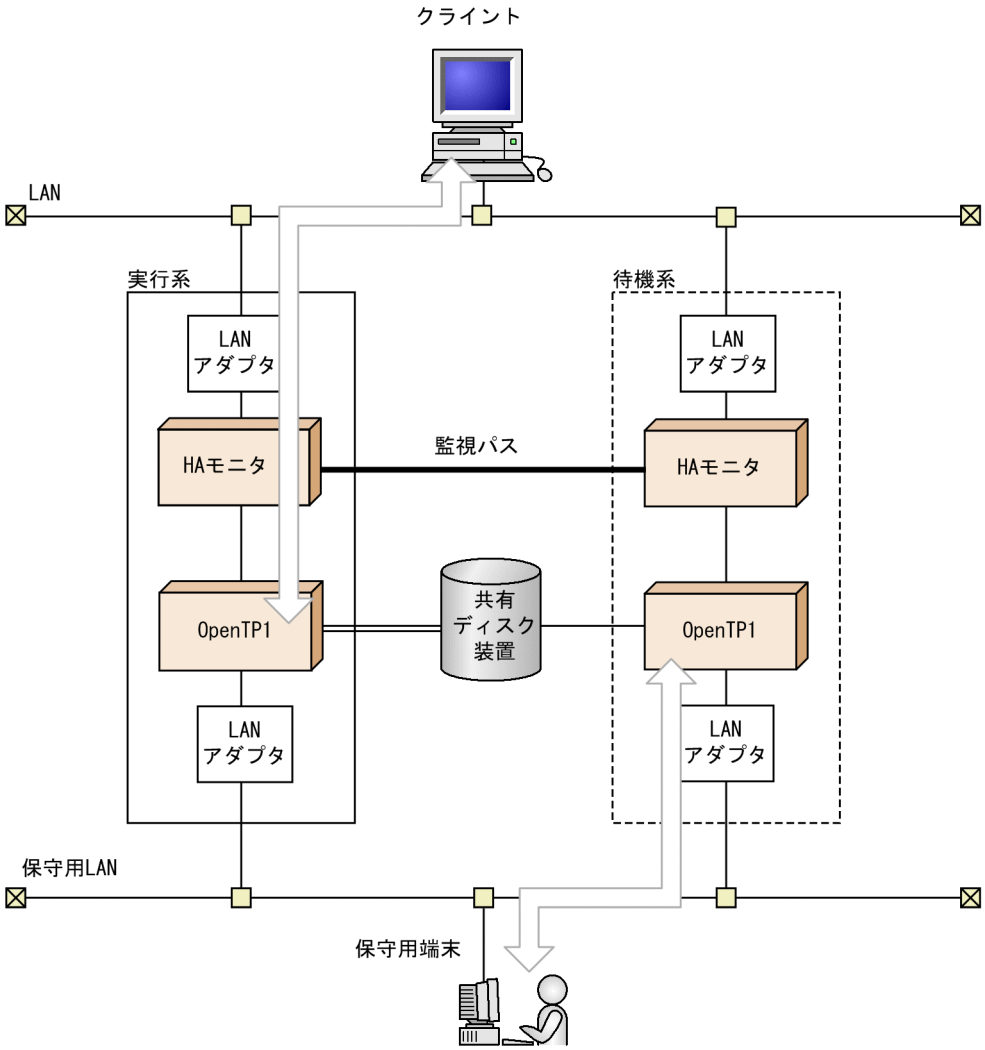
- HA モニタ，TP1/High Availability
  系切り替え機能を使う場合に，必ず前提となる製品です。
- TP1/NET/High Availability
  系切り替え機能とメッセージ制御機能（MCF）を使う場合に前提となる製品です。

現用系と予備系で共通して使うハードウェアを次に示します。

- **共有ディスク装置**
  系切り替え後に，実行系の OpenTP1 から待機系の OpenTP1 へ情報を引き継ぐために使う，ディスク（キャラクタ型スペシャルファイル）です。OpenTP1 ファイルなど，系と系の間で引き継ぐ必要がある情報を格納します。
  共有できる装置はディスク装置などです。
- LAN
  OpenTP1 のネームサービスで使うための LAN です。OpenTP1 の通信で使います。
  この LAN 以外にも，複数の系を保守するための LAN（**保守用 LAN**）が必要です。保守用 LAN は，OpenTP1 の通信には使いません。主に，待機系へのリモートアクセスに使います。
- **監視パス**
  互いの系の状態をやり取りするためのインタフェースです。HA モニタで使います。監視パスには**監視リンク**と**監視専用 LAN** があります。現用系と予備系がそれぞれ一つずつのときは監視リンクを，現用系と予備系の合計が三つ以上のときは監視専用 LAN を使います。

系切り替え機能を使う場合の構成を次の図に示します。次の図に示す以外にも，互いの系を監視するためのハードウェアとソフトウェアが必要になります。必要となる資源については，マニュアル「高信頼化システム監視機能 HA モニタ」を参照してください。

図 6-2　系切り替え機能を使う場合の構成




## 6.1.3　系切り替えの手順

### (1)　系切り替えをするための準備

#### (a)　共有ディスク装置の割り当て

OpenTP1 ファイルシステムを作成するキャラクタ型スペシャルファイルを共有指定します。共有するファイルは，互いの OpenTP1 システムから同じパス名で参照できるように設定します。

共有できるのは，キャラクタ型スペシャルファイルに作成したファイルだけです。通常ファイルの内容（例　$DCDIR/spool/ 下のファイル，$DCDIR/tmp/ 下のファイルなど）は引き継げません。

#### (b)　IP アドレスの設定

共有する IP アドレスを，システム共通定義の my_host オペランドに設定します。互いの OpenTP1 システムには，同じ IP アドレスを設定します。

### (c) HA モニタの環境設定

HA モニタの定義を設定します。HA モニタの環境設定については，マニュアル「高信頼化システム監視機能 HA モニタ」を参照してください。

## (2) HA モニタに定義する内容

HA モニタのサーバに関する定義（server 定義文）で，OpenTP1 の動作環境を指定します。server 定義文の name オペランドには，OpenTP1 ホームディレクトリの完全パス名を指定します。そのほかの環境設定については，マニュアル「高信頼化システム監視機能 HA モニタ」を参照してください。

## (3) OpenTP1 に定義する内容

現用系と予備系では，次の内容をすべて一致させてください。

- OpenTP1 のシステム定義
- ユーザサーバ（UAP）の実行形式ファイル
- トランザクションサービス制御用実行形式ファイル（trnlnkrm コマンドで再作成した場合）※
- OpenTP1 システムを構成する製品のバージョン
- OpenTP1 管理者の環境（ユーザ ID，グループ ID，環境変数）
- OpenTP1 ホームディレクトリの完全パス名
- OpenTP1 ファイル（キャラクタ型スペシャルファイル）の設定

注※

trnlnkrm コマンドを実行して OpenTP1 に登録するリソースマネジャの登録順序も一致させる必要があります。

現用系と予備系の定義で一致していない項目があっても，チェックはしていません。一致させていない場合の動作は保証しません。

### (a) システム定義の指定

系切り替え機能を使用する場合は，次に示すシステム定義を指定してください。

- システム構成定義には，系切り替え機能を使用する指定（ha_conf = Y）をしてください。指定しないと系切り替えはしません。
- 系切り替え時に，実行系の代替として待機系を起動する場合と，実行系の後処理だけ行わせるために待機系を起動する場合とで，システム環境定義のシステム開始方法（mode_conf の指定）は次のとおり指定内容が異なります。

  実行系の代替として待機系を起動する場合

  AUTO，または MANUAL1 を指定してください。AUTO と MANUAL1 では，OS 起動後の開始方法が異なるので，運用形態を基に選択してください。

MANUAL2 を指定すると，実行系／待機系のどちらかが異常終了したあとに，自動開始しません。再び系切り替えするためには，異常終了した OpenTP1 を運用コマンドで開始させておく必要があります。

実行系の後処理だけ行わせるために待機系を起動する場合

未決着トランザクションの決着や DB の整合性の確保などの，実行系の後処理だけを行わせるために待機系を起動する場合は，MANUAL2 を指定してください。この場合，待機系に縮退運転ができれば問題ないと判断できることが前提となります。系切り替え後に，待機系に実行系の後処理だけを行わせる方法については，「6.1.4(1)　OpenTP1 の開始と終了」を参照してください。

- 系が切り替わったあとの待機系の再開始（リラン）時間を短縮するため，OpenTP1 が起動する前に，ユーザサーバを起動することを指定できます。この場合，システム環境定義の user_server_ha オペランドに Y を指定してください。

## (4)　系切り替え機能を使用するときの注意

### (a)　使用する LAN

系切り替え機能を使用する場合，OpenTP1 を運用している LAN，および回線は，実行系ではオンライン状態にして，待機系ではオフライン状態にしておく必要があります。また，実行系で障害を検知して系切り替えをした場合には，障害が起こった系の LAN，および回線は自動的にオフライン状態となります。したがって，系切り替えに使用する LAN は，障害が起こって待機系になった系の保守や運用には使用できません。

障害が起こって待機系になった系の保守や運用には，コンソールを使用するか，系切り替えに使用しない LAN（保守用 LAN）を別に準備してください。

### (b)　使用するファイルシステム

系切り替え機能を使う場合は，OpenTP1 ファイルシステムを使ってください。通常の UNIX ファイルシステムを使った場合，OpenTP1 を起動できない場合があります。

### (c)　ミラーディスク機能を使用する場合

共有ディスク装置にミラーディスク機能を使用できるかどうかは，OS およびハードウェアによって決まります。OS およびハードウェアのミラーディスク機能，クラスタ機能などの仕様に基づいて使用してください。

ミラーディスク機能で二つのディスク装置に書き込んでいる最中に系切り替えが発生した場合，二つのディスク装置の間に書き込みの遅れなどによる状態の不一致が発生することがあります。系切り替え先のディスク装置に正しく書き込まれていないと，系切り替えをしても OpenTP1 は正しく動作できません。

次の OpenTP1 ファイルは，OpenTP1 を動作させるために重要なファイルです。

- ステータスファイル
- システムジャーナルファイル

- チェックポイントダンプファイル

これらの OpenTP1 ファイルをミラーディスク機能で使用する場合は，OS およびハードウェアで，共有ディスク装置にミラーディスク機能を利用できること，また，系切り替え機能を使用した場合でもディスク装置の内容が正しいことを確認した上で使用してください。

ミラーディスク装置を使えない OpenTP1 ファイルは，次に示す方法でファイルの信頼性を上げてください。

- ステータスファイル
  二重化してください。
- システムジャーナルファイル
  二重化してください。
- チェックポイントダンプファイル
  二重化するか，複数世代保証機能を使用してください。

#### (d) OpenTP1 のトレースファイルやログファイルの出力先

OpenTP1 のトレースファイルやログファイルの出力先を，共有ディスク装置上に指定しないでください。系切り替え発生時にアクセス不可となり，情報が欠けたり，エラーメッセージが出力されたりすることがあります。

### 6.1.4 系切り替え機能を使う場合の運用方法

系切り替え機能を使う場合の運用方法について説明します。

## (1) OpenTP1 の開始と終了

### (a) OpenTP1 の開始

系切り替え機能を使用する OpenTP1 は，両方の系をそれぞれ OpenTP1 の開始コマンド（dcstart コマンド）で開始します。自動起動モードの場合は，コマンドを実行しなくても開始します。

- **自動起動か手動起動かの決定**
  定義内容と，前回の終了形態の組み合わせで決まります。
- **実行系か待機系かの決定**
  HA モニタの環境設定（server 定義文）で，現用系を指定（initial オペランドに online を指定，または initial オペランドを省略）した方が，オンライン開始時に実行系になります。予備系を指定（initial オペランドに standby を指定）した方が，オンライン開始時に待機系になります。
- **正常開始か再開始かの決定**
  系切り替え機能を使用しない場合と同様に決まります。

系切り替え後に，待機系に実行系の後処理だけを行わせる方法

系切り替え後に，待機系に実行系の後処理だけを行わせるには，dcstart -U で待機系を開始します。

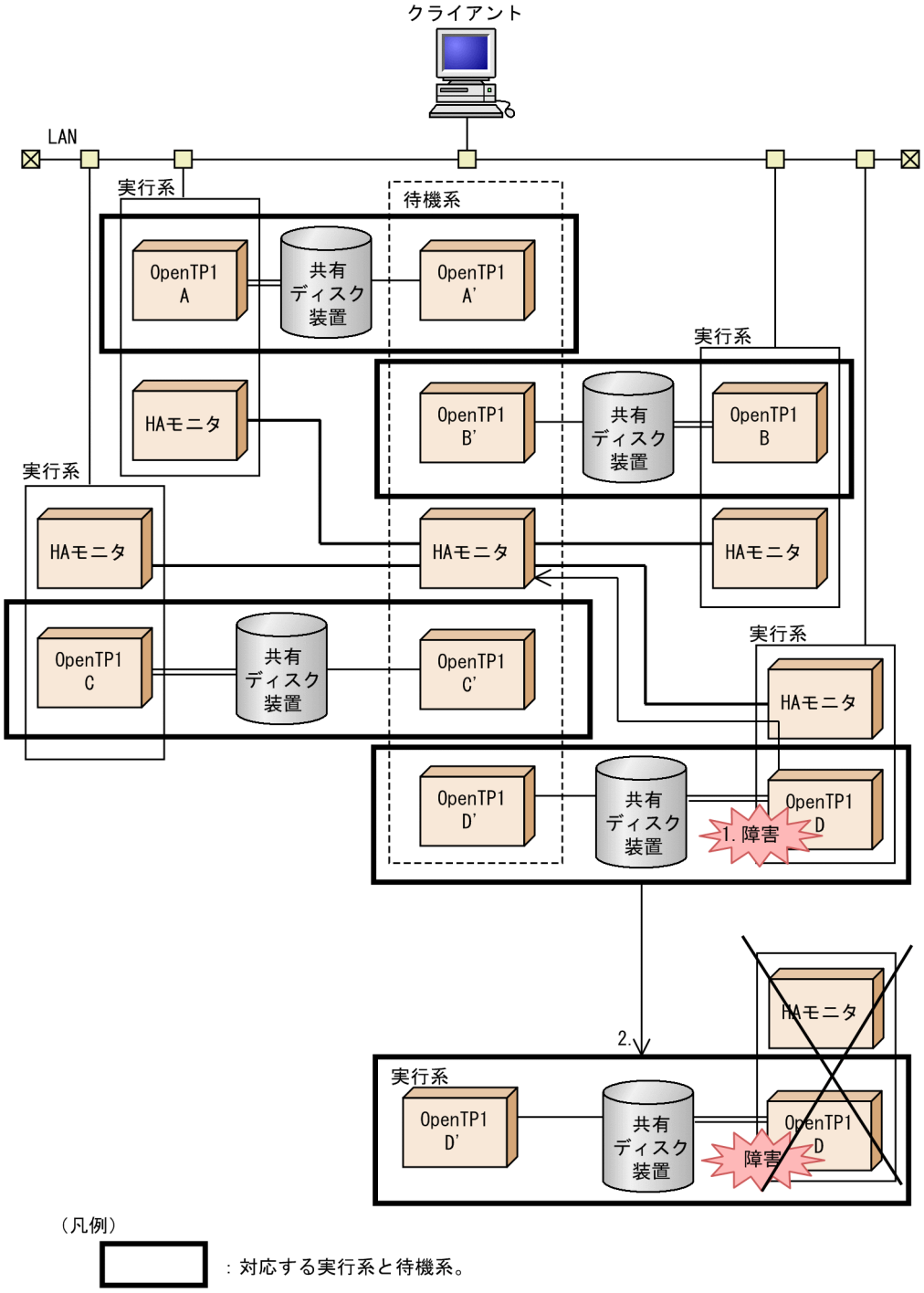
dcstart -U で待機系を開始すると，系切り替えが発生したときにユーザサーバが起動されません。これによって，実行系で障害が発生したときに，実行系の未決着トランザクションの決着や，DB の整合性の確保などの後処理を行うだけのものとして，待機系を使用できます。つまり，待機系のマシンに実行系と同等のリソースを必要としなくて済みます。

待機系の開始には dcstart コマンドを実行する必要があります。したがって，待機系を後処理用として使用する場合，手動起動の状態である必要があります。

次に，一つのノードを複数の OpenTP1 の待機系にした場合で，系切り替え後に，待機系に実行系の後処理だけを行わせるときの運用について説明します。

なお，一つのノードを複数の OpenTP1 の待機系にする場合は，それぞれの実行系に対応する待機系の OpenTP1 をそれぞれセットアップする必要があります。

図 6-3　系切り替え後に待機系に実行系の後処理だけを行わせるときの運用




説明

1. 実行系の OpenTP1 D に，障害が起こる。
2. 待機系の OpenTP1 D'が起動する。

   OpenTP1 D'は未決着トランザクションの決着や DB の整合性の確保を行います。オンライン処理は，OpenTP1 A，OpenTP1 B および OpenTP1 C が行うため，25%ダウンの縮退運転となります。

注

次の場合は，待機系は実行系の後処理だけ行うためには開始されません。

- システム環境定義のシステム開始方法（mode_conf の指定）に AUTO を指定した場合で，OpenTP1 が異常終了したとき，または前回の終了モードが異常終了以外で OS が起動したとき。
- システム環境定義のシステム開始方法（mode_conf の指定）に MANUAL1 を指定した場合で OpenTP1 が異常終了したとき。

系切り替え後に，待機系に実行系の後処理だけを行わせる運用をしていたとしても，上記の指定内容と前回終了状態の条件を満たしていた場合，dcstart コマンドが自動的に実行され，OpenTP1 が自動起動されます。これによって，待機系は実行系の代替用として待機状態になります。

待機系を再び後処理用の OpenTP1 として待機させたいときには，いったん OpenTP1 を停止したあと，再度 dcstart -U で開始してください。

## (b) OpenTP1 の終了

系切り替え機能を使用している OpenTP1 システムの終了方法を以降の表に示します。

### 表 6-1 実行系の OpenTP1 に対する終了コマンド

| 実行する終了コマンド | | コマンドの実行結果 |
|---|---|---|
| dcstop コマンドを実行 | オプション指定なし | 正常終了します。待機系の OpenTP1 も終了します。 |
| | -n オプションを指定 | 強制正常終了します。待機系の OpenTP1 も終了します。 |
| | -a オプションを指定 | 計画停止 A で終了します。待機系の OpenTP1 も終了します。 |
| | -b オプションを指定 | 計画停止 B で終了します。待機系の OpenTP1 も終了します。 |
| | -f オプションを指定 | 強制停止します。待機系の OpenTP1 も終了します。 |
| HA モニタの計画系切り替えコマンド（monswap コマンド）を実行 | | 強制停止で終了します。実行系の OpenTP1 が終了後，待機系の OpenTP1 に系切り替えをします。 |
| HA モニタの待機サーバの停止コマンド（monsbystp コマンド）を実行 | | 実行できません。 |

### 表 6-2 待機系の OpenTP1 に対する終了コマンド

| 実行する終了コマンド | | コマンドの実行結果 |
|---|---|---|
| dcstop コマンドを実行 | オプション指定なし | 実行できません。 |
| | -n オプションを指定 | |
| | -a オプションを指定 | |
| | -b オプションを指定 | |
| | -f オプションを指定 | 強制停止します。 |
| HA モニタの計画系切り替えコマンド（monswap コマンド）を実行 | | 実行できません。 |

| 実行する終了コマンド | コマンドの実行結果 |
|---|---|
| HA モニタの待機サーバの停止コマンド（monsbystp コマンド）を実行 | 待機系の OpenTP1 は終了します。OpenTP1 は再起動しません。 |

## (2) コマンド実行の制限

### (a) オフライン環境で実行するコマンドの注意

オフライン環境で実行するコマンドは，系切り替え形態を構成する両方の OpenTP1 システムを停止してから実行します。

dcstart コマンドは，もう一方の系が停止しているかどうかに関係なく，その系の OpenTP1 システムが停止していれば実行できます。ただし，待機系の OpenTP1 では，dcstart -n コマンドを実行しても-n オプションは無視されます。

### (b) オンライン環境で実行するコマンドの注意

オンライン環境で実行するコマンドは，実行系の OpenTP1 でだけ実行できます。待機系の OpenTP1 には，オンライン環境で実行するコマンドは実行できません。ただし，OpenTP1 の強制停止コマンド（dcstop -f）に限り，待機系の OpenTP1 でも実行できます。また，実行系のアンロードの負担を軽くするため，待機系のシステムジャーナルファイルをアンロードすることもできます。ただし，アンロードするコマンド（jnlunlfg コマンド）の実行には各種の留意点があります。待機系のアンロードについては，マニュアル「OpenTP1 運用と操作」の jnlunlfg コマンドの説明を参照してください。

### (c) 共有ディスク装置へのアクセス

待機系の OpenTP1 に，共有ディスク装置上の共有ファイルを操作するコマンドを実行しても，ファイル書き込みはできません。そのため，共有ファイルは破壊されません。ただし，予備系を開始させないで現用系だけで運用している場合は，予備系の OpenTP1 からファイル書き込みができるので，コマンドを実行するとファイルが破壊されることがあります。コマンドで OpenTP1 ファイル（キャラクタ型スペシャルファイル）を操作するときは，共有する両方の系の OpenTP1 を停止させてからにしてください。

## 6.1.5 強制停止処理中の系切り替え抑止機能

サーバモード※による系切り替え構成のシステムで，dcstop -f（-fd）コマンドで強制停止した場合，終了処理が途中で異常終了することがあります。

この場合，タイミングによって HA モニタは OpenTP1 の異常を検知し，系切り替えを実行してしまうことがあります。

系切り替え抑止機能を使用すると，OpenTP1 システムの強制停止処理中に異常終了しても，系切り替えを抑止できます。

注※ サーバモードの詳細については，HA モニタおよび Hitachi HA Toolkit Extension のマニュアルを参照してください。

## (1) 使用方法

この機能を使用するにはシステム共通定義の dc_deter_restart_on_stop_fail オペランドに"Y"を指定してください。

このオペランドは OpenTP1 システムの再開始時に変更できます。

## (2) 注意事項

強制停止（dcstop -f（-fd）コマンド）実行後，実行系 OpenTP1 システムは HA モニタに対して，待機系 OpenTP1 への停止指示をしますが，停止指示前の処理で異常終了した場合は系切り替えが発生します。

## 6.2　マルチノード機能（TP1/Multi）

**マルチノード機能**とは，OpenTP1 をクラスタ/並列システムに適用した場合に，OpenTP1 の運用を支援して，効率良くシステムを管理するための機能です。

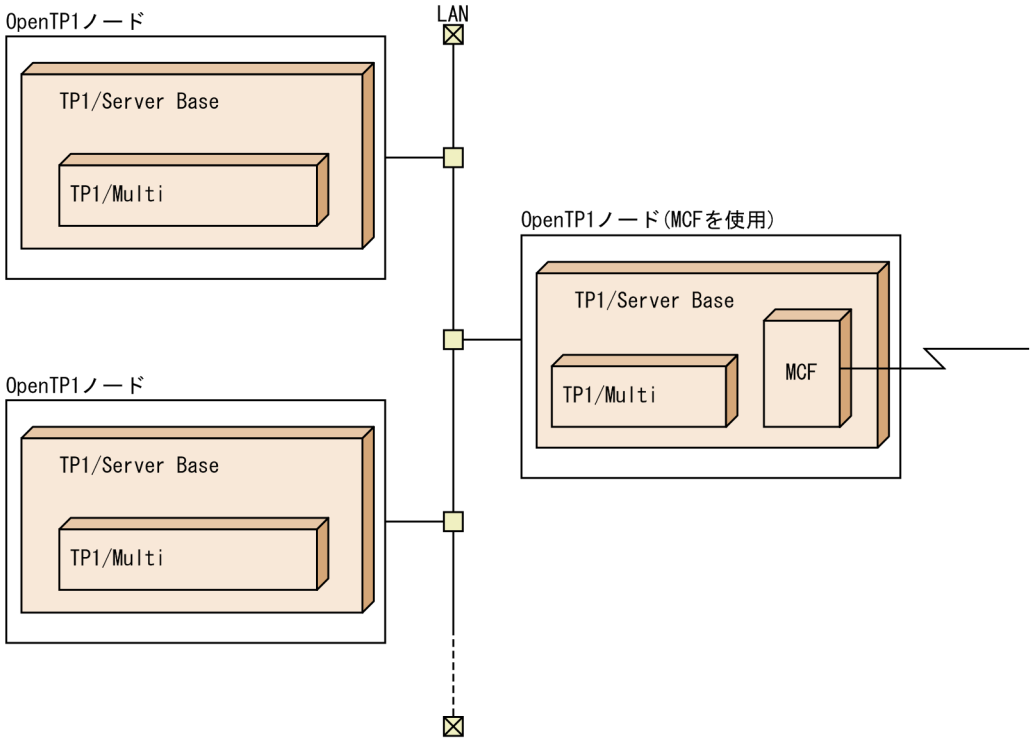
### 6.2.1　マルチノード機能の概要

**クラスタ/並列システム**とは，LAN で接続した複数のサーバマシンを並行に処理させて，大規模システムを構築する形態です。このクラスタ/並列システム形態で，一つのノードからすべての OpenTP1 システムを運用できるようにする機能を**マルチノード機能**といいます。マルチノード機能を使うと，ノードごとに OpenTP1 システムを管理する必要がなくなります。

マルチノード機能を使う場合には，すべての OpenTP1 システムに TP1/Multi が必要です。

マルチノード機能を使った OpenTP1 のソフトウェア構成を次の図に示します。

図 6-4　マルチノード機能を使った OpenTP1 のソフトウェア構成




#### (1)　マルチノード機能を使用する前提条件

マルチノード機能を使用する場合は，次のことが前提となります。

- マルチノード機能で管理する OpenTP1 のノードすべてに，TP1/Multi が組み込まれていること

- マルチノード機能で管理する OpenTP1 のノードすべてに，OpenTP1 のノードの**システム共通定義**に multi_node_option=Y と指定してあり，かつ**マルチノード構成定義**と**マルチノード物理定義**が作成してあること

## (2) マルチノード機能と OpenTP1 の機能の関係

クラスタ/並列システム形態で OpenTP1 を使用した場合の機能を，次に示します。

### (a) 単一の OpenTP1 の場合

通常の OpenTP1 の機能で運用できます。UAP も，各 OpenTP1 のノードで設定した環境で起動します。複数の UAP プロセスを並行して起動すること（**マルチサーバ機能**）も，複数のノード間でユーザサーバのサービスグループ単位で処理を振り分けること（**ノード間負荷バランス機能**）もできます。

### (b) 系切り替え機能を使用した OpenTP1 の場合

系切り替え構成の OpenTP1 の場合，系切り替えの組み合わせの単位を一つのノード（現用系と待機系が同じノード）と見なして処理します。系切り替えが起こっても，ノードが切り替わったことを意識しません。

系切り替え機能を使う場合は，**保守用 LAN**（系切り替えとは関係ない，OpenTP1 の通信に使わない LAN）が必要です。また，マルチノード物理定義のホスト名には，**保守用 LAN のホスト名**を指定してください。

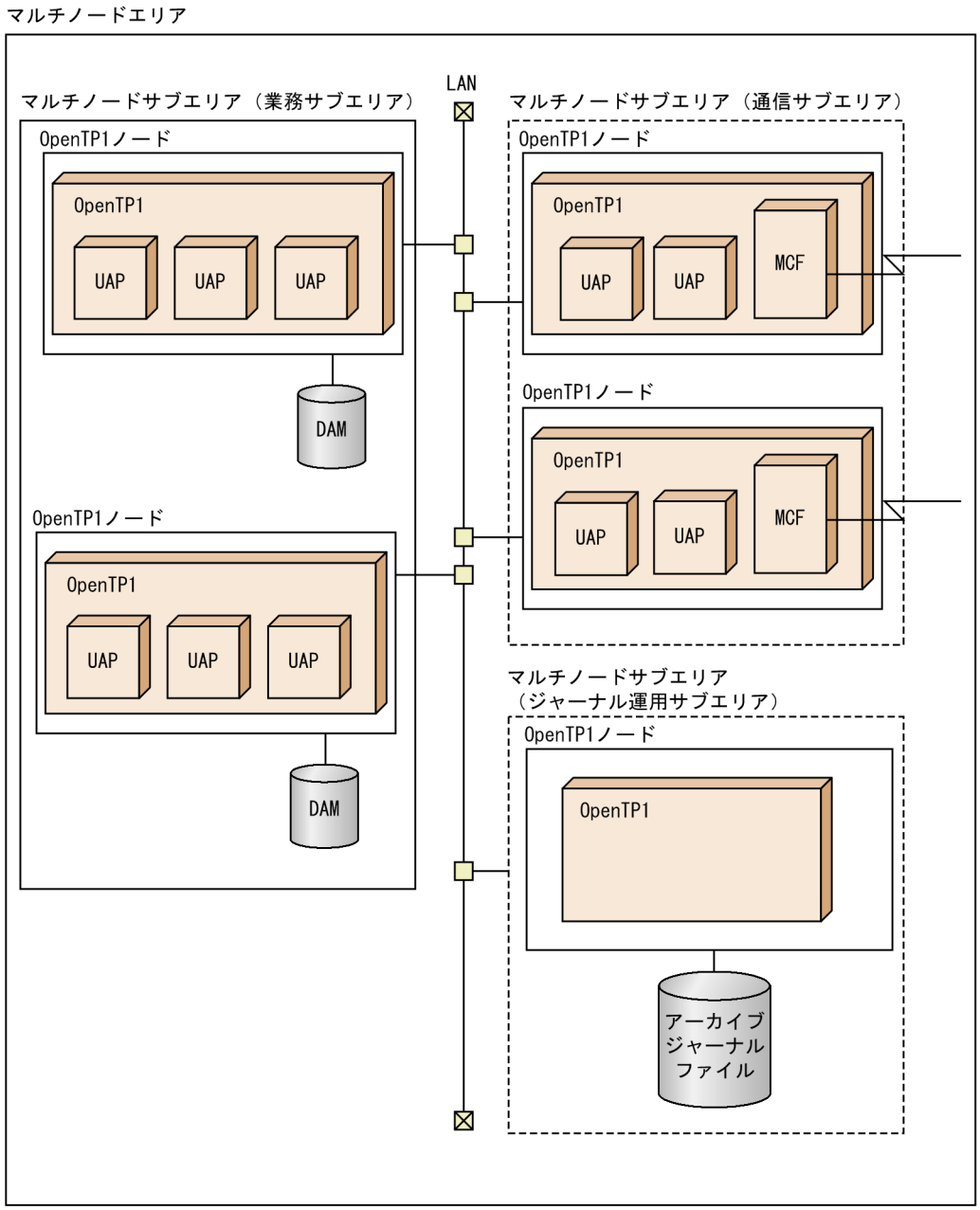
メッセージキューイング機能（TP1/Message Queue）を使った OpenTP1 システムでも，マルチノード機能を使えます。

## (3) クラスタ/並列システムを構成する要素

クラスタ/並列システムの OpenTP1 では，各ノードを管理するため，複数システムで構成される領域を，次のように管理しています。

クラスタ/並列システムに OpenTP1 を使用した形態を次の図に示します。

図 6-5　クラスタ/並列システムに OpenTP1 を使用した形態




## (a)　マルチノードエリア

クラスタ/並列システムを構成している OpenTP1 のまとまりを，**マルチノードエリア**といいます。クラスタ/並列システム内で一元的に管理したい範囲の OpenTP1 ノードを，すべてマルチノードエリアとします。

一つのクラスタ/並列システム内には，マルチノードエリアは一つだけです。

### (b) マルチノードサブエリア

マルチノードエリアを論理的に分割したまとまりを**マルチノードサブエリア**といいます。マルチノードエリアに属する OpenTP1 のノードを，共通する業務ごとにサブエリアに分割します。マルチノードサブエリアとして分割する例を次に示します。

- **業務サブエリア**
  **ユーザファイル**（DAM ファイル，TAM ファイルなど）を管理する OpenTP1 ノードのまとまりです。
- **広域網との通信サブエリア**
  TP1/Message Control を使用して広域網と通信する OpenTP1 ノードのまとまりです。
- **ジャーナル運用サブエリア**
  **アーカイブジャーナルファイル**を取得する OpenTP1 ノードのまとまりです。グローバルアーカイブジャーナル機能を使う場合に必要です。

グローバルアーカイブジャーナル機能については，「6.2.3　グローバルアーカイブジャーナル機能」を参照してください。

### (c) OpenTP1 ノード

マルチノードエリア，またはマルチノードサブエリアを構成する個々の OpenTP1 システムを **OpenTP1 ノード**といいます。それぞれの OpenTP1 ノードは，OpenTP1 のシステム共通定義で指定した**ノード識別子**で区別されます。

### (d) システム定義の指定

**マルチノード構成定義**で，それぞれのエリアに属する OpenTP1 ノードを指定します。OpenTP1 ノードのノード識別子として指定する値は，システム共通定義に指定したノード識別子です。

## 6.2.2 マルチノード機能でできる操作

クラスタ/並列システム形態にマルチノード機能を使うと，次のような操作ができます。コマンドについてはマニュアル「OpenTP1 運用と操作」を，関数についてはマニュアル「OpenTP1 プログラム作成の手引」を参照してください。

## (1) クラスタ/並列システム内にある OpenTP1 の開始と終了を管理

一つの OpenTP1 ノードから，コマンドを入力して，複数の OpenTP1 システムをマルチノードサブエリア単位に開始させたり終了させたりできます。マルチノードサブエリアを開始させる場合は **dcmstart コマンド**を，終了させる場合は **dcmstop コマンド**を実行します。

### (2) OpenTP1 ノードの状態を取得

一つの OpenTP1 ノードから，マルチノードエリアにある OpenTP1 ノードの状態を知ることができます。OpenTP1 ノードの状態を知りたい場合は，**dcndls コマンド**を実行します。

OpenTP1 ノードの状態は UAP から関数（**dc_adm_get_nd_status_××××関数**）を使って知ることもできます。

### (3) OpenTP1 ノードにあるユーザサーバの状態を取得

一つの OpenTP1 ノードから，OpenTP1 ノードにあるユーザサーバの状態を知ることができます。OpenTP1 ノードの状態を知りたい場合は，UAP から関数（**dc_adm_get_sv_status_××××関数**）を呼び出します。

### (4) マルチノードエリアの構成情報の取得

一つの OpenTP1 ノードから，マルチノードエリアを構成するすべての OpenTP1 ノードのノード識別子，または指定したマルチノードサブエリアに属する OpenTP1 ノードのノード識別子を知ることができます。OpenTP1 ノードのノード識別子を知りたい場合は，UAP から関数（**dc_adm_get_nodeconf_××××関数**，**dc_adm_get_node_id 関数**）を呼び出します。

## 6.2.3 グローバルアーカイブジャーナル機能

クラスタ/並列システム形態で OpenTP1 を使う場合でも，システムジャーナルファイルを OpenTP1 ノードごとにアンロードする必要があります。この負担を軽減するために，マルチノード機能のオプションとして，専用の OpenTP1 ノードに各 OpenTP1 ノードのシステムジャーナルをアーカイブできます。この機能を**グローバルアーカイブジャーナル機能**といいます。グローバルアーカイブジャーナル機能でアーカイブしたジャーナルをアンロードすると，ノードごとにジャーナルをアンロードする必要がなくなるため，OpenTP1 ノードの運用の負担を軽減できます。

グローバルアーカイブジャーナルでは，アーカイブするジャーナルの種類を選択できます。選択するジャーナルの種類は，システムジャーナル定義の jnl_arc_rec_kind オペランドに選択するジャーナルレコード種別を指定します。ジャーナルの種類については，「4.2.2 システムジャーナルファイル」を参照してください。

グローバルアーカイブジャーナル機能でアーカイブしたジャーナルファイルを**アーカイブジャーナルファイル**といいます。アーカイブジャーナルファイルについては，「4.2.6 アーカイブジャーナルファイル」を参照してください。

グローバルアーカイブジャーナル機能を使用した場合でも，各 OpenTP1 ノードの回復時はシステムジャーナルファイルを使用します。そのため，回復の性能に影響はありません。

ファイルの運用方法や障害時の対策については，マニュアル「OpenTP1 運用と操作」を参照してください。

## (1) グローバルアーカイブジャーナル機能を使用するノードの種類

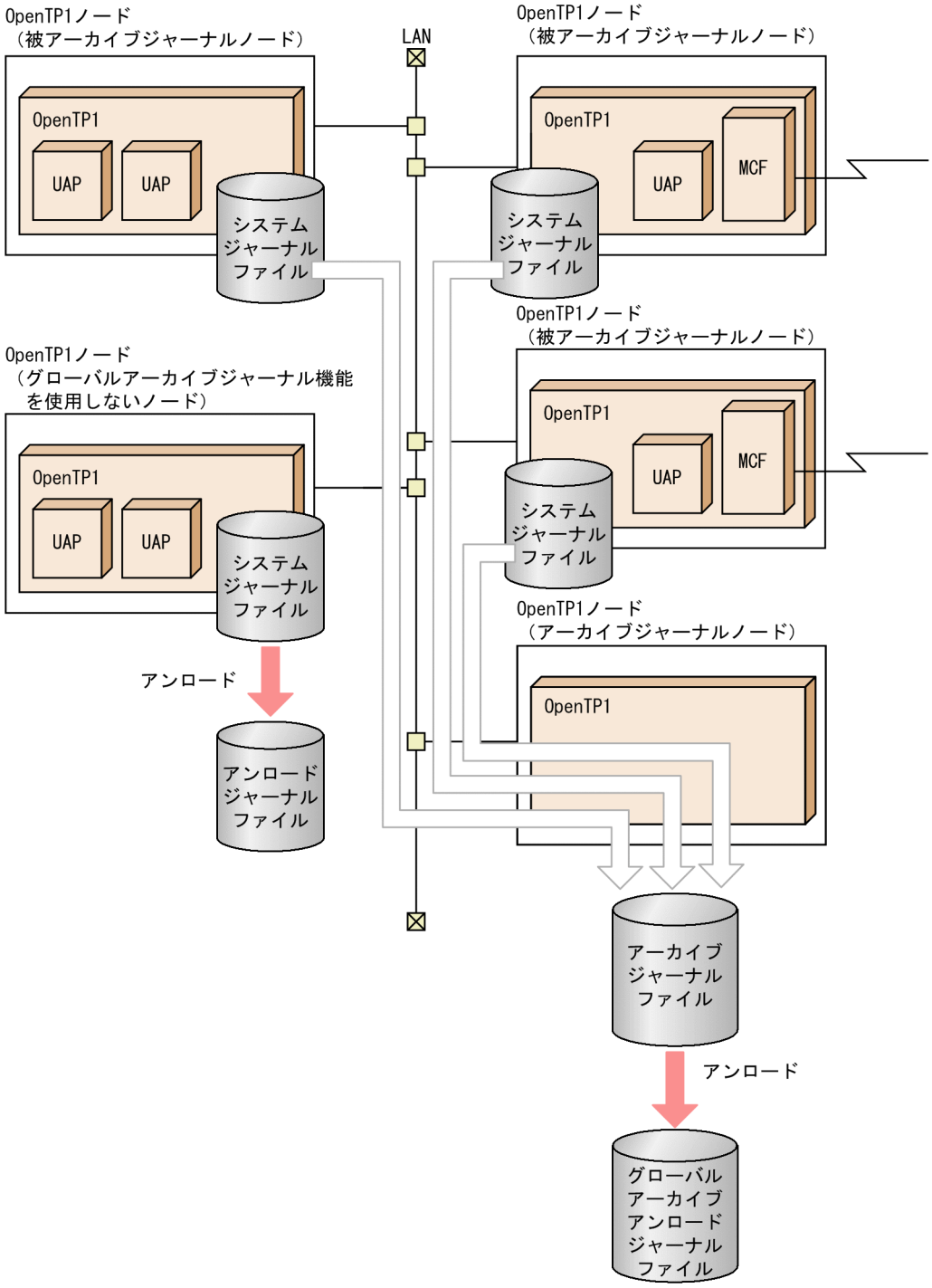
マルチノードエリアの任意の OpenTP1 ノードを，アーカイブジャーナルファイル専用のノードとします。このノードを**アーカイブジャーナルノード**といいます。このノードのアーカイブジャーナルファイルへ，各 OpenTP1 ノードからシステムジャーナルファイルをアーカイブします。アーカイブジャーナルノードへシステムジャーナルファイルをアーカイブする OpenTP1 ノードを**被アーカイブジャーナルノード**といいます。

アーカイブジャーナルノードと，ほかの OpenTP1 ノード（被アーカイブジャーナルノード，またはグローバルアーカイブジャーナル機能を使用しないノード）と異なる点を次に示します。

- ユーザサーバを起動できません。
- マルチノードエリアの OpenTP1 ノードをノードごとに開始する場合，最初に起動します。

グローバルアーカイブジャーナルサービスの概要を次の図に示します。

図 6-6　グローバルアーカイブジャーナルサービスの概要




## (2)　リソースグループ

アーカイブジャーナルノードには，1 から 16 種類までのアーカイブジャーナルファイルを作成できます。それぞれのファイルは使用目的で分けられるので，マルチノードのあるエリアを A アーカイブジャーナルとして取得，あるエリアを B アーカイブジャーナルとして取得，などの運用ができます。

使う目的別に分けたアーカイブジャーナルファイルのまとまりを**リソースグループ**といいます。グローバルアーカイブジャーナルサービスは，リソースグループ名を使ってアーカイブジャーナルファイルを管理します。

一つのリソースグループに属せる被アーカイブジャーナルノードは，20 ノードまでです。

リソースグループは，アーカイブジャーナルサービス定義のファイル名を示します。アーカイブジャーナルノードでリソースグループを幾つ使用するかは，グローバルアーカイブジャーナルサービス定義で指定します。

## (3) グローバルアーカイブジャーナル機能を使用する OpenTP1 ノードに指定する内容

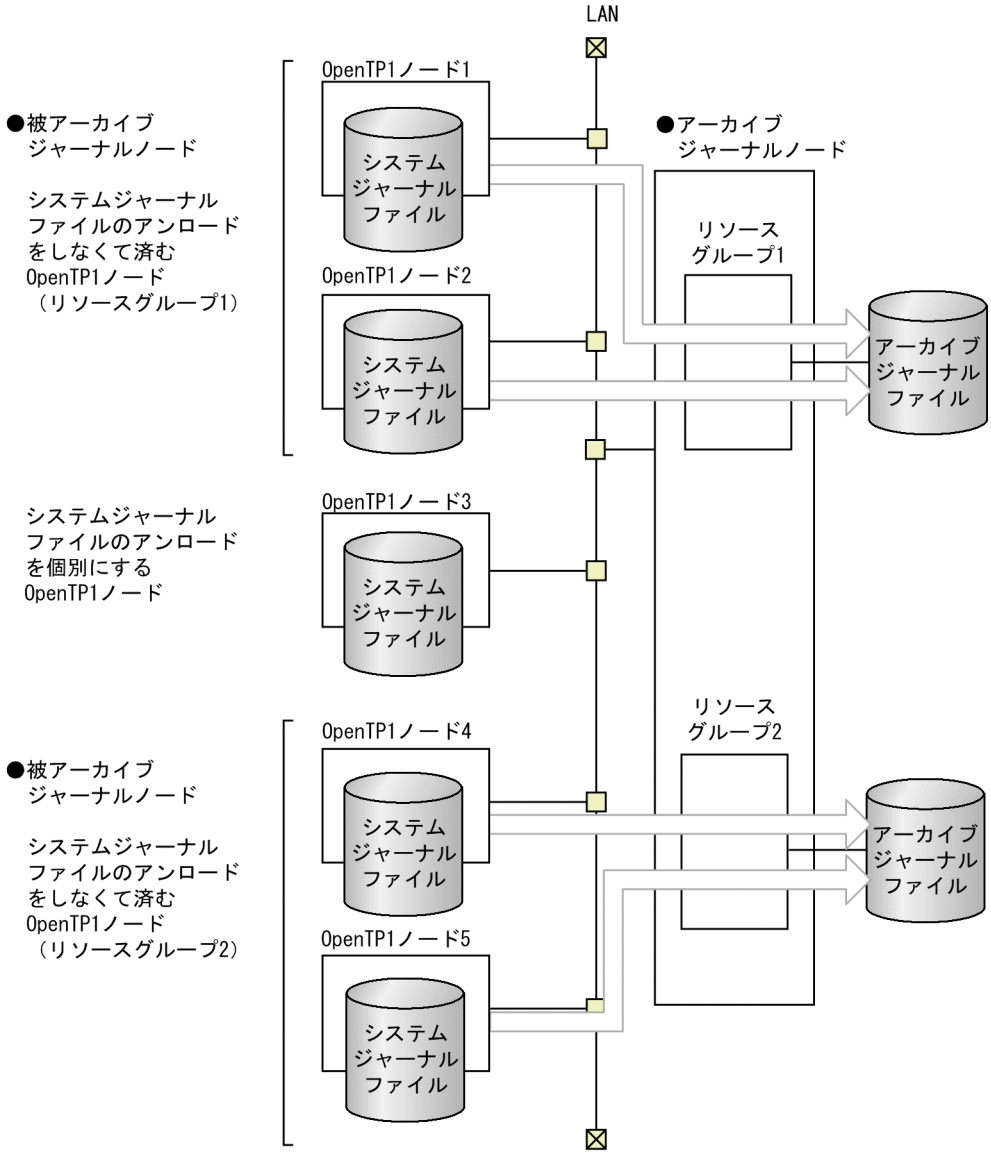
グローバルアーカイブジャーナル機能を使用するノードには，次に示すシステム定義を指定しておきます。このことで，グローバルアーカイブジャーナルサービスと各ノードのジャーナルサービスを関連づけられます。

- アーカイブジャーナルノードに指定する定義
  **グローバルアーカイブジャーナルサービス定義**と**アーカイブジャーナルサービス定義**を作成しておきます。
- 被アーカイブジャーナルノードに指定する定義
  **システムジャーナルサービス定義**の jnl_arc_name オペランドに，アーカイブジャーナルノードのノード識別子と，アーカイブジャーナルファイルのリソースグループ名を指定しておきます。

システムジャーナルファイルをアーカイブしない OpenTP1 ノードでは，個別にジャーナルをアンロードしてください。

グローバルアーカイブジャーナルサービスとリソースグループの関係を次の図に示します。

図 6-7　グローバルアーカイブジャーナルサービスとリソースグループの関係




## (4)　アーカイブジャーナルファイルのアンロード

アンロード待ちのファイルグループは，OpenTP1 のコマンドで，通常ファイルに複写（**アンロード**）する必要があります。アーカイブジャーナルファイルをアンロードしたファイルを，**グローバルアーカイブアンロードジャーナルファイル**といいます。

すべてのファイルグループがアンロードされていない状態でスワップ先がなくなった場合，グローバルアーカイブジャーナルサービスは異常終了します。

### (a)　グローバルアーカイブアンロードジャーナルファイルのジャーナル維持機能

アンロードした通常ファイルは，システムジャーナルファイルの場合と同様に，DAM ファイルの回復や稼働統計情報ファイルの編集ができます。また，ユーザジャーナルをオフラインのプログラムへ引き継げます。

グローバルアーカイブアンロードジャーナルファイルには，複数の OpenTP1 ノードのシステムジャーナル情報が入っています。このジャーナルをマージした稼働統計情報の編集，および複数の OpenTP1 ノードのジャーナルを時系列にソートするなど，各種のジャーナル編集ができます。

## (5) システムジャーナルファイルの状態

被アーカイブジャーナルノードの OpenTP1 ノードでは，次に示すシステムジャーナルファイルの状態が追加となります。

- **アーカイブ済み/未アーカイブ**
  アーカイブジャーナルファイルに転送，または出力されたファイルグループかどうかの状態

該当する被アーカイブジャーナルノードでアンロードが済んでいなくても，アーカイブ済みで上書きできる状態のファイルグループであれば，次回のスワップ先にできるファイルグループとなります。

## 6.3　マルチ OpenTP1

一つのノードに二つの OpenTP1 システムを置く形態を**マルチ OpenTP1** といいます。一つのノードで実際の業務用とテスト用の二つの OpenTP1 を稼働したりするときに使います。

個々の OpenTP1 は，独立して運用できます。マルチ OpenTP1 の場合，ノードアドレスを複数の OpenTP1 で共有することも，それぞれ異なるアドレスを使うこともできます。

### 6.3.1　マルチ OpenTP1 の形態

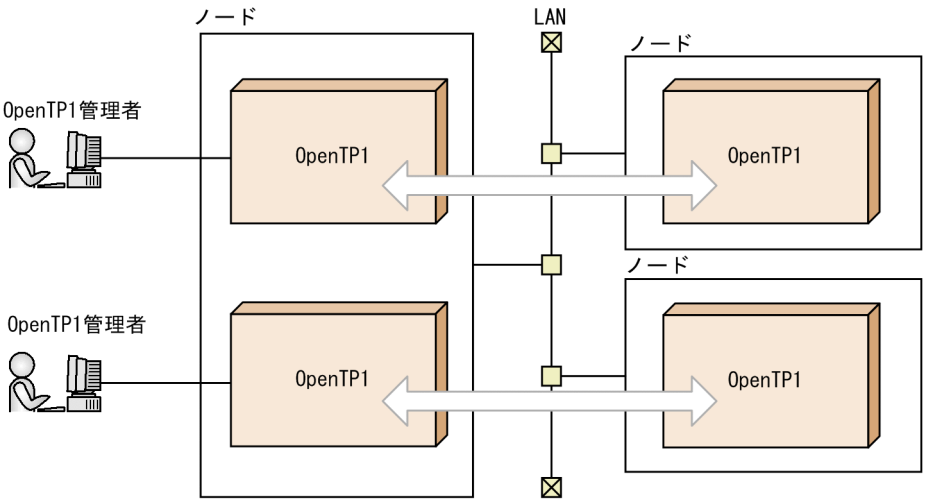
マルチ OpenTP1 では，複数の TP1/Server Base をそれぞれ異なる OpenTP1 ホームディレクトリにインストールします。

OpenTP1 管理者は，それぞれの OpenTP1 に必要です。この場合，OpenTP1 管理者を同じ利用者としても，異なる利用者としてもかまいません。

マルチ OpenTP1 の形態の場合，それぞれの OpenTP1 は **OpenTP1 識別子**で区別されます。さらに，ネームサービスとプロセスサービスのポート番号は，OpenTP1 ごとに異なる番号を指定してください。OpenTP1 識別子とポート番号は，システム共通定義で指定します。マルチ OpenTP1 の環境設定については，マニュアル「OpenTP1 運用と操作」を参照してください。

マルチ OpenTP1 を次の図に示します。

図 6-8　マルチ OpenTP1




#### (1)　マルチ OpenTP1 のコマンドを振り分けるサンプル

マルチ OpenTP1 のノードに rsh などを使ってほかのノードからコマンドを実行する場合，どちらの OpenTP1 でコマンドが実行されるかわかりません。そのため，ノード名を指定してコマンドを実行する必要があります。ノード名を指定してコマンドを実行するコマンド（シェルファイル）として，TP1/Server Base のサンプルに **delvcmd コマンド**が組み込んであります。

delvcmd コマンドの使い方については，マニュアル「OpenTP1 プログラム作成の手引」にある OpenTP1 のサンプルの説明を参照してください。

## 6.4　マルチホームドホストの形態での注意事項

二つ以上の物理ネットワークに接続している TCP/IP を使っているホストを，**マルチホームドホスト**といいます。このようなホストで稼働する OpenTP1 と通信するときは，そのホストにつながっている任意の LAN のアドレス（IP アドレス）を基にして通信します。通常の通信では，OpenTP1 で IP アドレス（ホスト名）を意識する必要はありません。

ただし，一つのホストで OpenTP1 を複数稼働させていて，そのうち一つを系切り替え機能で使っている場合，系の切り替えが起こることで，該当するホストの OpenTP1 と通信できない場合があります。このようなことが起こらないように，どの IP アドレス（ホスト名）がどの OpenTP1 に対応しているかを定義する必要があります。

### 6.4.1　系切り替えをするマルチホームドホスト形態に必要な定義

一つのホストにある複数の OpenTP1 のうち，どちらかを系切り替え形態で使う場合は，システム共通定義の my_host オペランドにホスト名を指定した上で，dcbindht 定義コマンドに次に示す項目を指定します。

- 使うホスト名（/etc/hosts ファイル，または DNS などで IP アドレスとマッピングできるホスト名）
- 使うネットワーク名（/etc/networks ファイル，または NIS などでネットワーク番号とマッピングできるネットワーク名）

### 6.4.2　OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態の例

OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態とは，系切り替えで IP アドレスを引き継ぐ方法です。

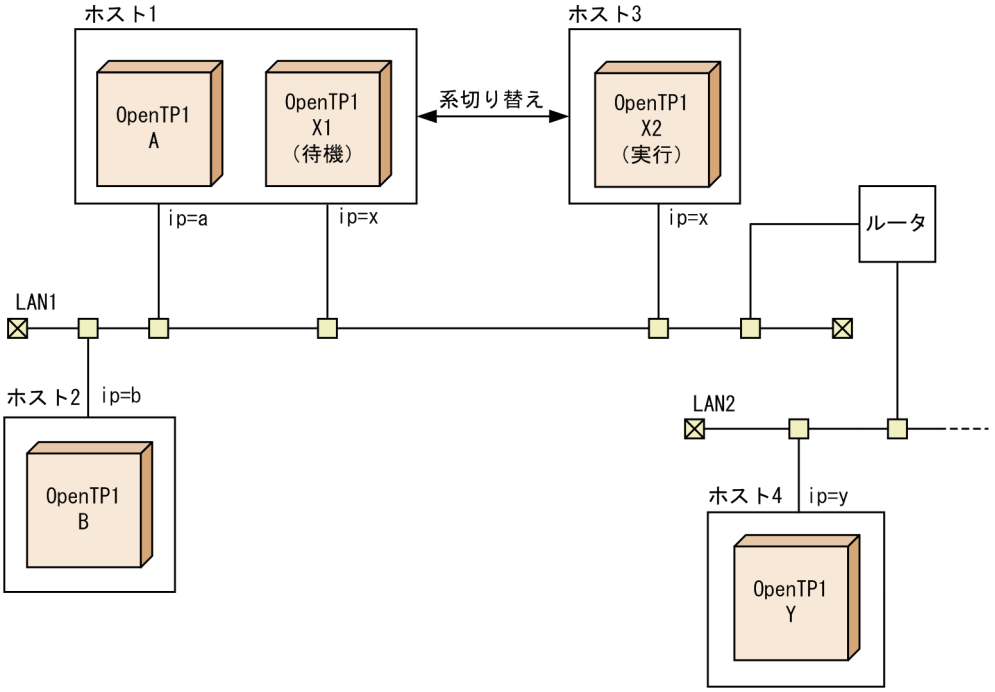
この方法は，RPC でサービスのアドレス情報を取得したあとで系切り替えが起こった場合でも，そのアドレスをそのまま使用してサービスが使えます。

一つのホストで複数の OpenTP1 が稼働することがある系切り替え構成を使うときは，dcbindht 定義コマンドで，常に同じアドレスを使うことを定義してください。

#### (1)　1：1 の系切り替え構成の場合の例

OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態例（1：1 の系切り替え構成の場合）を次の図に示します。

図 6-9　OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態例（系切り替え構成が一つの場合）




RPC でサービスを要求された OpenTP1 は，サービスのアドレス情報を取得するために使った IP アドレスを保持します。

上の図に示す構成で，OpenTP1-B から OpenTP1-A へサービスを要求するときには，IP アドレス「a」または「x」のどちらかを使います。

IP アドレス x を使ってサービスのアドレス情報を取得したあとで系切り替えが起こった場合，IP アドレス x は OpenTP1-X2 に引き継がれます。このとき，OpenTP1-B から OpenTP1-A へのサービス要求では，アドレス情報を取得したときの IP アドレス x を使って通信しようとします。IP アドレス x は，系切り替えによってホスト 3 に移っているため，OpenTP1-B は目的の OpenTP1-A と通信できなくなってしまいます。

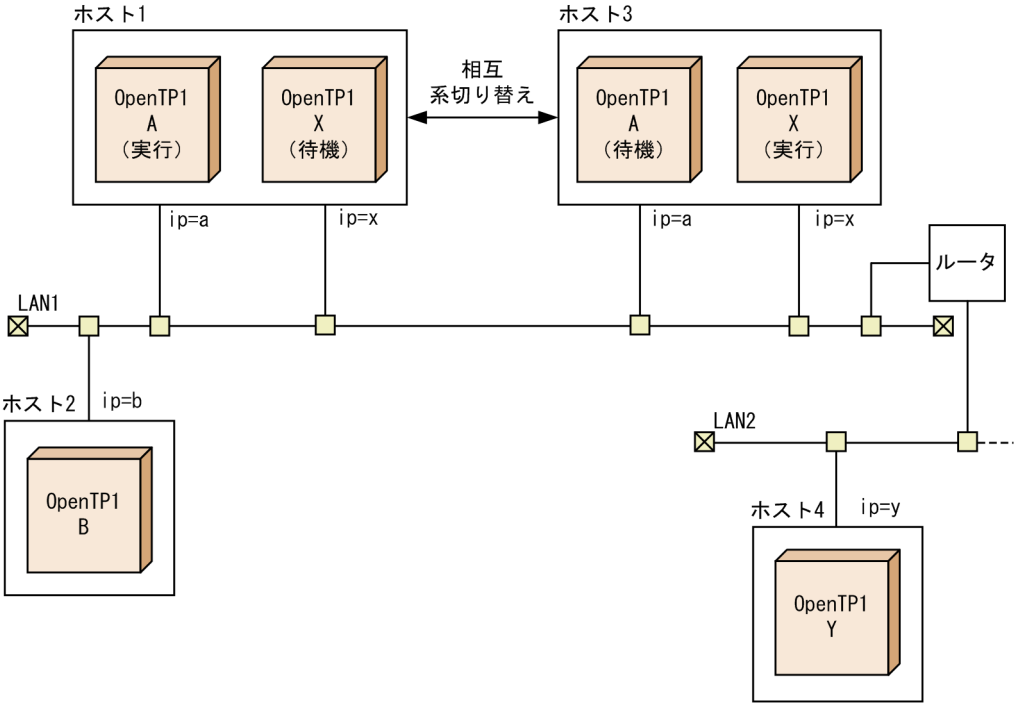
このようなことを防ぐため，システム共通定義の dcbindht 定義コマンドで，OpenTP1-A は常に IP アドレス a を使うことを定義しておきます。

同様に，OpenTP1-X と通信する OpenTP1-Y が IP アドレス a を使わないように，dcbindht 定義コマンドで，OpenTP1-X は常に IP アドレス x を使うように定義しておきます。

## (2)　相互系切り替え構成の場合の例

OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態例（相互系切り替え構成の場合）を次の図に示します。

図 6-10　OpenTP1 と IP アドレス（ホスト名）を対応づける必要がある形態例（相互系切り替え構成の場合）




上の図に示す構成では，ホスト 1 とホスト 3 にある OpenTP1 は，相互に実行系と待機系を配置しています。

この場合，OpenTP1-A には，系切り替えによって通信できたりできなかったりする ip=b の IP アドレスに影響されないように，dcbindht 定義コマンドの-h オプションに ip=a の IP アドレスを指定しておきます。同様に，OpenTP1-X にも，dcbindht 定義コマンドの-h オプションに ip=x の IP アドレスを指定しておきます。

# 7 OpenTP1 で使用するシステムの資源

OpenTP1 を使用したオンライントランザクション処理システムを設計したり，運用管理したりするためには，システムのプロセス構造，およびメモリ構造を知っておく必要があります。

この章では，OpenTP1 で使用するシステムの資源について説明します。

# 7.1 OpenTP1 のプロセス構造

システムサービスプロセスを次の表に示します。

## 表 7-1 システムサービスプロセス

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| prcd | 1※2 | プロセスサービス（スーパユーザプロセス） | プロセスサービス定義 | init または systemd | #3 | 1 |
| dcmond | 1 | OpenTP1 監視サービス | システム共通定義 | prcd | #1 | 0 |
| prctee | 1※3 | プロセスサービス（OpenTP1 の標準出力，標準エラー出力のリダイレクト機能） | なし | init または systemd | #1 | 0 |
| namd | 1 | ネームサービス | ネームサービス定義 | prcd | #3 | 1 |
| namaudtd | 0 または 1※4 | ネームサービス（監視機能） | ネームサービス定義 | prcd | #3 | 0 または 1※4 |
| scdd | 1 | スケジュールサービス | スケジュールサービス定義 | prcd | #3 | 1 |
| scdmltd | ≧0～n※5 | スケジュールサービス（マルチスケジューラ機能） | スケジュールサービス定義 | prcd | #1 | ≧0※6 |
| trnd | 1 | トランザクションサービス（トランザクション管理） | トランザクションサービス定義 | prcd | #3 | 1 |
| trnrvd | 1～128※7 | トランザクションサービス（トランザクション回復<並行回復プロセス数>） | トランザクションサービス定義 | prcd | #1 | 1 |
| trnrmd | 1 | トランザクションサービ | トランザクションサービス定義 | prcd | #3 | 1 |

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| trnrmd | 1 | ス（リソースマネジャ監視） | トランザクションサービス定義 | prcd | #3 | 1 |
| jnld | 1 | ジャーナルサービス（ジャーナル管理） | ジャーナルサービス定義 | prcd | #3 | 1 |
| jnlswd | 0 または 1※8 | ジャーナルサービス（ジャーナルファイル管理） | システムジャーナルサービス定義 | prcd | #3 | 0 または 1※8 |
| jnliod | 0～16※8，※9 | ジャーナルサービス（ジャーナルファイル入出力） | なし | prcd | #3 | 0～16※8，※9 |
| | 1～256※10 | ジャーナルサービス（アーカイブジャーナルファイル入出力）※11 | なし | prcd | #3 | 1～256※10 |
| jnlutld | 0 または 1※12 | ジャーナルユティリティサービス | システムジャーナルサービス定義 | prcd | #3 | 0 または 1※12 |
| jnlsdd | 1 | グローバルアーカイブジャーナルサービス※13 | システムサービス構成定義 | prcd | #3 | 1 |
| jard | 1 | グローバルアーカイブジャーナルサービス（グローバルジャーナル管理）※11 | グローバルアーカイブジャーナルサービス定義 | prcd | #3 | 1 |
| jarswd | 1～16 | グローバルアーカイブジャーナルサービス（アーカイブジャーナル | アーカイブジャーナルサービス定義 | prcd | #3 | 1 |

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| jarswd | 1～16 | ファイル管理）※11 | アーカイブジャーナルサービス定義 | prcd | #3 | 1 |
| jarrvd | 1～320※14 | グローバルアーカイブジャーナルサービス（ジャーナルデータアーカイブ）※11 | なし | prcd | #3 | 1～320※14 |
| itvd | 1 | インタバルサービス | インタバルサービス定義 | prcd | #3 | 1 |
| stsd | 1 | ステータスサービス | ステータスサービス定義 | prcd | #3 | 1 |
| cpdd | ≧0※8, ※15 | チェックポイントダンプサービス | ジャーナルサービス定義<br>チェックポイントダンプサービス定義 | prcd | #3 | ≧0※8, ※15 |
| tjld | 0または1※8 | トランザクションジャーナルサービス | なし | prcd | #3 | 0または1※8 |
| qued | 0または1※16 | メッセージキューサービス | メッセージキューサービス定義 | prcd | #3 | 0または1※16 |
| damd | 0または1※17 | DAMサービス | DAMサービス定義 | prcd | #3 | 0または1※17 |
| tamd | 0または1※18 | TAMサービス | TAMサービス定義 | prcd | #3 | 0または1※18 |
| tamiod | 0または1※18 | TAMサービス（TAMファイル入出力） | TAMサービス定義 | prcd | #3 | 0または1※18 |
| istd | 0または1※19 | ISTサービス | ISTサービス定義 | prcd | #3 | 0または1※19 |
| logd | 1 | ログサービス | ログサービス定義 | prcd | #3 | 1 |
| prfiop※20 | 0，8，または9※21 | 性能検証用トレース取得サービス | システム共通定義 | dcstart→initまたはsystemd※22 | #2 | 0 |

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| cltcond | 0 または 1※23 | クライアントサービス（CUP 実行プロセス） | クライアントサービス定義 | prcd | #1 | 0 または 1※23 |
| cltd | 0 または 1※24 | クライアントサービス | クライアントサービス定義 | prcd | #3 | 0 または 1※24 |
| clttrnd | 0 または 1※25 | クライアントサービス（トランザクショナル RPC 実行プロセス） | クライアントサービス定義 | prcd | #1 | 0 または 1※25 |
| xatd | 0 または 1※26 | XATMI サービス | なし | prcd | #3 | 0 または 1※26 |
| xatcd | 0 または 1※26 | XATMI 通信サービス | XATMI 通信サービス定義 | prcd | #3 | 0 または 1※26 |
| rmmd | 0 または 1※27 | リソースマネジャモニタサービス | RMM サービス定義 | prcd | #3 | 0 または 1※27 |
| admrsvre | 0 または 1 | プロセスサービス（部分回復） | プロセスサービス定義 | prcd | #3 | 0 |
| mcfmngrd | 0 または 1※28 | MCF サービス（MCF マネジャ） | MCF マネジャ定義 | prcd | #1 | 0 または 1※28 |
| mapsmgrd | 0 または 1※28，※29 | マッピングサービス | マッピングサービス定義<br>マッピングサービス属性定義 | prcd | #3 | 0 または 1※28，※29 |
| mcfcmdsv | 0 または 1※28，※30 | MCF サービス（MCF オンラインコマンド） | なし | prcd | #3 | 0 または 1※28，※30 |
| ユーザ指定※30 | 0～239※28，※32 | MCF サービス（MCF 通信） | MCF 通信構成定義<br>MCF アプリケーション定義 | prcd | #3 | 0 から 239※28，※32 |

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| mqad | 0 または 1※33 | MQA サービス（MQA サーバ） | MQA サービス定義 | prcd | #3 | 0 または 1※33 |
| mqaiod | 0 または 1 以上※33, ※34 | MQA サービス（MQA 入出力サーバ） | MQA サービス定義 | prcd | #3 | 0 または 1 以上※33, ※34 |
| mqcdtcp | 0 または 1※33, ※35 | MQA サービス（MQC リスナサーバ） | MQA サービス定義<br>MQC サービス定義 | prcd | #3 | 0 または 1※33, ※35 |
| mqacmd | 0 または 1※33 | MQA サービス（MQA コマンドサーバ） | MQA サービス定義 | prcd | #3 | 0 または 1※33 |
| mqamnd | 0 または 1※33 | MQA サービス（MQA 監視サーバ） | MQA サービス定義 | prcd | #3 | 0 または 1※33 |
| mqtdtcp | 0 または 1 以上※33, ※36 | MQA サービス（MQT サーバ） | MQT 通信構成定義<br>MQT サービス定義 | prcd | #3 | 0 または 1 以上※33, ※36 |
| mqtmngd | 0 または 1※33 | MQA サービス（MQT マネジャサーバ） | MQA サービス定義 | prcd | #3 | 0 または 1※33 |
| mqcgwp | 0 または 1 以上※33, ※37 | MQA サービス（MQC ゲートウェイサーバ） | MQC ゲートウェイサーバユーザサービス定義 | prcd | #3 | 0 または 1 以上※33, ※37 |
| mqrsup | 0 または 1※33, ※38 | MQA サービス（リポジトリ管理サーバ（mqrsup）） | リポジトリ管理サーバユーザサービス定義「$DCCONFPATH/mqrsup」 | prcd | #1 | 0 または 1※33, ※38 |
| mqrspp | 0 または 1※33, ※38 | MQA サービス（リポジトリ管理サーバ（mqrspp）） | リポジトリ管理サーバユーザサービス定義 | prcd | #1 | 0 または 1※33, ※38 |

| 実行形式ファイル名 | 稼働数 | サービス | 関連するシステム定義 | 親プロセス | プロセスダウン後の動作※1 | server_count の見積もりに加算する値 |
|---|---|---|---|---|---|---|
| mqrspp | 0 または 1※33，※38 | MQA サービス（リポジトリ管理サーバ（mqrspp）） | 「$DCCONFPATH/mqrspp」 | prcd | #1 | 0 または 1※33，※38 |
| rapclman | 0 または 1※39 | リモート API サービス（rap クライアントマネジャ） | rap クライアントマネジャサービス定義 | prcd | #3 | 0 または 1※39 |
| raplisnr | 0～1024※40 | リモート API サービス（rap リスナー） | rap リスナーサービス定義 | prcd | #1 | 0～1024※40 |
| rapserver | 0～1024※40 | リモート API サービス（rap サーバ） | rap リスナーサービス定義 | prcd | #1 | 0～1024※40 |
| rtsspp | 0 または 1※41 | リアルタイム統計情報サービス | リアルタイム取得項目定義 | prcd | #1 | 0 または 1※41 |
| rtssup | 0 または 1※42 | リアルタイム統計情報サービス | リアルタイム取得項目定義 | prcd | #1 | 0 または 1※42 |
| utod | 0 または 1※43 | テスタサービス | テスタサービス定義 | prcd | #3 | 0 または 1※43 |

注

「稼働数」は，一つのノード内で OpenTP1 が一つ稼働する場合の，一つのノード当たりのシステムサービスプロセス数です。

注※1

#1：プロセスは自動で再起動します。操作は不要です。ただし，クリティカルセクション内でダウンした場合は OpenTP1 システムがシステムダウンします。

#2：プロセスは再起動しないで OpenTP1 システムは縮退運転します。なお，手動でプロセスの再起動はできません。

#3：プロセスは再起動しないで OpenTP1 システムがシステムダウンします。OpenTP1 システムを再起動してください。

注※2

通常，稼働時のプロセス数は一つですが，このプロセスは OpenTP1 プロセスの起動処理を行っているため，主に次に示すようなタイミングで一時的に複数のプロセスが存在しているように見えることがあります。

・dcstart コマンドによる OpenTP1 開始時

・dcsvstart コマンドによるユーザサーバ起動時

・非常駐サーバのプロセス起動時

注※3

標準出力，標準エラー出力のリダイレクト用のプロセスです。このプロセスは，dcsetup コマンド，prcd の延長で作成されますが，このプロセスで取得した情報は，OpenTP1 のコマンド（prcls コマンド）では表示できません。内容を表示する場合は，OS のコマンド（ps コマンドなど）を実行してください。

注※4

次のどちらかの場合，稼働数が 1 になります。

・ネームサービス定義の name_audit_conf オペランドに 1 または 2 を指定している

・システム共通定義の name_service_mode オペランドに manager または agent を指定している

注※5

n：スケジュールサービス定義の scdmulti 定義コマンドの-m オプション指定値の総数です。

注※6

マルチスケジューラ機能使用時のマルチスケジューラグループ数です。

注※7

トランザクションサービス定義の trn_recovery_process_count オペランド指定値です。

注※8

システム共通定義の jnl_fileless_option オペランドに Y を指定している場合，稼働数が 0 になります。

注※9

この値は，次に示す計算式で算出されます。

a×b

a：ジャーナルを二重化しているときは 2，二重化していないときは 1。

b：システムジャーナルファイルの並列アクセス機能での**最大分散数**（システムジャーナルサービス定義の jnl_max_file_dispersion オペランド指定値）。

注※10

この値は，次に示す計算式で算出されます。

a×b×c

a：ジャーナルを二重化しているときは 2，二重化していないときは 1。

b：アーカイブジャーナルファイルの並列アクセス機能での**最大分散数**（アーカイブジャーナルサービス定義の jnl_max_file_dispersion オペランド指定値）。

c：グローバルアーカイブジャーナルサービス定義で指定した**リソース数**（jnldfsv -a に指定するリソースグループ名の数）。

注※11

グローバルアーカイブジャーナルサービスがあるノードで稼働するプロセスです。

注※12

システムジャーナルサービス定義の jnl_auto_unload オペランドに Y を指定している場合，1 になります。

注※13

被アーカイブジャーナルノードで稼働するプロセスです。

注※14

アーカイブジャーナルノードに転送中の被アーカイブジャーナルノードの数と同じです。

注※15

ジャーナルサービス定義の jnldfsv -c で指定するチェックポイントダンプサービス定義数です。

注※16

システムサービス構成定義の que_conf オペランドに Y を指定している場合，1 になります。

注※17

システムサービス構成定義の dam_conf オペランドに Y を指定している場合，1 になります。

注※18

システムサービス構成定義の tam_conf オペランドに Y を指定している場合，1 になります。

注※19

システムサービス構成定義の ist_conf オペランドに Y を指定している場合，1 になります。

注※20

トレース取得用のプロセスです。このプロセスは，prcd の延長で作成されますが，このプロセスは，OpenTP1 のコマンド（prcls コマンド）では表示できません。状態を表示する場合は，OS のコマンド（ps コマンドなど）を実行してください。

注※21

システムの稼働数は，次のとおりです。

・システム共通定義の prf_trace オペランドに N を指定した場合

稼働数は 0 になります。

・システム共通定義の prf_trace オペランドに Y を指定した場合

MCF を使用しないとき，稼働数は 8 になります。

MCF を使用するとき，稼働数は 9 になります。

なお，これらの稼働数はプロセスサービス定義の prc_process_count オペランドに加算する必要はありません。

注※22

親プロセスは，dcstart プロセス終了後，init または systemd に遷移します。

注※23

クライアントサービス定義の clt_cup_conf オペランドに Y を指定した場合，1 になります。

注※24

システムサービス構成定義の clt_conf オペランドに Y を指定している場合，1 になります。

注※25

クライアントサービス定義の clt_trn_conf オペランドを省略，または Y を指定している場合，1 になります。

注※26

システムサービス構成定義の xat_conf オペランドに Y を指定した場合，1 になります。

注※27

システムサービス構成定義の rmm_conf オペランドに Y を指定した場合，1 になります。

注※28

MCF を使用しない（システムサービス構成定義に dcsvstart -m システムサービス名の記載がない）場合，0 になります。

注※29

MCF マネジャ定義の mcfmcomn -m mapsvname に指定がある場合，1 になります。

注※30

MCF マネジャ定義の mcfmcomn -o cmdsvname に指定がある場合，1 になります。

注※31

TP1/Messaging を使用した場合，"mcfutcpd"および"mcfupsvd"で固定となります。

注※32

MCF マネジャ定義の mcfmcname -s mcfsvname の指定数です。

注※33

MQ を使用しない（システムサービス構成定義の mqa_conf オペランドを省略，または N を指定している）場合，0 になります。

注※34

MQA サービス定義の mqa_ioproc_num オペランド（mqaiod）の指定値については，定義内で指定しているキューファイル数に依存します（ただし，二重化定義のバックアップのキューファイル数は除外となります）。

・mqa_ioproc_num の指定値≦キューファイル数の場合：サーバ数＝ mqa_ioproc_num の指定値

・mqa_ioproc_num の指定値＞キューファイル数の場合：サーバ数＝キューファイル数

注※35

・MQA サービス定義で mqa_mqc_conf オペランドを省略，または N を指定している場合，0 になります。

・MQA サービス定義で mqa_mqc_conf オペランドに Y を指定している場合，1 になります。

注※36

この値は，次に示す計算式で算出されます。

a ＋ b ＋ c

a：MQA サービス定義の mqamqtnam 定義コマンドで，プロトコル名称に mqttcp と指定している定義数

b：

・MQT サーバで UOC を使用していない場合，0 になります。

・MQT サーバで UOC を使用している場合，次の指定の合計値です。

・MQA サービス定義の mqamqtnam 定義コマンドで，プロトコル名称が mqtu で始まるシステムサービス定義名と指定している定義数

・MQA サービス定義の mqamqtnam 定義コマンドで，プロトコル名称が mqttcpcsu で始まるシステムサービス定義名と指定している定義数

・MQA サービス定義の mqamqtnam 定義コマンドで，プロトコル名称が mqttcpcru で始まるシステムサービス定義名と指定している定義数

c：

・MQA サービス定義で mqa_mqr_conf オペランドを省略，または N を指定している場合，0 になります。

・MQA サービス定義で mqa_mqr_conf オペランドに Y を指定している場合，MQA サービス定義の mqamqtnam 定義コマンドで，プロトコル名称が mqttcpcr および mqttcpcs と指定している定義数です。

注※37

・MQA サービス定義で mqa_mqc_conf オペランドを省略，または N を指定している場合，0 になります。

・MQA サービス定義で mqa_mqc_conf オペランドに Y を指定している場合，MQC サービス定義（$DCCONFPATH/mqc）の中の mqcgwpnam 定義コマンドの指定数になります。

注※38

・MQA サービス定義で mqa_mqr_conf オペランドを省略，または N を指定している場合，0 になります。

・MQA サービス定義で mqa_mqr_conf オペランドに Y を指定している場合，1 になります。

注※39

リモート API クライアントマネジャ機能を使用する場合，1 になります。

注※40

リモート API 機能未使用時は 0 になります。

リモート API 機能使用時に起動する rap リスナーサービス定義数です。

注※41

リアルタイム統計情報サービスを起動する場合，1 になります。

注※42

リアルタイム統計情報サービスの拡張機能を起動する場合，1 になります。

注※43

システムサービス構成定義の uto_conf オペランドに Y を指定している場合，1 になります。

## 7.2　OpenTP1 のメモリ構造

OpenTP1 で使うメモリには，**ローカルメモリ**と**共用メモリ**があります。

### 7.2.1　ローカルメモリ

ローカルメモリとは，一つのプロセスだけからアクセスできる仮想記憶領域です。この領域には，OpenTP1 のプロセス固有情報やバッファが取られます。大部分は UAP が RPC でサービス要求したとき動的に確保されますが，一部はライブラリオブジェクトのデータセクションに含まれます。

### 7.2.2　共用メモリ

共用メモリとは，複数のプロセスで共有して参照・更新するメモリです。主に，OpenTP1 の制御テーブルと共用バッファが取られます。OpenTP1 の共用メモリは，OS の共用メモリ機能を利用していて，プロセス間通信のオーバヘッドを削減するために使われます。

#### (1)　OpenTP1 が使う共用メモリ

OpenTP1 が使う共用メモリの量は，システム環境定義の static_shmpool_size オペランドに静的共用メモリの総量を，dynamic_shmpool_size オペランドに動的共用メモリの総量を指定します。OpenTP1 はこの二つを合わせた量の共用メモリを使います。通常は，OpenTP1 の稼働中に常に使うデータは静的共用メモリに，サービス処理の実行時にオンデマンドで占有するデータを動的共用メモリに設定します。

共用メモリのバッファ領域をユーザサーバ間で共用する指定をすると，共用メモリの量を節約できます。バッファテーブルの共用については，「3.4.4　バッファ領域の共用による共用メモリの節約」を参照してください。

#### (2)　DAM サービス，TAM サービス，IST サービスで使用する共用メモリ

動的共用メモリの一部は，DAM ファイルの更新に使用します。動的共用メモリを効率良く利用するため，同時に起動する DAM ファイルを更新するトランザクションブランチの数を，DAM サービス定義で指定します。トランザクションブランチの数を指定することで，DAM ファイル更新用の動的共用メモリを，OpenTP1 の開始時に確保します。定義した数を超えるトランザクションブランチが同時に，DAM ファイルを更新しようとした場合には，トランザクション実行時に新たな動的共用メモリを一時的に確保します。一時的に確保した動的共用メモリは，トランザクション終了後に解放します。

DAM サービスを使う場合，システム環境定義で指定した共用メモリとは別に，DAM ブロックを退避するためのバッファ領域用に共用メモリを確保します。DAM サービス用の共用メモリ領域は，DAM サービス定義で指定します。

TAM サービスを使用する場合，システム環境定義で指定した共用メモリとは別に，TAM テーブル用に共用メモリを確保します。TAM サービス用の共用メモリ領域は，TAM サービス定義で指定します。

IST サービスを使う場合，システム環境定義で指定した共用メモリとは別に，IST テーブル用の共用メモリを確保します。IST テーブルに使う共用メモリ領域は，IST サービス定義で指定します。

## (3) MCF サービスで使用する共用メモリ

MCF サービスは，静的共用メモリおよび動的共用メモリを使用します。

静的共用メモリについては，次に示す二つの指定を両方とも行ってください。

- MCF マネジャ共通定義の-p オプションに静的共用メモリの使用量を指定してください。
- システム環境定義の static_shmpool_size オペランドに，次の値を含めて指定してください。

```
MCFマネジャ共通定義のコマンドの-pオプションの指定値
＋すべてのMCF通信サービスのMCFトレースバッファサイズ※
＋すべてのアプリケーション起動サービスのMCFトレースバッファサイズ※
```

注※

MCF トレースバッファサイズの計算式を次に示します。

```
mcfttrc定義コマンドの-tオプションのsizeオペランドの指定値×2
```

MCF 静的共用メモリが不足した場合，静的共用メモリ（システム環境定義の static_shmpool_size オペランドの指定値）の未使用領域から，MCF マネジャ共通定義（mcfmcomn）の-p オプションで指定したサイズの 1/2 のサイズ分を自動的に追加確保します。メモリの自動確保は最大 254 回まで行います。それでもメモリが不足した場合は，ログメッセージとして KFCA10230-E を出力し，障害情報を出力します。ただし，メモリの自動確保回数が 254 回以内であっても，メモリの自動確保時に静的共用メモリ不足を検出した場合は，ログメッセージとして KFCA10240-E を出力して，障害情報を出力します。

メモリ追加確保時に，ログメッセージとして KFCA10242-I を出力するかどうかは，MCF マネジャ共通定義（mcfmcomn）の-i オプションで指定します。静的共用メモリ不足の発生を検知したい場合は，-i オプションに msg を指定し，メッセージを出力するように指定してください。

## (4) リアルタイム統計情報サービスで使用する共用メモリ

リアルタイム統計情報サービスは，システム環境定義に定義した共用メモリのほかに，取得した統計情報を格納するための RTS サービス用の共用メモリを使用します。

RTS サービス用の共用メモリのサイズは，リアルタイム統計情報サービス定義の rts_service_max オペランド，および rts_item_max オペランドの指定値によって決定されます。

## (5) OpenTP1 監視サービスで使用する共用メモリ

OpenTP1 監視サービスは，OpenTP1 監視機能用の共用メモリを使用します。

OpenTP1 監視機能用の共用メモリの所要量は固定です。詳細については，マニュアル「OpenTP1 システム定義」の共用メモリの見積もり式を参照してください。

## 7.3 OpenTP1 が使用する TCP/IP 資源

OpenTP1 の RPC（dc_rpc_call 関数，dc_rpc_call_to 関数，および TCP/IP を使用する XATMI インタフェースなど）を使用して大規模なシステムを構築する場合，TCP/IP のポート数の不足や通信エラーが発生することがあります。

使用ポート数の制限方法やネットワーク環境のチューニング方法などについて説明します。

### 7.3.1 OpenTP1 で使用しているポート番号

TCP/IP でプロセス間通信をする場合，プロセス間でコネクションを確立してからデータを送受信します。ここでは，OpenTP1 で使用しているポート番号について説明します。

#### (1) 送信用ポート番号

送信用ポート番号は，OS の自動割り当てポート（短命ポート）の範囲で OS に割り当てられて決定します。送信用ポート番号は固定できません。

#### (2) 受信用ポート番号

受信用ポート番号は，OS の自動割り当てポート（短命ポート）の範囲で OS に割り当てられて決定しますが，OpenTP1 の一部のシステムサービスでは，受信用ポート番号を指定することができます。

OpenTP1 で使用している受信用ポート番号について次に示します。

- OS が自動割り当てする範囲の受信用ポート番号

  この値は OS で決まっています。値の確認方法や変更方法については，使用している OS のマニュアルを参照してください。

  システム共通定義で rpc_port_base オペランドを指定している場合は，OS が自動割り当てする受信用ポート番号の範囲内で，「rpc_port_base オペランドの指定値」から「rpc_port_base オペランドの指定値＋プロセスサービス定義の prc_process_count オペランドの指定値＋ 128」までの範囲の番号を優先的に使用します。ただし，これらの範囲から任意に選択された受信用ポート番号が使用中だった場合，このオペランドの指定は無視されます。

  OpenTP1 が制御するウェルノウンポート化されていないプロセスの受信用ポート番号は，OS が任意に割り当てた番号になります。

- 個別に指定する受信用ポート番号

  例えば，次に示すオペランドの指定値です。

  - システム共通定義の name_port オペランドおよび prc_port オペランド
  - スケジュールサービス定義の scd_port オペランド
  - rap リスナーサービス定義の rap_listen_port オペランド

- システム共通定義の「rpc_port_base オペランドの指定値」から「rpc_port_base オペランドの指定値＋プロセスサービス定義の prc_process_count オペランドの指定値＋ 128」までの受信用ポート番号

受信用ポート番号を指定できるシステム定義のオペランドを，次の表に示します。

### 表 7-2　受信用ポート番号を指定できるシステム定義のオペランド

| 使用する機能 | 定義名 | オペランド名／定義コマンド名 | 指定できる範囲 | デフォルト値 | 対象システムサービス |
|---|---|---|---|---|---|
| － | システム共通定義 | name_port | 5001～65535 | 10000 | ネームサービス |
| | スケジュールサービス定義 | scd_port | | 自動割り当てポートの範囲 | スケジュールサービス |
| マルチスケジューラ機能 | スケジュールサービス定義 | scdmulti 定義コマンドの-p オプション | 5001～65535 | 自動割り当てポートの範囲 | マルチスケジューラ |
| クライアント拡張機能 | クライアントサービス定義 | clt_port | 5001～65535 | 自動割り当てポートの範囲 | クライアント拡張サービス |
| | | clttrn_port | | | トランザクショナル RPC 実行プロセス |
| | | cltcon_port | | | トランザクショナル RPC 実行プロセス |
| リモート API 機能 | rap リスナーサービス定義 | rap_listen_port | 5001～65535 | 自動割り当てポートの範囲 | rap リスナーサービス |
| | rap クライアントマネジャサービス定義 | rap_client_manager_port | | | rap クライアントマネジャサービス |
| マルチノード連携制御機能（TP1/Multi） | システム共通定義 | prc_port | 5001～49999 | 自動割り当てポートの範囲 | マルチノード連携制御機能のサービス |
| － | システム共通定義 | rpc_port_base | 5001～65535 | 自動割り当てポートの範囲 | OpenTP1 制御下のウェルノウンポート化されていないシステムサーバ，ユーザサーバ（prfiop，運用コマンドは除く）。 |

（凡例）

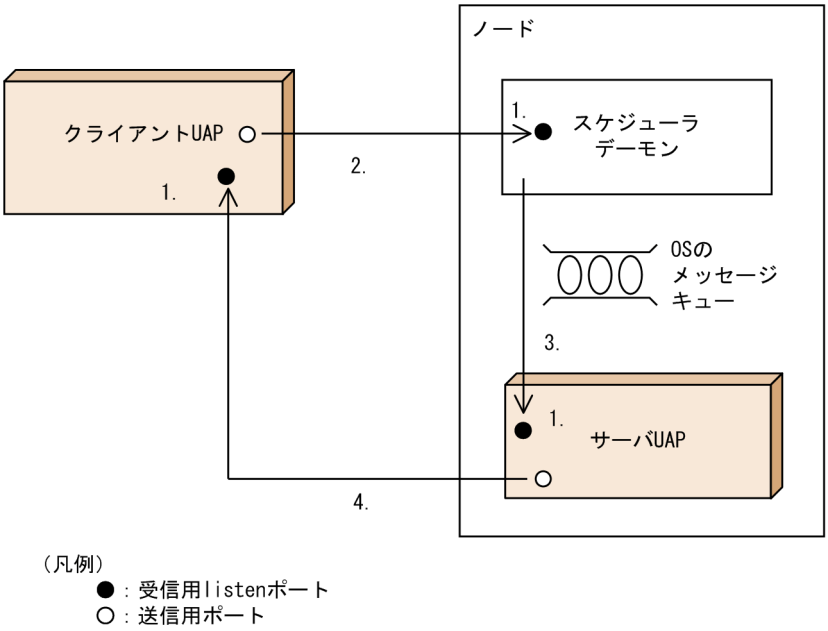
－：該当しません。

各オペランドの詳細については，マニュアル「OpenTP1 システム定義」を参照してください。

## 7.3.2　RPC によるポートの使用方法

RPC によるポートの使用方法について次の図に示します。

図 7-1　RPC によるポートの使用方法




1. OpenTP1 の各プロセス（システムプロセスおよびユーザプロセス）は，プロセス初期化処理で受信用ポートを一つ確保します。
2. RPC の発行時，クライアント UAP プロセスで送信用ポートを確保し，スケジューラデーモンの受信用ポートとコネクションを確立し，RPC を送信します。すでにコネクションが確立されている場合は，それを使用します。送信用ポートには，あて先プロセス（各ノードのスケジューラデーモン）ごとに異なる番号が割り当てられます。
3. OS のメッセージキューの機能で，サーバ UAP に連絡します。
4. RPC への応答時に，サーバ UAP プロセスは送信用ポートを確保し，クライアント UAP の受信用ポートとコネクションを確立し，応答を送信します。すでにコネクションが確立されている場合はそれを使用します。送信用ポートはあて先プロセス（クライアント UAP プロセス）ごとに異なる番号が割り当てられます。

   その後のコミット処理での送信では，すでにクライアント UAP とサーバ UAP との間にすでにコネクションが確立されているため，それを使用します。

## 7.3.3　ポート数の計算式

OpenTP1 で使用するポート数の計算式について次の表に示します。

表 7-3　OpenTP1 で使用するポート数の計算式

| 種類 | 計算式 |
|---|---|
| TCP/IP 通信受付<br>（受信用ポート） | OpenTP1 システムプロセス数＋ユーザプロセス数 |
| システムプロセス間通信用<br>（送信用ポート） | 他ノードと通信するシステムプロセス数※1×他ノード数 |
| RPC 送信用<br>（送信用ポート） | $\sum_{i=1}^{m} Ai + \sum_{j=1}^{n} Bj$<br>m：他ノードに RPC を実行する UAP プロセス数<br>n：他ノードに RPC を実行する rap サーバ数<br>A：該当する UAP i から RPC する他ノード数<br>B：該当する rap サーバ j から RPC する他ノード数 |
| RPC 応答用<br>（送信用ポート） | $\sum_{i=1}^{n} Ai \times 2$※3<br>n：RPC を受け付ける UAP プロセス数<br>A：該当する UAP i に RPC 要求を行ったクライアントプロセス数※2 |

注※1

次に示すプロセスの総数です。

- namd
- namaudtd（ノード監視機能を使用する場合）
- scdd
- scdmltd×（スケジュールサービス定義の scdmulti 定義コマンドの-m オプション指定値の総数）（マルチスケジューラ機能を使用する場合）
- istd（TP1/Shared Table Access を使用する場合）
- trnrmd
- trnrvd×（トランザクションサービス定義の trn_recovery_process_count オペランドの指定値）

注※2

応答送信先が TP1/EE の場合は，TP1/EE のクライアントプロセス（EE プロセス）数です。

注※3

RPC を受け付けた UAP プロセスがダウンした場合，admrsvre プロセスが代理でエラー応答を送信するため，admrsvre プロセスの応答送信用ポートとして計上します。

## 7.3.4　ポート数の制限方法

OpenTP1 を使用して大規模なシステムを構築する場合，TCP/IP が管理するポートが不足する可能性があります。このような場合に，OpenTP1 で使用するポートを制限する方法について説明します。

OpenTP1 は，コネクション確立時のオーバヘッドを削減するために，確立したコネクションを切断しないで保持し，同じプロセス間通信に再利用します。保持するコネクション数が上限になった場合，コネクションを確立したプロセス間の合意によって，コネクションを切断します。これを**一時クローズ処理**といいます。

ユーザは，システム共通定義，ユーザサービス定義，およびユーザサービスデフォルト定義で次に示すオペランドを指定できます。

```
set ipc_sockctl_highwater = a,b
set ipc_sockctl_watchtime = ソケット再利用可能監視時間
```

a：ソケットの一時クローズ処理開始数パーセンテージ

b：ソケットの一時クローズ処理非対象数パーセンテージ

- ソケットの一時クローズ処理開始数パーセンテージの指定

  OpenTP1 は，プロセス内のソケット用に使用しているファイル記述子の数が，次の値を超えた時点で，一時クローズ処理を開始します。

  ```
  max_socket_descriptorsオペランドの指定値
  ×（ipc_sockctl_highwaterオペランドの指定値a／100）
  ```

- ソケットの一時クローズ処理非対象数パーセンテージの指定

  一時クローズ処理の対象外とするコネクション数のパーセンテージを指定できます。一時クローズ処理の対象外となるコネクション数は，次の値です。

  ```
  max_socket_descriptorsオペランドの指定値
  ×（ipc_sockctl_highwaterオペランドの指定値b／100）
  ```

- ソケット再利用可能監視時間の指定

  ipc_sockctl_watchtime オペランドには，プロセス内のソケット用に使用しているファイル記述子の数が，max_socket_descriptors オペランドの指定値になった時点から，一時クローズ処理によってソケットが再利用できるようになるまでの監視時間（秒）を指定します。

同じプロセスとの間の通信が多く発生しない運用や，通信する相手プロセスが非常に多いシステムでは，保持しているコネクション数がある程度増えてきたところで，一時クローズ処理によって，適度にコネクションを解放して，1 プロセスで使用するソケット数を調整し，再利用できます。

また，OpenTP1 のプロセスから送信する場合は，コネクションを確立するときに送信用ポートを確保しますが，送信用ポートの個数は，一つのコンピュータで上限があるため，システム全体の合計が上限を超えないように，UAP プロセスが保持するコネクション数を調整する必要があります。

これらのオペランドの指定値が適切でないと，1 プロセス内で使用できるソケット数が上限に達してしまい，一時クローズ処理によるソケットの再利用が新たなコネクション確立要求に追いつかなかったり，システム全体で使用するポート数が TCP/IP の上限を超えてしまい，プロセスが異常終了することがあります。

### 7.3.5　一時クローズ処理とユーザ業務との関係

一時クローズ処理要求の送信は，1 コネクションに対して 1 回だけです。

一時クローズ処理要求の送信と応答の受信は，非同期に処理されるため，一時クローズ処理要求に対する応答がない状態が続いても，新しい RPC を受け付けたり，RPC に対する応答を送信することができるので，一時クローズ処理がユーザ業務に影響を与えることはありません。

### 7.3.6　一時クローズ処理要求の監視

一時クローズ処理要求監視機能を使用すると，一時クローズ処理要求が到着したかどうかを定期的に検査できます。この機能によって，OpenTP1 の SPP や MHP がサービス要求受信待ち状態でも，一時クローズ処理要求を受信できます。

一時クローズ処理では，コネクションを確立したプロセス間で，一時クローズ処理要求の送信，および一時クローズ処理要求の受信／応答の送信のやり取りがあって初めてコネクションを切断します。

つまり，コネクションの一方のプロセスで使用中のソケット数が最大数※に達して，一時クローズ処理要求を送信したとしても，もう一方のプロセスがこれを受信して応答を送信するまでは，コネクションを切断できません。OpenTP1 の SPP や MHP は，サービス要求の受信を待つ際に msgrcv システムコールを使用しています。そのため，いったん msgrcv システムコールで待ち状態に入ってしまうと，相手プロセスから一時クローズ処理要求が送信されたとしても，これを受信できず，コネクションを切断できません。

一時クローズ処理要求監視機能は，使用中のソケット数が最大数に達したプロセスが一時クローズ処理要求を送信した場合，相手プロセスが msgrcv システムコールでサービス要求受信待ちの状態でも，相手プロセスが一時クローズ処理要求の受信／応答の送信をできるようにするものです。ソケット用ファイル記述子の最大数は，max_socket_descriptors オペランドで設定します。

ユーザは，ユーザサービス定義，およびユーザサービスデフォルト定義で次に示すオペランドを指定できます。

```
set polling_control_data = ソケットの再利用指示（一時クローズ処理要求）の到着を検査するかどうか
set thread_yield_interval = ソケットの再利用指示（一時クローズ処理要求）を受信する契機を与えるインタバル時間
```

## 7.3.7　一時クローズ処理の実行状態の確認

一時クローズ処理の実行状態の情報を取得すると，一時クローズ処理が，ソケットを適度に再利用しながら実行されているかどうかを確認できます。この情報を基に，システムの規模，環境に合わせて，max_socket_descriptors オペランド，および，ipc_sockctl_highwater オペランドに最適な値を指定することで，オーバヘッドを削減できます。

一時クローズ処理の実行状態の情報を取得するには，rpcstat コマンドを使用します。このコマンドを実行すると，データ送受信処理を実行したあとに，一時クローズ処理の実行状態の情報を取得できます。一時クローズ処理の実行状態の情報は，データ送受信処理が実行されるごとには取得できません。データ送信，またはデータ受信処理が 10 回実行された場合，1 回の割合で取得できます。

rpcstat コマンドで取得できる情報を次の表に示します。

### 表 7-4　一時クローズ処理の実行状態の確認コマンドで取得できる情報

| 取得項目 | 説明 |
|---|---|
| max_socket_descriptors オペランドの設定値 | ソケット用ファイル記述子の最大数。 |
| ipc_sockctl_highwater=a,b の a の値 | ソケットの一時クローズ処理開始数パーセンテージ。 |
| ipc_sockctl_highwater=a,b の b の値 | ソケットの一時クローズ処理非対象数パーセンテージ。 |
| プロセス間通信で使用中のソケット数 | プロセス間通信で使用している，受け付け用，接続中，および接続済み状態のソケットの総数。 |
| 一時クローズ処理要求中のソケット数 | 該当プロセスが使用しているソケット数が一時クローズ処理開始数パーセンテージに達し，コネクションを確立しているプロセスとの一時クローズ処理の合意を待っている状態のソケットの総数。 |
| 一時クローズ処理済みのソケット数 | 該当プロセス，または，コネクション確立プロセスが使用しているソケット数が一時クローズ処理開始数パーセンテージに達し，コネクションを確立しているプロセスとの合意によって一時クローズが完了したソケットの総数。 |
| 一時クローズ処理送信拒否のソケット数 | コネクションを確立しているプロセスと一時クローズ処理の合意が行えないソケットの総数。 |
| 受信用ポート番号 | 該当プロセスが使用している受信用ポート番号。 |

## 7.3.8　ノード間通信時の毎回コネクション断機能

ノード間通信時の毎回コネクション断機能は，ノードをわたった OpenTP1 プロセス間の TCP/IP 通信の場合に限り，通信処理を終了した時点で毎回コネクションを切断するものです。

OpenTP1 は，次の処理でノード間の TCP/IP 通信をします。

- dc_rpc_call 関数の通信
- dc_rpc_call_to 関数の通信

- UAP とシステムサーバ間の通信
- OpenTP1 のシステムサーバ間の通信

ノード間通信時の毎回コネクション断機能は，ユーザサービス定義，ユーザサービスデフォルト定義，またはシステム共通定義の rpc_close_after_send オペランドに指定できます。

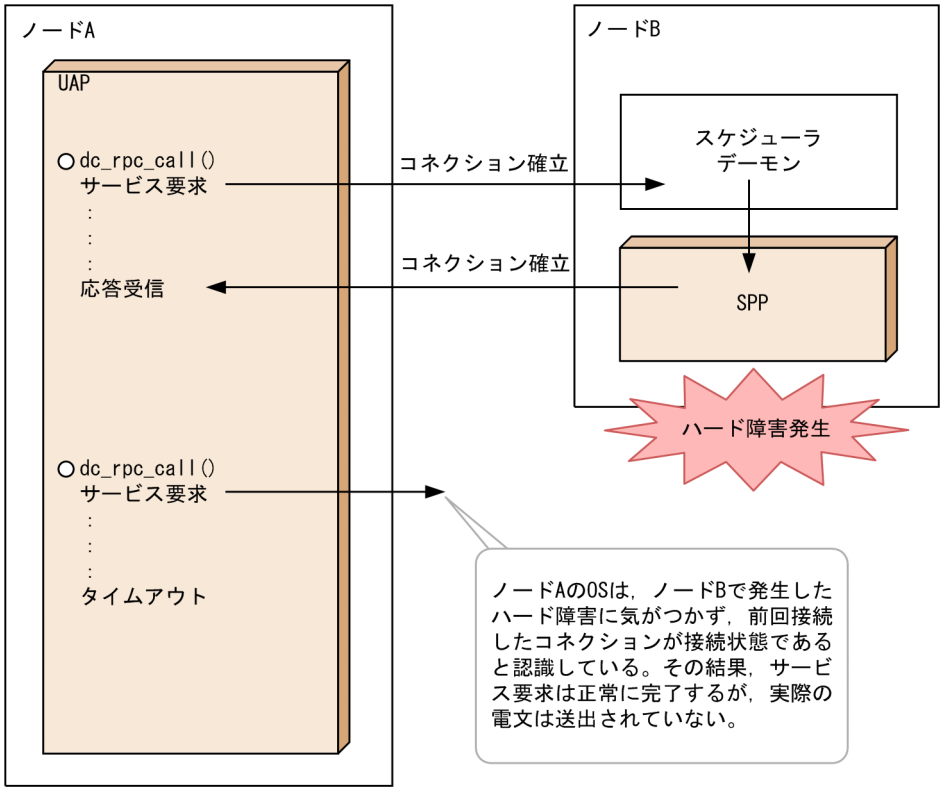
なお，次に示す機能では，ノード間の TCP/IP 通信であっても，ノード間通信時の毎回コネクション断機能は適用されません。

- OpenTP1 クライアント製品と TP1/Server Base 間の通信
- リモート API 機能を使用し，rap クライアントから TP1/Server Base の rap リスナー，rap サーバへの通信
- XA リソースサービスを使用したトランザクション連携機能
- TP1/Message Queue のチャネルを使用した，他の MQ システム（メッセージキューイング機能のキューマネジャがあるシステム）と TP1/Message Queue とのチャネル通信
- TP1/Message Queue のクライアント製品（TP1/Message Queue Access）である MQC クライアント機能と TP1/Message Queue の MQC サーバ機能との通信
- メッセージ制御機能（TP1/Message Control）を使用した，相手システムとの通信

## (1) 機能概要

ノード A の OpenTP1 の UAP から，dc_rpc_call によってノード B の OpenTP1 の提供するサービスを実行したあと，次の状態になった場合，ノード A の UAP はノード B の SPP やスケジューラデーモンとの間のコネクションが切断されたことを検知できません。

- ノード B の OpenTP1 がハード障害によってダウンした
- ノード A とノード B 間に無通信監視をする通信機器によってコネクションが切断された
  それは，ノード B の OS がコネクションの切断を通知できないからです。この状態でノード A の UAP がノード B のサービスを要求した場合，dc_rpc_call の送信処理は正常に完了しますが，実際に電文は送出されません。その結果，ノード A の UAP はタイムアウトによって異常を検知します。




このように OS がコネクションの切断を通知できないような障害を即時に検知するには，電文の送受信が終了するたびにコネクションを切断し，毎回新たにコネクションを確立する操作を実行する必要があります。

ノード間通信時の毎回コネクション断機能は，通信が終了した時点で毎回コネクションを切断する機能であり，コネクションを切断するのはノードをわたったプロセス間の通信の場合だけです。

ノード間通信時の毎回コネクション断機能は，他ノード上のプロセスと通信するたびにコネクションの確立／解放処理を実行するため，このオーバヘッドによって性能が低下することが考えられます。また，ノードをわたったプロセス間通信を行うたびに，コネクションを確立するため，OS の資源であるポートを消費します。一度使用したポートは，OS によって一定時間経過後に，再利用できます。ただし，再利用できるまでの時間は OS の設定に依存します。ポートの再利用間隔時間を長く設定していると，ポート不足が発生し，OpenTP1 だけでなく，他製品にも影響が出ます。ポートに関する設定などを十分に検討し，この機能を使用してください。ポートに関する OS の設定の詳細は，OS のマニュアルを参照してください。

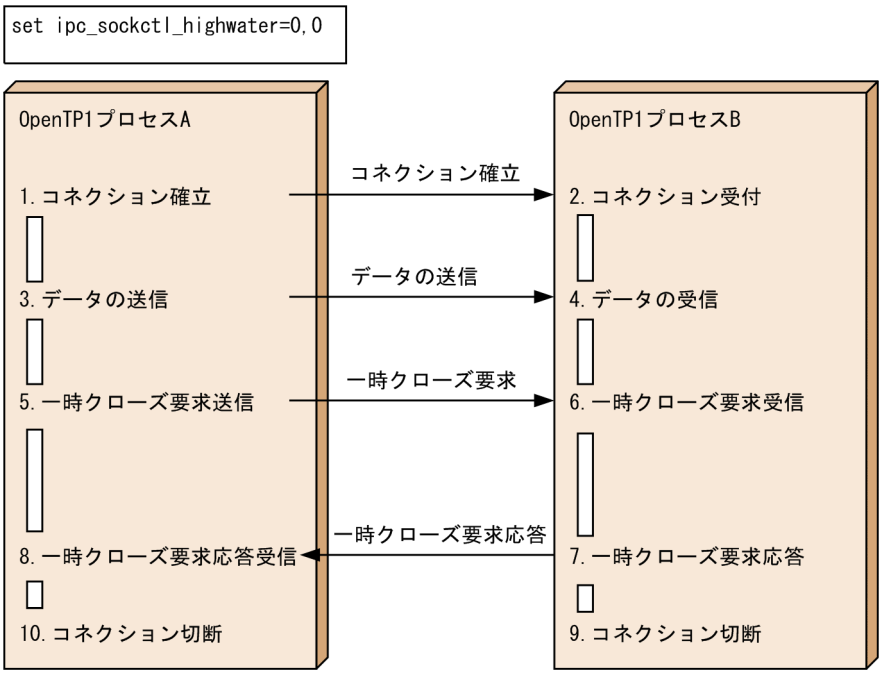
## (2)　一時クローズ処理とノード間通信時の毎回コネクション断機能との比較

一時クローズ処理は，保持するコネクション数が設定（ipc_sockctl_highwater 定義オペランド）の上限になった場合，コネクションを確立したプロセス間の合意によって，コネクションを切断します。一時クローズ処理は，自ノード内通信およびノード間通信で，すべてのコネクションに対して有効です。

OpenTP1 制御下のプロセスに ipc_sockctl_highwater=0,0 を設定すると，ノード間通信時の毎回コネクション断機能と同様に，通信ごとにコネクションを切断できます。しかし，ユーザ処理中や OpenTP1 のライブラリ関数実行中でも電文受信処理を実施していない場合は，一時クローズ処理が実行できません。これは，一時クローズ要求電文が受信できないため，プロセス間の合意が取れないからです。

OpenTP1 プロセス A に ipc_sockctl_highwater=0,0 を設定した場合の処理概要を次の図に示します。

図 7-2　一時クローズ処理概要




1. プロセス A から OpenTP1 プロセス B へコネクション確立要求。
2. プロセス B でコネクション確立要求受付。
3. プロセス A からプロセス B に対して，電文送信。
4. プロセス B において，電文受信。
5. プロセス A において，ipc_sockctl_highwater 定義オペランドの条件を満たしたため，プロセス B に対して，一時クローズ要求を送信。
6. プロセス B において，一時クローズ要求を受信。※1
7. プロセス B から一時クローズ要求の応答送信。
8. プロセス A において，一時クローズ要求応答受信。※1
9. プロセス B において，コネクションの切断。※2
10. プロセス A において，コネクションの切断。※2

注※1

一時クローズ要求および一時クローズ要求応答受信は，プロセスの処理が OpenTP1 のライブラリ関数を実行し，かつ，電文を受信できる状態にいる場合に受信します。ユーザ処理中や OpenTP1 のライブラリ関数実行中であっても電文受信処理を実施していない場合は，受信できません。

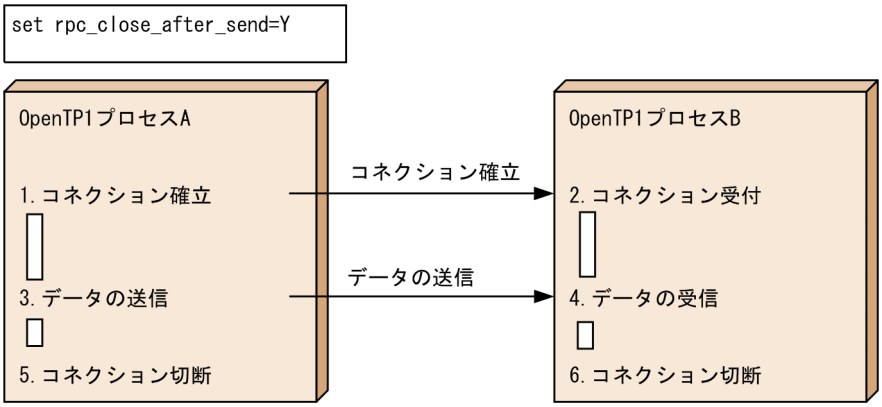
注※2

コネクションの切断の順序は，プロセス A とプロセス B 間で同期を取らずに行うため，順序が逆転するおそれがあります。

コネクションを切断時に，一時クローズ処理は，コネクションを確立したプロセス間で合意を取りますが，ノード間通信時の毎回コネクション断機能は，プロセス間の合意を取りません。そのため，一時クローズ処理と比較して，即時（電文送信直後，または電文受信直後）にコネクションの切断を実施できます。ノード間通信時の毎回コネクション断機能を実施できるのは，ノード間のコネクションだけです。

この機能の処理概要を次の図に示します。

### 図 7-3　ノード間通信時の毎回コネクション断オプション機能概要




1. プロセス A から OpenTP1 プロセス B へコネクション確立要求。
2. プロセス B でコネクション確立要求受付。
3. プロセス A からプロセス B に対して，電文送信。
4. プロセス B において，電文受信。
5. プロセス A において，コネクションの切断。※
6. プロセス B において，コネクションの切断。※

注※

コネクションの切断の順序は，プロセス A とプロセス B 間で同期を取らずに行うため，順序が逆転する可能性があります。

一時クローズ処理（ipc_sockctl_highwater=0,0 を設定）とノード間通信時の毎回コネクション断機能を適用した場合の機能比較を次の表に示します。

| 比較項目 | 一時クローズ処理<br>ipc_sockctl_highwater=0,0 | ノード間通信時の毎回コネクション断オプション機能 |
|---|---|---|
| OpenTP1 の通信性能 | × | ○ |
| コネクション切断検知の即時性 | × | ○ |
| コネクション切断の範囲 | ○ | × |
| 使用するポート数 | △ | △ |

（凡例）

○：優位

×：劣位
△：同等

コネクション切断検知の即時性を重視する場合はノード間通信時の毎回コネクション断オプション機能を使用してください。

コネクション切断の範囲を重視する場合は，一時クローズ処理を使用してください。

### (3) ノード間通信時の毎回コネクション断機能適用の注意事項

コネクションには，TCP のポートがくくりついています。TCP のポートの状態遷移には，TIME_WAIT の状態があります。OS は，ネットワーク上で遅れていたパケットが到着する可能性があるので，切断したコネクションで使用されていたポートを一定時間，再利用しないようにします。一定時間，再利用されないポートの状態を TIME_WAIT 状態といいます。詳細は，TCP/IP の文献を参照してください。

ノード間通信時の毎回コネクション断機能を使用すると，コネクションの切断が頻繁に行われ，短命ポート（Ephemeral port）が一定時間，TIME_WAIT 状態で残ります。TIME_WAIT 状態のポートは，再利用できないため，ポート不足が発生するおそれがあります。ポート不足が発生すると，通信障害が発生し，OpenTP1 がダウンする場合があります。

OS のコマンド（netstat など）で，ノードのポートの状態を確認し，ノード間通信時の毎回コネクション断オプション機能を使用しても，システムで使用できるポート数の上限値を超えない場合に適用してください。

## 7.3.9 ソケット数増加による資源の増減

OpenTP1 は，max_socket_descriptors オペランドの指定値に対応して，内部管理テーブルを確保します。max_socket_descriptors オペランドの指定値を増やすと OpenTP1 が確保する malloc エリアが増加します。

また，一時クローズ処理を開始すると，一時クローズ処理中のソケットを管理するためにも，malloc エリアが確保されます。ただし，一時クローズ処理要求に対する応答を受信してソケットがクローズされたあと，他コネクションに再利用される時点で解放されます。

## 7.3.10 ネットワーク環境のチューニング方法

大規模システムを構築する場合に，ネットワーク環境によって発生しやすい障害を防止する方法について説明します。

OpenTP1 の RPC は，TCP/IP のタイマ監視値を OS が実装しているタイマ値よりも小さい値で動作するように制御します。WAN を経由した遠隔地との通信など，伝送効率の悪いネットワークを使用した場合，通信障害が発生することがあります。

このような場合，ユーザは，システム共通定義，ユーザサービス定義，およびユーザサービスデフォルト定義で次に示すオペランドを指定することで，問題を解消できます。

```
set ipc_conn_interval = コネクション確立監視時間
set ipc_send_interval = データ送信監視間隔
set ipc_send_count = データ送信監視回数
set ipc_header_recv_time = 通信制御データの受信監視時間
set ipc_backlog_count = コネクション確立要求を格納するキューの長さ
```

- コネクション確立監視時間の指定

  送信先プロセスとのコネクションが未確立の場合，OpenTP1 は，TCP/IP の connect()システムコールを呼び出します。このとき，ソケットをノンブロッキングモードで使用しているため，すぐにコネクションが確立できない場合，select()システムコールを使用して TCP/IP からのコネクション確立完了のイベントを受信するまで監視します。

  ipc_conn_interval オペランドを指定することによって，select()システムコールを使用して TCP/IP からのコネクション確立完了のイベントを受信するまでの監視時間を指定できます。

- データ送信監視間隔およびデータ送信監視回数の指定

  コネクション確立後，OpenTP1 は，write()システムコールを使用して TCP/IP 通信バッファに送信データを書き込みます。

  WAN を経由する通信などでネットワークの伝送品質が悪くパケットの消失が繰り返し発生する場合や，TCP/IP 通信バッファ以上のデータを送信する場合は，TCP/IP 通信バッファに送信データを書き込めないで write()システムコールのリトライが発生する可能性があります。

  ipc_send_interval オペランドを指定することによって，write()システムコールのデータ送信監視間隔を指定できます。

  ipc_send_count オペランドを指定することによって，write()システムコールのデータ送信監視回数を指定できます。

- 通信制御データの受信監視時間の指定

  データ受信の開始後に，ネットワークの障害などによって OpenTP1 の通信制御データを正しく受信できない状態になると，他プロセスからのコネクション設定要求や新しいデータ受信処理を開始できない状態が続いてしまいます。

  ipc_header_recv_time オペランドを指定することによって，通信制御データを受信するまでの時間を監視できます。

- コネクション確立要求を格納するキュー（リッスンキュー）の長さの指定

  OpenTP1 は，コネクション接続要求を受信するソケットに対し，listen()システムコールを発行します。listen()システムコールに指定するリッスンキューの長さには，OpenTP1 をコンパイルした環境の OS が定義している SOMAXCONN を指定しています。SOMAXCONN の値については，「リリースノート」を参照してください。一度に大量のコネクション接続要求を受信すると，リッスンキューが不足し，コネクション接続元のコネクション接続要求はエラーリターンする可能性があります。システム共通定義の ipc_backlog_count オペランドを指定すると，リッスンキューの長さを変更できます。リッスンキューに指定できる最大値，および SOMAXCONN の値は動作環境の OS によって異なりま

す。そのため，ipc_backlog_count オペランドに動作環境の SOMAXCONN 以上の値を指定する場合，動作環境の OS のマニュアルを参照してこれらの値を確認し，指定する値を決定してください。

## 7.3.11　DNS，NIS 使用時の注意事項

OpenTP1 は，定義されたホスト名から IP アドレスを変換したり，RPC 通信先を求めたりするために，DNS や NIS を使用することがあります。これらの機能を使用するかどうかは OpenTP1 を使用する OS 環境の設定によって決定します。

DNS，NIS が障害になると，トランザクション処理時間が長くなったり，OpenTP1 が開始できなくなったり，または UAP が開始できなくなったりする障害が発生する可能性があります。

障害発生を防ぐために，次の観点からシステム設計をしてください。

高信頼性に対する観点

DNS サーバ，NIS サーバ障害時について設計する必要があります。

交代先の設計や，自サーバ内での解決方法を設計して信頼性を高めてください。

性能面に対する観点

DNS，NIS を使用する場合，ホスト名からアドレスを求めるときに通信が発生することがあります。

そのオーバヘッドを考慮してシステムを設計してください。

# 8 Docker および Kubernetes を使用した OpenTP1 の構成と構築

この章では，Docker および Kubernetes を使用した OpenTP1 の構成と構築について説明します。

なお，Docker および Kubernetes を使用した OpenTP1 のシステム構築の詳細については，取扱説明書を参照してください。

## 8.1　この章を読む際の留意事項

この章を読む際の留意事項を次に示します。

- この章は，Docker，Kubernetes の基本的な製品知識がある読者を対象としています。この章をお読みになる前に，各製品のマニュアルなどで各製品の基本知識を習得いただきますよう，お願いいたします。
- この章では，Docker イメージや，Pod などの Kubernetes に関する各種オブジェクトの作成方法など，環境構築方法については記載していません。TP1/Server Base に添付されている取扱説明書「Docker，Kubernetes を使用した OpenTP1 システムの構築」を参照してください。
- この章では，Kubernetes の Service オブジェクトを Service と表記しています。また，「スケジューラダイレクト機能を使用した RPC」および「通信先指定 RPC」を合わせて，「スケジューラダイレクト機能を使用した RPC」と表記しています。
- この章では，各機能を C 言語の関数名で説明します。C 言語の関数名に対応する COBOL 言語の API は，関数名を最初に使用する個所に【 】で囲んで表記します。それ以降は，C 言語の関数名に統一して説明します。

## 8.2 概要

Docker および Kubernetes を使用した OpenTP1 の概要を説明します。

### 8.2.1 コンテナサポート

次に示す製品は，Docker，Kubernetes を使用したコンテナ環境でも動作できるため，コンテナ内の TP1/Server Base と RPC 通信できます。

- Linux 版 TP1/Server Base 07-56 以降
- TP1/Client/J 07-53 以降

次に示す製品は，Docker，Kubernetes を使用したコンテナ環境では動作できませんが，応答電文受信用 IP アドレス通知機能を使用することでコンテナ内の上記製品と RPC 通信できます。

- Windows 版 TP1/Server Base 07-60 以降

### 8.2.2 前提

次のソフトウェアを前提とします。

#### (1) 前提ソフトウェア

##### (a) コンテナ

Docker

##### (b) オーケストレーションツール

Kubernetes

##### (c) コンテナ OS

Red Hat Enterprise Linux Server

#### (2) 前提条件

コンテナ内で OpenTP1 を使用する場合の前提条件を次に示します。

##### (a) Docker に関する前提条件

- 1 つの Pod に配置できる TP1/Server Base，TP1/Client/J が動作するコンテナは 1 つだけです。
- TP1/Server Base と TP1/Client/J を 1 つのコンテナにインストールできません。

- コンテナ内で使用可能なネットワークアダプタは 1 つだけです。

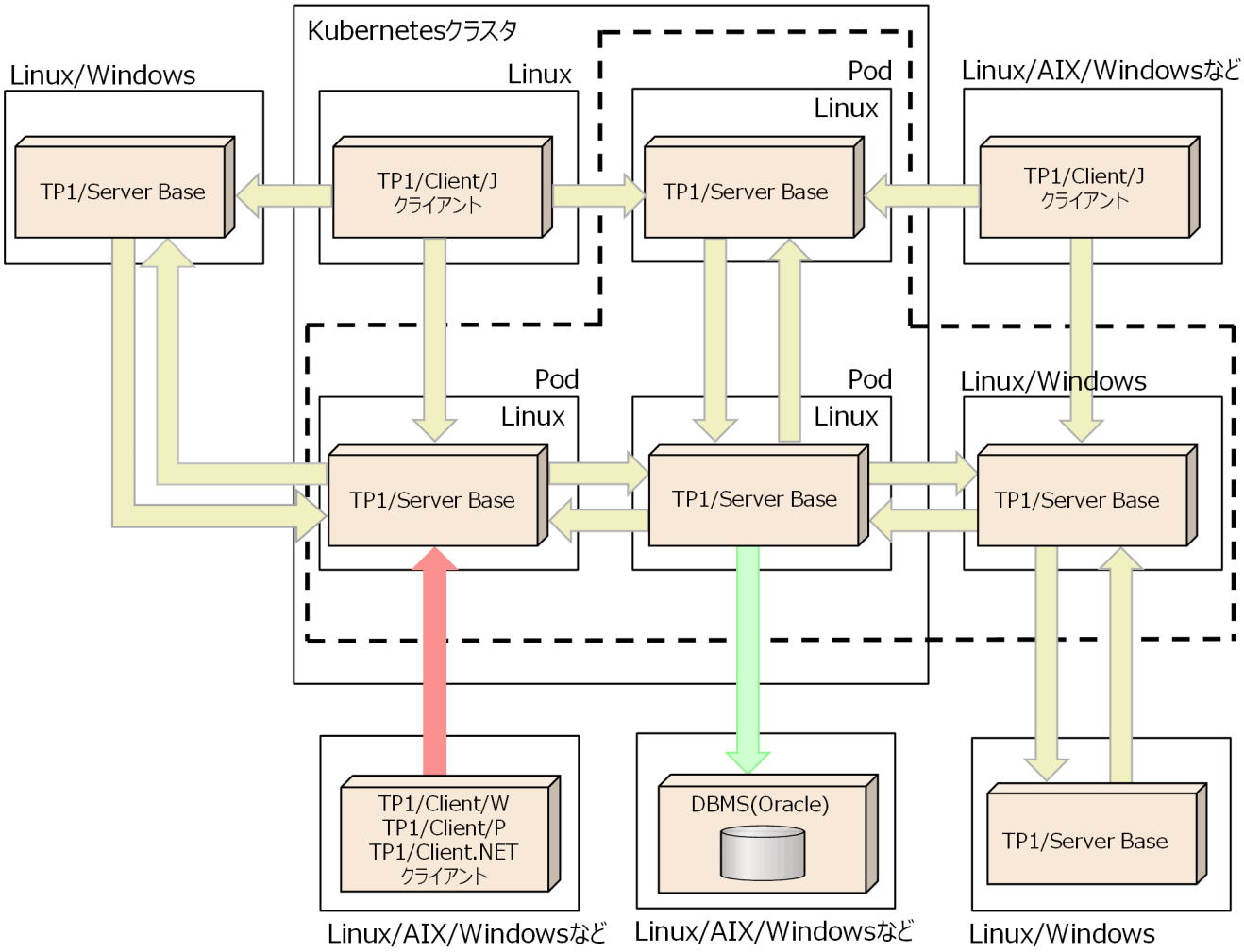
### (b) Kubernetes に関する前提条件

- Kubernetes のスケール制御に関する機能は使用できません。Deployment などのワークロードコントローラを使用したスケールイン・スケールアウト機能を使用しないでください。

## 8.3　システム構成

Docker，Kubernetes を使用した OpenTP1 の構成例を次に示します。

図 8–1　システム構成




注：次に示すバージョンを使用した場合の例です。
・Linux版 TP1/Server Base 07-56以降
・Windows版 TP1/Server Base 07-60以降
・TP1/Client/J 07-53以降

（凡例）

：all_nodeによるシステム連携

：リモートAPIを使用したRPC，
ネームサービスを使用したRPC，
スケジューラダイレクト機能を使用したRPCが可能

：リモートAPIを使用したRPC，
スケジューラダイレクト機能を使用したRPCが可能

：DBアクセス

リモート API 機能以外の RPC を使用する場合，次に示すバージョンを使用する必要があります。

- Linux 版 TP1/Server Base 07-56 以降
- Windows 版 TP1/Server Base 07-60 以降
- TP1/Client/J 07-53 以降

上記以外の OpenTP1 間で通信可能なバージョンの詳細については，「8.3.2　旧バージョンとの RPC 連携」を参照してください。

## 8.3.1　トランザクション連携

TP1/Server Base 間や TP1/Client/J からの RPC により，XA インタフェースに準拠した DBMS とトランザクション連携ができます。コンテナ環境でのトランザクション連携可否を次に示します。

### 表 8–1　トランザクション連携可否

| # | TP1/Server Base | DBMS※ | 連携可否 |
|---|---|---|---|
| 1 | コンテナ内 | コンテナ内 | × |
| 2 | | コンテナ外 | ○ |
| 3 | コンテナ外 | コンテナ内 | × |
| 4 | | コンテナ外 | ○ |

（凡例）

○：連携可能

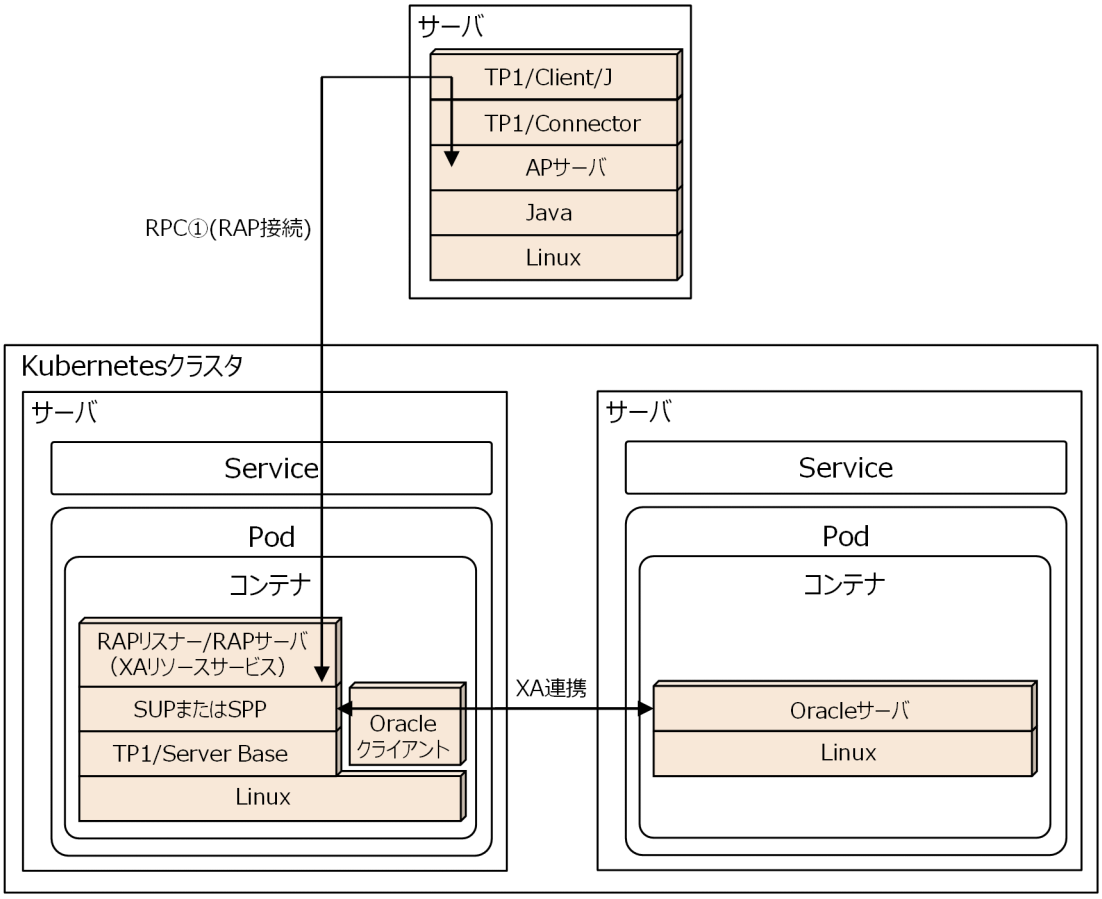
×：連携不可能

注※

DBMS は Oracle と HiRDB をサポート。

AP サーバからコンテナ内の TP1/Server Base に接続し，DBMS（コンテナ内の Oracle）とトランザクション連携する場合のシステム構成例を次の図に示します。

図 8-2　Oracle とのトランザクション連携




## (1)　サポートバージョン

コンテナ外の AP サーバ上の TP1/Connector 経由でコンテナ内の OpenTP1 に接続し，XA リソースサービスを使用してトランザクション制御を行う場合は，次のバージョンを使用してください。

表 8-2　サポートバージョン

| # | RPC 通信区間 | TP1/Client/J のバージョン | TP1/Server Base のバージョン |
|---|---|---|---|
| 1 | RPC① | 07-50 以降 | • Linux 版の場合：07-56 以降<br>• Windows 版の場合：07-60 以降 |

## 8.3.2　旧バージョンとの RPC 連携

旧バージョン（応答電文受信用 IP アドレス通知機能をサポートしていないバージョン）の TP1/Server Base，TP1/Client/J について，RPC 通信可否を次に示します。

## (1) 応答電文受信用 IP アドレス通知機能をサポートしている TP1/Server Base と旧バージョンとの RPC 通信可否

表 8-3 旧バージョンの TP1/Server Base からの RPC 通信可否

| # | 旧バージョンからの RPC 種別 | RPC 要求先 | | | |
|---|---|---|---|---|---|
| | | (コンテナ内) ipc_notify_response_host | | (コンテナ外) ipc_notify_response_host | |
| | | Y | N | Y | N |
| 1 | ネームサービスを使用した RPC | × | × | × | ○ |
| 2 | スケジューラダイレクト機能を使用した RPC | × | × | × | ○ |
| 3 | リモート API を使用した RPC | ○ | ○ | ○ | ○ |

(凡例)

○：通信可能

×：通信不可能

表 8-4 旧バージョンの TP1/Client/J からの RPC 通信可否

| # | 旧バージョンからの RPC 種別 | RPC 要求先 | | | |
|---|---|---|---|---|---|
| | | (コンテナ内) ipc_notify_response_host | | (コンテナ外) ipc_notify_response_host | |
| | | Y | N | Y | N |
| 1 | ネームサービスを使用した RPC | × | × | × | ○ |
| 2 | スケジューラダイレクト機能を使用した RPC | × | × | × | ○ |
| 3 | リモート API を使用した RPC | ○ | ○ | ○ | ○ |

(凡例)

○：通信可能

×：通信不可能

表 8-5 旧バージョンの TP1/Server Base への RPC 通信可否

| # | RPC 要求元 | | | 旧バージョンへの RPC 種別 | | |
|---|---|---|---|---|---|---|
| | | | | ネームサービスを使用した RPC | スケジューラダイレクト機能を使用した RPC | リモート API を使用した RPC |
| 1 | コンテナ内 | ipc_notify_response_host | Y | × | × | ○ |
| 2 | | | N | ○ | ○ | ○ |
| 3 | コンテナ外 | ipc_notify_response_host | Y | × | × | ○ |

| # | RPC 要求元 | | | 旧バージョンへの RPC 種別 | | |
|---|---|---|---|---|---|---|
| | | | | ネームサービスを使用した RPC | スケジューラダイレクト機能を使用した RPC | リモート API を使用した RPC |
| 4 | コンテナ外 | ipc_notify_response_host | N | ○ | ○ | ○ |

（凡例）

○：通信可能

×：通信不可能

## (2) 応答電文受信用 IP アドレス通知機能をサポートしている TP1/Client/J と旧バージョンとの RPC 通信可否

表 8-6 TP1/Client/J から旧バージョンの TP1/Server Base への RPC 通信可否

| # | RPC 要求元 | | | 旧バージョンへの RPC 種別 | | |
|---|---|---|---|---|---|---|
| | | | | ネームサービスを使用した RPC | スケジューラダイレクト機能を使用した RPC | リモート API を使用した RPC |
| 1 | コンテナ内 | dcnotifyreshost | Y | × | × | ○ |
| 2 | | | N | × | ○ | ○ |
| 3 | コンテナ外 | dcnotifyreshost | Y | × | × | ○ |
| 4 | | | N | × | ○ | ○ |

（凡例）

○：通信可能

×：通信不可能

## 8.3.3 他製品との RPC 連携

RPC インタフェースを持つ他製品との RPC 通信可否を次に示します。

## (1) コンテナ内の TP1/Server Base に対し RPC 要求を行う製品

表 8-7 他製品から TP1/Server Base への RPC

| # | RPC インタフェースを持つ他製品（コンテナ外） | RPC 要求先 | |
|---|---|---|---|
| | | コンテナ内（ipc_notify_response_host=Y） | コンテナ内（ipc_notify_response_host=N） |
| 1 | TP1/LiNK | △ | △ |

| # | RPC インタフェースを持つ他製品（コンテナ外） | | RPC 要求先 | |
|---|---|---|---|---|
| | | | コンテナ内（ipc_notify_response_host=Y） | コンテナ内（ipc_notify_response_host=N） |
| 2 | TP1/Client/J | 07-51 以前 | △ | △ |
| 3 | | 07-52 以降（dcnotifyreshost=N） | △ | △ |
| 4 | | 07-52 以降（dcnotifyreshost=Y） | ○ | △ |
| 5 | TP1/Client/P | | △ | △ |
| 6 | TP1/Client/W | | △ | △ |
| 7 | TP1/Client/J for .NET Framework | | △ | △ |
| 8 | TP1/Server Base Enterprise Option | | × | × |
| 9 | Hitachi System Information Capture（HSIC） | | × | × |

（凡例）

○：通信可能

△：リモート API 機能だけサポート

×：通信不可能

## (2)　コンテナ内の TP1/Server Base から RPC 要求を受ける製品

### 表 8–8　TP1/Server Base から他製品への RPC

| # | RPC インタフェースを持つ他製品（コンテナ外） | RPC 要求先 | |
|---|---|---|---|
| | | コンテナ内（ipc_notify_response_host=Y） | コンテナ内（ipc_notify_response_host=N） |
| 1 | TP1/LiNK | × | ○ |
| 2 | TP1/Server Base Enterprise Option | ○ | ○ |
| 3 | Hitachi System Information Capture（HSIC） | × | ○ |
| 4 | TMS-4V/SP/Server※1 | ○ | ○ |
| 5 | TMS-4V/SP/RemoteProcedureCall※1 | ○ | ○ |
| 6 | DCCM3/SERVER/TP1※1（VOS3 XDM/DCCM3,VOS1 DCCM3） | ○ | ○ |
| 7 | uCosminexus Application Server※2（TP1 インバウンドアダプタ） | × | ○ |

(凡例)

○：通信可能

×：通信不可能

注※1

該当製品がサポートしているのはリモート API 機能だけです。

注※2

該当製品がサポートしているのはスケジューラダイレクト機能だけです。

## 8.4　サポート機能

コンテナ内の OpenTP1 では，未サポート機能や，制限付きでサポートする機能があります。次に示す機能制限内での運用をお願いいたします。

### 8.4.1　サポート機能の詳細

コンテナ内の OpenTP1 でサポートする機能を次の表に示します。

表 8–9　TP1/Server Base のサポート機能

| # | 機能名 | | | サポート | 特記事項 |
|---|---|---|---|---|---|
| 1 | 統合システム運用管理機能 JP1 | | | × | − |
| 2 | トランザクション制御 | 分散トランザクション | | ○ | − |
| 3 | | XA リソースサービスによるトランザクション制御 | | × | − |
| 4 | | リソースマネジャ連携 | | ○ | 次のリソースマネジャが使用できます。<br>• TP1/FS/Direct Access<br>• TP1/FS/Table Access<br>• Oracle |
| 5 | クライアント/サーバ形態の通信 | OpenTP1 のリモートプロシジャコール（RPC）通信 | | ○ | 詳細は，「8.4.2　RPC 通信」を参照してください。 |
| 6 | | サービス情報検索の付加機能 | グローバル検索機能 | ○ | − |
| 7 | | | サービス情報優先度指定機能 | ○ | TP1/Server Base 07-57 以降でサポートしています。 |
| 8 | | OpenTP1 のノード管理 | 起動通知機能 | ○ | − |
| 9 | | | ノード監視機能 | ○ | − |
| 10 | | | ノード自動追加機能 | ○ | TP1/Server Base 07-57 以降でサポートしています。 |
| 11 | | XATMI インタフェースの通信 | | × | − |
| 12 | | TxRPC インタフェースの通信 | | × | − |
| 13 | メッセージ制御 | メッセージ制御機能 | | × | − |
| 14 | アプリケーションプログラムのスケジュール | プロセスの制御 | マルチサーバ負荷バランス機能 | ○ | − |
| 15 | | | スケジュールの優先度制御 | ○ | − |
| 16 | | | 非常駐 UAP プロセスのリフレッシュ機能 | ○ | − |

| # | 機能名 | | | サポート | 特記事項 |
|---|---|---|---|---|---|
| 17 | アプリケーションプログラムのスケジュール | プロセスの制御 | ノード間負荷バランス機能 | ○ | − |
| 18 | | | マルチスケジューラ機能 | ○ | − |
| 19 | | バッファ領域の共用による共用メモリの節約 | | ○ | − |
| 20 | OpenTP1 クライアント機能（TP1/Client/J） | CUP への一方通知機能 | | ○ | コンテナ内の CUP への一方通知機能は，TP1/Client/J だけをサポートしています。 |
| 21 | サービス関数動的ローディング機能 | | | ○ | − |
| 22 | OpenTP1 の運用を補助する機能 | 資源の排他制御 | | ○ | − |
| 23 | | ユーザジャーナルの取得 | | ○ | − |
| 24 | | ジャーナル維持機能 | | ○ | − |
| 25 | | メッセージログの操作 | | ○ | − |
| 26 | | メッセージログの通知 | | ○ | − |
| 27 | | 稼働統計情報 | | ○ | − |
| 28 | | リアルタイム統計情報サービス | | ○ | − |
| 29 | | OpenTP1 の状態確認機能 | | ○ | − |
| 30 | 監査ログによるシステムの監視 | | | × | − |
| 31 | OpenTP1 の監視 | | | ○ | − |
| 32 | OpenTP1 ファイルシステム | | | △ | 次のファイルシステムをサポートします。<br>• ステータスファイル<br>• システムジャーナルファイル<br>• チェックポイントダンプファイル<br>• TAM ファイル<br>• DAM ファイル<br>上記以外のファイルシステムはサポートしていません。 |
| 33 | 障害の原因解析（各種トレースファイルの取得） | | | ○ | − |
| 34 | 複数の OpenTP1 を使用する場合の機能 | 系切り替え機能（HA モニタを使用した系切り替え） | | × | − |
| 35 | | 系切り替え機能（Kubernetes の機能を使用した系切り替えおよび再開始） | | × | 取扱説明書の「5. TP1 サーバの系切り替え構成」 |

| # | 機能名 | | サポート | 特記事項 |
|---|---|---|---|---|
| 35 | 複数の OpenTP1 を使用する場合の機能 | 系切り替え機能<br>(Kubernetes の機能を使用した系切り替えおよび再開始) | × | に記載している機能はサポートしていません。 |
| 36 | | マルチノード機能 | × | − |
| 37 | | グローバルアーカイブジャーナル機能 | × | − |
| 38 | | マルチ OpenTP1 | × | − |
| 39 | TCP コネクション管理 | 一時クローズ機能 | ○ | − |
| 40 | | ノード間通信時の毎回コネクション断機能 | ○ | − |

(凡例)

○：サポート

△：制限付サポート

×：未サポート

−：該当しません

## 表 8-10　TP1/Client/J のサポート機能

| # | 機能名 | | | サポート |
|---|---|---|---|---|
| 1 | トランザクション制御 | 分散トランザクション | | ○ |
| 2 | | XA リソースサービスによるトランザクション制御 | | × |
| 3 | クライアント/サーバ形態の通信 | リモートプロシジャコール（RPC）通信 | リモート API 機能（RAP）を使用した RPC | ○ |
| 4 | | | ネームサービスを使用した RPC | ○<br>（TP1/Client/J 07-53 以降） |
| 5 | | | スケジューラダイレクト機能を使用した RPC | ○ |
| 6 | | スケジュール機能 | マルチスケジューラ機能 | ○ |
| 7 | | | スケジュールの優先度制御 | ○ |
| 8 | | データ圧縮機能 | | ○ |
| 9 | | ノード間負荷バランス機能 | | ○ |
| 10 | | TCP/IP 通信機能 | | × |
| 11 | | サーバからの一方通知受信機能 | | ○ |
| 12 | | TP1/Web 接続機能 | | ○ |
| 13 | | DCCM3 との接続機能 | | ○ |
| 14 | その他の機能 | 動的定義変更機能 | | ○ |
| 15 | | 障害の原因解析（各種トレースファイルの取得） | | ○ |

（凡例）

○：サポート

×：未サポート

## 8.4.2　RPC 通信

コンテナ内の TP1/Client/J，TP1/Server Base と RPC 通信する場合のサポート構成を次の表に示します。

## (1)　サポートする RPC 通信種別

次の表にサポートする RPC 通信種別を示します。

### 表 8-11　サポートする RPC 通信種別（TP1/Server Base-TP1/Server Base）

| # | RPC の種別 | TP1/Server Base の配置 | | サポート | 特記事項 |
|---|---|---|---|---|---|
| | | 送信元 TP1/Server Base | 送信先 TP1/Server Base | | |
| 1 | リモート API 機能（rap）を使用した RPC | コンテナ内 | コンテナ内 | ○ | － |
| 2 | | コンテナ外 | コンテナ内 | ○ | － |
| 3 | | コンテナ内 | コンテナ外 | ○ | － |
| 4 | ネームサービスを使用した RPC | コンテナ内 | コンテナ内 | ○ | － |
| 5 | | コンテナ外 | コンテナ内 | ○ | － |
| 6 | | コンテナ内 | コンテナ外 | ○ | － |
| 7 | スケジューラダイレクト機能を使用した RPC | コンテナ内 | コンテナ内 | ○ | 詳細は，「8.4.3　RPC 通信の注意事項」を参照してください。 |
| 8 | | コンテナ外 | コンテナ内 | ○ | |
| 9 | | コンテナ内 | コンテナ外 | ○ | － |

（凡例）

○：サポート

×：未サポート

－：該当しません

### 表 8-12　サポートする RPC 通信種別（TP1/Client/J-TP1/Server Base）

| # | RPC の種別 | TP1/Client/J，TP1/Server Base の配置 | | サポート | 特記事項 |
|---|---|---|---|---|---|
| | | 送信元 TP1/Client/J | 送信先 TP1/Server Base | | |
| 1 | リモート API 機能（rap）を使用した RPC | コンテナ内 | コンテナ内 | ○ | － |

| # | RPC の種別 | TP1/Client/J，TP1/Server Base の配置 | | サポート | 特記事項 |
|---|---|---|---|---|---|
| | | 送信元 TP1/Client/J | 送信先 TP1/Server Base | | |
| 2 | リモート API 機能（rap）を使用した RPC | コンテナ外 | コンテナ内 | ○ | － |
| 3 | | コンテナ内 | コンテナ外 | ○ | － |
| 4 | ネームサービスを使用した RPC | コンテナ内 | コンテナ内 | ○ | TP1/Client/J 07-53 以降 |
| 5 | | コンテナ外 | コンテナ内 | ○ | TP1/Client/J 07-53 以降 |
| 6 | | コンテナ内 | コンテナ外 | ○ | TP1/Client/J 07-53 以降 |
| 7 | スケジューラダイレクト機能を使用した RPC | コンテナ内 | コンテナ内 | ○ | － |
| 8 | | コンテナ外 | コンテナ内 | ○ | － |
| 9 | | コンテナ内 | コンテナ外 | ○ | － |

（凡例）

○：サポート

×：未サポート

－：該当しません

## 8.4.3　RPC 通信の注意事項

TP1/Server Base が次のシステム条件に該当する場合は，次に示す対応が必要です。

### (1)　システム条件（AND 条件）

条件 1.

TP1/Server Base が稼働するサーバマシンにホスト名（IP アドレス）が複数存在する。

条件 2.

TP1/Server Base が使用するホスト名（IP アドレス）のどれかが，TP1/Client/J が使用する IP アドレスと接続されていない（通信できない）構成である。

### (2)　対応方法

TP1/Client/J と TP1/Server Base 間で通信する IP アドレスは，1 つに固定してください。

TP1/Server Base のシステム定義で，システム共通定義の my_host オペランドと dcbindht 定義コマンドには，TP1/Client/J と接続可能なホスト名（IP アドレス）を指定してください。

## 8.4.4　コンテナ内の TP1/Server Base に関する注意事項

### (1)　システム定義に関する注意事項

コンテナ内では，システム共通定義の my_host オペランドは指定できません。

### (2)　namsvinf コマンドに関する注意事項

コンテナ内で namsvinf コマンドを実行する場合，次の点に注意してください。

#### (a)　システム共通定義の rpc_close_after_send に Y を指定する場合

namsvinf コマンドは，-b オプション以外のオプションを指定しないでください。

-b オプション以外を指定した場合，起動確認に失敗するためノードが起動中の場合でもノード稼働状況（STAT に表示される値）に'S'（送信可能）が表示されます。

#### (b)　システム共通定義の rpc_close_after_send に N を指定，または省略する場合

namsvinf コマンドは-b オプション以外のオプションも指定できます。

ただし，コネクション障害や一時クローズ処理によって，(a)と同様にノード稼働状況が'S'と表示される場合があります。その場合は，時間を置いて再度コマンドを実行してください。

## 8.5　システム構成と環境設定

コンテナ内の TP1/Server Base へ RPC する場合，通信先にはコンテナが動作するワーカーノード，またはマスターノードを設定します。

通信先とは，システム定義や TP1/Client/J 環境定義，および通信先を指定する API で設定するホスト名または IP アドレスと，ポート番号のことです。

通信先のポート番号については，マニュアル「OpenTP1 システム定義」またはマニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。

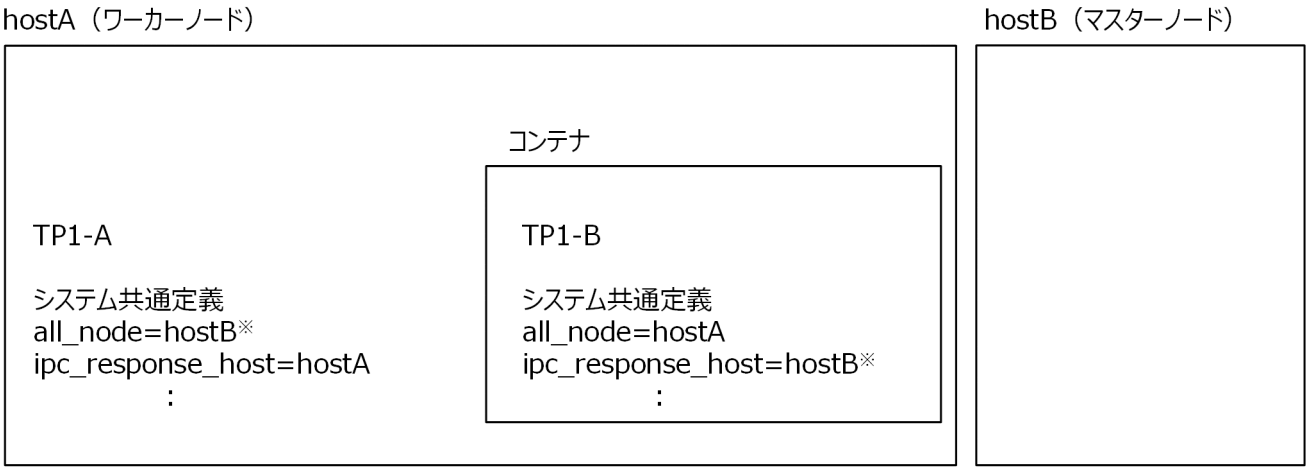
ただし，Kubernetes クラスタ内の同一ホスト上で，ホスト OS 上の TP1/Server Base と，コンテナ内の TP1/Server Base を連携する場合，コンテナが動作するワーカーノードを設定しても通信できません。その場合，次のとおりマスターノードを設定してください。

なお，マスターノードを設定したとしても，マスターノードのホスト OS 上の TP1/Server Base とコンテナ内の TP1/Server Base 間の通信はできません。

(例)

コンテナが動作するワーカーノードを hostA，マスターノードを hostB とします。

同一ホスト上（hostA）で TP1-A（コンテナ外）と TP1-B（コンテナ内）を連携する場合の設定




注※

TP1-A の all_node オペランド，および TP1-B の ipc_response_host オペランドにマスターノードを設定してください。

ほかに Kubernetes クラスタ内で連携するコンテナ内の TP1/Server Base がある場合は，上記 TP1-A，TP1-B の設定と同様にマスターノードを設定してください。

システム構成と RPC 種別によって，OpenTP1 の定義に設定する内容が異なります。RPC 種別ごとの定義設定について次に示します。

## 8.5.1　リモート API 機能を使用した RPC

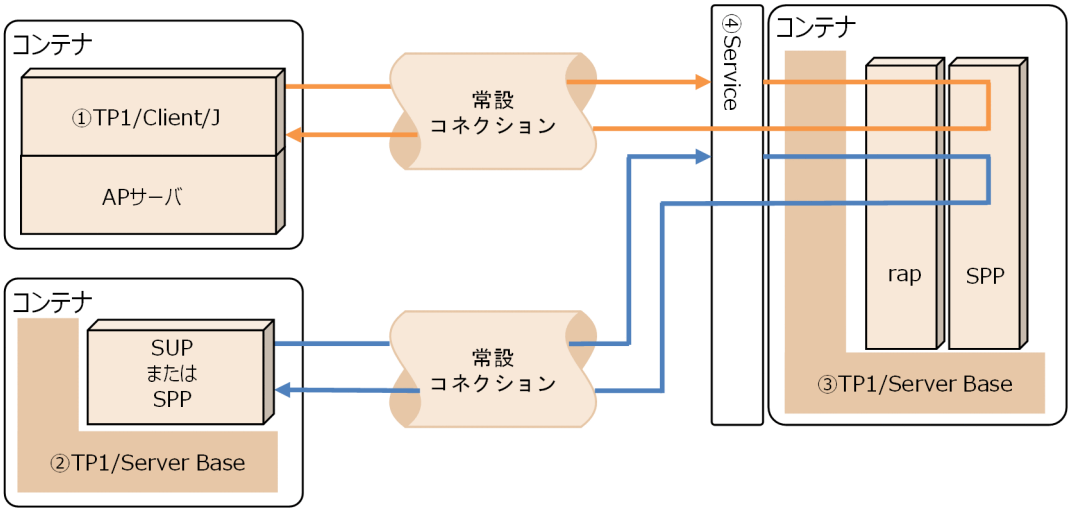
TP1/Client/J，TP1/Server Base からコンテナ内外の TP1/Server Base へリモート API 機能を使用した RPC ができます。システム形態に応じて TP1/Client/J，TP1/Server Base の定義設定，および Service の設定を行ってください。

### (1)　リモート API 機能を使用した RPC の構成例

コンテナ外の TP1/Server Base へ RPC を発行する場合は，マニュアル「OpenTP1 システム定義」またはマニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」に記載の定義設定方法で使用できます。

コンテナ内の TP1/Server Base へ RPC を発行する場合，RPC 先の TP1/Server Base に Service を作成する必要があります。

図 8-3　リモート API を使用した RPC の構成例




### (a)　TP1/Client/J 環境定義の設定【①TP1/Client/J】

次の TP1/Client/J 環境定義を設定してください。

表 8-13　TP1/Client/J 環境定義の設定

| # | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|
| 1 | dchost=ホスト名:ポート番号 | • ホスト名：rap リスナー用の Service が存在するホスト名<br>• ポート番号：④Service に設定した spec.ports.targetPort に対応する spec.ports.nodePort | コンテナ外の rap リスナーへ送信する場合は，特に制限はありません。<br>OpenTP1 マニュアルの説明に従い設定してください。 |

### (b)　システム定義の設定【②TP1/Server Base】

次のシステム定義を設定してください。

表 8–14　システム定義の設定【②TP1/Server Base】

| # | 定義名 | コマンド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | ユーザサービスネットワーク定義 | dcsvgdef -h ホスト名:ポート番号 -w | • ホスト名：rap リスナー用の Service が存在するホスト名<br>• ポート番号：④Service の spec.ports.targetPort に対応する spec.ports.nodePort | コンテナ外の rap リスナーへ送信する場合は，特に制限はありません。<br>OpenTP1 マニュアルの説明に従い設定してください。 |

### (c)　システム定義の設定【③TP1/Server Base】

次のシステム定義を設定してください。

表 8–15　システム定義の設定【③TP1/Server Base】

| # | 定義名 | コマンド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | rap リスナーサービス定義 | rap_listen_port | rap リスナーおよび rap サーバが使用するポート番号 | OpenTP1 マニュアルの説明に従い設定してください。 |

### (d)　図中④Service の設定

コンテナ内の③TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の③TP1/Server Base で rap_listen_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

TP1/Client/J や③TP1/Server Base からリモート API 機能を使用する場合は，Service の spec.ports.targetPort に対応する spec.ports.nodePort に対し接続してください。

## 8.5.2　ネームサービスを使用した RPC

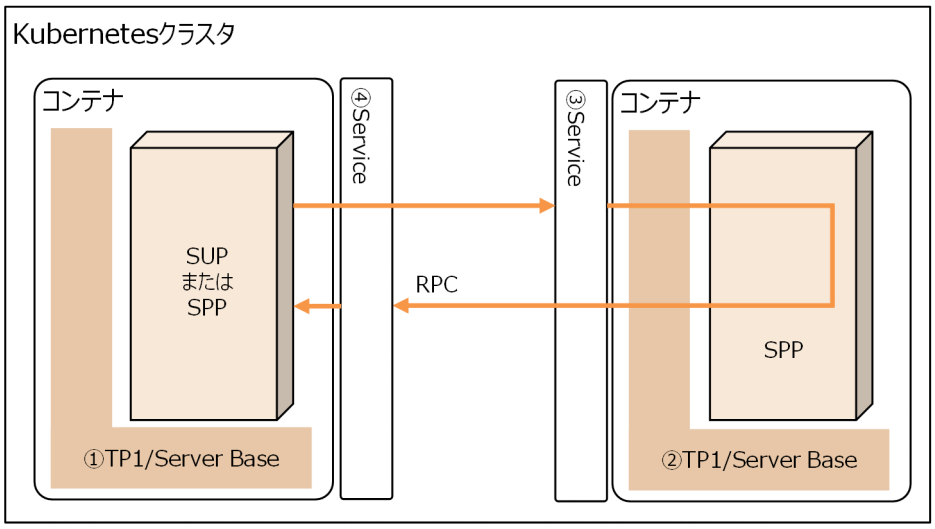
all_node オペランドに指定するホスト名には，Kubernetes のマスターノードまたは，ワーカーノードを指定できます。マスターノードを指定する場合は，OpenTP1 システム内のすべての TP1/Server Base の設定をマスターノードで統一してください。

ワーカーノードを指定する場合は，各 TP1/Server Base で Kubernetes クラスタ内の任意のワーカーノードを設定してください。

また，all_node オペランドに指定するホスト名は，ほかの TP1/Server Base に指定する ipc_response_host オペランドの値と一致させる必要があります。

## (1)　コンテナ内の TP1/Server Base 間の RPC

図 8-4　コンテナ内の TP1/Server Base 間の RPC




### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-16　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | rpc_port_base | 30000 以降のポート番号 | 「8.6.4　Service の作成」を参照し，ほかの Pod の使用範囲と重複しないポート番号を rpc_port_base に設定してください。 |
| 2 | | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 3 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用のホスト名 | |
| 4 | | all_node=ホスト名:ポート番号 | • ホスト名：ネームサービス用の Service が存在するホスト名<br>• ポート番号：③Service の spec.ports.targetPort に対応する spec.ports.nodePort | Kubernetes クラスタ内のどれかの Kubernetes ノードのホスト名を設定してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-16　システム定義の設定【①TP1/Server Base】」に示すオペランドを設定してください。all_node オペランドに指定するポート番号（相手ノードの name_port オペランドに指定したポート番号）は OpenTP1 システム内でユニークな値に設定する必要があります。

### (c)　図中③Service の設定

コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で name_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

### (d)　図中④Service の設定

コンテナ内の①TP1/Server Base で name_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

また，コンテナ内の TP1/Server Base からコンテナ内の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

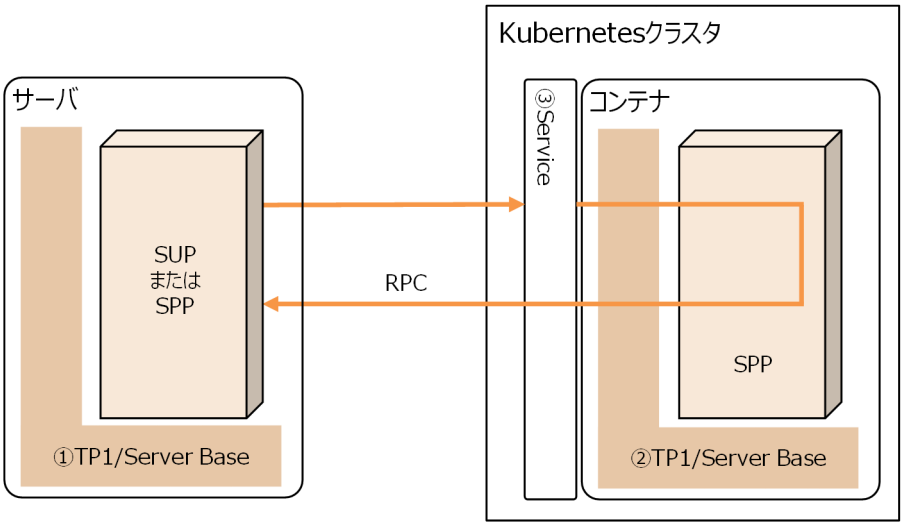
コンテナ内の①TP1/Server Base で，応答電文を受信するためのポート番号を Service の spec.ports.targetPort に設定してください。

応答電文の受信にはエフェメラルポートを使用するため，「8.6.4　Service の作成」で求めたエフェメラルポート数分のポートを Service に設定する必要があります。

Service へ設定が必要なポート番号の見積方法については，「8.6.4　Service の作成」を参照してください。

## (2)　TP1/Server Base とコンテナ内の TP1/Server Base 間の RPC

図 8-5　TP1/Server Base とコンテナ内の TP1/Server Base 間の RPC




### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-17　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 2 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用のホスト名 | |
| 3 | | all_node=ホスト名:ポート番号 | • ホスト名：ネームサービス用の Service が存在するホスト名<br>• ポート番号：③Service の spec.ports.targetPort に対応する spec.ports.nodePort | 図①TP1/Server Base：<br>Kubernetes クラスタ内のどれかの Kubernetes ノードのホスト名を設定してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。<br>なお，指定するホスト名には①TP1/Server Base が起動するホストのホスト名を指定しないでください。指定した場合 dc_rpc_call 関数が失敗します。<br>図②TP1/Server Base：<br>①TP1/Server Base が起動するホストのホスト名を設定してください。<br>また，ホスト名は①TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-17　システム定義の設定【①TP1/Server Base】」に示すオペランドを設定してください。all_node オペランドに指定するポート番号（相手ノードの name_port オペランドに指定したポート番号）は OpenTP1 システム内でユニークな値に設定する必要があります。

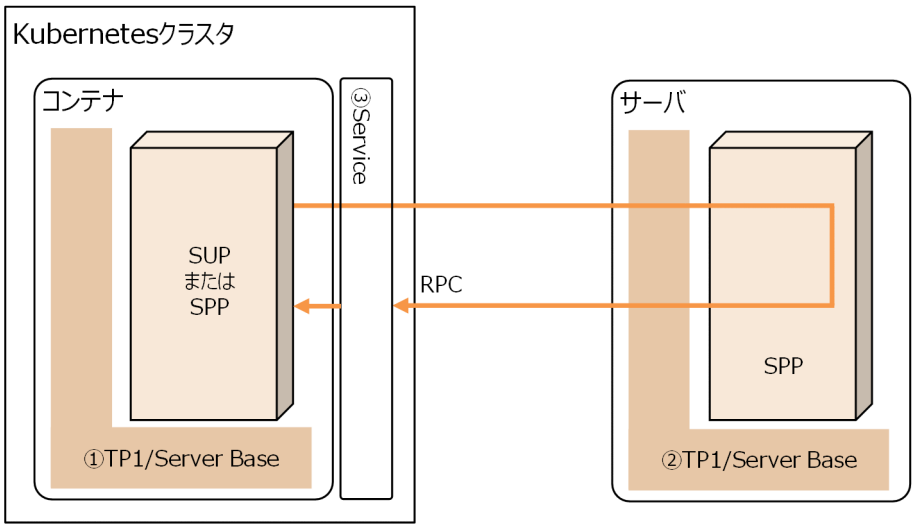
### (c)　図中③Service の設定

コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で name_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

## (3)　コンテナ内の TP1/Server Base と TP1/Server Base 間の RPC

図 8-6　コンテナ内の TP1/Server Base と TP1/Server Base 間の RPC




### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-18　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | rpc_port_base | 30000 以降のポート番号 | 「8.6.4　Service の作成」を参照し，ほかの Pod の使用範囲と重複しないポート番号を rpc_port_base に設定してください。 |
| 2 | | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 3 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用のホスト名 | |
| 4 | | all_node=ホスト名:ポート番号 | 図①TP1/Server Base：<br>• ホスト名：②TP1/Server Base が起動するホストのホスト名<br>• ポート番号：②TP1/Server Base の name_port<br>図②TP1/Server Base：<br>• ホスト名：ネームサービス用の Service が存在するホスト名<br>• ポート番号：③Service の pec.ports.targetPort に対応する spec.ports.nodePort | 図②TP1/Server Base で指定するホスト名は Kubernetes クラスタ内のどれかの Kubernetes ノードのホスト名を設定してください。<br>また，ホスト名は①TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-18　システム定義の設定【①TP1/Server Base】」に示すオペランドを設定してください。all_node オペランドに指定するポート番号（相手ノードの name_port オペランドに指定したポート番号）は OpenTP1 システム内でユニークな値に設定する必要があります。

### (c)　図中③Service の設定

コンテナ内の①TP1/Server Base で name_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

また，コンテナ内の TP1/Server Base からコンテナ外の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

コンテナ内の①TP1/Server Base で，応答電文を受信するためのポート番号を Service の spec.ports.targetPort に設定してください。

応答電文の受信にはエフェメラルポートを使用するため，「8.6.4　Service の作成」で求めた数のエフェメラルポートを Service に設定する必要があります。

Service へ設定が必要なポート番号の見積方法については，「8.6.4　Service の作成」を参照してください。

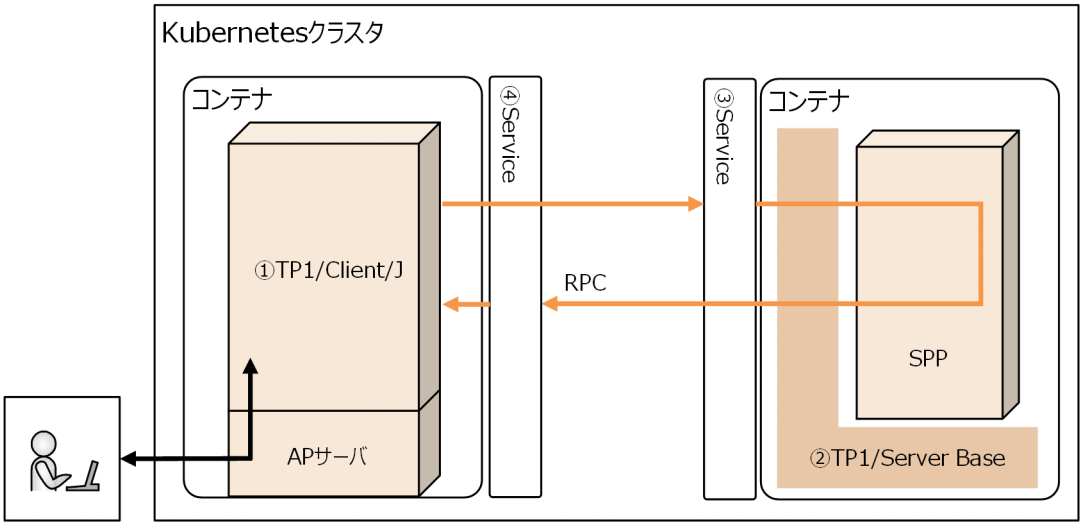
## 8.5.3　スケジューラダイレクト機能を使用した RPC

TP1/Client/J，TP1/Server Base からコンテナ内外の TP1/Server Base へスケジューラダイレクト機能を使用した RPC ができます。システム形態に応じて TP1/Client/J，TP1/Server Base の定義設定，および Service の設定を行ってください。

また，dchost オペランドや dcsvgdef コマンドに指定する通信先のホスト名は，通信先の TP1/Server Base に指定した ipc_response_host オペランドの値と一致させる必要があります。

## (1)　コンテナ内の TP1/Client/J とコンテナ内の TP1/Server Base 間の RPC

図 8-7　コンテナ内の TP1/Client/J とコンテナ内の TP1/Server Base 間の RPC




### (a)　TP1/Client/J 環境定義の設定【①TP1/Client/J】

TP1/Client/J インスタンスごとに，次の TP1/Client/J 環境定義を設定してください。

表 8-19　TP1/Client/J 環境定義の設定

| # | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|
| 1 | dccltcuprcvport | CUP の電文受信で使用するポート番号 | この構成では必ず dccltcuprcvport オペランドを設定してください。 |
| 2 | dchost=ホスト名:ポート番号 | rpcCall メソッドを発行する場合<br>• ホスト名：スケジュールサービス用の Service が存在するホスト名<br>• ポート番号：③Service の spec.ports.targetPort に対応する spec.ports.nodePort<br>rpcCallTo メソッドを発行する場合，このオペランドの設定は不要です。 | rpcCallTo メソッドを使用する場合は，左記のホスト名を設定した DCRpcBindTbl オブジェクトを作成してから，rpcCallTo メソッドを発行してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |
| 3 | dcnotifyreshost | Y | 詳細は，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。 |
| 4 | dcresponsehost | TP1/Server Base に通知する，応答電文受信用のホスト名 | |

### (b)　システム定義の設定【②TP1/Server Base】

次のシステム定義を設定してください。

表 8-20　システム定義の設定【②TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | スケジュールサービス定義 | scd_port | スケジュールサービスのポート番号 | － |
| 2 | システム共通定義 | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 3 | | ipc_response_host | TP1/Client/J に通知する，②TP1/Server Base の電文受信用のホスト名 | |

（凡例）

－：該当しません

### (c)　図中③Service の設定

コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

TP1/Client/J からスケジューラダイレクト機能を使用する場合は，Service の spec.ports.nodePort に対して接続してください。

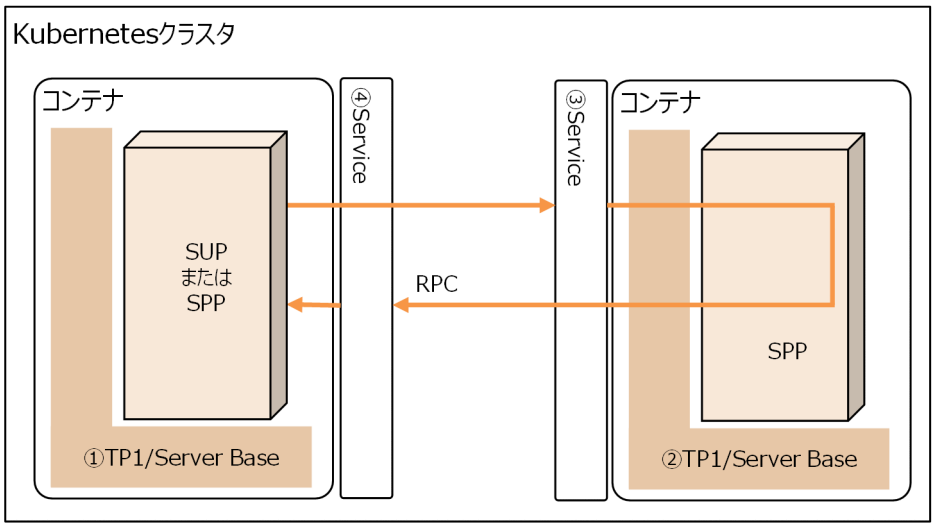
### (d)　図中④Service の設定

コンテナ内の TP1/Client/J からコンテナ内の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

コンテナ内の①TP1/Client/J で，dccltcuprcvport オペランドに指定したポート番号を，Service の spec.ports.targetPort および spec.ports.nodePort に設定してください。

## (2)　コンテナ内の TP1/Server Base 間の RPC

図 8-8　コンテナ内の TP1/Server Base 間の RPC



### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-21　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名／コマンド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | rpc_port_base | 30000 以降のポート番号 | 「8.6.4　Service の作成」を参照し，ほかの Pod の使用範囲と重複しないポート番号を rpc_port_base に設定してください。 |
| 2 | | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 3 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用のホスト名 | |
| 4 | ユーザサービスネットワーク定義 | dcsvgdef -h ホスト名-p ポート番号 | dc_rpc_call 関数を発行する場合<br>• ホスト名：スケジュールサービス用の Service が存在するホスト名<br>• ポート番号：<br>③Service の spec.ports.targetPort に対応する spec.ports.nodePort<br>dc_rpc_call_to 関数を発行する場合，このオペランドの設定は不要です。 | dc_rpc_call_to 関数を使用する場合は，DCRPC_DIRECT_SCHEDULE 関数で左記ホスト名を設定し，DCRPC_BINDING_TBL 構造体を作成してから dc_rpc_call_to 関数を発行してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-20　システム定義の設定【②TP1/Server Base】」に示すオペランドを設定してください。

### (c)　図中③Service の設定

コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

TP1/Server Base からスケジューラダイレクト機能を使用する場合は，Service の spec.ports.targetPort に対応する spec.ports.nodePort に対して接続してください。

### (d)　図中④Service の設定

コンテナ内の①TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

また．コンテナ内の TP1/Server Base からコンテナ内の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

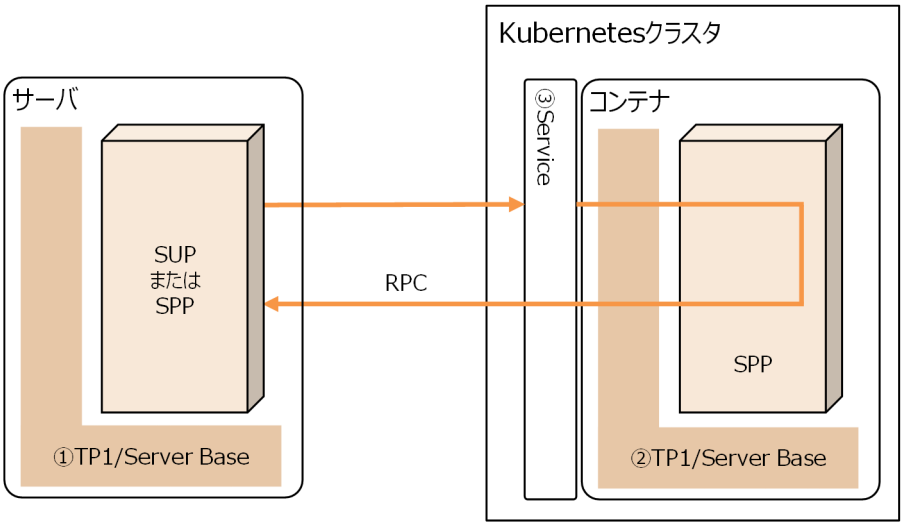
コンテナ内の①TP1/Server Base で，応答電文を受信するためのポート番号を Service の spec.ports.targetPort に設定してください。

応答電文の受信にはエフェメラルポートを使用するため，「8.6.4　Service の作成」で求めたエフェメラルポート数分のポートを Service に設定する必要があります。

Service へ設定が必要なポート番号の見積方法については，「8.6.4　Service の作成」を参照してください。

## (3)　TP1/Server Base とコンテナ内の TP1/Server Base 間の RPC

図 8-9　TP1/Server Base とコンテナ内の TP1/Server Base 間の RPC




### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-22　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 2 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用の Service が存在するホスト名 | |
| 3 | ユーザサービスネットワーク定義 | dcsvgdef -h ホスト名-p ポート番号 | dc_rpc_call 関数を発行する場合<br>• ホスト名：スケジュールサービス用の Service が存在するホスト名<br>• ポート番号： | dc_rpc_call_to 関数を使用する場合は，DCRPC_DIRECT_SCHEDULE 関数で左記ホスト名を設定し，DCRPC_BINDING_TBL 構造体を |

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 3 | ユーザサービスネットワーク定義 | dcsvgdef -h ホスト名-p ポート番号 | ③Service の spec.ports.targetPort に対応する spec.ports.nodePort<br>dc_rpc_call_to 関数を発行する場合，このオペランドの設定は不要です。 | 作成してから dc_rpc_call_to 関数を発行してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。<br>なお，指定するホスト名には①TP1/Server Base が起動するホストのホスト名を指定しないでください。指定した場合 dc_rpc_call 関数が失敗します。 |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-20　システム定義の設定【②TP1/Server Base】」に示すオペランドを設定してください。

### (c)　図中③Service の設定

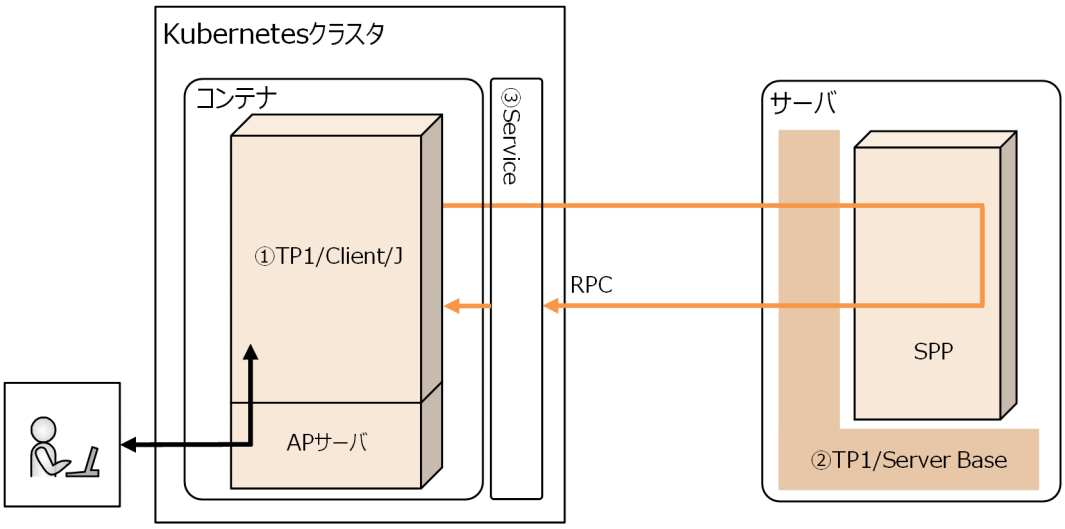
コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

TP1/Server Base からスケジューラダイレクト機能を使用する場合は，Service の spec.ports.targetPort に対応する spec.ports.nodePort に対して接続してください。

## (4)　コンテナ内の TP1/Client/J と TP1/Server Base 間の RPC

図 8-10　コンテナ内の TP1/Client/J と TP1/Server Base 間の RPC




### (a)　TP1/Client/J 環境定義の設定【①TP1/Client/J】

次の TP1/Client/J 環境定義を設定してください。

表 8-23　TP1/Client/J 環境定義の設定

| # | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|
| 1 | dccltcuprcvport | CUP の電文受信で使用するポート番号 | この構成では必ず dccltcuprcvport オペランドを設定してください。 |
| 2 | dcnotifyreshost | Y | 詳細は，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。 |
| 3 | dcresponsehost | TP1/Server Base に通知する，応答電文受信用の Service が存在するホスト名 | |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-20　システム定義の設定【②TP1/Server Base】」に示すオペランドを設定してください。

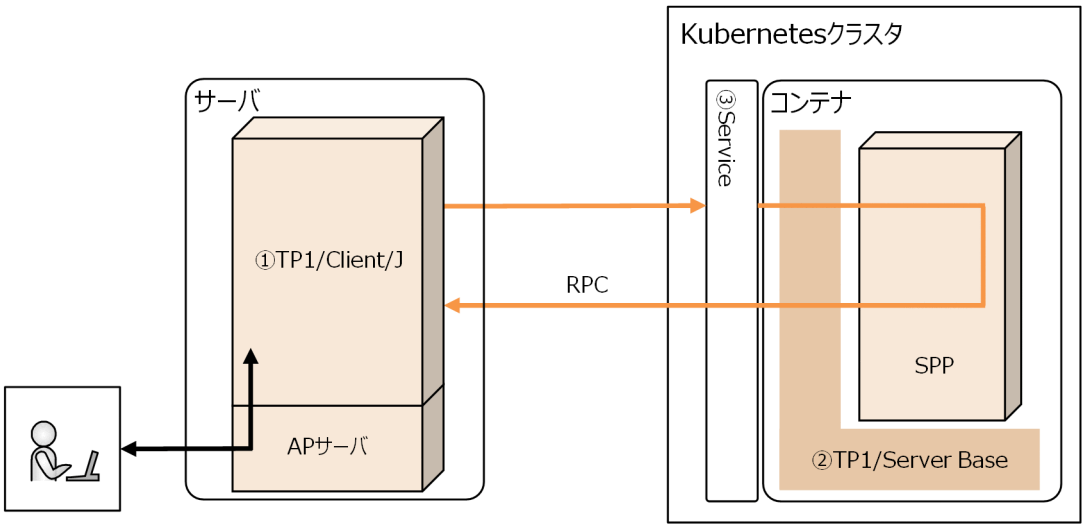
### (c)　Service の設定

コンテナ内の TP1/Client/J からコンテナ外の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

コンテナ内の①TP1/Client/J で，dccltcuprcvport オペランドに指定したポート番号を，Service の spec.ports.targetPort および spec.ports.nodePort に設定してください。

## (5)　TP1/Client/J とコンテナ内の TP1/Server Base 間の RPC

図 8-11　TP1/Client/J と TP1/Server Base 間の RPC




### (a)　TP1/Client/J 環境定義の設定【TP1/Client/J】

次の TP1/Client/J 環境定義を設定してください。

表 8-24　TP1/Client/J 環境定義の設定

| # | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|
| 1 | dchost=ホスト名:ポート番号 | rpcCall メソッドを発行する場合<br>• ホスト名：スケジュールサービス用の Service が存在するホスト名<br>• ポート番号：③Service の spec.ports.targetPort に対応する spec.ports.nodePort<br>rpcCallTo メソッドを発行する場合，このオペランドの設定は不要です。 | 通信先指定 RPC を使用する場合は，左記のホスト名を設定した DCRpcBindTbl オブジェクトを作成してから，rpcCallTo メソッドを発行してください。<br>また，ホスト名は②TP1/Server Base の ipc_response_host オペランドの値と一致させてください。 |
| 2 | dcnotifyreshost | Y | 詳細は，マニュアル「OpenTP1 クライアント使用の手引 TP1/Client/J 編」を参照してください。 |
| 3 | dcresponsehost | TP1/Server Base に通知する，応答電文受信用の Service が存在するホスト名 | |

## (b)　システム定義の設定【②TP1/Server Base】

「表 8-20　システム定義の設定【②TP1/Server Base】」に示すオペランドを設定し，スケジュールサービス用の Service を作成してください。

## (c)　図中③Service の設定

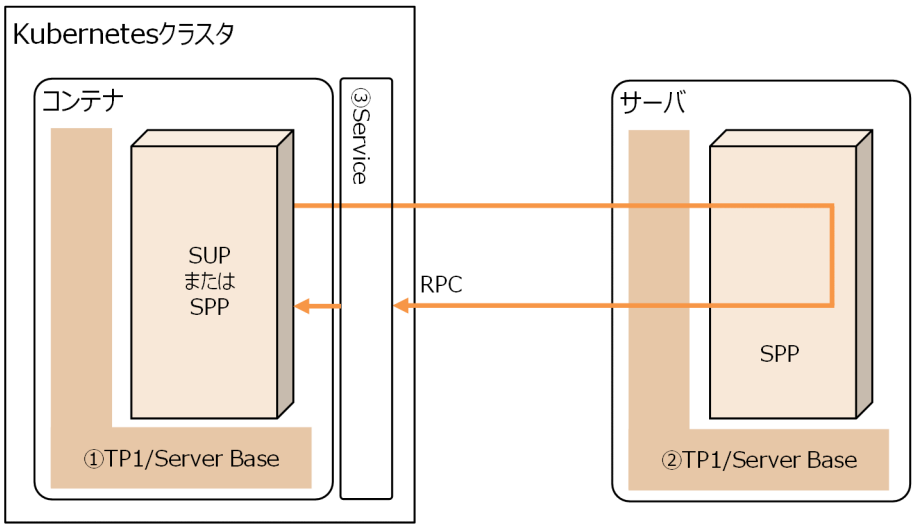
コンテナ内の TP1/Server Base へ RPC する場合は，Service を作成してください。

コンテナ内の②TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

TP1/Client/J からスケジューラダイレクト機能を使用する場合は，Service の spec.ports.targetPort に対応する spec.ports.nodePort に対して接続してください。

## (6)　コンテナ内の TP1/Server Base と TP1/Server Base 間の RPC

図 8-12　コンテナ内の TP1/Server Base と TP1/Server Base 間の RPC




### (a)　システム定義の設定【①TP1/Server Base】

次のシステム定義を設定してください。

表 8-25　システム定義の設定【①TP1/Server Base】

| # | 定義名 | オペランド名 | 設定内容 | 補足 |
|---|---|---|---|---|
| 1 | システム共通定義 | rpc_port_base | 30000 以降のポート番号 | 「8.6.4　Service の作成」を参照し，ほかの Pod の使用範囲と重複しないポート番号を rpc_port_base に設定してください。 |
| 2 | | ipc_notify_response_host | Y | 詳細は，マニュアル「OpenTP1 システム定義」を参照してください。 |
| 3 | | ipc_response_host | 通信先ノードの TP1/Server Base に通知する，応答電文受信用の Service が存在するホスト名 | |

### (b)　システム定義の設定【②TP1/Server Base】

「表 8-20　システム定義の設定【②TP1/Server Base】」に示すオペランドを設定してください。

### (c)　図中③Service の設定

コンテナ内の①TP1/Server Base で scd_port オペランドに設定したポート番号と同じポート番号を Service の spec.ports.targetPort に設定してください。

また，コンテナ内の TP1/Server Base からコンテナ外の TP1/Server Base に RPC する場合は，応答電文を受信するための Service を作成してください。

コンテナ内の①TP1/Server Base で，応答電文を受信するためのポート番号を Service の spec.ports.targetPort に設定してください。

応答電文の受信にはエフェメラルポートを使用するため，「8.6.4　Service の作成」で求めたエフェメラルポート数分のポートを Service に設定する必要があります。

Service へ設定が必要なポート番号の見積方法については，「8.6.4　Service の作成」を参照してください。

## 8.6　コンテナ環境独自の設定

OpenTP1 の提供機能の中で Kubernetes 環境固有の設定が必要となる機能について説明します。

### 8.6.1　OpenTP1 の開始と終了

#### (1)　TP1/Server Base

コンテナ内の TP1/Server Base は，Pod の起動と連動して開始・終了するように Pod を作成してください。

TP1/Server Base の開始については，コンテナ環境独自の「prcd 直接起動方式」を適用します。Pod のマニフェストの作成方法，TP1/Server Base の停止方法，および「prcd 直接起動方式」については，取扱説明書を参照してください。

#### (2)　TP1/Client/J

コンテナ内の TP1/Client/J は，Pod の起動と連動して Java アプリケーションを開始・終了するように Pod を作成してください。

Pod の作成方法については，取扱説明書を参照してください。

サーブレット形態で使用する場合は，アプリケーションサーバの説明書を参照してください。

### 8.6.2　TP1/Server Base の再開始

TP1/Server Base の再開始を行う場合は，Kubernetes の nodeSelector 機能などを使用し，再開始後も同じノードに Pod が起動するように設定してください。Kubernetes のコンテナ内の TP1/Server Base が使用する OpenTP1 ファイルシステムを，Kubernetes の永続ボリューム機能を使用しホスト OS と共有することで，OpenTP1 の再開始が可能になります。

永続ボリュームの設定については，取扱説明書を参照してください。

### 8.6.3　TP1/Server Base のノード追加

Kubernetes のスケール制御を使用した TP1/Server Base のノード追加はできません。

新たに TP1/Server Base のノードを追加する場合は，別の Pod を作成してください。

ネームサービス機能を使用した場合のノード追加手順の概要を次に示します。

追加するノードの設定方法

TP1/Server Base のノードを追加する場合，追加するノードでは，既存 TP1/Server Base ノードの指定値と一致しないユニークな値を設定する必要があります。

表 8-26　ユニークな値を設定する項目

| # | 項目 | 定義オペランド（システム共通定義） |
|---|---|---|
| 1 | OpenTP1 識別子 | system_id |
| 2 | ノード識別子 | node_id |
| 3 | ネームサービスのポート番号 | name_port |

上記に従い，新たに作成する Pod で OpenTP1 システム定義を作成してください。

次に，追加するノードの all_node またはドメイン定義ファイルに，既存の TP1/Server Base ノードのホスト名とポート番号をすべて記載後，必要に応じ Pod 作成前にノード固有の定義を設定してください。

すべての定義設定が完了後，Pod を作成します。

既存ノードの設定方法

既存ノードの Pod を停止し，all_node またはドメイン定義ファイルに，追加する TP1/Server Base ノードのホスト名とポート番号を設定し，Pod を再作成します。

定義の作成は ConfigMap でもできます。ConfigMap を使用した定義ファイルの作成方法や Pod の作成および停止方法については，取扱説明書を参照してください。

ネームサービス機能を使用するシステムで，ConfigMap を使用してノードを追加する手順を次に示します。

## (1)　ネームサービス機能を使用するシステム構成でのノード追加手順

次の構成例を基に，ノードを追加する手順を示します。

## (a)　ノード追加前のシステム構成

図 8-13　ノード追加前の OpenTP1 システム構成

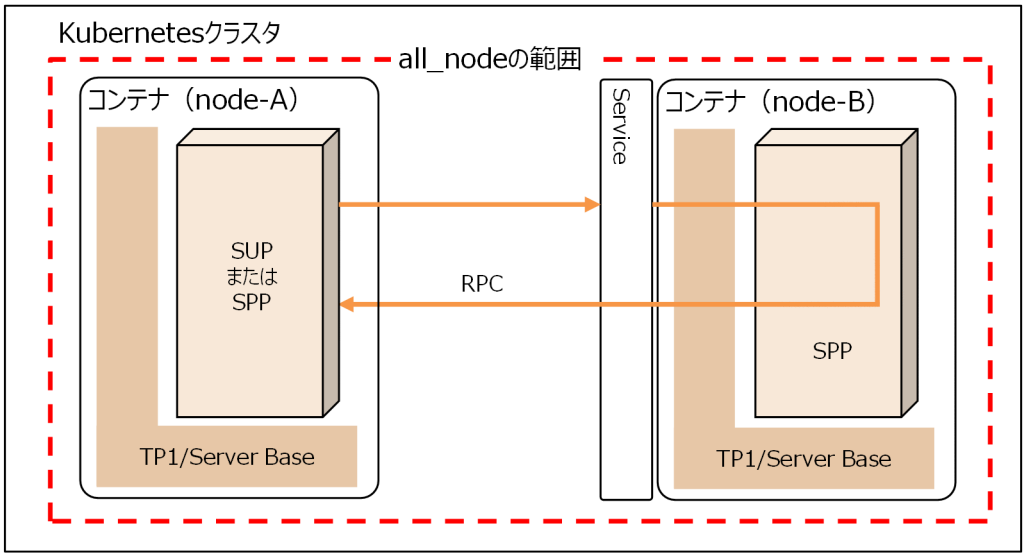
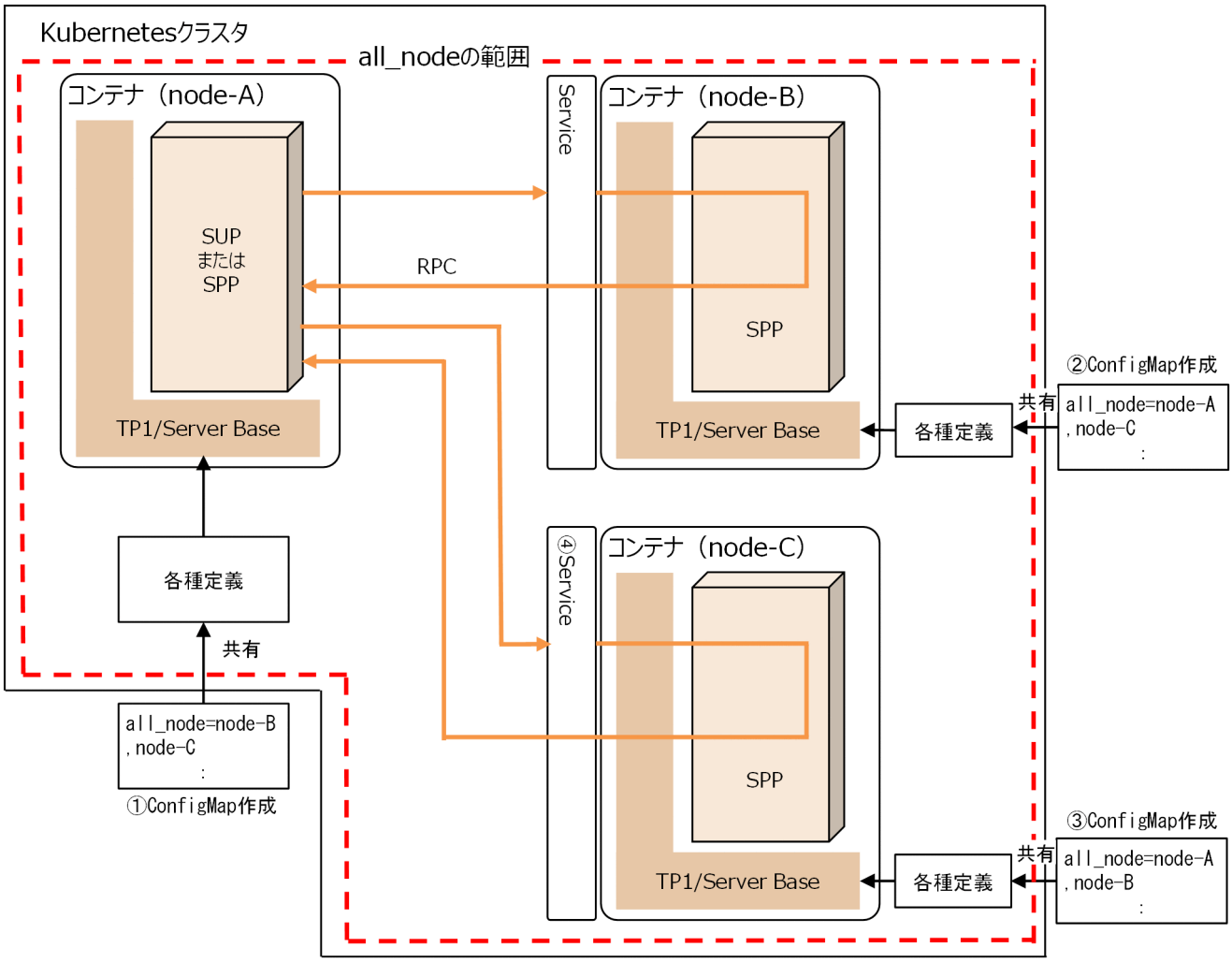



図 8-14　ノード追加後の OpenTP1 システム構成



### (b)　ノード追加手順

既存ノードで行う作業

1. 既存ノード（node-A，node-B）の Pod を停止する。
2. システム共通定義の all_node オペランドに，追加する TP1/Server Base のホスト名または IP アドレス（図 node-C）を設定する。
3. ConfigMap（図①ConfigMap，②ConfigMap）を作成する。
4. 既存ノード（node-A，node-B）の Pod を起動する。作成した ConfigMap の名称は，起動する Pod の spec.volumes.configMap.name に指定してください。

追加するノードで行う作業

1. 追加するノードのシステム共通定義を作成する。

   ・ノード識別子，OpenTP1 識別子，ネームサービスのポート番号に，既存ノードとは別の OpenTP1 システム内でユニークな値を設定する。

   ・all_node オペランドに，既存ノード（node-A，node-B）を設定する。
2. ConfigMap（図③ConfigMap）を作成する。
3. ネームサービスのポート番号と同じポート番号を Service の spec.ports.targetPort に設定し，Service（図④Service）を作成する。
4. 追加するノード（node-C）の Pod を起動する。作成した ConfigMap の名称は，起動する Pod の spec.volumes.configMap.name に指定してください。

## (2)　ネームサービス機能を使用するその他のシステム構成

### (a)　all_node_ex 定義を使用する場合

all_node_ex 定義に宛先ノードを追加する場合は，「8.6.3(1)　ネームサービス機能を使用するシステム構成でのノード追加手順」と同様の方法でホスト名を追加してください。

### (b)　ドメイン定義ファイルを使用する場合

ドメイン定義ファイルを使用する場合は，all_node のドメイン定義ファイル，および all_node_ex のドメイン定義ファイルに追加を行い，「8.6.3(1)　ネームサービス機能を使用するシステム構成でのノード追加手順」と同様の方法でホスト名を追加してください。

# 8.6.4　Service の作成

コンテナ内の OpenTP1 と通信する場合は，Pod 間の RPC 通信であっても Service を作成する必要があります。RPC 送信元は，Service が受付可能な IP アドレス，ポート番号を設定してコネクションを確立します。

ほかの OpenTP1 からのコネクション接続を受け付けるポート番号に応じて，Service を作成してください。

Service の作成方法の詳細は，取扱説明書を参照してください。

## (1)　必要なポート数の計算

TP1/Server Base が使用するポートには，サービス応答で使用するエフェメラルポート（短命ポート）と，スケジューラサービスなどのシステムプロセスが使用するユーザ指定ポートがあります。

サービス応答で使用するエフェメラルポートについては，次に示す計算式で求める必要があります。

### (a)　ネームサービスおよびスケジューラダイレクトを使用した RPC

コンテナ内の TP1/Server Base がサービス応答を受信するためのポート番号には，エフェメラルポートを使用します。

次の計算式で，コンテナ内の TP1/Server Base に必要なポート数を算出してください。

なお，非応答型の RPC を発行する場合は，エフェメラルポート数の算出と Service の作成は不要です。

必要なポート数（サービス応答受信用のポート数）
＝（プロセスサービス定義のprc_process_countオペランドに設定した値 ＋ 128）×1.5※

注※

安全係数。1.5～2.0 を推奨します。システムに応じて調整してください。

SUP や，非常駐プロセスなど，サービス応答受信用ポート数は，高負荷などの条件によって必要なポート数が計算式よりも多くなる場合があります。Service 作成後，十分なテストを実施してください。

TP1/Server Base が使用するサービス応答受信用ポート数が，Service から公開するポート数を上回った場合，該当するサービス応答でクライアント，サーバ双方にエラーとなることがあります。

## (2)　RPC で使用するポートのマッピング

Service を作成し，OpenTP1 が使用するエフェメラルポートやユーザ指定ポートと Service から公開するポートをマッピングしてください。

### (a)　ネームサービスおよびスケジューラダイレクトを使用した RPC（コンテナ内の TP1/Server Base から TP1/Server Base への RPC）

「8.6.4(1)　必要なポート数の計算」で求めた必要なポート数分のポート名を Service マニフェストに作成し，Service の受信ポート番号とコンテナから公開する TP1/Server Base のポート番号をマッピングしてください。ポート番号を設定する Service マニフェストのフィールドについては，「8.5　システム構成と環境設定」中の Service の説明を参照してください。

システム共通定義に rpc_port_base オペランドを設定している場合，Service の受信ポート番号の設定値は，rpc_port_base オペランド設定値を基準に，必要な数のポート番号をマッピングしてください。

### (b) スケジューラダイレクトを使用した RPC（コンテナ内の TP1/Server Base への RPC）

スケジュールサービス定義の scd_port オペランド，scdmulti 定義コマンドの-p オプションに設定したポート数分の Service を作成してください。

### (c) リモート API 機能を使用した RPC（コンテナ内の TP1/Server Base への RPC）

rap リスナーサービス定義の rap_listen_port オペランドに設定するポート数分の Service を作成してください。

### (d) TP1/Client/J の RPC 応答

コンテナ内の TP1/Client/J がサービス応答を受信する場合，TP1/Client/J 環境定義の dccltcuprcvport オペランドに設定したポート数分の Service を作成してください。

#### （例）rpc_port_base オペランドの設定値が 30000，必要なポート数が 200 個の場合

<service-sample1.yaml>

```
apiVersion: v1
kind: Service
metadata:
  name: service-sample1
 :〈略〉
 spec:
  ports:
  - name: ”tp1-port001”・・ポート名
     protocol: TCP”
     port: 30000
     nodePort: 30000
     targetPort: 30000
  - name: ”tp1-port002”・・ポート名
     protocol: TCP”
     port: 30001
     nodePort: 30001
     targetPort: 30001
      :
    :〈中略〉
  - name: ”tp1-port099”・・ポート名
     protocol: TCP”
     port: 30098
      nodePort: 30098
     targetPort: 30098
  - name: ”tp1-port100”・・ポート名
     protocol: TCP”
     port: 30099
     nodePort: 30099
     targetPort: 30099
  - name: ”tp1-port101”・・ポート名
     protocol: TCP”
     port: 30100
     nodePort: 30100
     targetPort: 30100
  - name: ”tp1-port102”・・ポート名
     protocol: TCP”
```

```
    port: 30101
    nodePort: 30101
    targetPort: 30101
  - name: "tp1-port103"・・ポート名
    protocol: TCP"
    port: 30102
    nodePort: 30102
    targetPort: 30102
     :
   :〈中略〉
  - name: "tp1-port199"・・ポート名
    protocol: TCP"
    port: 30198
    nodePort: 30198
    targetPort: 30198
  - name: "tp1-port200"・・ポート名
    protocol: TCP"
    port: 30199
    nodePort: 30199
    targetPort: 30199
```

注

- spec.ports.nodePort，spec.ports.port：Service の受信ポート番号
- spec.ports.targetPort：OpenTP1 が使用するエフェメラルポートやユーザ指定ポート

注意事項

Kubernetes クラスタ内で TP1/Server Base が起動する Pod を複数起動する場合は，Pod ごとに Service の作成が必要になります。Service マニフェストに設定するポート番号が，ほかの Pod で作成する Service マニフェストのポート番号と重複する場合は，Service の作成に失敗するため，重複しないように調整してください。

### （例）dccltcuprcvport オペランドの設定値が 30000 の場合

<service-sample3.yaml>

```
apiVersion: v1
kind: Service
metadata:
  name: service-sample3
 :〈略〉
 spec:
  ports:
  - name: "tp1clt-port"・・ポート名
    protocol: TCP"
    port: 30000
    nodePort: 30000
    targetPort: 30000
```

注

- spec.ports.nodePort，spec.ports.port：Service の受信ポート番号
- spec.ports.targetPort：TP1/Client/J 環境定義の dccltcuprcvport オペランドに設定したポート番号

## 8.7 障害対応

OpenTP1 システムプロセスがダウンした場合，OpenTP1 が動作していた Pod も停止します。その後，Kubernetes が Pod の停止を検出し，Kubernetes クラスタ内で Pod を再起動します。

Pod 内で作成・更新されたファイルは Pod の停止と共に消失します。そのため，OpenTP1 が出力するトラブルシュート情報（トレース，コアファイルなど）は，ホスト OS 上のディレクトリに格納することを推奨します。ホスト OS 上で準備するディレクトリについては，取扱説明書を参照してください。

## 8.8　制限事項

制限事項について説明します。

### 8.8.1　TP1/Server Base の制限事項

HA モニタおよび Kubernetes の機能を使用した系切り替え構成はサポートしていません。

制限事項については，「表 8-9　TP1/Server Base のサポート機能」および「表 8-11　サポートする RPC 通信種別（TP1/Server Base-TP1/Server Base）」を参照してください。

### 8.8.2　TP1/Client/J の制限事項

#### (1)　TP1/Client/J の機能

TP1/Client/J の機能の制限事項については「表 8-10　TP1/Client/J のサポート機能」および「表 8-12　サポートする RPC 通信種別（TP1/Client/J-TP1/Server Base）」を参照してください。

#### (2)　Java プログラムの種類

Java アプレットは非対応です。

Java アプリケーションおよび Java サーブレットにだけ対応しています。

# 付録

# 付録 A　TP1/Message Control で使える通信プロトコル対応製品

OpenTP1 で使える通信プロトコルに対応した製品について説明します。これらの製品を使う場合は，TP1/Message Control と TP1/NET/Library が前提となります。

## 付録 A.1　通信プロトコル対応の製品

OpenTP1 の通信プロトコル対応の製品は次のとおりです。

### (1)　IP ネットワーク対応の製品

- TP1/NET/TCP/IP
  TCP/IP プロトコルで接続されたシステム間で，メッセージの通信をする場合に使う製品です。
- TP1/NET/UDP
  UDP/IP プロトコルで接続されたシステム間で，メッセージの通信をする場合に使う製品です。

### (2)　OSI 基本参照モデルのネットワーク対応の製品

- TP1/NET/OSI-TP
  OSI TP プロトコルに準拠した通信をする場合に使う製品です。
- TP1/NET/OSAS-NIF
  NIF/OSI プロトコルに準拠した通信をする場合に使う製品です。
- TP1/NET/User Agent
  OSAS/UA プロトコルに準拠した通信をする場合に使う製品です。

### (3)　GUI 対応の製品

- TP1/NET/XMAP3
  XMAP3 を使ったメッセージ送受信をする場合に使う製品です。

### (4)　SNA ネットワーク対応の製品

- TP1/NET/SLU - TypeP2
  SNA の SLU-TypeP プロトコル（二次局）に準拠した通信をする場合に使う製品です。

## 付録 A.2　通信プロトコル対応製品と接続できるシステム

OpenTP1 の通信プロトコル対応製品と接続できるシステムの関係を次の表に示します。

## 表 A–1　OpenTP1 の通信プロトコル対応製品の詳細

| プロトコル製品 | 使うプロトコル | 主な通信相手プログラム | 通信に使う回線 |
|---|---|---|---|
| TP1/NET/OSI-TP | OSI TP プロトコル | TP1/NET/OSI-TP<br>VOS3 XDM/DCCM3（OSAS/TP/DCCM3） | LAN |
| TP1/NET/OSAS-NIF | NIF/OSI プロトコル | TP1/NET/OSAS-NIF<br>VOS3 TMS-4V/SP（OSAS/NF/4VSP）<br>VOS3 XDM/DCCM3（OSAS/UA2/DCCM3） | LAN |
| TP1/NET/User Agent | OSAS/UA プロトコル | VOS3 TMS-4V/SP（OSAS/UA/4VSP）<br>VOS3 XDM/DCCM3（OSAS/UA2/DCCM3） | LAN |
| TP1/NET/TCP/IP | TCP/IP プロトコル | TP1/NET/TCP/IP<br>TCP/IP プロトコルに対応したプログラム | LAN |
| TP1/NET/XMAP3 | XMAP3 | XMAP3 | LAN |
| TP1/NET/SLU - TypeP2 | SNA プロトコル（二次局） | SNA（SLU-TypeP）プロトコル（一次局）に対応したプログラム | LAN |
| TP1/NET/UDP | UDP/IP プロトコル | TP1/NET/UDP<br>UDP/IP プロトコルに対応したプログラム | LAN |

# 付録 B　OpenTP1 の関数とコマンドの一覧

OpenTP1 で使えるライブラリ関数一覧を，次の表に示します。メッセージキューイング機能（TP1/Message Queue）のライブラリ関数については，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

## 表 B-1　OpenTP1 のライブラリ関数の一覧

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| リモートプロシジャコール | アプリケーションプログラムの開始 | dc_rpc_open | CBLDCRPC('OPEN ') |
| | SPP のサービス開始 | dc_rpc_mainloop | CBLDCRSV('MAINLOOP') |
| | 遠隔サービスの要求 | dc_rpc_call | CBLDCRPC('CALL ') |
| | 通信先を指定した遠隔サービスの呼び出し※1 | dc_rpc_call_to | － |
| | 処理結果の非同期受信 | dc_rpc_poll_any_replies | CBLDCRPC('POLLANYR') |
| | エラーが発生した非同期 RPC 要求の記述子の取得 | dc_rpc_get_error_descriptor | CBLDCRPC('GETERDES') |
| | 処理結果の受信の拒否 | dc_rpc_discard_further_replies | CBLDCRPC('DISCARDF') |
| | 非同期応答型 RPC の特定応答電文廃棄 | dc_rpc_discard_specific_reply | CBLDCRPC('DISCARDS') |
| | サービス関数のリトライ | dc_rpc_service_retry | CBLDCRPC('SVRETRY') |
| | サービス要求のスケジュールプライオリティの設定 | dc_rpc_set_service_prio | CBLDCRPC('SETSVPRI') |
| | サービス要求のスケジュールプライオリティの参照 | dc_rpc_get_service_prio | CBLDCRPC('GETSVPRI') |
| | サービスの応答待ち時間の参照 | dc_rpc_get_watch_time | CBLDCRPC('GETWATCH') |
| | サービスの応答待ち時間の更新 | dc_rpc_set_watch_time | CBLDCRPC('SETWATCH') |
| | クライアント UAP のノードアドレスの取得 | dc_rpc_get_callers_address | CBLDCRPC('GETCLADR') |
| | ゲートウェイのノードアドレスの取得 | dc_rpc_get_gateway_address | CBLDCRPC('GETGWADR') |
| | CUP への一方通知 | dc_rpc_cltsend | CBLDCRPC('CLTSEND ') |
| | アプリケーションプログラムの終了 | dc_rpc_close | CBLDCRPC('CLOSE ') |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| リモート API 機能 | rap リスナーとのコネクション確立 | dc_rap_connect | CBLDCRAP('CONNECT ')<br>CBLDCRAP('CONNECTX') |
| | rap リスナーとのコネクション解放 | dc_rap_disconnect | CBLDCRAP('DISCNCT ') |
| トランザクション制御 | トランザクションの開始 | dc_trn_begin | CBLDCTRN('BEGIN ') |
| | 連鎖モードのコミット | dc_trn_chained_commit | CBLDCTRN('C-COMMIT') |
| | 連鎖モードのロールバック | dc_trn_chained_rollback | CBLDCTRN('C-ROLL ') |
| | 非連鎖モードのコミット | dc_trn_unchained_commit | CBLDCTRN('U-COMMIT') |
| | 非連鎖モードのロールバック | dc_trn_unchained_rollback | CBLDCTRN('U-ROLL ') |
| | 現在のトランザクションに関する情報の報告 | dc_trn_info | CBLDCTRN('INFO ') |
| | リソースマネジャ接続先選択 | dc_trn_rm_select | CBLDCTRN('RMSELECT') |
| システム運用の管理 | 運用コマンドの実行 | dc_adm_call_command | CBLDCADM('COMMAND ') |
| | ユーザサーバの開始処理完了の報告 | dc_adm_complete | CBLDCADM('COMPLETE') |
| | ユーザサーバの状態の報告 | dc_adm_status | CBLDCADM('STATUS ') |
| 監査ログの出力 | 監査ログの出力 | dc_log_audit_print | CBLDCADT('PRINT ') |
| メッセージログの出力 | メッセージログの出力 | dc_logprint | CBLDCLOG('PRINT ') |
| ユーザジャーナルの取得 | ユーザジャーナルの取得 | dc_jnl_ujput | CBLDCJNL('UJPUT ') |
| ジャーナルデータの編集※2 | jnlrput 出力ファイルのクローズ | － | CBLDCJUP('CLOSERPT') |
| | jnlrput 出力ファイルのオープン | － | CBLDCJUP('OPENRPT ') |
| | jnlrput 出力ファイルからジャーナルデータの入力 | － | CBLDCJUP('RDGETRPT') |
| メッセージ送受信 | MCF 環境のオープン | dc_mcf_open | CBLDCMCF('OPEN ') |
| | MHP のサービス開始 | dc_mcf_mainloop | CBLDCMCF('MAINLOOP') |
| | メッセージの受信 | dc_mcf_receive | CBLDCMCF('RECEIVE ') |
| | 応答メッセージの送信 | dc_mcf_reply | CBLDCMCF('REPLY ') |
| | メッセージの送信 | dc_mcf_send | CBLDCMCF('SEND ') |
| | メッセージの再送 | dc_mcf_resend | CBLDCMCF('RESEND ') |
| | 同期型のメッセージ受信 | dc_mcf_recvsync | CBLDCMCF('RECVSYNC') |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| メッセージ送受信 | 同期型のメッセージ送信 | dc_mcf_sendsync | CBLDCMCF('SENDSYNC') |
| | 同期型のメッセージ送受信 | dc_mcf_sendrecv | CBLDCMCF('SENDRECV') |
| | 一時記憶データの受け取り | dc_mcf_tempget | CBLDCMCF('TEMPGET ') |
| | 一時記憶データの更新 | dc_mcf_tempput | CBLDCMCF('TEMPPUT ') |
| | 継続問い合わせ応答の終了 | dc_mcf_contend | CBLDCMCF('CONTEND ') |
| | アプリケーションプログラムの起動 | dc_mcf_execap | CBLDCMCF('EXECAP ') |
| | アプリケーション情報通知 | dc_mcf_ap_info | CBLDCMCF('APINFO ') |
| | UOC へのアプリケーション情報通知 | dc_mcf_ap_info_uoc | － |
| | ユーザタイマ監視の設定 | dc_mcf_timer_set | CBLDCMCF('TIMERSET') |
| | ユーザタイマ監視の取り消し | dc_mcf_timer_cancel | CBLDCMCF('TIMERCAN') |
| | MHP のコミット | dc_mcf_commit | CBLDCMCF('COMMIT ') |
| | MHP のロールバック | dc_mcf_rollback | CBLDCMCF('ROLLBACK') |
| | MCF 環境のクローズ | dc_mcf_close | CBLDCMCF('CLOSE ') |
| | MCF 通信サービスの状態取得 | dc_mcf_tlscom | CBLDCMCF('TLSCOM ') |
| | コネクションの状態取得 | dc_mcf_tlscn | CBLDCMCF('TLSCN ') |
| | コネクションの確立 | dc_mcf_tactcn | CBLDCMCF('TACTCN ') |
| | コネクションの解放 | dc_mcf_tdctcn | CBLDCMCF('TDCTCN ') |
| | サーバ型コネクションの確立要求の受付状態取得 | dc_mcf_tlsln | CBLDCMCF('TLSLN ') |
| | サーバ型コネクションの確立要求の受付開始 | dc_mcf_tonln | CBLDCMCF('TONLN ') |
| | サーバ型コネクションの確立要求の受付終了 | dc_mcf_tofln | CBLDCMCF('TOFLN ') |
| | アプリケーションに関するタイマ起動要求の削除 | dc_mcf_adltap | CBLDCMCF('ADLTAP ') |
| | 論理端末の状態取得 | dc_mcf_tlsle | CBLDCMCF('TLSLE ') |
| | 論理端末の閉塞 | dc_mcf_tdctle | CBLDCMCF('TDCTLE ') |
| | 論理端末の閉塞解除 | dc_mcf_tactle | CBLDCMCF('TACTLE ') |
| | 論理端末の出力キュー削除 | dc_mcf_tdlqle | CBLDCMCF('TDLQLE ') |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| DAM ファイルサービス | 論理ファイルのオープン | dc_dam_open | CBLDCDAM('DCDAMSVC','OPEN') |
| | 論理ファイルからブロックの入力 | dc_dam_read | CBLDCDAM('DCDAMSVC','READ') |
| | 論理ファイルのブロックの更新 | dc_dam_rewrite | CBLDCDAM('DCDAMSVC','REWT') |
| | 論理ファイルへブロックの出力 | dc_dam_write | CBLDCDAM('DCDAMSVC','WRIT') |
| | 論理ファイルのクローズ | dc_dam_close | CBLDCDAM('DCDAMSVC','CLOS') |
| | 論理ファイルの閉塞 | dc_dam_hold | CBLDCDAM('DCDAMSVC','HOLD') |
| | 論理ファイルの閉塞の解除 | dc_dam_release | CBLDCDAM('DCDAMSVC','RLES') |
| | 論理ファイルの状態の参照 | dc_dam_status | CBLDCDAM('DCDAMSVC','STAT') |
| | 回復対象外 DAM ファイル使用の開始 | dc_dam_start | CBLDCDAM('DCDAMSVC','STRT') |
| | 回復対象外 DAM ファイル使用の終了 | dc_dam_end | CBLDCDAM('DCDAMSVC','END ') |
| | 物理ファイルの割り当て | dc_dam_create | CBLDCDMB('DCDAMINT','CRAT') |
| | 物理ファイルのオープン | dc_dam_iopen | CBLDCDMB('DCDAMINT','OPEN') |
| | 物理ファイルからブロックの入力 | dc_dam_get | CBLDCDMB('DCDAMINT','GET ') |
| | 物理ファイルへブロックの出力 | dc_dam_put | CBLDCDMB('DCDAMINT','PUT ') |
| | 物理ファイルのブロックの検索 | dc_dam_bseek | CBLDCDMB('DCDAMINT','BSEK') |
| | 物理ファイルからブロックの直接入力 | dc_dam_dget | CBLDCDMB('DCDAMINT','DGET') |
| | 物理ファイルへブロックの直接出力 | dc_dam_dput | CBLDCDMB('DCDAMINT','DPUT') |
| | 物理ファイルのクローズ | dc_dam_iclose | CBLDCDMB('DCDAMINT','CLOS') |
| TAM ファイルサービス | TAM テーブルのオープン※1 | dc_tam_open | － |
| | TAM テーブルからレコードの入力 | dc_tam_read | CBLDCTAM('FxxR')('FxxU') |
| | TAM テーブルのレコード入力を前提の更新 | dc_tam_rewrite | CBLDCTAM('MFY ')('MFYS')('STR ') |
| | TAM テーブルのレコードの更新/追加 | dc_tam_write | CBLDCTAM('MFY ')('MFYS')('STR ') |
| | TAM テーブルのレコードの削除 | dc_tam_delete | CBLDCTAM('ERS ')('ERSR') |
| | TAM テーブルのレコードの入力取り消し※1 | dc_tam_read_cancel | － |
| | TAM テーブルの状態の取得 | dc_tam_get_inf | CBLDCTAM('GST ') |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| TAM ファイルサービス | TAM テーブルの情報の取得 | dc_tam_status | CBLDCTAM('INFO') |
| | TAM テーブルのクローズ※1 | dc_tam_close | − |
| IST サービス | IST テーブルのオープン | dc_ist_open | CBLDCIST('DCISTSVC','OPEN') |
| | IST テーブルからレコードの入力 | dc_ist_read | CBLDCIST('DCISTSVC','READ') |
| | IST テーブルへレコードの出力 | dc_ist_write | CBLDCIST('DCISTSVC','WRIT') |
| | IST テーブルのクローズ | dc_ist_close | CBLDCIST('DCISTSVC','CLOS') |
| 資源の排他制御 | 資源の排他 | dc_lck_get | CBLDCLCK('GET ') |
| | 全資源の排他の解除 | dc_lck_release_all | CBLDCLCK('RELALL ') |
| | 資源名称を指定した排他の解除 | dc_lck_release_byname | CBLDCLCK('RELNAME ') |
| XATMI インタフェース | リクエスト/レスポンス型サービスの呼び出しと応答の受信 | tpcall() | TPCALL |
| | リクエスト/レスポンス型サービスの呼び出し | tpacall() | TPACALL |
| | リクエスト/レスポンス型サービスからの非同期応答の受信 | tpgetrply() | TPGETRPLY |
| | リクエスト/レスポンス型サービスのキャンセル | tpcancel() | TPCANCEL |
| | 会話型サービスとのコネクションの確立 | tpconnect() | TPCONNECT |
| | 会話型サービスとのコネクションの切断 | tpdiscon() | TPDISCON |
| | 会話型サービスからのメッセージの受信 | tprecv() | TPRECV |
| | 会話型サービスへのメッセージの送信 | tpsend() | TPSEND |
| | 型付きバッファの割り当て | tpalloc() | −※3 |
| | 型付きバッファの解放 | tpfree() | −※3 |
| | 型付きバッファのサイズの変更 | tprealloc() | −※3 |
| | 型付きバッファの情報の取得 | tptypes() | −※3 |
| | サービス名の広告 | tpadvertise() | TPADVERTISE |
| | サービス名の広告の取り消し | tpunadvertise() | TPUNADVERTISE |
| | サービス関数のテンプレート | tpservice() | TPSVCSTART |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| XATMI インタフェース | サービス関数からのリターン | tpreturn() | TPRETURN |
| TX インタフェース | トランザクションの開始 | tx_begin() | TXBEGIN |
| | トランザクションのコミット | tx_commit() | TXCOMMIT |
| | 現在のトランザクションに関する情報の返却 | tx_info() | TXINFORM |
| | リソースマネジャ集合のオープン | tx_open() | TXOPEN |
| | トランザクションのロールバック | tx_rollback() | TXROLLBACK |
| | リソースマネジャ集合のクローズ | tx_close() | TXCLOSE |
| | commit_return 特性の設定 | tx_set_commit_return() | TXSETCOMMITRET |
| | transaction_control 特性の設定 | tx_set_transaction_control() | TXSETTRANCTL |
| | transaction_timeout 特性の設定 | tx_set_transaction_timeout() | TXSETTIMEOUT |
| マルチノード機能※1 | OpenTP1 ノードのステータス取得の開始 | dc_adm_get_nd_status_begin | − |
| | OpenTP1 ノードのステータスの取得 | dc_adm_get_nd_status_next | − |
| | 指定した OpenTP1 ノードのステータスの取得 | dc_adm_get_nd_status | − |
| | OpenTP1 ノードのステータス取得の終了 | dc_adm_get_nd_status_done | − |
| | ノード識別子の取得の開始 | dc_adm_get_nodeconf_begin | − |
| | ノード識別子の取得 | dc_adm_get_nodeconf_next | − |
| | ノード識別子の取得の終了 | dc_adm_get_nodeconf_done | − |
| | 指定したノード識別子の取得 | dc_adm_get_node_id | − |
| | ユーザサーバのステータス取得の開始 | dc_adm_get_sv_status_begin | − |
| | ユーザサーバのステータスの取得 | dc_adm_get_sv_status_next | − |
| | 指定したユーザサーバのステータスの取得 | dc_adm_get_sv_status | − |
| | ユーザサーバのステータス取得の終了 | dc_adm_get_sv_status_done | − |

| 機能 | | ライブラリ関数名 | |
|---|---|---|---|
| | | C 言語ライブラリ | COBOL-UAP 作成用プログラム |
| オンラインテスタの管理 | ユーザサーバのテスト状態の報告 | dc_uto_test_status | CBLDCUTO('T-STATUS') |
| 性能検証用トレース | ユーザ固有の性能検証用トレースの取得 | dc_prf_utrace_put | CBLDCPRF('PRFPUT ') |
| | 性能検証用トレース取得通番の通知 | dc_prf_get_trace_num | CBLDCPRF('PRFGETN ') |
| リアルタイム統計情報サービス | 任意区間でのリアルタイム統計情報の取得 | dc_rts_utrace_put | CBLDCRTS('RTSPUT ') |

(凡例)

－：該当しません。

注※1

COBOL-UAP 作成用プログラムは使えません。

注※2

ジャーナルデータの編集では，C 言語の API は使えません。

注※3

XATMI インタフェースでは，該当する COBOL の API はありません。

OpenTP1 のコマンド一覧を，次の表に示します。メッセージキューイング機能（TP1/Message Queue）のコマンドについては，マニュアル「TP1/Message Queue 使用の手引」を参照してください。

## 表 B-2　OpenTP1 のコマンド一覧

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| システム管理 | OpenTP1 の OS への登録と削除 | dcsetup | スーパユーザ |
| | プロセスサービスの再起動および定義の反映 | dcreset | OpenTP1 管理者 |
| | OpenTP1 の内部制御用資源の確保と解放 | dcmakeup | OpenTP1 管理者 |
| | OpenTP1 の開始 | dcstart | OpenTP1 管理者 |
| | OpenTP1 の終了 | dcstop | OpenTP1 管理者※ |
| | システム統計情報の取得開始，終了 | dcstats | OpenTP1 管理者 |
| | マルチノードエリア，サブエリアの開始 | dcmstart | OpenTP1 管理者 |
| | マルチノードエリア，サブエリアの終了 | dcmstop | OpenTP1 管理者 |
| | シナリオテンプレートからの OpenTP1 コマンドの実行 | dcjcmdex | OpenTP1 管理者 |
| | システム定義のオペランドの指定 | dcjchconf | OpenTP1 管理者 |
| | ドメイン定義ファイルの更新 | dcjnamch | OpenTP1 管理者 |
| | OpenTP1 ノードの状態表示 | dcndls | 一般ユーザ |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| システム管理 | 共用メモリの状態表示 | dcshmls | 一般ユーザ |
| | OpenTP1 の状態表示 | dcstatus | OpenTP1 管理者 |
| | 一時クローズ処理の実行状態の表示 | rpcstat | 一般ユーザ |
| | OpenTP1 の標準出力，標準エラー出力のリダイレクト | prctee | OpenTP1 管理者 |
| | prctee プロセスの停止と再開始 | prctctrl | スーパユーザ |
| | 保守資料の取得 | dcrasget | OpenTP1 管理者 |
| | システム統計情報の標準出力へのリアルタイム編集出力 | dcreport | OpenTP1 管理者 |
| | トラブルシュート情報の削除 | dccspool | OpenTP1 管理者 |
| | システム定義のチェック | dcdefchk | OpenTP1 管理者 |
| | 製品情報の表示 | dcpplist | OpenTP1 管理者 |
| サーバ管理 | サーバの開始 | dcsvstart | OpenTP1 管理者 |
| | サーバの終了 | dcsvstop | OpenTP1 管理者 |
| | サーバの状態表示 | prcls | 一般ユーザ |
| | ユーザサーバ，およびユーザサーバから起動されるコマンドのサーチパスの表示 | prcpathls | 一般ユーザ |
| | ユーザサーバ，およびユーザサーバから起動されるコマンドのサーチパスの変更 | prcpath | OpenTP1 管理者 |
| | UAP 共用ライブラリのサーチパスの表示 | prcdlpathls | 一般ユーザ |
| | UAP 共用ライブラリのサーチパスの変更 | prcdlpath | OpenTP1 管理者 |
| | OpenTP1 のプロセスの強制停止 | prckill | OpenTP1 管理者 |
| スケジュール管理 | スケジュールの状態表示 | scdls | 一般ユーザ |
| | スケジュールの閉塞 | scdhold | OpenTP1 管理者 |
| | スケジュールの再開始 | scdrles | OpenTP1 管理者 |
| | プロセス数の変更 | scdchprc | OpenTP1 管理者 |
| | プロセスの停止および再起動 | scdrsprc | OpenTP1 管理者 |
| トランザクション管理 | トランザクションの状態表示 | trnls | 一般ユーザ |
| | トランザクションの強制コミット | trncmt | OpenTP1 管理者 |
| | トランザクションの強制ロールバック | trnrbk | OpenTP1 管理者 |
| | トランザクションの強制終了 | trnfgt | OpenTP1 管理者 |
| | トランザクション統計情報の取得開始，終了 | trnstics | OpenTP1 管理者 |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| トランザクション管理 | 未決着トランザクション情報ファイルの削除 | trndlinf | OpenTP1 管理者 |
| | OSI TP 通信の未決着トランザクション情報の表示 | tptrnls | 一般ユーザ |
| XA リソース管理 | XAR イベントトレース情報の表示 | xarevtr | 一般ユーザ |
| | XAR ファイルの状態表示 | xarfills | 一般ユーザ |
| | XAR トランザクション状態の変更 | xarforce | OpenTP1 管理者 |
| | XA リソースサービスの閉塞 | xarhold | OpenTP1 管理者 |
| | XAR ファイルの作成 | xarinit | OpenTP1 管理者 |
| | XAR トランザクション情報の表示 | xarls | 一般ユーザ |
| | XA リソースサービスの閉塞解除 | xarrles | OpenTP1 管理者 |
| | XAR ファイルの削除 | xarrm | OpenTP1 管理者 |
| 排他管理 | 排他情報の表示 | lckls | 一般ユーザ |
| | 排他制御用テーブルのプール情報の表示 | lckpool | 一般ユーザ |
| | デッドロック情報ファイルとタイムアウト情報ファイルの削除 | lckrminf | OpenTP1 管理者 |
| ネーム管理 | OpenTP1 起動確認，キャッシュ削除 | namalivechk | OpenTP1 管理者 |
| | ドメイン代表スケジュールサービスの登録，削除 | namdomainsetup | スーパユーザ |
| | ドメイン構成の変更（システム共通定義使用） | namndchg | OpenTP1 管理者 |
| | ドメイン構成の変更（ドメイン定義ファイル使用） | namchgfl | OpenTP1 管理者 |
| | 起動通知情報の強制的無効化 | namunavl | OpenTP1 管理者 |
| | OpenTP1 のサーバ情報の表示 | namsvinf | OpenTP1 管理者 |
| | RPC 抑止リストの操作 | namblad | OpenTP1 管理者 |
| | ノード情報の削除 | namndrm | OpenTP1 管理者 |
| | マネジャノードの変更 | nammstr | OpenTP1 管理者 |
| | ノードリストファイルの作成 | namnlcre | OpenTP1 管理者 |
| | ノードリストファイルの内容表示 | namnldsp | 一般ユーザ |
| | ノードリストファイルの削除 | namnldel | OpenTP1 管理者 |
| | ノードのオプション情報の変更 | namndopt | OpenTP1 管理者 |
| メッセージログ管理 | メッセージログファイルの内容表示 | logcat | 一般ユーザ |
| | メッセージログのリアルタイム出力機能の切り替え | logcon | OpenTP1 管理者 |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| 監査ログ管理 | 監査ログ機能の環境設定 | dcauditsetup | スーパユーザ |
| OpenTP1 ファイル管理 | OpenTP1 ファイルシステムの初期設定 | filmkfs | OpenTP1 管理者 |
| | OpenTP1 ファイルシステムの状態表示 | filstatfs | 一般ユーザ |
| | OpenTP1 ファイルシステムの内容表示 | fills | 一般ユーザ |
| | OpenTP1 ファイルシステムのバックアップ | filbkup | OpenTP1 管理者 |
| | OpenTP1 ファイルシステムのリストア | filrstr | OpenTP1 管理者 |
| | OpenTP1 ファイルグループの変更 | filchgrp | OpenTP1 管理者 |
| | OpenTP1 ファイルのアクセス許可モードの変更 | filchmod | OpenTP1 管理者 |
| | OpenTP1 ファイル所有者の変更 | filchown | OpenTP1 管理者 |
| ステータスファイル管理 | ステータスファイルの作成，初期設定 | stsinit | OpenTP1 管理者 |
| | ステータスファイルの状態表示 | stsls | 一般ユーザ |
| | ステータスファイルの内容表示 | stsfills | 一般ユーザ |
| | ステータスファイルのオープン | stsopen | OpenTP1 管理者 |
| | ステータスファイルのクローズ | stsclose | OpenTP1 管理者 |
| | ステータスファイルの削除 | stsrm | OpenTP1 管理者 |
| | ステータスファイルのスワップ | stsswap | OpenTP1 管理者 |
| ジャーナル関係のファイル管理 | ジャーナル関係のファイルの初期設定 | jnlinit | OpenTP1 管理者 |
| | ジャーナル関係のファイル情報の表示 | jnlls | 一般ユーザ |
| | 再開始中読み込み済ジャーナル関係のファイル情報の表示 | jnlrinf | OpenTP1 管理者 |
| | ジャーナル関係のファイルのオープン | jnlopnfg | OpenTP1 管理者 |
| | ジャーナル関係のファイルのクローズ | jnlclsfg | OpenTP1 管理者 |
| | ジャーナル関係の物理ファイルの割り当て | jnladdpf | OpenTP1 管理者 |
| | ジャーナル関係の物理ファイルの削除 | jnldelpf | OpenTP1 管理者 |
| | ジャーナル関係のファイルのスワップ | jnlswpfg | OpenTP1 管理者 |
| | ジャーナル関係のファイルの削除 | jnlrm | OpenTP1 管理者 |
| | ジャーナル関係のファイルのステータス変更 | jnlchgfg | OpenTP1 管理者 |
| | ジャーナル関係のファイルのアンロード | jnlunlfg | OpenTP1 管理者 |
| | 自動アンロード機能の制御 | jnlatunl | OpenTP1 管理者 |
| | ジャーナル関係のファイルの回復 | jnlmkrf | OpenTP1 管理者 |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| ジャーナル関係のファイル管理 | ファイル回復用ジャーナルの集積 | jnlcolc | OpenTP1 管理者 |
| | アンロードジャーナルファイルの複写 | jnlcopy | OpenTP1 管理者 |
| | アーカイブ状態の表示 | jnlarls | 一般ユーザ |
| | アンロードジャーナルファイル，またはグローバルアーカイブアンロードジャーナルファイルの編集出力 | jnledit | OpenTP1 管理者 |
| | アンロードジャーナルファイル，またはグローバルアーカイブアンロードジャーナルファイルのレコード出力 | jnlrput | OpenTP1 管理者 |
| | アンロードジャーナルファイル，およびグローバルアーカイブアンロードジャーナルファイルの時系列ソート，およびマージ | jnlsort | OpenTP1 管理者 |
| | 稼働統計情報の出力 | jnlstts | OpenTP1 管理者 |
| | MCF 稼働統計情報の出力 | jnlmcst | OpenTP1 管理者 |
| | リソースグループの接続の強制解除 | jnlardis | OpenTP1 管理者 |
| DAM ファイル管理 | 物理ファイルの初期設定 | damload | OpenTP1 管理者 |
| | 論理ファイルの状態表示 | damls | 一般ユーザ |
| | 論理ファイルの追加 | damadd | OpenTP1 管理者 |
| | 論理ファイルの切り離し | damrm | OpenTP1 管理者 |
| | 論理ファイルの論理閉塞 | damhold | OpenTP1 管理者 |
| | 論理ファイルの閉塞解除 | damrles | OpenTP1 管理者 |
| | 物理ファイルの削除 | damdel | OpenTP1 管理者 |
| | 物理ファイルのバックアップ | dambkup | OpenTP1 管理者 |
| | 物理ファイルのリストア | damrstr | OpenTP1 管理者 |
| | 論理ファイルの回復 | damfrc | OpenTP1 管理者 |
| | キャッシュブロック数しきい値の設定 | damchdef | OpenTP1 管理者 |
| | キャッシュブロック数の取得 | damchinf | 一般ユーザ |
| TAM ファイル管理 | TAM ファイルの初期作成 | tamcre | OpenTP1 管理者 |
| | TAM テーブルの状態表示 | tamls | 一般ユーザ |
| | TAM テーブルの追加 | tamadd | OpenTP1 管理者 |
| | TAM テーブルの切り離し | tamrm | OpenTP1 管理者 |
| | TAM テーブルの論理閉塞 | tamhold | OpenTP1 管理者 |
| | TAM テーブルの閉塞解除 | tamrles | OpenTP1 管理者 |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| TAM ファイル管理 | TAM テーブルのロード | tamload | OpenTP1 管理者 |
| | TAM テーブルのアンロード | tamunload | OpenTP1 管理者 |
| | TAM ファイルの削除 | tamdel | OpenTP1 管理者 |
| | TAM ファイルのバックアップ | tambkup | OpenTP1 管理者 |
| | TAM ファイルのリストア | tamrstr | OpenTP1 管理者 |
| | TAM ファイルの回復 | tamfrc | OpenTP1 管理者 |
| | TAM 排他資源名称の変換 | tamlckls | 一般ユーザ |
| | ハッシュ形式の TAM ファイルおよび TAM テーブルのシノニム情報の表示 | tamhsls | OpenTP1 管理者 |
| メッセージキューファイル管理 | キューグループの状態表示 | quels | 一般ユーザ |
| | メッセージキュー用物理ファイルの割り当て | queinit | OpenTP1 管理者 |
| | メッセージキュー用物理ファイルの削除 | querm | OpenTP1 管理者 |
| リソースマネジャ管理 | リソースマネジャの情報の表示 | trnlsrm | 一般ユーザ |
| | リソースマネジャの登録 | trnlnkrm | OpenTP1 管理者 |
| | トランザクション制御用オブジェクトファイルの作成 | trnmkobj | OpenTP1 管理者 |
| トレース管理 | UAP トレースの編集出力 | uatdump | 一般ユーザ |
| | RPC トレースのマージ | rpcmrg | 一般ユーザ |
| | RPC トレースの出力 | rpcdump | 一般ユーザ |
| | 共用メモリダンプの出力 | usmdump | OpenTP1 管理者 |
| リモート API 管理 | リモート API 機能の実行環境の設定 | rapsetup | OpenTP1 管理者 |
| | リモート API 機能に使用する定義の自動生成 | rapdfgen | OpenTP1 管理者 |
| | rap リスナーおよび rap サーバの状態表示 | rapls | OpenTP1 管理者 |
| 性能検証用トレース管理 | トレース情報ファイルの編集出力 | prfed | OpenTP1 管理者 |
| | トレース情報ファイルの取り出し | prfget | OpenTP1 管理者 |
| リアルタイム統計情報サービス管理 | RTS ログファイルの編集出力 | rtsedit | 一般ユーザ |
| | リアルタイム統計情報の標準出力への出力 | rtsls | 一般ユーザ |
| | リアルタイム統計情報サービスの実行環境の設定 | rtssetup | OpenTP1 管理者 |
| | リアルタイム統計情報の設定変更 | rtsstats | OpenTP1 管理者 |
| OpenTP1 解析支援 | 性能検証用トレース解析 | dcalzprf | OpenTP1 管理者 |
| コネクション管理 | コネクションの状態表示 | mcftlscn | 一般ユーザ |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| コネクション管理 | コネクションの確立 | mcftactcn | OpenTP1 管理者 |
| | コネクションの解放 | mcftdctcn | OpenTP1 管理者 |
| | コネクションの切り替え | mcftchcn | OpenTP1 管理者 |
| | ネットワークの状態表示 | mcftlsln | 一般ユーザ |
| | サーバ型コネクションの確立要求の受付開始 | mcftonln | OpenTP1 管理者 |
| | サーバ型コネクションの確立要求の受付終了 | mcftofln | OpenTP1 管理者 |
| | メッセージ多重処理状況の表示 | mcftlstrd | OpenTP1 管理者 |
| アプリケーション管理 | アプリケーションの状態表示 | mcfalsap | 一般ユーザ |
| | アプリケーションの閉塞 | mcfadctap | OpenTP1 管理者 |
| | アプリケーションの閉塞解除 | mcfaactap | OpenTP1 管理者 |
| | アプリケーション異常終了回数の初期化 | mcfaclcap | OpenTP1 管理者 |
| | アプリケーションに関するタイマ起動要求の表示 | mcfalstap | 一般ユーザ |
| | アプリケーションに関するタイマ起動要求の削除 | mcfadltap | OpenTP1 管理者 |
| アプリケーション運用支援 | アプリケーションプログラムの起動 | mcfuevt | 一般ユーザ |
| 論理端末管理 | 論理端末の状態表示 | mcftlsle | 一般ユーザ |
| | 論理端末の閉塞 | mcftdctle | OpenTP1 管理者 |
| | 論理端末の閉塞解除 | mcftactle | OpenTP1 管理者 |
| | 論理端末のメッセージキューの先頭スキップ | mcftspqle | OpenTP1 管理者 |
| | 論理端末の出力キュー処理の保留 | mcfthldoq | OpenTP1 管理者 |
| | 論理端末の出力キュー処理の保留解除 | mcftrlsoq | OpenTP1 管理者 |
| | 論理端末の出力キュー削除 | mcftdlqle | OpenTP1 管理者 |
| | 論理端末に関するメッセージジャーナルの取得開始 | mcftactmj | OpenTP1 管理者 |
| | 論理端末に関するメッセージジャーナルの取得終了 | mcftdctmj | OpenTP1 管理者 |
| | 論理端末に対する継続問い合わせ応答処理の強制終了 | mcftendct | OpenTP1 管理者 |
| | 端末代行の開始 | mcftstalt | OpenTP1 管理者 |
| | 端末代行の終了 | mcftedalt | OpenTP1 管理者 |
| サービスグループ管理 | サービスグループの状態表示 | mcftlssg | 一般ユーザ |
| | サービスグループの閉塞 | mcftdctsg | OpenTP1 管理者 |

| 機能 | | コマンド名 | 実行権限 |
|---|---|---|---|
| サービスグループ管理 | サービスグループの閉塞解除 | mcftactsg | OpenTP1 管理者 |
| | サービスグループの入力キュー処理の保留 | mcfthldiq | OpenTP1 管理者 |
| | サービスグループの入力キュー処理の保留解除 | mcftrlsiq | OpenTP1 管理者 |
| | サービスグループの入力キュー削除 | mcftdlqsg | OpenTP1 管理者 |
| サービス管理 | サービスの状態表示 | mcftlssv | 一般ユーザ |
| | サービスの閉塞 | mcftdctsv | OpenTP1 管理者 |
| | サービスの閉塞解除 | mcftactsv | OpenTP1 管理者 |
| バッファ管理 | バッファグループの使用状況表示 | mcftlsbuf | 一般ユーザ |
| マップ管理 | マップファイルのパス名変更 | dcmapchg | OpenTP1 管理者 |
| | マップファイルのロード済み資源の表示 | dcmapls | OpenTP1 管理者 |
| キュー管理 | 入出力キューの内容複写 | mcftdmpqu | 一般ユーザ |
| MCF トレース取得管理 | MCF トレースファイルの強制スワップ | mcftswptr | OpenTP1 管理者 |
| | MCF トレース取得の開始 | mcftstrtr | OpenTP1 管理者 |
| | MCF トレース取得の終了 | mcftstptr | OpenTP1 管理者 |
| MCF 稼働統計情報管理 | MCF 稼働統計情報の編集 | mcfreport | OpenTP1 管理者 |
| | MCF 稼働統計情報の出力 | mcfstats | OpenTP1 管理者 |
| MCF 通信サービス管理 | MCF 通信サービスの部分停止 | mcftstop | OpenTP1 管理者 |
| | MCF 通信サービスの部分開始 | mcftstart | OpenTP1 管理者 |
| | MCF 通信サービスの状態参照と開始待ち合わせ | mcftlscom | OpenTP1 管理者 |
| ユーザタイマ管理 | ユーザタイマ監視の状態表示 | mcftlsutm | 一般ユーザ |

注

プロトコルによって固有のコマンドがあります。プロトコル固有のコマンドについては，マニュアル「OpenTP1 プロトコル」の該当するプロトコル編を参照してください。

注※

dcstop コマンドを UAP から実行する場合は，バックグラウンドで実行してください。

# 付録 C　バージョンアップ時の変更点

各バージョンでの変更点を次に示す分類ごとに示します。

- 関数，定義およびコマンドの追加と削除
- 動作の変更
- 関数，定義およびコマンドのデフォルト値の変更

## 付録 C.1　07-56 での変更点

TP1/Server Base 07-56 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–1　TP1/Server Base 07-56 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システム共通定義<br>• ipc_notify_response_host オペランド<br>• ipc_response_host オペランド |
| | コマンド | なし |

TP1/Server Base 07-56 での動作の変更はありません。

TP1/Server Base 07-56 でのデフォルト値の変更はありません。

## 付録 C.2　07-53 での変更点

TP1/Server Base 07-53 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–2　TP1/Server Base 07-53 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システム共通定義<br>• tp1_monitor_time オペランド<br>• tp1_monitor_kill_signal オペランド |
| | コマンド | なし |

TP1/Server Base 07-53 での動作の変更点を次の表に示します。

表 C–3　TP1/Server Base 07-53 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | トランザクションサービス定義の trn_extend_function の指定値に 00000002 を追加 |
| メッセージ | なし |
| その他 | プロセスサービスを無応答状態から自動で復帰できるよう変更（OpenTP1 監視機能） |

TP1/Server Base 07-53 でのデフォルト値の変更はありません。

## 付録 C.3　07-52 での変更点

TP1/Server Base 07-52 での関数，定義およびコマンドの追加と削除を次の表に示します。

表 C–4　TP1/Server Base 07-52 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システムジャーナルサービス定義<br>• jnl_auto_unload_continue オペランド |
| | | システム環境定義<br>• ha_switch_error_retry_count オペランド<br>• ha_switch_error_retry_interval オペランド |
| | コマンド | dcstatus コマンド |
| 削除 | なし | |

TP1/Server Base 07-52 での動作の変更はありません。

TP1/Server Base 07-52 でのデフォルト値の変更はありません。

## 付録 C.4　07-51 での変更点

TP1/Server Base 07-51 での関数，定義およびコマンドの追加と削除を次の表に示します。

表 C–5　TP1/Server Base 07-51 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システム共通定義<br>• rpc_close_after_send オペランド |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | プロセスサービス定義<br>• prc_hugepage_group_id オペランド |
| | | ユーザサービスデフォルト定義<br>• rpc_close_after_send オペランド |
| | | ユーザサービス定義<br>• rpc_close_after_send オペランド |
| | コマンド | dcshmls コマンドの-b オプション |
| | | scdls コマンド<br>• -am オプション<br>• -s サーバ名 -m オプション |

TP1/Server Base 07-51 での動作の変更点を次の表に示します。

## 表 C-6　TP1/Server Base 07-51 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | 次のオペランドに指定できる値（0～1945600）を，32 ビット版の場合は 0～1945600，64 ビット版の場合は 0～67108864 に変更<br>• システム環境定義<br>static_shmpool_size<br>dynamic_shmpool_size |
| | 次のオペランドに Linux の Hugepage 機能を指定できるよう変更<br>• システム環境定義<br>shmpool_attribute<br>• DAM サービス定義<br>dam_cache_attribute<br>• TAM サービス定義<br>tam_pool_attri |
| | 次のオペランドに 65 および 67 を指定できるように変更<br>• システム共通定義<br>prc_coredump_filter<br>• ユーザサービスデフォルト定義<br>prc_coredump_filter<br>• ユーザサービス定義<br>prc_coredump_filter |
| メッセージ | KFCA00107-E の errno 値に ENOMEM および EPERM を追加。 |
| | KFCA00880-I の使用方法のメッセージに次のオプションを追加<br>• -am<br>• -s サーバ名 -m |

| 分類 | 内容 |
|---|---|
| その他 | チェックポイントダンプファイルのサイズの見積もり式<br>32bit と 64bit で見積もり式を変更した。 |
| | ステータスファイルのサイズの見積もり式<br>32bit と 64bit でサイズを変更した。 |

TP1/Server Base 07-51 でのデフォルト値の変更はありません。

TP1/Message Control 07-51，TP1/NET/Library 07-51 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–7　TP1/Message Control 07-51，TP1/NET/Library 07-51 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | MCF マネジャ定義<br>• mcfmuap 定義コマンドの-c オプション<br>itqretryorder オペランド |
| | コマンド | なし |
| 削除 | コマンド | dcmapchg コマンド<br>• --mP<br>• -aP |
| | | mcftactcn コマンド<br>• -S |

TP1/Message Control 07-51，TP1/NET/Library 07-51 での動作の変更点はありません。

TP1/Message Control 07-51，TP1/NET/Library 07-51 でのデフォルト値の変更はありません。

## 付録 C.5　07-50 での変更点

TP1/Server Base 07-50 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–8　TP1/Server Base 07-50 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | システム環境定義<br>• default_value_option オペランド |
| | | チェックポイントダンプサービス定義<br>• cpd_message_id_change_level オペランド |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | ジャーナルサービス定義<br>• jnl_message_id_change_level オペランド |
| | | 性能検証用トレース定義<br>• prf_buff_size オペランド |
| | | ユーザサービスデフォルト定義<br>• rpc_rap_inquire_time_check オペランド |
| | | ユーザサービス定義<br>• rpc_rap_inquire_time_check オペランド |
| | コマンド | filmkfs コマンドの-t オプション |
| | | filrstr コマンドの-f オプション |
| | | filstatfs コマンドの-T オプション |

TP1/Server Base 07-50 での動作の変更点を次の表に示します。

## 表 C–9　TP1/Server Base 07-50 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | システム環境定義の server_count オペランドに指定できる最大値（4096）を，9999 に変更 |
| | スケジュールサービス定義の scd_server_count オペランドに指定できる最大値（4096）を，8192 に変更 |
| メッセージ | KFCA01861-E の「コマンドで発生したエラー」に VERSION を追加 |
| | KFCA01502-I の使用方法のメッセージに-f オプションを追加 |
| | KFCA01505-I の使用方法のメッセージに-t オプションを追加 |
| | KFCA01551-I の使用方法のメッセージに-T オプションを追加 |
| その他 | 性能検証用トレース情報を格納するバッファサイズを 1024 キロバイト（デフォルト値）に変更したことに伴い，静的共用メモリの使用量が 928 キロバイト増加 |

TP1/Server Base 07-50 でのデフォルト値の変更点を次の表に示します。

## 表 C–10　TP1/Server Base 07-50 でのデフォルト値の変更

| 分類 | 内容 |
|---|---|
| 定義 | システム共通定義の trn_prf_trace_level オペランドのデフォルト値を 00000001 から 00000003 に変更 |
| | システム共通定義の ipc_listen_sockbufset オペランドのデフォルト値を N から Y に変更 |
| | システム共通定義の fil_prf_trace_delay_time オペランドのデフォルト値を 10 から 3 に変更 |
| | ロックサービス定義の lck_deadlock_info オペランドのデフォルト値を N から Y に変更 |

| 分類 | 内容 |
|---|---|
| 定義 | スケジュールサービス定義の ipc_tcpnodelay オペランドのデフォルト値を N から Y に変更 |
| | ログサービス定義の log_filesize オペランドのデフォルト値を 1024 から 10240 に変更 |
| | 性能検証用トレース定義の prf_file_size オペランドのデフォルト値を 1024 から 10240 に変更 |
| | ユーザサービスデフォルト定義の ipc_tcpnodelay オペランドのデフォルト値を N から Y に変更 |
| その他 | 性能検証用トレース情報を格納するバッファサイズ（96 キロバイト）を 1024 キロバイトに変更 |

TP1/Message Control 07-50，TP1/NET/Library 07-50 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–11　TP1/Message Control 07-50，TP1/NET/Library 07-50 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | MCF マネジャ定義<br>• mcfmuap 定義コマンドの-c オプション<br>otqinhold オペランド<br>errevt_recovery オペランド |
| | | システムサービス共通情報定義<br>• mcf_start_watch_interval オペランド |
| | コマンド | mcftlscom コマンド<br>• -x オプション |
| | | mcfmngr コマンド<br>• -r オプション |
| | | mcfcomn コマンド<br>• -r オプション |
| | | mcfpsvr コマンド<br>• -r オプション |
| | | mcfapli コマンド<br>• -r オプション |
| | | mcflink コマンド<br>• -r オプション |
| 削除 | なし | |

TP1/Message Control 07-50，TP1/NET/Library 07-50 での動作の変更点を次の表に示します。

## 表 C–12　TP1/Message Control 07-50，TP1/NET/Library 07-50 での動作の変更

| 分類 | 内容 |
|---|---|
| 関数 | 次の関数で，UAP トレースを取得するように変更<br>• dc_mcf_adltap<br>• dc_mcf_tactcn<br>• dc_mcf_tactle<br>• dc_mcf_tdctcn<br>• dc_mcf_tdctle<br>• dc_mcf_tdlqle<br>• dc_mcf_tlscn<br>• dc_mcf_tlscom<br>• dc_mcf_tlsle<br>• dc_mcf_tlsln<br>• dc_mcf_tofln<br>• dc_mcf_tonln<br>• CBLDCMCF('ADLTAP ')<br>• CBLDCMCF('TACTCN ')<br>• CBLDCMCF('TACTLE ')<br>• CBLDCMCF('TDCTCN ')<br>• CBLDCMCF('TDCTLE ')<br>• CBLDCMCF('TDLQLE ')<br>• CBLDCMCF('TLSCN ')<br>• CBLDCMCF('TLSCOM ')<br>• CBLDCMCF('TLSLE ')<br>• CBLDCMCF('TLSLN ')<br>• CBLDCMCF('TOFLN ')<br>• CBLDCMCF('TONLN ') |
| 定義 | mcfmsvg 定義コマンドの指定数の上限を 4096 から 8192 に変更 |
| | mcfaalcap 定義コマンドの指定数の上限を 4096 から 8192 に変更 |
| | システムサービス共通情報定義<br>• max_socket_descriptors オペランドの指定値の上限を 2047 から 3596 に変更<br>• max_open_fds オペランドの指定値の上限を 2016 から 4032 に変更 |
| コマンド | 次のコマンドで，環境変数 DCMCFCMDLOG を設定していなくてもコマンドログを取得するように変更<br>• dcmapchg<br>• mcfstats<br>• mcftactsg<br>• mcftactsv<br>• mcftdctsg<br>• mcftdctsv<br>• mcftdlqsg<br>• mcfthldiq |

| 分類 | 内容 |
|---|---|
| コマンド | • mcftlscom<br>• mcftrlsiq<br>• mcftstart<br>• mcftstop |
| その他 | エラーイベントの割り当て先がディスクキューの場合，エラーイベントを OpenTP1 の再開始時に引き継げるように変更（UAP 共通定義（mcfmuap -c）の errevt_recovery オペランドで変更前の動作に戻すことができます） |

TP1/Message Control 07-50，TP1/NET/Library 07-50 でのデフォルト値の変更点を次の表に示します。

### 表 C–13　TP1/Message Control 07-50，TP1/NET/Library 07-50 でのデフォルト値の変更

| 分類 | 内容 |
|---|---|
| 定義 | MCF マネジャ定義<br>• mcfmcomn 定義コマンドの-i オプションのデフォルト値（inc）を msg に変更<br>• mcfmuap 定義コマンドの-e オプションの segsize オペランドのデフォルト値（512）を 32768 に変更<br>• mcfmuap 定義コマンドの-c オプションの order オペランドのデフォルト値（function）を commit に変更<br>• mcfmuap 定義コマンドの-c オプションの noansreply オペランドのデフォルト値（no）を yes に変更 |
| | システムサービス共通情報定義<br>• mcf_prf_trace_level オペランドのデフォルト値（00000000）を 00000001 に変更 |

## 付録 C.6　07-07 での変更点

TP1/Server Base 07-07 での関数，定義およびコマンドの追加と削除はありません。

TP1/Server Base 07-07 での動作の変更点を次の表に示します。

### 表 C–14　TP1/Server Base 07-07 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | prcsvpath に指定するパス名の最大文字数（255）を 1023 文字に変更 |
| | 次の実時間監視機能を使用したとき，マイナス誤差による実時間監視満了（指定された時間より早くタイムアウト）が発生しないように変更<br>ユーザサービス定義<br>• service_expiration_time（サービス関数開始から終了までの実行監視時間）<br>• trn_expiration_time（トランザクションブランチ限界経過時間）<br>• trn_completion_limit_time（トランザクション完了限界時間） |

| 分類 | 内容 |
|---|---|
| 定義 | ユーザサービスデフォルト定義<br>• service_expiration_time（サービス関数開始から終了までの実行監視時間）<br>• trn_expiration_time（トランザクションブランチ限界経過時間）<br>• trn_completion_limit_time（トランザクション完了限界時間）<br>トランザクションサービス定義<br>• trn_expiration_time（トランザクションブランチ限界経過時間）<br>• trn_completion_limit_time（トランザクション完了限界時間）<br>クライアントサービス定義<br>• trn_expiration_time（トランザクションブランチ限界経過時間）<br>• trn_completion_limit_time（トランザクション完了限界時間）<br>rap リスナーサービス定義<br>• trn_expiration_time（トランザクションブランチ限界経過時間）<br>• trn_completion_limit_time（トランザクション完了限界時間）<br>ネットワークコミュニケーション定義<br>• mcfaalcap 定義コマンドの-v オプションの ntmetim（非トランザクション MHP 限界経過時間）<br>• mcfmuap 定義コマンドの-u オプションの ntmetim（非トランザクション MHP 限界経過時間） |
| | ログサービス定義の log_notify_xxx オペランドを Linux 版でも動作するように変更 |
| コマンド | prcpath に指定するパス名の最大文字数（255）を 1023 文字に変更 |
| | prcpathls で出力するパス名の最大文字数（255）を 1023 文字に変更 |
| | dcdefchk -l に次の定義チェックを追加・変更<br>• DCSYSLOGCTYPE に euc が指定されているかどうかのチェックを追加（Linux 版だけ）<br>• Linux 版の log_notify_xxx の定義チェックを AIX や HP-UX 版と同様のチェックに変更 |
| | logcon コマンドにコマンドログ取得機能を追加 |
| | filmkfs コマンド<br>• -s オプションの指定値の上限を 2048 から 4096 に変更<br>• Linux 版の-n オプションに指定可能な上限を AIX や HP-UX 版と同じ上限（4095）に変更 |

TP1/Server Base 07-07 でのデフォルト値の変更はありません。

## 付録 C.7　07-06 での変更点

TP1/Server Base 07-06 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–15　TP1/Server Base 07-06 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システム共通定義 |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | • name_service_mode オペランド<br>• name_manager_node オペランド<br>• name_remove_down_node オペランド<br>• name_node_add_policy オペランド |
| | | ネームサービス定義<br>• name_start_watch_time オペランド<br>• name_start_retry_count オペランド<br>• name_start_retry_interval オペランド<br>• name_start_error オペランド<br>• name_sync_ready_time オペランド<br>• namnlfil 定義コマンド |
| | | スケジュールサービス定義<br>• scdbufgrp 定義コマンドの-e，-s，-p オプション |
| | | ユーザサービスデフォルト定義<br>• scd_refresh_process オペランド<br>• scd_process_ctl_opt オペランド<br>• scdbufgrp 定義コマンドの-s，-p オプション |
| | | ユーザサービス定義<br>• scd_refresh_process オペランド<br>• scd_process_ctl_opt オペランド<br>• scdbufgrp 定義コマンドの-s，-p オプション |
| | コマンド | dcalzprf コマンド |
| | | nammstr コマンド |
| | | namndopt コマンド |
| | | namndrm コマンド |
| | | namnlcre コマンド |
| | | namnldel コマンド |
| | | namnldsp コマンド |
| | | namsvinf コマンドの-x オプション |
| 削除 | 定義 | MCF アプリケーション定義<br>• mcfaalcap 定義コマンドの type オペランド |
| | | MCF マネジャ定義<br>• mcfmcomn 定義コマンドの-r，-c オプション |

TP1/Server Base 07-06 での動作の変更点を次の表に示します。

表 C–16　TP1/Server Base 07-06 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | スケジュールサービス定義の scdbufgrp 定義コマンドの-n オプションに指定できる最大値（61440）を，31457280 に変更 |
| | 次のオペランドに 0 を指定できるように変更<br>• システム共通定義<br>ipc_recvbuf_size<br>ipc_sendbuf_size<br>• ユーザサービスデフォルト定義<br>ipc_recvbuf_size<br>ipc_sendbuf_size<br>• ユーザサービス定義<br>ipc_recvbuf_size<br>ipc_sendbuf_size |
| コマンド | ユーザサーバごとに，共有化したバッファの使用サイズを制限できるようにし，scdls コマンドの出力形式を変更 |
| メッセージ | なし |
| その他 | 性能検証用トレースの情報を CSV 形式で出力し，編集できるように変更 |
| | メッセージキューサービス定義の定義ファイル数（メッセージキューサーバ数）を 0～1 に変更 |
| | ネームサービス定義の max_socket_descriptors オペランドの算出式を変更 |
| | 被アーカイブジャーナルノードの不正ジャーナル情報ファイルの最大ファイル数（jnl_dual=N：1,jnl_dual=Y：2）を 1 に変更 |

TP1/Server Base 07-06 でのデフォルト値の変更はありません。

## 付録 C.8　07-05 での変更点

TP1/Server Base 07-05 での関数，定義およびコマンドの追加と削除を次の表に示します。

表 C–17　TP1/Server Base 07-05 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | dc_trn_rm_select |
| | | CBLDCTRN('RMSELECT') |
| | 定義 | システム共通定義<br>• prc_coredump_filter オペランド |
| | | rap リスナーサービス定義<br>• scs_prf_trace_level<br>• rap_extend_function |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | ユーザサービスデフォルト定義<br>• prc_coredump_filter オペランド<br>• rap_extend_function オペランド<br>• trnrmid 定義コマンドの-k オプション |
| | | ユーザサービス定義<br>• prc_coredump_filter オペランド<br>• rap_extend_function オペランド<br>• trnrmid 定義コマンドの-k オプション |
| | コマンド | なし |
| 削除 | なし | |

TP1/Message Control 07-05，TP1/NET/Library 07-05 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–18　TP1/Message Control 07-05，TP1/NET/Library 07-05 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | COMMIT － MHP のコミット |
| | 定義 | MCF マネジャ定義<br>• mcfmuap 定義コマンドの-c オプション<br>commitdml オペランド<br>noansreply オペランド<br>• mcfmuap 定義コマンドの-r オプション<br>reschedulecnt オペランド<br>rescheduleint オペランド<br>reschedulelog オペランド |
| | | MCF アプリケーション定義<br>• mcfaalcap 定義コマンドの-d オプション<br>reschedulecnt オペランド<br>rescheduleint オペランド<br>reschedulelog オペランド |
| | コマンド | mcftlsbuf コマンド<br>• -m オプション<br>• -r オプション |
| | | mcftlsle コマンド<br>• -m オプション<br>• -r オプション |
| 削除 | なし | |

TP1/Server Base 07-05 での動作の変更点はありません。

TP1/Message Control 07-05，TP1/NET/Library 07-05 での動作の変更点を次に示します。

### 表 C–19　TP1/Message Control 07-05，TP1/NET/Library 07-05 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | なし |
| コマンド | なし |
| メッセージ | KFCA10365-I<br>• 出力情報に最大未送信メッセージ数の情報を追加 |
| | KFCA10366-I<br>• 出力情報に最大バッファ使用数の情報を追加 |
| | KFCA10505-I，KFCA10509-I<br>• 出力情報に-m オプションと-r オプションを追加 |

TP1/Server Base 07-05 でのデフォルト値の変更はありません。

TP1/Message Control 07-05，TP1/NET/Library 07-05 でのデフォルト値の変更はありません。

## 付録 C.9　07-04 での変更点

TP1/Server Base 07-04 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–20　TP1/Server Base 07-04 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | チェックポイントダンプサービス定義<br>• jnl_cdskip_limit オペランド<br>• jnl_cdskip_msg オペランド |
| | | プロセスサービス定義<br>• prc_take_over_dlpath オペランド※ |
| | コマンド | trncmt コマンド<br>• -q オプション |
| | | trnrbk コマンド<br>• -q オプション |
| | | trnfgt コマンド<br>• -q オプション |
| | | prcdlpath コマンド※ |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | コマンド | prcdlpathls コマンド※ |
| 削除 | なし | |

注※

ご使用の OS が Windows の場合はサポートしていません。

TP1/Server Base 07-04 での動作の変更点を次の表に示します。

### 表 C–21　TP1/Server Base 07-04 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | なし |
| コマンド | なし |
| メッセージ | KFCA00974-I，KFCA00976-I，KFCA00977-I<br>• 出力情報に-q オプションを追加 |

TP1/Server Base 07-04 でのデフォルト値の変更はありません。

## 付録 C.10　07-03 での変更点

TP1/Server Base 07-03 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–22　TP1/Server Base 07-03 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | システム共通定義<br>• fil_prf_trace_delay_time オペランド<br>• fil_prf_trace_option オペランド<br>• jnl_prf_event_trace_level オペランド<br>• uap_trace_file_put オペランド |
| | | ロックサービス定義<br>• lck_prf_trace_level オペランド |
| | | ネームサービス定義<br>• name_cache_validity_time オペランド |
| | | rap リスナーサービス定義<br>• ipc_sockctl_highwater<br>• ipc_sockctl_watchtime |
| | | JNL 性能検証用トレース定義 |

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | LCK 性能検証用トレース定義 |
| | | リアルタイム取得項目定義<br>• rts_mcf_ap_scd_stay オペランド<br>• rts_mcf_ap_usr_srvc オペランド<br>• rts_mcf_in_msg_scd_wait オペランド<br>• rts_mcf_out_msg_norm_scd_wait オペランド<br>• rts_mcf_out_msg_prio_scd_wait オペランド<br>• rts_mcf_out_msg_resp_scd_wait オペランド<br>• rts_mcf_out_msg_sync_scd_wait オペランド<br>• rts_mcf_que_scd_wait_num オペランド |
| | | ユーザサービスデフォルト定義<br>• uap_trace_file_put オペランド |
| | | ユーザサービス定義<br>• uap_trace_file_put オペランド |
| | コマンド | filstatfs コマンド<br>• -S オプション |
| | | prctctrl コマンド |
| | | prfed コマンド<br>• -v オプション |
| | | prfget コマンド<br>• -f オプションの指定値に_mc，_fl，_jl，_lk を追加 |
| | | uatdump コマンド<br>• -f オプション |
| 削除 | なし | |

TP1/Message Control 07-03，TP1/NET/Library 07-04 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–23　TP1/Message Control 07-03，TP1/NET/Library 07-04 での関数，定義およびコマンドの追加と削除

| 種類 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | なし |
| | コマンド | mcfalstap コマンド |
| | | mcftlsutm コマンド |
| 削除 | なし | |

TP1/Server Base 07-03 での動作の変更点を次の表に示します。

## 表 C–24　TP1/Server Base 07-03 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | システム共通定義<br>• all_node オペランドにサービス情報優先度指定機能を指定できるように変更<br>• name_domain_file_use オペランドで使用有無を決定するドメイン定義ファイルに，優先選択ノードの定義ファイルを追加 |
| | 性能検証用トレース取得サービスの稼働数，および関連する定義を変更 |
| コマンド | OpenTP1 の標準出力，標準エラー出力をリダイレクトする prctee プロセスを停止・再開始できるよう変更 |
| | 次のコマンドの出力形式を変更<br>• namsvinf |
| | OpenTP1 ファイルシステムのユーザ領域情報として使用中領域と未使用領域（空き領域）の一覧を表示できるよう変更 |
| | トレース情報ファイルの編集結果を csv 形式で出力できるように変更 |
| メッセージ | KFCA26954-W，KFCA26956-W，KFCA26965-E，KFCA27790-W<br>• 出力情報に，送信元 IP アドレスと送信元ポート番号の情報を追加 |
| その他 | 次に示すイベントトレース情報を取得するように変更<br>• FIL イベントトレース<br>• JNL 性能検証用トレース<br>• LCK 性能検証用トレース |
| | プロセスをアボートしなくても UAP トレースを取得できるように変更 |
| | 静的共用メモリの算出式を変更 |
| | 共用メモリプールのサイズの算出式を変更 |
| | 監視イベントに次のイベントを追加<br>• OpenTP1 サービス開始<br>• OpenTP1 サービス停止 |
| | UNIX のメッセージ送受信関数で使用する資源の見積もり式で，次の資源の被アーカイブノードの見積もり式を変更<br>• メッセージ ID<br>• OpenTP1 のすべてのメッセージのうちの最大待ち合わせメッセージ数 |

TP1/Message Control 07-03，TP1/NET/Library 07-04 での動作の変更点を次の表に示します。

## 表 C–25　TP1/Message Control 07-03，TP1/NET/Library 07-04 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | mcftalcle 定義コマンドの-m オプションに 0 を指定，または指定を省略した場合の解釈値を，無制限から 65535 に変更 |

| 分類 | 内容 |
|---|---|
| 定義 | 次の定義コマンドで，-j オプションの指定値の上限を 131072 から 4000000 に変更<br>• mcfmcomn<br>• mcfmuap<br>• mcftcomn |
| その他 | MCF ダンプファイルの出力上限数を 99 から 3 に変更 |
| | アプリケーションに関するタイマ起動要求の状態を表示できるように変更 |
| | ユーザタイマ監視の状態を表示できるように変更 |

TP1/Server Base 07-03 でのデフォルト値の変更点を次の表に示します。

### 表 C–26　TP1/Server Base 07-03 でのデフォルト値の変更

| 分類 | 内容 |
|---|---|
| 定義 | トランザクションサービス定義<br>• thread_stack_size オペランドについて，AIX 版の uCosminexus TP1/Server Base(64)を使用している場合のデフォルト値を 65536 に変更 |
| | TAM サービス定義<br>• tam_pool_attri オペランドのデフォルト値を次のとおり変更<br>・HP-UX または Solaris の場合：fixed<br>・AIX，Linux または Windows の場合：free |

TP1/Message Control 07-03，TP1/NET/Library 07-04 での関数，定義およびコマンドのデフォルト値の変更はありません。

## 付録 C.11　07-02 での変更点

TP1/Server Base 07-02 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–27　TP1/Server Base 07-02 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | dc_log_audit_print |
| | | CBLDCADT('PRINT') |
| | 定義 | システム共通定義<br>• nam_prf_trace_level オペランド<br>• jnl_fileless_option オペランド |
| | | XA リソースサービス定義<br>• xar_prf_trace_level オペランド |
| | | ジャーナルサービス定義 |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | • jnl_watch_time オペランド |
| | | システムジャーナルサービス定義<br>• jnl_max_file_dispersion オペランド<br>• jnl_min_file_dispersion オペランド<br>• jnladdpf 定義コマンドの-e オプション |
| | | ログサービス定義<br>• log_audit_out オペランド<br>• log_audit_path オペランド<br>• log_audit_size オペランド<br>• log_audit_count オペランド<br>• log_audit_message オペランド |
| | | rap リスナーサービス定義<br>• rap_term_disconnect_time オペランド<br>• rap_stay_watch_time オペランド<br>• rap_stay_warning_interval オペランド<br>• log_audit_out_suppress オペランド<br>• log_audit_message オペランド<br>• watch_time オペランド |
| | | rap クライアントマネジャサービス定義<br>• log_audit_out_suppress オペランド<br>• log_audit_message オペランド |
| | | 性能検証用トレース定義<br>• prf_trace_backup オペランド |
| | | XAR 性能検証用トレース定義 |
| | | リアルタイム統計情報サービス定義<br>• rts_log_file_backup オペランド |
| | | リアルタイム取得項目定義<br>• rts_scd_svc_scd_wait オペランド<br>• rts_scd_svc_using_buf オペランド<br>• rts_scd_parallel オペランド |
| | | ユーザサービスデフォルト定義<br>• log_audit_out_suppress オペランド<br>• log_audit_message オペランド<br>• scdsvcdef 定義コマンド |
| | | ユーザサービス定義<br>• log_audit_out_suppress オペランド<br>• log_audit_message オペランド<br>• service オペランドの指定値に「UAP 共用ライブラリ名」を追加<br>• scdsvcdef 定義コマンド |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | コマンド | dcauditsetup コマンド |
| | | dcsetup コマンド<br>• -j オプション |
| | | dcstart コマンド<br>• -b オプション |
| | | dcstop コマンド<br>• -q オプション |
| | | prfget コマンド<br>• -f オプションの指定値に_nm，_xr，_pr を追加 |
| | | scdls コマンド<br>• -ae オプション<br>• -e オプション<br>• -t オプション |
| 削除 | 定義 | ユーザサービスデフォルト定義<br>• thdlock_sleep_time オペランド |
| | | ユーザサービス定義<br>• thdlock_sleep_time オペランド |

TP1/Message Control 07-02，TP1/NET/Library 07-03 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–28　TP1/Message Control 07-02，TP1/NET/Library 07-03 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | dc_mcf_adltap |
| | | dc_mcf_tactcn |
| | | dc_mcf_tactle |
| | | dc_mcf_tdctcn |
| | | dc_mcf_tdctle |
| | | dc_mcf_tdlqle |
| | | dc_mcf_tlscn |
| | | dc_mcf_tlscom |
| | | dc_mcf_tlsle |
| | | dc_mcf_tlsln |
| | | dc_mcf_tofln |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | dc_mcf_tonln |
| | | CBLDCMCF('ADLTAP ') |
| | | CBLDCMCF('TACTCN ') |
| | | CBLDCMCF('TACTLE ') |
| | | CBLDCMCF('TDCTCN ') |
| | | CBLDCMCF('TDCTLE ') |
| | | CBLDCMCF('TDLQLE ') |
| | | CBLDCMCF('TLSCOM ') |
| | | CBLDCMCF('TLSCN ') |
| | | CBLDCMCF('TLSLE ') |
| | | CBLDCMCF('TLSLN ') |
| | | CBLDCMCF('TOFLN ') |
| | | CBLDCMCF('TONLN ') |
| | 定義 | MCF マネジャ共通定義<br>• mcfmcomn 定義コマンドの-i オプション |
| | コマンド | mcftlssg コマンド<br>• -m オプション |
| | | mcftlscom コマンド<br>• -w オプション<br>• -t オプション |
| 削除 | なし | |

TP1/Server Base 07-02 での動作の変更点を次の表に示します。

## 表 C–29　TP1/Server Base 07-02 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | システム共通定義，およびシステムサービス共通情報定義<br>• thdlock_sleep_time オペランドの最小値（15）を 1 に変更 |
| | プロセスサービス定義<br>• prc_prf_trace オペランドが Y（デフォルト）の場合に取得されるイベントを追加 |
| | トランザクションサービス定義<br>• trn_tran_process_count オペランドの指定値の範囲（1～8192）に，MCF サービスを使用する場合の指定範囲（32 ビットの場合 1～7484，64 ビットの場合 1～6893）を追加 |
| コマンド | dcdefchk コマンド |

| 分類 | 内容 |
|---|---|
| コマンド | • log_syslog_elist オペランド，および log_syslog_elist_rint オペランドの論理チェック（LOG-0011）を変更<br>• DCSYSLOGCTYPE の論理チェック（LOG-0011）を追加<br>• 監査ログ関連オペランドの論理チェック（LOG-0013 および LOG-0014）を追加 |
| | dcrasget コマンド<br>• Linux の場合，-c オプション指定時に取得するファイルの拡張子を.Z から.gz に変更 |
| | ジャーナルファイルレスモードで次に示すコマンドを実行したとき，KFCA01141-E メッセージを出力するように変更<br>• jnlls<br>• jnlunlfg<br>• jnlchgfg<br>• jnlopnfg<br>• jnlclsfg<br>• jnlswpfg<br>• jnlrinf<br>• jnlarls<br>• jnlatunl<br>• jnladdpf<br>• jnldelpf |
| | usmdump コマンド<br>• Linux の場合，共用メモリダンプファイルの拡張子を.Z から.gz に変更 |
| その他 | damrstr コマンドに指定したバックアップファイルのサイズをチェックして，不正なバックアップファイルを検知した場合は，KFCA02512-E メッセージを出力するように変更 |
| | tamrstr コマンドに指定したリストア元ファイルのサイズをチェックして，不正なリストア元ファイルを検知した場合は，KFCA26209-E メッセージを出力するように変更 |
| | Linux（IPF）の場合，syslog メッセージの出力に失敗した場合にリトライできるように変更 |
| | Linux の場合，共用メモリダンプファイルの拡張子を.Z から.gz に変更 |
| | 次に示すイベントトレース情報を取得するように変更<br>• XAR 性能検証用トレース<br>• NAM イベントトレース<br>• プロセスサービスのイベントトレース |
| | rap クライアントからの要求が，rap サーバ割り当て待ちで rap リスナーサービス定義の rap_stay_warning_interval オペランドの指定値を超えて滞留した場合に，KFCA27764-W メッセージを出力するように変更 |
| | リアルタイム統計情報サービスの開始時に，RTS ログファイルのバックアップファイルを作成するように変更 |

TP1/Message Control 07-02，TP1/NET/Library 07-03 での動作の変更点を次の表に示します。

### 表 C–30　TP1/Message Control 07-02，TP1/NET/Library 07-03 での動作の変更

| 分類 | 内容 |
|---|---|
| その他 | OpenTP1 終了時，一定時間が経過しても入力キューにメッセージが残っている場合，サービスグループごとに KFCA16532-I メッセージを出力し，残っているメッセージの処理が完了したときに KFCA16533-I メッセージを出力するように変更 |
| | OpenTP1 終了時，一定時間が経過しても出力キューにメッセージが残っている場合，論理端末ごとに KFCA16534-I メッセージを出力し，残っているメッセージの処理が完了したときに KFCA16535-I メッセージを出力するように変更 |
| | OpenTP1 終了時，一定時間が経過してもプロトコル制御の終了準備を待ち合わせている場合，MCF 通信サービスごとに KFCA16536-I メッセージを出力し，プロトコル制御の終了準備処理が完了したときに KFCA16537-I メッセージを出力するように変更 |
| | MHP サービス関数動的ローディング機能を使用できるように変更 |
| | キューを監視している場合，および相手システムからの応答メッセージ受信を待ち合わせている場合に，監視処理中および処理完了時にログメッセージを出力することで，処理状況を把握できるように変更 |
| | 次の操作を，ライブラリ関数でできるよう変更<br>• コネクションの状態表示，確立，および解放<br>• サーバ型コネクションの確立要求の状態表示，および受付開始・終了<br>• アプリケーションに関するタイマ起動要求の削除<br>• 論理端末の状態表示，閉塞，閉塞解除，および出力キューの削除<br>• MCF 通信サービスの状態取得 |
| | 最大未処理受信メッセージ数を表示できるように変更 |

TP1/Server Base 07-02 でのデフォルト値の変更点を次の表に示します。

### 表 C–31　TP1/Server Base 07-02 でのデフォルト値の変更

| 分類 | 内容 |
|---|---|
| 定義 | システム共通定義<br>• client_uid_check オペランドのデフォルト値（Y）を AIX，Linux または Solaris の場合は N に変更（HP-UX または Windows の場合は Y） |

TP1/Message Control 07-02，TP1/NET/Library 07-03 での関数，定義およびコマンドのデフォルト値の変更はありません。

## 付録 C.12　07-01 での変更点

TP1/Server Base 07-01 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–32　TP1/Server Base 07-01 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | ネームサービス定義<br>• name_nodeid_check_message オペランド |
| | | XA リソースサービス定義<br>• xar_msdtc_use オペランド |
| | コマンド | dcpplist コマンド |
| | | dcdefchk コマンド<br>• -l オプション<br>• -c オプション<br>• -w オプション<br>• -e オプション |
| | | xarinit コマンド<br>• -s オプション |
| | | xarls コマンド<br>• -r オプション |
| 削除 | なし | |

TP1/Message Control 07-01，TP1/NET/Library 07-01 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–33　TP1/Message Control 07-01，TP1/NET/Library 07-01 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |
| | 定義 | ユーザサービス定義<br>• mcf_prf_trace オペランド |
| | | MCF マネジャ定義<br>• mcfmsvg 定義コマンド |
| | | MCF アプリケーション定義<br>• mcfaalcap 定義コマンドの-N オプションの modelname オペランド |
| | | MCF 性能検証用トレース定義<br>• prf_file_size オペランド<br>• prf_file_count オペランド |
| | | システムサービス情報定義<br>• mcf_prf_trace オペランド |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | システムサービス共通情報定義<br>• mcf_prf_trace_level オペランド<br>• DCMCFQUEBAK オペランド |
| | コマンド | mcftlsln コマンド |
| | | mcftofln コマンド |
| | | mcftonln コマンド |
| 削除 | なし | |

TP1/Server Base 07-01 での動作の変更点を次の表に示します。

## 表 C–34　TP1/Server Base 07-01 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | リアルタイム取得項目定義<br>• rts_jnl_read オペランドで取得する情報（「回数」）を，「回数，最大時間，最小時間，平均時間」に変更<br>• rts_jnl_write オペランドで取得する情報（「回数」）を，「回数，最大時間，最小時間，平均時間」に変更 |
| コマンド | xarevtr コマンド<br>• .NET Framework からの接続に対して「アプリケーションサーバ名称」を表示<br>•「アプリケーションサーバ XID 情報」の表示（GTRID の先頭 28 バイト以内）を GTRID の先頭 64 バイト以内に拡大<br>•「アプリケーションサーバ XID 情報」の表示（BQUAL の先頭 28 バイト以内）を BQUAL の先頭 64 バイト以内に拡大 |
| | xarforce コマンド<br>• -t オプションに指定する「OpenTP1 トランザクション ID」の最大文字数（56 文字）を 80 文字に拡大<br>• -u オプションに指定する「クライアントトランザクション ID」の最大文字数（256 文字）を 280 文字に拡大 |
| | xarls コマンド<br>• 表示する「OpenTP1 トランザクション ID」の最大文字数（56 文字）を 80 文字に拡大<br>• 表示する「クライアントトランザクション ID」の最大文字数（256 文字）を 280 文字に拡大 |
| | rtsedit コマンド<br>• -m オプションの出力結果に RTS ログファイルのバージョン情報を追加 |
| | xarfills コマンド<br>• 出力結果に「格納可能 RI 長」を追加 |
| | xarls コマンド<br>• -c オプションの出力結果に「DID 情報」,「ノード ID 情報」および「MSDTC 連携で使用する回復情報」を追加 |

| 分類 | 内容 |
|---|---|
| その他 | rap リスナーおよび RTSSPP（リアルタイム統計情報サービスの拡張機能）に対して，次に示すコマンドが有効にならないように変更<br>scdchprc，scdhold，scdrles，scdrsprc |
| | XA リソースサービスを使用した RPC コールにデータ圧縮機能を追加 |

TP1/Message Control 07-01，TP1/NET/Library 07-01 での動作の変更点を次の表に示します。

### 表 C–35　TP1/Message Control 07-01，TP1/NET/Library 07-01 での動作の変更

| 分類 | 内容 |
|---|---|
| その他 | 次の指定をしたときに，MCF 性能検証用トレース情報を取得するように変更<br>• システム共通定義の prf_trace オペランドに Y を指定，または指定を省略<br>• システムサービス共通情報定義の mcf_prf_trace_level オペランドに 00000001 を指定 |
| | MCF 通信プロセス識別子を表示できるように変更 |
| | サーバ型コネクションの確立要求の受付開始・終了を，手動でできるように変更 |

TP1/Server Base 07-01，TP1/Message Control 07-01，および TP1/NET/Library 07-01 での関数，定義およびコマンドのデフォルト値の変更はありません。

## 付録 C.13　07-00 での変更点

TP1/Server Base 07-00 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C–36　TP1/Server Base 07-00 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | dc_rts_utrace_put |
| | | CBLDCRTS('RTSPUT ') |
| | 定義 | システム環境定義<br>• user_command_online_tp1mngr_id オペランド※1 |
| | | システム共通定義<br>• dcstart_wakeup_retry_count オペランド<br>• ipc_listen_sockbufset オペランド |
| | | ユーザサービスデフォルト定義<br>• ipc_listen_sockbufset オペランド |
| | | ユーザサービス定義<br>• ipc_listen_sockbufset オペランド |
| | | ネームサービス定義 |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | • name_audit_conf オペランドの指定可能値に 2 を追加<br>• name_audit_watch_time オペランド<br>• name_rpc_control_list オペランド |
| | | トランザクションサービス定義<br>• trn_completion_limit_time オペランド<br>• trn_rcv_open_close_scope オペランド<br>• trnstring 定義コマンドの-s オプション |
| | | rap リスナーサービス定義<br>• rap_message_id_change_level オペランド |
| | | リアルタイム統計情報サービス定義 |
| | | リアルタイム取得項目定義 |
| | コマンド | dcdefchk コマンド |
| | | dcsvgdef コマンド<br>• -t オプション |
| | | namblad コマンド |
| | | ntbtail コマンド※2 |
| | | rtsedit コマンド |
| | | rtsls コマンド |
| | | rtssetup コマンド |
| | | rtsstats コマンド |
| | | tp1console コマンド※2 |
| 削除 | なし | |

注※1

ご使用の OS が Linux または Windows の場合はサポートしていません。

注※2

ご使用の OS が Windows 07-00-03 以降の場合だけサポートしています。機能の詳細については，マニュアル「OpenTP1 使用の手引 Windows(R)編」を参照してください。

TP1/Message Control 07-00，TP1/NET/Library 07-00 での関数，定義およびコマンドの追加と削除を次の表に示します。

### 表 C-37　TP1/Message Control 07-00，TP1/NET/Library 07-00 での関数，定義およびコマンドの追加と削除

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 関数 | なし |

| 種別 | 分類 | 内容 |
|---|---|---|
| 追加 | 定義 | MCF マネジャ定義<br>• mcfmuap 定義コマンドの-c オプションの order オペランド |
| | | 環境変数 DCMCFCMDLOG |
| | コマンド | なし |
| 削除 | 関数 | なし |
| | 定義 | システムサービス構成定義<br>• mrs_conf オペランド |
| | | MCF マネジャ定義<br>• mcfmcomn 定義コマンドの-r オプション<br>• mcfmcomn 定義コマンドの-t オプションの delayed オペランド<br>• mcfmcomn 定義コマンドの-c オプション<br>• mcfmrclnt 定義コマンド<br>• mcfmrerun 定義コマンド |
| | | MCF アプリケーション定義<br>• mcfaalcap 定義コマンドの-g オプションの type オペランド |
| | | リモート MCF マネジャ定義<br>• mcfrcomn 定義コマンド<br>• mcfrserv 定義コマンド |
| | | システムサービス情報定義<br>• critical オペランド |
| | コマンド | なし |

TP1/Server Base 07-00 での動作の変更点を次の表に示します。

## 表 C–38　TP1/Server Base 07-00 での動作の変更

| 分類 | 内容 |
|---|---|
| 関数 | メッセージログファイルにメッセージログとして出力する文字列の長さの上限値（222 バイト）を，Linux の場合は 444 バイトに変更した。<br>• -dc_logprint<br>• CBLDCLOG('PRINT ') |
| 定義 | システム環境定義<br>• server_count オペランドの最大値（2048）を 4096 に拡大※1 |
| コマンド | jnlcolc コマンド，jnlcopy コマンド，jnledit コマンド，jnlrput コマンドおよび jnlsort コマンド<br>• 指定できるアンロードファイルの数（64 個）を 256 個に拡大 |
| | jnledit コマンド<br>• 出力結果に「MCF レコード作成日時」および「トランザクションブランチ開始日時」を追加 |
| | prctee コマンド |

| 分類 | 内容 |
|---|---|
| コマンド | • メッセージの出力先（/tmp）を$DCDIR/spool に変更※2<br>• エラーメッセージ出力先（/tmp/betran.log）を$DCDIR/spool/.prctee.log に変更※2 |
| | xarevtr コマンド<br>• 出力結果に，Cosminexus V6.5 以降からの接続に対して「アプリケーションサーバ名称」を表示 |
| メッセージ | KFCA00107-E<br>• 詳細情報を，モジュール名称から OpenTP1 ファイルシステム名に変更 |
| | KFCA00854-E<br>•「格納できなかったメッセージサイズ」の情報を追加 |
| その他 | 指定可能なホスト名長（63 文字）を 255 文字に拡大 |
| | JP1/AJS2 - Scenario Operation 用サンプルシナリオテンプレートのコマンド優先度（「なし」）を「3」に変更 |
| | UNIX ドメイン通信ファイルのアクセスモードの設定（755）を 777 に変更※2 |
| | 日本語 UTF-8 メッセージ（LANG ja_JP.UTF-8）に対応※2 |

注※1

ご使用の OS が Linux の場合はサポートしていません。

注※2

ご使用の OS が Linux の場合だけサポートしています。

TP1/Message Control 07-00，TP1/NET/Library 07-00 での動作の変更点を次の表に示します。

### 表 C–39　TP1/Message Control 07-00，TP1/NET/Library 07-00 での動作の変更

| 分類 | 内容 |
|---|---|
| 定義 | MCF 通信構成定義<br>• mcftenv 定義コマンドの-a オプションの構文要素（〈1～8 文字の識別子〉）を〈1～8 文字の英数字〉に変更 |
| コマンド | 次に示すコマンドに，コマンドログ取得機能を追加<br>mcfaactap，mcfaclcap，mcfadctap，mcfadltap，mcfstats，mcftactcn，mcftactle，mcftactmj，mcftactsg，mcftactsv，mcftchcn，mcftdctcn，mcftdctle，mcftdctmj，mcftdctsg，mcftdctsv，mcftdlqle，mcftdlqsg，mcftdmpqu，mcfthldiq，mcfthldoq，mcftrlsiq，mcftrlsoq，mcftspqle，mcftstart，mcftstop，mcfuevt |

TP1/Server Base 07-00 での関数，定義およびコマンドのデフォルト値の変更はありません。

TP1/Message Control 07-00，TP1/NET/Library 07-00 でのデフォルト値の変更点を次の表に示します。

表 C-40　TP1/Message Control 07-00，TP1/NET/Library 07-00 でのデフォルト値の変更

| 分類 | 内容 |
|---|---|
| 定義 | MCF 通信構成定義<br>• mcfttrc 定義コマンドの disk オペランドのデフォルト値（no）を yes に変更 |

# 付録 D　リモートプロシジャコールの処理の概要

ここでは，使用する機能や状況別に，リモートプロシジャコールの処理の概要について説明します。ここで説明する処理の概要は，すべてキュー受信型のリモートプロシジャコールについての説明です。

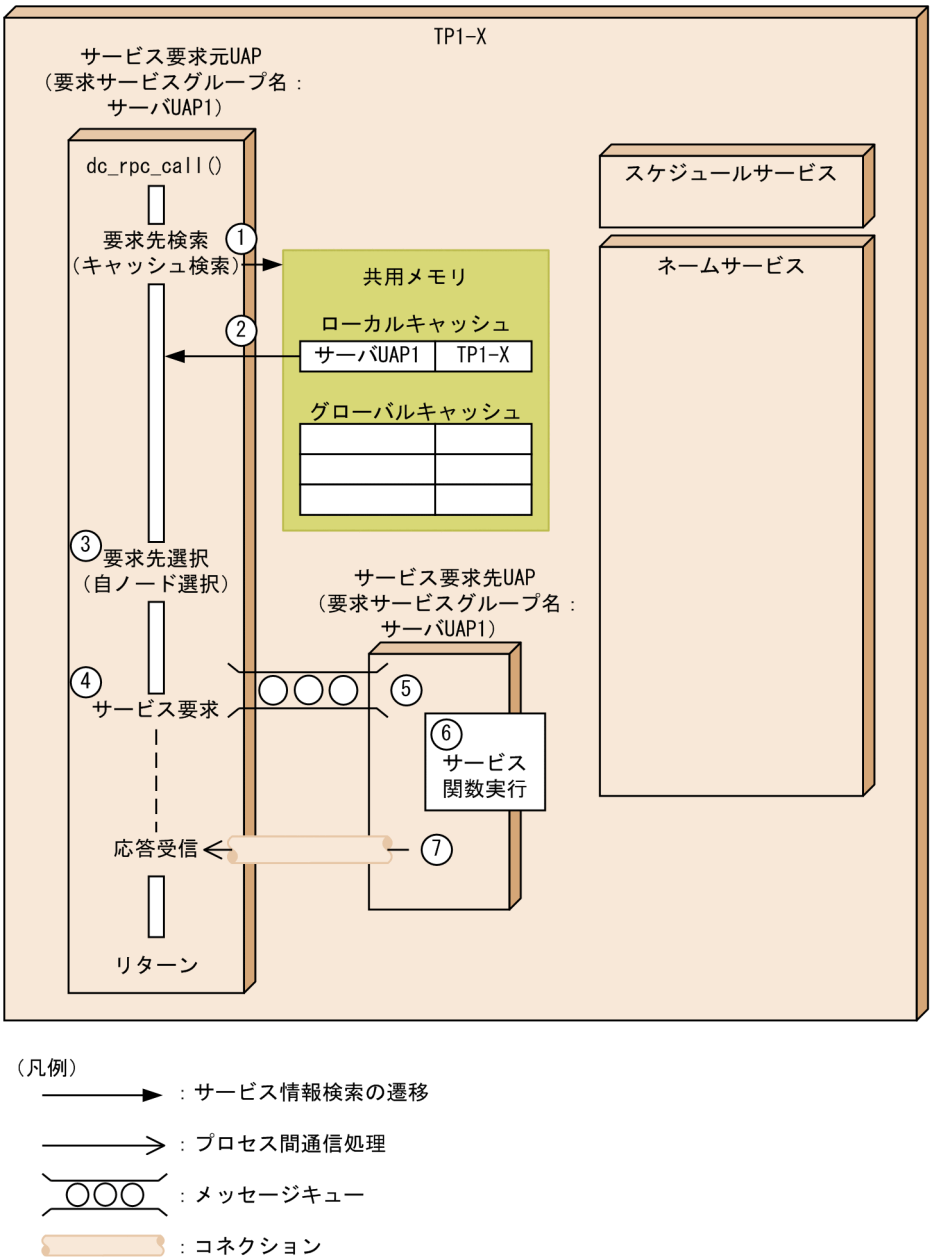
なお，説明中のローカルキャッシュとは，ネームサービスが自 OpenTP1 で起動しているサーバのサービス情報を管理する領域を指します。また，グローバルキャッシュとは，ネームサービスが他ノードで起動しているサーバのサービス情報を管理する領域を指します。

アプリケーションプログラムの作成方法については，マニュアル「OpenTP1 プログラム作成の手引」，説明中の関数の文法については，マニュアル「OpenTP1 プログラム作成リファレンス C 言語編」，またはマニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」を参照してください。

## 付録 D.1　自ノードへのリモートプロシジャコールの処理の概要

自ノードへのリモートプロシジャコールの処理の概要について，次の図に示します。

図 D-1　自ノードへのリモートプロシジャコールの処理の概要




図で示した TP1-X の処理の流れについて，次に説明します。番号は，図中の番号と対応しています。

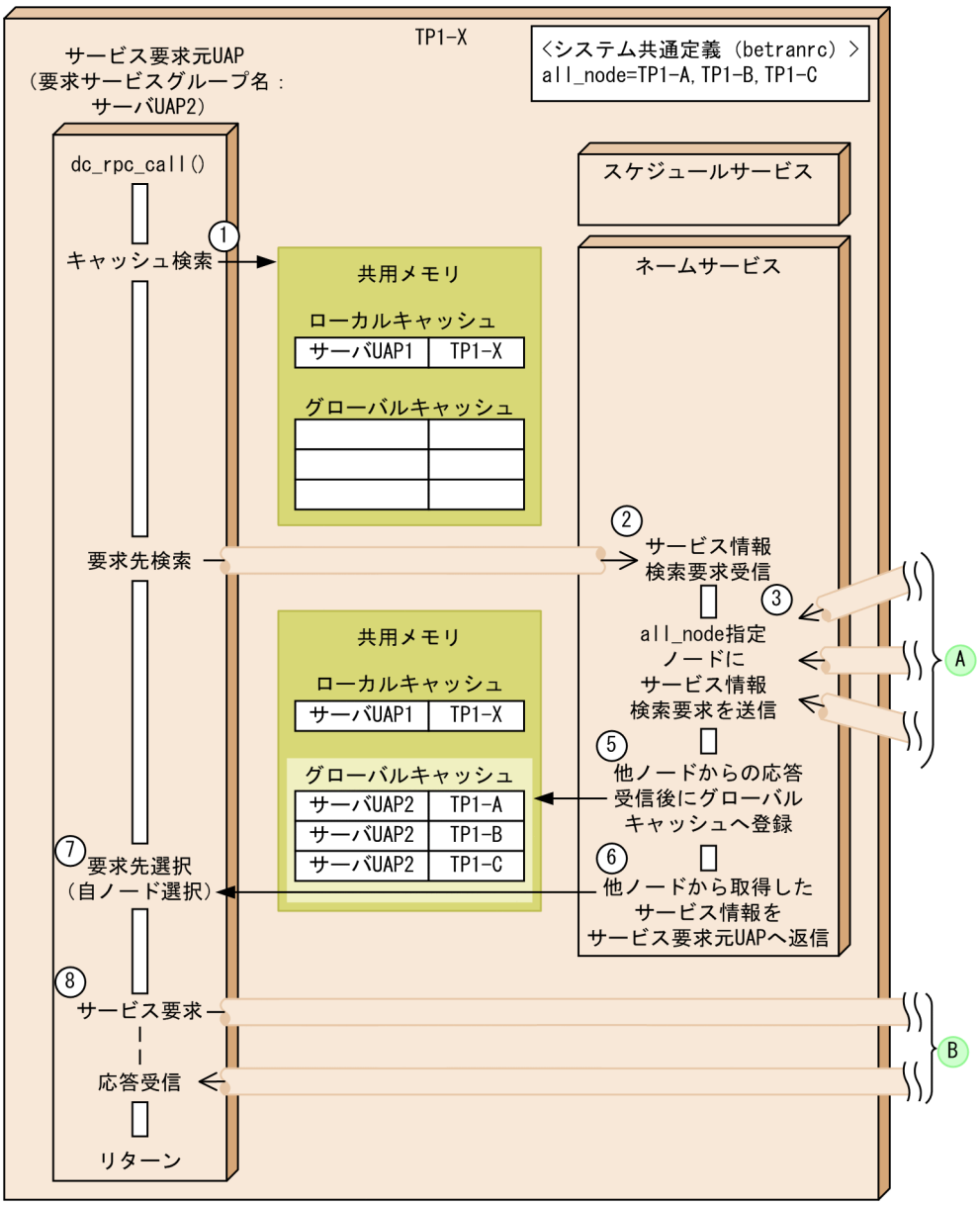
1. dc_rpc_call 関数の引数で指定されたサービスグループ名を検索キーに，該当するサービス情報がローカルキャッシュとグローバルキャッシュに登録されているかを確認します。
2. 該当するサービス情報がローカルキャッシュにだけ登録されていたため，ローカルキャッシュに登録されているサービス情報だけをサービス要求元 UAP へ返します。
3. 自ノードで起動している UAP のサービス情報しか取得できなかったため，TP1-X は自ノードで起動しているサービス要求先 UAP（サーバ UAP1）を選択します。
4. 自 OpenTP1 内の UAP へのサービス要求の場合，スケジュールサービスを経由しないで，サービス要求先 UAP（サーバ UAP1）のメッセージキューに対して，直接サービス要求を登録します。

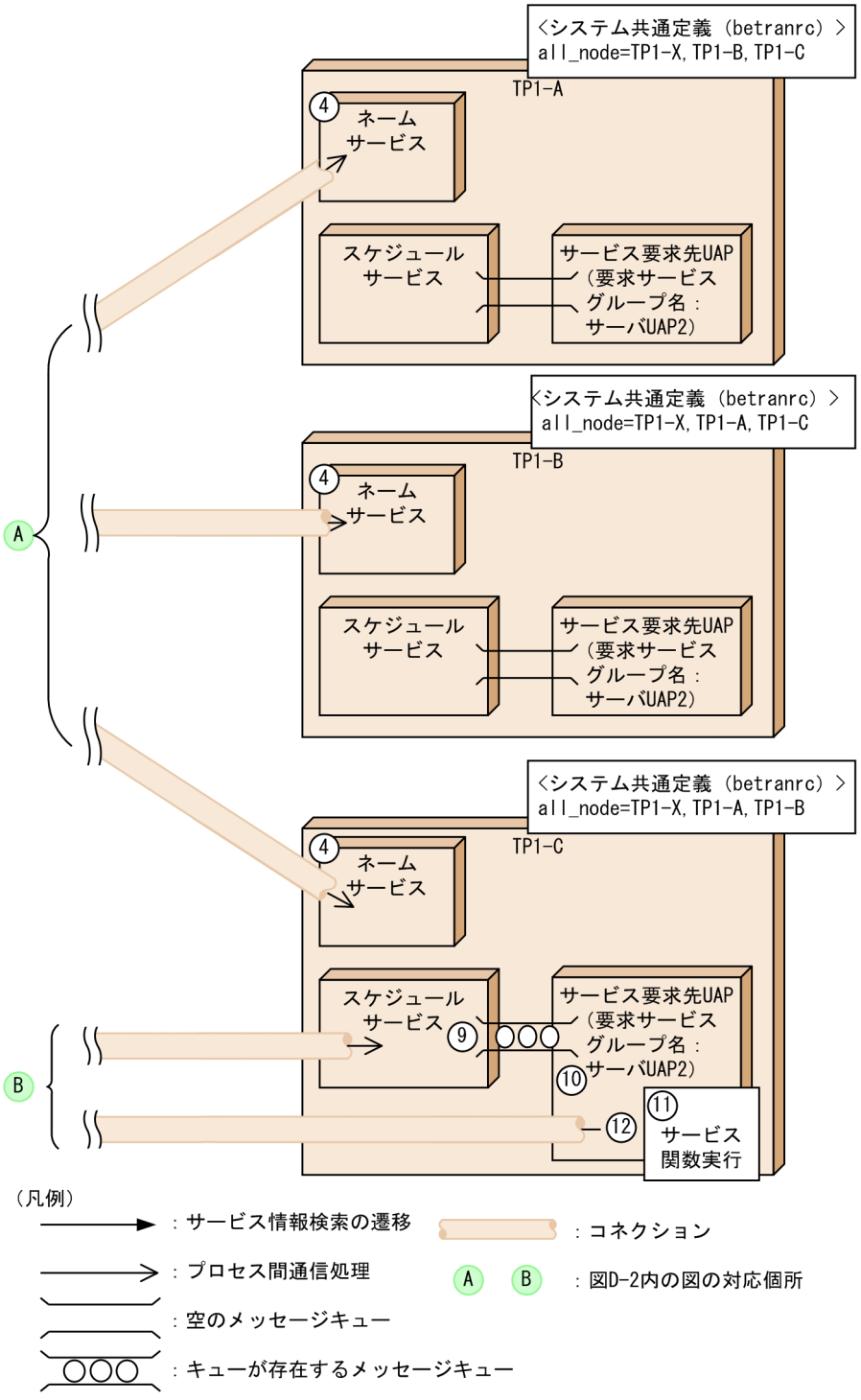
5. サービス要求が登録されたサービス要求先 UAP（サーバ UAP1）では，メッセージキューからサービス要求を取り出します。
6. サービス関数を実行します。
7. サービス関数の実行後，サービス要求を行ったサービス要求元 UAP に対して，サービス要求先 UAP（サーバ UAP1）から，直接応答メッセージを送信します。

## 付録 D.2　他ノードへのリモートプロシジャコールの処理の概要

他ノードへのリモートプロシジャコールの処理の概要について，次の図に示します。

図 D-2　他ノードへのリモートプロシジャコールの処理の概要






図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.～3.および5.～8.はTP1-Xの処理，4.はTP1-A，TP1-B，TP1-Cの処理，9.～12.はTP1-Cの処理についての説明です。

1. dc_rpc_call関数の引数に指定されたサービスグループ名を検索キーに，該当するサービス情報がローカルキャッシュとグローバルキャッシュに登録されているかを確認します。
2. 該当するサービス情報が存在しなければ，ネームサービスにサービス情報の検索要求を送信します。

該当するサービス情報がローカルキャッシュ，グローバルキャッシュに存在しても，次の条件を満たす場合は，グローバルキャッシュに存在するサービス情報をいったん削除したあと，ネームサービスへサービス情報の検索要求を送信します。

- 該当するサービス情報がグローバルキャッシュに存在し，該当するサービス情報の有効時間※1 が経過した場合。
- サービス要求先 UAP が自ノードで起動したあと，該当するサービスに対して初めて行ったサービス情報の検索要求だった場合。

3. サービス情報の検索要求を受信した TP1-X のネームサービスは，システム共通定義の all_node オペランドに指定されたノード（TP1-A，TP1-B，TP1-C）に対してサービス情報の検索要求を送信します。

   ただし，all_node オペランドを指定していない場合は，サービス情報の検索要求は送信しません。また，all_node オペランドに指定されたサービス要求先が RPC 抑止リスト※2 に登録されていた場合，該当するノードに対してサービス情報の検索要求は送信しません。
4. サービス情報の検索要求を受信した TP1-A，TP1-B，TP1-C のネームサービスでは，ローカルキャッシュに該当するサービスグループが存在するか検索し，その結果を TP1-X のネームサービスに送信します。
5. 他ノードに対するサービス要求の応答が返ってきたため，TP1-X のネームサービスは，受信したサービス情報をグローバルキャッシュに登録します。
6. TP1-X のネームサービスは，検索して取得したサービス情報を，サービス要求元 UAP に返します。
7. 複数の同一サービスグループ名称の UAP が起動されていた場合は，TP1 の内部処理によって，その中から一つのサービス要求先 UAP（サーバ UAP2）を選択します。この説明では，TP1-C をあて先に選択すると仮定します。
8. 7.で選択した TP1-C のスケジュールサービスに対して，サービス実行要求を送信します。
9. サービス要求を受信したスケジュールサービスでは，該当するサービスグループのサービス要求先 UAP（サーバ UAP2）のメッセージキューに対し，サービス要求を登録します。
10. サービス要求が登録されたサービス要求先 UAP（サーバ UAP2）では，メッセージキューからサービス要求を取り出します。
11. サービス関数を実行します。
12. サービス関数の実行後，サービス要求元 UAP に対して，サービス要求先 UAP（サーバ UAP2）から，直接応答メッセージを送信します。

注※1

サービス情報を取得した時間からネームサービス定義の name_cache_validity_time オペランドで指定した値までの時間を指します。

name_cache_validity_time オペランドについては，マニュアル「OpenTP1 システム定義」を参照してください。

注※2

未起動ノードの情報を保持しているリストを指します。

他ノードのネームサーバに対する通信に失敗した場合，RPC 抑止リストに登録されます。RPC 抑止リストにノードが登録されると，該当するノードのサービス情報はグローバルキャッシュからすべて削除されます。ノード監視機能を使用し，停止状態だったノードが稼働状態になったと判断した場合，該当するノードの情報を RPC 抑止リストから削除します。

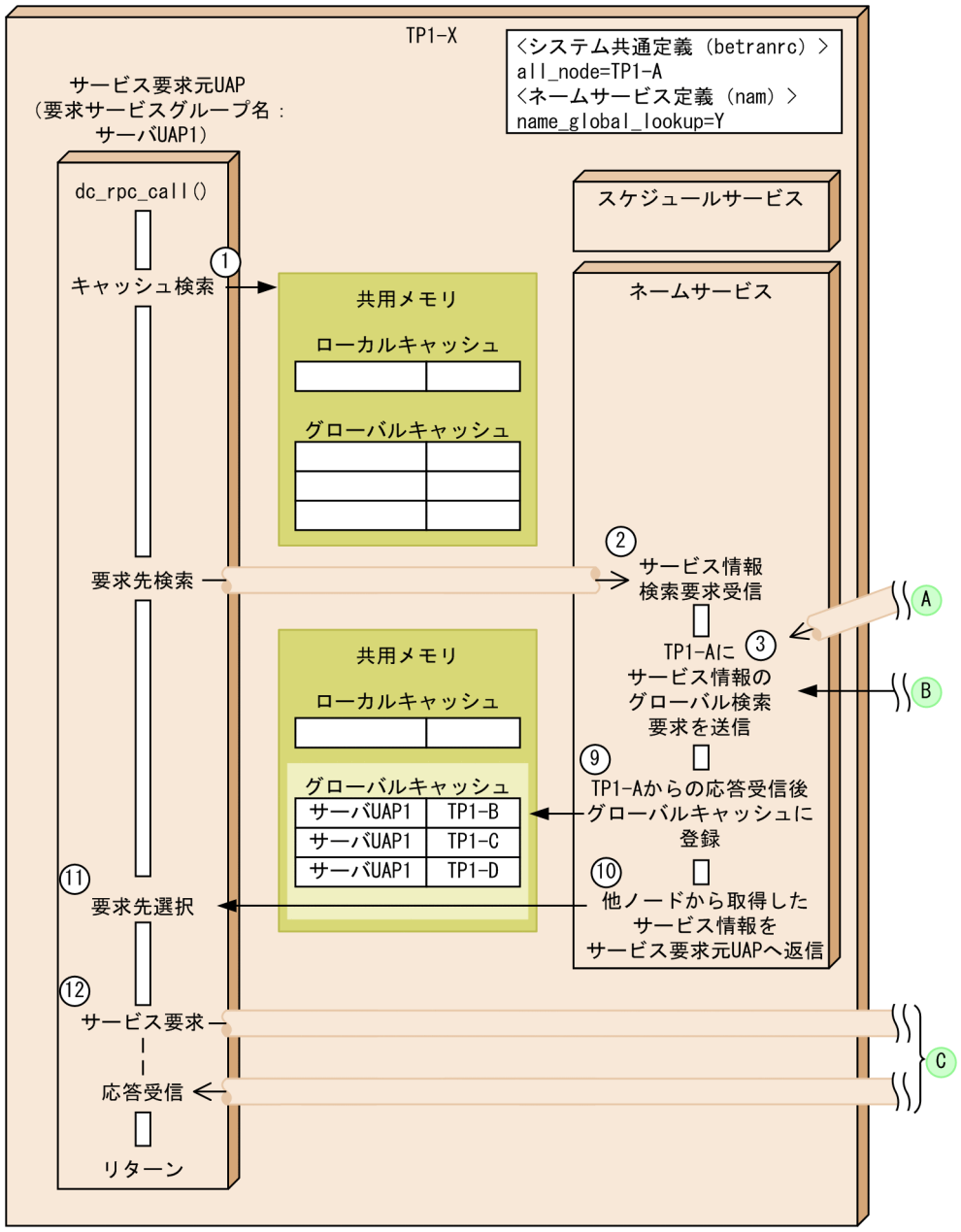
詳細については，「3.2.3(3)　RPC 抑止リストに登録されたノードの監視機能」を参照してください。

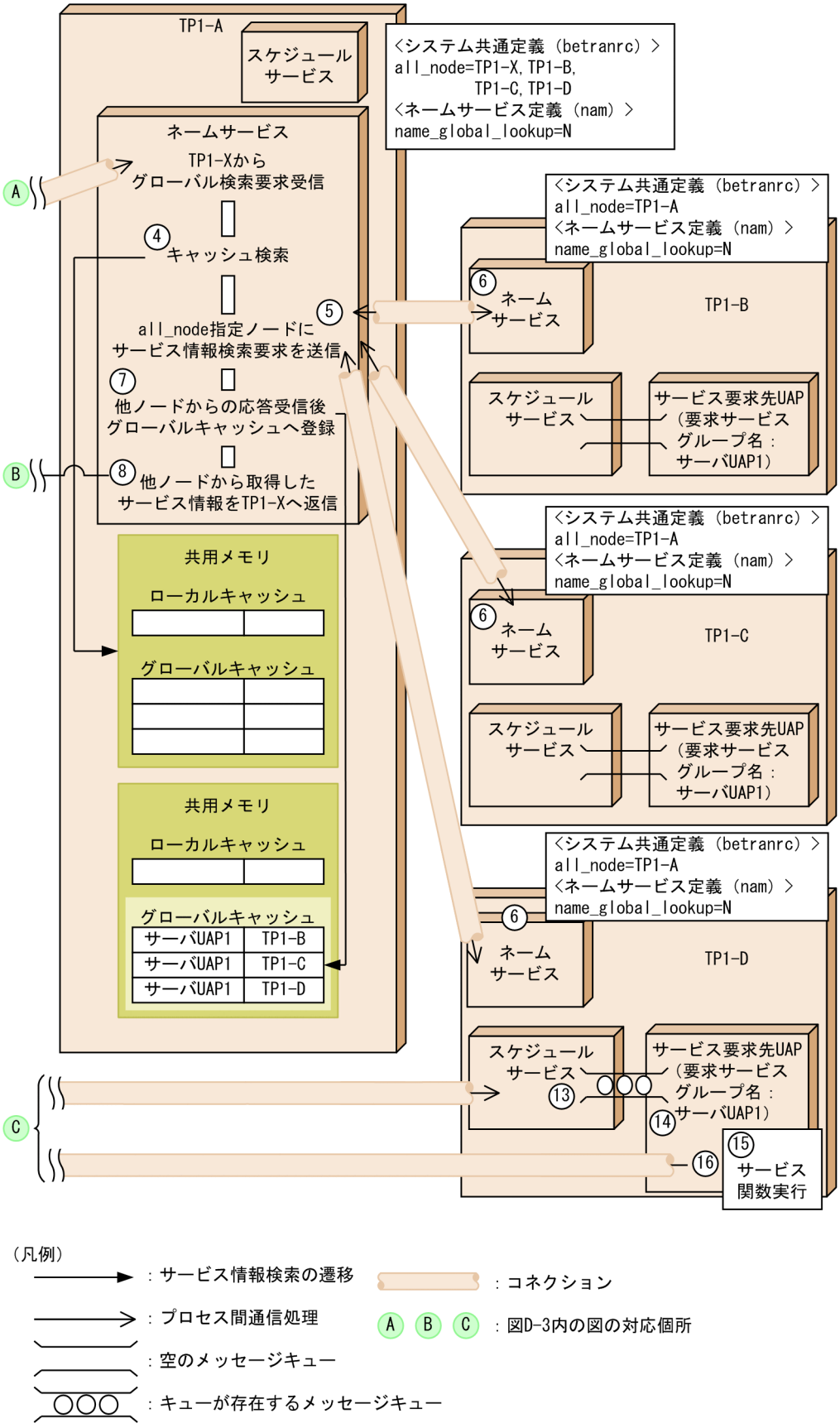
## 付録 D.3　グローバル検索処理の概要

グローバル検索とは，自 OpenTP1 が all_node オペランドに指定したノードと該当するノードの all_node オペランドに指定されたノードまで，サービス検索する検索方法です。グローバル検索機能については，「3.2.2(1)　グローバル検索機能」を参照してください。

ここでは，グローバル検索処理の概要について，次の図に示します。詳細なサービス情報の検索条件については，「付録 D.2　他ノードへのリモートプロシジャコールの処理の概要」を参照してください。

## 図 D-3　グローバル検索処理の概要






（凡例）

：サービス情報検索の遷移

：コネクション

：プロセス間通信処理

A B C ：図D-3内の図の対応個所

：空のメッセージキュー

：キューが存在するメッセージキュー

図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.～3. および 9.～12.は TP1-X の処理，4.，5.，7.，8.は TP1-A の処理，6.は TP1-B，TP1-C，TP1-D の処理，13.～16.は TP1-D の処理についての説明です。

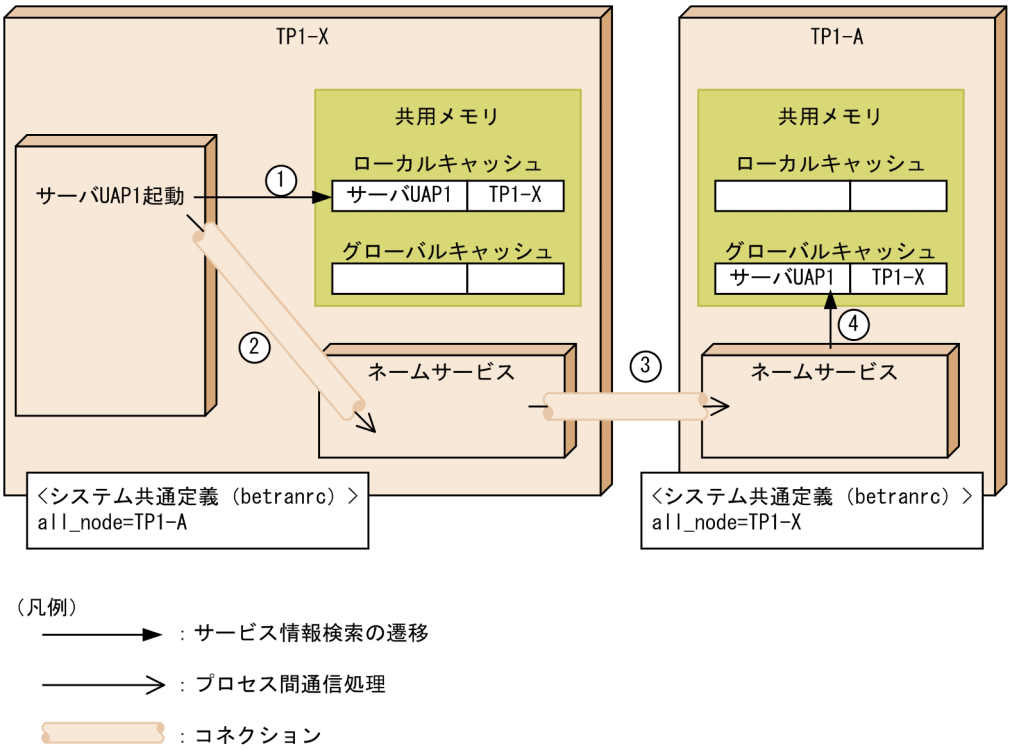
1. dc_rpc_call 関数の引数に指定されたサービスグループ名を検索キーに，該当するサービス情報がローカルキャッシュとグローバルキャッシュに登録されているかを確認します。
2. 該当するサービス情報が存在しなければ，ネームサービスにサービス情報の検索要求を送信します。
3. TP1-X は，システム共通定義の all_node オペランドに指定されたノード（TP1-A）に対してサービス情報のグローバル検索要求を送信します。
4. グローバル検索要求を受信した TP1-A のネームサービスは，ローカルキャッシュとグローバルキャッシュに該当するサービスグループが存在するか検索します。
5. 該当するサービス情報がキャッシュに存在しなければ，TP1-A はシステム共通定義の all_node オペランドに指定されたノード（TP1-B，TP1-C，TP1-D）に対してサービス情報の検索要求を送信します。
6. サービス情報の検索要求を受信した TP1-B，TP1-C，TP1-D のネームサービスは，ローカルキャッシュに該当するサービスグループが存在するか検索し，その結果を TP1-A のネームサービスに送信します。
7. 他ノードに対するサービス要求の応答が返ってきたため，TP1-A のネームサービスは，受信したサービス情報をグローバルキャッシュに登録します。
8. TP1-A のネームサービスは，検索して取得したサービス情報を，サービス要求元ノードである TP1-X のネームサービスに送信します。
9. TP1-A に対するグローバルサービス検索要求の応答が返ってきたため，TP1-X のネームサービスは，受信したサービス情報をグローバルキャッシュに登録します。
10. TP1-X のネームサービスは，検索して取得したサービス情報を，サービス要求元 UAP に返します。
11. 複数の同一サービスグループ名称の UAP が起動されていた場合は，その中から一つのサービス要求先 UAP（サーバ UAP1）を選択します。この説明では，TP1-D をあて先に選択すると仮定します。
12. 11.で選択した TP1-D のスケジュールサービスに対して，サービス実行要求を送信します。
13. サービス要求を受信したスケジュールサービスでは，該当するサービスグループのサービス要求先 UAP（サーバ UAP1）のメッセージキューに対し，サービス要求を登録します。
14. サービス要求が登録されたサービス要求先 UAP（サーバ UAP1）では，メッセージキューからサービス要求を取り出します。
15. サービス関数を実行します。
16. サービス関数の実行後，サービス要求元 UAP に対して，サービス要求先 UAP（サーバ UAP1）から，直接応答メッセージを送信します。

## 付録 D.4　サービス情報の登録・削除処理の概要

ここでは，サービス情報の登録・削除処理の概要について説明します。

サービス情報の登録処理の概要について，次の図に示します。

図 D-4　サービス情報の登録処理の概要




図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.～3.は TP1-X の処理，4.は TP1-A の処理についての説明です。

1. サーバ UAP1 を起動すると，自 OpenTP1（TP1-X）のローカルキャッシュに起動したサーバ UAP1 のサービス情報を登録します。
2. TP1-X のネームサービスに対してサービス情報の登録要求を送信します。
3. サービス情報の登録要求を受信したネームサービスは，システム共通定義の all_node オペランドに指定した，TP1-A のネームサービスにサービス情報の登録要求を送信します。
4. サービス情報の登録要求を受信した TP1-A のネームサービスは，送信元がシステム共通定義の all_node オペランドに指定されているか確認します。

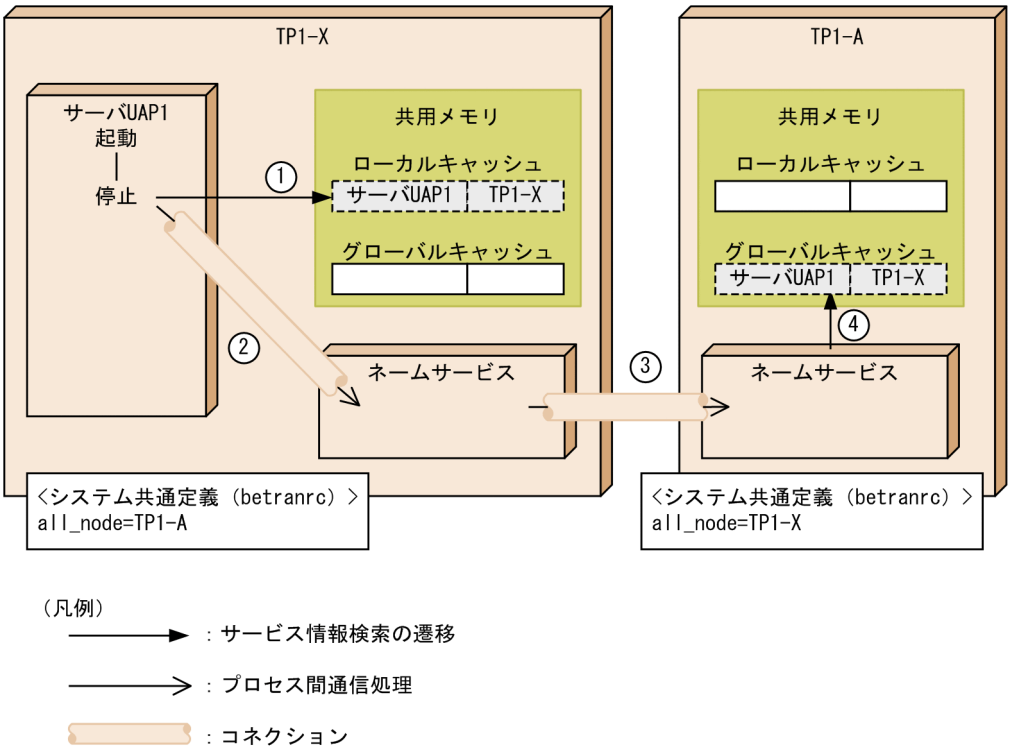
   送信元が all_node オペランドに指定されている場合，サービス情報の登録要求をされたサーバ UAP1 のサービス情報をグローバルキャッシュに登録します。

   送信元を all_node オペランドに指定していない場合は，サービス情報は登録しません。

   この図では，送信元である TP1-X が all_node オペランドに指定されているので，サーバ UAP1 のサービス情報を TP1-A のグローバルキャッシュに登録しています。

次に，サービス情報の削除処理の概要について，次の図に示します。

図 D-5　サービス情報の削除処理の概要




図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.～3.は TP1-X の処理，4.は TP1-A の処理についての説明です。

1. サーバ UAP1 を停止すると，自 OpenTP1（TP1-X）のローカルキャッシュに登録されている停止したサーバ UAP1 のサービス情報を削除します。
2. TP1-X のネームサービスに対してサービス情報の削除要求を送信します。
3. サービス情報の削除要求を受信したネームサービスは，システム共通定義の all_node オペランドに指定した，TP1-A のネームサービスにサービス情報の削除要求を送信します。
4. サービス情報の削除要求を受信した TP1-A のネームサービスは，送信元がシステム共通定義の all_node オペランドに指定されているか確認します。

   送信元が all_node オペランドに指定されている場合，サービス情報の削除要求をされたサーバ UAP のサービス情報をグローバルキャッシュから削除します。

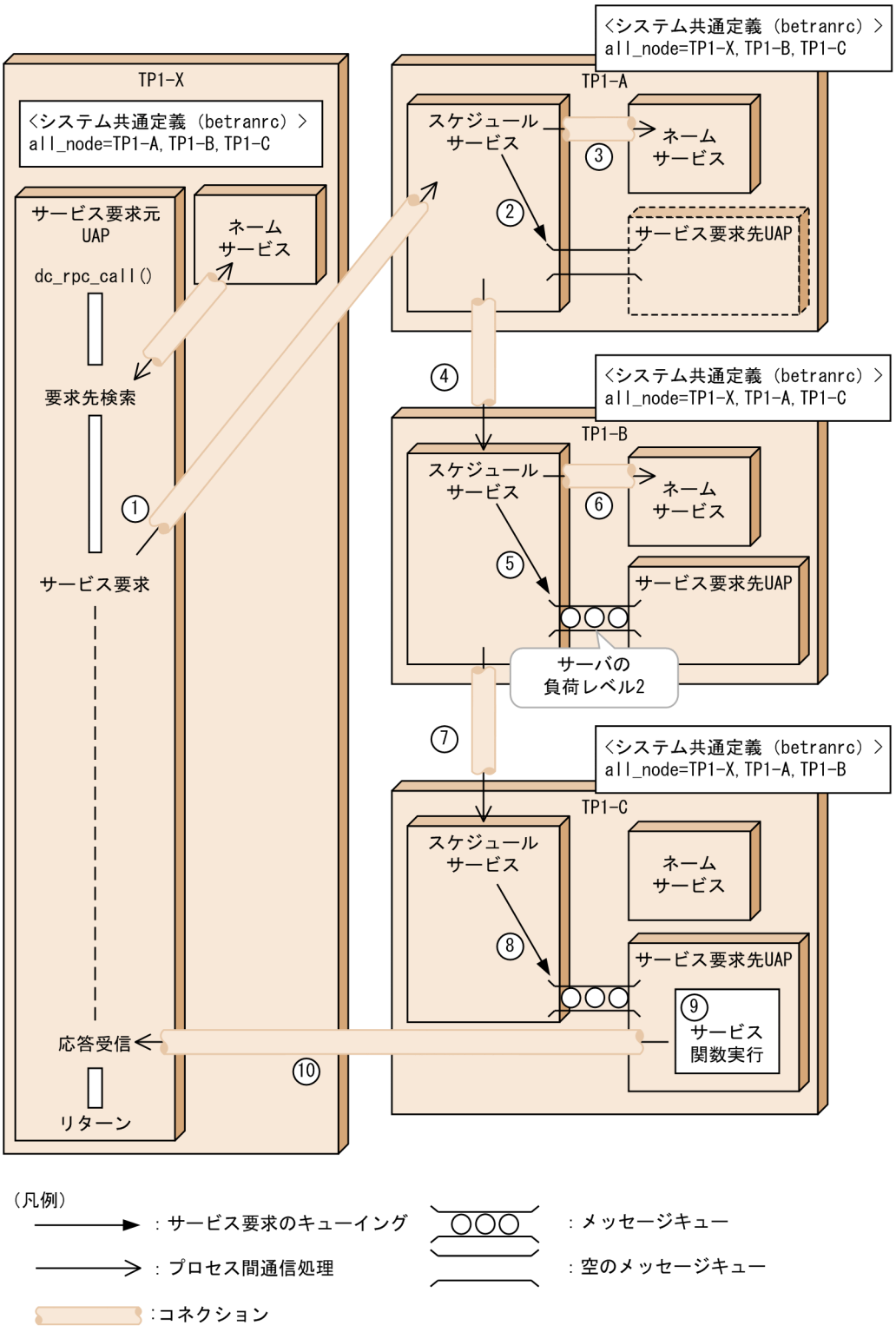
   送信元を all_node オペランドに指定していない場合は，サービス情報は削除しません。

   この図では，送信元である TP1-X が all_node オペランドに指定されているので，サーバ UAP1 のサービス情報が TP1-A のグローバルキャッシュから削除されています。

## 付録 D.5　他ノードへの転送処理の概要

他ノードへの転送処理の概要について，次の図に示します。

## 図 D-6　他ノードへの転送処理の概要







図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.は TP1-X の処理，2.～10.は TP1-A，TP1-B，TP1-C の処理についての説明です。

1. TP1-X はネームサービスを検索した結果，TP1-A ノードのスケジュールサービスにサービス要求をします。

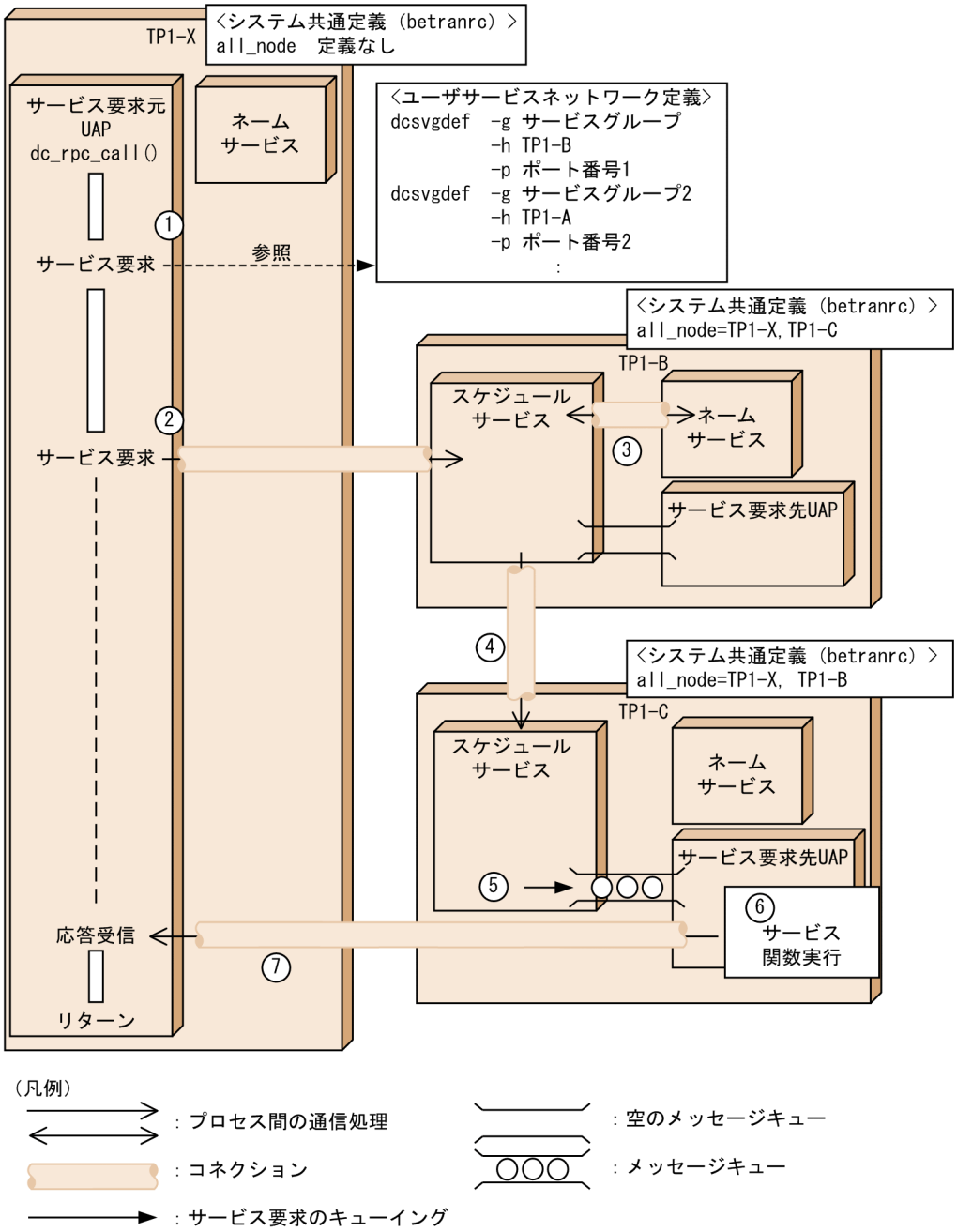
2. TP1-A のスケジュールサービスがサービス要求しようとした時点で，TP1-A のサーバが閉塞しているため，キューイングしません。
3. スケジュールサービスは，ネームサービスに対して転送先を検索します。

   サービス要求元の RPC 要求が dc_rpc_call_to 関数でスケジューラダイレクト機能を使用している場合は，転送しないでサービス要求元にエラーを返します。
4. TP1-B のスケジュールサービスに転送します。
5. サーバの負荷レベルが 2 で高負荷のためキューイングしません。
6. スケジュールサービスは，ネームサービスに対して転送先を検索します。
7. TP1-C のスケジュールサービスに転送します。
8. サーバが閉塞中，または高負荷ではないので，サービス要求をメッセージキューに登録します。
9. サービス関数を実行します。
10. サービス関数の実行後，サービス要求を行ったサービス要求元 UAP に対して，サービス要求先 UAP（サーバ UAP1）から，直接応答メッセージを送信します。

## 付録 D.6　dcsvgdef 定義コマンドを使用したリモートプロシジャコールの処理の概要

ユーザサービスネットワーク定義の dcsvgdef 定義コマンドを使用したリモートプロシジャコールの処理の概要について，次の図に示します。

## 図 D-7　dcsvgdef 定義コマンドを使用したリモートプロシジャコールの処理の概要




図で示した処理の流れについて，次に説明します。番号は，図中の番号と対応しています。なお，1.～2.は TP1-X の処理，3.～4.は TP1-B，5.～7.は TP1-C の処理についての説明です。

1. TP1-X はネームサービスを検索しないで，ユーザサービスネットワーク定義からサービス要求の送信先 TP1-B を求めます。
2. TP1-B のスケジュールサービスに送信します。
3. TP1-B で起動しているスケジュールサービスがネームサービスを検索し，TP1-B および TP1-C に UAP が存在するという結果が通知されます。
4. ランダムに決めた結果，TP1-C に転送します。

5. TP1-C のスケジュールサービスは，サービス要求をメッセージキューにキューイングします。

6. サービス関数を実行します。

7. サービス関数の実行後，サービス要求を行ったサービス要求元 UAP に対して，サービス要求先 UAP（サーバ UAP1）から，直接応答メッセージを送信します。

# 付録 E　用語解説

## (英字)

### CUP（Client User Program)

OpenTP1 クライアント機能（TP1/Client）で使用する，クライアント専用の UAP のことです。エラーログ機能と UAP トレース機能を使用できます。

### DAM ファイル（Direct Access Method)

OpenTP1 専用の直接編成ファイルを DAM ファイルといいます。相対ブロック番号をキーとした直接アクセス法で参照，または更新できます。参照だけの場合は順アクセスもできます。

### DTP モデル（Distributed Transaction Processing)

オープンシステムの標準化を目的とした団体である X/Open が規定する，分散処理システムモデルのことです。DTP モデルは，トランザクション処理を管理，実行するトランザクションマネジャ，各種資源を管理するリソースマネジャ（RM)，および業務処理をする UAP（AP）から構成されます。

### IST テーブル

IST サービスで使う，データを格納するテーブルのことです。IST サービスを使うときには，IST テーブルの名称を関数に設定してアクセスします。IST テーブルは各システムの TP1/Shared Table Access が管理しているので，テーブルの実体がどのノードにあるかを UAP で意識しなくても，IST テーブルへアクセスできます。

### MCF メイン関数

OpenTP1 のリソースマネジャである TP1/Message Control を使用する場合は，ユーザがメイン関数を作成します。TP1/Message Control のメイン関数を MCF メイン関数といいます。MCF メイン関数では，main（）と dc_mcf_svstart 関数を呼び出します。システムで UOC を使用する場合は，MCF メイン関数に UOC の関数アドレスを設定しておきます。

MCF メイン関数は，通信プロトコル対応向けとアプリケーション起動サービス向けで，別に作成します。

### MHP（Message Handling Program)

OpenTP1 の UAP のうち，TP1/Message Control を使用したメッセージ送受信の業務で使用する UAP のことです。MHP は，ほかのシステムから送信されたメッセージを受信するサービス関数と，サービス関数をまとめるメイン関数から構成されます。

### MQA キューファイル

メッセージキューイング機能を使った通信をする場合に使う，メッセージキューを格納するファイルのことです。OpenTP1 の場合，キューファイルは OpenTP1 ファイルシステム上に作成します。

### MQA サービス，MQT サービス

メッセージキューイング機能を使った通信をする場合に必要な，OpenTP1 のシステムサービスです。MQA サービスとは，メッセージキューや UAP プロセスを管理するシステムサービスです。MQT サービスとは，システム間の通信（TCP/IP プロトコル）とのインタフェースを管理するシステムサービスです。

### MQI（Message Queue Interface）

メッセージキューイング機能を使った通信をする場合に，UAP で使う命令文（API）のことです。MQI は，WebSphere MQ で標準化しています。

### MQ システム

メッセージキューイング機能のキューマネジャがあるシステム（ノード）のことです。

### OpenTP1 インストールディレクトリ

OpenTP1 をインストールするディレクトリのことです。OpenTP1 インストールディレクトリのパスは，使用する OS によって異なります。OS ごとの OpenTP1 インストールディレクトリのパスは次のとおりです。

- AIX，HP-UX または Solaris の場合：/BeTRAN
- Linux の場合：/opt/OpenTP1
- Windows の場合：C:¥OpenTP1
  C:¥は使用している環境によって異なります。

### OpenTP1 管理者

OpenTP1 を管理する，UNIX OS の利用者のことです。OpenTP1 管理者をどの利用者名称とするかは，スーパユーザが決めます。OpenTP1 管理者は，OpenTP1 を使う上で重要な権限を持つユーザです。システムの機密保護上，利用者名称にはパスワードを必ず設定して，限られた人だけが OpenTP1 管理者の利用者名称を使えるようにしてください。

### OpenTP1 ノード

マルチノードエリア，またはマルチノードサブエリアを構成する個々の OpenTP1 システムを OpenTP1 ノードといいます。それぞれの OpenTP1 ノードは，OpenTP1 のシステム共通定義で指定したノード識別子で区別されます。系切り替え構成は，一つの OpenTP1 ノードと見なします。

### OpenTP1 ファイルシステム/OpenTP1 ファイル

OpenTP1 専用のファイルシステムを OpenTP1 ファイルシステム，OpenTP1 ファイルシステムの個々のファイルを OpenTP1 ファイルといいます。OpenTP1 ファイルシステムとして使用する領域をキャラクタ型スペシャルファイル上に作成して，OpenTP1 のコマンドで初期化します。その後，使用する目的別のコマンドで OpenTP1 ファイルを初期化してから使用します。

### OpenTP1 ホームディレクトリ

OpenTP1 で使う各種ファイル，またはディレクトリを格納しているディレクトリのことです。

OpenTP1 では，OpenTP1 ホームディレクトリを DCDIR という環境変数で管理しています。OpenTP1 が組み込んであるマシンでは，任意のディレクトリから環境変数$DCDIR と指定して移動すると，OpenTP1 ホームディレクトリに移れます。

OpenTP1 の各マニュアルでは，OpenTP1 ホームディレクトリを$DCDIR/と表記しています。

### OSI TP（Open Systems Interconnection transaction processing）

OSI 参照モデルに準拠した，分散トランザクション処理に必要な通信手順を規定したプロトコルのことです。OpenTP1 が OSI TP を使う通信には，次に示す 2 とおりがあります。

- クライアント/サーバ形態の通信で通信プロトコルに使う場合

  サービス要求と応答で通信する形態です。通信には，XATMI インタフェースの API を使います。この形態の通信をする場合には，TP1/Server Base に加えて，TP1/NET/Library と TP1/NET/OSI-TP-Extended が必要です。

- メッセージ送受信形態の通信で通信プロトコルに使う場合

  メッセージの送信と受信で通信する形態です。通信には，MCF の API（例　dc_mcf_send 関数）を使います。この形態の通信をする場合には，TP1/Server Base に加えて，TP1/Message Control，TP1/NET/Library，および TP1/NET/OSI-TP が必要です。

### RPC（Remote Procedure Call）

→リモートプロシジャコール欄を参照してください。

### RPC 抑止リスト

OpenTP1 システムが，未起動の OpenTP1 ノードの情報を保持しているリストのことです。

### SPP（Service Providing Program）

OpenTP1 の UAP のうち，ファイルへのアクセスなどサーバの役割をするプログラムのことです。SPP は，クライアント UAP から要求されたサービスを実行するサービス関数と，サービス関数をまとめるメイン関数から構成されます。

### SUP（Service Using Program）

OpenTP1 の UAP のうち，SPP に処理要求をするだけの，クライアント専用のプログラムのことです。ほかの UAP にサービスを提供するための関数は持ちません。

### TAM ファイル（Table Access Method）

OpenTP1 専用の単純構造テーブルを使用してアクセスできるファイルを TAM ファイルといいます。単純構造テーブルへは，テーブル名とキー値によってアクセスできます。アクセスすることで，レコードの検索，更新，追加，削除，および検索の取消ができます。さらに，テーブル情報を取得できます。

### TCP/IP（Transmission Control Protocol/Internet Protocol）

米国国防総省高等研究計画局（DARPA）のプロジェクトである ARPANET が開発したプロトコルのことです。TCP/IP プロトコルは，主に LAN で使われます。OpenTP1 が TCP/IP を使う通信には，次に示す 2 とおりがあります。

- **クライアント/サーバ形態の通信で通信プロトコルに使う場合**

  サービス要求と応答で通信する形態です。通信には，OpenTP1 独自の RPC（例 dc_rpc_call 関数），XATMI インタフェースの API，および TxRPC インタフェースの通信を使います。この形態の通信をする場合には，TP1/Server Base が必要です。

- **メッセージ送受信形態の通信で通信プロトコルに使う場合**

  メッセージの送信と受信で通信する形態です。通信には，MCF の API（例　dc_mcf_send 関数）を使います。この形態の通信をする場合には，TP1/Server Base に加えて，TP1/Message Control，TP1/NET/Library，および TP1/NET/TCP/IP が必要です。

### TP モニタ（Transaction Processing）

トランザクション処理の監視，および制御をするソフトウェアのことです。オンラインシステムを構築するための基盤となる機能を提供しています。主な機能として，端末やほかのコンピュータとデータをやり取りするための通信機能，トランザクションを効率的に処理するためのスケジュール機能，トラブルが起こってもデータとトランザクションの消失や不整合を防ぐ回復機能があります。

### UAP（User Application Program）

ユーザの業務をプログラムとして作成したものです。アプリケーションプログラムともいいます。

### UOC（User Own Coding）

メッセージ送受信の UAP をより多様な業務に対応させるためにユーザがコーディングするプログラムです。起動するアプリケーション名を決定したり，UAP が送信したメッセージを他システムに送信する前に編集できます。

### WebSphere MQ

米国 IBM 社が開発した，メッセージ蓄積型の通信を実現する製品群のことです。OpenTP1 では，TP1/Message Queue を使うとメッセージ蓄積型の通信ができるようになります。

## （ア行）

### アーカイブジャーナルファイル

クラスタ/並列システム形態で OpenTP1 を使用する場合に，グローバルアーカイブジャーナル機能で使用する，各 OpenTP1 ノードから集められたジャーナルを格納するファイルです。

### アソシエーション

通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合に，システム間で確立する論理的な通信路のことです。このマニュアルでは，通信プロトコルに OSI TP を使ったクライアント/サーバ形態の通信をする場合に限り，論理的な通信路の呼称をアソシエーションと表記します。

### アプリケーション

OpenTP1 の業務処理の総称です。アプリケーションとして作成するプログラムをアプリケーションプログラム（またはユーザアプリケーションプログラム　UAP）といいます。UAP を OpenTP1 に登録して，サーバとして業務を実行するプロセスをユーザサーバといいます。

### 一方送信メッセージ

TP1/Message Control を使ったメッセージ送受信の業務で，応答を返さないメッセージを一方送信メッセージといいます。一方送信メッセージには，送信優先度を付けることができます。

### エージェントノード

ノード自動追加機能を使用した OpenTP1 システムを構築するノードのうち，マネジャノードによって，ノードの停止および開始を監視されるノードのことをいいます。

### 応答メッセージ

→問い合わせ応答メッセージ欄を参照してください。

## （カ行）

### 稼働統計情報

ユーザサーバ，および OpenTP1 のシステムサービスの稼働情報を稼働統計情報といいます。稼働統計情報を編集して出力することで，OpenTP1 の稼働状況を知ることができます。

### 監査ログ

システム構築者，運用者，および使用者が OpenTP1 のプログラムに対して実行した操作，およびその操作に伴うプログラムの動作の履歴が出力されるファイルです。監査ログには「いつ」，「だれが」，「何をしたか」などが記録されます。そのため，システムの使用状況，不正アクセスなどを監査する資料として使用できます。

### キューグループ ID

TP1/Message Control を使ったメッセージ送受信で使用する，メッセージキューファイルの識別子のことです。OpenTP1 のコマンドでキューグループの状態を表示する場合に使用します。

### キューマネジャ

メッセージキューイング機能を管理するソフトウェア製品のことです。メッセージキューイング機能を使って通信するシステムには，キューマネジャが必要になります。OpenTP1 の場合は，TP1/Message Queue がキューマネジャの役割をします。

### クライアント/サーバ

プログラムとプログラムで通信する場合の関係を示す用語です。業務処理を依頼する方をクライアント，要求を受けて業務を実行する方をサーバといいます。クライアント/サーバとは，プログラム間の相対的な関係を示します。

### クライアント UAP/サーバ UAP

複数の UAP プロセス間でクライアント/サーバ型の通信をするときに，業務処理を依頼するためにサービスを要求する UAP をクライアント UAP，要求を受けて業務を実行する UAP をサーバ UAP といいます。クライアント UAP となるのは，SUP，SPP，および MHP です。サーバ UAP となるのは SPP です。

### クラスタ/並列システム

複数のサーバを LAN で接続して，一つのサーバの処理能力ではできないような業務を，サーバ同士で連携して実現するシステム形態のことです。クラスタ/並列システムで OpenTP1 を使用する場合は，運用支援のため，TP1/Multi を使用します。

### グローバルアーカイブジャーナルサービス

クラスタ/並列システム形態で OpenTP1 を使用する場合に，各 OpenTP1 ノードからジャーナルを集める機能のことです。グローバルアーカイブジャーナルサービスを使用することで，OpenTP1 ノードごとにジャーナルをアンロードする手間が省けます。

### グローバルキャッシュ

ネームサービスが他ノードで起動しているサーバのサービス情報を管理する領域を示します。

### グローバルトランザクション

トランザクション処理を行うUAPプロセスをトランザクションブランチといいます。RPCを使用した場合，複数のUAPプロセスで構成されたトランザクション処理ができます。このように複数のUAPプロセスで構成されたトランザクションブランチの集合をグローバルトランザクションといいます。

トランザクションを開始したトランザクションブランチをルートトランザクションブランチといいます。

### コネクション

TP1/Message Controlを使ったメッセージ送受信の業務で，ほかのシステムとOpenTP1システムとの間に確立する論理的な通信路をコネクション（通信プロトコルによってはアソシエーション）といいます。このマニュアルではメッセージ送受信の説明に限り，呼称をコネクションで統一しています。

### コミット

トランザクションの同期点を取得できたことをコミットといいます。コミットできた場合，該当するトランザクション処理が有効となります。

## （サ行）

### サービス

クライアント/サーバシステムでは，クライアントから要求された手続きを総称してサービスといいます。UAPのコーディング時には，C言語の場合は関数として，COBOL言語の場合はサブルーチンとして，サービスを作成します。作成したサービスのことを，C言語の場合はサービス関数，COBOL言語の場合はサービスプログラムといいます。

### サービス関数動的ローディング機能

UAP共用ライブラリ化したサービス関数を，動的にローディングする（読み込む）機能です。UAP共用ライブラリ化とは，UAPのソースファイルを翻訳（コンパイル）して作成したUAPオブジェクトファイルを結合（リンケージ）して，共用ライブラリとしてまとめることです。

サービス関数動的ローディング機能を使うと，サービス関数を追加または削除する場合に，スタブの変更，およびUAPの実行形式ファイルの再生成をしなくても，ユーザサービス定義のserviceオペランドの変更だけでサービス関数を追加または削除できます。UAP起動時にサービス関数をローディングするため，UAPの実行形式ファイル作成時には，スタブおよびサービス関数は不要です。

### サービスグループ

OpenTP1のサーバUAPは，クライアントからの手続き要求を処理するサービスの集合です。このことから，サーバUAPのことをサービスグループといいます。OpenTP1のリモートプ

ロシジャコールの関数 dc_rpc_call（）では，サービスを要求するときにサービスグループ名とサービス名を引数に設定します。

### サービス情報優先度指定機能

ネームサービスがサービス要求元のクライアント UAP にサービス情報を返すときに，特定のノードのサービス情報を優先的に返す機能です。サービス情報を優先的に返すノードを，優先選択ノードといいます。

この機能を使用すると，通常は優先選択ノードのサーバ UAP を使用し，障害が発生した場合だけ優先選択ノード以外のノードのサーバ UAP を使用できるため，サーバ UAP を実行系と待機系に分けて扱うことができます。

### システムサービス

OpenTP1 の UAP をユーザサーバということに対して，OpenTP1 の個々の機能を指してシステムサービスといいます。

### シナリオ

シナリオテンプレートを業務に関連づけて運用手順として実行できるようにしたものです。JP1/AJS2 - Scenario Operation が，シナリオテンプレートをシナリオとして JP1/AJS に登録し，JP1/AJS が実行します。

### シナリオジョブ

シナリオジョブテンプレートを業務に関連づけて運用手順として実行できるようにしたものです。JP1/AJS2 - Scenario Operation が使用します。シナリオ中に定義されているコマンド，シェルスクリプト，Windows 実行ファイルなどを実行させるためのオブジェクトです。

### シナリオジョブテンプレート

JP1/AJS2 - Scenario Operation が使用するシナリオテンプレートの部品です。シナリオテンプレート中に定義されているコマンド，シェルスクリプト，Windows 実行ファイルなどを定義しています。OpenTP1 では，シナリオテンプレートを利用する運用で使用します。

### シナリオテンプレート

運用手順を，テンプレート（ひな形）として部品化したものです。JP1/AJS2 - Scenario Operation が使用します。OpenTP1 では，シナリオテンプレートを利用する運用で使用します。

### シナリオ変数

JP1/AJS2 - Scenario Operation で，運用環境によって変化する情報を，シナリオに応じてあらかじめ設定しておく変数です。OpenTP1 のディレクトリ情報などを設定します。

### シナリオライブラリ

JP1/AJS2 - Scenario Operation で，シナリオテンプレートを管理するフォルダです。

### ジャーナル

システムを回復したり各種の稼働状況を検知したりするために，OpenTP1 が取得する履歴情報をジャーナルといいます。OpenTP1 では，次に示すジャーナルを取得します。

- システムの全面回復，部分回復に使用するジャーナル情報
- システムのトレース情報
- ユーザが任意に取得するジャーナル情報

### 出力キュー

TP1/Message Control を使ったメッセージ送受信の業務で，ほかのシステムへ送信するメッセージの待ち行列を格納するキューを出力キューといいます。

### スーパユーザ（superuser）

UNIX OS の最高権限を持つユーザのことです。UNIX ファイルシステムのすべてのファイルに対してアクセス権を持ちます。スーパユーザの利用者名称は root で固定されています。

### スケジュール機能

ユーザサーバを負荷の量に応じて起動させたり停止させたりする，OpenTP1 の機能です。スケジュールとプロセスを管理することで，負荷を分散したり，使用するプロセス数を抑えたりでき，プロセスが増え過ぎて性能が下がることを避けられます。

### スタブ

クライアント/サーバ形態の通信で，サービス要求とサーバ UAP のサービスとを結び付けるライブラリの役割をするプログラムをスタブといいます。ユーザが作成した RPC インタフェース定義ファイルから，スタブを生成するコマンドでスタブを作成します。作成したスタブは C 言語のプログラムの形式です。このスタブをコンパイルして UAP の実行形式ファイルにリンケージします。スタブが必要な UAP は，RPC を使用する場合は SPP，XATMI インタフェースの通信を使用する場合は SUP と SPP です。また，MHP にもスタブが必要です。ただし，すべてのサービス関数を UAP 共用ライブラリ化してサービス関数動的ローディング機能を使う場合，スタブは不要です。UAP 共用ライブラリ化とは，UAP のソースファイルを翻訳（コンパイル）して作成した UAP オブジェクトファイルを結合（リンケージ）して，共用ライブラリとしてまとめることです。

スタブについては，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

### ステータスサービス

OpenTP1 の機能の一つで，UAP の実行状態などシステム内の各種情報を管理する機能です。

### 全面回復

OpenTP1 に障害が発生した場合に，すべての資源を障害発生前の状態に戻すことを全面回復といいます。

## （タ行）

### チェックポイントダンプ

OpenTP1 に障害が発生した場合に，全面回復に掛かる時間を短縮するために使用するファイルをチェックポイントダンプといいます。ジャーナルの情報を一定間隔で取得して，全面回復時には最新のチェックポイントダンプの情報とそれ以降のジャーナルの情報を参照して回復します。このことで，取得したジャーナルを最初から読み込まなくて済むので，回復時間を短縮できます。

### デッドロック

複数の UAP が複数の資源を確保しようとして，互いの UAP が資源を解放するのを待ち続けて処理が止まってしまう状態のことをデッドロックといいます。

### 問い合わせ応答メッセージ

メッセージ送受信の業務で，メッセージを送信して応答を返す形態で使用するメッセージのことです。

### 同期点

トランザクション処理の区切りを同期点といいます。トランザクション処理が正常終了したことを示す同期点処理をコミット，トランザクション処理がうまくいかなかったため無効にする同期点処理をロールバックといいます。

### ドメイン

ネットワークを論理的に区切った単位のことです。ドメインを管理する場合は，DNS や NIS を使います。大規模なネットワークでクライアント/サーバ形態の通信をする場合には，OpenTP1 のリモートプロシジャコールにドメイン修飾をして，スケジュールの効率を上げることができます。

### トランザクショナル RPC

トランザクションとして実行している UAP プロセスの処理から RPC を使うと，サービス要求先の UAP プロセスの処理もトランザクションの処理にできます。このような RPC をトランザクショナル RPC といいます。サービス要求先のプロセスの処理がトランザクションとなるには，サービス要求先のプロセスがトランザクション属性（ユーザサービス定義で atomic_update = Y を指定）であることが前提です。

### トランザクション

ファイルからデータを読み出して，変更したデータを書き込む処理（更新処理）では，データの一貫性を保持するため，途中で分けられません。このような処理の単位をトランザクションといいます。トランザクションの処理結果は，有効にするか無効にするかのどちらかに必ず決められます。

### トランザクションブランチ

RPC を使用した複数の UAP プロセスにわたる処理をトランザクションとする処理（グローバルトランザクション）を構成するそれぞれの UAP プロセスをトランザクションブランチといいます。トランザクションを開始したトランザクションブランチを，ルートトランザクションブランチといいます。

### トランザクションマネジャ（Transaction Manager）

トランザクション処理を管理，および実行する機能のことです。

### トリガ

メッセージキューイング機能を使った通信をする場合に，メッセージキューにメッセージが到着したことを自システムの UAP へ知らせる機能のことです。トリガイベントを受信する UAP をトリガモニタアプリケーションといいます。

## （ナ行）

### 入力キュー

メッセージ送受信の業務で，ほかのシステムから受信したメッセージの待ち行列を格納するキューを入力キューといいます。

### ノード

ホスト，または OpenTP1 ノードを意味します。

### ノード間負荷バランス機能

同じサービスグループ名のユーザサーバを複数のノードに置いて，そのサービスグループ名でサービスを要求されたときにどちらのノードのユーザサーバでも処理できるようにする負荷分散の形態です。ノードのスケジュール状態に応じて，より効率的に処理できるノードへ負荷を分散できます。

### ノードリスト

マネジャノードが管理するノード情報のリストのことです。共用メモリ上で管理します。ノード自動追加機能を使用するすべてのノードで共有し，システム共通定義の all_node オペランドに定義していた情報を，管理します。

### ノードリストの整合性

マネジャノードは，エージェントノードから定期的にノード情報を取得し，エージェントノードにノードリストを配布します。これによって，ノード自動追加機能を使用するすべてのノードでノード情報を共有できます。このことをノードリストの整合性といいます。

### ノードリストファイル

ノード自動追加機能でノードリストの引き継ぎをするときに作成するファイルです。OpenTP1 ファイルシステム上に作成し，OpenTP1 のオンライン中に，共用メモリ上にあるノードリストの情報を格納します。

### ノーマルノード

ノード自動追加機能を使用した OpenTP1 システムを構築するノードのうち，次のノードのことをいいます。

- バージョン 07-05 以前の OpenTP1 を使用しているノード
- ノード自動追加機能を使用していないノード（システム共通定義の name_service_mode オペランドに normal を指定または省略）

## （ハ行）

### 部分回復

OpenTP1 システム全体の回復を全面回復ということに対して，UAP の回復を部分回復といいます。UAP に障害が発生した場合，UAP のトランザクション処理を無効にして，資源を障害発生前の状態に戻します。

### プロセス

ユーザサーバ，または OpenTP1 が，OS の作業領域を使用することで生成される作業領域の処理をプロセスといいます。OpenTP1 では，ユーザサーバのプロセスが必要以上に増えたり減ったりしないように，使用するプロセスの総数を管理しています。

### ホスト

ネットワークにつながれた，OpenTP1 が稼働する一つの計算機（マシン）のことです。マルチ OpenTP1 の場合は，複数の OpenTP1 から構成される一つのホストとなります。

## （マ行）

### マネジャノード

ノード自動追加機能を使用する OpenTP1 システムを構築するノードのうち，OpenTP1 システムを構成するすべてのノードの情報を管理するノードのことをいいます。OpenTP1 システム内で一つだけ必要です。

### マルチ OpenTP1

一つのホストに複数の OpenTP1 がある形態です。それぞれの OpenTP1 は個別に運用します。

### マルチサーバ

OpenTP1 のユーザサーバプロセスを複数起動させて，複数のサービス要求を処理する機能です。マルチサーバによって，UAP の処理効率が上がります。

### メッセージキュー（メッセージキューイング機能の用語）

メッセージキューイング機能を使った通信をする場合に使う，メッセージを格納するキューのことです。メッセージキューに登録したメッセージは，キューマネジャが通信相手に送信します。MQA サービス定義，および MQI の引数の設定で，必要なメッセージを受信した順序に関係なく取り出すこともできます。

### メッセージキュー（メッセージ送受信の用語）

メッセージ送受信の業務で，メッセージを格納するファイルをメッセージキューといいます。メッセージキューには，TP1/Message Control で受信したメッセージを格納する入力キューと，ほかのシステムへ送信するメッセージを格納する出力キューがあります。

### メッセージキューイング機能

米国 IBM 社が開発した，メッセージ蓄積型の通信手順のことです。メッセージキューイング機能を使った通信では，UAP が登録したメッセージをキューマネジャが送受信するため，システム間の通信手順や通信障害時の処理を UAP で意識しなくて済みます。また，UAP は任意のタイミングでメッセージを扱えるので，電子メールのような運用ができます。UAP からキューへアクセスするときには，WebSphere MQ 標準の API（MQI　Message Queue Interface）を使います。

### メッセージ出力通番

メッセージ送受信の業務で，複数のメッセージを送信する場合に，ユーザが送信抜けや二重の送信を認識できるようにするために OpenTP1 が付ける通し番号のことです。

### メッセージログ

OpenTP1 が出力するシステム管理情報をメッセージログといいます。メッセージログは，操作画面に出力されて，同じ情報が専用のファイルに格納されます。格納したメッセージログを編集出力する場合は，OpenTP1 のコマンド（logcat コマンド）を実行します。

OpenTP1 のメッセージログを，システム内に専用に作成するアプリケーションプログラムへ通知できます。通知を受信したアプリケーションプログラムは，他社のネットワーク管理システムへ OpenTP1 の状態を知らせることができます。メッセージログを通知する場合は，OpenTP1 のログサービス定義の log_notify_out オペランドに Y を指定しておいてください。

## （ヤ行）

### ユーザサーバ

OpenTP1 の UAP，および OpenTP1 の UAP を実行するプロセスの総称です。UAP の実行形式ファイルが，OpenTP1 のサーバの業務をすることからユーザサーバといいます。

### ユーザジャーナル（User Journal）

システムジャーナルファイルに取得するユーザ任意の情報をユーザジャーナルといいます。ユーザジャーナルは，UAP の処理から取得できます。

## （ラ行）

### リアルタイム統計情報

システム全体，サーバおよびサービス単位で，リアルタイムに出力できる統計情報を，リアルタイム統計情報といいます。リアルタイム統計情報を出力することで，OpenTP1 システムの稼働状況をリアルタイムに把握でき，システムの運用管理や障害復旧を迅速に行えます。

### リアルタイム統計情報サービス

OpenTP1 の機能の一つで，OpenTP1 システムの稼働状況をリアルタイムに把握するためのリアルタイム統計情報を管理する機能です。

### リソースマネジャ（Resource Manager）

分散処理システムでの，資源を管理する機能の総称です。DBMS はリソースマネジャです。

### リモートプロシジャコール（Remote Procedure Call）

UAP を実行するプロセス間で通信する機能を，リモートプロシジャコール（RPC）といいます。OpenTP1 の UAP は，ほかのシステムの UAP と RPC で通信します。RPC を使用するときは，通信相手がクライアント/サーバシステムのどのノードの UAP かを意識する必要はありません。

### ルートトランザクションブランチ

グローバルトランザクションに属するトランザクションブランチのうち，トランザクションを開始した UAP プロセスをルートトランザクションブランチといいます。

### ローカルキャッシュ

ネームサービスが自 OpenTP1 で起動しているサーバのサービス情報を管理する領域を示します。

### ロールバック

トランザクション処理を無効にする同期点処理をロールバックといいます。UAP から関数を発行する場合と，OpenTP1 から UAP の処理を無効にする場合があります。

### 論理端末

メッセージ送受信の業務で，ほかのシステムにある通信相手の論理的な名称です。UAP からメッセージ送受信する場合には，論理端末名称を使用します。

### 論理メッセージ

メッセージ送受信の業務で，システム間でやり取りする業務単位でのメッセージを論理メッセージといいます。論理メッセージは，一つのセグメントから構成される場合と，複数のセグメントから構成される場合があります。

# 索引

## N



## O



## P



## Q



## R



## S



## T

## U



## X



## あ



## い

## う



## え



## お



## か



## き



## く



## け



## こ



## さ

## し



## す



## せ



## そ



## た



## ち



## つ



## て

## と



## な



## に



## ね



## の



## は



## ひ



## ふ

## れ



## ろ