HITACHI
Inspire the Next

OpenTP1 Version 7
分散トランザクション処理機能

OpenTP1 テスタ・UAP トレース使用の手引

解説・手引・文法・操作書

3000-3-D57-31

前書き

■ 対象製品

マニュアル「OpenTP1 解説」を参照してください。

■ 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

2020 年 4 月　3000-3-D57-31

■ 著作権

変更内容

変更内容（3000-3-D57-31） uCosminexus TP1/Server Base 07-53，uCosminexus TP1/Server Base(64) 07-53

追加・変更内容	変更個所
ユーザサービス定義の，子プロセスを生成する関数についての注意事項を変更した。	3.1.4(4)

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-D57-30）uCosminexus TP1/Server Base 07-50，uCosminexus TP1/Online Tester 07-50

追加・変更内容
DML 形式の COMMIT 文を発行する場合，コミット要求でエラーが発生しても，UAP でエラーを検出できない説明を追加した。
使用できるデバッガの種別を COBOL2002 に修正した。
次のオペランドの説明を追加・変更した。 • テスタサービス定義 ・uto_server_count の指定可能範囲値。 ・rpc_trace，rpc_trace_name，rpc_trace_size の説明。 • ユーザサービス定義 ・putenv/dcputenv 形式の DCUTOTRCACCL を追加。
全入出力データトレースの取得時間を短縮できるようにした。これに伴い，ユーザサービス定義に次の putenv/dcputenv 形式の追加および説明を追加した。 DCUTOTRCACCL
運用コマンド結果データファイルの注意事項について，DML の SEND 文で運用コマンドを発行する場合のデータ部の指定方法を，テスタファイル作成機能の使用有無別に記載するように変更した。
UAP の作成について，次の注意事項の説明を追加した。 • アーカイブライブラリまたは共用ライブラリを使用する場合のリンケージ方法。 • オンラインテスタが提供するシミュレート機能で未サポートの MCF 関数を使用する場合のリンケージ方法。
UAP トレース情報の取得で，同一テストユーザ ID でトレース情報を出力する UAP を並行して実行した場合の説明について内容を変更した。
utotrcout コマンドの-n オプションの説明を変更した。
退避コアファイルのディレクトリとファイル名および UAP トレース編集出力ファイルのファイル名の注意書きを変更した。
uatdump コマンドについて次を変更した。 • オプション • 出力メッセージ

追加・変更内容
• 注意事項

変更内容（3000-3-D57-20） 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
プロセスをアボートしなくても UAP トレース（UAP トレースデータファイル）を取得できるようにした。 これに伴い，uatdump コマンドに-f オプションを追加した。

変更内容（3000-3-D57-10） uCosminexus TP1/Server Base 07-02

追加・変更内容
rpc_trace_name 定義オペランドのパス名の環境変数指定について，説明を追加した。

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 のテスタおよび UAP トレースの機能と使い方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。

■ 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，このマニュアルは，マニュアル「OpenTP1 解説」を前提としていますので，あらかじめお読みいただくことをお勧めします。

■ マニュアルの構成

このマニュアルは，次に示す編と章から構成されています。

第 1 編　テスタ・UAP トレースの概要

第 1 章　概要

テスタの種類と概要，および UAP トレースの概要について説明しています。

第 2 編　オンラインテスタ（TP1/Online Tester）

第 2 章　機能

TP1/Server Base のオンラインテスタ（TP1/Online Tester）の機能について説明しています。

第 3 章　テスト環境の設定

TP1/Online Tester で実行するテスト環境を設定するための定義情報について説明しています。

第 4 章　テストの実行

テストで使用する UAP の作成方法，サービスの要求方法，およびテスト情報の編集方法について説明しています。

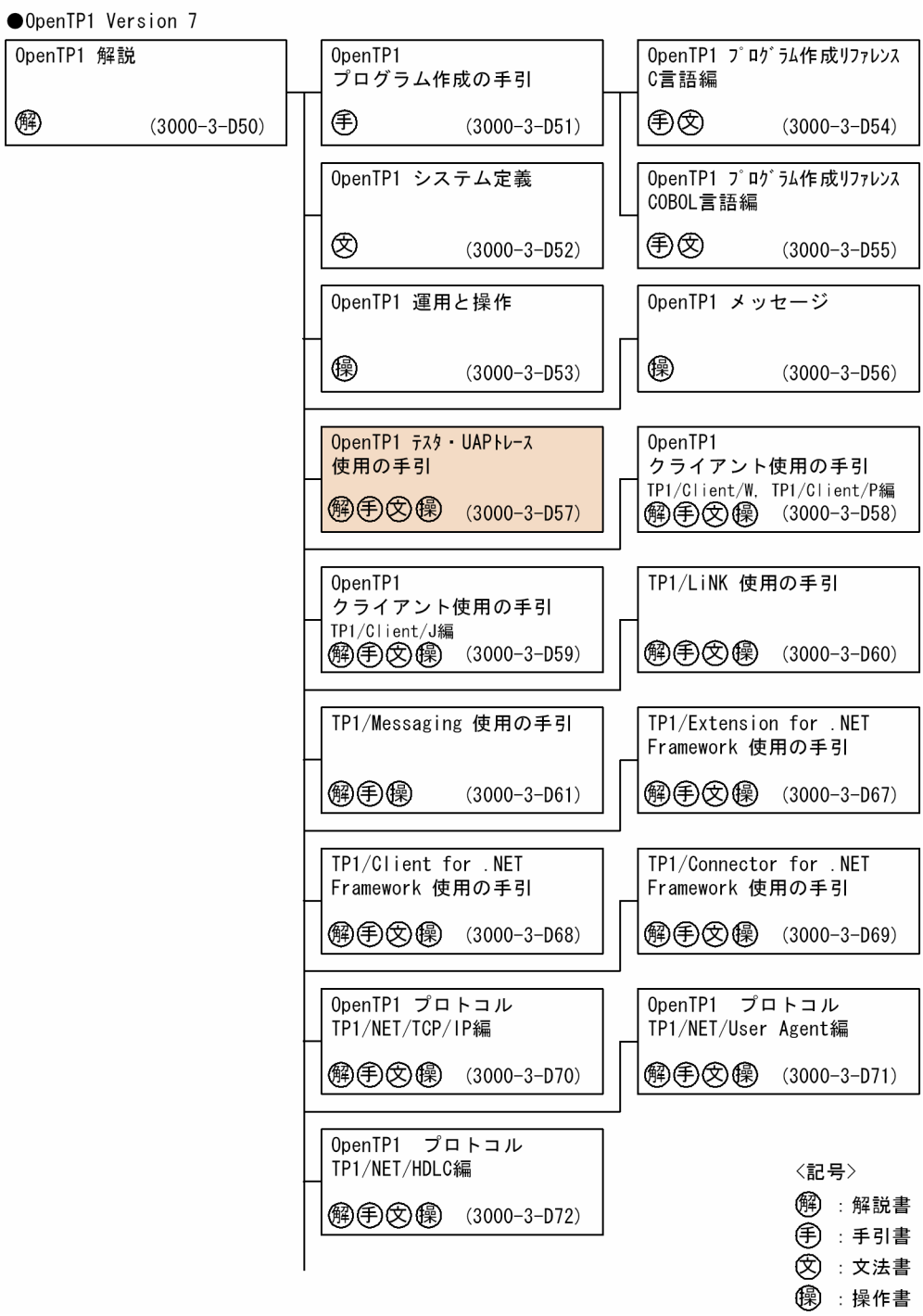
第 5 章　運用コマンド

テストで使用する運用コマンドについて説明しています。

第 6 章　障害対策

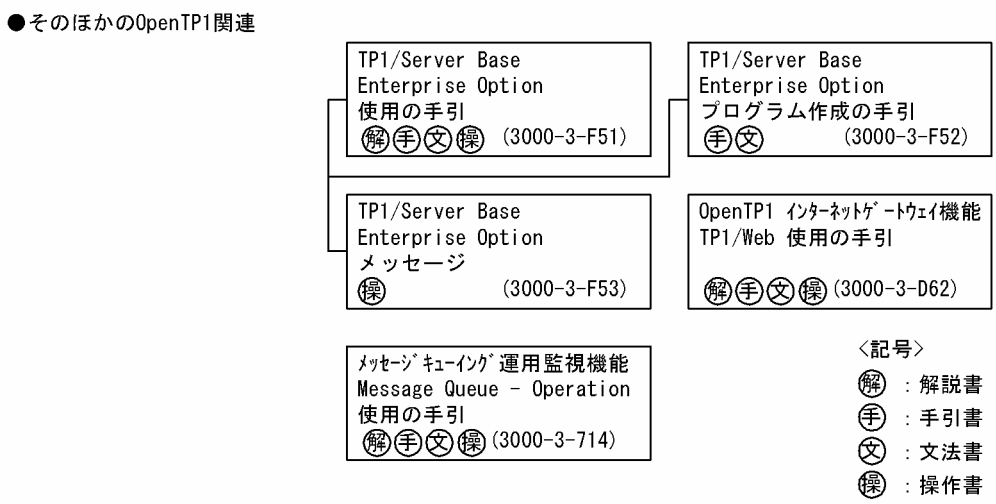
TP1/Online Tester に関連して発生する障害とその対策について説明しています。

■ 関連マニュアル


●OpenTP1 Version 7
OpenTP1 解説
㊎ (3000-3-D50)
OpenTP1
プログラム作成の手引
㊍ (3000-3-D51)
OpenTP1 システム定義
㊏ (3000-3-D52)
OpenTP1 運用と操作
㊐ (3000-3-D53)
OpenTP1 ﾃｽﾀ・UAPﾄﾚｰｽ
使用の手引
㊎㊍㊏㊐ (3000-3-D57)
OpenTP1
クライアント使用の手引
TP1/Client/J編
㊎㊍㊏㊐ (3000-3-D59)
TP1/Messaging 使用の手引
㊎㊍㊐ (3000-3-D61)
TP1/Client for .NET
Framework 使用の手引
㊎㊍㊏㊐ (3000-3-D68)
OpenTP1 プロトコル
TP1/NET/TCP/IP編
㊎㊍㊏㊐ (3000-3-D70)
OpenTP1 プロトコル
TP1/NET/HDLC編
㊎㊍㊏㊐ (3000-3-D72)
OpenTP1 ﾌﾟﾛｸﾞﾗﾑ作成ﾘﾌｧﾚﾝｽ
C言語編
㊍㊏ (3000-3-D54)
OpenTP1 ﾌﾟﾛｸﾞﾗﾑ作成ﾘﾌｧﾚﾝｽ
COBOL言語編
㊍㊏ (3000-3-D55)
OpenTP1 メッセージ
㊐ (3000-3-D56)
OpenTP1
クライアント使用の手引
TP1/Client/W, TP1/Client/P編
㊎㊍㊏㊐ (3000-3-D58)
TP1/LiNK 使用の手引
㊎㊍㊏㊐ (3000-3-D60)
TP1/Extension for .NET
Framework 使用の手引
㊎㊍㊏㊐ (3000-3-D67)
TP1/Connector for .NET
Framework 使用の手引
㊎㊍㊏㊐ (3000-3-D69)
OpenTP1 プロトコル
TP1/NET/User Agent編
㊎㊍㊏㊐ (3000-3-D71)
〈記号〉
㊎ : 解説書
㊍ : 手引書
㊏ : 文法書
㊐ : 操作書


OpenTP1　プロトコル
TP1/NET/OSI-TP編
解手文操　(3000-3-D73)
OpenTP1　プロトコル
TP1/NET/XMAP3編
解手文操　(3000-3-D74)
OpenTP1 プロトコル
TP1/NET/User Datagram
Protocol編
解手文操　(3000-3-D75)
OpenTP1　プロトコル
TP1/NET/X25編
解手文操　(3000-3-D76)
OpenTP1　プロトコル
TP1/NET/HSC編
解手文操　(3000-3-D77)
OpenTP1 プロトコル
TP1/NET/NCSB編
解手文操　(3000-3-D78)
OpenTP1　プロトコル
TP1/NET/OSAS-NIF編
解手文操　(3000-3-D79)
OpenTP1　プロトコル
TP1/NET/Secondary
Logical Unit - TypeP2編
解手文操　(3000-3-D80)
OpenTP1　プロトコル
TP1/NET/X25-Extended編
解手文操　(3000-3-D82)
OpenTP1 ﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞ機能
TP1/Message Queue
使用の手引
解手文操　(3000-3-D90)
OpenTP1 ﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞ機能
TP1/Message Queue
メッセージ
操　(3000-3-D91)
OpenTP1 ﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞ機能
TP1/Message Queue
プログラム作成の手引
手　(3000-3-D92)
OpenTP1 ﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞ機能
TP1/Message Queue
プログラム作成リファレンス
文　(3000-3-D93)
ﾒｯｾｰｼﾞｷｭｰｲﾝｸﾞｱｸｾｽ機能
TP1/Message Queue Access
使用の手引
解手文操　(3000-3-D94)
〈記号〉
解　：解説書
手　：手引書
文　：文法書
操　：操作書

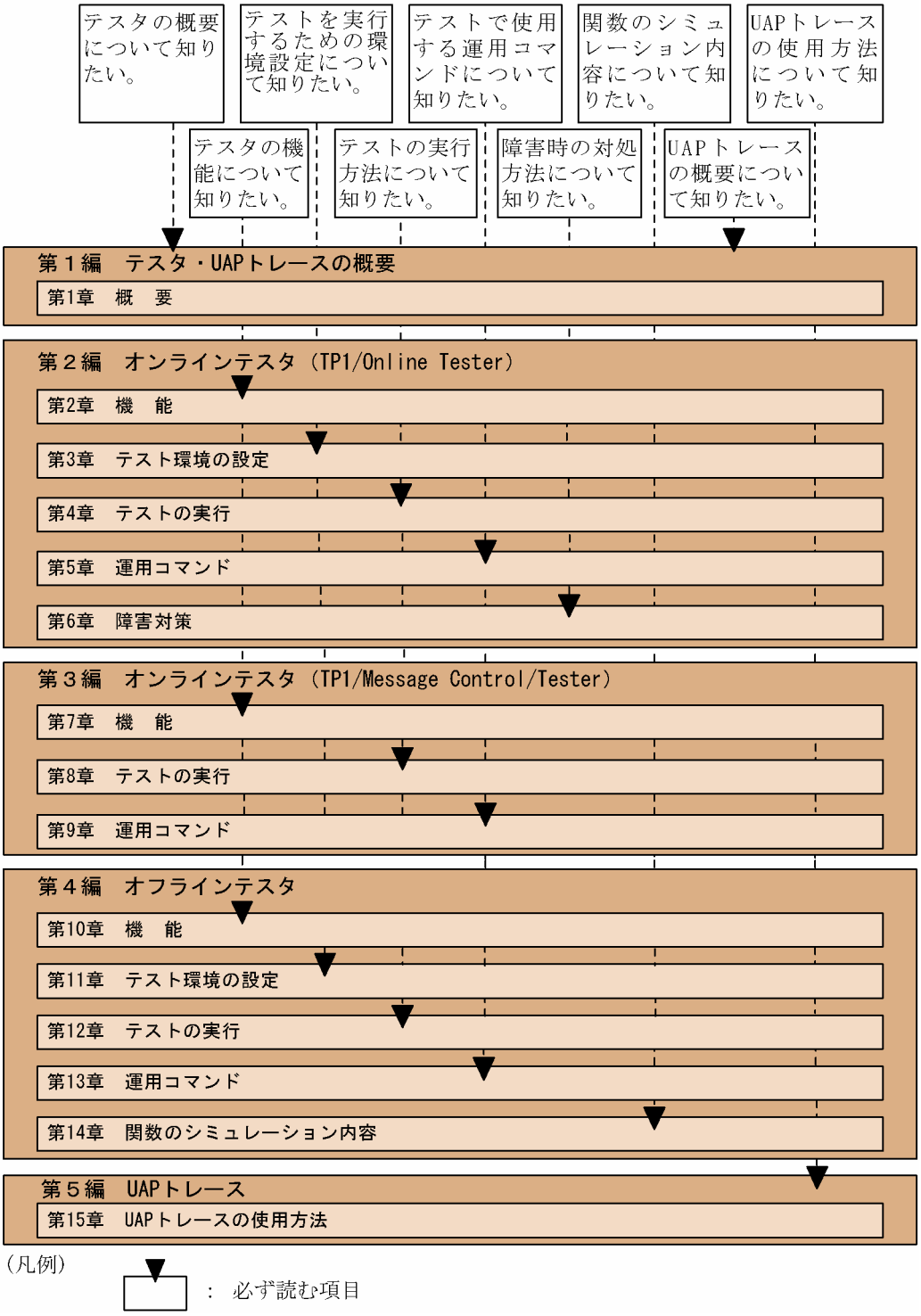
●そのほかのOpenTP1関連



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。

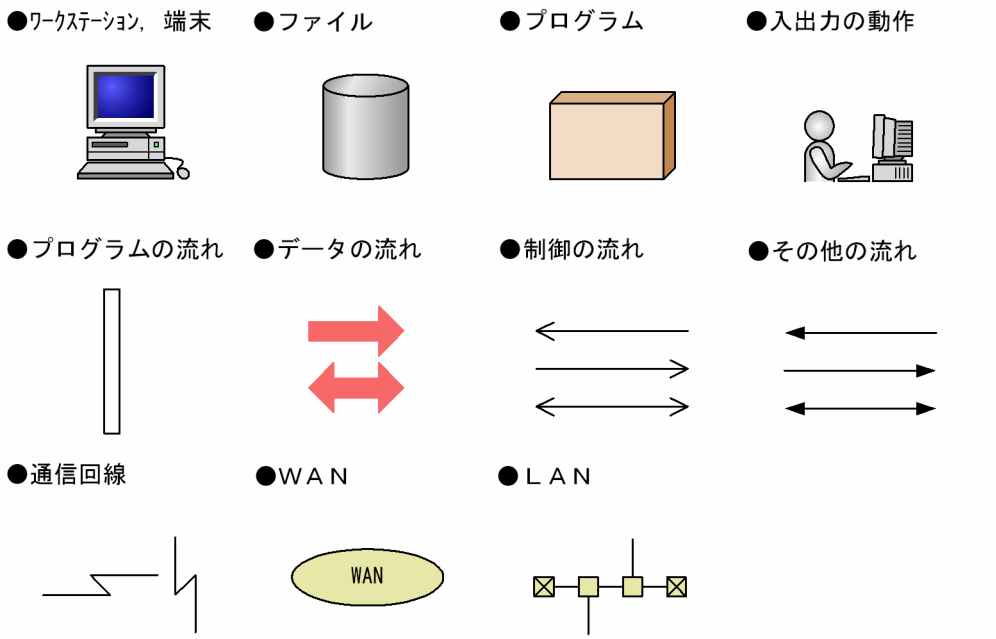
■ 読書手順

このマニュアルは，利用目的に合わせて章を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。



■ 図中で使用する記号

このマニュアルで使用する記号を，次のように定義します。



■ 文法の記号

このマニュアルで使用する各種記号を説明します。

(1)文法記述記号

文法の記述形式について説明する記号です。

文法記述記号	意味
{ } 波括弧	この記号で囲まれている複数の項目のうちから一つを選択できることを示します。 （例）start {SPP|MHP} SPP，MHP の二つのうちから，どちらか一つを指定します。 また，次の場合は，短縮して指定する場合の指定値であることを示します。 （例）mcftules -e {backout|ba} -e オプションのフラグ引数を backout と指定するか，または短縮して ba と指定します。
〔 〕 きっ甲	この記号で囲まれている項目は，省略できることを示します。 （例）utfstart 〔-s〕 -s は省略できることを示します。
| ストローク	この記号で区切られた項目は選択できることを示します。 （例）stop {SPP|MHP} SPP か MHP を指定できることを示します。
＿ 下線	きっ甲〔 〕で囲まれた複数項目のうちで，きっ甲内のすべてを省略したときに，OpenTP1 で仮定する標準値を示します。

文法記述記号	意味
_ 下線	（例）[uoc_conf=Y|N] uoc_conf オペランドを省略した場合，N が仮定されます。
… 点線	記述が省略されていることを示します。この記号の直前に示された項目を繰り返し複数個指定できます。 （例）utols サーバ名 ［サーバ名］… サーバ名を続けて指定できることを示します。

(2)属性表示記号

ユーザ指定値の範囲などを説明する記号の一覧を示します。

属性表示記号	意味
～	この記号のあとにユーザ指定値の属性を示します。
《　》	ユーザが指定を省略したときの値を示します。
〈　〉	ユーザ指定値の構文要素を示します。
((　))	ユーザ指定値の指定範囲を示します。

(3)構文要素

説明で使用する構文要素の一覧を示します。すべて**半角文字**で指定します。

構文要素	意味
英字	アルファベット（A～Z，a～z），＿（アンダスコア）
英数字	英字と数字（0～9）
英字記号	アルファベット（A～Z，a～z），#，@，¥
英数字記号	英字記号と数字（0～9）
符号なし整数	数字（0～9）
16 進数	数字（0～9），A～F，a～f
識別子	先頭がアルファベット（A～Z，a～z）で始まる英数字列
記号名称	先頭が英字記号で始まる英数字記号列
パス名	記号名称，／，およびピリオド（.） ただし，パス名は使用する OS に依存します。

■ このマニュアルでの表記

(1)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 例　$DCDIR	%aaaaaa% 例　%DCDIR%
パス名の区切り文字	:	;
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。 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。 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。 例　mcfmngrd	ファイル名に拡張子を付加して指定します。 例　mcfmngrd.exe
make コマンド	make	nmake

(2)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

■ 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

英略語	英字での表記
C/S	Client/Server
CRM	Communication Resource Manager
CUP	Client User Program
DAM	Direct Access Method
DBMS	Database Management System
DML	Data Manipulation Language
DNS	Domain Name System
FEP	Front End Processor
GUI	Graphical User Interface
HA	High Availability
HI-ODTP	Hitachi - Open Distributed Transaction Processing Adapter
ISAM	Indexed Sequential Access Method
IST	Internode Shared Table
LAN	Local Area Network
MCF	Message Control Facility
MHP	Message Handling Program
MQA	Message Queue Access
MQI	Message Queue Interface
NIF/HNA	Network Interface Feature/Hitachi Network Architecture
NIF/OSI	Network Interface Feature/OSI
OS	Operating System
OSI	Open Systems Interconnection
OSI TP	Open Systems Interconnection Transaction Processing
PC	Personal Computer
PRF	Performance
RM	Resource Manager
RPC	Remote Procedure Call
SPP	Service Providing Program
STDL	Structured Transaction Definition Language

英略語	英字での表記
SUP	Service Using Program
TAM	Table Access Method
TCP/IP	Transmission Control Protocol/Internet Protocol
TM	Transaction Manager
UAP	User Application Program
UOC	User Own Coding
WAN	Wide Area Network
WS	Workstation

■ 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。

■ その他の前提条件

このマニュアルをお読みになる際のその他の前提情報については，マニュアル「OpenTP1 解説」を参照してください。

目次

1 概要

OpenTP1 が提供する各種テスタと，UAP トレースの概要について説明します。

1.1 テスタと UAP トレース

OpenTP1 には，UAP の動作を確認するためのテスト用のプログラムとして，**テスト支援プログラム（テスタ）**があります。また，UAP の動作を調べるためのトラブルシュート用の機能として，**UAP トレース機能**があります。

なお，テスタには，オンライン環境下で使用する**オンラインテスタ**と，オフライン環境下で使用する**オフラインテスタ**があります。さらに，オンラインテスタには，**TP1/Server Base のオンラインテスタ**と **TP1/Message Control のオンラインテスタ**の二つがあります。

UAP トレース機能は TP1/Server Base で使用できます。

使用するテスタによって，次に示すプログラムプロダクトが必要です。

TP1/Server Base のオンラインテスタを使用する場合

TP1/Online Tester

TP1/Message Control のオンラインテスタを使用する場合

TP1/Message Control/Tester

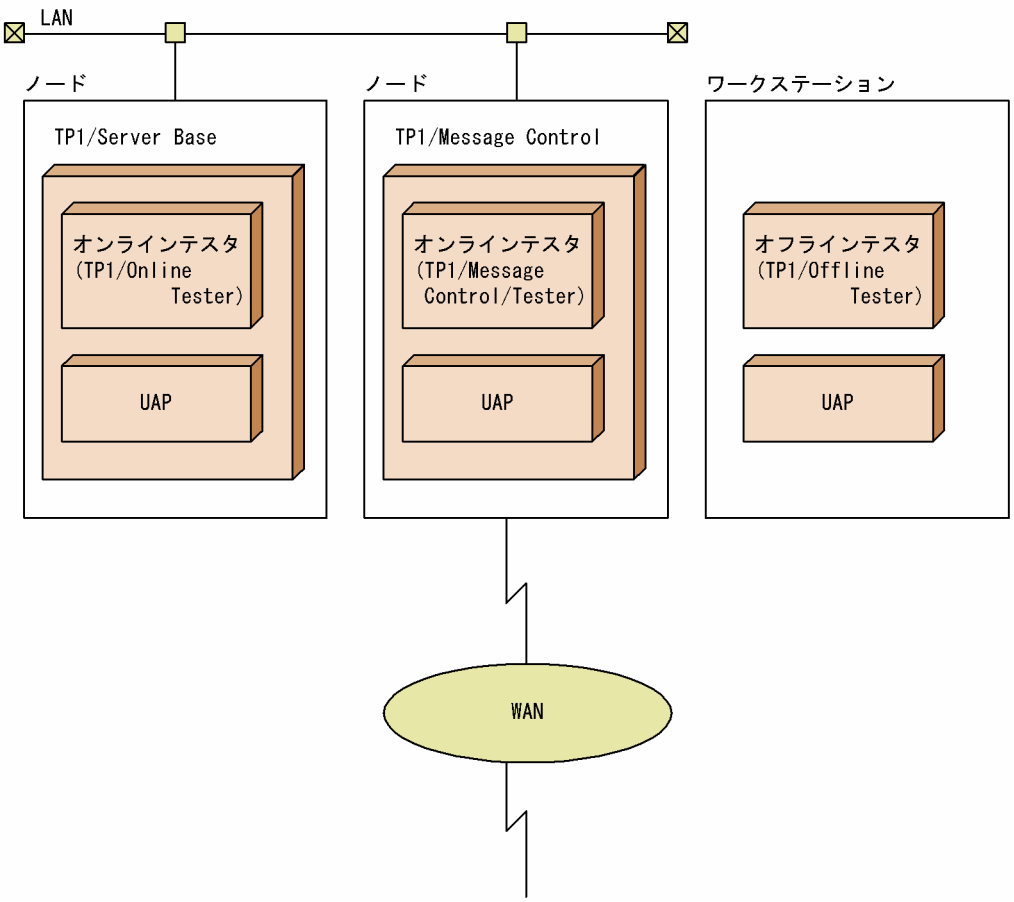
オフラインテスタを使用する場合

TP1/Offline Tester

1.2　テスタの種類と概要

OpenTP1 のテスタを次の図に示します。

図 1-1　OpenTP1 のテスタ



1.2.1　オンラインテスタ

(1)　オンラインテスタ（TP1/Online Tester）

TP1/Server Base のオンラインテスタ（以降，**オンラインテスタ**と呼びます）で使用できる機能を次に示します。各機能の説明や使用方法については，「第 2 編　オンラインテスタ（TP1/Online Tester)」を参照してください。

- クライアント UAP，サーバ UAP のシミュレート
- TP1/Message Control のシミュレート
- 資源更新処理の無効化
- UAP から発行する運用コマンドのシミュレート
- テスタファイル（テスト用データファイル）の作成・編集出力

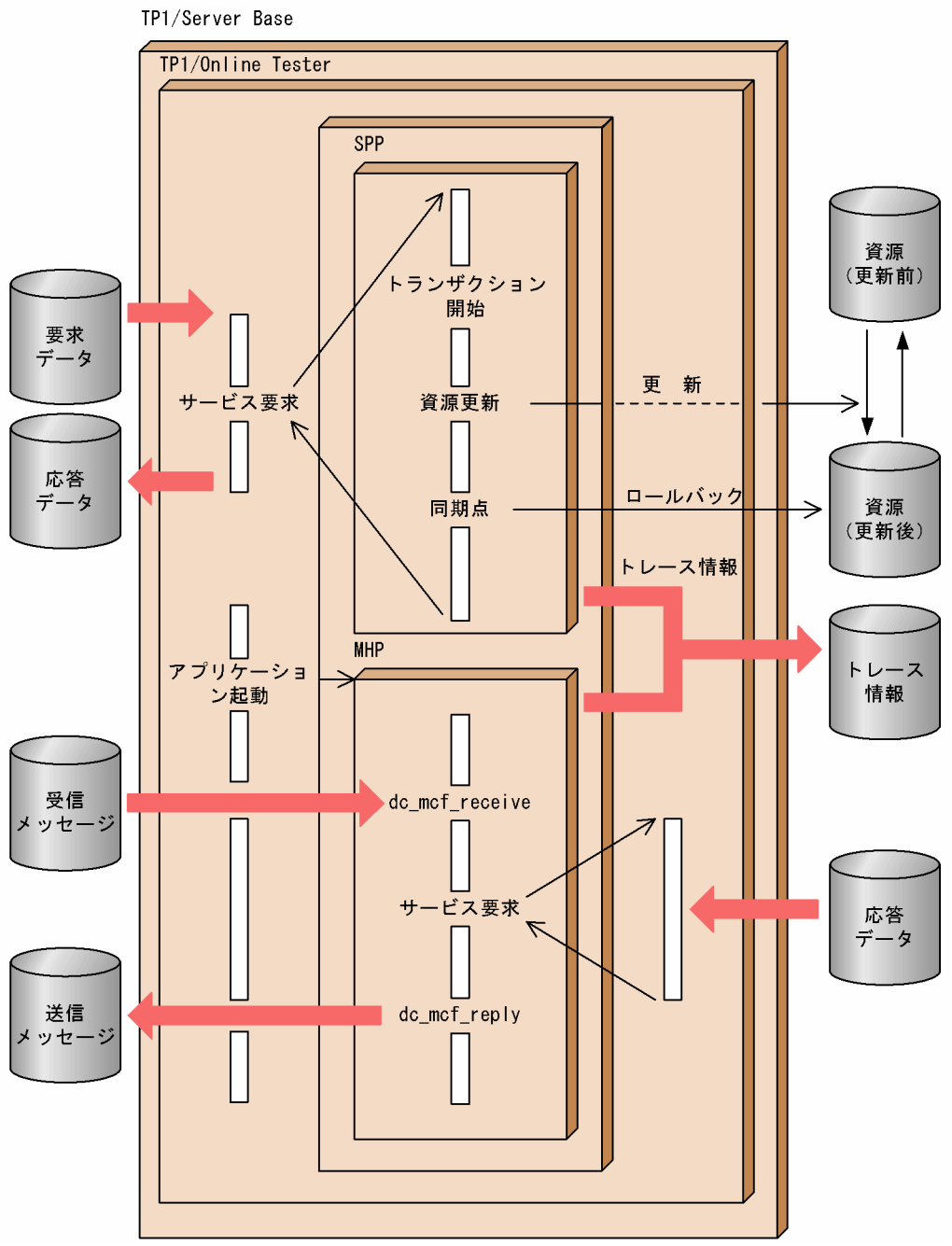
- UAP トレース情報の取得・編集出力
- UAP 送信メッセージの取得・編集
- デバッガの連動

オンラインテスタを使用すると，オンライン環境下で SUP，SPP，および MHP をテストし，動作を確認できます。

なお，オンラインテスタでは，TP1/Server Base を前提としています。

オンラインテスタの概要を次の図に示します。

図 1-2　オンラインテスタの概要



(2)　オンラインテスタ（TP1/Message Control/Tester）

TP1/Message Control のオンラインテスタ（以降，**MCF オンラインテスタ**と呼びます）で使用できる機能を次に示します。各機能の説明や使用方法については，「第 3 編　オンラインテスタ（TP1/Message Control/Tester）」を参照してください。

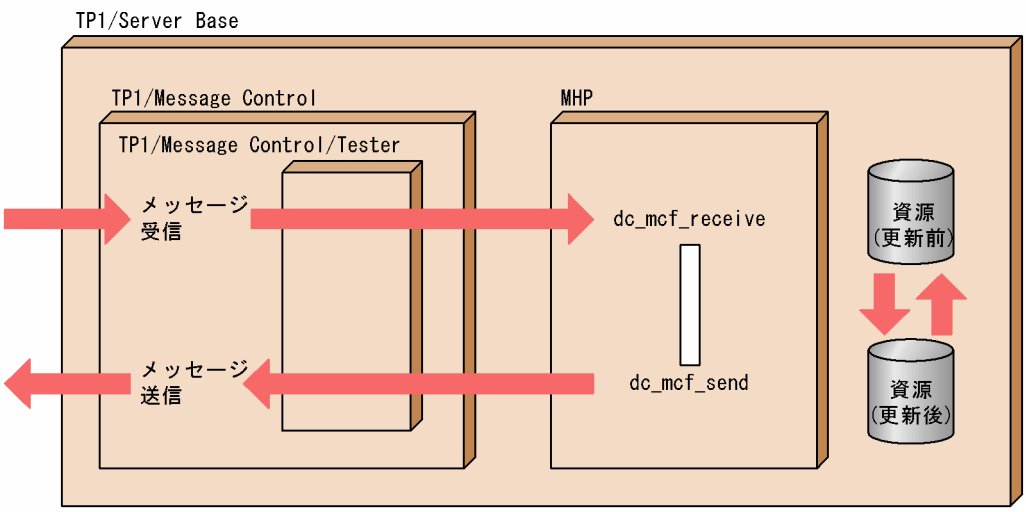
- TP1/Message Control 以外の資源更新処理の無効化
- 送信メッセージの無効化
- アプリケーション起動メッセージの無効化
- エラーイベントの抑止
- MHP の自動閉塞機能の抑止
- UAP トレース情報の取得

MCF オンラインテスタでは，UAP トレース情報を取得する場合を除いて，オンラインテスタ（TP1/Online Tester）は不要です。

同じ MHP が同時に，MCF オンラインテスタとオンラインテスタのテスト対象となった場合は，MCF オンラインテスタの指定が優先されます。

MCF オンラインテスタの概要を次の図に示します。

図 1-3　MCF オンラインテスタの概要



1.2.2　オフラインテスタ（TP1/Offline Tester）

オフラインテスタで使用できる機能を次に示します。各機能の説明や使用方法については，「第 4 編　オフラインテスタ」を参照してください。

- クライアント UAP，サーバ UAP のシミュレート

- TP1/Message Control のシミュレート
- ファイルサービスのシミュレート
- UAP から発行する運用コマンドのシミュレート
- テスタファイル（テスト用データファイル）の作成
- コマンドの連続実行
- デバッガの連動
- オフラインテスタトレース情報の取得

オフラインテスタを使用すると，オフライン環境下で SPP，および MHP をテストし，動作を確認できます。

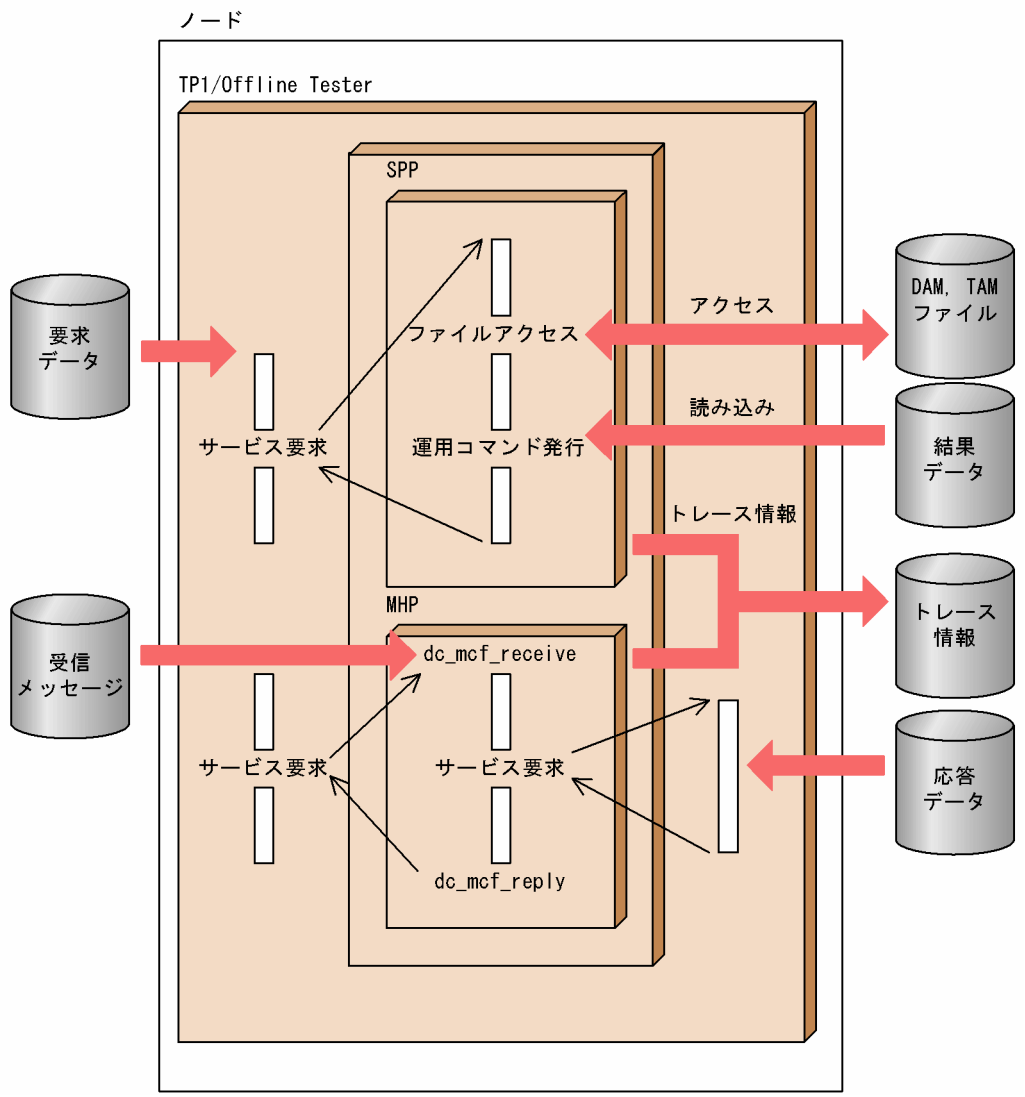
テスト中に使用する関数によって，次のプログラムの提供するヘッダファイルを使用して UAP をコンパイルします。

- TP1/Server Base
  TP1/Server Base の提供する関数を使用する場合
- TP1/Message Control
  メッセージ送受信関数を使用する場合
- TP1/FS/Direct Access
  DAM サービス関数を使用する場合
- TP1/FS/Table Access
  TAM サービス関数を使用する場合
- TPI/Shared Table Access
  IST サービス関数を使用する場合

また，オフラインテスタで使用する UAP を作成するために，TP1/Server Base の **stbmake** コマンドが必要です。UAP 作成時には，TP1/Server Base の stbmake コマンドのファイルをコピーして使用してください。

オフラインテスタの概要を次の図に示します。

図 1-4　オフラインテスタの概要



1.3　UAP トレースの概要

OpenTP1 では UAP の障害に備えて，UAP で使用したライブラリ関数の発行履歴を取得しています。この取得した情報から，エラーリターンした関数や，UAP がアクセスしようとした資源がわかります。この情報を編集出力することによって，ユーザは UAP に障害が起こった原因を解析でき，UAP を修正したり，システムを再構築したりする目安にできます。この機能を **UAP トレース**といいます。

UAP トレースでは，情報を SUP，SPP，および MHP が稼働していたプロセス単位で取得します。

UAP トレースは，UAP が異常終了した際に次のファイルがあれば，自動でファイルに編集出力されます。

- **UAP トレースデータファイル**
- **退避コアファイル**

また，TP1/Server Base の **uatdump** コマンドで，ユーザが標準出力に編集出力させることもできます。uatdump コマンドについては，「15.2.2　UAP トレースを編集出力する方法」を参照してください。

UAP トレースは，オンラインテスタやオフラインテスタを使った，UAP のテストにも使用できます。UAP のテストでは，UAP トレースでテスト処理の流れを解析できます。

オンラインテスタでは，TP1/Server Base の UAP トレースでトレース情報を取得します。オフラインテスタでは，専用のトレース情報を取得します。

2 機能

オンラインテスタで実行するテストのための，各種機能について説明します。

2.1　オンラインテスタの機能

オンラインテスタには，UAP をテストするための次のような機能があります。

1. クライアント UAP シミュレート機能

   クライアント UAP がなくてもサーバ UAP がテストできるよう，クライアント UAP の処理をシミュレートする機能です。

2. サーバ UAP シミュレート機能

   サーバ UAP がなくてもクライアント UAP がテストできるよう，サーバ UAP の処理をシミュレートする機能です。

3. MCF シミュレート機能

   MCF（TP1/Message Control）がなくても，MHP および MHP からサービス要求される SPP がテストできるよう，MCF が制御するメッセージ送受信処理をシミュレートする機能です。

4. 資源更新処理無効化機能

   テストする UAP が実業務用の資源を更新しないよう，資源の更新処理を無効化する機能です。

5. 運用コマンドシミュレート機能

   テストする UAP が運用コマンドを発行する場合に，コマンドの処理をシミュレートする機能です。

6. テスタファイル作成・編集機能

   各シミュレート機能使用時に必要となるテスタファイルを作成し，編集出力する機能です。

7. UAP トレース情報取得機能

   テストする UAP の UAP トレース情報を取得する機能です。

8. UAP トレース情報マージ・編集機能

   複数ファイルに取得した UAP トレース情報をマージし，編集出力する機能です。

9. 送信メッセージ編集機能

   テストする UAP の送信メッセージを取得し，編集出力する機能です。

10. デバッガ連動機能

    テストする UAP をデバッガの制御下で動作させる機能です。

2.2　クライアント UAP のシミュレート

オンラインテスタでは，クライアント UAP の代わりとなってサーバ UAP にサービスを要求できます。そのため，ユーザはクライアント UAP がなくてもサーバ UAP をテストできます。これを，**クライアント UAP シミュレート機能**といいます。

クライアント UAP のシミュレートは，オンラインテスタのコマンドで実行します。ユーザは，サーバ UAP に渡す処理データを，あらかじめ**サービス要求データファイル**に作成しておきます。サーバ UAP からの応答データは，コマンドで指定したファイル（**サービス応答データファイル**）に取得されます。

サービス要求データファイルには，シミュレートする UAP のインタフェース別に，次の 2 種類があります。

- RPC 要求データファイル（RPC インタフェースの UAP シミュレート時）
- XATMI 要求データファイル（XATMI インタフェースの UAP シミュレート時）

また，サービス応答データファイルにも，シミュレートする UAP のインタフェース別に，次の 2 種類があります。

- RPC 応答データファイル（RPC インタフェースの UAP シミュレート時）
- XATMI 応答データファイル（XATMI インタフェースの UAP シミュレート時）

クライアント UAP シミュレート機能を使用してサーバ UAP をテストする場合，テストするサーバ UAP が**テスト専用 UAP** であることを，ユーザサービス定義で定義しておきます。テスト専用 UAP とは，テストの対象となる UAP として，オンラインテスタのすべての機能を使用できる（**テストモード**で動作する）UAP のことです。

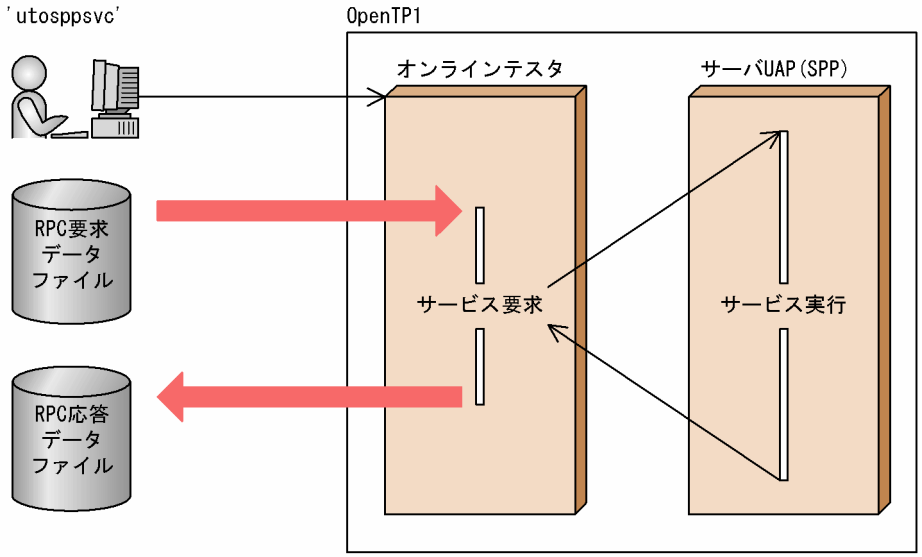
また，ユーザサービス定義では，テスト専用 UAP の代わりに使用可能 UAP として定義してもかまいません。使用可能 UAP とは，テスト対象 UAP からサービス要求される場合にだけテストモードで動作する SPP のことです。

2.2.1　RPC インタフェースのクライアント UAP のシミュレート

RPC インタフェースに従ってサービス要求するクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ **RPC 要求データファイル**に作成しておきます。サーバ UAP からの応答データは，コマンドで指定したファイル（**RPC 応答データファイル**）に取得します。

RPC インタフェースのクライアント UAP シミュレート機能の概要を，次の図に示します。

図 2-1　RPC インタフェースのクライアント UAP のシミュレート



2.2.2　XATMI インタフェースのクライアント UAP のシミュレート

オンラインテスタでは，XATMI インタフェースに従ったサービス要求（リクエスト/レスポンス型，会話型）の際にも，クライアント UAP シミュレート機能を使用できます。

(1)　リクエスト/レスポンス型のサービス要求

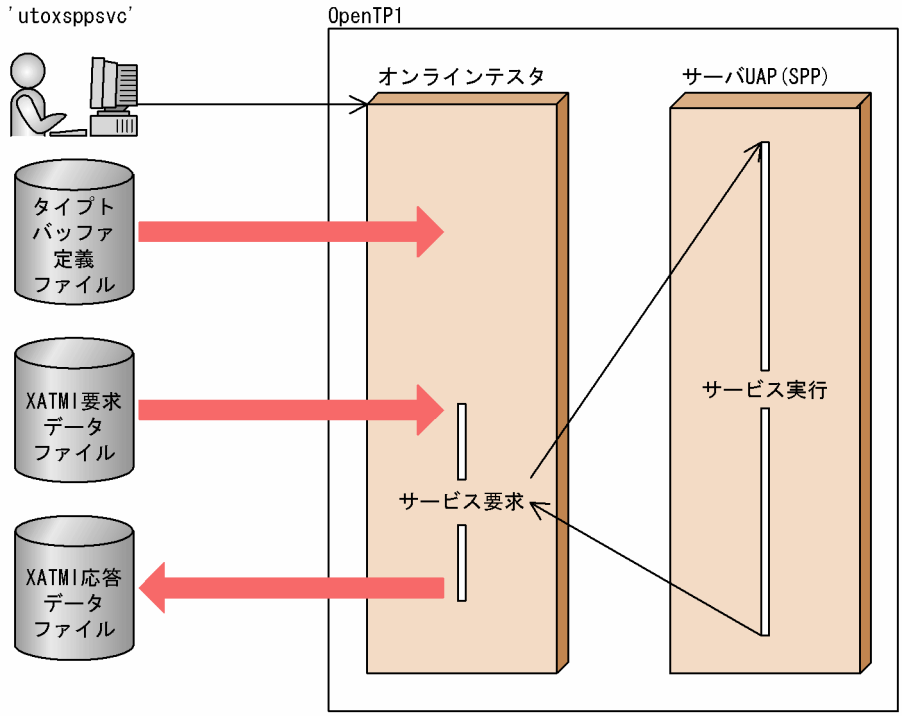
リクエスト/レスポンス型のサービス要求でクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ **XATMI 要求データファイル**に作成しておきます。サーバ UAP からの応答データは，コマンドで指定したファイル（**XATMI 応答データファイル**）に取得します。

XATMI を使用する際に必要となるタイプトバッファの情報は，**タイプトバッファ定義ファイル**に設定しておきます。

なお，XATMI 要求データファイルのヘッダ部分には，リクエスト/レスポンス型でサービス要求する関数の種別を，あらかじめ設定しておく必要があります。

リクエスト/レスポンス型のサービス要求でのクライアント UAP シミュレート機能の概要を，次の図に示します。

図 2-2　リクエスト/レスポンス型のサービス要求でのクライアント UAP のシミュレート



(2)　会話型のサービス要求

会話型のサービス要求でクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ XATMI 要求データファイルに作成しておきます。XATMI 要求データファイルのヘッダ部分には，会話型でサービス要求する関数の種別を，あらかじめ設定しておきます。サーバ UAP からの応答データは，コマンドで指定したファイル（XATMI 応答データファイル）に取得します。

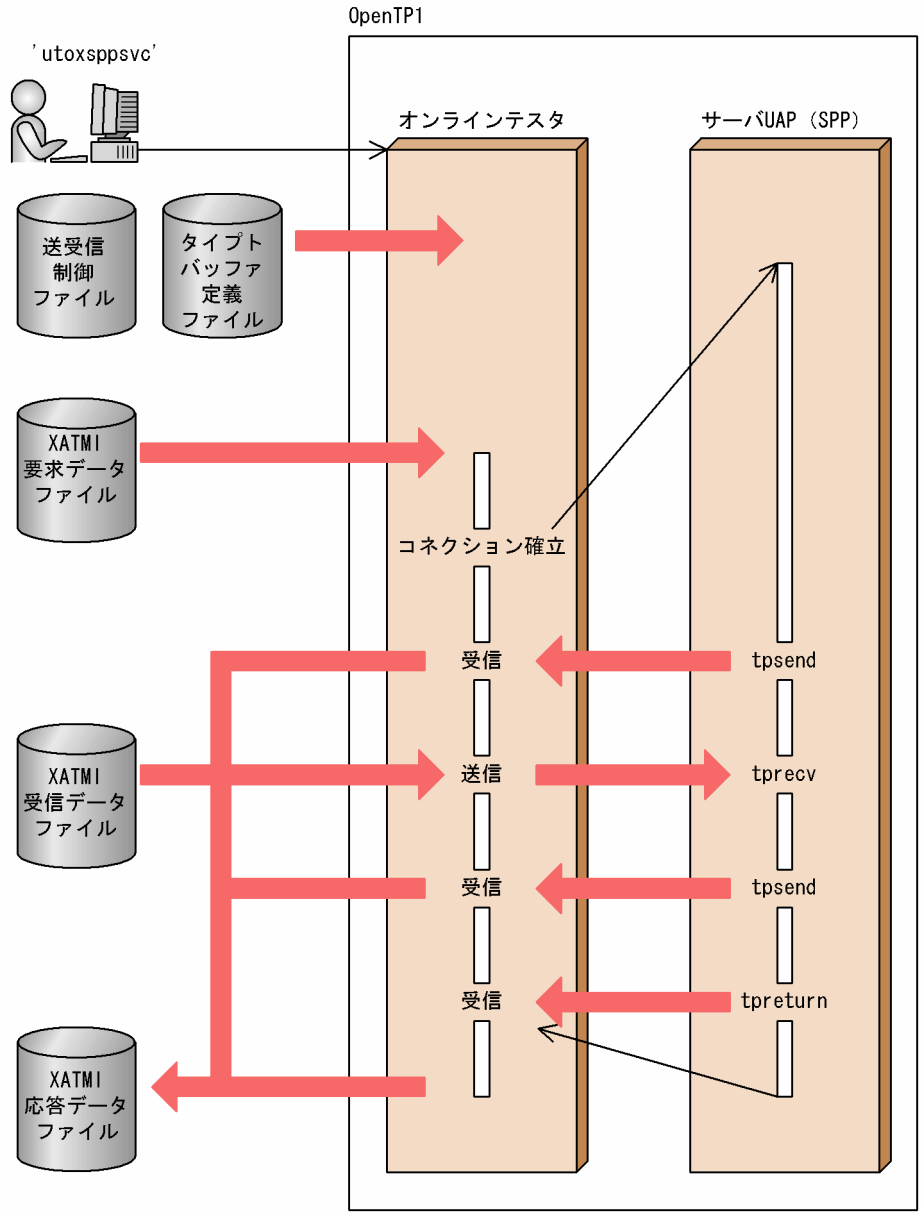
XATMI を使用する際に必要となるタイプトバッファの情報は，タイプトバッファ定義ファイルに設定しておきます。

また，ユーザは，サービス中の送受信手順を，**送受信制御ファイル**に設定しておきます。テストするサーバ UAP がサービス中に受信するデータは，あらかじめ **XATMI 受信データファイル**に作成し，送受信制御ファイルでファイル名を指定しておきます。サーバ UAP が送信したデータは，応答データと同様に XATMI 応答データファイルに取得します。

なお，クライアント UAP シミュレート機能でファイルに取得したサーバ UAP からの応答データや送信データは，サーバ UAP シミュレート機能でクライアント UAP への要求データや受信データとして使用できます。その際には，応答データや送信データを取得した XATMI 応答データファイルの構造を，XATMI 要求データファイルや XATMI 受信データファイルの構造にバイナリエディタで修正してから使用してください。

会話型のサービス要求でのクライアント UAP シミュレート機能の概要を，次の図に示します。

図 2-3　会話型のサービス要求でのクライアント UAP のシミュレート



2.3 サーバ UAP のシミュレート

オンラインテスタでは，サーバ UAP の代わりとなってクライアント UAP が要求するサービスを実行できます。そのため，ユーザはサーバ UAP がなくてもクライアント UAP をテストできます。これを，**サーバ UAP シミュレート機能**といいます。

サーバ UAP のシミュレートは，OpenTP1 のコマンドで，シミュレートするサーバ UAP を起動して実行します。ユーザは，あらかじめクライアント UAP に渡す応答データを，**サービス応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オンラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

サービス応答データファイルには，シミュレートする UAP のインタフェース別に，次の 2 種類があります。

- RPC 応答データファイル（RPC インタフェースの UAP シミュレート時）
- XATMI 応答データファイル（XATMI インタフェースの UAP シミュレート時）

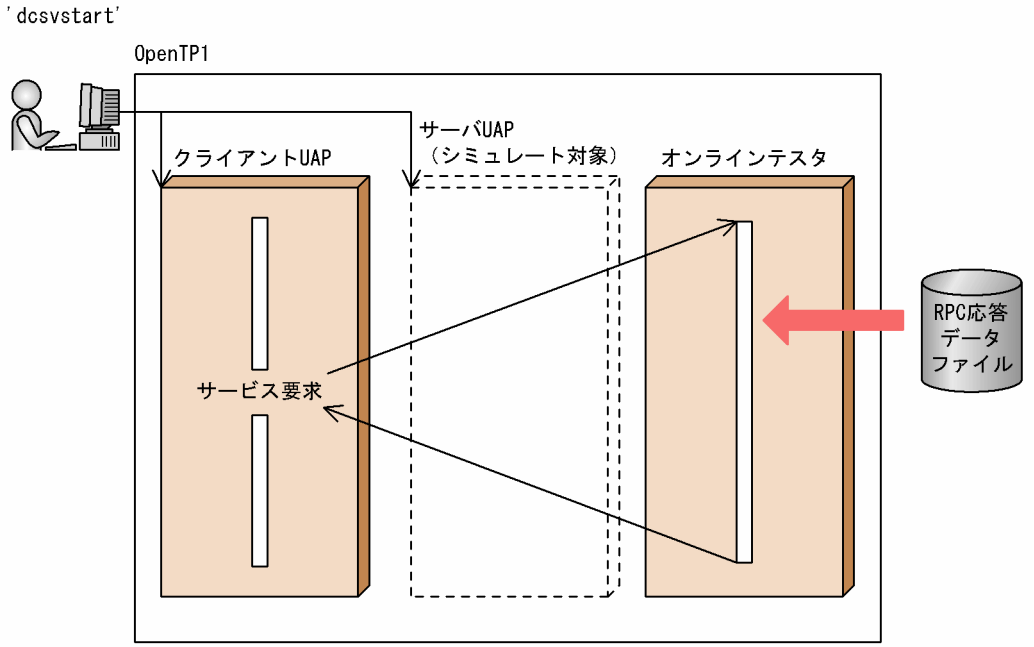
サーバ UAP シミュレート機能を使用してクライアント UAP をテストする場合，シミュレートするサーバ UAP が**ダミー SPP** であることを，ユーザサービス定義で定義し，あらかじめ起動しておきます。ダミー SPP とは，サーバ UAP シミュレート機能使用時に起動しても，実際にはプロセスが生成されない SPP のことです。

2.3.1 RPC インタフェースのサーバ UAP のシミュレート

RPC インタフェースに従ってサービス要求されるサーバ UAP をシミュレートする場合，ユーザは，あらかじめクライアント UAP へ返す応答データを，**RPC 応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オンラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

RPC インタフェースのサーバ UAP シミュレート機能の概要を，次の図に示します。

図 2-4　RPC インタフェースのサーバ UAP のシミュレート



2.3.2　XATMI インタフェースのサーバ UAP のシミュレート

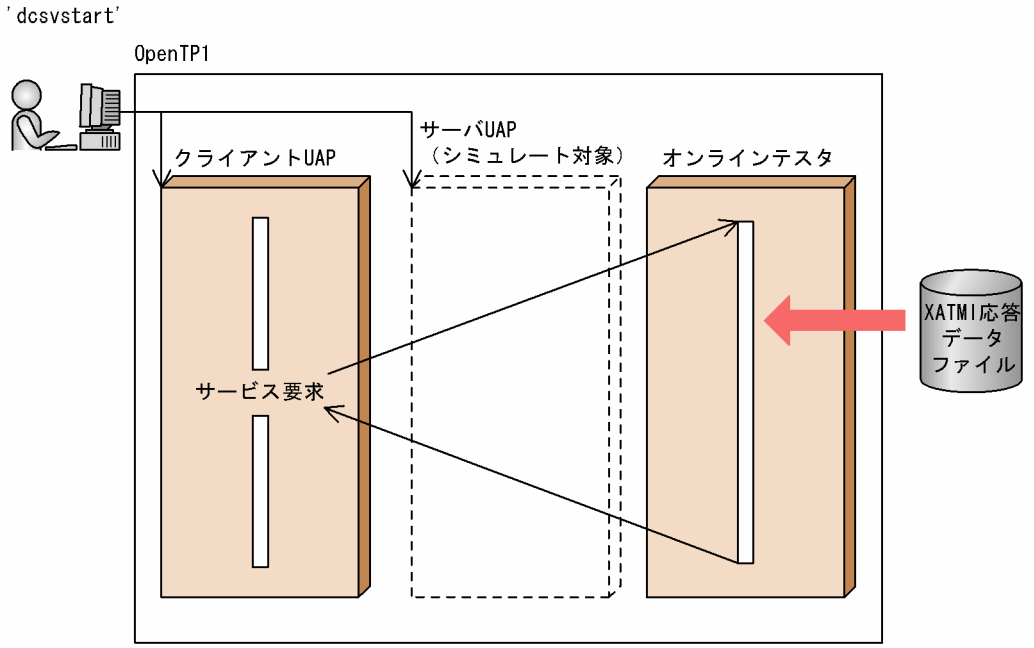
オンラインテスタでは，XATMI インタフェースに従ったサービス要求（リクエスト/レスポンス型，会話型）の際にも，サーバ UAP シミュレート機能を使用できます。

(1)　リクエスト/レスポンス型のサービス要求

リクエスト/レスポンス型のサービス要求でサーバ UAP をシミュレートする場合，ユーザは，クライアント UAP へ返す応答データを，あらかじめ **XATMI 応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オンラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

リクエスト/レスポンス型のサービス要求でのサーバ UAP シミュレート機能の概要を，次の図に示します。

図 2-5　リクエスト/レスポンス型のサービス要求でのサーバ UAP のシミュレート



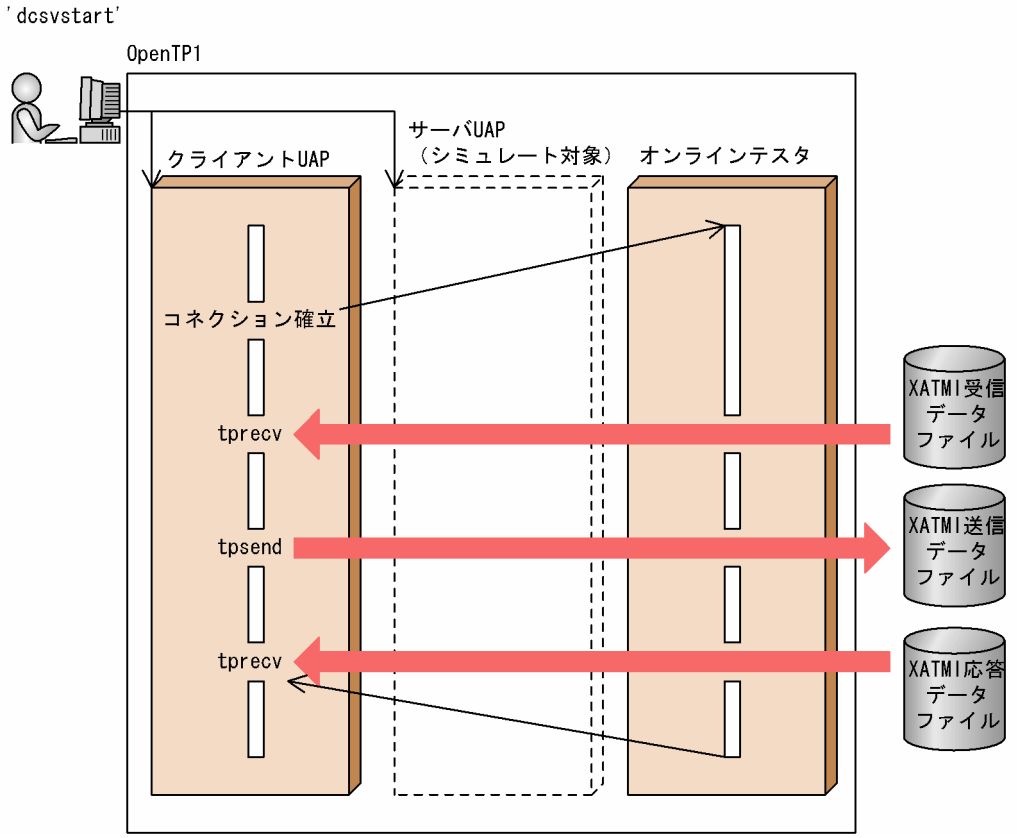
(2)　会話型のサービス要求

会話型のサービス要求でサーバ UAP をシミュレートする場合，ユーザは，クライアント UAP が受け取るデータを，あらかじめ **XATMI 受信データファイル**や XATMI 応答データファイルに作成しておきます。クライアント UAP から受信要求があると，オンラインテスタはまず，XATMI 受信データファイルから受信データを 1 データずつ読み込み，クライアント UAP に返します。XATMI 受信データファイル内のすべてのデータを返したあと，さらに受信要求があると，応答データを XATMI 応答データファイルから読み込み，クライアント UAP に返します。

クライアント UAP が送信したデータは，ユーザサービス定義の指定によって，オンラインテスタが作成する **XATMI 送信データファイル**に取得します。

会話型のサービス要求でのサーバ UAP シミュレート機能の概要を，次の図に示します。

図 2-6　会話型のサービス要求でのサーバ UAP のシミュレート



2.4 MCF のシミュレート

オンラインテスタは，MCF の代わりとなって MHP とのメッセージ送受信を実行できます。そのため，ユーザは MCF がなくても MHP や，MHP からサービス要求する SPP をテストできます。これを，**MCF シミュレート機能**といいます。

MHP のアプリケーション起動は，オンラインテスタのコマンドで実行します。ユーザは，MHP へ渡すメッセージを，あらかじめ **MCF 受信メッセージファイル**に作成しておきます。MHP，および SPP が送信したメッセージは，オンラインテスタが作成する **MCF 送信メッセージファイル**に取得します。

送信メッセージは，オンラインテスタのコマンドで編集したり，特定のメッセージを MCF 受信メッセージファイルの形式にして再利用したりできます。

なお，MCF シミュレート機能を使用した MHP は，オンラインテスタが管理しています。MCF が起動されていても，MCF はこの MHP を管理しません。そのため，MCF が提供する運用コマンドは，この MHP に対しては使用できません。

2.4.1 MCF シミュレート関数

MCF シミュレート機能は，MHP を MCF が提供するライブラリではなく，オンラインテスタのライブラリとリンケージして実行します。オンラインテスタのライブラリとリンケージすると，MHP で使われている関数は，オンラインテスタ用の関数に置き換えられます。この関数を **MCF シミュレート関数**といいます。

MCF シミュレート機能を使用する場合は，関数を MCF シミュレート関数に置き換えられた MHP が**シミュレート MHP** であることを，ユーザサービス定義で定義しておきます。シミュレート MHP とは，MCF シミュレート関数を使用することで，オンラインテスタのすべての機能を使用できる（テストモードで動作する）MHP のことです。ただし，シミュレート MHP はオンラインテスタから SPP として管理されます。

なお，通常の MHP（MCF が提供するライブラリとリンケージした MHP）は，オンラインテスタではテストできません。

2.4.2 メッセージ送受信のシミュレート

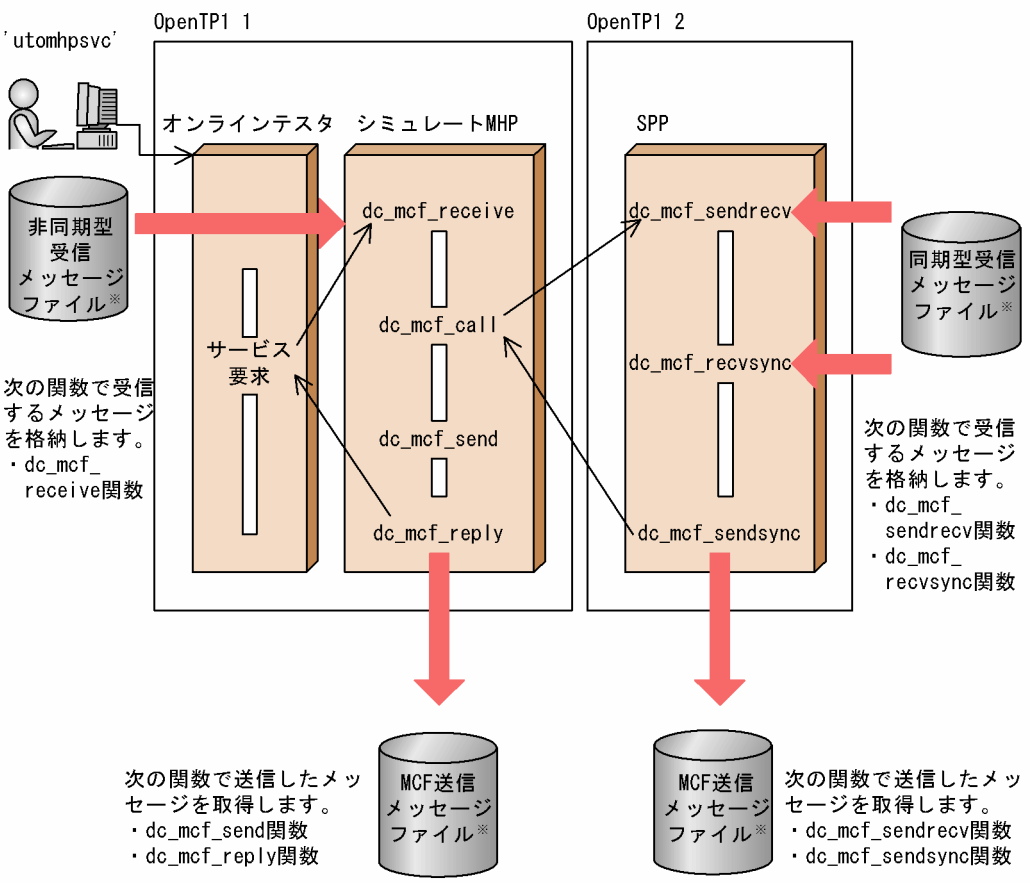
メッセージ送受信のシミュレートは，MCF シミュレート関数で実行します。非同期型メッセージ送受信と同期型メッセージ送受信では，受信メッセージはそれぞれ別の MCF 受信メッセージファイルに作成します。

非同期型メッセージ送受信をシミュレートするときは，**非同期型受信メッセージファイル**を使用します。非同期型受信メッセージファイルには，一つの論理メッセージだけを格納しておきます。

同期型メッセージ送受信をシミュレートするときは，**同期型受信メッセージファイル**を使用します。同期型受信メッセージファイルには，1 回のサービス実行中に同期型で受信するすべての論理メッセージを格納しておきます。

メッセージ送受信のシミュレートの概要を，次の図に示します。

図 2-7　メッセージ送受信のシミュレート



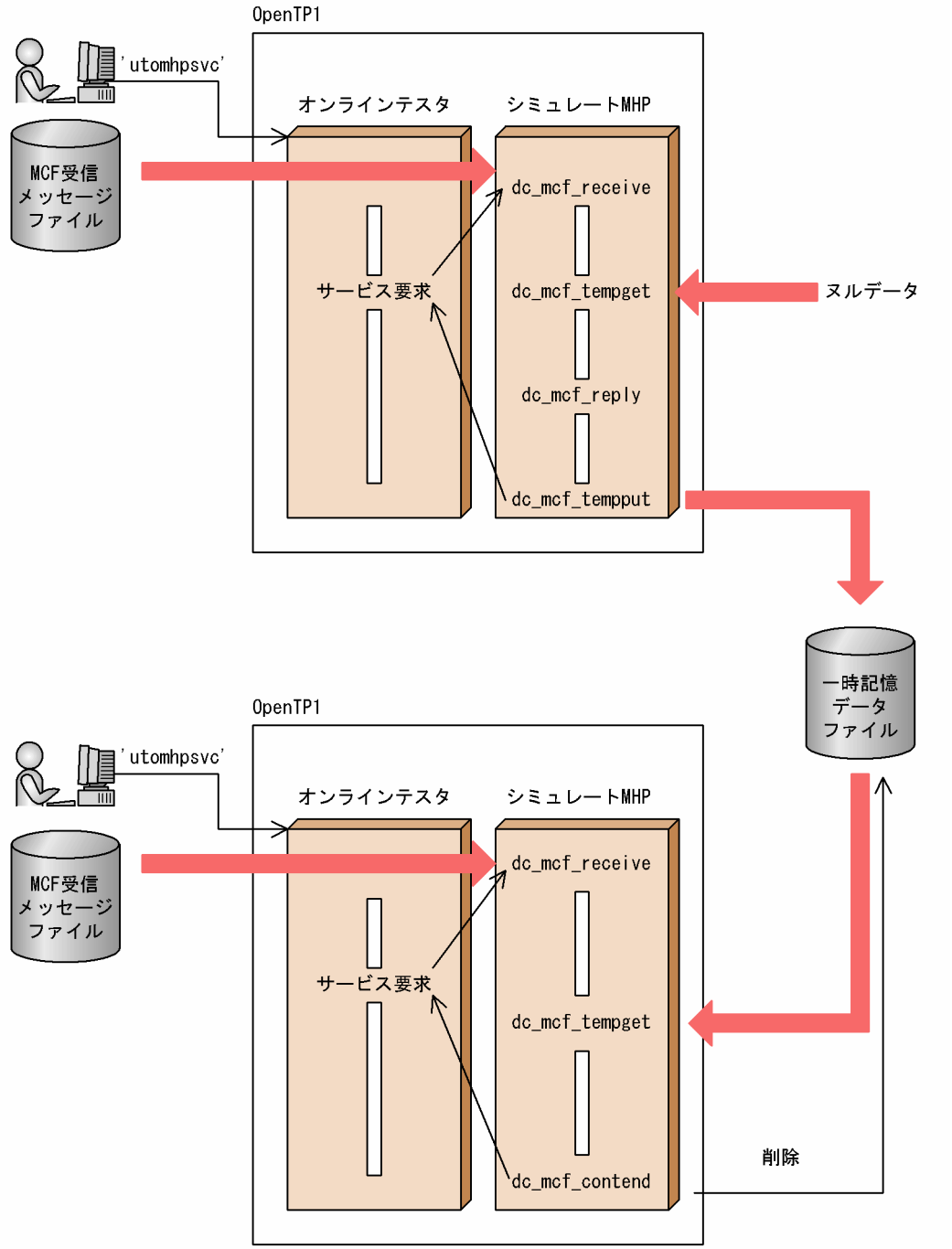
注※ MCF受信メッセージファイル（非同期型・同期型）はテストユーザIDごと，かつ，論理端末ごとにユーザが作成します。
MCF送信メッセージファイルはテストユーザIDごとにオンラインテスタが作成します。

2.4.3　継続問い合わせ応答のシミュレート

継続問い合わせ応答のシミュレートは，オンラインテスタのコマンドで実行します。一時記憶データは，オンラインテスタが作成する**一時記憶データファイル**に取得します。このファイルは，継続問い合わせ応答終了時に自動的に削除されます。

継続問い合わせ応答のシミュレートの概要を，次の図に示します。

図 2-8　継続問い合わせ応答のシミュレート



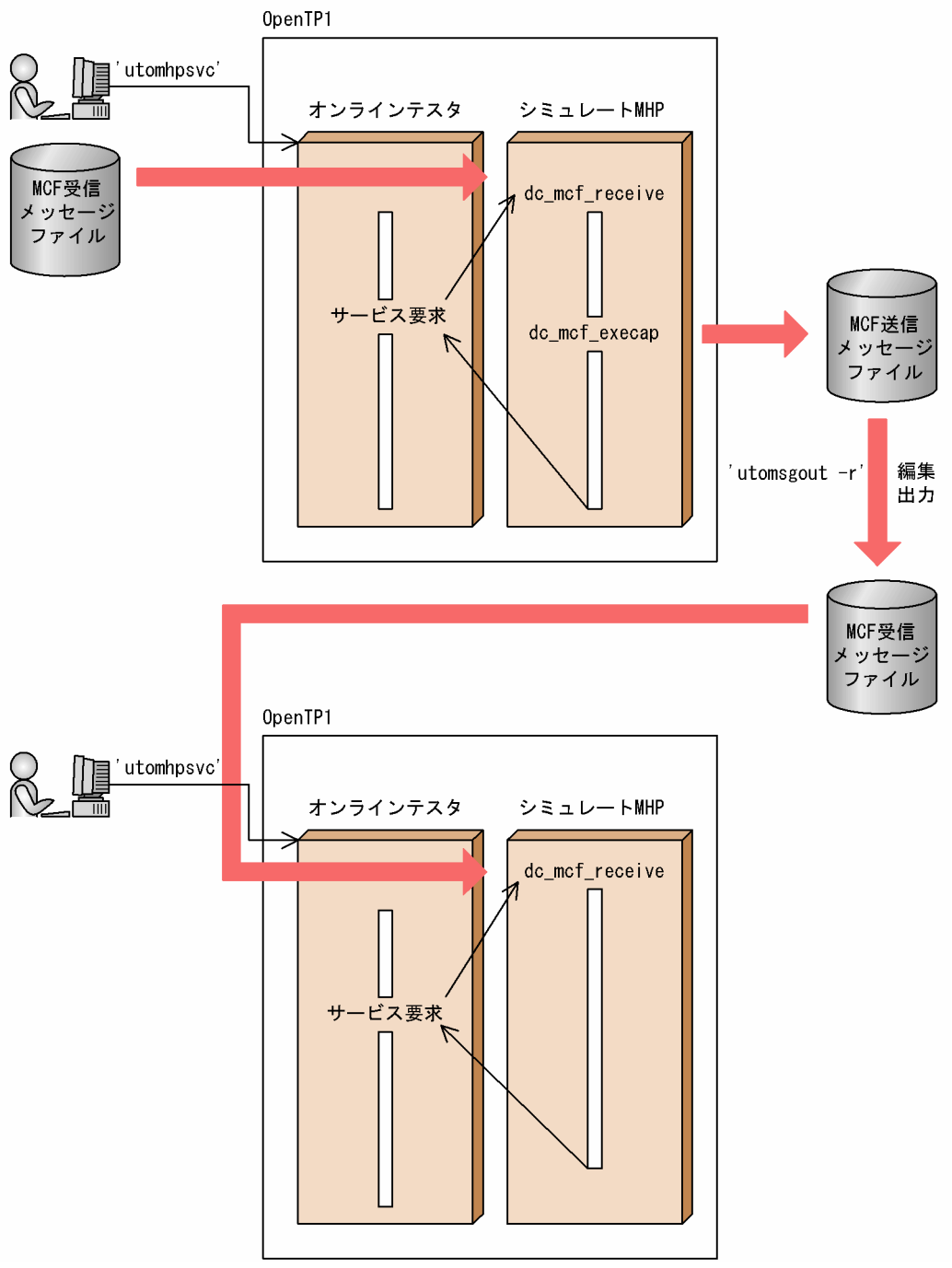
2.4.4　アプリケーションプログラム起動要求のシミュレート

アプリケーションプログラム起動要求のシミュレートは，オンラインテスタのコマンドで実行します。UAP 内からアプリケーションプログラムの起動を要求しても，アプリケーションプログラムは起動しないで，引き渡すデータをオンラインテスタが作成する MCF 送信メッセージファイルに取得します。このデータを渡してアプリケーションプログラムを起動する場合は，まず，オンラインテスタのコマンドで引き渡すデータを別ファイルに取り出します。このファイルを MCF 受信メッセージファイルとして，ユーザがコ

マンドでアプリケーションプログラムを起動させると，実際には起動しなかったアプリケーションプログラムの起動要求をシミュレートできます。

アプリケーションプログラム起動要求のシミュレートの概要を，次の図に示します。

図 2-9　アプリケーションプログラム起動要求のシミュレート



2.4.5　同期点処理のシミュレート

テストする MHP からコミット要求またはロールバック要求の関数が発行された場合，オンラインテスタでは実際に関数を実行します。ただし，コミット要求に関しては，ユーザサービス定義の指定でコミットするかロールバックするかを決定します。

また，ロールバック要求中にプロセス終了や再スケジューリングの指示があっても，関数は必ずリターンされます。

オンラインテスタでは，プロセスの終了や再スケジューリングができません。該当する処理は，テストする MHP 内に作り込んでください。

なお，DML 形式の COMMIT 文を発行する場合，コミット要求でエラーが発生しても，UAP でエラーを検出できません。

2.5 資源更新処理の無効化

オンラインテスタでは，テストで更新した資源を回復できます。これを，**資源更新処理無効化機能**といいます。

更新した資源は，正常に終了したトランザクションをロールバックさせることで回復されます。正常に終了したトランザクションをコミットさせるかロールバックさせるかは，グローバルトランザクション単位に，ルートトランザクションブランチが発生した UAP のユーザサービス定義で指定します。複数のトランザクションブランチが発生する場合も，各 UAP での指定に関係なく，ルートトランザクションブランチが発生した UAP での指定が有効になります。

また，テストするトランザクションで取得した，トランザクションに依存するジャーナルは，通常のジャーナルと同様に，TP1/Server Base の jnledit コマンドで編集出力できます。

2.6　UAP から発行する運用コマンドのシミュレート

オンラインテスタは，UAP から dc_adm_call_command 関数で運用コマンドの実行を要求した場合，コマンドの実行をシミュレートできます。これを，**運用コマンドシミュレート機能**といいます。

運用コマンドの実行をシミュレートするかどうかは，ユーザサービス定義で UAP ごとに指定します。運用コマンドのシミュレートには次の 2 種類があり，ユーザが選択できます。

2.6.1　コマンドの実行をスキップする場合

運用コマンドを実行しないで，スキップします。

ユーザサービス定義で運用コマンド実行のスキップを指定すると，コマンドの実行結果（関数のリターン情報）には仮定値として次の情報が設定されます。

- シェルの終了コード
  0 が設定されます。
- 標準出力，標準エラー出力の出力データ
  ヌル文字が設定されます。
- 標準出力，標準エラー出力の出力データ長
  0 が設定されます。

2.6.2　コマンドの実行をファイルで代替する場合

運用コマンドを実行しないで，実行結果をファイルで代替します。

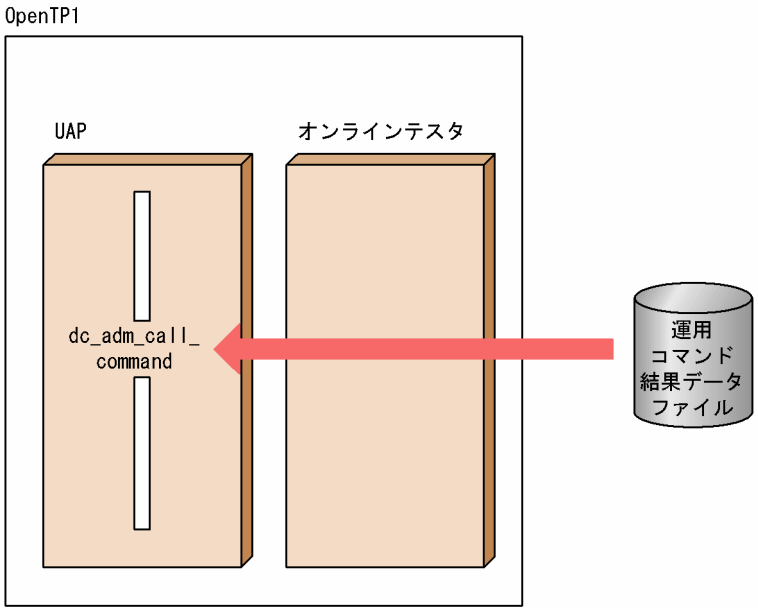
ユーザサービス定義で運用コマンド実行のファイル代替を指定すると，コマンドの実行結果を，**運用コマンド結果データファイル**の内容で代替します。

UAP で dc_adm_call_command 関数が実行されると，オンラインテスタはコマンドの実行結果データをファイルから読み込み，UAP に返します。

運用コマンド結果データファイルはサービスごとに作成し，ユーザがあらかじめコマンドの実行結果データを設定しておきます。一つのサービスで複数回 dc_adm_call_command 関数を発行する場合は，発行する回数分のデータを作成します。SPP のメイン関数や SUP で発行する場合も同様です。

運用コマンドの実行結果をファイルで代替する場合の概要を，次の図に示します。

図 2-10　運用コマンド実行結果のファイル代替



2.7　テスタファイルの作成・編集出力

オンラインテスタで各種のシミュレート機能を使用するためには，機能ごとに専用のデータファイルを作成する必要があります。これを**テスタファイル**といいます。

ここでは，テスタファイルの作成，および編集出力について説明します。

2.7.1　テスタファイルの作成

テスタファイルはそれぞれ専用のデータ形式で記述されていますが，オンラインテスタでは，ユーザがコマンドで簡単にテスタファイルを作成できます。これを，**テスタファイル作成機能**といいます。

テスタファイル作成機能で作成できるテスタファイルの一覧を，次の表に示します。

表 2-1　テスタファイル作成機能で作成できるテスタファイルの一覧

テスタファイル名		作成者	テスタファイルを使用する機能
サービス要求データファイル	RPC 要求データファイル	ユーザ	クライアント UAP シミュレート機能
	XATMI 要求データファイル	ユーザ	クライアント UAP シミュレート機能
サービス応答データファイル	RPC 応答データファイル	ユーザ	サーバ UAP シミュレート機能
		オンラインテスタ	クライアント UAP シミュレート機能
	XATMI 応答データファイル	ユーザ	サーバ UAP シミュレート機能
		オンラインテスタ	クライアント UAP シミュレート機能
XATMI 受信データファイル		ユーザ	クライアント UAP シミュレート機能
MCF 受信メッセージファイル	非同期型受信メッセージファイル	ユーザ	MCF シミュレート機能
	同期型受信メッセージファイル	ユーザ	MCF シミュレート機能
運用コマンド結果データファイル		ユーザ	運用コマンドシミュレート機能

テスタファイル作成機能では，次に示す二つの方式でテスタファイルを作成します。

- テストデータ定義ファイル方式

  ユーザがあらかじめ作成したテストデータ定義ファイルのデータを使用して，テスタファイルを作成します。テストデータ定義ファイルのデータは，テキストエディタで作成できます。また，テストデータ定義ファイルには，複数のテスタファイルのデータをまとめて設定できます。
- ジャーナルデータ方式

アンロードジャーナルファイルのレコードデータや，RPC トレースファイルのトレースデータを使用して，テスタファイルを作成します。ジャーナルデータはあらかじめ運用コマンドでデータを取り出しておきます。

なお，作成するテスタファイル種別によって，データを取り出すコマンドやデータの種類が異なります。また，ジャーナルデータ方式では運用コマンド結果データファイルは作成できません。作成するテスタファイルの種別と対応するデータ取り出しコマンドおよび有効データ内容の一覧を，次の表に示します。

表 2-2　作成するテスタファイルの種別と対応するデータ取り出しコマンドおよび有効データ内容一覧

テスタファイル名	データ取り出しコマンド名	有効とするデータの内容
RPC 要求データファイル	rpcdump コマンド	• rpcdump コマンドで取り出した RPC トレースデータの RPC 要求送信データのうち，最初に該当したデータ
XATMI 要求データファイル	rpcdump コマンド	• rpcdump コマンドで取り出した RPC トレースデータの XATMI リクエスト/レスポンス型要求送信データのうち，tpcall 関数，tpacall 関数のデータで最初に該当したデータ • rpcdump コマンドで取り出した RPC トレースデータの XATMI 会話型要求送信データのうち，tpconnect 関数のデータで最初に該当したデータ
RPC 応答データファイル	rpcdump コマンド	• rpcdump コマンドで取り出した RPC トレースデータの RPC 応答送信データのうち，最初に該当したデータ
XATMI 応答データファイル	rpcdump コマンド	• rpcdump コマンドで取り出した RPC トレースデータの XATMI リクエスト/レスポンス型および会話型応答送信データのうち，tpreturn 関数のデータで最初に該当したデータ
XATMI 受信データファイル	rpcdump コマンド	• rpcdump コマンドで取り出した RPC トレースデータの XATMI 会話型受信データのうち，tprecv 関数のすべてのデータ
非同期型受信メッセージファイル	jnlrput コマンド	• jnlrput コマンドで取り出したアンロードジャーナルファイルの ij レコードのデータおよび mj レコードの入力メッセージデータ（複数のレコードデータがある場合は，データの順序識別子が最初に's'または'l'となったデータまでを有効とします）
同期型受信メッセージファイル	jnlrput コマンド	• jnlrput コマンドで取り出したアンロードジャーナルファイルの mj レコードの入力メッセージデータ
運用コマンド結果データファイル	－	－

（凡例）

－：ジャーナルデータやトレースデータからは，ファイルを作成できません。

2.7.2　テスタファイルの編集出力

オンラインテスタでは，作成したテスタファイルの内容を編集して出力できます。これを**テスタファイル編集出力機能**といいます。

テスタファイル内容の編集出力は，オンラインテスタのコマンドで実行します。

コマンドを入力すると，指定したテスタファイル内のデータを指定したテスタファイル種別の形式に合わせて編集し，標準出力に出力します。

テスタファイル編集出力機能で編集出力できるテスタファイルの一覧を，次の表に示します。

表 2-3　テスタファイル編集出力機能で編集出力できるテスタファイルの一覧

テスタファイル名		作成者	テスタファイルを使用する機能
サービス要求データファイル	RPC 要求データファイル	ユーザ	クライアント UAP シミュレート機能
	XATMI 要求データファイル	ユーザ	クライアント UAP シミュレート機能
サービス応答データファイル	RPC 応答データファイル	ユーザ	サーバ UAP シミュレート機能
		オンラインテスタ	クライアント UAP シミュレート機能
	XATMI 応答データファイル	ユーザ	サーバ UAP シミュレート機能
		オンラインテスタ	クライアント UAP シミュレート機能
XATMI 送受信データファイル	XATMI 送信データファイル	オンラインテスタ	クライアント UAP シミュレート機能
			サーバ UAP シミュレート機能
	XATMI 受信データファイル	ユーザ	クライアント UAP シミュレート機能
MCF 受信メッセージファイル	非同期型受信メッセージファイル	ユーザ	MCF シミュレート機能
	同期型受信メッセージファイル	ユーザ	MCF シミュレート機能
運用コマンド結果データファイル		ユーザ	運用コマンドシミュレート機能

2.8　テスト情報の取得

2.8.1　UAP トレース情報の取得

オンラインテスタは，テストモードで動作している UAP について，各 OpenTP1 関数の入り口と出口とで，UAP トレース情報をファイルに取得します。これを，**UAP トレース情報取得機能**といいます。

OpenTP1 が提供する関数のうち，ユーザサーバや RPC の関数，および DAM や TAM などのファイルにアクセスする関数では，関数に指定したすべての入出力データを，トレース情報として取得できます。これを**全入出力データトレース取得機能**といいます。全入出力データトレース取得機能を使用できる関数を，次の表に示します。

表 2-4　全入出力データ取得機能を使用できる関数の一覧

機能	関数名	
	C 言語形式	COBOL 言語形式
ユーザサーバ	サービス関数開始	サービスプログラム開始
	サービス関数終了	サービスプログラム終了
	サービス関数開始（リトライ時）	サービスプログラム開始（リトライ時）
	サービス関数終了（リトライ時）	サービスプログラム終了（リトライ時）
リモートプロシジャコール	dc_rpc_call	CBLDCRPC('CALL ')
	dc_rpc_cltsend	CBLDCRPC('CLTSEND ')
DAM ファイルサービス	dc_dam_read	CBLDCDAM('DCDAMSVC','READ')
	dc_dam_rewrite	CBLDCDAM('DCDAMSVC','REWT')
	dc_dam_write	CBLDCDAM('DCDAMSVC','WRIT')
TAM ファイルサービス	dc_tam_read	CBLDCTAM('FxxR')('FxxU')
	dc_tam_rewrite	CBLDCTAM('MFY ')('MFYS')('STR ')
	dc_tam_write	
IST サービス	dc_ist_read	CBLDCIST('DCISTSVC','READ')
	dc_ist_write	CBLDCIST('DCISTSVC','WRIT')
ユーザジャーナルの取得	dc_jnl_ujput	CBLDCJNL('UJPUT ')

全入出力データをトレース情報として取得するタイミングは，関数によって異なります。トレース情報の取得タイミングを次に示します。

- ユーザサーバ

開始時に入力データを，終了時に出力データを取得します。

- データを送受信する関数

  関数の入り口でサービス要求の送信データおよびCUPへの通知データを，関数の出口でサービス応答の受信データを取得します。

- ファイルデータを読み込む関数

  関数の出口で取得します。

- ファイルデータを書き込む関数

  関数の入り口で取得します。

トレース情報を取得する**トレースファイル**は，オンラインテスタが最初のトレース情報を取得するとき，OpenTP1 ごとに自動で作成されます。トレースファイルが満杯になった場合は，別ファイルにスワップします。

なお，トレース情報は，サービス関数終了時などに複数の OpenTP1 関数分をまとめてファイルに取得します。また，UAP が異常終了した場合は，退避コアファイルからトレース情報を抽出してファイルに取得します。そのため，UAP 実行中にオンラインが即時停止した場合や，UAP 異常終了時に退避コアファイルを取得していない場合は，トレース情報をファイルに取得できないことがあります。

2.8.2 UAP トレース情報のマージ・編集出力

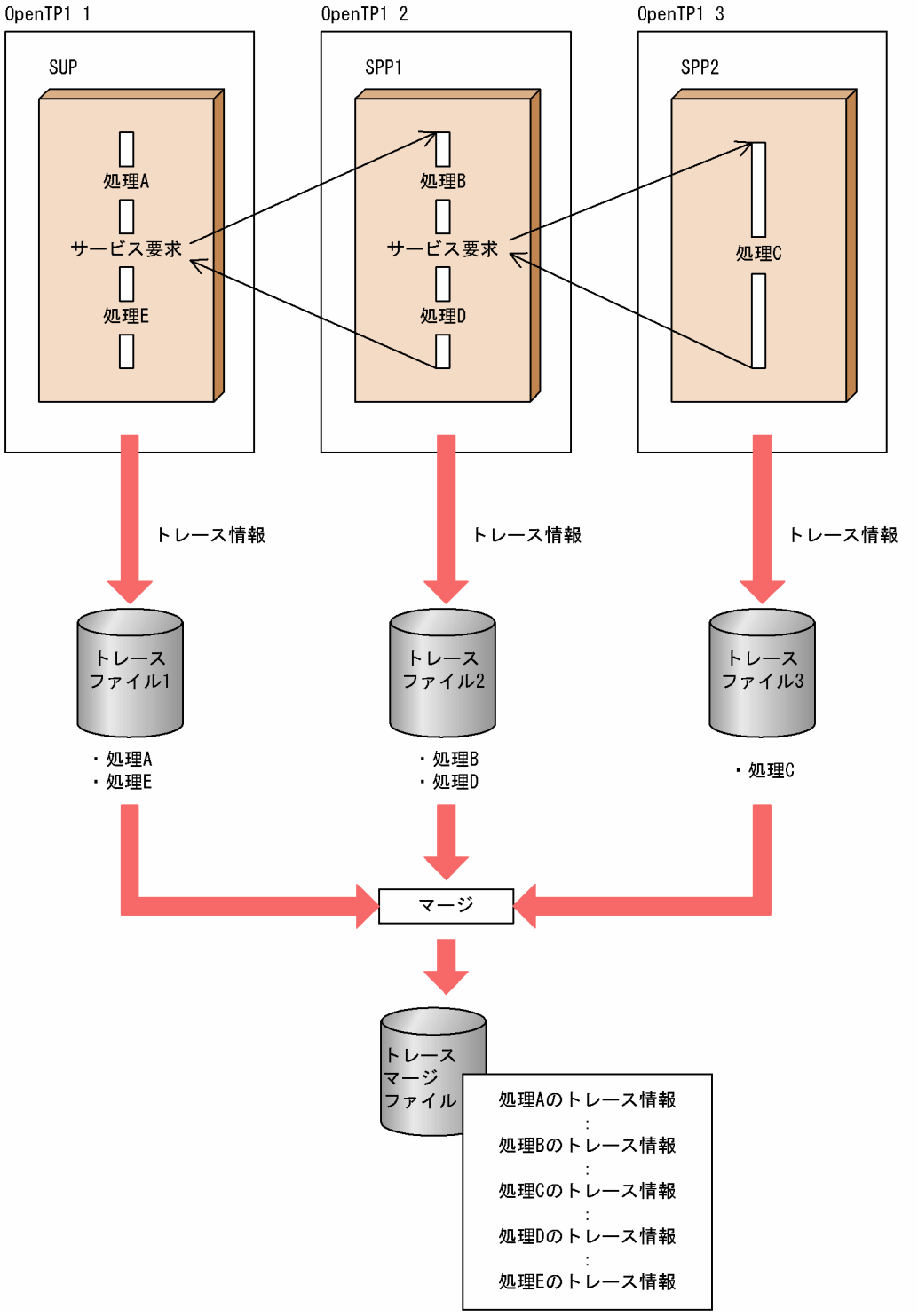
オンラインテスタでは，複数のトレースファイルに取得した UAP トレース情報を，一つのファイルにマージし，編集出力できます。これを，**UAP トレース情報マージ・編集機能**といいます。

トレース情報のマージは，オンラインテスタのコマンドで実行します。

トレースファイルを複数指定してコマンドを実行すると，トレース情報はサービスの流れに沿って一つのファイルにマージされます。そのため，複数の OpenTP1 が取得したトレース情報をグローバルトランザクション単位で取得順にまとめたい場合や，トレースファイル満杯時に別のファイルにスワップしたものを一つにまとめたい場合に使用できます。

UAP トレース情報のマージ結果を，次の図に示します。

図 2-11　UAP トレース情報のマージ結果



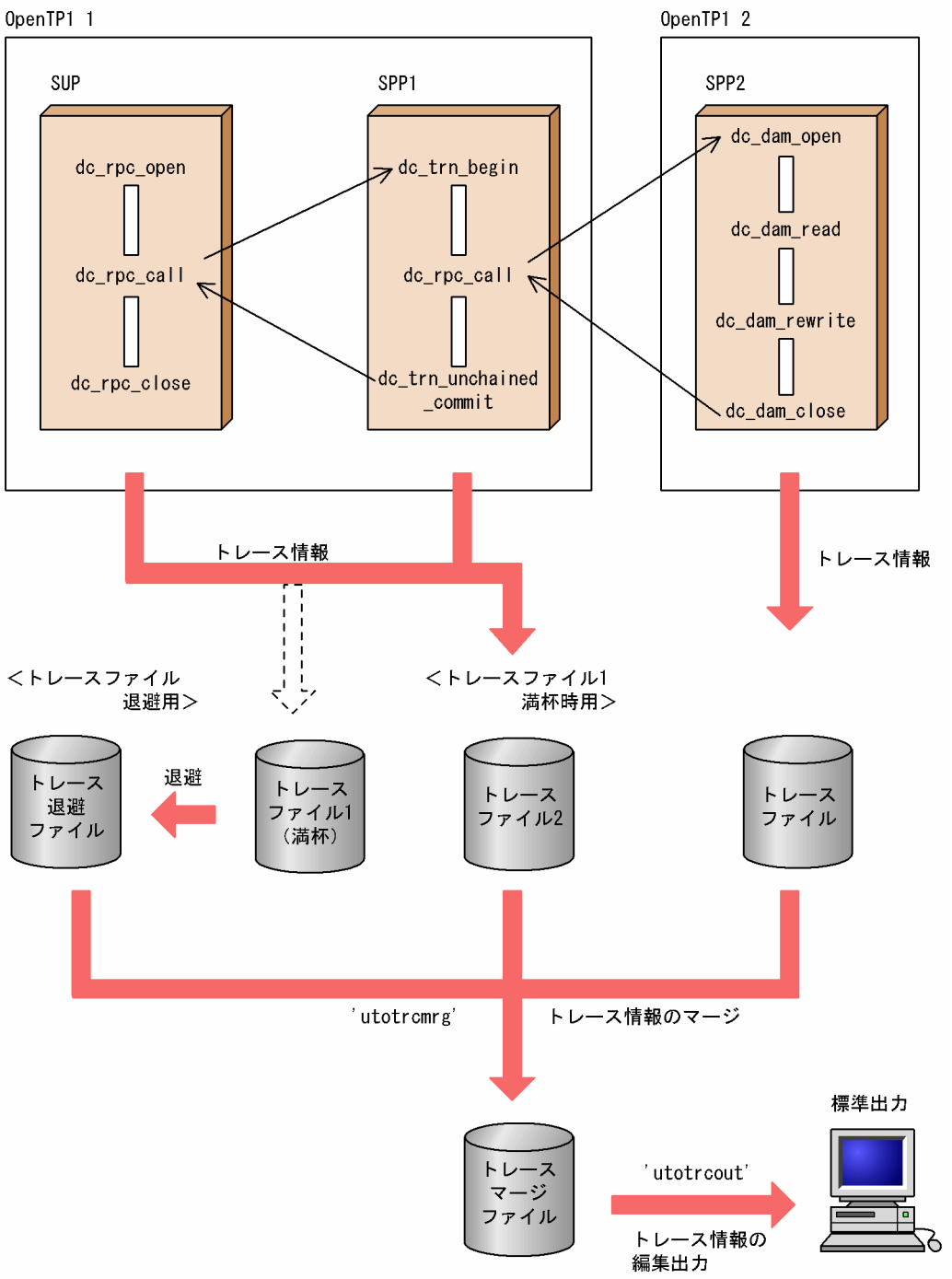
また，トレース情報はオンラインテスタのコマンドで編集して出力できます。このとき，トレース情報の取得日時で出力する範囲を指定できます。

トレース情報の出力形式には，次のものがあります。

- トレースファイル中の，すべてのトレース情報
- トレースファイル中の，関数名などの一部のトレース情報

UAP トレース情報の取得から，マージ・編集出力までの機能の概要を，次の図に示します。

図 2-12　UAP トレース情報の取得とマージ・編集出力



2.8.3　送信メッセージの編集

MCF シミュレート機能使用時に MCF 送信メッセージファイルに取得した送信メッセージは，編集して出力できます。これを，**送信メッセージ編集機能**といいます。

送信メッセージの編集は，オンラインテスタのコマンドで実行します。コマンドを入力すると，MCF 送信メッセージファイルのデータをコマンドで指定したファイル，または標準出力に編集して出力します。

2.9 デバッガの連動

オンラインテスタでは，オンライン環境下でテスト専用 UAP の SUP，SPP，および MHP をデバッガと連動しながらテストを実行できます。これを，**デバッガ連動機能**といいます。

ユーザサービス定義で UAP ごとにデバッガの連動を指定し，オンラインテスタのコマンドで実行すると，UAP の main 関数からデバッガと連動して動作します。

デバッガを連動させると，1 ステップ単位のデバッグやバッチ形式のデバッグが，簡単にできます。

使用できるデバッガを次に示します。

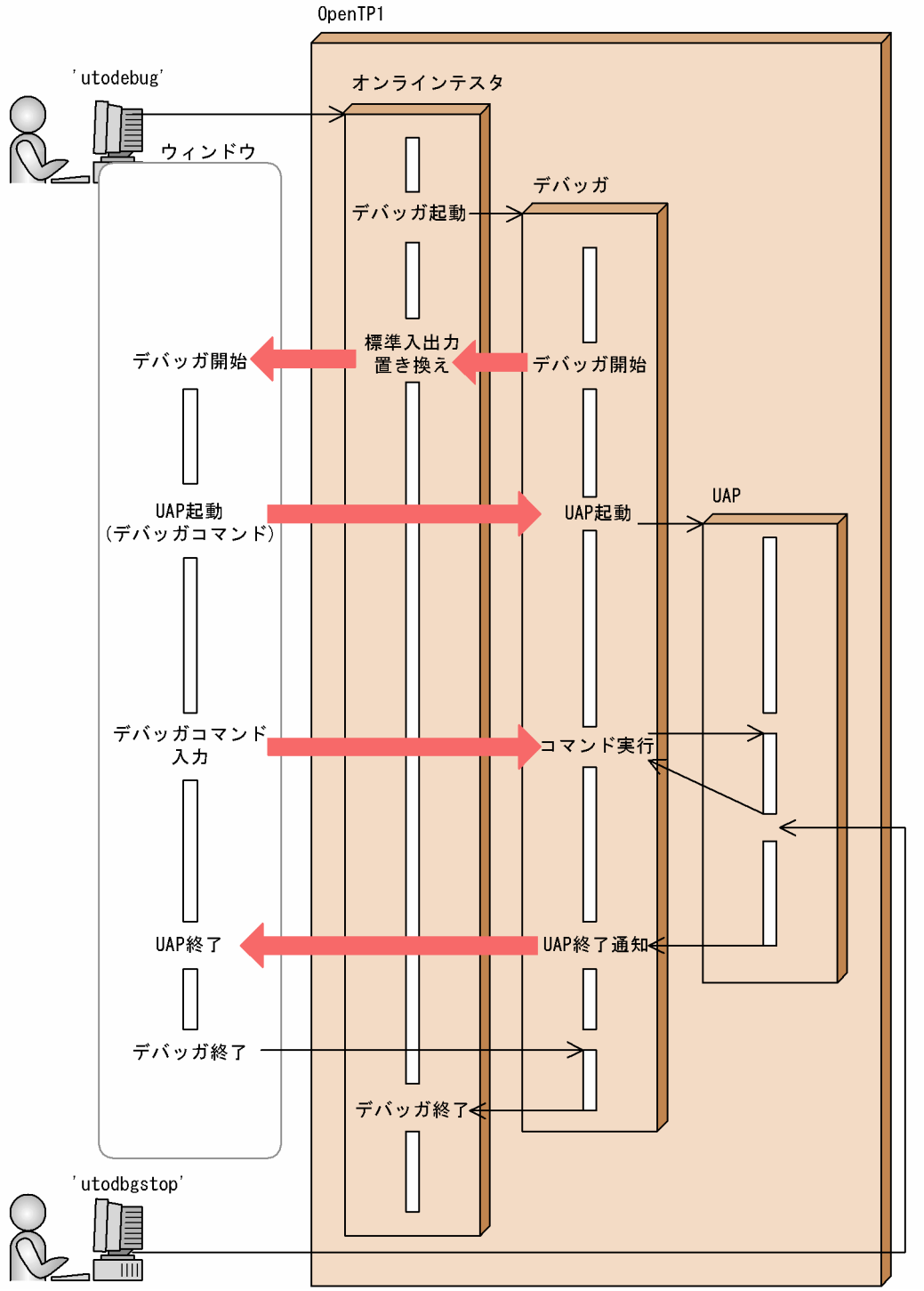
- dbx
- cbltd（COBOL2002）
- cblcv（COBOL2002）

オンライン環境下でデバッガと連動してテストする UAP は，あらかじめ，オフライン環境下でテストを実行してください。また，複数のユーザが同一ノード上でデバッガと連動してテストを実行することは避けてください。これは，デバッガ連動中の UAP が異常終了した場合に，OpenTP1 システムへの影響を防ぐためです。

デバッガと連動して動作する UAP のトレース情報は，デバッガと連動していない UAP と同様に取得できます。ただし，オンラインテスタがトレース情報を出力するタイミングによって，一部またはすべてのトレース情報が取得できない場合があります。

デバッガ連動機能の概要を，次の図に示します。

図 2-13　デバッガの連動



3 テスト環境の設定

オンラインテスタでテストを実行するために必要な，環境設定の方法について説明します。

3.1　オンラインテスタのシステム定義

オンラインテスタのシステム定義について説明します。定義の構成，規則などについては，マニュアル「OpenTP1 システム定義」を参照してください。

3.1.1　システムサービス構成定義

OpenTP1 のシステムサービス構成定義（定義ファイル名：$DCCONFPATH/sysconf）に，次の定義を追加して定義します。

(1)　形式

(a)　set 形式

```
〔set uto_conf=Y|N〕
```

(b)　コマンド形式

なし。

(2)　機能

システム起動時にオンラインテスタを一緒に開始するかどうかを定義します。

(3)　説明

(a)　set 形式のオペランド

- uto_conf=Y ｜ N　　～《N》
  このノードでオンラインテスタを使用するかどうかを指定します。
  Y：オンラインテスタを使用します。
  N：オンラインテスタを使用しません。

(b)　コマンド形式

なし。

3.1.2　テスタサービス定義

定義ファイル$DCCONFPATH/uto を作成し，テスタサービス定義を定義します。

(1) 形式

(a) set 形式

```
〔set uto_server_count=最大テスト用ユーザサーバ数〕
〔set max_trace_file_size=最大トレースファイル容量〕
〔set max_message_file_size=最大MCF送信メッセージファイル容量〕
〔set watch_time=最大応答待ち時間〕
〔set rpc_trace=Y|N〕
〔set rpc_trace_name="RPCトレース情報を取得するファイル名"〕
〔set rpc_trace_size=RPCトレース情報を取得するファイルのサイズ〕
```

(b) コマンド形式

```
{{ 〔utoterm 〔-p OSITP|other〕 論理端末名称〕 }}
```

(2) 機能

オンラインテスタサービスを実行するための環境を定義します。

(3) 説明

(a) set の形式オペランド

- uto_server_count=最大テスト用ユーザサーバ数

  ～〈符号なし整数〉((1～240))《64》

  オンラインテスタのテスト対象として起動するユーザサーバ数を指定します。

- max_trace_file_size=最大トレースファイル容量

  ～〈符号なし整数〉((0～2000000))《64》(単位：キロバイト)

  UAP トレース情報を取得するトレースファイルの最大容量を指定します。ただし，トレースファイルは 128 バイトのヘッダ（管理情報）を持つため，その大きさを加えた値を指定します。

  0 を指定した場合には，オンラインテスタは UAP トレース情報をファイルに取得しません。

  なお，トレースファイルはテストユーザ ID ごとに最大二つ作成します。これは，トレースファイルが満杯になった場合にトレース情報が消えるのを防ぐためです。

  トレースファイルが満杯になると，そのたびにトレース情報の退避や満杯になったトレースファイルの削除が必要になります。最大トレースファイル容量には，できるだけ満杯にならない容量を指定してください。

  OpenTP1 システムでトレースファイルとして使用する最大容量は，次の式で表されます。

  トレースファイルの最大容量

  =max_trace_file_size オペランドの指定値×2×ユーザ数×1024 バイト

- max_message_file_size=最大 MCF 送信メッセージファイル容量

  ～〈符号なし整数〉((0～2000000))《64》(単位：キロバイト)

オンラインテスタの MCF シミュレート機能使用時，次に示す関数で送信するメッセージを格納する，MCF 送信メッセージファイルの最大容量を指定します。

- dc_mcf_send 関数
- dc_mcf_reply 関数
- dc_mcf_sendsync 関数
- dc_mcf_sendrecv 関数
- dc_mcf_execap 関数

ただし，各送信メッセージは 128 バイトの管理データを持ち，MCF 送信メッセージファイルは 128 バイトのヘッダ（管理情報）を持つため，その大きさを加えた値を指定します。

0 を指定した場合には，オンラインテスタは送信メッセージをファイルに取得しません。

なお，MCF 送信メッセージファイルはテストユーザ ID ごとに作成するため，OpenTP1 システムで MCF 送信メッセージファイルとして使用する最大容量は，次の式で表されます。

MCF 送信メッセージファイルの最大容量

=max_message_file_size オペランドの指定値×ユーザ数×1024 バイト

- watch_time=最大応答待ち時間

  ～〈符号なし整数〉((0～65535))（単位：秒）

  RPC を使用してプロセス間で通信する場合，サービス要求を送信してからサービスの応答が返るまでの待ち時間の最大値を指定します。

  OpenTP1 の終了処理では，このオペランドで指定した時間，終了処理を待ち合わせる場合があります。このため，大きな値を指定した場合，OpenTP1 の終了処理に時間が掛かるときがあります。

  指定時間を過ぎても応答がない場合は，RPC は送受信タイムアウトとしてエラーリターンします。

  0 を指定した場合は，応答を受信するまで待ち続けます。0 を指定した場合，OpenTP1 が終了しないときがあります。

  省略した場合は，システム共通定義の watch_time オペランドの値が仮定されます。

- rpc_trace=Y | N

  RPC トレースで情報を取得するかどうかを指定します。

  Y：情報を取得します。

  N：情報を取得しません。

  省略した場合は，システム共通定義の rpc_trace オペランドの値が仮定されます。

- rpc_trace_name="RPC トレース情報を取得するファイル名"

  ～〈パス名〉

  RPC トレース情報を取得するファイルのパス名を指定します。

  パス名のうち，RPC トレースを取得するファイル名（デフォルトは rpctr）の最大長は，13 文字です。

  パス名に環境変数を指定する場合，パス名の先頭に環境変数を指定してください（指定例 $DCDIR/tmp/ファイル名）。

  省略した場合は，システム共通定義の rpc_trace_name オペランドの値が仮定されます。

- rpc_trace_size=RPC トレース情報を取得するファイルのサイズ
  ～〈符号なし整数〉((1024～2147483648))（単位：バイト）
  RPC トレース情報を取得するファイルのサイズを指定します。
  省略した場合は，システム共通定義の rpc_trace_size オペランドの値が仮定されます。

(b) コマンド形式

次の項に記述しています。

3.1.3 テスタサービス定義（コマンド形式）

(1) utoterm（論理端末情報の指定）

名称

論理端末情報の指定

形式

```
{{〔utoterm〔-p OSITP|other〕論理端末名称〕}}
```

機能

MCF シミュレート機能を使用して，DML で作成した MHP をテストする場合に，各論理端末の情報を指定します。

なお，すでにこの定義で指定されている論理端末名称を再度指定すると，二重登録として警告のメッセージが表示され，論理端末名称の指定が無効になります。

オプション

- -p　OSITP ｜ other　　～《other》
  プロトコル種別を指定します。DML で作成した MHP をテストする場合に指定します。
  OSITP：OSI TP プロトコル
  other：OSI TP 以外のプロトコル
- 論理端末名称　　～〈1～8 文字の識別子〉
  論理端末名称を指定します。

3.1.4 ユーザサービス定義

OpenTP1 のユーザサービス定義（定義ファイル名：$DCCONFPATH/ユーザサーバ名）に，次の定義を追加して定義します。

(1) 形式

(a) set 形式

```
〔set test_mode=target|usable|dmyspp|simmhp|no〕
〔set test_transaction_commit=Y|N〕
〔set test_adm_call_command=do|skip|file〕
〔set test_xatmi_send_file=Y|N〕
〔set test_debugger="{dbx|cbltd|cblcv} 〔コマンド引数〕 "〕
〔set test_data_trace=Y|N〕
```

(b) コマンド形式

なし。

(c) putenv 形式

```
〔putenv DCUTOTRCACCL 0|1〕
```

(d) dcputenv 形式

```
〔dcputenv DCUTOTRCACCL 0|1〕
```

(2) 機能

OpenTP1 のユーザサービス定義に，サービスグループごとに追加して定義することで，ユーザサーバでオンラインテスタを実行できます。

(3) 説明

(a) set 形式のオペランド

- test_mode=target | usable | dmyspp | simmhp | no 〜《no》

  オンラインテスタの起動時に，UAP がオンラインテスタの機能の対象かどうかを指定します。

  target：テスト専用 UAP

  テスト専用の UAP として，資源更新処理無効化機能や UAP トレース情報取得機能など，オンラインテスタの全機能の対象とする場合に指定します。

  この UAP からテスト対象外の UAP には，サービス要求できません。また，テスト対象外の UAP からこの UAP にも，サービス要求できません。

  usable：使用可能 UAP

  テスト対象の UAP からサービス要求する SPP の場合に指定します。

  この UAP は，テスト対象の UAP からサービス要求された場合はテストモードで動作し，資源更新処理無効化機能などの機能が使用できます。

テスト対象外の UAP からサービス要求された場合は非テストモードで動作し，オンラインテスタの機能は使用できません。

dmyspp：ダミー SPP

オンラインテスタのサーバ UAP シミュレート機能を使用して，実際には起動しない SPP をシミュレートする場合に指定します。

simmhp：シミュレート MHP

オンラインテスタの MCF シミュレート機能で，MCF シミュレート関数とリンケージした MHP に対して指定します。

no：テスト対象外 UAP

テスト対象外の UAP の場合に指定します。テスト対象の UAP からこの指定をした UAP には，サービス要求はできません。

このオペランドでの指定と UAP で使用できるオンラインテスタの機能との関係，およびサービス要求元の UAP とサービス要求先の UAP の関係を，それぞれ以降の表に示します。

表 3-1　test_mode オペランドの指定による機能の使用可否

使用できる機能	target	usable	dmyspp	simmhp	no
クライアント UAP シミュレート機能	○	○	−	−	×
サーバ UAP シミュレート機能	○	△	−	○	×
MCF シミュレート機能	○※	△	−	○※	×
資源更新処理無効化機能	○	△	−	○	×
運用コマンドシミュレート機能	○	△	−	○	×
UAP トレース情報取得機能	○	△	−	○	×
デバッガ連動機能	○	×	−	○	×

（凡例）

○：使用できます。

×：使用できません。

−：対象外です。

△：関数の種類によって次のように変わります。

・メイン関数

使用できません。

・サービス関数

クライアント UAP シミュレート機能を使用してサービス要求される場合は，使用できます。

それ以外の場合は，サービス要求元の UAP（サービスが複数 UAP にわたっている場合は，最初にサービス要求した UAP）の使用可否に従います。

注※

オンラインテスタの提供する MCF シミュレート関数とリンケージする必要があります。

表 3-2　サービス要求元の UAP とサービス要求先の UAP の関係

サービス要求元		サービス要求先				
		target	usable	dmyspp	simmhp	no
target		○	○	○	−	×
usable	テストモード	○	○	○	−	×
	非テストモード	×	○	×	−	○
dmyspp		−	−	−	−	−
simmhp		○	○	○	−	×
no		×	○	×	−	○

（凡例）

○：サービスを要求できます。

×：サービスを要求できません。

−：対象外です。

- test_transaction_commit=Y | N　　～《N》

  この UAP で発生した，テストモードで動作するトランザクションを，同期点でコミットするかロールバックするかを指定します。

  Y：コミットします。

  N：ロールバックします。

- test_adm_call_command=do | skip | file　　～《do》

  この UAP で発行した dc_adm_call_command 関数で，運用コマンドの実行をシミュレートするかどうかを指定します。

  do：実行します。

  skip：実行しないで，仮定値を実行結果として使用します。

  この指定は，test_mode オペランドの値に target か simmhp を指定した場合，または usable を指定してテストモードで動作している場合だけ有効です。

  file：実行しないで，運用コマンド結果データファイル内のデータを実行結果として使用します。

  この指定は，test_mode オペランドの値に target か simmhp を指定した場合，または usable を指定してテストモードで動作している場合だけ有効です。

- test_xatmi_send_file=Y | N　　～《N》

  XATMI インタフェースで会話型のサービスを要求する場合，サーバ UAP シミュレート機能でシミュレート対象に指定した UAP に対して送信されたデータを，XATMI 送信データファイルに出力するかどうかを指定します。

  Y：ファイルに出力します。

  N：ファイルに出力しません。

  なお，UAP をシミュレート対象（test_mode オペランドで dmyspp を指定）としていない場合は，この指定は無視されます。

- test_debugger="{dbx ｜ cbltd ｜ cblcv} 〔コマンド引数〕"

  デバッガと連動させて UAP を起動させる場合，連動するデバッガコマンド名称と，そのデバッガコマンドに対するコマンド引数を指定します。

  この定義を指定した UAP を utodebug コマンドで起動すると，指定したデバッガと連動して起動します。この定義を指定した UAP に対して誤って dcsvstart コマンド，または dcstart コマンドで起動すると，コマンドがエラーになり，エラーメッセージが出力されます。

  utodebug コマンドで起動した UAP は，utodebug コマンドを実行したウィンドウ以外のウィンドウから，utodbgstop コマンドを実行して終了させてください。

  utodbgstop 以外のコマンドで終了させた場合，OpenTP1 システムとオンラインテスタの間で管理している UAP の状態がすれ違う場合があります。また，実行したコマンドはデバッガの終了待ち状態になります。

  デバッガと連動して起動した UAP プロセスは，再度起動できません。そのため，デバッガと連動した UAP プロセスが終了したあとに UAP を再度実行する場合は，いったんデバッガを停止させて utodebug コマンドを再度実行する必要があります。

  この定義を指定した UAP は，該当する UAP のユーザサービス定義で指定した parallel_count オペランドの値に関係なく，一つのプロセスで起動します。

  また，デバッガと連動して起動した UAP に対する閉塞，および閉塞解除は要求できません。

- test_data_trace=Y ｜ N　　～《N》

  この UAP で発行した関数のすべての入出力データを，トレース情報として取得するかどうかを指定します。全入出力データをトレース情報として取得できる関数については，「2.8.1　UAP トレース情報の取得」を参照してください。

  Y：全入出力データを UAP トレース情報として取得します。

  この指定は，テスタサービス定義の max_trace_file_size オペランドに 1 以上の値を指定して，UAP がテスト対象として動作する場合に有効となります。

  N：入出力データの一部を UAP トレース情報として取得します。

  この指定は，テスタサービス定義の max_trace_file_size オペランドに 1 以上の値を指定して，UAP がテスト対象として動作する場合に有効となります。

(b)　コマンド形式

なし。

(c)　putenv 形式のオペランド

- DCUTOTRCACCL　0 ｜ 1

  大きなデータに対して全入出力データトレースを取得する際，従来どおりの取得方式とするか，取得時間を短縮させる取得方式とするかを指定します。送受信メッセージや入出力データの大きさが数 MB に達するような場合は，この putenv 形式定義に 1 を指定することを推奨します。

  0：従来どおりの方法で全入出力データトレースを取得します。

  1：短縮方式で全入出力データトレースを取得します。

このオペランドが未指定の場合，または不当な値が指定された場合は，0 が指定されたものとします。

ユーザサービス定義の test_data_trace オペランドに Y が指定されていない場合は，指定内容を無効とします。

このオペランドは，ユーザサービスデフォルト定義にも指定できます。

(d) dcputenv 形式のオペランド

- DCUTOTRCACCL　0 | 1

  大きなデータに対して全入出力データトレースを取得する際，従来どおりの取得方式とするか，取得時間を短縮させる取得方式とするかを指定します。送受信メッセージや入出力データの大きさが数 MB に達するような場合は，この dcputenv 形式定義に 1 を指定することを推奨します

  0：従来どおりの方法で全入出力データトレースを取得します。

  1：短縮方式で全入出力データトレースを取得します。

  このオペランドが未指定の場合，または不当な値が指定された場合は，0 が指定されたものとします。

  ユーザサービス定義の test_data_trace オペランドに Y が指定されていない場合は，指定内容を無効とします。

  このオペランドは，ユーザサービスデフォルト定義にも指定できます。

(4) 注意事項

- test_mode オペランドに simmhp を指定した場合，ユーザサービス定義のそのほかの指定は，すべて SPP の指定と同様に指定してください

  （例　type=other）。

  なお，receive_from オペランドには queue を指定してください。

- ユーザサービス定義で指定するオンラインテスタ用の定義は，ユーザサービスデフォルト定義では指定できません。これは，実業務用の UAP が誤ってテスト環境で動作するのを防ぐためです。

- MCF シミュレート機能で，トランザクション MHP として実行するものが一つでもある場合は，ユーザサービス定義の atomic_update オペランドの値に，Y を指定してください。

- テスト対象の UAP のスケジュール優先度は，ユーザサービス定義の schedule_priority オペランドの指定に従います。テスト対象の UAP と実業務用の UAP とを同時に実行させる場合は，実業務用の UAP への性能面での影響を考慮して，テスト対象の UAP の優先度を決定してください。

- ユーザサービス定義の uap_trace_max オペランドに 0 を指定していると，テスタサービス定義の max_trace_file_size オペランドに 1 以上を指定していても，UAP トレース情報を取得できない旨の警告メッセージが出力されます。

- オンラインテスタを使用していない場合，test_mode オペランドに no 以外を指定した UAP を起動すると，dc_rpc_open 関数でエラーリターンします。これは，テスト用の UAP が，誤って実業務用 UAP として動作するのを防ぐためです。

- UAP をデバッガと連動させる場合，test_adm_call_command オペランドには file または skip を指定してください。test_adm_call_command オペランドに do を指定してデバッガと連動した UAP で，dc_adm_call_command 関数を発行すると，UAP が応答待ち状態になり，デバッガの制御ができ

なくなります。この場合は，utodbgstop コマンドで UAP を終了させてから，デバッガを終了させてください。

- デバッガと連動する UAP では，子プロセスを生成する関数（fork()，system()など）は発行しないでください。これらの関数を発行した場合，UAP が応答待ち状態になり，デバッガの制御ができなくなります。この場合は，utodbgstop コマンドで UAP を終了させてから，デバッガを終了させてください。
- マルチノード環境での，デバッガ連動機能は使用できません。
- 実業務用の UAP が起動している OpenTP1 システムでは，できるだけ，デバッガと連動した UAP のテストを実行しないでください。これは，UAP が動作する OpenTP1 システムの環境によって，デバッガと連動している UAP の終了，または異常終了を契機にシステムがダウンするのを防ぐためです。
- デバッガと連動する UAP のユーザサービス定義に指定する，各監視時間の指定値は，デバッガ連動時の動作や作業時間を考慮して決定してください。指定値によって，タイムアウトエラーが頻繁に発生する場合があります。
- test_adm_call_command オペランドに do を指定した UAP から，dc_adm_call_command 関数の引数に設定した dcsvstart コマンドでテストモード UAP を起動する場合は，関数を発行する UAP のユーザサービス定義に環境変数 DCUTOKEY を設定してください。
- DCUTOTRCACCL の指定値が未指定の場合，または不当な値が指定された場合は，0 が指定されたものとします。ユーザサービス定義，ユーザサービスデフォルト定義の指定内容による動作の違いを次に示します。

	指定定義名	putenv，dcputenv 定義指定値								
DCUTOTRCACCL 指定値	ユーザサービス定義	1	1	1	0	0	0	無	無	無
	ユーザサービスデフォルト定義	1	0	無	1	0	無	1	0	無
全入出力データトレースの取得方法		短	短	短	従	従	従	短	従	従

（凡例）

無：指定なし

短：短縮方式で全データ取得

従：従来方式で全データ取得

3.1.5 タイプトバッファの設定

XATMI での UAP のシミュレート時に必要になる，タイプトバッファの情報を設定します。タイプトバッファの情報は，**タイプトバッファ定義ファイル**（ファイル名は任意）に格納します。

(1) 形式

△$_0$タイプ△$_1$サブタイプ△$_1$バッファ長

（凡例）

△$_0$：0 個以上の空白文字，およびタブ

△$_1$：1 個以上の空白文字，およびタブ

(2) オペランド

- **タイプ　～〈8 文字の英大文字〉**

  バッファのタイプとして，次のどれかを指定します。

  - X_COMMON
  - X_C_TYPE

- **サブタイプ　～〈1～16 文字の英数字〉**

  バッファのサブタイプを指定します。サブタイプが 16 文字を超える場合は，先頭から 16 文字だけを有効とします。

  X_COMMON，X_C_TYPE で，それぞれ最大 512 個のサブタイプを定義できます。512 個を超えて定義した場合はエラーになり，utoxsppsvc コマンドが中断されます。

  同じサブタイプ名の指定が重複した場合は，最初の定義が有効になります。二つ目以降はエラーになり，エラーメッセージが出力されます。

  ただし，定義内容がまったく同じ場合は，エラーメッセージは出力されません。

- **バッファ長　～〈10 進数字〉**

  バッファ長を指定します。バッファ長は，TP1/Server Base の stbmake コマンドで作成したスタブソース，および stbmake コマンドに-p オプションを指定した場合の出力結果を参照して，確認してください。

(3) 定義例

```
# タイプトバッファの定義
X_COMMON subtype1 256
X_COMMON subtype2 128
X_C_TYPE subtype3 128
#
```

(4) 注意事項

- 1 行には一つのサブタイプ名を定義します。
- 1 行の長さは，改行コードを含めて 512 バイトまでです。
- コメントを記述する場合は，行の先頭に'#'を記述します。'#'の前には空白文字，およびタブだけが指定できます。

  また，タイプトバッファの定義の後ろには，コメントは記述できません。
- 有効な定義が一つもない場合，このファイルのエラーになりませんが，サービス要求時などでタイプトバッファを確保する際にエラーになります。

3.1.6　送受信手順の設定

XATMI での UAP のシミュレートで，会話型のサービスを要求する際に必要になる，送受信の手順を設定します。送受信手順は，**送受信制御ファイル**（ファイル名は任意）に格納します。

送受信制御ファイルには，テスト対象の SPP にデータを送信するための send 文と，テスト対象の SPP からデータを受信するための recv 文を定義します。

なお，一度もデータを送受信しない場合でも，会話型のサービス要求時は必ず送受信制御ファイルを作成してください。

(1)　形式

(a)　send 文

```
△0send〔△1XATMI受信データファイル名〕
```

（凡例）

△0：0 個以上の空白文字，およびタブ

△1：1 個以上の空白文字，およびタブ

(b)　recv 文

```
△0recv△1タイプ△1サブタイプまたはバッファ長〔△1フラグ〔,フラグ〕…〕
```

（凡例）

△0：0 個以上の空白文字，およびタブ

△1：1 個以上の空白文字，およびタブ

(2)　オペランド

(a)　send 文

- send

  定義名として，send を指定します。

- XATMI 受信データファイル名　　～〈パス名〉

  テスト対象の SPP が受信するデータを格納した，XATMI 受信データファイルの名称を指定します。

  この指定を省略した場合は，一つ前に定義した send 文で指定した，XATMI 受信データファイルのデータが使用されます。最初に定義した send 文でこの指定を省略した場合は，エラーになります。

(b)　recv 文

- recv

定義名として，recv を指定します。

- **タイプ　　～〈8 文字の英大文字〉**

  受信バッファのタイプとして，次のどれかを指定します。

  - X_OCTET
  - X_COMMON
  - X_C_TYPE

- **サブタイプ　　～〈1～31 文字の英数字〉**

  タイプに X_COMMON，または X_C_TYPE を指定した場合に，受信バッファのサブタイプを指定します。

- **バッファ長　　～〈10 進数字〉**

  タイプに X_OCTET を指定した場合に，受信バッファ長を指定します。

- **フラグ**

  受信要求（tprecv 関数を発行）する場合に設定する，次のフラグを指定します。

  - TPNOCHANGE
  - TPNOBLOCK
  - TPNOTIME
  - TPSIGRSTRT

  フラグの指定が必要ない場合は指定しないでください。

  複数の情報を指定する場合は，各情報を','（コンマ）で区切ります。','の前後には空白文字やタブを入れないでください。

(3)　定義例

```
# 送受信手順の定義
# 会話先のサービス名:service01
send sendfile1
recv X_OCTET 128 TPNOCHANGE
send sendfile2
recv X_COMMON subtype1 TPNOTIME,TPSIGRSTRT
#
```

(4)　注意事項

- 1 行の長さは，改行コードを含めて 512 バイトまでです。
- コメントを記述する場合は，行の先頭に'#'を記述します。'#'の前には空白文字，およびタブだけが指定できます。

  また，送受信手順の定義の後ろには，コメントは記述できません。

- send 文と recv 文が一つも定義されてない場合でも，エラーにはなりません。ただし，utoxsppsvc コマンド実行時に，コネクションを確立した時点で処理が終了します。
- utoxsppsvc コマンドでは，会話型のサービス要求時，送受信制御ファイルでの指定に従って tpsend 関数と tprecv 関数を発行します。イベント TPEV_SVCSUCC または TPEV_SVCFAIL が発生した場合は，それ以降の send 文や recv 文を無視して，コマンドが正常終了します。
- recv 文での定義と，utoxsppsvc コマンドが発行する XATMI 関数との関係を次に示します。

```
（例）recv X_COMMON subtype1 TPNOCHANGE  と定義した場合

  ptr = tpalloc(X_COMMON, subtype1,0)
                  1.       2.
  tprecv(cd,&ptr,&len,TPNOCHANGE,&revent);
                            3.
```

1. タイプ
2. サブタイプ
3. フラグ

3.2　環境変数の設定

複数のユーザが同じ OpenTP1 システムでテストを実行した場合，トレース情報などが混ざってテスト結果の確認が難しくなるおそれがあります。このため，オンラインテスタでは各ユーザごとに ID（**テストユーザ ID**）を設定します。オンラインテスタはこのテストユーザ ID によって，トレース情報や MCF 送信メッセージの出力先ファイルを振り分けます。

テストユーザ ID は，次の条件でユーザごとに固有の値を設定します。

設定する環境変数	設定する値の属性	文字数
DCUTOKEY	半角の英数字（a～z，A～Z，0～9）	4 文字以内

テストユーザ ID の設定によって，トレースファイルや MCF 送信メッセージファイルをテストユーザ ID ごとに作成，使用するなどの管理ができます。

テストユーザ ID は次のタイミングで取得します。

- OpenTP1 の dcsvstart コマンドによる UAP の起動時
- ユーザサービス構成定義での dcsvstart コマンド指定時
- オンラインテスタの utosppsvc コマンドおよび utoxsppsvc コマンドによる SPP へのサービス要求時
- オンラインテスタの utomhpsvc コマンドによる MHP へのサービス要求時

なお，テスト対象の UAP が正常終了処理中に，OpenTP1 システムを強制停止（dcsvstop コマンドの-f 指定）するか，システムダウンが発生すると，OpenTP1 システムの再開始時にテストユーザ ID を'_uto'と仮定して再度起動することがあります。その際，テストユーザ ID を仮定して起動したことを知らせるメッセージを出力します。必要であれば該当する UAP をいったん停止させてから，再度 dcsvstart コマンドで起動してください。

3.3　ユーザが作成するファイル

オンラインテスタを使用するためにユーザが作成するテスタファイルの一覧，およびそのファイル名を，それぞれ以降の表に示します。

テスト用ディレクトリの作成については，「3.4.1　テスト用ディレクトリの作成」を参照してください。

表 3-3　ユーザが作成するテスタファイルの一覧

テスタファイル種別		用途と内容	作成タイミング	削除者	削除タイミング
サービス要求データファイル	RPC 要求データファイル	RPC インタフェースのクライアント UAP シミュレート機能使用時，サービス要求先に渡すデータを格納します。	サービス要求前	ユーザ	任意
	XATMI 要求データファイル	XATMI インタフェースのクライアント UAP シミュレート機能使用時，サービス要求先に渡すデータを格納します。	サービス要求前	ユーザ	任意
サービス応答データファイル	RPC 応答データファイル	RPC インタフェースのサーバ UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	シミュレート対象の SPP の起動時	ユーザ	任意
	XATMI 応答データファイル	XATMI インタフェースのサーバ UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	シミュレート対象の SPP の起動時	ユーザ	任意
XATMI 受信データファイル		XATMI インタフェースの会話型のサービス要求の際に，UAP の tprecv 関数で受信するデータを格納します。	サービス要求前，またはシミュレート対象の SPP の起動時※	ユーザ	任意
MCF 受信メッセージファイル	非同期型受信メッセージファイル	MCF シミュレート機能使用時，MHP の dc_mcf_receive 関数に渡すメッセージを格納します。	サービス要求前	ユーザ	任意
	同期型受信メッセージファイル	MCF シミュレート機能使用時，UAP の dc_mcf_recvsync 関数や dc_mcf_sendrecv 関数に渡すメッセージを格納します。	サービス要求前	ユーザ	任意
運用コマンド結果データファイル		運用コマンドシミュレート機能使用時，実行結果として UAP に返すデータを格納します。	サービス要求前	ユーザ	任意

注

ユーザが作成するファイルは，次のファイルを除いたすべてのファイルで，オフラインテスタのものがそのまま使用できます。

・XATMI 受信データファイル

・同期型受信メッセージファイル

・運用コマンド結果データファイル

ただし，複数のオフラインテスタ用データファイルを cat コマンドで一つのファイルに編集すれば，これらのファイルでもオフラインテスタで使用できます。

注※

クライアント UAP シミュレート機能使用時はサービス要求前に，サーバ UAP シミュレート機能使用時はシミュレート対象の SPP の起動時に作成します。

表 3-4　ユーザが作成するテスタファイルに付けるファイル名

テスタファイル種別		ファイル名
サービス要求データファイル	RPC 要求データファイル	任意
	XATMI 要求データファイル	
サービス応答データファイル	RPC 応答データファイル	$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/svc サービス名※1
	XATMI 応答データファイル	$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/xsv サービス名※1
XATMI 受信データファイル		$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/xrv サービス名※1，※2，※3
MCF 受信メッセージファイル	非同期型受信メッセージファイル	$DCDIR/spool/uto/テストユーザ ID/xx....xx（xx....xx は任意）
	同期型受信メッセージファイル	$DCDIR/spool/uto/テストユーザ ID/recv 論理端末名称※4（ヘッダセグメントファイル：$DCDIR/spool/uto/テストユーザ ID/recvh 論理端末名称）
運用コマンド結果データファイル	SPP のサービス関数で使用する場合	$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/cmd サービス名※1
	SUP と SPP のメイン関数で使用する場合	$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/cmd

注※1

サービス名が 11 文字を超える場合，サービス名の先頭から 5 文字と末尾から 6 文字を合わせた名前にします。

（例）サービス名 "uapservice0001"の場合 → "uapsece0001"

注※2

サービス名が 15 文字を超える場合，サービス名の先頭からの 15 文字の中で，先頭から 5 文字と末尾から 6 文字を合わせた名前にします。

（例）サービス名 "uapxatmiservice0001"の場合 → "uapxaervice"

注※3

クライアント UAP シミュレート機能使用時は任意です。

注※4

dc_mcf_recvsync 関数，dc_mcf_sendrecv 関数の引数に設定した論理端末名称です。

3.3.1　サービス要求データファイル

(1)　RPC 要求データファイル

RPC インタフェースのクライアント UAP シミュレート機能使用時，utosppsvc コマンドで指定したサービスのサービス関数に渡すデータを格納します。一つのファイルには，一つのデータを作成します。

(a)　ファイルの構造

(b)　ファイルの内容

項目	位置	長さ（バイト）	内容
データ長	0	4	サービス関数に渡すデータの長さを指定します (1～DCRPC_MAX_MESSAGE_SIZE の指定値)。
応答領域長	4	4	サービス関数に渡す応答領域長を指定します (1～DCRPC_MAX_MESSAGE_SIZE の指定値)。
データ	8	n	サービス関数に渡すデータを指定します。

(c)　注意事項

- サービス関数の引数との関係を次に示します。

```
サービス関数　(in, in_len, out, out_len)
              1.  2.      3.
```

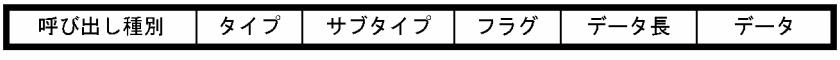
  1. データ
  2. データ長
  3. 応答領域長

- RPC 要求データファイルは，オフラインテスタのファイルも使用できます。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。

(2)　XATMI 要求データファイル

XATMI インタフェースのクライアント UAP シミュレート機能使用時，サービスに対応したサービス関数に渡すデータを格納します。一つのファイルには，一つのデータを作成します。

(a)　ファイルの構造



(b) ファイルの内容

項目	位置	長さ（バイト）	内容
呼び出し種別	0	8	サービスを呼び出す際に，呼び出し元の関数の種別を指定します。 call：tpcall 関数からの呼び出し acall：tpacall 関数からの呼び出し connect：tpconnect 関数からの呼び出し
タイプ	8	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE
サブタイプ	16	16	サブタイプを 16 文字以下の文字列で指定します。ただし，タイプに'X_OCTET'を指定した場合は，サブタイプはヌル文字とします。
フラグ	32	4	サービス関数に渡す次のフラグを，16 進数で指定します。呼び出し情報での指定によって限定されます。 0x00000000：0（call，acall 指定時だけ） 0x00000004：TPNOREPLY（acall 指定時だけ） 0x00000008：TPNOTRAN 0x00000100：TPNOCHANGE（call，acall 指定時だけ） 0x00000800：TPSENDONLY（connect 指定時だけ） 0x00001000：TPRECVONLY（connect 指定時だけ） なお，サービス要求時に TPNOTIME と TPSIGRSTRT は必ず設定されます。TPNOBLOCK は設定しません。
データ長	36	4	サービス関数に渡すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	40	n	サービス関数に渡すデータを指定します。

(c) 注意事項

- サービス関数の引数との関係を次に示します。

```
void tpservice(svcinf)
     TPSVCINFO *svcinf;

     struct TPSVCINFO {
               char name[32];
               char *data;       →1.
               long len;         →2.
               long flags;
               int cd;
          }
```

1. タイプ，サブタイプにマッピングされたデータが格納されているアドレス
2. data で示すデータの長さ

- utoxsppsvc コマンドで発行する XATMI 関数との関係を次に示します。

```
idata=tpalloc (type, subtype, ilen);
                 2.      3.      4.
tpcall (svc, idata, ilen, odata, olen, flags);
  1.                  4.                  5.
tpacall (svc, idata, ilen, flags);
  1.                   4.    5.
tpconnect (svc, idata, ilen, flags);
    1.                   4.     5.
```

  1. 呼び出し種別に対応した XATMI 関数
  2. タイプ名
  3. サブタイプ名
  4. データ長
  5. フラグ（指定したフラグを実際のフラグの値に変換して指定）

- XATMI 要求データファイルは，オフラインテスタのファイルも使用できます。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- サブタイプ名が 16 文字に満たない場合，サブタイプ名の後ろにヌル文字を付けて指定してください。
- タイプが X_OCTET 以外の場合，ファイル内で指定したサブタイプのデータ長と，utoxsppsvc コマンドで指定するサブタイプのデータ長が異なると，XATMI 要求データファイルのデータ不正になります。
- フラグには TPNOCHANGE のコードを指定できますが，無視されます。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。

  また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

  バウンダリ調整の内容は，stbmake コマンドで作成したスタブソース，および stbmake コマンドに-p オプションを指定した場合の出力結果を参照して，確認してください。

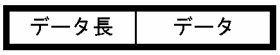
3.3.2　サービス応答データファイル

(1)　RPC 応答データファイル

RPC インタフェースのサーバ UAP シミュレート機能使用時，シミュレート対象の SPP にサービスを要求した際に，サービスの要求元へ返す応答データを格納します。一つのファイルには，一つのデータを作成します。

また，クライアント UAP シミュレート機能使用時，テスト対象の UAP からの応答データを格納します。

(a) ファイルの構造



(b) ファイルの内容

項目	位置	長さ（バイト）	内容
データ長	0	4	サービス要求元へ返すデータの長さを指定します(0～2147483647)。
データ	4	n	サービス要求元へ返すデータを指定します。

(c) 注意事項

- サービス要求元のサービス要求関数（dc_rpc_call 関数）の引数との関係を次に示します。

```
dc_rpc_call (……, in, in_len, out, out_len)
                                  1.
```

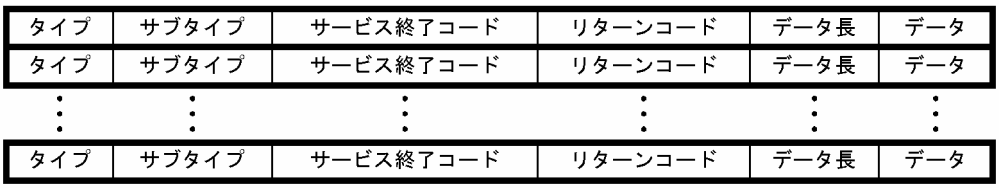
1. データ

- RPC 応答データファイルは，オフラインテスタのファイルも使用できます。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。

(2) XATMI 応答データファイル

XATMI インタフェースのサーバ UAP シミュレート機能使用時，シミュレート対象の SPP にサービスを要求した際に，サービスの要求元へ返す応答データを格納します。一つのファイルには，複数のデータが作成できます。

(a) ファイルの構造



(b) ファイルの内容

項目	位置	長さ（バイト）	内容
タイプ	0	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE

項目	位置	長さ（バイト）	内容
サブタイプ	8	16	サブタイプを 16 文字以下の文字列で指定します。 ただし，タイプに'X_OCTET 'を指定した場合は，サブタイプはヌル文字とします。
サービス終了コード	24	4	tpreturn 関数の rval で指定する次のどちらかの値を，16 進数で指定します。この値は，tperrno 領域に設定されます。 0x04000000：TPSUCCESS 0x20000000：TPFAIL
リターンコード	28	4	tpreturn 関数の rcode で指定する値を 16 進数で指定します。この値は，tpurcode 領域に設定されます。
データ長	32	4	サービス要求元に返すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	36	n	サービス要求元に返すデータを指定します。

(c) 注意事項

- サービス終了関数（tpreturn 関数）の引数との関係を次に示します。

```
tpreturn (rval, rcode, data, len, ……)
          1.    2.    3.    4.
```

  1. サービス終了コード
  2. リターンコード
  3. タイプ，サブタイプで確保されたバッファに格納したデータ
  4. データ長

- XATMI 応答データファイルは，オフラインテスタのファイルも使用できます。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。

  また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

  バウンダリ調整の内容は，stbmake コマンドで作成したスタブソース，および stbmake コマンドに-p オプションを指定した場合の出力結果を参照して，確認してください。

3.3.3 XATMI 受信データファイル

会話型のサービス要求時，tprecv 関数で UAP が受け取るメッセージを格納します。一つのファイルには複数のデータを作成でき，順番に tprecv 関数へ渡します。

XATMI 受信データファイルは，サービスごとに作成します。

(1) ファイルの構造

(2) ファイルの内容

項目	位置	長さ（バイト）	内容
共用領域	0	36	XATMI 送信データファイルとの共用領域です。 空白，またはヌル文字を指定します。
タイプ	36	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE
サブタイプ	44	16	サブタイプを 16 文字以下の文字列で指定します。 ただし，タイプに'X_OCTET'を指定した場合は，サブタイプはヌル文字とします。
イベントフラグ	60	4	tprecv 関数に渡す次のイベントフラグのどれかを，16 進数で指定します。 0x00000000：0 0x00000001：TPEV_DISCONIMM 0x00000002：TPEV_SVCERR 0x00000004：TPEV_SVCFAIL 0x00000008：TPEV_SVCSUCC 0x00000020：TPEV_SENDONLY
データ長	64	4	tprecv 関数に渡すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	68	n	tprecv 関数に渡すデータを指定します。

(3) 注意事項

- メッセージ受信関数（tprecv 関数）の引数との関係を次に示します。

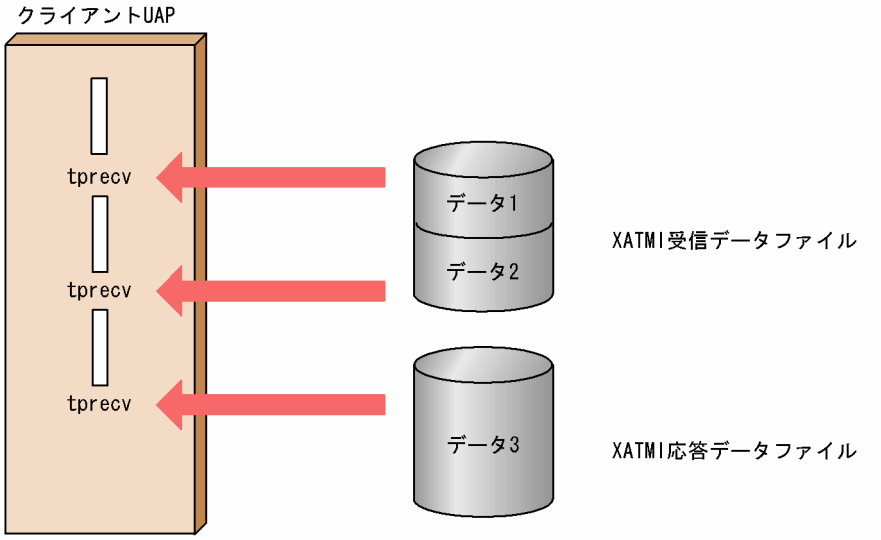
```
tprecv（……, data, len, ……, revent）
              1.    2.          3.
```

1. タイプ，サブタイプで確保されたバッファに格納したデータ
2. データ長

3. イベントフラグ

- サーバ UAP シミュレート機能で使用する場合に tprecv 関数に渡すデータと，XATMI 受信データファイル，XATMI 応答データファイルとの関係を，次の図に示します。

図 3-1　受信データとテスタファイルの関係



サーバ UAP シミュレート機能で使用する場合，受信データは実行単位に作成します。一つのサービスが複数回 tprecv 関数を発行する場合は，発行回数分のデータを作成しておく必要があります。ただし，最後の tprecv 関数に渡すデータは，XATMI 応答データファイル内にデータを格納できます。

データの数より tprecv 関数の発行回数が多い場合は，tpreturn 関数のデータが受信されたものとし，それ以降は，エラーとなります。

XATMI 受信データファイルは，サービス単位にオープン，クローズします。

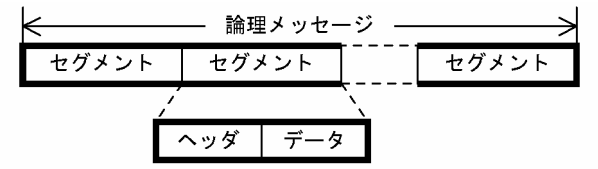
- XATMI 受信データファイルは，オフラインテスタのファイルは使用できません。ただし，複数のオフラインテスタ用 XATMI 受信データファイルを，cat コマンドで一つのファイルに編集すると，オンラインテスタでも使用できます。
- サーバ UAP シミュレート機能使用時に出力された送信データを格納した XATMI 送信データファイルは，そのまま XATMI 受信データファイルとして使用できます。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- サーバ UAP シミュレート機能で使用する場合，イベントフラグに TPEV_SENDONLY 以外を指定すると，クライアント UAP で発行した tprecv 関数は，それ以上会話を続行できないイベントを受け取ります。そのため，それ以降のデータは使用されません。また，イベント発生時のグローバル変数 tpurcode には，0 が設定されます。
- クライアント UAP シミュレート機能で使用する場合，イベントフラグの指定は，0 と TPEV_SENDONLY だけが有効です。それ以外の指定は無効になります。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。

また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

バウンダリ調整の内容は，stbmake コマンドで作成したスタブソース，および stbmake コマンドに-p オプションを指定した場合の出力結果を参照して，確認できます。

3.3.4 MCF 受信メッセージファイル

1 論理メッセージは，一つまたは複数のセグメントで構成されています。1 セグメントはセグメント情報の入ったヘッダ部と，メッセージデータであるデータ部で構成されています。



セグメントには，次の五つの種別があります。

- 単セグメント
  1 論理メッセージが 1 セグメントで構成されている場合のセグメント
- 先頭セグメント
  1 論理メッセージが複数セグメントで構成されている場合の先頭セグメント
- 中間セグメント
  1 論理メッセージが複数セグメントで構成されている場合の中間セグメント
- 最終セグメント
  1 論理メッセージが複数セグメントで構成されている場合の最終セグメント
- ヘッダセグメント
  二つのメッセージを連結する場合に，先頭に付けるセグメント

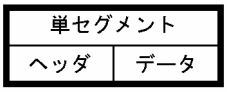
なお，セグメントの種別は，ヘッダ部に指定します。

(1) 非同期型受信メッセージファイル

MCF 関数（dc_mcf_receive 関数）で，UAP が受け取るメッセージを格納します。一つのファイルには，一つの論理メッセージを作成します。ヘッダセグメントを使用すると，メッセージの前にデータを付けられます。

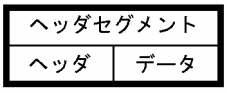
(a) ファイルの構造

■ 1 論理メッセージ 1 セグメントのとき



■ 1 論理メッセージ複数セグメントのとき

■ ヘッダセグメントのとき



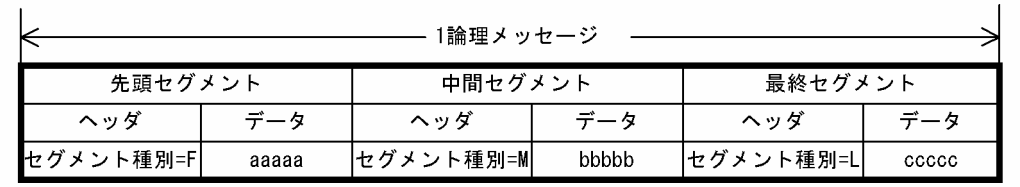
(b) ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	入出力論理端末名称	0	9	MCF 関数でやり取りするための論理端末名称を指定します（最後のヌル文字を含みます）。 複数セグメントのときは，各セグメントで同じものを指定します。
	マップ名	9	9	マップ名を指定します（最後のヌル文字を含みます）。 複数セグメントのときは，各セグメントで同じものを指定します。 この指定はマップ名を返す関数だけ有効です。
	予備	18	9	ヌル文字を指定します。
	セグメント種別	27	1	次の文字のどれかを指定します。 F：先頭セグメントのとき M：中間セグメントのとき L：最終セグメントのとき O：単セグメントのとき H：ヘッダセグメントのとき
	メッセージ長	28	4	メッセージの長さを指定します（0～2147483647）。
データ	メッセージ	32	n	セグメントの実体（データ）を，メッセージ長で指定した長さで指定します。

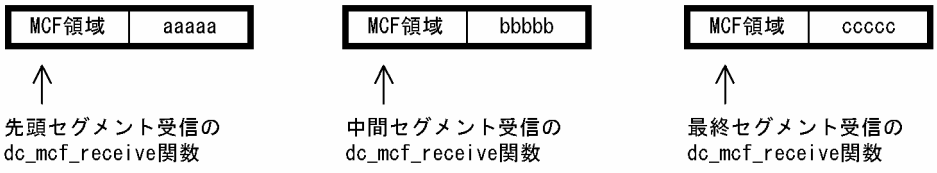
(c) 注意事項

- 非同期型受信メッセージファイルの構造と，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。

＜ファイルの構造＞



＜UAPが受信するメッセージ＞



- ヘッダセグメントを連結すると，別ファイルに作成したデータを先頭セグメント，または単セグメントと合成して UAP に渡せます。ヘッダセグメントと，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。

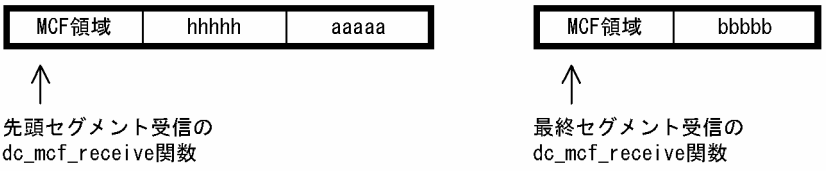
＜ファイルAの構造＞

ヘッダセグメント	
ヘッダ	データ
セグメント種別=H	hhhhh

＜ファイルBの構造＞

先頭セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ
セグメント種別=F	aaaaa	セグメント種別=L	bbbbb

＜UAPが受信するメッセージ（ファイルAとファイルBを連結）＞



- セグメント種別が F（先頭セグメント）のセグメントと M（中間セグメント）のセグメントは，同じものとして扱われます。また，L（最終セグメント）のセグメントと O（単セグメント）のセグメントも，同じものとして扱われます。

  例えば，セグメント種別が F，M，L の 3 セグメントから成るファイルは，M，M，O の 3 セグメントから成るファイルと同等に扱われます。

- MHP とのメッセージ送受信時にセグメントヘッダに指定したセグメント種別と，実行時のファイルの種別との関係を次に示します。なお，セグメント種別の指定に誤りがある場合は，最初のメッセージ受信時に受信要求関数がエラーリターンします。

■ ヘッダセグメント以外を格納した非同期型受信メッセージファイルの場合

セグメント種別に L，または O を指定したメッセージを受信すると，メッセージ完了と判断して以降のセグメントを無視します。

セグメント種別			MHP が受信するセグメント
先頭セグメント	中間セグメント	最終セグメント	
F	M	L	F，M，L
F	L	X	F，L※1
L	X	X	L※2
X	M	L	受信しません※3。
F	X	L	

（凡例）

X：F，M，L，および O 以外の指定

注※1

3 回目の受信要求は，1 論理メッセージ受信済みとしてエラーリターンします。

注※2

中間セグメント以降は無視されます。

注※3

セグメント種別不正としてメッセージを出力し，受信要求関数がエラーリターンします。

■ ヘッダセグメントを格納した非同期型受信メッセージファイルの場合

ファイルの最初のセグメントだけが有効になります。

セグメント種別	MHP が受信するセグメント
H	H※1
H ＋ X	
X	受信しません※2。
X ＋ H	

（凡例）

X：H 以外の指定

注※1

ただし，F，M，L，および O のどれかと連結した形で渡されます。

注※2

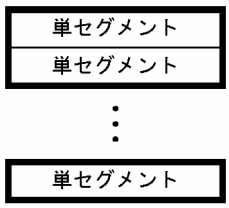
セグメント種別不正としてメッセージを出力し，受信要求関数がエラーリターンします。

(2) 同期型受信メッセージファイル

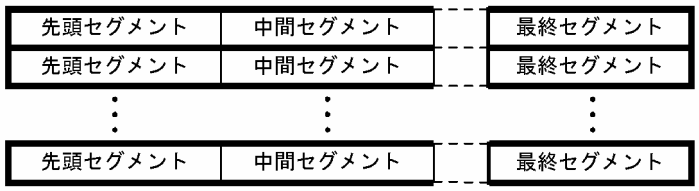
MCF 関数（dc_mcf_recvsync 関数，dc_mcf_sendrecv 関数）で，UAP が受け取る同期型のメッセージを格納します。同期型受信メッセージファイルには，複数論理メッセージ分のデータを格納できます。ヘッダセグメントを使用すると，メッセージの前にデータを付けられます。

(a) ファイルの構造

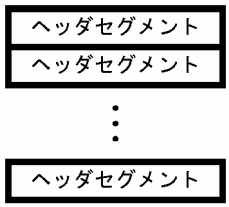
■ 1 論理メッセージ 1 セグメントのとき



■ 1 論理メッセージ複数セグメントのとき



■ ヘッダセグメントのとき



(b) ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	入出力論理端末名称	0	9	MCF 関数でやり取りするための論理端末名称を指定します（最後のヌル文字を含みます）。 複数セグメントのときは，論理メッセージごとに各セグメントで同じものを指定します。
	マップ名	9	9	マップ名を指定します（最後のヌル文字を含みます）。 複数セグメントのときは，各セグメントで同じものを指定します。 この指定はマップ名を返す関数だけ有効です。
	予備	18	9	ヌル文字を指定します。
	セグメント種別	27	1	次の文字のどれかを指定します。 F：先頭セグメントのとき M：中間セグメントのとき L：最終セグメントのとき O：単セグメントのとき H：ヘッダセグメントのとき

項目		位置	長さ（バイト）	内容
ヘッダ	メッセージ長	28	4	メッセージの長さを指定します（0～2147483647）。
データ	メッセージ	32	n	セグメントの実体（データ）を，メッセージ長で指定した長さで指定します。

(c) 注意事項

- 同期型受信メッセージファイルの構造と，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。

＜ファイルの構造＞

先頭セグメント		中間セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ	ヘッダ	データ
セグメント種別=F	aaaaa	セグメント種別=M	bbbbb	セグメント種別=L	ccccc
先頭セグメント		中間セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ	ヘッダ	データ
セグメント種別=F	ddddd	セグメント種別=M	eeeee	セグメント種別=L	fffff

＜UAPが受信するメッセージ＞

MCF領域	aaaaa
MCF領域	bbbbb
MCF領域	ccccc

← 先頭セグメント受信のdc_mcf_recvsync関数
← 中間セグメント受信のdc_mcf_recvsync関数
← 最終セグメント受信のdc_mcf_recvsync関数

MCF領域	ddddd
MCF領域	eeeee
MCF領域	fffff

← 先頭セグメント受信のdc_mcf_recvsync関数
← 中間セグメント受信のdc_mcf_recvsync関数
← 最終セグメント受信のdc_mcf_recvsync関数

- ヘッダセグメントを連結すると，別ファイルに作成したデータを先頭セグメント，または単セグメントと合成して UAP に渡せます。ヘッダセグメントと，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。

＜ファイルAの構造＞

ヘッダセグメント	
ヘッダ	データ
セグメント種別=H	h0001

ヘッダセグメント	
ヘッダ	データ
セグメント種別=H	h0002

＜ファイルBの構造＞

先頭セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ
セグメント種別=F	aaaaa	セグメント種別=L	bbbbb

先頭セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ
セグメント種別=F	ccccc	セグメント種別=L	ddddd

＜UAPが受信するメッセージ（ファイルAとファイルBを連結）＞

MCF領域	h0001	aaaaa
MCF領域	bbbbb	

← 先頭セグメント受信のdc_mcf_sendrecv関数
← 最終セグメント受信のdc_mcf_recvsync関数

MCF領域	h0002	ccccc
MCF領域	ddddd	

← 先頭セグメント受信のdc_mcf_sendrecv関数
← 最終セグメント受信のdc_mcf_recvsync関数

- MCF シミュレート機能を使用して，UAP で同期型メッセージ受信によって複数論理メッセージを受信する場合，受信メッセージの先頭に付けるヘッダセグメントは，各論理メッセージに対応させて格納してください。複数論理メッセージのうち，ヘッダセグメントの必要がない論理メッセージがある場合は，メッセージ長に 0 を指定したダミーのヘッダセグメントを格納しておく必要があります。なお，ヘッダセグメントが必要な論理メッセージがない場合は，ヘッダセグメントを格納したファイルは作成する必要はありません。ヘッダセグメントと，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。

＜ファイルAの構造＞

ヘッダセグメント	
ヘッダ	データ
セグメント種別=H	なし

ヘッダセグメント	
ヘッダ	データ
セグメント種別=H	h0001

＜ファイルBの構造＞

先頭セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ
セグメント種別=F	aaaaa	セグメント種別=L	bbbbb

先頭セグメント		最終セグメント	
ヘッダ	データ	ヘッダ	データ
セグメント種別=F	ccccc	セグメント種別=L	ddddd

＜UAPが受信するメッセージ（ファイルAとファイルBを連結）＞

MCF領域	aaaaa
MCF領域	bbbbb

← 先頭セグメント受信のdc_mcf_sendrecv関数
← 最終セグメント受信のdc_mcf_recvsync関数

MCF領域	h0001	ccccc
MCF領域	ddddd	

← 先頭セグメント受信のdc_mcf_sendrecv関数
← 最終セグメント受信のdc_mcf_recvsync関数

- セグメント種別がF（先頭セグメント）のセグメントとM（中間セグメント）のセグメントは，同じものとして扱われます。また，L（最終セグメント）のセグメントとO（単セグメント）のセグメントも，同じものとして扱われます。

  例えば，セグメント種別がF，M，Lの３セグメントから成るファイルは，M，M，Oの３セグメントから成るファイルと同等に扱われます。

- MHP とのメッセージ送受信時にセグメントヘッダに指定したセグメント種別と，実行時のファイルの種別との関係を次に示します。なお，セグメント種別の指定に誤りがある場合は，最初のメッセージ受信時に受信要求関数がエラーリターンします。

■ ヘッダセグメント以外を格納した同期型受信メッセージファイルの場合

セグメント種別に L，または O を指定したメッセージを受信すると，メッセージ完了と判断して以降のセグメントを無視します。

セグメント種別			MHP が受信するセグメント
先頭セグメント	中間セグメント	最終セグメント	
F	M	L	(F，M，L)
M	M	L	(M，M，L)
O	O	O	(O)，(O)，(O)
F	L	M	(F，L)，(M)
X	M	L	受信しません※。
F	X	L	

(凡例)

X：F，M，L，および O 以外の指定

(　)：1 論理メッセージの範囲

注※

セグメント種別不正としてメッセージを出力し，受信要求関数がエラーリターンします。

■ ヘッダセグメントを格納した同期型受信メッセージファイルの場合

ファイルの全ヘッダセグメントが有効になります。

セグメント種別	MHP が受信するセグメント
H	H※1
H + H	H，H※1
H + X	受信しません※2。
X	
X + H	

(凡例)

X：H 以外の指定

注※1

ただし，F，M，L，および O のどれかと連結した形で渡されます。

注※2

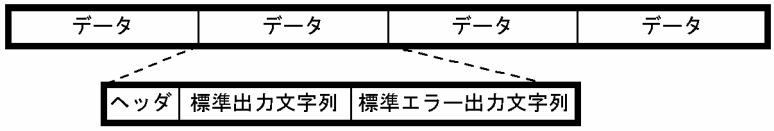
セグメント種別不正としてメッセージを出力し，受信要求関数がエラーリターンします。

3.3.5　運用コマンド結果データファイル

運用コマンドシミュレート機能使用時に，コマンドの実行結果として UAP に返すデータを格納します。一つのファイルには，一つのサービスで dc_adm_call_command 関数を発行する回数分のデータを格納します。

運用コマンド結果データファイルは，サービスごとに作成します。

(1)　ファイルの構造



(2)　ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	運用コマンドの結果コード	0	4	dc_adm_call_command 関数の引数 stat に設定する値を指定します。
	標準出力文字列長	4	4	標準出力文字列の長さ（ヌル文字を含みます）を指定します（0～2147483647）。
	標準エラー出力文字列長	8	4	標準エラー出力文字列の長さ（ヌル文字を含みます）を指定します（0～2147483647）。
標準出力文字列		12	n	dc_adm_call_command 関数の引数 outmsg に設定するデータを指定します（最後のヌル文字を含みます。ヌル文字が付けられていない場合は文字列の最後の文字をヌル文字に置き換えます）。 標準出力文字列長が 0 の場合は指定しても無視されます。
標準エラー出力文字列		−	n	dc_adm_call_command 関数の引数 errmsg に設定するデータを指定します（最後のヌル文字を含みます。ヌル文字が付けられていない場合は文字列の最後の文字をヌル文字に置き換えます）。 標準エラー出力文字列長が 0 の場合は指定しても無視されます。

（凡例）

−：該当しません。

(3)　注意事項

- 運用コマンド結果データファイルは，オフラインテスタのファイルも使用できます。ただし，複数回 dc_adm_call_command 関数を発行する場合は，発行回数分のデータ（ファイル）を，cat コマンドで一つのファイルに編集する必要があります。

- 標準出力文字列と標準エラー出力文字列の最後には，ヌル文字を付けてください。ヌル文字が付いていない場合は，文字列の最後の文字がヌル文字に置き換えられます。また，文字列長に 0 を指定した場合は，文字列を指定しても無視されます。
- DML の SEND 文で運用コマンドを発行する場合は，データ部に次のように指定します。

  (i) テスタファイル作成機能を使用しない場合

  標準出力文字列長

  0 を指定します。

  標準エラー出力文字列長

  0 を指定します（標準エラー出力を受け取らない指定の場合）。

  (ii) テスタファイル作成機能を使用する場合

  標準出力文字列長

  1 を指定します。

  標準出力文字列

  '¥0'を指定します。

  標準エラー出力文字列長

  1 を指定します（標準エラー出力を受け取らない指定の場合）。

  標準エラー出力文字列

  '¥0'を指定します（標準エラー出力を受け取らない指定の場合）。

3.4　ファイルの作成

ここでは，テスタファイルを格納するために必要なディレクトリの作成，テスタファイルを簡単に作成するためのテストデータ定義ファイルの作成，およびオンラインテスタが作成するファイルについて説明します。

3.4.1　テスト用ディレクトリの作成

テスト用のファイル（テスタファイル）を格納するディレクトリ$DCDIR/spool/uto は，オンラインテスタのインストール時にモード 0777 で OpenTP1 が作成します。また，UAP 実行中のトレースファイル，または MCF 送信メッセージファイル作成時にディレクトリ$DCDIR/spool/uto/テストユーザ ID がなければ，オンラインテスタがモード 0777 で作成します。

テスト前に MCF 受信メッセージファイルなどをユーザが作成する場合は，ユーザがディレクトリ$DCDIR/spool/uto/テストユーザ ID（必要があれば，$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名）を作成します。モードは，UAP でこのファイルを作成できるように設定しておいてください。

3.4.2　テストデータ定義ファイルの作成

テストデータ定義ファイルを作成すると，テスタファイル作成機能で簡単にテスタファイルを作成できます。

テストデータ定義ファイルの名称は任意です。テストデータ定義ファイルから作成できるテスタファイルを次に示します。

- RPC 要求データファイル
- XATMI 要求データファイル
- RPC 応答データファイル
- XATMI 応答データファイル
- XATMI 受信データファイル
- 非同期型受信メッセージファイル
- 同期型受信メッセージファイル
- 運用コマンド結果データファイル

テストデータ定義ファイルの作成手順を，次に示します。

1. テキストエディタでテストデータ定義ファイルを作成します。
2. 記述内容を確認してテストデータ定義ファイルをクローズします。
3. utofilcre コマンドにテストデータ定義ファイルを指定して実行します。

テスタファイルが作成されます。

(1) 形式

```
#コメント          ……1.
start テスタファイル識別子 テスタファイル種別 出力先ファイル名 ……2.
キーワード = 入力データ          ……5.
キーワード = 入力データ
sep          ……3.
キーワード = 入力データ
  :          :
  :          :
キーワード = 入力データ
end          ……4.
```

なお，形式中の太字の数字は，「(3) 説明」の番号と対応しています。

(2) 機能

テスタファイルに設定するテストデータを定義すると，テスタファイル作成コマンドでテスタファイルを作成できます。

なお，定義ファイルでの 1 行の長さは，改行コードを含めて 512 バイトまでです。

(3) 説明

1.コメント文

コメントを記述します。行の先頭に'#'を記述します。

- **コメント**

  コメントを，1 行で記述します。

2.start 文

一つのテスタファイルの入力データの開始を宣言します。各テスタファイルの入力データごとに，必ず記述します。

一つのテストデータ定義ファイル内に複数のテスタファイルの入力データを作成した場合，1 テスタファイルの入力データの最後に end 文を記述します。

- **ファイル識別子　　～〈14 文字以内の英数字〉**

テストデータ定義ファイル中に記述した各テスタファイルのデータを区別するための識別子を指定します。1 テストデータ定義ファイルで，一意の名称を指定します。識別子に使用する英数字は，a～z，A～Z，0～9 の値で指定してください。

- テスタファイル種別

  テスタファイルの種別を指定します。指定できるテスタファイル種別を次に示します。

  RRQ：RPC 要求データファイル

  XRQ：XATMI 要求データファイル

  RRT：RPC 応答データファイル

  XRT：XATMI 応答データファイル

  XRV：XATMI 受信データファイル

  NRV：非同期型受信メッセージファイル

  SRV：同期型受信メッセージファイル

  COM：運用コマンド結果データファイル

- 出力先ファイル名　　～〈パス名〉

  入力データを基に作成する，テスタファイルの名称を指定します。

  一つのテストデータ定義ファイル内に，複数のテスタファイル種別の入力データを作成する場合は，ファイル種別ごとに異なった出力先ファイル名を指定します。

  テスタファイル種別が異なる入力データで出力先ファイル名を重複して指定すると，指定したファイルはテストデータが追加された状態で作成されます。この場合，エラーにはなりませんが，作成されたテスタファイルがテストで使用できないことがあります。

  また，すでにあるファイル名を指定した場合は，テストデータがそのファイルに追加された状態で作成されます。

3.sep 文

一つのファイルに複数のデータを記述するテスタファイルを作成する場合に，一つのデータの区切りを指定します。

次のテスタファイルを作成する際に指定できます。

- XATMI 受信データファイル
- 同期型受信メッセージファイル
- 運用コマンド結果データファイル

4.end 文

一つのテスタファイルの入力データの終了を宣言します。各テスタファイルの入力データごとに，必ず記述します。

5.入力データ定義文

各テスタファイルの入力データを定義します。

入力データには，あらかじめ設定できる情報が決められている**固定情報データ**と，ユーザが任意の情報を設定できる**ユーザデータ**（キーワードが data の場合）とがあります。一つのテストデータ内では，固定情報データはすべてユーザデータの前に記述してください。

なお，一つのテストデータ内では，入力データを重複して指定できません。ただし，運用コマンド結果データファイルでは，標準出力文字列データと標準エラー出力文字列データ設定するため，ユーザデータを 2 回指定します。

固定情報データを指定する場合の入力データの形式は，「(5)　テスタファイルのキーワードに対応する入力データの形式」の表を参照してください。

- **キーワード**

  各テスタファイル特有のデータを区別するための，キーワードを指定します。

  キーワードの前後の空白文字やタブコードは無視されます。

- **入力データ**

  キーワードに対応する入力データを指定します。

  入力データの前後の空白文字やタブコードは無視されます。

(4)　入力データにユーザデータを指定する場合に必要な設定

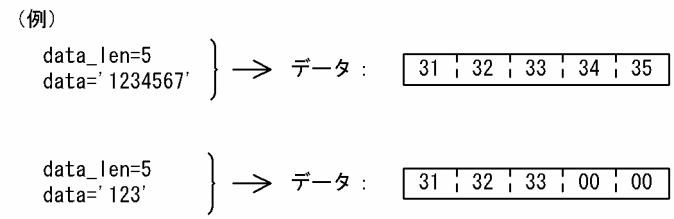
ユーザデータを指定する場合の，入力データの形式を次に示します。

(a)　ユーザデータ長の設定

ユーザデータ全体のデータ長は，固定情報データとして次の形式で設定します。

data_len=バイト数

ユーザデータとして設定したデータが，バイト数に設定したデータ長を超えた場合は，データを切り捨ててメッセージを出力します。データがデータ長に満たない場合は，それ以上のデータは設定しません。



(b)　ユーザデータの初期化

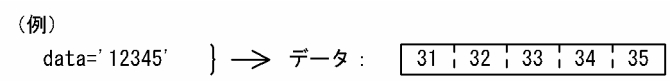
ユーザデータは，テスタファイル作成コマンドで，指定されたユーザデータ長分を初期化します。

(c)　文字データの設定

文字データは，次の形式で設定します。

data='データ'

文字データで記述した場合は，データの最後にヌル文字を付けません。



(d)　バイナリデータの設定

バイナリデータは，次の形式で設定します。

data=データ

データは 10 進数と 16 進数で記述できます。記述方法を次に示します。

- **10 進数**：数値をそのまま設定します。
- **16 進数**：数値の前に'0x'を付けます。

(例)

data=5　→　データ：10 進数の 5
data=0x05　→　データ：16 進数の 5

データは，int 型で設定されます。

(e)　特殊文字の設定

改行コード，タブコード，ヌル文字，' ' '（一重引用符）および'¥'は，文字データ中で特殊文字として扱います。これらの文字を記述する場合は，次の形式で記述してください。

記述する文字	記述形式
改行コード	¥n
タブコード	¥t
ヌル文字	¥0
' ' '（一重引用符）	¥'
'¥'	¥¥

(f)　ファイルからのデータ読み込みの設定

ファイルから読み込んだデータをユーザデータとして使用する場合は，次の形式で設定します。

data=(file)ファイルのパス名

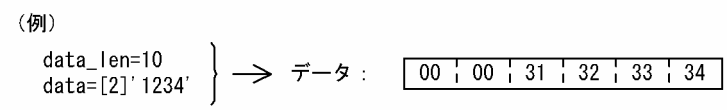
(例)

data=(file)/tmp/datafile　→　データ：/tmp/datafile のデータを設定

(g)　ユーザデータの開始位置の設定

ユーザデータを任意の位置から設定する場合は，次の形式で設定します。

data=[ユーザデータ先頭からのオフセット]データ



(h)　複数データ形式の設定

ユーザデータに複数のデータ型を使用する場合は，次の形式で設定します。

```
data=データ
    =データ
       :
       :
```

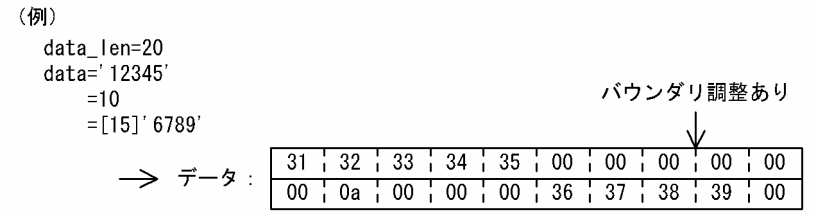
（例）

```
data=0x00000001   →   1番目のデータ
   ='ABCDEF'      →   2番目のデータ
```

(i)　バウンダリの調整

複数のデータ型の記述で前後のデータ型が異なる場合，テスタファイル作成コマンドで自動的に前のデータとのバウンダリを調整してデータを設定します。ただし，次の場合にはバウンダリを調整しません。

- ファイルからユーザデータを読み込む場合
- ユーザデータの開始位置を設定した場合



(5)　テスタファイルのキーワードに対応する入力データの形式

各テスタファイルのキーワードと，それに対応する入力データの形式の一覧を，以降の表に示します。

指定する情報については，「3.3　ユーザが作成するファイル」の各テスタファイルの説明を参照してください。

表 3-5　RPC 要求データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
out_len	応答領域長	dc_rpc_call 関数に指定する応答領域長を，10 進数または 16 進数で設定します。data より前で設定します。
data_len	データ長	dc_rpc_call 関数でサーバ UAP に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	dc_rpc_call 関数でサーバ UAP に渡すユーザデータを設定します。

表 3-6　XATMI 要求データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
call_kind	呼び出し種別	サービス要求する関数種別として，次の文字列のどれかを設定します。 • call • acall • connect data より前で設定します。
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET • X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
flag	フラグ	サービス関数に渡すフラグとして，次の文字列を設定します。複数のフラグを設定する場合は，'｜'で区切ります。 • 0 • TPNOREPLY • TPNOTRAN • TONOCHANGE • TPSENDONLY • TPRECVONLY data より前で設定します。
data_len	データ長	tpcall 関数，tpacall 関数，tpconnect 関数でサーバ UAP に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	tpcall 関数，tpacall 関数，tpconnect 関数でサーバ UAP に渡すユーザデータを設定します。

表 3-7　RPC 応答データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
data_len	データ長	サービス終了時にクライアント UAP に返すユーザデータ長を，10 進数または 16 進数で設定します。data より前で設定します。
data	データ	サービス終了時にクライアント UAP に返すユーザデータを設定します。

表 3-8　XATMI 応答データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET • X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
rval	サービス終了コード	サービス終了コードとして，次の文字列のどれかを設定します。 • TPSUCCESS • TPFAIL data より前で設定します。
rcode	リターンコード	リターンコードを 10 進数または 16 進数で設定します。data より前で設定します。
data_len	データ長	サービス終了時にクライアント UAP に返すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	サービス終了時にクライアント UAP に返すユーザデータを設定します。

表 3-9　XATMI 受信データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET • X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
event	イベントフラグ	tprecv 関数に渡すイベントフラグとして，次の文字列のどれかを設定します。 • 0 • TPEV_DISCONIMM • TPEV_SVCERR • TPEV_SVCFAIL

キーワード	指定する情報	説明
event	イベントフラグ	• TPEV_SVCSUCC • TPEV_SENDONLY data より前で設定します。
data_len	データ長	tprecv 関数に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	tprecv 関数に渡すユーザデータを設定します。
sep	sep 文	複数サービス分のデータを記述する場合に，1 サービス分のデータの最後に記述します。ただし，最終データのあとには記述しません。

注

複数サービス分のデータを設定する場合は，buff_type 以下のデータを繰り返し設定します。

表 3-10　非同期型受信メッセージファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
termname	入出力論理端末名称	dc_mcf_receive 関数に渡す入出力論理端末名称を，8 文字以内の文字列で設定します。data より前で設定します。
mapname	マップ名	dc_mcf_receive 関数に渡すマップ名を，8 文字以内の文字列で設定します。data より前で設定します。
seg_kind	セグメント種別	dc_mcf_receive 関数に渡すセグメント種別として，次の文字のどれかを設定します。 • F • M • L • 0 • H 複数セグメント分のデータがある場合は，次のどれかの順序で設定します。 • F…M…L • F…F…L • M…M…L • L • H • 0 data より前で設定します。
data_len	メッセージ長	dc_mcf_receive 関数に渡すセグメントのユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	メッセージ	dc_mcf_receive 関数に渡すセグメントのユーザデータを設定します。

注

複数セグメント分のデータを設定する場合は，seg_kind 以下のデータを繰り返し設定します。

表 3-11　同期型受信メッセージファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
termname	入出力論理端末名称	dc_mcf_sendrecv 関数，dc_mcf_recvsync 関数に渡す入出力論理端末名称を，8 文字以内の文字列で設定します。data より前で設定します。
mapname	マップ名	dc_mcf_sendrecv 関数，dc_mcf_recvsync 関数に渡すマップ名を，8 文字以内の文字列で設定します。data より前で設定します。
seg_kind	セグメント種別	dc_mcf_sendrecv 関数，dc_mcf_recvsync 関数に渡すセグメント種別として，次の文字のどれかを設定します。 • F • M • L • 0 • H 複数セグメント分のデータがある場合は，次のどれかの順序で設定します。 • F…M…L • F…F…L • M…M…L • F…M…L…F…M…L • F…F…L…M…M…L • M…M…L…M…M…L • L • H • 0 • L…L…F…M…L…L • 0…0…F…M…L…0 • H…H…H data より前で設定します。
data_len	メッセージ長	dc_mcf_sendrecv 関数，dc_mcf_recvsync 関数に渡すセグメントのユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	メッセージ	dc_mcf_sendrecv 関数，dc_mcf_recvsync 関数に渡すセグメントのユーザデータを設定します。
sep	sep 文	複数メッセージ分のデータを記述する場合に，1 メッセージ分のデータの最後に記述します。ただし，最終データのあとには記述しません。

注 1

複数メッセージ分のデータを設定する場合は，termname 以下のデータを繰り返し設定します。

注 2

複数セグメント分のデータを設定する場合は，seg_kind から data までのデータを繰り返し設定します。

表 3-12　運用コマンド結果データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
status_code	運用コマンドの結果コード	運用コマンドが返す結果コードを，10 進数または 16 進数で設定します。data より前で設定します。
outsize	標準出力文字列長	運用コマンドが標準出力に出力するメッセージ長を，10 進数または 16 進数で設定します。data より前で設定します。
errsize	標準エラー出力文字列長	運用コマンドが標準出力エラーに出力するメッセージ長を，10 進数または 16 進数で設定します。data より前で設定します。
data	標準出力文字列	運用コマンドが標準出力に出力するメッセージを，文字データで設定します。
data	標準エラー出力文字列	運用コマンドが標準エラー出力に出力するメッセージを，文字データで設定します。
sep	sep 文	複数コマンド分のデータを記述する場合に，1 コマンド分のデータの最後に記述します。ただし，最終データのあとには記述しません。

注

複数コマンド分のデータを設定する場合は，status_code 以下のデータを繰り返し設定します。

3.4.3　オンラインテスタが作成するファイル

オンラインテスタ使用時に，オンラインテスタが作成するファイルの一覧とそのファイル名を，以降の表に示します。

表 3-13　オンラインテスタが作成するファイルの一覧

ファイル種別		用途と内容	作成タイミング	削除者	削除タイミング
サービス応答データファイル	RPC 応答データファイル	RPC インタフェースのクライアント UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	サービス要求リターン後※1	ユーザ	任意
	XATMI 応答データファイル	XATMI インタフェースのクライアント UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	サービス要求リターン後※1	ユーザ	任意
XATMI 送信データファイル		XATMI インタフェースの会話型のサービス要求で UAP シミュレート機能を使用する際に，次の関数で送信したデータを格納します。 • tpsend 関数	左記の関数内※2	ユーザ	任意
MCF 送信メッセージファイル		MCF シミュレート機能使用時，次の関数で送信したメッセージを格納します。 • dc_mcf_reply 関数 • dc_mcf_send 関数 • dc_mcf_sendsync 関数 • dc_mcf_sendrecv 関数	左記の関数内※2	ユーザ	任意

ファイル種別	用途と内容	作成タイミング	削除者	削除タイミング
MCF 送信メッセージファイル	• dc_mcf_execap 関数	左記の関数内※2	ユーザ	任意
一時記憶データファイル	MCF シミュレート機能使用時，UAP の dc_mcf_tempput 関数で更新し dc_mcf_tempget 関数で受け取るデータを格納します。	左記の関数内※1	オンラインテスタ※3	dc_mcf_contend 関数発行時
トレースファイル	各 OpenTP1 関数の UAP トレース情報を取得します。	オンラインテスタ（UAP）で最初のトレース情報取得時	ユーザ	任意※4

注※1

すでにファイルがある場合は，そのファイルに上書きします。

注※2

すでにファイルがある場合は，そのファイルに追加して書き込みます。

注※3

dc_mcf_contend 関数を発行する UAP を動作させない場合は，ユーザが任意のタイミングで削除します。

注※4

ファイルが満杯になった場合は，ユーザが別ファイル名で退避したあと，削除します。

表 3-14　オンラインテスタが作成するテスタファイルに付けるファイル名

テスタファイル種別		ファイル名
サービス応答データファイル	RPC 応答データファイル	utosppsvc コマンドで指定したファイル名
	XATMI 応答データファイル	utoxsppsvc コマンドで指定したファイル名
XATMI 送信データファイル		$DCDIR/spool/uto/テストユーザ ID/ユーザサーバ名/xsd サービス名※
MCF 送信メッセージファイル		$DCDIR/spool/uto/テストユーザ ID/sendmsg
一時記憶データファイル		$DCDIR/spool/uto/テストユーザ ID/utotmp 論理端末名称
トレースファイル	ファイル 1	$DCDIR/spool/uto/テストユーザ ID/trace1
	ファイル 2	$DCDIR/spool/uto/テストユーザ ID/trace2

注※

サービス名が 11 文字を超える場合，サービス名の先頭から 5 文字と末尾から 6 文字を合わせた名前にします。

（例）サービス名　"uapservice0001"の場合　→　"uapsece0001"

サービス名が 15 文字を超える場合，サービス名の先頭からの 15 文字の中で，先頭から 5 文字と末尾から 6 文字を合わせた名前にします。

（例）サービス名　"uapxatmiservice0001"の場合　→　"uapxaervice"

4 テストの実行

オンラインテスタでの，テストの実行方法について説明します。

4.1 UAP の作成

MCF シミュレート機能を使用しない UAP は，実業務用の UAP と同様に作成してください。実業務用の UAP の作成方法は，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。

MCF シミュレート機能を使用する UAP は，オンラインテスタが提供するシミュレート関数ライブラリを使用して作成します。ただし，リソースマネジャに TP1/Message Control を登録してあるかどうかで，作成方法が変わります。

リソースマネジャに TP1/Message Control を登録していない場合は，TP1/Message Control の提供するライブラリ（libmcf.a または libmcf2.a）の代わりに，オンラインテスタが提供する MCF シミュレート関数ライブラリとリンケージします※1※2。

そのため，MCF シミュレート関数のライブラリとのリンケージを示す，'-lmuto'を指定します。TP1/Message Control の提供するライブラリとのリンケージを示す，'-lmcf'（または'-lmcf2'）を指定する必要はありません。

また，COBOL 言語や DML で作成した UAP についても，同様に'-lmcf'（または'-lmcf2'）を'-lmuto'に変更してください。

リソースマネジャに TP1/Message Control を登録している場合は，MCF シミュレート関数ライブラリ（libmuto.a）のあとに，TP1/Message Control の提供するライブラリ（libmcf.a または libmcf2.a）を付けてリンケージします。

MCF シミュレート機能を使用する UAP のコンパイルコマンドの例を，次に示します。

● リソースマネジャに TP1/Message Control を登録していない場合

```
cc -qchars=signed -o example exmain.c exsv1.c exsv2.c ex_sstb.c
   -I$DCDIR/include -brtl -bexpall -L$DCDIR/lib
   -lmuto -lbetran2 -ltactk -lc
```

（凡例）

exmain.c：メイン関数

exsv1.c：サービス関数 1

exsv2.c：サービス関数 2

ex_sstb.c：スタブで作成したスタブソース

● リソースマネジャに TP1/Message Control を登録している場合※3

```
cc -qchars=signed -o example exmain.c exsv1.c exsv2.c ex_sstb.c
   $DCDIR/spool/trnrmcmd/userobj/extrn.o
   -I$DCDIR/include -brtl -bexpall -L$DCDIR/lib
   -lmuto -lmcf2 -lmnet -lbetran2 -ltactk -lc
```

（凡例）

exmain.c：メイン関数

exsv1.c：サービス関数 1

exsv2.c：サービス関数 2

ex_sstb.c：スタブで作成したスタブソース

extrn.o：trnmkobj コマンドで作成したトランザクション制御用オブジェクトファイル名

注※1

アーカイブライブラリを使用する場合は libmuto.a，共用ライブラリを使用する場合は libmuto.so または libmuto.sl とリンケージしてください。ただし，AIX 版ではアーカイブライブラリだけ提供します。

注※2

次の表に示す MCF 関数を使用する UAP の場合は，libmuto.a の代わりに libnpmuto.a とリンケージします。リンケージオプションは'-lmuto'の代わりに'-lnpmuto'を指定してください。

これらの関数を UAP で発行した場合，次のように動作します。

- C 言語の場合，リターン値には「0」を返し，OpenTP1 が引数に返す値のうち，数値データには「0」，文字データには「¥0」を返します。
- COBOL 言語の場合，ステータスコードには「00000」，OpenTP1 が引数に返す値のうち，数値データには「0」，文字データには「半角スペース」を返します。
- トレース情報の取得や MCF 関数のシミュレートは処理しません。

表 4-1　オンラインテスタが提供するシミュレート機能未サポートの MCF 関数一覧

関数名（C 言語）	関数名（COBOL 言語）
dc_mcf_adltap()	CBLDCMCF('ADLTAP ')
dc_mcf_ap_info()	CBLDCMCF('APINFO ')
dc_mcf_tactcn()	CBLDCMCF('TDCTCN ')
dc_mcf_tactle()	CBLDCMCF('TACTLE ')
dc_mcf_tdctcn()	CBLDCMCF('TDCTCN ')
dc_mcf_tdctle()	CBLDCMCF('TDCTLE ')
dc_mcf_tdlqle()	CBLDCMCF('TDLQLE ')
dc_mcf_timer_cancel()	CBLDCMCF('TIMERCAN ')
dc_mcf_timer_set()	CBLDCMCF('TIMERSET ')
dc_mcf_tlscn()	CBLDCMCF('TLSCN ')
dc_mcf_tlscom()	CBLDCMCF('TLSCOM ')
dc_mcf_tlsle()	CBLDCMCF('TLSLE ')
dc_mcf_tlsln()	CBLDCMCF('TLSLN ')
dc_mcf_tofln()	CBLDCMCF('TOFLN ')

関数名（C 言語）	関数名（COBOL 言語）
dc_mcf_tonln()	CBLDCMCF('TONLN ')

なお，上記以外の TP1/Message Control 03-03 より前にサポートされた MCF 関数については，「2.4　MCF のシミュレート」に示すように動作します。ただし，TP1/Message Control 03-03 以降に追加された機能には対応していません。

注※3

TP1/Message Control に関するコンパイルコマンドのオプションについては，TP1/Message Control のリリースノートを参照してください。

4.2 SPP へのサービス要求

ここでは，クライアント UAP のシミュレート時およびサーバ UAP のシミュレート時に，SPP にサービス要求する場合について説明します。

4.2.1 クライアント UAP のシミュレート

(1) RPC インタフェースのクライアント UAP のシミュレート

RPC インタフェースのクライアント UAP のシミュレートは，utosppsvc コマンドを入力して実行します。このコマンドの処理中に dc_rpc_call 関数を発行することで SPP にサービスを要求します。

(2) XATMI インタフェースのクライアント UAP のシミュレート

XATMI インタフェースのクライアント UAP のシミュレートは，utoxsppsvc コマンドを入力して実行します。このコマンドの処理中に次の関数を発行することで，SPP にサービスを要求します。

- リクエスト/レスポンス型の場合：tpcall 関数または tpacall 関数
- 会話型の場合：tpconnect 関数

4.2.2 サーバ UAP のシミュレート

(1) RPC インタフェースのサーバ UAP のシミュレート

RPC インタフェースのサーバ UAP のシミュレート時には，サービス要求する SPP をダミー SPP として起動します。シミュレートの対象となるダミー SPP は，ユーザサービス定義の test_mode オペランドに dmyspp を指定しておきます。

ダミー SPP は，OpenTP1 の dcsvstart コマンドで起動します。ダミー SPP へのサービス要求には，dc_rpc_call 関数を発行します。

ダミー SPP は，OpenTP1 の dcsvstop コマンド，または dcstop コマンドで停止します。

(2) XATMI インタフェースのサーバ UAP のシミュレート

XATMI インタフェースのサーバ UAP のシミュレート時には，サービス要求する SPP をダミー SPP として起動します。シミュレートの対象となるダミー SPP は，ユーザサービス定義の test_mode オペランドに dmyspp を指定しておきます。

ダミー SPP は，OpenTP1 の dcsvstart コマンドで起動し，dcsvstop コマンド，または dcstop コマンドで停止します。

サーバ UAP シミュレート機能を使用して，会話型でサービス要求した場合，クライアント UAP が，tprecv 関数でサービスの終了を示すイベントを受け取らないままプロセス，またはサービスを終了すると，テスタデーモン上に会話状態を管理するテーブルが残ったままになります。この場合は，ダミー SPP をいったん停止させたあと，再度起動してください。

4.3 MHP へのサービス要求

MCF のシミュレート時には，テストする MHP をシミュレート MHP として起動します。テストの対象となるシミュレート MHP は，ユーザサービス定義の test_mode オペランドに simmhp を指定しておきます。

シミュレート MHP は，OpenTP1 の dcsvstart コマンドまたはユーザサービス構成定義の dcsvstart 指定で起動します。

MHP へのサービス要求は，utomhpsvc コマンドで実行します。utomhpsvc コマンド以外でサービスを要求すると，オンラインテスタはエラーメッセージを出力して，要求したサービスの実行をスキップします。この時，いったんオンラインテスタでサービスを受け付けるため，dc_rpc_call 関数は正常終了しますが，サービスの応答データの内容は保証しません。

シミュレート MHP は，SPP として動作します。そのため，シミュレート MHP は SPP に対するコマンドを使用して運用する必要があります。ただし，utosppsvc コマンドは使用できません。

シミュレート MHP は，OpenTP1 の dcsvstop コマンド，または dcstop コマンドで停止します。

4.4　テスタファイルの作成

テスタファイルの作成は，utofilcre コマンドで実行します。

テスタファイルの作成では，テストデータ定義ファイルを使用する場合と，運用コマンドが出力するデータを使用する場合とで，作成の手順やコマンドの入力形式が異なります。

4.4.1　テストデータ定義ファイルを使用したテスタファイルの作成

テストデータ定義ファイルを使用してテスタファイルを作成する場合の手順を，次に示します。

（例）RPC 応答データファイルと運用コマンド結果データファイルを作成

1. テストデータ定義ファイルを，テキストエディタでオープンします。

```
'vi testenv_file'
```

2. RPC 応答データファイルと運用コマンド結果データファイルの入力データを設定します。

viエディタ：testenv_fileの内容

```
#RPC応答データファイルのデータ定義1
start test1 RRT /tmp/rpcrtnf01
data_len＝20
data＝'abcdefg'
    ＝0x0008
end
#RPC応答データファイルのデータ定義2
start test2 RRT /tmp/rpcrtnf02
data_len＝20
data＝'abcdefg'
    ＝0x0008
end
#運用コマンド結果データファイルのデータ定義
start test3 COM /tmp/comrtnf03
status-code＝-1
outsize＝20
errsize＝10
data＝'abcdefg'
data＝'abcdefg'
end
```

3. 記述内容を確認して，テストデータ定義ファイルをクローズします。
4. utofilcre コマンドにテストデータ定義ファイルを指定して実行します。

```
'utofilcre -e testenv_file'
```

4.4.2　運用コマンドの出力データを使用したテスタファイルの作成

運用コマンドの出力データを使用してテスタファイルを作成する場合の手順を，次に示します。

（例）RPC 要求データファイルを作成

1. RPC トレースデータファイルを編集出力して，テストデータに使用するトレースデータを決定します。

   （例：トレース番号　6 番のデータを使用）

```
'rpcdump rpctrc_file'
```

2. 使用する RPC トレースデータをトレースデータファイルの形式で出力して，ファイルに格納します。

```
'rpcdump -r -n6,6 rpctrc_file > testdata_file'
```

3. utofilcre コマンドに，作成するテスタファイル名，テスタファイル種別，および RPC トレースデータを格納したファイルを指定して実行します。

```
'utofilcre -o rpcreqfile -k RRQ -i testdata_file'
```

4.5　テスト情報の編集

4.5.1　テスト状況の表示

オンラインテスタを使用したテストの状況は，utols コマンドで表示できます。

表示できる情報を，次に示します。

- UAP のテストモード（ユーザサービス定義の test_mode オペランドの指定）
- UAP を起動したユーザのテストユーザ ID
- サーバ名
- サービスグループ名

表示内容については，「5.1　テストで使用する運用コマンド」を参照してください。

4.5.2　UAP トレース情報の取得

オンラインテスタで取得する UAP トレース情報は，OpenTP1 の UAP トレースのものと同じです。

ただし，次の条件を満たすとき，各 OpenTP1 関数の入り口でのトレース情報と，オンラインテスタ用のトレース情報（テスタ情報）を追加して取得します。

- ユーザサービス定義の test_mode オペランドに target，simmhp を指定した場合，または usable を指定し，テストモードで動作させた場合
- ユーザサービス定義の uap_trace_max オペランドに 1 以上を指定した（または指定を省略した）場合
- トレースを取得する UAP を OpenTP1 の dcsvstart コマンド，またはユーザサービス構成定義の dcsvstart 指定で起動した場合

オンラインテスタでは，トレース情報をグループにまとめて，次のタイミングでトレースファイルに出力します。テスタ情報は，トレース情報をトレースファイルに出力する単位で，1 回出力します。

- dc_rpc_mainloop 関数開始時
- dc_mcf_mainloop 関数開始時
- dc_rpc_call 関数開始時
- dc_rpc_close 関数終了時
- RPC サービス関数終了時
- tpcall 関数開始時
- tpacall 関数開始時

- tpconnect 関数開始時
- XATMI サービス関数終了時

一つのグループ内の情報で UAP トレース領域が満杯になった場合は，ラップアラウンドして再使用する前に，それまでの情報をトレースファイルに出力します。

トレースファイルは，OpenTP1 ごと，かつテストユーザ ID ごとに作成します。そのため，同一テストユーザ ID でトレース情報を出力する複数の UAP を並行して実行すると，複数 UAP のトレース情報が混ざり，UAP トレースファイルに不正な内容が出力されるおそれがあります。また，複数 UAP を並行して実行すると排他待ちが発生し，サービス要求がタイムアウトするおそれがあります。オンラインテスタ使用時には，同一テストユーザ ID でトレース情報を出力する UAP を並行して実行しないでください。

また，二つのトレースファイルのうち，一方の大きさがテスタサービス定義の max_trace_file_size オペランドに指定した値を超えた場合，スワップメッセージを出力します。以降の UAP トレースは，もう一方のトレースファイルに取得されます。両方のトレースファイルが満杯になると，トレース情報は取得できません。

そのため，スワップメッセージが出力された場合，ユーザは満杯になったトレースファイルを別のファイルにコピーし，満杯になったトレースファイルを削除する必要があります。その後，もう一方のトレースファイルも満杯になった時には，削除したファイル名でファイルを作成し，そのファイルに引き続きトレース情報を取得します。ただし，トレース情報取得中のファイルを削除すると，それ以降の情報が取得できなくなるので注意してください。

なお，dc_trn_info 関数のトレース情報は，取得されません。

4.5.3 UAP トレース情報のマージ・編集出力

UAP トレース情報のマージは，utotrcmrg コマンドで実行します。

utotrcmrg コマンドを入力すると，指定した複数のトレースファイル内のデータが，グループごとにサービス実行順に並べられ，指定したファイル（トレースマージファイル）に出力されます。出力先に指定したファイルがすでにある場合は，内容を削除したあとに書き込まれます。

トレースマージファイルは，ファイル種別が異なる以外はトレースファイルと同じファイル形式なので，トレースマージファイルもマージできます。また，トレース情報取得中のトレースファイルもマージできます。

UAP トレース情報の編集出力は，utotrcout コマンドで実行します。

utotrcout コマンドを入力すると，指定されたトレースファイル，またはトレースマージファイルの内容を編集して，標準出力に出力します。

utotrcout コマンドでは，次のような形式でトレース情報を出力できます。

- 特定のサービスのトレース情報だけを抜き出した形式

- 特定のサーバのトレース情報だけを抜き出した形式
- トレース情報の一部を抜き出した形式
- トレース情報をサービスの実行順に並べ替えた形式
- 指定した時間帯に取得したトレース情報だけを抜き出した形式

出力形式については，「5.1　テストで使用する運用コマンド」を参照してください。

4.5.4　MCF シミュレート関数の UAP トレース

MCF シミュレート関数でも，UAP トレース情報が取得されます。このとき，MCF アプリケーション定義など，オンラインテスタでは解析できない情報を必要とするトレース情報は取得できません。そのため，オンラインテスタはダミーの値を設定して対応します。

オンラインテスタが取得できない情報と，代わりに設定するダミーの値を次の表に示します。

表 4-2　オンラインテスタが取得できない情報とダミーの値

関数名	取得できない情報	設定するダミーの値
dc_mcf_mainloop 関数（開始時）	アプリケーション名	********
	入力元論理端末名	********
	アプリケーションの型	0
dc_mcf_mainloop 関数（リターン時）	アプリケーション名	********

なお，MCF シミュレート関数の dc_mcf_mainloop 関数は，dc_rpc_mainloop 関数を使用しています。そのため，dc_mcf_mainloop 関数発行時はトレース情報として dc_rpc_mainloop 関数の情報も出力されます。

4.5.5　送信メッセージの編集出力

MCF シミュレート機能使用時に取得した送信メッセージの編集出力は，utomsgout コマンドで実行します。utomsgout コマンドを入力すると，指定された MCF 送信メッセージファイル内のデータを，標準出力，または指定したファイルに編集して出力します。

utomsgout コマンドでは，次のような形式で送信メッセージを出力できます。

- 送信メッセージの一覧を簡略化した形式
- MCF 受信メッセージファイルの形式
- MCF 受信メッセージファイルの形式で出力しなかったメッセージだけを抜き出した形式

- 送信メッセージの中で，いちばん古いメッセージを抜き出した形式
- 送信メッセージの中で，いちばん新しいメッセージを抜き出した形式
- 特定の関数で取得したメッセージをだけ抜き出した形式
- 送信メッセージファイルの一部のメッセージだけを抜き出した形式
- 特定のサービスで送信したメッセージだけを抜き出した形式

出力形式については，「5.1　テストで使用する運用コマンド」を参照してください。

4.5.6　UAP の応答データの確認

クライアント UAP シミュレート機能で出力された RPC 応答データファイルと XATMI 応答データファイルは，ダンプ編集出力して内容を確認できます。

出力データの形式については，「3.3　ユーザが作成するファイル」を参照してください。

4.5.7　UAP の送信データの確認

サーバ UAP シミュレート機能で出力された XATMI 送信データファイルは，ダンプ編集出力して内容を確認できます。

出力データの形式については，「3.3　ユーザが作成するファイル」を参照してください。

なお，共用領域の内容は，次のとおりです。

項目		位置	長さ（バイト）	内容
共用領域	サービス名	0	32	送信先のサービス名が格納されます。
	コール記述子	32	4	送信時のコール記述子が格納されます。

5 運用コマンド

オンラインテスタの運用コマンドの使用方法について説明します。

5.1　テストで使用する運用コマンド

オンラインテスタの運用コマンドについて説明します。コマンドの形式，規則などについては，マニュアル「OpenTP1 運用と操作」を参照してください。

テストで使用する運用コマンドの一覧を，次の表に示します。

表 5-1　テストで使用する運用コマンドの一覧

コマンド名	機能
utodbgstop	デバッガと連動している UAP を停止します。
utodebug	UAP をデバッガと連動して起動します。
utofilcre	テスタファイルを作成します。
utofilout	テスタファイルの内容を編集出力します。
utols	テストの状況を表示します。
utomhpsvc	MHP にサービスを要求します。
utomsgout	送信メッセージを編集出力します。
utosppsvc	RPC インタフェースの SPP にサービスを要求します。
utotrcmrg	UAP トレース情報をマージします。
utotrcout	UAP トレース情報を編集出力します。
utoxsppsvc	XATMI インタフェースの SPP にサービスを要求します。

5.1.1　utodbgstop（デバッガと連動する UAP の停止）

(1)　名称

デバッガと連動する UAP の停止

(2)　形式

```
utodbgstop 〔-f〕 サーバ名
```

(3)　機能

デバッガと連動する UAP を停止します。

utodbgstop コマンドは OpenTP1 システムが稼働しているマシン上にある，utodebug コマンドを実行したウィンドウ以外のウィンドウで実行します。

なお，utodbgstop コマンドで UAP を停止させたあとは，すみやかにデバッガも停止させてください。デバッガを停止させるまで，utodbgstop コマンド，および utodebug コマンドは，応答待ち状態になります。

また，いったん utodbgstop コマンドで停止させた UAP は，デバッガコマンドで再度起動できません。

指定したサーバがデバッガと連動していない場合は，コマンドエラーとなります。

utodbgstop コマンドは，テスタサービスが起動中のときだけ使用できます。

(4) オプション

- -f
  指定したサーバを強制停止します。このオプションの指定を省略すると，該当するサーバを正常終了します。

(5) コマンド引数

- サーバ名　～〈1～8 文字の識別子〉
  停止させる，デバッガと連動する UAP のサーバ名を指定します。

(6) 注意事項

- コマンド入力後に，次に示すメッセージおよび要因コードが出力される場合がありますが，無視してください。
  メッセージ ID：KFCA01844-E
  要因コード：STATUS
  　　EXIT
  　　ABORTING
  　　ABORT
- コマンド入力後に，次に示すメッセージおよび要因コードが出力された場合は，デバッガを必ず停止させてください。
  メッセージ ID：KFCA01844-E
  要因コード：CRITICAL

5.1.2 utodebug（デバッガと連動する UAP の起動）

(1) 名称

デバッガと連動する UAP の起動

(2) 形式

```
utodebug サーバ名
```

(3) 機能

デバッガと連動する UAP を起動します。

utodebug コマンドを実行したウィンドウは，デバッガとの入出力に使用します。

utodebug コマンドを実行する場合，プロセスサービス定義の prcsvpath オペランドの指定，または prcpath コマンドで，サーチパス名に$DCDIR/bin，/usr/bin，および/bin を追加する必要があります。

utodebug コマンドは OpenTP1 システムが稼働しているマシン上のウィンドウで実行します。一つのウィンドウ上では，一つの UAP をデバッガと連動してテストできます。デバッガを停止させるまで，このウィンドウ上でそのほかのコマンドは実行できません。

指定したサーバのユーザサービス定義の test_mode オペランドに，target または simmhp 以外を指定した場合は，コマンドエラーとなります。また，指定したサーバがすでに起動している場合も，コマンドエラーとなります。

utodebug コマンドは，テスタサービスが起動中のときだけ使用できます。

(4) コマンド引数

- サーバ名　　～〈1～8 文字の識別子〉
  デバッガと連動してテストを実行する UAP のサーバ名を指定します。

(5) 注意事項

- デバッガと連動している UAP が正常終了，または異常終了した場合は，デバッガを停止させてください。
- デバッガ連動中にデバッガプロセスを強制的に終了させると，デバッガと連動していた UAP のプロセスが完全に終了しないで残ってしまう場合があります。この場合は，残っているプロセスをコマンドで終了させてください。
- デバッガ連動中に utodebug コマンドを強制的に終了させると，デバッガプロセスに対する入出力とシェルに対する入出力が混在するため，デバッガを制御できなくなります。この場合は，デバッガプロセスとデバッガと連動している UAP のプロセスを強制的に終了させてください。
- デバッガとの連動テスト中にデバッガが制御できなくなった場合は，utodebug コマンドプロセスとデバッガプロセス，およびデバッガと連動している UAP プロセスを強制的に終了させてください。その後，必要であれば再度，utodebug コマンドを実行してください。なお，デバッガプロセスと連動している UAP プロセスのプロセス ID は prcls コマンドで取得できます。

5.1.3　utofilcre（テスタファイルの作成）

(1)　名称

テスタファイルの作成

(2)　形式

```
utofilcre{-e テストデータ定義ファイル名
    |-o テスタファイル名 -k テスタファイル種別 〔-i 入力データファイル名〕}
```

(3)　機能

指定したテストデータ定義ファイル，または運用コマンドで取り出したアンロードジャーナルファイルのレコードデータや RPC トレースデータから，テスタファイルを作成します。

(4)　オプション

- -e　テストデータ定義ファイル名　　～〈パス名〉

  作成するテスタファイルの入力データを定義した，テストデータ定義ファイルの名称を指定します。

  -o，-k，および-i オプションとは同時に指定できません。

- -o　テスタファイル名　　～〈パス名〉

  運用コマンドで取り出したデータから作成する，テスタファイルの名称を指定します。このオプションを指定する場合，-k オプションを必ず指定してください。

  -e オプションとは同時に指定できません。

- -k　テスタファイル種別

  運用コマンドで取り出したデータから作成する，テスタファイルの種別を指定します。指定できるテスタファイル種別を次に示します。

  RRQ：RPC 要求データファイル

  RRT：RPC 応答データファイル

  XRQ：XATMI 要求データファイル

  XRT：XATMI 応答データファイル

  XRV：XATMI 受信データファイル

  NRV：非同期型受信メッセージファイル

  SRV：同期型受信メッセージファイル

  なお，運用コマンド結果データファイルは，コマンドから取り出したデータからファイルを作成できません。そのため，このオプションに運用コマンド結果データファイルは指定できません。

  このオプションを指定する場合，-o オプションを必ず指定してください。

  -e オプションとは同時に指定できません。

- -i　入力データファイル名　　～〈パス名〉

  運用コマンドで取り出したデータを格納した，入力データファイルの名称を指定します。このオプションを指定する場合，-o オプションを必ず指定してください。

  -o オプション指定時に，このオプションの指定を省略すると，標準入力からの入力と見なします。

  -e オプションとは同時に指定できません。

(5)　注意事項

- -o オプションを指定した場合に-i オプションの指定を省略すると，標準入力からの入力となります。そのため，入力ファイルをパイプ，リダイレクションなどで指定してください。なお，入力ファイルが指定されない場合，コマンドは入力待ち状態になります。この場合は，コマンドを強制的に終了させてください。
- アンロードジャーナルファイルの mj レコードデータには，マップ名が取得されていません。このため，-o オプションを指定して，非同期型受信メッセージファイル，または同期型受信メッセージファイルを作成する場合，入力データに mj レコードデータを指定すると，マップ名に次の仮定値を設定します。

  マップ名の仮定値：UTOMAP

5.1.4　utofilout（テスタファイルの編集出力）

(1)　名称

テスタファイルの編集出力

(2)　形式

```
utofilout -k テスタファイル種別 テスタファイル名
```

(3)　機能

指定したテスタファイルの内容を，指定したテスタファイル種別のデータ形式で編集して，標準出力に出力します。

テスタファイル種別には，編集出力するテスタファイルの種別を指定します。異なるテスタファイル種別を指定すると，指定したテスタファイル種別のデータ形式で編集し，編集できるデータであれば編集結果を出力します。編集できないデータの場合は，コマンドエラーとなります。

(4)　オプション

- -k　テスタファイル種別

  編集出力するテスタファイルの種別を指定します。指定できるテスタファイル種別を次に示します。

RRQ：RPC 要求データファイル

RRT：RPC 応答データファイル

XRQ：XATMI 要求データファイル

XRT：XATMI 応答データファイル

XRV：XATMI 受信データファイル，および XATMI 送信データファイル

NRV：MCF 受信メッセージファイル

（非同期型受信メッセージファイル，および同期型受信メッセージファイル）

COM：運用コマンド結果データファイル

(5) コマンド引数

- テスタファイル名　　～〈パス名〉

  編集するテスタファイルの名称を指定します。

(6) 出力形式（-k オプションに RRQ を指定した場合）

```
ファイル種別 = RPC request data file （RRQ）                                    ]1.
ファイル名称 = /tmp/rrqfile                                                     ]
--------------------------------------------------------------------------------
No.1                                                                            ]2.
  応答領域長 = 256                                                              ]3.
  データ長 = 260                                                                ]
  データ内容                                                                    ]
    00000000   52504320 72657175 65737420 64617461  RPC request data            ]
    00000010   00000000 00000000 00000000 00000000  ................            ]4.
    00000020 - 000000ff :      SAME DATA                                        ]
    00000100   00000000                                                         ]
     └──────┘  └──────────────────────────────────┘ └──────────────┘
       5.                      6.                       7.
```

（凡例）

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ

   同じデータの場合には，次のように表示します。

   （最初に一致したデータのロケーション）

   －（最後に一致したデータのロケーション）：SAME DATA
5. ユーザデータのロケーション
6. ユーザデータの 16 進数表示部
7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

RPC 要求データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

応答領域長

RPC 要求データファイルのヘッダに指定した応答領域長（10 進数　単位：バイト）

データ長

RPC 要求データファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに RRQ を指定した場合の出力例

```
ファイル種別 = RPC request data file  (RRQ)
ファイル名称 = /tmp/rrqfile
------------------------------------------------------------------------
No.1
  応答領域長 = 256
  データ長 = 20
  データ内容
    00000000   52504320 72657175 65737420 64617461  RPC request data
    00000010   00000000                              ....
```

(7)　出力形式（-k オプションに RRT を指定した場合）

```
ファイル種別 = RPC response data file  (RRT)                          ]1.
ファイル名称 = /tmp/rrtfile                                           ]
------------------------------------------------------------------------
No.1                                                                  ]2.
  データ長 = 260                                                      ]3.
  データ内容                                                          ]
    00000000   52504320 72657370 6f6e7365 20646174  RPC response dat  ]
    00000010   61000000 00000000 00000000 00000000  a...............  ]
    00000020   00000000 00000000 00000000 00000000  ................  ]4.
    00000020 - 000000ff :      SAME DATA                              ]
    00000100    00000000                                              ]
     └──────┘   └──────────────────────────────┘ └──────────────┘
       5.                    6.                       7.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ

   同じデータの場合には，次のように表示します。

（最初に一致したデータのロケーション）

－（最後に一致したデータのロケーション）：SAME DATA

5. ユーザデータのロケーション

6. ユーザデータの 16 進数表示部

7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

RPC 応答データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

データ長

RPC 応答データファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに RRT を指定した場合の出力例

```
ファイル種別 = RPC response data file  (RRT)
ファイル名称 = /tmp/rrtfile
--------------------------------------------------------------------------------
No.1
  データ長 = 20
  データ内容
    00000000   52504320 72657370 6f6e7365 20646174  RPC response dat
    00000100   61000000                              a...
```

(8)　出力形式（-k オプションに XRQ を指定した場合）

```
ファイル種別 = XATMI request data file  (XRQ)                          ┐1.
ファイル名称 = /tmp/xrqfile                                            ┘
--------------------------------------------------------------------------------
No.1                                                                   ┐2.
  呼出し種別 = call                                                    ┐
  フラグ = 0x0000108  (TPNOTRAN)  (TPNOCHANGE)                         │
  タイプ = X_OCTET                                                     │3.
  サブタイプ = ****                                                    │
  データ長 = 260                                                       ┘
  データ内容                                                           ┐
    00000000   74706361 6c6c2058 5f4f4354 45542064  tpcall X_OCTET d   │
    00000010   61746100 00000000 00000000 00000000  ata.............   │
    00000020   00000000 00000000 00000000 00000000  ................   │4.
    00000030 - 000000ff :      SAME DATA                               │
    00000100   00000000                                                ┘
      └──┘  └──────────────────────────────────┘ └──────────────┘
       5.                  6.                        7.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ

   同じデータの場合には，次のように表示します。

   (最初に一致したデータのロケーション)

   －(最後に一致したデータのロケーション)：SAME DATA
5. ユーザデータのロケーション
6. ユーザデータの 16 進数表示部
7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

XATMI 要求データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

呼出し種別

XATMI 要求データファイルのヘッダに指定した呼出し種別（最大 7 文字）

文字列が設定されていない場合は'****'を表示します。

フラグ

XATMI 要求データファイルのヘッダに指定したフラグ（8 けた）

タイプ

XATMI 要求データファイルのヘッダに指定したバッファタイプ（最大 8 文字）

文字列が設定されていない場合は'****'を表示します。

サブタイプ

XATMI 要求データファイルのヘッダに指定したバッファサブタイプ（最大 16 文字）

文字列が設定されていない場合は'****'を表示します。

データ長

XATMI 要求データファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに XRQ を指定した場合の出力例

```
ファイル種別 = XATMI request data file （XRQ）
ファイル名称 = /tmp/xrqfile
--------------------------------------------------------------------------------
No.1
  呼出し種別 = call
  フラグ = 0x0000108 （TPNOTRAN） （TPNOCHANGE）
  タイプ = X_OCTET
  サブタイプ = ****
  データ長 = 20
  データ内容
    00000000   74706361 6c6c2058 5f4f4354 45542064  tpcall X_OCTET d
    00000010   61746100                             ata.
```

(9) 出力形式（-k オプションに XRT を指定した場合）

```
ファイル種別 = XATMI response data file （XRT）                          ┐1.
ファイル名称 = /tmp/xrtfile                                              ┘
--------------------------------------------------------------------------------
No.1                                                                     ┐2.
  タイプ = X_OCTET                                                       ┐
  サブタイプ = ****                                                      │
  終了コード = 0x04000000 （TPFAIL）                                     │3.
  リターンコード = 22                                                    │
  データ長 = 260                                                         ┘
  データ内容                                                             ┐
    00000000   74707265 7475726e 20545046 41494c20  tpreturn TPFAIL      │
    00000010   64617461 00000000 00000000 00000000  data............      │
    00000020   00000000 00000000 00000000 00000000  ................      │4.
    00000030 - 000000ff :      SAME DATA                                   │
    00000100   00000000                                                    ┘
      └──┘   └──────────────────────────────────┘ └──────────────┘
       5.                  6.                        7.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ

   同じデータの場合には，次のように表示します。

   (最初に一致したデータのロケーション)

   －（最後に一致したデータのロケーション)：SAME DATA
5. ユーザデータのロケーション
6. ユーザデータの 16 進数表示部
7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

XATMI 応答データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

タイプ

XATMI 応答データファイルのヘッダに指定したバッファタイプ（最大 8 文字）

文字列が設定されていない場合は'****'を表示します。

サブタイプ

XATMI 応答データファイルのヘッダに指定したバッファサブタイプ（最大 16 文字）

文字列が設定されていない場合は'****'を表示します。

終了コード

XATMI 応答データファイルのヘッダに指定したサービス終了コード（8 けた）

リターンコード

XATMI 応答データファイルのヘッダに指定したリターンコード（10 進数　最大 11 けた）

データ長

XATMI 応答データファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに XRT を指定した場合の出力例

```
ファイル種別 = XATMI response data file （XRT）
ファイル名称 = /tmp/xrtfile
------------------------------------------------------------------------
No.1
  タイプ = X_OCTET
  サブタイプ = ****
  終了コード = 0x04000000 （TPFAIL）
  リターンコード = 22
  データ長 = 20
  データ内容
    00000000   74707265 7475726e 20545046 41494c20  tpreturn TPFAIL
    00000010   64617461                            data
```

(10)　出力形式（-k オプションに XRV を指定した場合）

```
ファイル種別 = XATMI receive/send data file （XRV）                    ┐1.
ファイル名称 = /tmp/xrvfile                                            ┘
------------------------------------------------------------------------
No.1                                                                   ┐2.
  タイプ = X_OCTET                                                     │
  サブタイプ = ****                                                    │3.
  イベントフラグ = 0x00000008 （TPEV_SVCSUCC）                         │
  データ長 = 260                                                       ┘
  データ内容                                                           ┐
    00000000   74707265 63762072 65637620 64617461  tprecv recv data  │
    00000010   00000000 00000000 00000000 00000000  ................  │4.
    00000020 - 000000ff :      SAME DATA                               │
    00000100   00000000                                                ┘
     └──┘   └──────────────────────────────────┘ └──────────┘
      5.                    6.                      7.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ

   同じデータの場合には，次のように表示します。

   (最初に一致したデータのロケーション)

   －（最後に一致したデータのロケーション)：SAME DATA
5. ユーザデータのロケーション
6. ユーザデータの 16 進数表示部
7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

XATMI 受信データファイル，および XATMI 送信データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

タイプ

XATMI 受信データファイル，および XATMI 送信データファイルのヘッダに指定したバッファタイプ（最大 8 文字）

文字列が設定されていない場合は'****'を表示します。

サブタイプ

XATMI 受信データファイル，および XATMI 送信データファイルのヘッダに指定したバッファサブタイプ（最大 16 文字）

文字列が設定されていない場合は'****'を表示します。

イベントフラグ

XATMI 受信データファイル，および XATMI 送信データファイルのヘッダに指定したイベントフラグ（8 けた）

データ長

XATMI 受信データファイル，および XATMI 応答データファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに XRV を指定した場合の出力例

```
ファイル種別 = XATMI receive/send data file （XRV）
ファイル名称 = /tmp/xrvfile
------------------------------------------------------------------------
No.1
  タイプ = X_OCTET
  サブタイプ = ****
  イベントフラグ = 0x00000008 （TPEV_SVCSUCC）
  データ長 = 20
  データ内容
    00000000   74707265 63762072 65637620 64617461  tprecv recv data
    00000010   00000000                             ....
```

(11) 出力形式（-k オプションに NRV を指定した場合）

```
ファイル種別 = MCF receive message file （NRV）                      ]1.
ファイル名称 = /tmp/nrvfile
------------------------------------------------------------------------
No.1                                                                  ]2.
  論理端末名称 = TERM01                                               ]
  マップ名称 = MAP01                                                  ]3.
  セグメント種別 = 0                                                  ]
  データ長 = 260                                                      ]
  データ内容
    00000000   4d434620 72656376 206d6573 73616765  MCF recv message  ]
    00000010   00000000 00000000 00000000 00000000  ................  ]4.
    00000020 - 000000ff :     SAME DATA                               ]
    00000100   00000000                                               ]
    └──┘   └──────────────────┘  └──────┘
       5.                   6.                       7.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. ユーザデータ
   同じデータの場合には，次のように表示します。
   (最初に一致したデータのロケーション)
   − （最後に一致したデータのロケーション）：SAME DATA
5. ユーザデータのロケーション
6. ユーザデータの 16 進数表示部
7. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

MCF 受信メッセージファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

論理端末名称

MCF 受信メッセージファイルのヘッダに指定した論理端末名称（最大 8 文字）

文字列が設定されていない場合は'****'を表示します。

マップ名称

MCF 受信メッセージファイルのヘッダに指定したマップ名称（最大 8 文字）

文字列が設定されていない場合は'****'を表示します。

セグメント種別

MCF 受信メッセージファイルのヘッダに指定したセグメント種別（1 文字）

文字が設定されていない場合は'****'を表示します。

データ長

MCF 受信メッセージファイルのヘッダに指定したデータ長（10 進数　単位：バイト）

- -k オプションに NRV を指定した場合の出力例

```
ファイル種別 = MCF receive message file  (NRV)
ファイル名称 = /tmp/nrvfile
------------------------------------------------------------------------
No.1
  論理端末名称 = TERM01
  マップ名称 = MAP01
  セグメント種別 = L
  データ長 = 20
  データ内容
    00000000   4d434620 72656376 206d6573 73616765  MCF recv message
    00000010   32000000                              2...
No.2
  論理端末名称 = TERM02
  マップ名称 = MAP02
  セグメント種別 = 0
  データ長 = 20
  データ内容
    00000000   4d434620 72656376 206d6573 73616765  MCF recv message
    00000010   33000000                              3...
```

(12) 出力形式（-k オプションに COM を指定した場合）

```
ファイル種別 = operation command result data file （COM）                    ]1.
ファイル名称 = /tmp/comfile                                                  ]
------------------------------------------------------------------------
No.1                                                                        ]2.
  コマンド結果コード = 1                                                     ]
  標準出力データ長 = 260                                                     ]3.
  標準エラー出力データ長 = 260                                               ]
  標準出力データ内容                                                         ]
    00000000   7374646f 75742064 61746120 00000000  stdout data ....        ]
    00000010   00000000 00000000 00000000 00000000  ................        ]4.
    00000020 - 000000ff  :      SAME DATA                                   ]
    00000100   00000000                                                     ]
  標準エラー出力データ内容                                                   ]
    00000000   73746465 72722064 61746120 00000000  stderr data ....       ]
    00000010   00000000 00000000 00000000 00000000  ................        ]5.
    00000020 - 000000ff :      SAME DATA                                    ]
    00000100   00000000                                                     ]
    └──────┘   └──────────────────────────────────┘ └──────────────┘
      6.                      7.                        8.
```

(凡例)

1. ファイル情報
2. データ番号
3. 固有情報データ
4. 標準出力データ（ユーザデータ）

   同じデータの場合には，次のように表示します。

   (最初に一致したデータのロケーション)

   －（最後に一致したデータのロケーション）：SAME DATA
5. 標準エラー出力データ（ユーザデータ）

   同じデータの場合には，次のように表示します。

   (最初に一致したデータのロケーション)

   －（最後に一致したデータのロケーション）：SAME DATA
6. ユーザデータのロケーション
7. ユーザデータの 16 進数表示部
8. ユーザデータの ASCII 文字表示部

(説明)

ファイル種別

運用コマンド結果データファイルのテスタファイル種別の表示

ファイル名称

指定したテスタファイルパス名（最大 64 文字）

データ番号

ファイルの先頭から付けた一連のデータ番号（最大 10 けた）

コマンド結果コード

運用コマンド結果データファイルのヘッダに指定したコマンド結果コード（10 進数　最大 11 けた）

標準出力データ長

運用コマンド結果データファイルのヘッダに指定した標準出力文字列のデータ長（10 進数　単位：バイト）

標準エラー出力データ長

運用コマンド結果データファイルのヘッダに指定した標準エラー出力文字列のデータ長（10 進数　単位：バイト）

- -k オプションに COM を指定した場合の出力例

```
ファイル種別 = operation command result data file （COM）
ファイル名称 = /tmp/comfile
------------------------------------------------------------------------
No.1
  コマンド結果コード = 1
  標準出力データ長 = 260
  標準エラー出力データ長 = 260
  標準出力データ内容
    00000000   7374646f 75742064 61746120 00000000  stdout data ....
    00000010   00000000                              ....
  標準エラー出力データ内容
    00000000   73746465 72722064 61746120 00000000  stderr data ....
    00000010   00000000                              ....
```

5.1.5　utols（テスト状況の表示）

(1)　名称

テスト状況の表示

(2)　形式

```
utols 〔サーバ名〔サーバ名〕…〕
```

(3)　機能

テスタサービスが管理しているテスト UAP の状態を標準出力に出力します。ただし，出力対象がない場合は，何も出力しません。

utols コマンドは，テスタサービスが起動中のときだけ使用できます。

(4)　コマンド引数

- サーバ名　　～〈1～8 文字の識別子〉

状態を表示させるユーザサーバの名称を指定します。

このコマンド引数の指定を省略すると，すべてのテスト対象のユーザサーバの状態を表示します。

(5) 出力形式

```
モード ID サーバ名 サービスグループ名                          デバッガ
-------------------------------------------------------------------
target usr1 spp01   group1                                  dbx
dmyspp usr1 spp02   group2
simmhp usr2 spp03   group3                                  dbx
usable **** spp03   group3
└─────┘ └──┘ └────┘ └──────┘                                 └──┘
  1.     2.    3.       4.                                    5.
```

1. 該当する UAP のテストモード情報（ユーザサービス定義の test_mode オペランドの指定）を次のどれかで表示します。

   target：test_mode=target で起動

   usable：test_mode=usable で起動

   dmyspp：test_mode=dmyspp で起動

   simmhp：test_mode=simmhp で起動

2. UAP を起動したユーザのテストユーザ ID

   UAP のテストモードが usable の場合は，'****'を表示します。

3. サーバ名（8 文字以内）

4. サービスグループ名（31 文字以内）

   サービスグループの指定がない場合は，何も表示しません。

5. UAP と連動しているデバッガ名（8 文字以内）

   デバッガと連動していない場合は，何も表示しません。

(6) 注意事項

テスト対象の UAP が起動中，または停止中に，その UAP を強制停止させた場合や OpenTP1 システムが即時停止した場合，起動していない UAP や停止済みの UAP についての情報を表示することがあります。この場合，誤って表示された UAP を再度起動すると，正しい情報が表示されます。

5.1.6 utomhpsvc（MHP へのサービス要求）

(1) 名称

MHP へのサービス要求

(2) 形式

```
utomhpsvc 〔-t MCF受信メッセージヘッダファイル名〕
          〔-n〕サービスグループ名 サービス名 MCF受信メッセージファイル名
```

(3) 機能

MCF シミュレート機能使用時，MHP に対して特定のサービスの実行を要求します。サービス要求先の MHP は，オンラインテスタが提供する MCF シミュレート関数ライブラリと結合させた，シミュレート MHP として起動している必要があります。

テストモードで起動していない MHP にサービス要求した場合は，コマンドエラーとなります。また，テストモードで動作する SPP にサービス要求した場合，その動作は保証しません。

サービス要求後，システム共通定義の RPC 最大応答待ち時間（watch_time オペランドの指定値）を超えて応答がない場合は，送受信タイムアウトになり，コマンドを受け付けません。

(4) オプション

- -t　MCF 受信メッセージヘッダファイル名　　～〈1～14 文字の英数字〉
  受信メッセージに付けるヘッダセグメントを格納した，MCF 受信メッセージファイルの名称を指定します。
  このオプションの指定を省略すると，受信メッセージにヘッダセグメントを付けません。
- -n
  指定したサービスを，非トランザクション MHP として実行します。このオプションの指定を省略すると，サービスはトランザクション MHP として実行します。

(5) コマンド引数

- サービスグループ名　　～〈1～31 文字の識別子〉
  実行するサービスのあるサービスグループの名称を指定します。
- サービス名　　～〈1～31 文字の識別子〉
  実行するサービスの名称を指定します。
- MCF 受信メッセージファイル名　　～〈1～14 文字の英数字〉
  受信メッセージを格納した，MCF 受信メッセージファイルの名称を指定します。

5.1.7 utomsgout（送信メッセージの編集出力）

(1) 名称

送信メッセージの編集出力

(2) 形式

```
utomsgout〔{-i|-r 出力先ファイル名}〕〔-w〕〔{-o|-l}〕
          〔-f 関数名〕〔-n 番号〕〔-t 論理端末名称〕
          〔-s サービスグループ名〔,サービス名〕…〕
          MCF送信メッセージファイル名
```

(3) 機能

オンラインテスタが出力した送信メッセージ情報を編集して，標準出力に出力します。ただし，-r オプションを指定した場合は，標準出力には出力しないで，指定したファイルに出力します。

指定した MCF 送信メッセージファイルに OpenTP1 が送信メッセージを出力中の場合，コマンドエラーとなります。

各オプションは，次のように分類されます。

- 出力形式を変更するためのオプション ： -i，-r
- 出力対象を限定するためのオプション ： -f，-l，-n，-o，-s，-t，-w

フラグ引数を持つオプションを重複して指定すると，あとに指定したオプションが有効になります。

(4) オプション

- -i

  送信メッセージの一覧を簡略した形式で表示します。

  -r オプションとは同時に指定できません。

- -r　出力先ファイル名　　～〈パス名〉

  指定したメッセージを出力するファイルの名称を指定します。指定したメッセージは，MCF 受信メッセージファイルのデータ形式で出力されます。出力先のファイルは，そのまま MCF 受信メッセージファイルとして使用できます。

  -i オプションとは同時に指定できません。

  -i，および-r オプションの指定を両方とも省略すると，セグメント情報と送信メッセージ情報を標準出力に出力します。

- -w

  -r オプションで出力していないメッセージだけを編集して出力します。

  このオプションの指定を省略すると，すべてのメッセージを編集して出力します。

- -o

  編集の対象となるメッセージの中で，いちばん古いメッセージを一つ出力する場合に指定します。

  -l オプションとは同時に指定できません。

  このオプションと-l オプションの指定を両方とも省略すると，すべてのメッセージを出力します。

- -l

  編集の対象となるメッセージの中で，いちばん新しいメッセージを一つ出力する場合に指定します。

  -o オプションとは同時に指定できません。

  このオプションと-o オプションの指定を両方とも省略すると，すべてのメッセージを出力します。

- -f　関数名

  特定の関数で取得したメッセージを出力する場合に指定します。指定できる関数名を次に示します。

  send：dc_mcf_send 関数

  reply：dc_mcf_reply 関数

  execap：dc_mcf_execap 関数

  sendrecv：dc_mcf_sendrecv 関数

  sendsync：dc_mcf_sendsync 関数

  なお，MCF シミュレート機能では，dc_mcf_resend 関数が発行されても送信メッセージは再送（再出力）されません。そのため，このオプションに dc_mcf_resend 関数は指定できません。

- -n　番号

  出力するメッセージを番号で選択する場合に指定します。各メッセージの番号は，-i オプションで一覧表示して確認してください。

  -n オプションと，-i，-r オプション以外のオプションを同時に指定すると，-n オプションの指定が優先されます。

- -t　論理端末名称　　～〈1～8 文字の識別子〉

  送信先の論理端末名称をキーにして出力する場合，その論理端末名称を指定します。

- -s　サービスグループ名　　～〈1～31 文字の識別子〉

  サービス名　　～〈1～31 文字の識別子〉

  特定のサービスで送信したメッセージ情報を出力する場合に，サービスグループ名とサービス名を一緒に指定します。このとき，サービスグループ名とサービス名の間を','（コンマ）で区切ります。

  サービスグループごとに，複数のサービスを指定できます。複数のサービスを指定する場合，サービス名とサービス名の間を','で区切ります。','の前後に空白などは入れないでください。

  サービス名だけでは指定できません。

  サービス名の指定を省略すると，指定したサービスグループにあるすべてのサービスの送信メッセージ情報を編集して出力します。

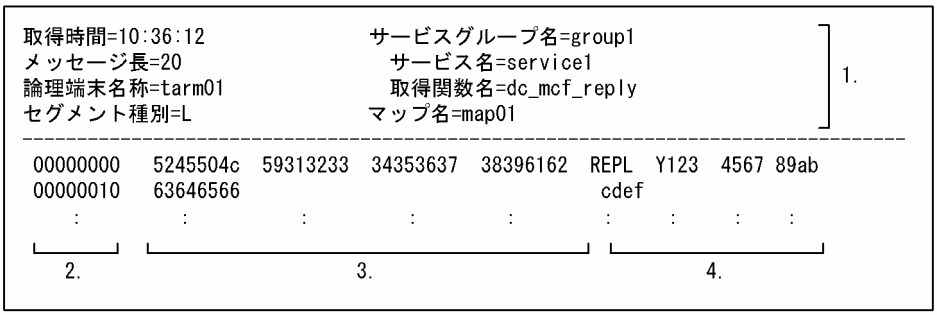
  このオプションの指定を省略すると，すべてのサービスグループにあるすべてのサービスで送信したメッセージ情報を出力します。

(5)　コマンド引数

- MCF 送信メッセージファイル名　　～〈パス名〉

  送信メッセージを格納した，MCF 送信メッセージファイルの名称を指定します。

(6) 出力形式（-i，および-r オプションの指定を省略した場合）



1. 編集出力した送信メッセージの情報
   - 送信メッセージの取得時間（時：分：秒）
   - メッセージの長さ（10 けた以内）
   - 論理端末名称（8 文字以内）
   - メッセージを送信したサービスグループ名（31 文字以内）
     サービスグループ名が不明の場合は，'****'を表示します。
   - メッセージを送信したサービス名（31 文字以内）
     サービス名が不明の場合は，'****'を表示します。
   - 次の取得関数名のどれか
     dc_mcf_send 関数
     dc_mcf_reply 関数
     dc_mcf_execap 関数
     dc_mcf_sendrecv 関数
     dc_mcf_sendsync 関数
   - 次のセグメント種別のどれか
     M：中間セグメント
     L：最終セグメント
   - マップ名
     マップ名がない場合は，'****'を表示します。
2. 相対ロケーション
3. ダンプ表示（16 進数字）
4. アスキー文字表示
   文字表示できない場合は，'.'（ピリオド）で表示します。

- -i，および-r オプションの指定を省略した場合の出力例

```
取得時間=10：36：12          サービスグループ名=group1
メッセージ長=20                サービス名=service1
論理端末名称=tarm01            取得関数名=dc_mcf_reply
セグメント種別=L             マップ名=map01
------------------------------------------------------------------------
 00000000 5245504c 59313233 34353637 38396162 REPL Y123 4567 89ab
 00000010 63646566                               cdef

取得時間=10：36：13          サービスグループ名=group2
メッセージ長=10                サービス名=service2
論理端末名称=tarm01            取得関数名=dc_mcf_send
セグメント種別=L             マップ名=****
------------------------------------------------------------------------
 00000000  53454e44 30303030 3030                send 0000 00
                                        :
                                        :
```

(7) 出力形式（-i オプションを指定した場合）

```
 番号  関数名     サービスグループ名                          サービス名
-------------------------------------------------------------------------
   1   reply     group1                                    service1
   2   send      group2                                    service2
   :     :         :                                         :
  └┘  └──┘     └────┘                                    └──────┘
  1.    2.        3.                                        4.
```

1. ファイル内のメッセージ番号
2. メッセージを送信した関数名

   send：dc_mcf_send 関数

   reply：dc_mcf_reply 関数

   execap：dc_mcf_execap 関数

   sendrecv：dc_mcf_sendrecv 関数

   sendsync：dc_mcf_sendsync 関数
3. メッセージを送信したサービスグループ名（31 文字以内）

   サービスグループ名が不明の場合は，'****'を表示します。
4. メッセージを送信したサービス名（31 文字以内）

   サービス名が不明の場合は，'****'を表示します。

- **-i オプションを指定した場合の出力例**

```
    番号 関数名      サービスグループ名              サービス名
------------------------------------------------------------------------
       1 reply      group1                      service1
       2 send       group2                      service2
       3 execap     group2                      ****
       4 send       ****                        service3
                                    :
                                    :
```

(8) 注意事項

- MCF シミュレート関数で取得する送信メッセージは，関数発行時点で MCF 送信メッセージファイルに書き込まれます。ロールバックが発生しても，MCF 送信メッセージファイル内の送信メッセージは残ります。
- utomsgout コマンドの-r オプションで出力したメッセージのセグメント種別は，［M…］L となります。例えば，3 セグメントで構成される論理メッセージを出力した場合 F，M，L となるセグメント種別が，実際は M，M，L と出力されます。M，M，L と出力されても，このままオンラインテスタ，またはオフラインテスタの入力に使用できます。

5.1.8 utosppsvc（RPC インタフェースの SPP へのサービス要求）

(1) 名称

RPC インタフェースの SPP へのサービス要求

(2) 形式

```
utosppsvc サービスグループ名 サービス名 RPC要求データファイル名 〔RPC応答データファイル名〕
```

(3) 機能

RPC インタフェースの SPP に対して，特定のサービスの実行を要求します。ただし，トランザクショナル RPC を期待する SPP（サービス要求元であらかじめトランザクションを生成していることを前提とする SPP）に対して，サービスの実行を要求できません。また，RPC インタフェースの SPP 以外の UAP に対して utosppsvc コマンドを実行すると，コマンドエラーとなります。

サービス要求後，システム共通定義の RPC 最大応答待ち時間（watch_time オペランドの指定値）を超えて応答がない場合は，送受信タイムアウトになり，コマンドを受け付けません。

また，このコマンドは，シミュレート MHP に対しては使用できません。

(4) コマンド引数

- **サービスグループ名　～〈1～31 文字の識別子〉**
  実行するサービスのあるサービスグループの名称を指定します。
- **サービス名　～〈1～31 文字の識別子〉**
  実行するサービスの名称を指定します。
- **RPC 要求データファイル名　～〈パス名〉**
  サービス要求時の入力データを格納した，RPC 要求データファイルの名称を指定します。

- RPC 応答データファイル名　　～〈パス名〉
  サービス実行後の応答データを格納する，RPC 応答データファイルの名称を指定します。
  このコマンド引数の指定を省略すると，応答データは破棄されます。
  なお，出力先のファイルがすでにある場合，そのファイルに上書きして出力します。ファイルがない場合は，オンラインテスタがファイルを作成して出力します。

5.1.9　utotrcmrg（UAP トレース情報のマージ）

(1)　名称

UAP トレース情報のマージ

(2)　形式

```
utotrcmrg -o トレースマージファイル名 トレースファイル名 トレースファイル名〔トレースファイル名 …〕
```

(3)　機能

指定した複数のトレースファイルのトレース情報を，サービスの実行順に並べ，指定したファイルに出力します。

まったく同じトレース情報が複数ある場合，一つだけ出力します。

異なるバージョンのオンラインテスタで取得したトレースファイルをマージした場合，マージ後のトレース情報の順番は実際の取得順序と異なることがあります。

(4)　オプション

- -o　トレースマージファイル名　　～〈パス名〉
  マージしたトレース情報を出力する，トレースマージファイルの名称を指定します。

(5)　コマンド引数

- トレースファイル名　　～〈パス名〉
  マージするトレースファイル，またはトレースマージファイルの名称を指定します。

(6)　注意事項

- 指定したトレースファイルにネスト管理できないバージョンのトレース情報がある場合，警告メッセージが出力され，時系列にマージされます。
- ネスト管理に必要なトレース情報がない場合，警告メッセージが出力されます。

5.1.10　utotrcout（UAP トレース情報の編集出力）

(1)　名称

UAP トレース情報の編集出力

(2)　形式

```
utotrcout 〔-s サービスグループ名〔,サービス名〕…〕
          〔-v サーバ名〕〔-i〕〔-n〕
          〔-t 〔編集開始日時〕〔,編集終了日時〕〕編集対象ファイル名
```

(3)　機能

指定したトレースファイル，またはトレースマージファイルのトレース情報を編集して標準出力に出力します。

(4)　オプション

- -s　サービスグループ名　　～〈1～31 文字の識別子〉

  サービス名　　～〈1～31 文字の識別子〉

  特定のサービスのトレース情報を編集出力する場合に，そのサービスを含むサービスグループの名称とサービスの名称を一緒に指定します。このとき，サービスグループ名とサービス名の間を','（コンマ）で区切ります。

  サービスグループごとに，複数のサービスを指定できます。複数のサービスを指定する場合，サービス名とサービス名の間を','で区切ります。','の前後に空白などを入れないでください。また，サービス名だけでは指定できません。

  サービス名の指定を省略すると，指定したサービスグループにあるすべてのサービスのトレース情報を編集出力します。

  このオプションの指定を省略すると，指定したファイル内にあるすべてのサービスグループのトレース情報を編集出力します。

  -v オプションと同時に指定すると，両方の条件を満たすトレース情報を編集出力します。

  -n オプションと同時に指定すると，指定したサービスの要求先のトレース情報も編集出力します。

- -v　サーバ名　　～〈1～8 文字の識別子〉

  特定のサーバのトレース情報を編集出力する場合に指定します。

  このオプションの指定を省略すると，指定したファイル内にあるすべてのサーバのトレース情報を編集出力します。

  -s オプションと同時に指定すると，両方の条件を満たすトレース情報を編集出力します。

  -n オプションと同時に指定すると，指定したサーバの要求先のトレース情報も編集出力します。

- -i

編集対象のファイル中に取得されているトレース情報の関数名など，一部の情報を標準出力に出力します。

このオプションの指定を省略すると，トレース情報のすべての情報を標準出力に出力します。

- -n

  編集対象のファイル中に取得されているトレース情報を，サービスの実行順に並べ替え，標準出力に出力します。

- -t　編集開始日時，編集終了日時

  トレース情報の編集範囲を編集対象日時で指定します。編集対象日時は，最初にサービス要求したプロセスの取得日時を基に，補正した日時です。

  編集開始日時，および編集終了日時は，1970 年 1 月 1 日 0 時 0 分 0 秒から，現在の日時までの範囲で指定します。

  編集開始日時を省略すると，編集対象ファイルの先頭から，指定した編集終了日時までを出力します。

  編集終了日時を省略すると，指定した編集開始日時から，編集対象ファイルの最後までを出力します。

  編集開始日時，編集終了日時は，次の形式で指定します。

  hhmmss〔MMDD〔YYYY〕〕

  hh：時（00≦hh≦23）

  mm：分（00≦mm≦59）

  ss：秒（00≦ss≦59）

  MM：月（01≦MM≦12）

  DD：日（01≦DD≦31）

  YY：年（西暦）（1970≦YY≦9999）

  編集開始日時，または編集終了日時の「年」の指定を省略すると，当年の指定日時と見なされます。また，「年月日」の指定を省略すると，当年，当月，当日の指定時刻と見なされます。

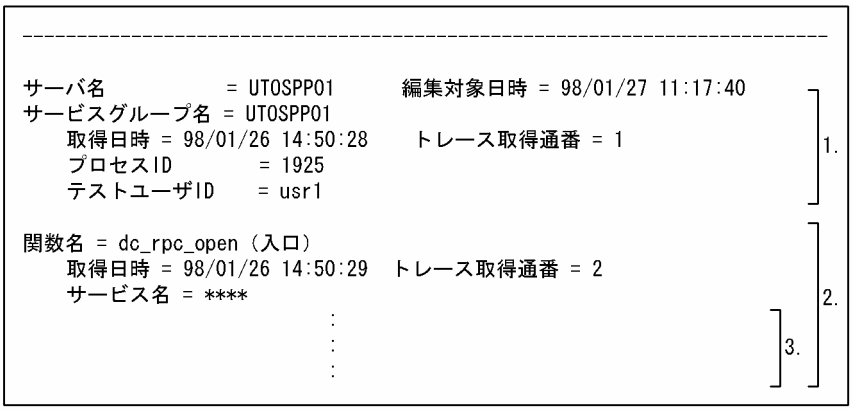
  なお，編集開始日時と編集終了日時のどちらかは，必ず指定してください。

(5)　コマンド引数

- 編集対象ファイル名　　～〈パス名〉

  編集するトレースファイルの名称，またはトレースマージファイルの名称を指定します。

(6) 出力形式（-i オプションを省略した場合）



1. テスタ情報
   - 起動した UAP のサーバ名（8 文字以内）
   - 最初にサービス要求したプロセスの取得日時を基に補正した日時
     年（西暦下 2 けた）/月/日△時：分：秒の形式
   - 起動していたサービスがあるサービスグループの名称（31 文字以内）
     SUP の場合は，'****'を表示します。
   - テスタ情報，または UAP トレースを取得した日時
     年（西暦下 2 けた）/月/日△時：分：秒の形式
   - トレースを取得したエントリの順序番号（6 けた）
   - 該当する UAP トレースを取得したプロセスのプロセス ID
   - UAP を起動したユーザのテストユーザ ID（4 文字以内）
2. UAP トレース情報（uatdump -e コマンドの出力形式と同じ）
   - 取得したトレース情報の種別
   - トレース情報を取得したタイミング
     関数がサービス関数開始・終了の場合，表示しません。
   - テスタ情報，または UAP トレースを取得した日時
     年（西暦下 2 けた）/月/日△時：分：秒の形式
   - トレースを取得したエントリの順序番号（6 けた）
   - UAP を起動したサービスの名称（31 文字以内）
     SUP またはサービス名が特定できない場合は，'****'を表示します。
3. OpenTP1 関数の呼び出し情報の出力領域

- **-i オプションを省略した場合の出力例**

```
--------------------------------------------------------------------------------

サーバ名           = sppni01          編集対象日時 = 98/03/08 16:22:42
サービスグループ名 = sppni01
    取得日時 = 98/03/08 16:22:42       トレース取得通番 = 1
    プロセスID        = 3895
    テストユーザID    = dam

関数 = dc_rpc_open（入口）
    取得日時 = 98/03/08 16:22:42       トレース取得通番 = 2
    サービス名 = ****
      サーバ名 = sppni01
      オプションフラグ = 0x00000000   (DCNOFLAGS)

関数 = dc_rpc_open（出口）
    取得日時 = 98/03/08 16:22:43       トレース取得通番 = 3
    サービス名 = ****
      サーバ名 = sppni01
      オプションフラグ = 0x00000000   (DCNOFLAGS)
    リターンコード = 0 （正常終了）

関数 = dc_rpc_mainloop（入口）
    取得日時 = 98/03/08 16:22:43       トレース取得通番 = 4
    サービス名 = ****
      オプションフラグ = 0x00000000   (DCNOFLAGS)                              1.
--------------------------------------------------------------------------------
サーバ名       = supni01              編集対象日時 = 98/03/08 16:22:45
サービスグループ名  = ****
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 1
    プロセスID      = 3898
    テストユーザID  = dam

関数 = dc_rpc_open（入口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 2
    サービス名 = ****
      サーバ名 = supni01
      オプションフラグ = 0x00000000   (DCNOFLAGS)

関数 = dc_rpc_open（出口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 3
    サービス名 = ****
      サーバ名 = supni01
      オプションフラグ = 0x00000000   (DCNOFLAGS)
    リターンコード = 0 （正常終了）

関数 = dc_adm_complete（入口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 4
    サービス名 = ****
      オプションフラグ = 0x00000000   (DCNOFLAGS)

関数 = dc_adm_complete（出口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 5
    サービス名 = ****
      オプションフラグ = 0x00000000   (DCNOFLAGS)
    リターンコード = 0 （DC_OK）                                               2.
```

（続く）

（続き）

```
関数 = dc_rpc_call（入口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 6
    サービス名 = ****
  呼び出すサービスのサービスグループ名 = sppni01
      呼び出すサービス名 = svccal
      送信データ長（1024）
      ------ 送信データ ------
      000078    00000000 00000001 00000001 00000000    .... .... .... ....
      000088    00000000 00000073 70706e69 30320000    .... ...s ppni 02..
      000098    00000000 00000000 00000000 00000000    .... .... .... ....
      0000a8    00000000 00000073 76636461             .... ...s vcda
      受信データ長（1024）
      オプションフラグ = 0x00000000   (DCNOFLAGS)
```

2.

```
 ---------------------------------------------------------------------------

サーバ名            = sppni01          編集対象日時 = 98/03/08 16:22:45
サービスグループ名  = sppni01
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 5
    プロセスID     = 3895
    テストユーザID = dam

関数 = サービス関数開始
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 6
    サービス名 = svccal
      コール元ノード名 = 2C3G009
      コール元サービスグループ名 = ****
      コール元サービス名 = ****
      入力電文長（1024）
      ------ 入力電文 ------
      000098    00000000 00000001 00000001 00000000    .... .... .... ....
      0000a8    00000000 00000073 70706e69 30320000    .... ...s ppni 02..
      0000b8    00000000 00000000 00000000 00000000    .... .... .... ....
      0000c8    00000000 00000073 76636461             .... ...s vcda

関数 = dc_trn_begin（入口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 7
    サービス名 = svccal

関数 = dc_trn_begin（出口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 8
    サービス名 = svccal
      ------ XID ------
      000034    01030000 00000018 00000024 00000013    .... .... ...$ ....
      000044    00000f37 40404040 5075746f 00000000    ...7 @@@@ Puto ....
      000054    00000000 00000001 00000013 00000001    .... .... .... ....
      000064    ffffffff 40404040 5075746f 00000f37    .... @@@@ Puto ...7
      000074    00000000 00000000 00000000 00000000    .... .... .... ....
      000084 - 0000a4 :      SAME DATA
      0000b4     00000000 00000000 00000000            .... .... ....
    リターンコード = 0  (DC_OK)
```

3.

（続く）

（続き）

```
関数 = dc_dam_open（入口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 9
    サービス名 = svccal
      リクエストコード = OPEN
      論理ファイル名 = TAMTABLE
      オプションフラグ = 0x00000002   (DCDAM_BLOCK_EXCLUSIVE)

関数 = dc_dam_open（出口）
    取得日時 = 98/03/08 16:22:45       トレース取得通番 = 10
    サービス名 = svccal
      リクエストコード = OPEN
      論理ファイル名 = TAMTABLE
      オプションフラグ = 0x00000002   (DCDAM_BLOCK_EXCLUSIVE)
    リターンコード = 16842753 （ファイル記述子）

関数 = dc_dam_write（入口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 11
    サービス名 = svccal
      リクエストコード = WRIT
      ファイル記述子 = 16842753
      オプションフラグ = 0x00000000   (DCNOFLAGS)
      キーの数 = 1 バッファ長 = 512
      相対ブロック番号 = 3
      000054    00000000 00000000 00000000 00000000    .... .... .... ....
      000064 - 0000e4 :      SAME DATA
      0000f4    00000000 00000000                      .... ....

関数 = dc_dam_write（出口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 12
    サービス名 = svccal
      リクエストコード = WRIT
      論理ファイル名 = TAMTABLE
      ファイル記述子 = 16842753
      オプションフラグ = 0x00000000   (DCNOFLAGS)
      キーの数 = 1 バッファ長 = 512
      相対ブロック番号 = 3
      000054    00000003 7069643d 33383936 20757064    .... pid= 3896  upd
      000064    6174655f 636f756e 743d3100 00000000    ate_ coun t=1. ....
      000074    00000000 00000000 00000000 00000000    .... .... .... ....
      000084 - 0000e4 :      SAME DATA
      0000f4    00000000 00000000                      .... ....
    リターンコード = 0 （DC_OK）

関数 = dc_dam_close（入口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 13
    サービス名 = svccal
      リクエストコード = CLOS
      ファイル記述子 = 16842753
      オプションフラグ = 0x00000000   (DCNOFLAGS)
```

3.

（続く）

（続き）

```
関数 = dc_dam_close（出口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 14
    サービス名 = svccal
      リクエストコード = CLOS
      論理ファイル名 = TAMTABLE
      ファイル記述子 = 16842753
      オプションフラグ = 0x00000000  (DCNOFLAGS)
      リターンコード = 0  (DC_OK)

関数 = dc_trn_unchained_commit（入口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 15
    サービス名 = svccal

関数 = dc_trn_unchained_commit（出口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 16
    サービス名 = svccal
    リターンコード = 0  (DC_OK)

関数 = サービス関数終了
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 17
    サービス名 = svccal
      コール元ノード名 = 2C3G009
      コール元サービスグループ名 = ****
      コール元サービス名 = ****
      出力電文長（13）
      ------ 出力電文 ------
      000098    7376636e 6964616d 20656e64 00           svcn idam  end .

--------------------------------------------------------------------------------

サーバ名          = supni01          編集対象日時 = 98/03/08 16:22:46
サービスグループ名 = ****
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 7
    プロセスID     = 3898
    テストユーザID = dam

関数 = dc_rpc_call（出口）
    取得日時 = 98/03/08 16:22:46      トレース取得通番 = 8
    サービス名 = ****
  呼び出すサービスのサービスグループ名 = sppni01
      呼び出すサービス名 = svccal
      送信データ長（1024）
      ----- 送信データ -----
      000078  00000000 00000001 00000001 00000000  .... .... .... ....
      000088  00000000 00000073 70706e69 30320000  .... ...s ppni 02..
      000098  00000000 00000000 00000000 00000000  .... .... .... ....
      0000a8  00000000 00000073 76636461          .... ...s vcda
      受信データ長（13）
      ----- 受信データ -----
      0000b8  7376636e 6964616d 20656e64 00       svcn idam  end .
      オプションフラグ = 0x00000000（DCNOFLAGS）
    リターンコード = 0  （正常終了）
```

3.

4.

（続く）

（続き）

```
関数 = dc_rpc_close（入口）
    取得日時 = 98/03/08 16:22:46       トレース取得通番 = 9
    サービス名 = ****
      オプションフラグ = 0x00000000（DCNOFLAGS）

関数 = dc_rpc_close（出口）
    取得日時 = 98/03/08 16:22:46       トレース取得通番 = 10
    サービス名 = ****
      オプションフラグ = 0x00000000（DCNOFLAGS）                    4.


--------------------------------------------------------------------------------

サーバ名           = sppni01              編集対象日時 = 98/03/08 16:27:37
サービスグループ名 = sppni01
    取得日時 = 98/03/08 16:27:37       トレース取得通番 = 18
    プロセスID     = 3895
    テストユーザID = dam

関数 = dc_rpc_mainloop（出口）
    取得日時 = 98/03/08 16:27:37       トレース取得通番 = 19
    サービス名 = ****
      オプションフラグ = 0x00000000（DCNOFLAGS）
    リターンコード = 0 （正常終了）                                 5.

関数 = dc_rpc_close（入口）
    取得日時 = 98/03/08 16:27:37       トレース取得通番 = 20
    サービス名 = ****
      オプションフラグ = 0x00000000（DCNOFLAGS）

関数 = dc_rpc_close（出口）
    取得日時 = 98/03/08 16:27:37       トレース取得通番 = 21
    サービス名 = ****
      オプションフラグ = 0x00000000（DCNOFLAGS）
```

- SPP 起動時に取得したトレース情報
- SUP 起動時に取得したトレース情報
- サービス実行時に取得したトレース情報
- SUP 終了時に取得したトレース情報
- SPP 終了時に取得したトレース情報

(7) 出力形式（-i オプションを指定した場合）

```
--------------------------------------------------------------------------------

98/01/26 14:48:37 〈98/01/26 15:50:28〉                          〈1925〉    1.
UTOSPP01（UTOSPP01）〈0〉
dc_rpc_open（入口）
    :                                                                      2.
    :
```

1. テスタ情報

- 最初にサービス要求したプロセスの取得日時を基に補正した日時
  年（西暦下２けた）/月/日△時：分：秒の形式

- テスタ情報を取得した日時
  年（西暦下 2 けた）/月/日△時：分：秒の形式
- 該当する UAP トレースを取得したプロセスのプロセス ID
- 起動した UAP のサーバ名（8 文字以内）
- 起動していたサービスがあるサービスグループの名称（31 文字以内）
  SUP の場合は，'****'を表示します。
- 取得した UAP のネスト番号
  ネスト管理できないトレース情報の場合は，'0'を表示します。
  サービス要求元がシミュレートしたクライアント UAP や，TP1/Client の UAP の場合，サービス要求先のネスト番号は 1 から表示します。

2. UAP トレース情報
   - 取得したトレース情報の種別
   - トレース情報を取得したタイミング
     関数がサービス関数開始・終了の場合は，UAP を起動したサービスの名称（31 文字以内）を表示します。

- **-i オプションを指定した場合の出力例**

```
--------------------------------------------------------------------------
98/03/08 16:22:42 〈98/03/08 16:22:42〉                          〈3895〉┐
sppni01 (sppni01) 〈0〉                                                 │
dc_rpc_open（入口）                                                    │1.
dc_rpc_open（出口）                                                    │
dc_rpc_mainloop（入口）                                                ┘
--------------------------------------------------------------------------
98/03/08 16:22:45 〈98/03/08 16:22:45〉                          〈3898〉┐
supni01 (****) 〈0〉                                                    │
dc_rpc_open（入口）                                                    │
dc_rpc_open（出口）                                                    │2.
dc_adm_complete（入口）                                                │
dc_adm_complete（出口）                                                │
dc_rpc_call（入口）                                                    ┘
--------------------------------------------------------------------------
98/03/08 16:22:45 〈98/03/08 16:22:45〉                          〈3895〉┐
sppni01 (sppni01) 〈1〉                                                 │
サービス関数開始（svccal）                                             │
dc_trn_begin（入口）                                                   │
dc_trn_begin（出口）                                                   │
dc_dam_open（入口）                                                    │
dc_dam_open（出口）                                                    │3.
dc_dam_write（入口）                                                   │
dc_dam_write（出口）                                                   │
dc_dam_close（入口）                                                   │
dc_dam_close（出口）                                                   │
dc_trn_unchained_commit（入口）                                        │
dc_trn_unchained_commit（出口）                                        │
サービス関数終了（svccal）                                             ┘
--------------------------------------------------------------------------
98/03/08 16:22:46 〈98/03/08 16:22:46〉                          〈3898〉┐
supni01 (****) 〈0〉                                                    │
dc_rpc_call（出口）                                                    │4.
dc_rpc_close（入口）                                                   │
dc_rpc_close（出口）                                                   ┘
--------------------------------------------------------------------------
98/03/08 16:22:46 〈98/03/08 16:22:46〉                          〈3895〉┐
sppni01 (sppni01) 〈0〉                                                 │
dc_rpc_mainloop（出口）                                                │5.
dc_rpc_close（入口）                                                   │
dc_rpc_close（出口）                                                   ┘
```

- SPP 起動時に取得したトレース情報
- SUP 起動時に取得したトレース情報
- サービス実行時に取得したトレース情報
- SUP 終了時に取得したトレース情報
- SPP 終了時に取得したトレース情報

(8) 注意事項

- 編集対象ファイル中に，このコマンドより古いバージョンのトレース情報がある場合，警告メッセージを出力し，指定したファイル内に格納されている順序どおりに編集出力します。
- -n オプション指定時に，必要なトレース情報がない（途中のトレース情報が抜けている）場合，警告メッセージを出力します。
- -n オプション指定時，-t オプションの編集開始日時に，編集対象日時と次のグループの編集対象日時との間の日時を指定した場合，次のグループ以降のトレース情報を編集対象とします。

- -n オプション指定時，サービス要求先のトレース情報の編集対象日時が，-t オプションの編集終了日時を超えた場合でも，サービス要求元のネストのレベルに戻るまでのトレース情報を，編集対象とします。
- 全入出力データ取得を指定して取得したトレース情報がある場合，入出力データの途中でテスタ情報が出力されることがあります。
- 各オプションを組み合わせて指定した場合，有効となる組み合わせを次に示します。

指定するオプション	-s	-v	-n	-t	-i
-s	-s	-s，-v	-s，-n	-s，-t	-s，-i
-v	－	-v	-v，-n	-v，-t	-v，-i
-n	－	－	-n	-n，-t	-n，-i
-t	－	－	－	-t	-t，-i
-i	－	－	－	－	-i

（凡例）

-x：-x オプションだけ有効となります。

-x，-y：-x，-y オプションとも有効となります。

－：該当しません。

5.1.11 utoxsppsvc （XATMI インタフェースの SPP へのサービス要求）

(1) 名称

XATMI インタフェースの SPP へのサービス要求

(2) 形式

```
utoxsppsvc 〔-f 送受信制御ファイル名〕 サービス名
           タイプトバッファ定義ファイル名 XATMI要求データファイル名
           〔XATMI応答データファイル名〕
```

(3) 機能

XATMI インタフェースの SPP に対して，特定のサービスの実行を要求します。XATMI を使用した SPP 以外の UAP に対して utoxsppsvc コマンドを実行すると，コマンドエラーとなります。

また，このコマンドは，シミュレート MHP に対しては使用できません。

(4) オプション

- -f 送受信制御ファイル名 ～〈パス名〉

  会話型でサービス要求する場合に，送受信手順を定義した，送受信制御ファイルの名称を指定します。

(5) コマンド引数

- サービス名　〜〈1〜31 文字の識別子〉
  実行するサービスの名称を指定します。
- タイプトバッファ定義ファイル名　〜〈パス名〉
  タイプトバッファの情報を定義した，タイプトバッファ定義ファイルの名称を指定します。
- XATMI 要求データファイル名　〜〈パス名〉
  サービス要求（コネクション確立）時の入力データを格納した，XATMI 要求データファイルの名称を指定します。
- XATMI 応答データファイル名　〜〈パス名〉
  サービス実行中に受け取る受信データやサービス実行後の応答データを格納する，XATMI 応答データファイルの名称を指定します。
  このコマンド引数の指定を省略すると，応答データは破棄されます。
  なお，出力先のファイルがすでにある場合，そのファイルに上書きして出力します。ファイルがない場合は，オンラインテスタがファイルを作成して出力します。

(6) 注意事項

- 複数サービスとの会話はできません。
- サービス要求がリクエスト/レスポンス型か会話型かは，XATMI 要求データファイル内の呼出し種別に設定します。
- リクエスト/レスポンス型のサービス要求で-f オプションを指定すると，無視して処理を続行します。会話型のサービス要求で-f オプションを省略すると，コマンドエラーとなります。
- トランザクション内からのサービス要求はシミュレートできません。
- 送受信制御ファイル内に有効な行がない場合は，エラーにはなりませんが，コネクション確立要求直後にコマンドが終了します。
- 受信データや応答データが一つも受け取れなかった場合，XATMI 応答データファイルは作成されません。一つでもデータを受け取った場合は，その後エラーが発生してもファイルが作成され，エラーまでのデータは残ります。
- タイプトバッファ定義ファイルに指定したタイプトバッファ長と，XATMI 要求データファイルや XATMI 応答データファイルに指定したデータ長が異なる場合は，データファイルのデータ不正となります。
- タイプトバッファ定義ファイルに指定したタイプトバッファ長と，SPP が管理しているバッファ長が異なる場合，次のような現象が発生します。
  1. utoxsppsvc コマンドでのサービス要求のエラー
  2. utoxsppsvc コマンドでのデータ受信エラー
  3. SPP でのデータ受信エラー

- XATMI 応答データファイルがすでにある場合，コマンド開始時にファイル内のデータを削除します。そのため，その後のデータ出力がなくても，ファイル内にあったデータは残りません。

6 障害対策

オンラインテスタに関連して発生する障害と，その対策について説明します。

6.1　オンラインテスタの障害対策

オンラインテスタの障害対策について説明します。オンラインテスタに関連しない障害については，マニュアル「OpenTP1 運用と操作」を参照してください。

6.1.1　障害発生時の現象と原因

オンラインテスタに関連する障害の現象と，推定できる原因を，次の表に示します。

表 6-1　オンラインテスタに関連する障害の現象と原因

現象	原因	対策記述個所
オンラインテスタのコマンドが正常終了しません。	システム定義でオンラインテスタの使用を定義していません。	6.1.2(1)
	オプションまたはコマンド引数の指定が誤っています。	
	コマンドの実行に必要なファイルが使用できません。	
UAP トレース情報が取得できません。	システム定義でテスト対象を指定していません。	6.1.2(2)
	システム定義で最大トレースファイル容量に 0 を指定しています。	
	システム定義で UAP トレース格納最大数に 0 を指定しています。	
送信メッセージが取得できません。	システム定義でテスト対象を指定していません。	6.1.2(3)
	システム定義で最大 MCF 送信メッセージファイル容量に 0 を指定しています。	
	送信メッセージ量が上限値を超えています。	
会話型のサービス要求での送信データが取得できません。	システム定義で送信データの出力を指定していません。	6.1.2(4)
テスト対象の UAP が開始できません。	テストユーザ ID を設定していません。	6.1.4(1)
	システム定義でテスト対象を指定していません。	
テスト対象外の UAP が開始できません。	システム定義でテスト対象を指定しています。	6.1.4(2)
テスト対象の UAP で dc_rpc_open 関数がエラーリターンします。	システム定義でオンラインテスタの使用を定義していません。	6.1.4(3)
テスト対象の UAP がテストユーザ ID：_uto で再起動されます。	OpenTP1 本体とオンラインテスタとの間で UAP の状態管理に矛盾があります。	6.1.4(4)
テスト対象の UAP が異常終了後に回復しません。	テスト対象の UAP がデバッガと連動しています。	6.1.4(5)
デバッガと連動する UAP でタイムアウトエラーが頻繁に発生します。	ユーザサービス定義の各監視時間に，適切な値を指定していません。	6.1.4(6)

6.1.2　オンラインテスタの障害

オンラインテスタに障害が発生した場合の対処方法を説明します。

(1)　オンラインテスタのコマンドが正常終了しない場合

次の処置をしたあと，再度コマンドを実行してください。

- システムサービス構成定義で，オンラインテスタの使用が定義されていない場合，OpenTP1 をいったん停止させ，オンラインテスタの使用を定義（uto_conf オペランドに Y を指定）したあと，再度 OpenTP1 を起動してください。
- コマンドのオプションまたはコマンド引数の指定が誤っている場合，正しいオプションまたはコマンド引数を指定してください。
- コマンドの実行に必要なファイルがない，またはアクセス権限がないために使用できない場合，ファイルを作成するかアクセス権限を変更してください。

(2)　UAP トレース情報が取得できない場合

(a)　トレース情報がまったく取得できない場合

テスタサービス定義で，最大トレースファイル容量（max_trace_file_size オペランド）に 0 を指定している場合，OpenTP1 をいったん停止させ，1 以上の値を指定したあと，再度 OpenTP1 を起動してください。

(b)　特定の UAP でトレース情報が取得できない場合

次の処置をしてください。

- ユーザサービス定義でテスト対象に指定していない場合，UAP をいったん停止させ，テスト対象に指定（test_mode オペランドに target を指定）したあと，再度 UAP を起動してください。
- ユーザサービス定義で UAP トレース格納最大数（uap_trace_max オペランド）に 0 を指定している場合，UAP をいったん停止させ，1 以上の値を指定したあと，再度 UAP を起動してください。

(c)　途中からのトレース情報が取得できない場合

次の処置をしてください。

- トレースファイルが満杯になっている場合，満杯になったトレースファイルを別のファイルに退避したあと，削除してください。
- UAP 実行中にオンラインが即時停止した場合，オンライン再開始後，再度 UAP を実行させてください。
- UAP が異常を検知したときに，退避コアファイルを取得しないで即時停止した場合，退避コアファイルを取得するようにプログラムを変更したあと，再度 UAP を実行させてください。

(3) 送信メッセージが取得できない場合

(a) 送信メッセージがまったく取得できない場合

テスタサービス定義で最大 MCF 送信メッセージファイル容量（max_message_file_size オペランド）に 0 を指定している場合，OpenTP1 をいったん停止させ，1 以上の値を指定したあと，再度 OpenTP1 を起動してください。

(b) 特定の UAP で送信メッセージが取得できない場合

ユーザサービス定義でテスト対象に指定していない場合，UAP をいったん停止させ，テスト対象に指定（test_mode オペランドに target を指定）したあと，再度 UAP を起動してください。

(c) 途中からの送信メッセージが取得できない場合

MCF 送信メッセージファイルが満杯になっている場合，満杯になった MCF 送信メッセージファイルを別のファイルに退避したあと，削除してください。

(4) 会話型サービス要求での送信データが取得できない場合

(a) 送信データがまったく取得できない場合

ユーザサービス定義で送信データの出力（test_xatmi_send_file オペランド）に N を指定している場合，UAP をいったん停止させ，Y を指定したあと，再度 UAP を起動してください。

6.1.3 ファイルの障害

オンラインテスタが作成するファイルに障害が発生した場合，エラーメッセージにファイル名と要因コードが表示されますので，それを基に原因を調査し，対処してください。

6.1.4 UAP の障害

UAP に関する障害が発生した場合の対処方法を説明します。

(1) テスト対象の UAP を開始できない場合

次の処置をしたあと，再度 UAP を起動してください。

- テストユーザ ID が設定されていない場合，テストユーザ ID を設定してください。
- テストユーザ ID に誤りがある場合，正しいテストユーザ ID を再度設定してください。
- ユーザサービス定義でテスト対象に指定していない場合，UAP をいったん停止させ，テスト対象に指定（test_mode オペランドに target を指定）したあと，再度 UAP を起動してください。

(2)　テスト対象外の UAP を開始できない場合

ユーザサービス定義でテスト対象に指定されている場合，テスト対象外に指定（test_mode オペランドに no を指定）するか，またはその定義文を削除したあと，再度 UAP を起動してください。

(3)　テスト対象の UAP で dc_rpc_open 関数がエラーリターンする場合

システムサービス構成定義で，オンラインテスタの使用が定義されていない場合，OpenTP1 をいったん停止させ，オンラインテスタの使用を定義（uto_conf オペランドに Y を指定）したあと，再度 OpenTP1 を起動してください。

(4)　テスト対象の UAP がテストユーザ ID：_uto で開始された場合

次のような場合，テスト対象の UAP がテストユーザ ID に_uto を設定して再起動することがあります。

- UAP の正常終了中に OpenTP1 システム，またはその UAP を強制終了させたあとに，UAP が再起動する場合
- OpenTP1 システムが異常終了したあとに，UAP が再起動する場合

このとき，テストユーザ ID に_uto を設定して再起動する旨を知らせるメッセージが出力されます。この UAP を別のテストユーザ ID で実行したい場合は，UAP をいったん停止させたあと，再度起動してください。

(5)　テスト対象の UAP が異常終了後に回復しない場合

テスト対象の UAP がデバッガと連動している場合，UAP が異常終了しても回復処理は行われません。このとき，UAP の回復を抑止する旨を知らせるメッセージが出力されます。この UAP を再起動する場合は，次の処置をしてください。

- デバッガプロセスが残っている場合は，デバッガを停止させてください。

(6)　デバッガと連動する UAP でタイムアウトエラーが頻繁に発生する場合

ユーザサービス定義の各監視時間の指定値によって，デバッガと連動して起動した UAP で，タイムアウトエラーが頻繁に発生する場合があります。タイムアウトエラーが発生した場合の現象と関連する定義の内容を，次の表に示します。

表 6-2　デバッガと連動する UAP で発生するタイムアウトエラーの現象と関連する定義内容

現象	関連するユーザサービス定義の set 形式
dc_rpc_call 関数がタイムアウトしてエラーリターンします。	set watch_time
タイムアウトエラーによって，該当するトランザクションブランチのプロセスが異常終了して，回復処理が行われます。その後，UAP が強制終了となります。	set trn_expiration_time set trn_cpu_time

現象	関連するユーザサービス定義の set 形式
タイムアウトエラーによって，該当する UAP が異常終了して，UAP が強制終了となります。	set watch_next_chain_time
タイムアウトエラーによって，該当する UAP が異常終了しますが，サービスグループの閉塞が行われず，UAP は強制終了となります。	set term_watch_time

7 機能

TP1/Message Control オンラインテスタの次に示す機能について説明します。

・TP1/Message Control 以外の資源更新処理の無効化

・送信メッセージの無効化

・アプリケーション起動メッセージの無効化

・エラーイベントの抑止

・MHP の自動閉塞機能の抑止

・UAP トレース情報の取得

7.1 MHP のテスト

TP1/Message Control オンラインテスタ（以降，MCF オンラインテスタと呼びます）では，新規に作成した MHP，または変更した MHP をテストし，動作を確認できます。ただし，テストする MHP は，次に示す二つの条件を同時に満たしている必要があります。

- トランザクション内の MHP である
- TP1/Message Control から直接起動される MHP である

MHP をテストするためには，まず MCF オンラインテスタの使用宣言コマンド（mcfutfst コマンド）を実行して，MCF オンラインテスタの機能を使用できる状態にします。その後，テスト開始コマンド（mcftules コマンド，mcfauaps コマンド，または mcftusgs コマンド）を実行してテストを開始します。なお，MCF オンラインテスタの機能が使用できるかどうかを，状態表示コマンド（mcflsutf コマンド）で確認できます。

MCF オンラインテスタでは，次に示す単位でテストができます。

- 論理端末単位
- アプリケーション単位
- サービスグループ単位

テスト単位によってテスト開始のコマンドが異なります。

論理端末単位の場合，mcftules コマンドを使用します。テスト範囲は，mcftules コマンドに指定した論理端末からのメッセージをアプリケーションが受け取ってから，テスト中のメッセージがなくなるまでの間です。

アプリケーション単位の場合，mcfauaps コマンドを使用します。テスト範囲は，mcfauaps コマンドで指定したアプリケーションがメッセージを受け取ってからテスト中のメッセージがなくなるまでの間です。なお，アプリケーション単位でテストする場合は，mcfauaps コマンドの-k オプションで，テストするアプリケーションの種別を指定できます。指定できる種別は，ユーザアプリケーション，または MCF イベントです。

サービスグループ単位の場合，mcftusgs コマンドを使用します。テスト範囲は，mcftusgs コマンドで指定したサービスグループが起動している間です。

また，テスト開始コマンドには，ユーザが使用する MCF オンラインテスタの機能を，オプションで指定します。指定できる機能を次に示します。

7.1.1 MCF 以外の資源更新処理の無効化

テスト中の MHP がメッセージ処理中に更新した MCF 以外のリソースマネジャの資源を，トランザクション終了時，更新前の状態に回復します。そのため，テスト後に資源を回復する必要はありません。

この機能を使用する場合，テスト開始コマンドの-e オプションに backout を指定します。

7.1.2 送信メッセージの無効化

テスト中の MHP が送信したメッセージを無効にします。そのため，オンラインの業務に影響を与えないで MHP をテストできます。

無効になるのは，テスト中の MHP が発行した次に示す関数によって送信されるメッセージです。

- dc_mcf_send 関数（メッセージの送信）
- dc_mcf_sendsync 関数（同期型のメッセージの送信）
- dc_mcf_resend 関数（メッセージの再送）

この機能を使用する場合，テスト開始コマンドの-e オプションに swmsg を指定します。

なお，次に示す関数によって送信されるメッセージは無効にできません。

- dc_mcf_reply 関数（応答メッセージの送信）
- dc_mcf_sendrecv 関数（同期型のメッセージの送受信）

7.1.3 アプリケーション起動メッセージの無効化

テスト中の MHP が送信した，分岐のアプリケーション起動メッセージを無効にします。そのため，オンラインの業務に影響を与えないで MHP をテストできます。

この機能を使用する場合，テスト開始コマンドの-e オプションに execap を指定します。

なお，応答メッセージに関するアプリケーション起動メッセージは無効にできません。

7.1.4 エラーイベントの抑止

テスト中の MHP で発生するエラーイベントを抑止します。そのため，オンラインの業務に影響を与えないで MHP をテストできます。

抑止できるエラーイベントを次に示します。

- 不正アプリケーション名検出通知イベント（ERREVT1）
  ただし，抑止できるのは論理端末単位のテストの場合だけです。
- dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄通知イベント（ERREVT2）
- 閉塞に伴って発生するメッセージ廃棄通知イベント（ERREVT2）

- MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）

この機能を使用する場合，テスト開始コマンドの-e オプションに errevt を指定します。

なお，次に示すエラーイベントは抑止できません。

- 未処理送信メッセージ廃棄イベント（ERREVTA）
- タイマ起動メッセージ廃棄通知イベント（ERREVT4）

7.1.5　MHP の自動閉塞機能の抑止

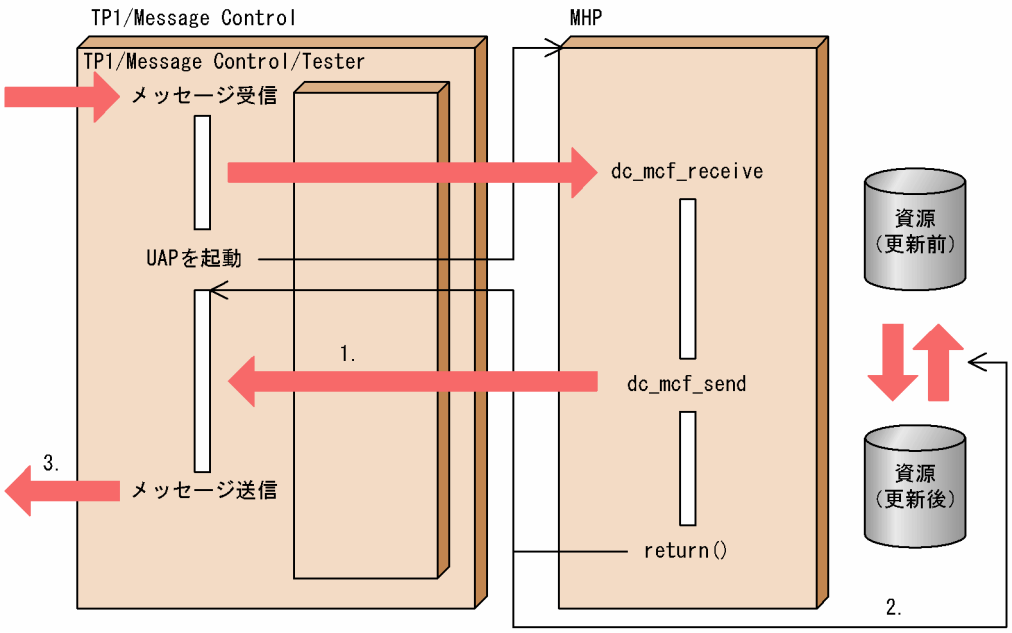
MHP が異常終了したときに働く MHP の自動閉塞機能を抑止します。そのため，運用コマンドを入力して閉塞解除する必要はありません。

この機能を使用する場合，テスト開始コマンドの-e オプションに holdlimit を指定します。

テスト開始コマンドを実行したあとは，MCF オンラインテスタを使用しない場合と同様の手順でアプリケーションを起動します。

メッセージの受信から送信までのトランザクション処理の例を次の図に示します。MHP が異常終了した場合は，トランザクションを開始する前の状態に回復されます。

図 7-1　メッセージ受信から送信までのトランザクション処理の例



(説明)

1. テスト中の送信メッセージは，次に示す表に従った扱いになります。

メッセージ種別	テスト開始コマンドの指定	
	swmsg 指定あり	swmsg 指定なし
問い合わせ応答メッセージ	送信する	送信する
分岐メッセージ	送信しない※	送信する

注※

送信時にインタフェースをチェックし，エラーのときはエラーステータスコードを返します。

2. TP1/Message Control 以外のリソースマネジャの資源を，テスト開始前の状態に回復する指定をしている場合，TP1/Message Control 以外のリソースマネジャの資源を回復します。

   資源回復の指定がない場合，更新した資源は更新したままとなります。

3. 送信メッセージがあれば，送信メッセージを出力します。

7.2　テスト情報の取得

7.2.1　UAP トレース情報の取得

テスト中の MHP のトレース情報を取得できます。そのため，テスト中の MHP の動作を確認できます。ただし，TP1/Server Base のオンラインテスタを使用する必要があります。

この機能を使用する場合，MCF オンラインテスタの使用宣言コマンド（mcfutfst コマンド）の-u オプションにテストユーザ ID を指定し，テスト開始コマンド（mcftules コマンド，mcfauaps コマンド，または mcftusgs コマンド）の-e オプションに trace を指定します。

8 テストの実行

テストの開始と終了，テストモードが重複した場合の扱い，テストモード情報の引き継ぎと編集について説明します。

8.1　テストの開始と終了

テスト開始コマンド（mcftules コマンド，mcfauaps コマンド，または mcftusgs コマンド）実行からテスト終了コマンド（mcftulee コマンド，mcfauape コマンド，または mcftusge コマンド）実行までの，MCF オンラインテスタの機能が有効である状態を**テストモード**といいます。

8.1.1　テストの開始とテスト環境の指定

MCF オンラインテスタの使用宣言コマンド（mcfutfst コマンド）を実行して MCF オンラインテスタの機能を使用できる状態にしたあと，テスト開始コマンドでテストの開始を宣言し，**テスト環境**（ユーザが使用する MCF オンラインテスタの機能）を指定します。テスト開始コマンドの指定内容を**テストモード情報**といいます。

MCF オンラインテスタの機能が使用できる状態かどうかは，MCF オンラインテスタの状態表示コマンド（mcflsutf コマンド）を実行すると，確認できます。

(1)　テストの開始

(a)　論理端末単位のテスト

mcftules コマンドでテストを開始します。

mcftules コマンドを実行すると，指定した名称の論理端末から起動するアプリケーションが，すべてテストモードになります。この状態を論理端末がテストモードであるといいます。

(b)　アプリケーション単位のテスト

mcfauaps コマンドでテストを開始します。

mcfauaps コマンドを実行すると，指定した名称のアプリケーションがテストモードになります。この状態をアプリケーションがテストモードであるといいます。

アプリケーションのテストは，既存のユーザアプリケーションプログラムにアプリケーションの処理を新規に追加した場合に実行します。

(c)　サービスグループ単位のテスト

mcftusgs コマンドでテストを開始します。

mcftusgs コマンドを実行すると，指定した名称のサービスグループがテストモードになります。この状態をサービスグループがテストモードであるといいます。

(d) テスト開始コマンド実行時の注意

なお，テスト開始コマンドは，コネクションを閉塞してメッセージ送受信のない状態にしてから実行してください。メッセージ送受信中にテスト開始コマンドを実行すると，該当するアプリケーションが次に起動されたときからテストモードになります。

(2) テスト環境の指定

テスト開始コマンドに，次に示すテスト環境を指定します。

- TP1/Message Control 以外の資源更新処理の無効化
- 送信メッセージの無効化
- アプリケーション起動メッセージの無効化
- エラーイベントの抑止
- MHP の自動閉塞機能の抑止
- UAP トレース情報の取得

(3) テストモードの有効範囲

テストモードのアプリケーションに対する入力メッセージ，およびテストモードの論理端末から入力したメッセージを，**テストモードメッセージ**といいます。

テストモードの有効範囲は，MHP がテストモードメッセージを受信してから，テスト中のメッセージがなくなるまでの間です。

8.1.2 テストの終了

運用コマンドを入力できるワークステーションからテスト終了コマンド（mcftulee コマンド，mcfauape コマンド，または mcftusge コマンド）を入力して，テストの終了を宣言します。

テスト終了コマンドを実行すると，指定した論理端末，アプリケーション，またはサービスグループのテストモードが解除されます。

8.2　テストモードの重複

あるアプリケーションに対して，テストモードになる指示単位が二つ以上ある場合，そのアプリケーションに対するテスト環境は，論理端末，アプリケーション，サービスグループの順に有効になります。

例えば，あるアプリケーションをテストモードの論理端末から入力する場合，該当するアプリケーションのテスト環境を mcfauaps コマンドで指定しても，mcftules コマンドで指定した論理端末のテスト環境が有効になります。テストモードでない論理端末から入力したアプリケーションに対して mcfauaps コマンドでテスト環境を指定した場合は，そのテスト環境が有効になります。

論理端末とアプリケーションのテストモードの重複指定に関する扱いを，次の表に示します。

表 8-1　テストモードの重複指定に関する扱い

論理端末	アプリケーション	サービスグループ	テストモード情報が有効な単位
○	○	○	論理端末
		×	論理端末
	×	○	論理端末
		×	論理端末
×	○	○	アプリケーション
		×	アプリケーション
	×	○	サービスグループ
		×	－

（凡例）

○：テスト指定があります。

×：テスト指定がありません。

－：該当しません。

8.3　テストモード情報の引き継ぎ

テスト開始コマンドで指定したテストモード情報は，テスト中の MHP が dc_mcf_execap 関数（アプリケーションプログラムの起動）を発行した場合，起動先の MHP に引き継がれます。

テストモード情報の引き継ぎを次の表に示します。

表 8-2　テストモード情報の引き継ぎ

<table>
<tr><th rowspan="4">論理端末</th><th rowspan="4">アプリケーション</th><th rowspan="4">サービスグループ</th><th colspan="4">dc_mcf_execap で指定したアプリケーション</th></tr>
<tr><td colspan="2">○</td><td colspan="2">×</td></tr>
<tr><td colspan="2">サービスグループ</td><td colspan="2">サービスグループ</td></tr>
<tr><td>○</td><td>×</td><td>○</td><td>×</td></tr>
<tr><td rowspan="4">○</td><td rowspan="2">○</td><td>○</td><td colspan="4" rowspan="4">論理端末に対するテストモード情報を引き継ぎます。</td></tr>
<tr><td>×</td></tr>
<tr><td rowspan="2">×</td><td>○</td></tr>
<tr><td>×</td></tr>
<tr><td rowspan="4">×</td><td rowspan="2">○</td><td>○</td><td colspan="2" rowspan="4">dc_mcf_execap で指定したアプリケーションに対するテストモード情報を引き継ぎます。</td><td rowspan="2">dc_mcf_execap で起動された MHP に対するテストモード情報を引き継ぎます。</td><td>起動元のアプリケーションに対するテストモード情報を引き継ぎます。</td></tr>
<tr><td>×</td></tr>
<tr><td rowspan="2">×</td><td>○</td><td>起動元の MHP に対するテストモード情報を引き継ぎます。</td></tr>
<tr><td>×</td><td>－</td></tr>
</table>

（凡例）

○：テスト指定があります。

×：テスト指定がありません。

－：該当しません。

8.4　テスト情報の編集

8.4.1　テストモード情報の表示

テスト開始コマンドで指定した論理端末，アプリケーションまたはサービスグループ（MHP）のテストモード情報は，テストモード情報の表示コマンド（mcftulsle，mcfaulsap，または mcftulssg コマンド）を入力すると，標準出力に出力できます。そのため，オペレータはオンライン中のテスト状況をつかめ，テストを監視できます。

8.4.2　UAP トレース情報の取得

MHP の UAP トレース情報を取得するためには，あらかじめ次に示す指定をしておく必要があります。

- システムサービス構成定義で uto_conf=Y と指定
- テスタサービス定義の max_trace_file_size に最大トレースファイル容量を指定
- ユーザサービス定義で test_mode=no と指定
- MCF オンラインテスタの使用宣言コマンドの-u オプションにテストユーザ ID を指定
- MCF オンラインテスタのテスト開始コマンドの-e オプションに trace を指定

8.4.3　UAP トレース情報のマージ・編集出力

MHP の UAP トレース情報を取得し，オンラインテスタの utotrcout コマンドを実行すると，MHP のトレース情報を編集して，標準出力に出力できます。出力形式は TP1/Server Base のオンラインテスタの仕様に従います。

なお，次に示す関数のトレース情報は出力されません。

- dc_mcf_open 関数
- dc_mcf_close 関数
- dc_mcf_regster 関数
- dc_mcf_mainloop 関数

MHP のトレース情報を出力するディレクトリは，「$DCDIR/spool/uto/テストユーザ ID」となります。テストユーザ ID は，MCF オンラインテスタの使用宣言コマンドの-u オプションに指定したテストユーザ ID です。

なお，MHP のトレースファイルには，「trace1」と「trace2」の二つがあります。テスタサービス定義で指定した最大トレースファイル容量を超える書き込みが発生すると，ファイルが切り替わります。このと

き，トレースファイルが満杯になって切り替わった旨のメッセージが出力されます。このメッセージが出力されたら，ユーザは満杯になったトレースファイルの内容を別ファイルにコピーしたあと，満杯になったトレースファイルを削除してください。

システムサービス構成定義，テスタサービス定義，およびユーザサービス定義については「3.1　オンラインテスタのシステム定義」を参照してください。また，utotrcout コマンドについては「5.1　テストで使用する運用コマンド」を参照してください。

9 運用コマンド

MCF オンラインテスタで使用する運用コマンドについて説明します。

9.1　テストで使用する運用コマンド

MCF オンラインテスタで使用できる運用コマンドについて説明します。運用コマンドの形式，規則などについては，マニュアル「OpenTP1 運用と操作」を参照してください。

テストで使用する運用コマンドの一覧を，次の表に示します。

表 9-1　テストで使用する運用コマンドの一覧

コマンド名	機能
mcfutfst	MCF オンラインテスタの使用を宣言します。
mcflsutf	MCF オンラインテスタの状態を表示します。

9.1.1　mcfutfst（MCF オンラインテスタの使用宣言）

(1)　名称

MCF オンラインテスタの使用宣言

(2)　形式

```
mcfutfst 〔-u テストユーザID〕
```

(3)　機能

MCF オンラインテスタの使用を宣言します。

mcfutfst コマンドを実行して MCF オンラインテスタの使用を宣言しないと，mcfutfst コマンド，および mcflsutf コマンド以外の MCF オンラインテスタのコマンドは受け付けられません。

すでに MCF オンラインテスタの使用を宣言している場合は，mcfutfst コマンドは受け付けられません。

mcfutfst コマンドは，コネクションを閉塞してメッセージ送受信のない状態にしてから実行してください。

なお，MCF オンラインテスタは TP1/Message Control 終了時に終了します。

(4)　オプション

- -u　テストユーザ ID　　～〈1～4 文字の識別子〉

  トレースファイルのディレクトリを決めるテストユーザ ID を指定します。

  MHP のトレース情報を取得する場合は，このオプションを必ず指定してください。

9.1.2　mcflsutf（MCF オンラインテスタの状態表示）

(1)　名称

MCF オンラインテスタの状態表示

(2)　形式

```
mcflsutf
```

(3)　機能

MCF オンラインテスタの機能が使用できる状態かどうかを標準出力に出力します。

MCF オンラインテスタの機能を使用する場合は，mcfutfst コマンドを実行して，MCF オンラインテスタの使用を宣言しておく必要があります。

(4)　出力形式

```
A00  MCFモード=TEST テストユーザID=mhp
 1.              2.                3.
```

1. MCF マネジャプロセス識別子，および MCF 通信プロセス識別子
2. MCF モードの表示
   TEST：MCF オンラインテスタの機能が使用できる状態
   NORMAL：MCF オンラインテスタの機能が使用できない状態
3. テストユーザ ID が指定されていない場合は，'****'を表示します。

9.2　論理端末単位のテストで使用する運用コマンド

MCF オンラインテスタの，論理端末単位のテストで使用するコマンドについて説明します。運用コマンドの形式，規則などについては，マニュアル「OpenTP1 運用と操作」を参照してください。

論理端末単位のテストで使用する運用コマンドの一覧を，次の表に示します。

表 9-2　論理端末単位のテストで使用する運用コマンドの一覧

コマンド名	機能
mcftulsle	論理端末のテストモード情報を表示します。
mcftules	論理端末単位のテストを開始します。
mcftulee	論理端末単位のテストを終了します。

9.2.1　mcftulsle（論理端末のテストモード情報の表示）

(1)　名称

論理端末のテストモード情報の表示

(2)　形式

```
mcftulsle -l 論理端末名称
```

(3)　機能

指定した論理端末のテストモード情報を標準出力に出力します。

(4)　オプション

- -l　論理端末名称　　～〈1～8 文字の識別子〉

  テストモード情報を表示する論理端末の名称を指定します。

  論理端末名称に '*'を指定すると，テストモードのすべての論理端末についてテストモード情報を表示します。また，論理端末名称の先行文字列に続いて '*'を指定（**先行文字列***）すると，先行文字列で始まるすべての論理端末のテストモード情報を表示します。

  複数の論理端末名称は指定できません。

(5)　出力形式

```
A01 LEual1 back trac swms erre exec hold
 1.   2.    3.   4.   5.   6.   7.   8.
```

1. MCF マネジャプロセス識別子，および MCF 通信プロセス識別子
2. 論理端末名称（8 文字以内）
3. トランザクション終了時，資源をテスト前の状態に回復するかどうかの表示
   back：回復する
   nobk：回復しない
4. テストモードのトランザクションの処理中に，MHP のトレース情報を取得するかどうかの表示
   trac：取得する
   notr：取得しない
5. テストモードのトランザクションが発行した送信メッセージを無効にするかどうかの表示
   swms：無効にする
   nosw：無効にしない
6. エラーイベントの起動を抑止するかどうかの表示
   erre：抑止する
   noer：抑止しない
7. テストモードのトランザクションが発行したアプリケーション起動メッセージを無効にするかどうかの表示
   exec：無効にする
   noex：無効にしない
8. テストモードのトランザクションが異常終了した場合に，MHP の自動閉塞機能を抑止するかどうかの表示
   hold：抑止する
   noho：抑止しない

9.2.2 mcftules（論理端末単位のテストの開始）

(1) 名称

論理端末単位のテストの開始

(2) 形式

```
mcftules 〔-e ”〔{backout|ba}〕 〔{trace|tr}〕 〔{swmsg|sw}〕
         〔{errevt|er}〕 〔{execap|ex}〕 〔{holdlimit|ho}〕”〕

         -l 論理端末名称
```

(3) 機能

指定した論理端末をテストモードにし，テストを開始します。

mcftules コマンドは，コネクションを閉塞してメッセージ送受信のない状態にしてから実行してください。

(4) オプション

- -e

  テストモードのオプションを指定します。

  複数のフラグ引数を指定するときは，引用符（"）で囲んで，フラグ引数とフラグ引数との間を空白で区切ります。

  (フラグ引数)

  backout

  トランザクション終了時，テスト中にトランザクションで使用した資源をテスト前の状態に回復します。

  このフラグ引数の指定を省略すると，テスト前の状態に回復しないで，テスト中に更新した資源の状態が有効になります。

  trace

  テストモードのトランザクションの処理中に，MHP の UAP トレース情報を取得します。

  このフラグ引数の指定を省略すると，MHP の UAP トレース情報は取得されません。

  swmsg

  テストモードのトランザクションの処理中に MHP が送信したメッセージを無効にします。無効になるのは，テスト中の MHP が発行した次に示す関数によって送信されるメッセージです。

  ・dc_mcf_send 関数（メッセージの送信）

  ・dc_mcf_sendsync 関数（同期型のメッセージの送信）

  ・dc_mcf_resend 関数（メッセージの再送）

  このフラグ引数の指定を省略すると，これらの関数によって送信したメッセージが有効になります。

  errevt

  テスト中にエラーイベントが発生した場合，エラーイベントの起動を抑止します。抑止できるエラーイベントを次に示します。

  ・不正アプリケーション名検出通知イベント（ERREVT1）

  ・dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄通知イベント（ERREVT2）

  ・閉塞に伴って発生するメッセージ廃棄通知イベント（ERREVT2）

  ・MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）

  このフラグ引数の指定を省略すると，これらのエラーイベントの起動は抑止されません。

execap

テストモードのトランザクションが発行した分岐のアプリケーション起動メッセージを無効にします。

このフラグ引数の指定を省略すると，分岐のアプリケーション起動メッセージが有効になります。

holdlimit

テストモードの MHP が異常終了した場合に，MHP の自動閉塞機能を抑止します。

このフラグ引数の指定を省略すると，MHP の自動閉塞機能は抑止されません。

- -l　論理端末名称　　～〈1～8 文字の識別子〉

テストを開始する論理端末の名称を指定します。

論理端末名称に'*'および'先行文字列*'と指定した一括指定はできません。

複数の論理端末名称は指定できません。

9.2.3　mcftulee（論理端末単位のテストの終了）

(1)　名称

論理端末単位のテストの終了

(2)　形式

```
mcftulee -l 論理端末名称
```

(3)　機能

指定した論理端末のテストモードを解除し，テストを終了します。

(4)　オプション

- -l　論理端末名称　　～〈1～8 文字の識別子〉

テストを終了する論理端末の名称を指定します。

論理端末名称に'*'および'先行文字列*'と指定した一括指定はできません。

複数の論理端末名称は指定できません。

9.3　アプリケーション単位のテストで使用する運用コマンド

MCF オンラインテスタの，アプリケーション単位のテストで使用するコマンドについて説明します。運用コマンドの形式，規則などについては，マニュアル「OpenTP1 運用と操作」を参照してください。

アプリケーション単位のテストで使用する運用コマンドの一覧を，次の表に示します。

表 9-3　アプリケーション単位のテストで使用する運用コマンドの一覧

コマンド名	機能
mcfaulsap	アプリケーションのテストモード情報を表示します。
mcfauaps	アプリケーション単位のテストを開始します。
mcfauape	アプリケーション単位のテストを終了します。

9.3.1　mcfaulsap（アプリケーションのテストモード情報の表示）

(1)　名称

アプリケーションのテストモード情報の表示

(2)　形式

```
mcfaulsap -s {MCF通信プロセス識別子|アプリケーション起動プロセス識別子}
          -a アプリケーション名　〔-k アプリケーション名種別〕
```

(3)　機能

指定したアプリケーションのテストモード情報を標準出力に出力します。

(4)　オプション

- -s　MCF 通信プロセス識別子｜アプリケーション起動プロセス識別子
  ～〈16 進数字〉((01～ef))
  MCF 通信プロセス識別子，またはアプリケーション起動プロセス識別子を指定します。
  ERREVT，または dc_mcf_execap 関数で指定したアプリケーションをテストする場合，アプリケーション起動プロセス識別子を指定します。そのほかの場合は MCF 通信プロセス識別子を指定します。
  複数の MCF 通信プロセス識別子，または複数のアプリケーション起動プロセス識別子は指定できません。
- -a　アプリケーション名　　～〈1～8 文字の識別子〉
  テストモード情報を表示するアプリケーションの名称を指定します。

アプリケーション名に '*'を指定すると，テストモードのすべてのアプリケーションについてテストモード情報を表示します。また，アプリケーション名の先行文字列に続いて '*'を指定（**先行文字列***）すると，先行文字列で始まるすべてのアプリケーションのテストモード情報を表示します。

複数のアプリケーション名は指定できません。

- **-k　アプリケーション名種別**

  -a オプションで指定するアプリケーションの種別を指定します。

  user：ユーザアプリケーション

  mcf：MCF イベント

  このオプションの指定を省略すると，-a オプションで指定するアプリケーション名は，ユーザアプリケーション名であると仮定されます。

(5)　出力形式

```
A01 user aprep01 back trac swms erre exec hold
 1.   2.     3.    4.   5.   6.   7.   8.   9.
```

1. MCF マネジャプロセス識別子，MCF 通信プロセス識別子，またはアプリケーション起動プロセス識別子
2. アプリケーション名種別の表示

   user：ユーザアプリケーション

   mcf：MCF イベント
3. アプリケーション名，または MCF イベント名
4. トランザクション終了時，資源をテスト前の状態に回復するかどうかの表示

   back：回復する

   nobk：回復しない
5. テストモードのトランザクションの処理中に，MHP のトレース情報を取得するかどうかの表示

   trac：取得する

   notr：取得しない
6. テストモードのトランザクションが発行した送信メッセージを無効にするかどうかの表示

   swms：無効にする

   nosw：無効にしない
7. エラーイベントの起動を抑止するかどうかの表示

   erre：抑止する

   noer：抑止しない
8. テストモードのトランザクションが発行したアプリケーション起動メッセージを無効にするかどうかの表示

   exec：無効にする

noex：無効にしない

9. テストモードのトランザクションが異常終了した場合に，MHP の自動閉塞機能を抑止するかどうかの表示

hold：抑止する

noho：抑止しない

9.3.2 mcfauaps（アプリケーション単位のテストの開始）

(1) 名称

アプリケーション単位のテストの開始

(2) 形式

```
mcfauaps -s {MCF通信プロセス識別子|アプリケーション起動プロセス識別子}
         〔-e ”〔{backout|ba}〕 〔{trace|tr}〕 〔{swmsg|sw}〕
         〔{errevt|er}〕 〔{execap|ex}〕 〔{holdlimit|ho}〕”〕
         -a アプリケーション名 〔-k アプリケーション名種別〕
```

(3) 機能

指定したアプリケーションをテストモードにし，テストを開始します。

mcfauaps コマンドは，コネクションを閉塞してメッセージ送受信のない状態にしてから実行してください。

(4) オプション

- -s MCF 通信プロセス識別子｜アプリケーション起動プロセス識別子

  ～〈16 進数字〉((01～ef))

  MCF 通信プロセス識別子，またはアプリケーション起動プロセス識別子を指定します。

  dc_mcf_execap 関数で指定したアプリケーションをテストする場合，アプリケーション起動プロセス識別子を指定します。ERREVT をテストする場合は，次の表に従って指定します。そのほかの場合は MCF 通信プロセス識別子を指定します。

  複数の MCF 通信プロセス識別子，または複数のアプリケーション起動プロセス識別子は指定できません。

表 9–4 ERREVT をテストする場合に指定する識別子（mcfauaps コマンド）

テストする ERREVT	指定する識別子	
	MCF 通信プロセス識別子	アプリケーション起動プロセス識別子
不正アプリケーション名通知イベント（ERREVT1）	○	×

テストする ERREVT	指定する識別子	
	MCF 通信プロセス識別子	アプリケーション起動プロセス識別子
dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄イベント（ERREVT2）	×	○
閉塞に伴って発生するメッセージ廃棄イベント（ERREVT2）	○※	○※
MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）	×	○

（凡例）

○：指定してください。

×：指定しないでください。

注※

MCF 通信プロセス識別子とアプリケーション起動プロセス識別子を，両方とも指定します。

- -e

テストモードのオプションを指定します。

複数のフラグ引数を指定するときは，引用符（"）で囲んで，フラグ引数とフラグ引数との間を空白で区切ります。

（フラグ引数）

backout

トランザクション終了時，テストモードのトランザクションで使用した資源をテスト前の状態に回復します。

このフラグ引数の指定を省略すると，テスト前の状態に回復しないで，テスト中に更新した資源の状態が有効になります。

trace

テストモードのトランザクションの処理中に，MHP の UAP トレース情報を取得します。

このフラグ引数の指定を省略すると，MHP の UAP トレース情報は取得されません。

swmsg

テストモードのトランザクションの処理中に MHP が送信したメッセージを無効にするかどうかを指定します。無効になるのは，テスト中の MHP が発行した次に示す関数によって送信されるメッセージです。

・dc_mcf_send 関数（メッセージの送信）

・dc_mcf_sendsync 関数（同期型のメッセージの送信）

・dc_mcf_resend 関数（メッセージの再送）

このフラグ引数の指定を省略すると，これらの関数によって送信されるメッセージが有効になります。

errevt

テスト中にエラーイベントが発生した場合，エラーイベントの起動を抑止します。抑止できるエラーイベントを次に示します。

・不正アプリケーション名検出通知イベント（ERREVT1）

ただし，抑止できるのは論理端末単位のテストの場合だけです。

・dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄通知イベント（ERREVT2）

・閉塞に伴って発生するメッセージ廃棄通知イベント（ERREVT2）

・MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）

このフラグ引数の指定を省略すると，これらのエラーイベントの起動は抑止されません。

execap

テストモードのトランザクションが発行した，分岐のアプリケーション起動メッセージを無効にします。

このフラグ引数の指定を省略すると，分岐のアプリケーション起動メッセージが有効になります。

holdlimit

テストモードの MHP が異常終了した場合に，MHP の自動閉塞機能を抑止します。

このフラグ引数の指定を省略すると，MHP の自動閉塞機能は抑止されません。

- **-a　アプリケーション名　　～〈1～8 文字の識別子〉**

  テストを開始するアプリケーションの名称を指定します。

  アプリケーション名に'*'および'先行文字列*'と指定した一括指定はできません。

  複数のアプリケーション名は指定できません。

- **-k　アプリケーション名種別**

  -a オプションで指定するアプリケーションの種別を指定します。

  user：ユーザアプリケーション

  mcf：MCF イベント

  このオプションの指定を省略すると，-a オプションで指定するアプリケーション名は，ユーザアプリケーション名であると仮定されます。

9.3.3　mcfauape（アプリケーション単位のテストの終了）

(1)　名称

アプリケーション単位のテストの終了

(2)　形式

```
mcfauape -s {MCF通信プロセス識別子|アプリケーション起動プロセス識別子}
         -a アプリケーション名　〔-k アプリケーション名種別〕
```

(3) 機能

指定したアプリケーションのテストモードを解除し，テストを終了します。

(4) オプション

- -s MCF 通信プロセス識別子｜アプリケーション起動プロセス識別子
  ～〈16 進数字〉((01～ef))

  MCF 通信プロセス識別子，またはアプリケーション起動プロセス識別子を指定します。

  dc_mcf_execap 関数で指定したアプリケーションをテストする場合，アプリケーション起動プロセス識別子を指定します。ERREVT をテストする場合は，次の表に従って指定します。そのほかの場合は MCF 通信プロセス識別子を指定します。

  複数の MCF 通信プロセス識別子，または複数のアプリケーション起動プロセス識別子は指定できません。

  表 9-5 ERREVT をテストする場合に指定する識別子（mcfauape コマンド）

テストする ERREVT	指定する識別子	
	MCF 通信プロセス識別子	アプリケーション起動プロセス識別子
不正アプリケーション名通知イベント（ERREVT1）	○	×
dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄イベント（ERREVT2）	×	○
閉塞に伴って発生するメッセージ廃棄イベント（ERREVT2）	○※1	○※2
MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）	×	○

（凡例）

○：指定してください。

×：指定しないでください。

注※1

受信メッセージによる起動先が閉塞の場合です。

注※2

dc_mcf_execap 関数による起動先が閉塞の場合です。

- -a アプリケーション名　～〈1～8 文字の識別子〉

  テストを終了するアプリケーションの名称を指定します。

  アプリケーション名に'*'および'先行文字列*'と指定した一括指定はできません。

  複数のアプリケーション名は指定できません。

- -k アプリケーション名種別

  -a オプションで指定するアプリケーションの種別を指定します。

  user：ユーザアプリケーション

mcf：MCF イベント

このオプションの指定を省略すると，-a オプションで指定するアプリケーション名は，ユーザアプリケーション名であると仮定されます。

9.4　サービスグループ単位のテストで使用する運用コマンド

MCF オンラインテスタの，サービスグループ単位のテストで使用するコマンドについて説明します。運用コマンドの形式，規則などについては，マニュアル「OpenTP1 運用と操作」を参照してください。

サービスグループ単位のテストで使用する運用コマンドの一覧を，次の表に示します。

表 9-6　サービスグループ単位のテストで使用する運用コマンドの一覧

コマンド名	機能
mcftulssg	サービスグループのテストモード情報を表示します。
mcftusgs	サービスグループ単位のテストを開始します。
mcftusge	サービスグループ単位のテストを終了します。

9.4.1　mcftulssg（サービスグループのテストモード情報の表示）

(1)　名称

サービスグループのテストモード情報の表示

(2)　形式

```
mcftulssg  -g  サービスグループ名
```

(3)　機能

指定したサービスグループのテストモード情報を標準出力に出力します。

(4)　オプション

- -g　サービスグループ名　　～〈1～31 文字の識別子〉

  サービスグループ名を指定します。

  サービスグループ名に'*'を指定すると，テストモードのすべてのサービスグループについてテストモード情報を出力します。また，サービスグループ名の先行文字列に続いて'*'を指定（**先行文字列***）すると，先行文字列で始まるすべてのサービスグループのテストモード情報を出力します。

(5)　出力形式

```
A01 SVG01 LEual1 back trac swms erre exec hold
 1.   2.    3.    4.   5.    6.   7.  8.   9.
```

1. MCF マネジャプロセス識別子，および MCF 通信プロセス識別子
2. サービスグループ名
3. 論理端末名称（8 文字以内）
4. トランザクション終了時，資源をテスト前の状態に回復するかどうかの表示
   back：回復する
   nobk：回復しない
5. テストモードのトランザクションの処理中に，MHP のトレース情報を取得するかどうかの表示
   trac：取得する
   notr：取得しない
6. テストモードのトランザクションが発行した送信メッセージを無効にするかどうかの表示
   swms：無効にする
   nosw：無効にしない
7. エラーイベントの起動を抑止するかどうかの表示
   erre：抑止する
   noer：抑止しない
8. テストモードのトランザクションが発行したアプリケーション起動メッセージを無効にするかどうかの表示
   exec：無効にする
   noex：無効にしない
9. テストモードのトランザクションが異常終了した場合に，MHP の自動閉塞機能を抑止するかどうかの表示
   hold：抑止する
   noho：抑止しない

9.4.2 mcftusgs（サービスグループ単位のテストの開始）

(1) 名称

サービスグループ単位のテストの開始

(2) 形式

```
mcftusgs -g サービスグループ名 〔-e ”〔{backout|ba}〕 〔{trace|tr}〕
                                〔{swmsg|sw}〕 〔{errevt|er}〕
                                〔{execap|ex}〕 〔{holdlimit|ho}〕”〕
```

(3) 機能

指定したサービスグループをテストモードにします。mcftusgs コマンドは，コネクションを閉塞してメッセージ送受信のない状態にしてから実行してください。

(4) オプション

- -g　サービスグループ名　　～〈1～31 文字の識別子〉

  テストを開始するサービスグループ名称を指定します。

  サービスグループ名に'*'および'先行文字列*'と指定した一括指定はできません。

  複数のサービスグループ名は指定できません。

- -e

  テストモードのオプションを指定します。

  複数のフラグ引数を指定するときは，引用符（"）で囲んで，フラグ引数とフラグ引数との間を空白で区切ります。

  (フラグ引数)

  backout

  トランザクション終了時，テスト中にトランザクションで使用した資源をテスト前の状態に回復します。

  このフラグ引数の指定を省略すると，テスト前の状態に回復しないで，テスト中に更新した資源の状態が有効になります。

  trace

  テストモードのトランザクションの処理中に，MHP の UAP トレース情報を取得します。

  このフラグ引数の指定を省略すると，MHP の UAP トレース情報は取得されません。

  swmsg

  テストモードのトランザクションの処理中に MHP が送信したメッセージを無効にします。無効になるのは，テスト中の MHP が発行した次に示す関数によって送信されるメッセージです。

  ・dc_mcf_send 関数（メッセージの送信）

  ・dc_mcf_sendsync 関数（同期型のメッセージの送信）

  ・dc_mcf_resend 関数（メッセージの再送）

  このフラグ引数の指定を省略すると，これらの関数によって送信したメッセージが有効になります。

  errevt

  テスト中にエラーイベントが発生した場合，エラーイベントの起動を抑止します。抑止できるエラーイベントを次に示します。

  ・不正アプリケーション名検出通知イベント（ERREVT1）

  ただし，抑止できるのは論理端末単位のテストの場合だけです。

  ・dc_mcf_receive 関数を発行する前に異常終了して通知されるメッセージ廃棄通知イベント（ERREVT2）

・閉塞に伴って発生するメッセージ廃棄通知イベント（ERREVT2）
・MHP 実行中に異常終了して通知される UAP 異常終了通知イベント（ERREVT3）
このフラグ引数の指定を省略すると，これらのエラーイベントの起動は抑止されません。

execap
テストモードのトランザクションが発行した分岐のアプリケーション起動メッセージを無効にします。
このフラグ引数の指定を省略すると，分岐のアプリケーション起動メッセージが有効になります。

holdlimit
テストモードの MHP が異常終了した場合に，MHP の自動閉塞機能を抑止します。
このフラグ引数の指定を省略すると，MHP の自動閉塞機能は抑止されません。

9.4.3 mcftusge（サービスグループ単位のテストの終了）

(1) 名称

サービスグループ単位のテストの終了

(2) 形式

```
mcftusge -g サービスグループ名
```

(3) 機能

指定したサービスグループのテストモードを解除し，テストを終了します。

(4) オプション

- -g サービスグループ名 ～〈1～31 文字の識別子〉
  テストを終了するサービスグループの名称を指定します。
  サービスグループ名に'*'および'先行文字列*'と指定した一括指定はできません。
  複数のサービスグループ名は指定できません。

10 機能

オフラインテスタで実行するテストのための，各種機能について説明します。

10.1　オフラインテスタの機能

オフラインテスタには，UAP をテストするための次のような機能があります。

1. クライアント UAP シミュレート機能

   クライアント UAP がなくてもサーバ UAP がテストできるよう，クライアント UAP の処理をシミュレートする機能です。

2. サーバ UAP シミュレート機能

   サーバ UAP がなくてもクライアント UAP がテストできるよう，サーバ UAP の処理をシミュレートする機能です。

3. MCF シミュレート機能

   MHP および MHP からサービス要求される SPP がテストできるよう，メッセージ送受信処理をシミュレートする機能です。

4. ファイルサービスシミュレート機能

   UAP からの DAM ファイルや TAM ファイルに対するアクセスがテストできるよう，DAM サービスや TAM サービスをシミュレートする機能です。

5. OpenTP1 提供関数シミュレート機能

   OpenTP1 本体が提供する関数と同じ名称のシミュレート関数を使用して，関数の処理をシミュレートする機能です。

6. 運用コマンドシミュレート機能

   テストする UAP が運用コマンドを発行する場合に，コマンドの処理をシミュレートする機能です。

7. テスタファイル作成機能

   各シミュレート機能使用時に必要となるテスタファイルを作成する機能です。

8. 連続実行コマンド機能

   ファイルに設定したオフラインテスタのサブコマンドを，テスト中に連続して実行する機能です。

9. デバッガ連動機能

   テストする UAP をデバッガの制御下で動作させる機能です。

10. オフラインテスタトレース情報取得機能

    テストする UAP のトレース情報を取得する機能です。

10.2　クライアント UAP のシミュレート

オフラインテスタでは，クライアント UAP の代わりとなってサーバ UAP にサービスを要求できます。そのため，ユーザはクライアント UAP がなくてもサーバ UAP をテストできます。これを，**クライアント UAP シミュレート機能**といいます。

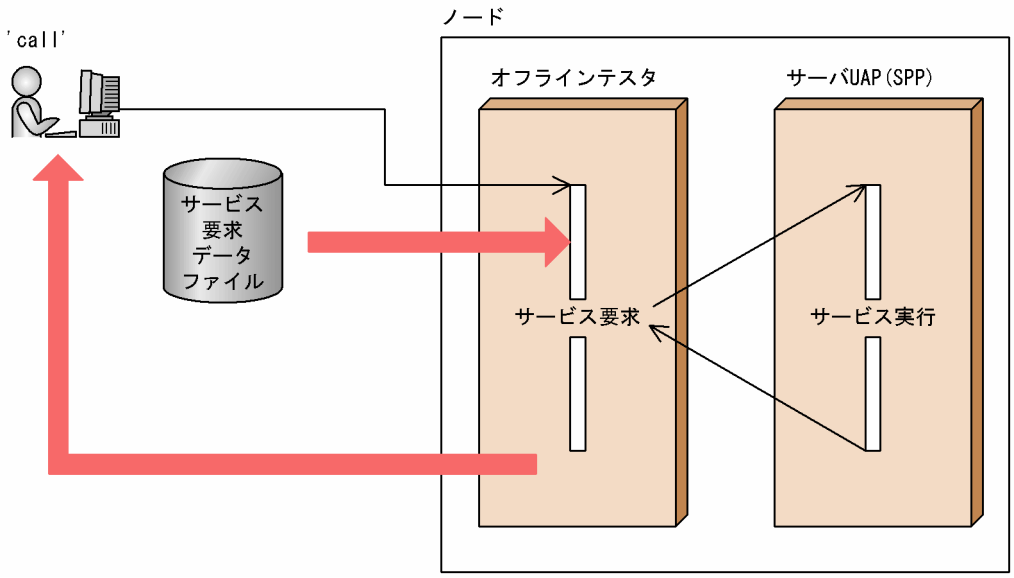
クライアント UAP のシミュレートは，オフラインテスタのコマンドで実行します。ユーザは，サーバ UAP に渡す処理データを，あらかじめ**サービス要求データファイル**に作成しておきます。

サービス要求データファイルには，シミュレートする UAP のインタフェース別に，次の 3 種類があります。

- RPC 要求データファイル（RPC インタフェースの UAP シミュレート時）
- XATMI 要求データファイル（XATMI インタフェースの UAP シミュレート時）
- TxRPC 要求データファイル（TxRPC インタフェースの UAP シミュレート時）

クライアント UAP シミュレート機能の概要を，次の図に示します。

図 10–1　クライアント UAP のシミュレート



10.2.1　RPC インタフェースのクライアント UAP のシミュレート

RPC インタフェースに従ってサービス要求するクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ **RPC 要求データファイル**に作成しておきます。

10.2.2　XATMI インタフェースのクライアント UAP のシミュレート

XATMI インタフェースに従ってサービス要求するクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ **XATMI 要求データファイル**に作成しておきます。

また，会話型のサービスを要求する場合は，テストするサーバ UAP がサービス中に受信するデータを，あらかじめ **XATMI 受信データファイル**に作成しておきます。サーバ UAP からの送信データがある場合は，サービスごとにオフラインテスタがファイル名を問い合わせてきます。ユーザがコマンドで **XATMI 送信データファイル**のファイル名を連絡すると，送信データを XATMI 送信データファイルに取得します。

10.2.3　TxRPC インタフェースのクライアント UAP のシミュレート

TxRPC インタフェースに従ってサービス要求するクライアント UAP をシミュレートする場合，ユーザは，サーバ UAP に渡す処理データを，あらかじめ **TxRPC 要求データファイル**に作成しておきます。

10.3　サーバ UAP のシミュレート

オフラインテスタでは，サーバ UAP の代わりとなってクライアント UAP が要求するサービスを実行できます。そのため，ユーザはサーバ UAP がなくてもクライアント UAP をテストできます。これを，**サーバ UAP シミュレート機能**といいます。

サーバ UAP のシミュレートは，OpenTP1 のコマンドでシミュレートするサーバ UAP を起動して実行します。ユーザは，クライアント UAP へ渡す応答データを，あらかじめ**サービス応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オフラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

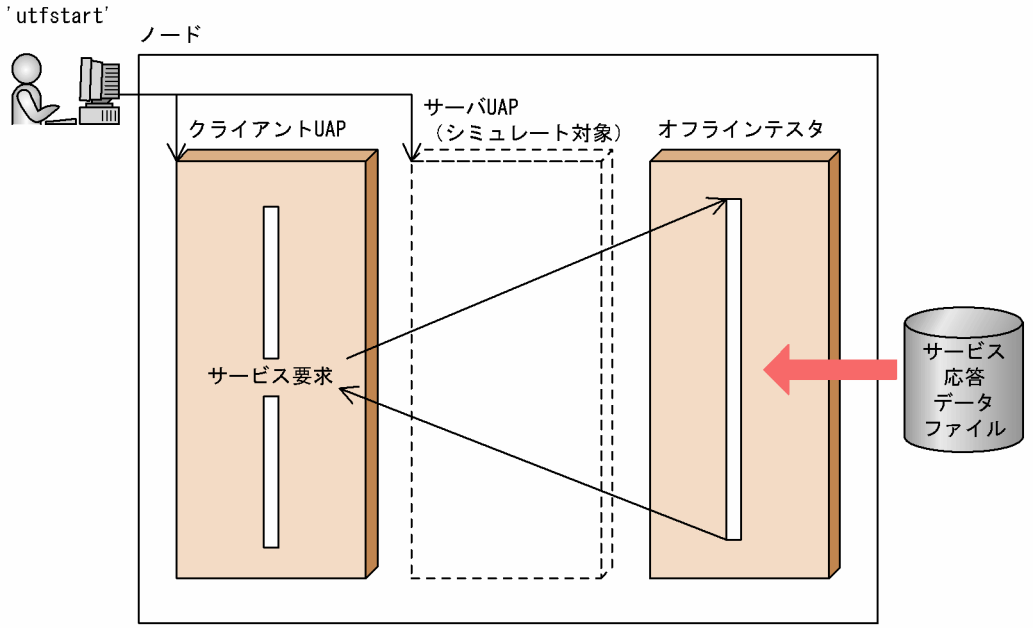
サービス応答データファイルには，シミュレートする UAP のインタフェース別に，次の 3 種類があります。

- RPC 応答データファイル（RPC インタフェースの UAP シミュレート時）
- XATMI 応答データファイル（XATMI インタフェースの UAP シミュレート時）
- TxRPC 応答データファイル（TxRPC インタフェースの UAP シミュレート時）

シミュレートするサーバ UAP は，オフラインテスタ環境定義で，あらかじめ**シミュレート対象**であることを定義しておきます。このサーバ UAP はシミュレートの対象となり，実際には起動されません。

サーバ UAP シミュレート機能の概要を，次の図に示します。

図 10-2　サーバ UAP のシミュレート



10.3.1　RPC インタフェースのサーバ UAP のシミュレート

RPC インタフェースに従ってサービス要求されるサーバ UAP をシミュレートする場合，ユーザは，クライアント UAP へ返す応答データを，あらかじめ **RPC 応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オフラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

10.3.2　XATMI インタフェースのサーバ UAP のシミュレート

XATMI インタフェースに従ってサービス要求されるサーバ UAP をシミュレートする場合，ユーザは，クライアント UAP へ返す応答データを，あらかじめ **XATMI 応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オフラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

また，会話型のサービスを要求する場合は，テストするクライアント UAP がサービス中に受信するデータを，あらかじめ **XATMI 受信データファイル**に作成しておきます。クライアント UAP からの送信データがある場合は，サービスごとにオフラインテスタがファイル名を問い合わせてきます。ユーザがコマンドで **XATMI 送信データファイル**のファイル名を連絡すると，送信データを XATMI 送信データファイルに取得します。

10.3.3　TxRPC インタフェースのサーバ UAP のシミュレート

TxRPC インタフェースに従ってサービス要求されるサーバ UAP をシミュレートする場合，ユーザは，クライアント UAP へ返す応答データを，あらかじめ **TxRPC 応答データファイル**に作成しておきます。クライアント UAP からサービス要求があると，オフラインテスタは応答データをファイルから読み込み，クライアント UAP に返します。

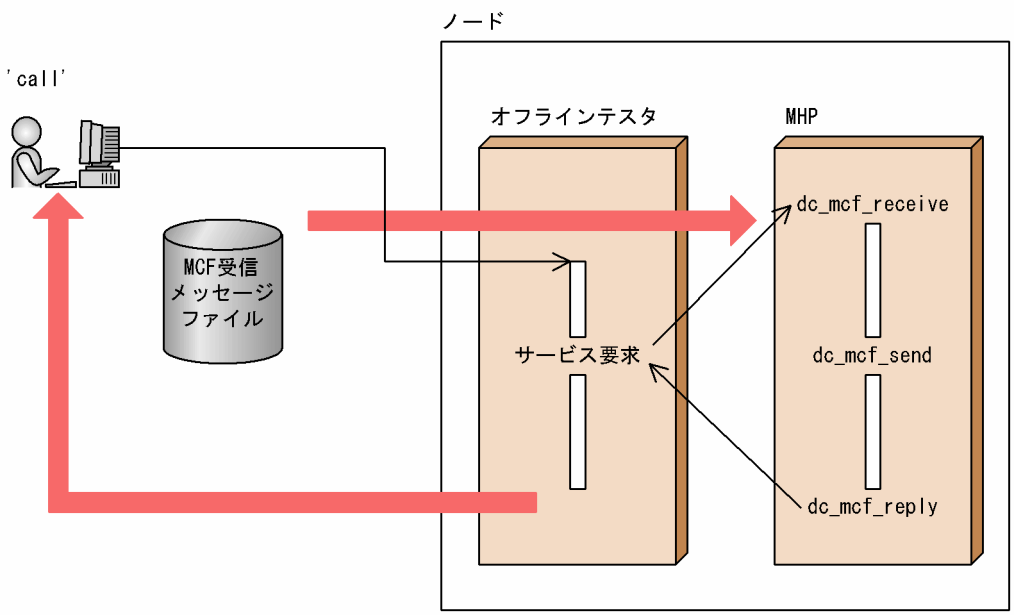
10.4　MCF のシミュレート

オフラインテスタは，TP1/Message Control（以降，MCF と呼びます）の代わりとなって MHP とのメッセージ送受信を実行できます。そのため，ユーザは MCF がなくても MHP をテストできます。これを，**MCF シミュレート機能**といいます。

MHP のアプリケーションの起動は，オフラインテスタのコマンドで実行します。ユーザは，MHP に渡すメッセージを，あらかじめ **MCF 受信メッセージファイル**に作成しておきます。

MCF シミュレート機能の概要を，次の図に示します。

図 10–3　MCF のシミュレート



10.5　ファイルサービスのシミュレート

ここでは，ファイルに対するアクセスをテストするための，ファイルサービスのシミュレートについて説明します。

10.5.1　DAM サービスのシミュレート

オフラインテスタは，UAP からの DAM ファイルに対するアクセスをテストするために，DAM サービスをシミュレートします。これを，**DAM サービスシミュレート機能**といいます。

エディタ，または DAM ファイル作成シミュレート関数（dc_dam_create 関数）で作成したファイルは，TP1/FS/Direct Access のファイルインタフェースで扱えます。ユーザは，オフラインテスタ環境定義で，論理ファイル名と実際のファイルが対になるように定義しておきます。

オフラインテスタでシミュレートする DAM ファイルは，UAP からの更新要求に対しては即時更新（ただし，遅延書き込み）されます。UAP の異常終了や，ロールバック要求が発生した場合は，ファイルは戻さないで，更新したままになります。

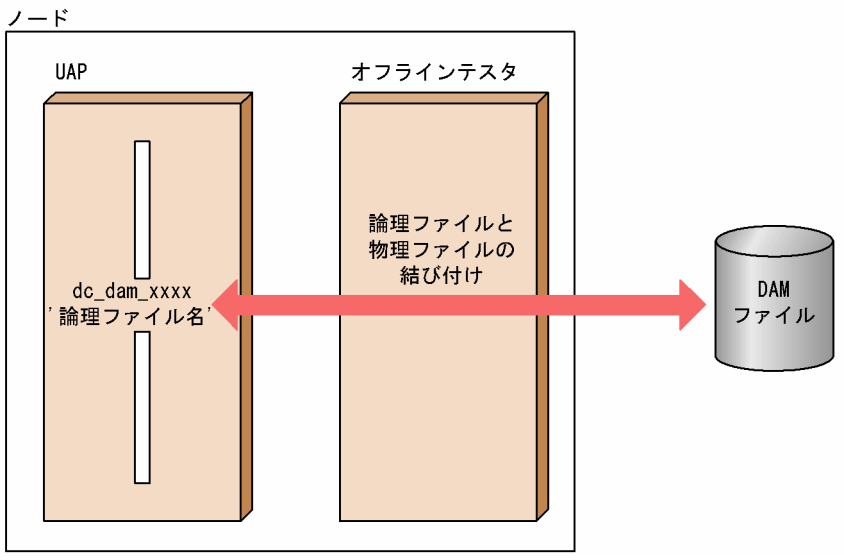
また，オフラインテスタ起動時のオプションの指定で，ファイルの更新を抑止できます。この場合，UAP から更新要求の関数が発行されてもファイル上では更新されません。更新要求後に再入力したデータは，更新要求前の内容のままになります。

さらに，オフラインテスタ環境定義で DAM ファイルの排他の有無を指定できます。ただし，排他は関数での指定に関係なく，ファイル単位になります。

なお，dc_trn_unchained_commit 関数を発行しても，DAM ファイルはクローズしません。

DAM サービスシミュレート機能の概要を，次の図に示します。

図 10-4　DAM サービスのシミュレート



10.5.2　TAM サービスのシミュレート

オフラインテスタは，UAP からの TAM ファイルに対するアクセスをテストするために，TAM サービスをシミュレートします。これを，**TAM サービスシミュレート機能**といいます。

オフラインテスタのコマンド（utftamcre コマンド）で作成したファイルは，TP1/FS/Table Access のファイルインタフェースで扱えます。ユーザは，オフラインテスタ環境定義で，論理ファイル名と実際のファイルが対になるように定義しておきます。

オフラインテスタでシミュレートする TAM ファイルは，TP1/FS/Table Access と同じ TAM データファイルを使用できるほか，インデクスの索引方式も同じです。ただし，TAM ファイルに DAM サービスの関数でアクセスする機能は使用できません。また，このファイルは共用メモリ量の削減のため，TAM ファイルの管理部とインデクス部だけを共用メモリ上に置いているので，データ部には直接 TAM ファイル上でアクセスします。

オフラインテスタでシミュレートする TAM ファイルは，UAP からの更新要求に対しては即時更新（ただし，遅延書き込み）します。UAP の異常終了や，ロールバック要求が発生した場合は，ファイルは戻さないで，更新したままになります。

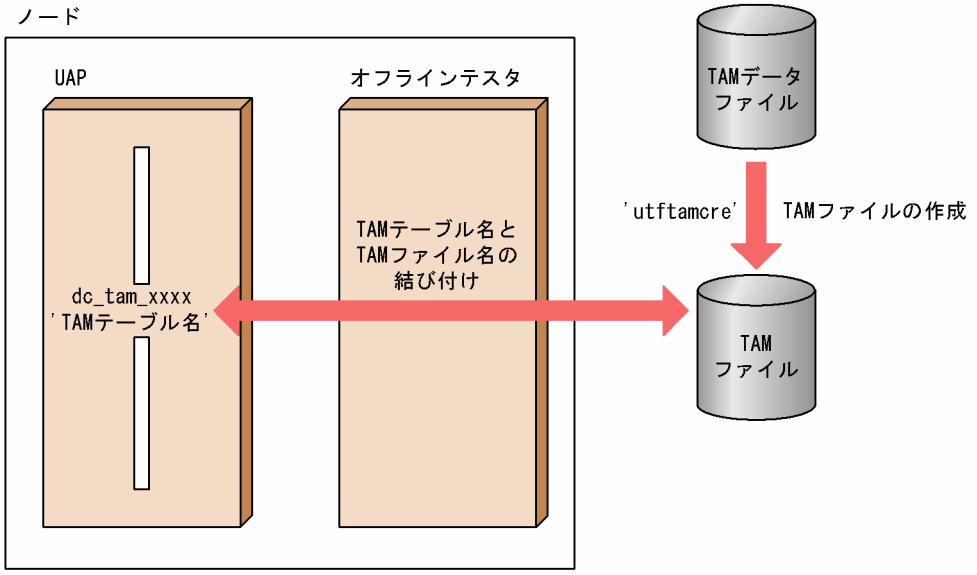
また，オフラインテスタ起動時のオプションの指定で，ファイルの更新を抑止できます。この場合，UAP から更新要求の関数が発行されてもファイル上では更新されません。更新要求後に再入力したデータは，更新要求前の内容のままになります。

さらに，オフラインテスタ環境定義で TAM ファイルの排他の有無を指定できます。ただし，排他は関数での指定に関係なく，TAM ファイル単位になります。

なお，dc_trn_unchained_commit 関数を発行しても，TAM ファイルはクローズしません。

TAM サービスシミュレート機能の概要を，次の図に示します。

図 10–5　TAM サービスのシミュレート



10.6　OpenTP1 提供関数のシミュレート

オフラインテスタには，OpenTP1 本体が提供する関数の代わりとして，各関数ごとに OpenTP1 が提供する関数と同じ名称の関数が用意されています。これを，**シミュレート関数**といいます。シミュレート関数は，UAP とリンケージして使用します。

また，OpenTP1 が提供する関数の実行に対するリターン値を，**関数リターン値ファイル**にあらかじめ設定しておくことで，設定した情報をリターン値として UAP に返すことができます。この機能は，オフラインテスタがする引数チェックに対して，エラーがなかった場合に実行します。引数チェックでエラーが見つかると，エラーに該当するリターン値が UAP に返ります。

ただし，DAM および TAM 関連の関数は，エラーのリターン値を設定したときはその値が返りますが，正常のリターン値を設定したときは，実際に処理した結果がリターン値として返ります。そのため，テスト処理中にエラーが発生すると，そのエラーのリターン値が返ります。

XATMI インタフェースの tpsend 関数と tprecv 関数では，関数リターン値ファイルにイベント名が設定できます。また，TP1/Multi の関数（dc_adm_get_～関数）では，出力データ（ノード識別子とサーバ名）も設定できます。

各シミュレート関数と，その設定できるリターン値については，「14.　関数のシミュレーション内容」を参照してください。

TP1/Shared Table Access が提供する関数を使用する場合は，その関数が使用する IST テーブルをオフラインテスタ起動時のオフラインテスタ環境定義で定義する必要があります。

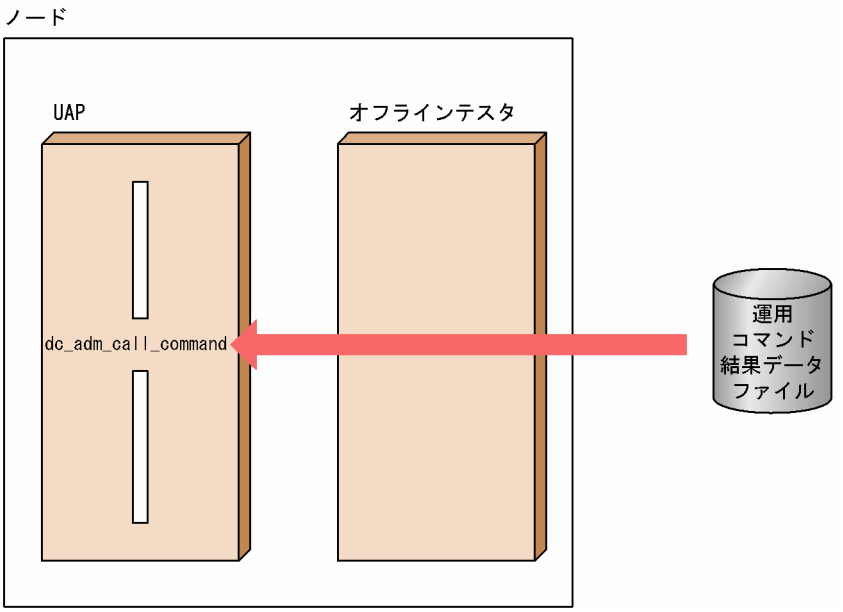
10.7　UAP から発行する運用コマンドのシミュレート

オフラインテスタは，UAP から dc_adm_call_command 関数で運用コマンドの実行を要求した場合，コマンドの実行をシミュレートできます。これを，**運用コマンドシミュレート機能**といいます。

運用コマンドの実行をシミュレートするために，ユーザはあらかじめコマンドの実行結果データを，**運用コマンド結果データファイル**に作成しておきます。UAP で dc_adm_call_command 関数が実行されると，オフラインテスタはコマンドの実行結果データをファイルから読み込み，UAP に返します。

運用コマンドシミュレート機能の概要を，次の図に示します。

図 10–6　UAP から発行する運用コマンドのシミュレート



10.8　テスタファイルの作成

オフラインテスタで各種のシミュレート機能を使用するためには，機能ごとに専用のデータファイルを作成する必要があります。これを**テスタファイル**といいます。

テスタファイルはそれぞれ専用のデータ形式で記述されていますが，オフラインテスタでは，ユーザがコマンドで簡単にテスタファイルを作成できます。これを，**テスタファイル作成機能**といいます。

テスタファイル作成機能で作成できるテスタファイルの一覧を，次の表に示します。

表 10-1　テスタファイル作成機能で作成できるテスタファイルの一覧

テスタファイル名		テスタファイルを使用する機能
サービス要求データファイル	RPC 要求データファイル	クライアント UAP シミュレート機能
	XATMI 要求データファイル	クライアント UAP シミュレート機能
	TxRPC 要求データファイル	クライアント UAP シミュレート機能
サービス応答データファイル	RPC 応答データファイル	サーバ UAP シミュレート機能
	XATMI 応答データファイル	サーバ UAP シミュレート機能
	TxRPC 応答データファイル	サーバ UAP シミュレート機能
XATMI 受信データファイル		クライアント UAP シミュレート機能
MCF 受信メッセージファイル		MCF シミュレート機能
運用コマンド結果データファイル		運用コマンドシミュレート機能

テスタファイル作成機能では，ユーザがあらかじめ作成した**テストデータ定義ファイル**のデータを使用して，テスタファイルを作成します。テストデータ定義ファイルのデータは，テキストエディタで作成できます。また，テストデータ定義ファイルには，複数のテスタファイルのデータをまとめて設定できます。

なお，テスタファイルは，各ファイルの形式に従ってバイナリエディタでも作成できます。

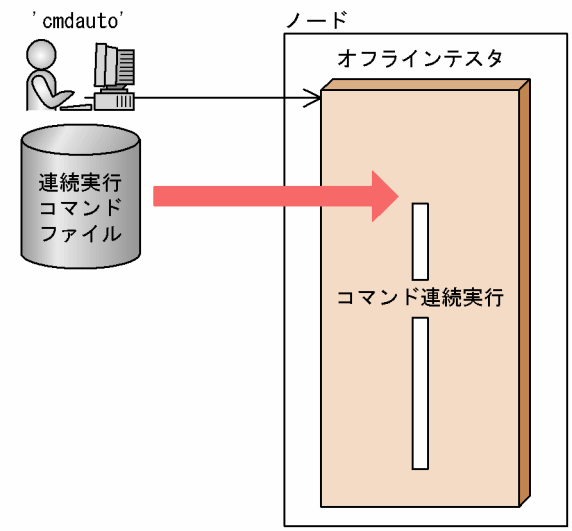
10.9　コマンドの連続実行

オフラインテスタでは，あらかじめファイルに登録しておいたオフラインテスタのコマンドを，順番に実行できます。これを，**連続実行コマンド機能**といいます。

ユーザがコマンドを**連続実行コマンドファイル**に設定しておくと，オフラインテスタがファイルを読み，設定された順にコマンドを実行します。応答用のサブコマンドが指定されていると，そのサブコマンドを実行しますが，指定されていないと，ユーザからの応答待ちになります。そのため，あらかじめテスト順序が決まっている場合などに有効です。

連続実行コマンド機能の概要を，次の図に示します。

図 10-7　コマンドの連続実行



10.10　デバッガの連動

オフラインテスタでは，UAP をデバッガの下で実行できます。これを，**デバッガ連動機能**といいます。

オフラインテスタ環境定義で UAP ごとにデバッガの連動を指定すると，UAP の main 関数からデバッガの制御下で動作します。

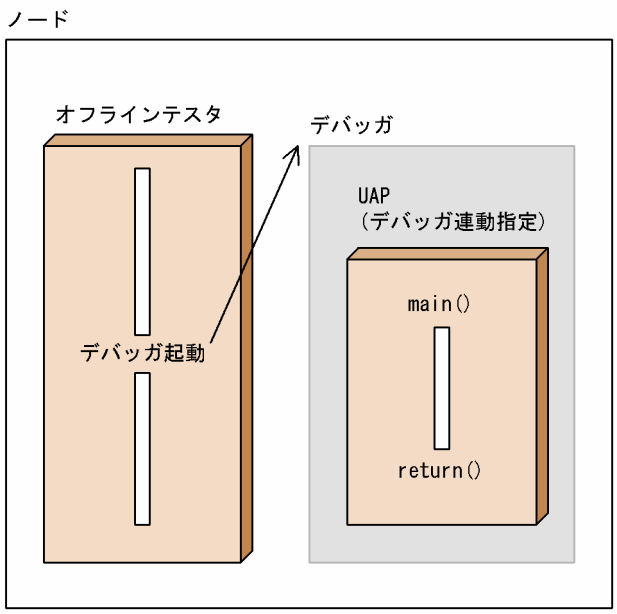
デバッガを連動させると，1 ステップ単位のデバッグやバッチ形式のデバッグが，簡単にできます。

デバッガとして使用できるのは，次の二つです。

- dbx
- cbltd（COBOL2002）

デバッガ連動機能の概要を，次の図に示します。

図 10–8　デバッガの連動



10.11　テスト情報の取得

10.11.1　オフラインテスタトレース情報の取得

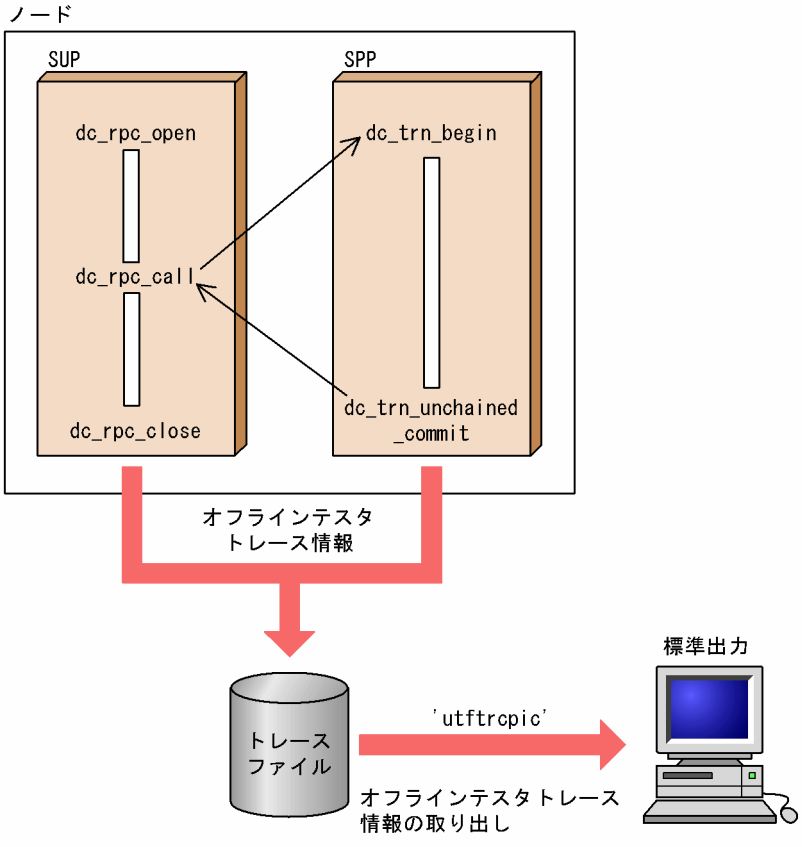
オフラインテスタでは，OpenTP1 の提供する関数発行時の引数の内容や，リターン情報を，トレース情報として標準出力，またはオフラインテスタ環境定義で指定したファイルに出力できます。これを，**オフラインテスタトレース情報取得機能**といいます。標準出力と，ファイルに出力されるトレース情報の内容や形式は同じものです。

また，トレース情報を取得した**トレースファイル**からは，オフラインテスタのコマンドで特定のサービスの情報だけを取り出せます。

なお，複数のオフラインテスタ実行時に，出力先として同じトレースファイルを共用すると，ファイルの内容が破壊されます。トレースファイルは共用しないでください。

オフラインテスタトレース情報取得機能の概要を，次の図に示します。

図 10-9　オフラインテスタトレース情報の取得



11 テスト環境の設定

オフラインテスタで，テストを実行するために必要な環境設定の方法について説明します。

11.1　オフラインテスタのシステム定義

オフラインテスタのシステム定義について説明します。定義の構成，規則などについては，マニュアル「OpenTP1 システム定義」を参照してください。

11.1.1　オフラインテスタ環境定義

オフラインテスタ環境定義では，オフラインテスタの次のような実行条件を定義します。

- UAP の定義
- RPC 要求データファイル格納ディレクトリの定義
- XATMI 要求データファイル格納ディレクトリの定義
- TxRPC 要求データファイル格納ディレクトリの定義
- RPC 応答データファイル格納ディレクトリの定義
- XATMI 応答データファイル格納ディレクトリの定義
- TxRPC 応答データファイル格納ディレクトリの定義
- XATMI 送受信データファイル格納ディレクトリの定義
- MCF 受信メッセージファイル格納ディレクトリの定義
- 運用コマンド結果データファイル格納ディレクトリの定義
- 連続実行コマンドファイル格納ディレクトリの定義
- DAM ファイルの定義
- TAM ファイルの定義
- IST テーブルの定義
- 関数リターン値ファイルの定義
- トレースファイルの定義
- プロトコルの定義

各定義は，**オフラインテスタ環境定義ファイル**上に記述します。オフラインテスタ環境定義ファイルには，ユーザが任意のファイル名を付けます。このファイル名は，オフラインテスタを開始させる際の，コマンド引数になります。

オフラインテスタ環境定義の規則を次に示します。

1. 一つの定義は一つの行に記述します。
2. 定義は，半角文字で記述します。大文字と小文字は区別して記述します。
3. 行（定義）の終わりには，','（コンマ）を記述します。コンマ以降はコメントと見なされます。

4. 全体の終わりには，';'（セミコロン）を記述します。セミコロン以降はコメントと見なされます。
5. 定義解析中，次のような場合には，エラーメッセージが出力されたあと，その定義を無視して定義解析が続行されます。
   - 指定されたディレクトリおよびファイル（DAM ファイル以外）がない場合
   - 書き込み先のファイルが書き込みモードでオープンできないなど，必要なアクセスができない場合
   - 定義解析中にエラーがあった場合

   解析終了後，オフラインテスタの起動を続行するかどうか，ユーザの指定待ちになります。ユーザの指定を次に示します。

   '1'：オフラインテスタの起動を続行します。

   '2'（または'end'）：オフラインテスタの起動を中断します。
6. 定義内容は省略できません。省略すると書式エラーになります。
7. 書式エラーになった場合，その定義内容の有効・無効の判断を次の表に示します。

表 11-1　書式エラーの内容と定義内容の判断

定義名	書式エラーの内容	定義内容	有効時の仮定
UAP の定義	サービスグループ名，実行形式プログラム名，およびユーザサービス定義ファイル名の指定なし	無効	−
	'N'と'F'を同時に指定	有効	'F'だけを仮定
	文末に','または';'なし	有効	文末に','を仮定
RPC 要求データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
XATMI 要求データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
TxRPC 要求データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
RPC 応答データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
XATMI 応答データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
TxRPC 応答データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定
XATMI 送受信データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	−
	文末に','または';'なし	有効	文末に','を仮定

定義名	書式エラーの内容	定義内容	有効時の仮定
MCF 受信メッセージファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
運用コマンド結果データファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
連続実行コマンドファイル格納ディレクトリの定義	ディレクトリ名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
DAM ファイルの定義	物理ファイル名，論理ファイル名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
TAM ファイルの定義	TAM テーブル名，TAM ファイル名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
IST テーブルの定義	IST テーブル名，レコード長，レコード数の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
関数リターン値ファイルの定義	ファイル名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
トレースファイルの定義	ファイル名の指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
プロトコルの定義	プロトコルの指定なし	無効	－
	文末に','または';'なし	有効	文末に','を仮定
その他	上記以外の定義名を指定	無効	－

（凡例）

－：該当しません。

オフラインテスタ環境定義の定義例を次に示します。

〈定義例〉

```
# UAPの定義
SPP = spp1 spp1.out spp1user,
SPP = spp2 DUMMY DUMMY F,
SPP = spp3 spp.out spp3user D dbx,
SPP = spp4 spp4.out spp4user,
SPP = spp5 spp5.out spp5user N D dbx -I /betran/utf/uap/src spp5.out,
MHP = mhp1 mhp1.out mhp1usr,
#
# RPC要求データファイル格納ディレクトリの定義
```

```
rpc_message = /betran/utf/rpcmsg,
#
# XATMI要求データファイル格納ディレクトリの定義
tp_message = /betran/utf/xatmimsg,
#
# TxRPC要求データファイル格納ディレクトリの定義
txrpc_message = /betran/utf/txrpcmsg,
#
# RPC応答データファイル格納ディレクトリの定義
rpc_return_data = /betran/utf/rpc_return,
#
# XATMI応答データファイル格納ディレクトリの定義
tp_return_data = /betran/utf/tp_return,
#
# TxRPC応答データファイル格納ディレクトリの定義
txrpc_return_data = /betran/utf/tx_return,
#
# XATMI送受信データファイル格納ディレクトリの定義
tp_converse = /betran/utf/tp_converse,
#
# MCF受信メッセージファイル格納ディレクトリの定義
mcf_message = /betran/utf/mcfmsg,
#
# 運用コマンド結果データファイル格納ディレクトリの定義
adm_call_cmd = /betran/utf/etc/call_cmd_val,
#
# 連続実行コマンドファイル格納ディレクトリの定義
cmdfile = /betran/utf/etc,
#
# DAMファイルの定義
damfile = damfile1 /betran/utf/dam/damfile1,
damfile = damfile2 /betran/utf/dam/damfile2 N,
#
# TAMファイルの定義
tamtable = tamtable1 /betran/utf/dam/tamfile1,
tamtable = tamtable2 /betran/utf/dam/tamfile2 N,
#
# ISTテーブルの定義
isttable = isttable 128 64,
isttable = ist2 4 256,
#
# 関数リターン値ファイルの定義
func_value_file = /betran/utf/etc/return_val,
#
# トレースファイルの定義
tracefile = /betran/utf/log/trace,
#
# プロトコルの定義
protocol = OSI/TP;
#
```

(1) UAP の定義

(a) 形式

```
{SPP|MHP}={サービスグループ名|サービス名} 実行形式プログラム名
          ユーザサービス定義ファイル名〔 T 〕
          〔{F|D デバッガ名|〔N〕〔D デバッガ名〕}〕{,|;}
```

(b) 機能

オフラインテスタの下で実行する UAP について，次の項目を定義します。

- UAP の種別（SPP，MHP）
- サービスグループ名
- サービスグループ名に対応した実行形式プログラム名
- サービスグループ名に対応したユーザサービス定義ファイル名
- サーバ UAP シミュレート機能を使用するかどうか
- デバッガ連動機能を使用するかどうか

なお，定義の対象となるのは SPP と MHP だけです。SUP は，オフラインテスタでのテストの対象外となります。

(c) オペランド

- **SPP | MHP**

  定義名として，サービスグループの種別を指定します。

  SPP：SPP を示します。

  MHP：MHP を示します。

- **サービスグループ名**

  サービスグループ名を指定します。TxRPC インタフェースの UAP シミュレート機能を実行する場合は，IDL ファイルに指定したインタフェース名を指定します。

- **サービス名**

  XATMI インタフェースのサーバ UAP シミュレート機能を実行する場合に，サービス名を指定します。

- **実行形式プログラム名**

  このサービスグループを実行する UAP 名（実行形式プログラム名）を指定します。サービス関数をシミュレートする場合は，実在しない名称でもかまいません。

- **ユーザサービス定義ファイル名**

  このサービスグループの実行環境を定義した，ユーザサービス定義ファイルの名称を指定します。
  TxRPC インタフェースの UAP シミュレート機能を実行する場合は，**txidl** コマンドで作成したユーザ

サービス定義ファイルを指定します。サービス関数をシミュレートする場合は，実在しない名称でもかまいません。

- T

  TxRPC インタフェースの UAP シミュレート機能を使用する場合に指定します。この指定は，サービスグループ種別に SPP を指定した場合だけ指定できます。MHP を指定した場合はエラーとなります。

- F

  サーバ UAP シミュレート機能を使用する場合に指定します。

  この指定がない場合は，オフラインテスタ開始時にサービスグループを起動します。

- D　デバッガ名　〔デバッガ引数〕

  デバッガの下で UAP を実行させる場合に指定します。

  デバッガ名として指定できるのは次の値です。ただし，デバッガ名のチェックはしません。

  dbx：デバッガとして dbx を使用する場合に指定します。

  cbltd：デバッガとして COBOL2002 を使用する場合に指定します。

  この指定では，デバッガに渡す引数が指定できます。引数を指定しない場合は，実行形式プログラム名を引数として指定します。引数を指定する場合は，その引数をそのままデバッガに渡す引数にします。

  デバッガへ渡す引数で指定する，実行可能ファイル名と実行形式プログラム名は同じにしてください。異なっていてもエラーにはなりませんが，デバッガには引数で指定された実行可能ファイル名が渡され，オフラインテスタでは実行形式プログラム名で管理されます。

- N

  オフラインテスタ開始時に，該当するサービスグループ（UAP）の起動を抑止して，オフラインテスタ開始後に start サブコマンドでサービスグループを起動したい場合に指定します。

  この指定がない場合は，オフラインテスタ開始時にサービスグループを起動します。

(d)　注意事項

同じサービスグループ名を，重複して定義できません。重複して定義した場合は，先に定義した方が有効になります。次に示す例では，'spp1'が重複して指定されているため，2 行目の定義がエラーになります。

(例)

```
SPP = spp1 spp1.load spp1usr,
SPP = spp1 spp2.load spp2usr,  ← エラーになります。
```

(e)　定義例

```
# UAPの定義
SPP = spp1 spp1.out spp1user,
SPP = spp2 DUMMY DUMMY F,
SPP = spp3 spp.out spp3user D dbx,
SPP = spp4 spp4.out spp4user,
SPP = spp5 spp5.out spp5user N D dbx -I /betran/utf/uap/src spp5.out,
MHP = mhp1 mhp1.out mhp1usr;
#
```

(2) RPC 要求データファイル格納ディレクトリの定義

(a) 形式

```
rpc_message = RPC要求データファイル格納ディレクトリ名{,|;}
```

(b) 機能

RPC 要求データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- rpc_message
  定義名として，rpc_message を指定します。
- RPC 要求データファイル格納ディレクトリ名
  RPC 要求データファイルを格納する，ディレクトリのパス名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，RPC 要求データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# RPC要求データファイル格納ディレクトリの定義
rpc_message = /betran/utf/rpcmsg;
#
```

(3) XATMI 要求データファイル格納ディレクトリの定義

(a) 形式

```
tp_message = XATMI要求データファイル格納ディレクトリ名{,|;}
```

(b) 機能

XATMI 要求データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- tp_message
  定義名として，tp_message を指定します。
- XATMI 要求データファイル格納ディレクトリ名
  XATMI 要求データファイルを格納する，ディレクトリのパス名を指定します。

オフラインテスタを実行中のディレクトリと異なる場合は，XATMI 要求データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# XATMI要求データファイル格納ディレクトリの定義
tp_message = /betran/utf/xatmimsg;
#
```

(4) TxRPC 要求データファイル格納ディレクトリの定義

(a) 形式

```
txrpc_message = TxRPC要求データファイル格納ディレクトリ名{,|;}
```

(b) 機能

TxRPC 要求データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- txrpc_message

  定義名として，txrpc_message を指定します。

- TxRPC 要求データファイル格納ディレクトリ名

  TxRPC 要求データファイルを格納する，ディレクトリのパス名を指定します。

  オフラインテスタを実行中のディレクトリと異なる場合は，TxRPC 要求データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# TxRPC要求データファイル格納ディレクトリの定義
txrpc_message = /betran/utf/txrpcmsg;
#
```

(5) RPC 応答データファイル格納ディレクトリの定義

(a) 形式

```
rpc_return_data = RPC応答データファイル格納ディレクトリ名{,|;}
```

(b) 機能

RPC 応答データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- rpc_return_data

  定義名として，rpc_return_data を指定します。

- RPC 応答データファイル格納ディレクトリ名

  RPC 応答データファイルを格納する，ディレクトリのパス名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，RPC 応答データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# RPC応答データファイル格納ディレクトリの定義
rpc_return_data = /betran/utf/rpc_return;
#
```

(6) XATMI 応答データファイル格納ディレクトリの定義

(a) 形式

```
tp_return_data = XATMI応答データファイル格納ディレクトリ名{,|;}
```

(b) 機能

XATMI 応答データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- tp_return_data

  定義名として，tp_return_data を指定します。

- XATMI 応答データファイル格納ディレクトリ名

  XATMI 応答データファイルを格納する，ディレクトリのパス名を指定します。

  オフラインテスタを実行中のディレクトリと異なる場合は，XATMI 応答データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# XATMI応答データファイル格納ディレクトリの定義
tp_return_data = /betran/utf/tp_return;
#
```

(7) TxRPC 応答データファイル格納ディレクトリの定義

(a) 形式

```
txrpc_return_data = TxRPC応答データファイル格納ディレクトリ名{,|;}
```

(b) 機能

TxRPC 応答データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- txrpc_return_data

  定義名として，txrpc_return_data を指定します。

- TxRPC 応答データファイル格納ディレクトリ名

  TxRPC 応答データファイルを格納する，ディレクトリのパス名を指定します。

  オフラインテスタを実行中のディレクトリと異なる場合は，TxRPC 応答データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# TxRPC応答データファイル格納ディレクトリの定義
txrpc_return_data = /betran/utf/tx_return;
#
```

(8) XATMI 送受信データファイル格納ディレクトリの定義

(a) 形式

```
tp_converse = XATMI送受信データファイル格納ディレクトリ名{,|;}
```

(b) 機能

XATMI 送信データファイルと XATMI 受信データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- tp_converse

  定義名として，tp_converse を指定します。

- XATMI 送受信データファイル格納ディレクトリ名

XATMI 送信データファイルと XATMI 受信データファイルを格納する，ディレクトリのパス名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，XATMI 送信データファイルと XATMI 受信データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# XATMI送受信データファイル格納ディレクトリの定義
tp_converse = /betran/utf/tp_converse;
#
```

(9) MCF 受信メッセージファイル格納ディレクトリの定義

(a) 形式

```
mcf_message = MCF受信メッセージファイル格納ディレクトリ名{,|;}
```

(b) 機能

MCF 受信メッセージファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- mcf_message

  定義名として，mcf_message を指定します。

- MCF 受信メッセージファイル格納ディレクトリ名

  MCF 受信メッセージファイルを格納する，ディレクトリのパス名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，MCF 受信メッセージファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# MCF受信メッセージファイル格納ディレクトリの定義
mcf_message = /betran/utf/mcfmsg;
#
```

(10) 運用コマンド結果データファイル格納ディレクトリの定義

(a) 形式

```
adm_call_cmd = 運用コマンド結果データファイル格納ディレクトリ名{,|;}
```

(b) 機能

運用コマンド結果データファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- adm_call_cmd

  定義名として，adm_call_cmd を指定します。

- 運用コマンド結果データファイル格納ディレクトリ名

  運用コマンド結果データファイルを格納する，ディレクトリのパス名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，運用コマンド結果データファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# 運用コマンド結果データファイル格納ディレクトリの定義
adm_call_cmd = /betran/utf/etc/call/cmd/val;
#
```

(11) 連続実行コマンドファイル格納ディレクトリの定義

(a) 形式

```
cmdfile = 連続実行コマンドファイル格納ディレクトリ名{,|;}
```

(b) 機能

連続実行コマンドファイルを格納する，ディレクトリの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- cmdfile

  定義名として，cmdfile を指定します。

- 連続実行コマンドファイル格納ディレクトリ名

  連続実行コマンドファイルを格納する，ディレクトリのパス名を指定します。

  オフラインテスタを実行中のディレクトリと異なる場合は，連続実行コマンドファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# 連続実行コマンドファイル格納ディレクトリの定義
cmdfile = /betran/utf/etc;
#
```

(12) DAMファイルの定義

(a) 形式

```
damfile = 論理ファイル名 物理ファイル名 〔N〕{,|;}
```

(b) 機能

DAMサービスのシミュレートで使用するファイルを，論理ファイル名と物理ファイル名で対にして指定します。

UAPがアクセスするすべてのDAMファイルについて定義します。

(c) オペランド

- damfile
  定義名として，damfileを指定します。
- 論理ファイル名
  論理ファイル名を指定します。
- 物理ファイル名
  オフラインテスタ用DAMファイルの名称を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，DAMファイルを格納しているディレクトリ名を付けて指定します。
- N
  関数での指定に関係なく，排他をしない場合に指定します。

(d) 定義例

```
# DAMファイルの定義
damfile = damfile1 /betran/utf/dam/damfile1,
damfile = damfile2 /betran/utf/dam/damfile2 N;
#
```

(13) TAMファイルの定義

(a) 形式

```
tamtable = TAMテーブル名 TAMファイル名 〔N〕{,|;}
```

(b) 機能

TAMサービスのシミュレートで使用するTAMファイルを，TAMテーブル名とTAMファイル名で対にして指定します。

UAPがアクセスするすべてのTAMファイルについて定義します。

(c) オペランド

- tamtable
  定義名として，tamtable を指定します。
- TAM テーブル名
  TAM テーブル名を指定します。TAM テーブル名は TAM サービスの関数で使用する名称です。
- TAM ファイル名
  オフラインテスタ用 TAM ファイルの名称を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，TAM ファイルを格納しているディレクトリ名を付けて指定します。
- N
  関数での指定に関係なく，排他をしない場合に指定します。また，COBOL の UAP で TAM ファイルにアクセスする場合に指定します。

(d) 定義例

```
# TAMファイルの定義
tamtable = tamtable1 /betran/utf/tam/tamfile1,
tamtable = tamtable2 /betran/utf/tam/tamfile2 N;
#
```

(e) 注意事項

- TAM テーブル名と TAM ファイル名は，重複して指定できません。重複して指定した場合は，先に定義したものが有効になります。

(14) IST テーブルの定義

(a) 形式

```
isttable = ISTテーブル名 レコード長 レコード数{,|;}
```

(b) 機能

IST サービスのシミュレートで使用する IST テーブルを，IST テーブル名，レコード長とレコード数で組にして指定します。

UAP がアクセスするすべての IST テーブルについて定義します。

IST テーブルは 64 個まで定義できます。

(c) オペランド

- isttable
  定義名として，isttable を指定します。

- IST テーブル名

  IST テーブル名を指定します。IST テーブル名は IST サービスの関数で使用する名称です。

- レコード長

  IST テーブルのレコード長をバイト単位で指定します。

- レコード数

  IST テーブルが持つレコードの個数を指定します。

(d) 定義例

```
# ISTテーブルの定義
isttable = isttbl1 8 12,
isttable = isttbl2 10 20;
#
```

(15) 関数リターン値ファイルの定義

(a) 形式

```
func_value_file = 関数リターン値ファイル名{,|;}
```

(b) 機能

関数リターン値を設定する，ファイルの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- func_value_file

  定義名として，func_value_file を指定します。

- 関数リターン値ファイル名

  関数リターン値ファイル名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，関数リターン値ファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# 関数リターン値ファイルの定義
func_value_file = /betran/utf/etc/return_val;
#
```

(16) トレースファイルの定義

(a) 形式

```
tracefile = トレースファイル名{,|;}
```

(b) 機能

オフラインテスタトレース情報を格納する，ファイルの名称を定義します。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- tracefile
  定義名として，tracefile を指定します。
- トレースファイル名
  トレースファイル名を指定します。オフラインテスタを実行中のディレクトリと異なる場合は，トレースファイルを格納しているディレクトリ名を付けて指定します。

(d) 定義例

```
# トレースファイルの定義
tracefile = /betran/utf/log/trace;
#
```

(17) プロトコルの定義

(a) 形式

```
protocol = プロトコル名{,|;}
```

(b) 機能

TP1/Message Control でのプロトコルを定義します。この定義は，COBOL，DML で作成した UAP をテストする場合でだけ有効です。OSI TP 以外のプロトコルの場合は，この定義はしません。重複して定義した場合は，最後に定義したものが有効になります。

(c) オペランド

- protocol
  定義名として，protocol を指定します。
- プロトコル名
  プロトコル名を指定します。
  OSI/TP：OSI TP プロトコルを示します。

ただし，OSI TP 以外を指定した場合は，エラーになります。

(d) 定義例

```
# プロトコルの定義
protocol = OSI/TP;
#
```

11.1.2 ユーザサービス定義

オフラインテスタの実行に必要な定義を，ユーザサービス定義で定義します。定義方法や記述形式は，OpenTP1 のユーザサービス定義と同じです。ユーザサービス定義については，マニュアル「OpenTP1 システム定義」を参照してください。

(1) 形式

(a) set 形式

```
set service = ”サービス名 = エントリポイント名”
              〔,”サービス名 = エントリポイント名”〕
              … 〔set server_type = betran|xatmi〕
```

(b) putenv 形式

```
{{ 〔putenv 環境変数名 環境変数値〕}}
```

(2) 機能

ユーザサービス定義として定義することで，オフラインテスタを実行できます。

(3) オペランド

(a) set 形式

- service = "サービス名 = エントリポイント名"

  このサービスグループに属するサービス名と，そのサービスを使用できるエントリポイント名の組を，すべてのサービスについて指定します。

  エントリポイント名とは C 言語の関数であり，COBOL 言語のプログラム名，または入り口名のことです。ここで指定するエントリポイント名は，RPC（XATMI）インタフェース定義に指定する名称と同じものを指定します。

  RPC（XATMI）インタフェースのサービス関数，および RPC（XATMI）インタフェース定義については，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。

- server_type = betran | xatmi　　～《betran》
  OpenTP1（RPC）の関数を使用するか，XATMI の関数を使用するかを指定します。
  betran：OpenTP1（RPC）の関数を使用します。
  xatmi：XATMI の関数を使用します。

(b)　putenv 形式

- 環境変数名
  このサービスグループのプロセスで，指定した環境変数に値を設定します。
  COBOL 言語の動作環境を OpenTP1 が起動するため，COBOL 言語環境の設定に使用します。UAP の実行形式プログラムごとに，任意の環境変数を与えられます。標準 C 言語ライブラリ'putenv'を参照してください。

(4)　定義例

(a)　set 形式

```
set service      = ”service = xwsvkd0100”
set server_type  = betran
```

(b)　putenv 形式

```
putenv    CBLCORE 1
```

11.1.3　関数リターン値の設定

OpenTP1 が提供する関数のリターン値として任意の値を返す場合，あらかじめ関数リターン値ファイルを作成して，リターン値を設定します。

また，XATMI 関数の tpsend 関数と tprecv 関数のイベントタイプや，マルチノード機能で使用する関数に渡す出力データ（ノード識別子，サーバ名）も設定できます。

定義方法や記述形式は，オフラインテスタ環境定義と同じです。

(1)　形式

(a)　リターン値を設定する場合

```
{関数名|プログラム名(要求コード)} = リターン値 {,|;}
```

(b) イベントタイプを設定する場合

```
{tpsend|tprecv} = TPEEVENT,
{tpsend(event)|tprecv(event)} = イベントタイプ {,|;}
```

(c) 出力データを設定する場合

```
{マルチノード機能で使用する関数名(node_id)|マルチノード機能で使用する関数名(sv_name)} =
{ノード識別子|サーバ名}{,|;}
```

(2) 機能

OpenTP1 提供関数のリターン値として返す，任意の値を定義します。

また，XATMI 関数の tpsend 関数と tprecv 関数のイベントタイプや，TP1/Multi の関数に渡す出力データ（ノード識別子，サーバ名）の値を定義します。

(3) オペランド

(a) リターン値を設定する場合

- 関数名 | プログラム名(要求コード)

  リターン値を返す関数名，またはプログラム名(要求コード)を指定します。

  関数名

  C 言語の関数に対するリターン値を設定します。

  プログラム名(要求コード)

  COBOL 言語のプログラムに対するリターン値を設定します。( )の中には要求コードを設定します。

- リターン値　　～〈1～39 文字の英数字〉

  関数，またはプログラムによって返されるリターン値（COBOL 言語ではリターンコード）を設定します。リターン値は，定数名を大文字で設定します。TX 関数で COBOL 言語のリターンコードを設定する場合は，定数名でも設定します。

  また，リターン値は，定数名を使わないで数値（10 進数）でも設定できます。数値で指定する場合に指定できる範囲を，次に示します。

インタフェース	指定できる数値の範囲
C 言語インタフェース	-99999～99999
COBOL 言語インタフェース	0～99999※

注

範囲外の場合は，文字列と見なします。

注※

TX 関数については，-99999～99999 の範囲で指定できます。

なお，定義されていない定数名が指定された場合や，数値が誤って指定された場合（数字以外の指定など）は，関数が正常にリターンしたと見なします。

(b) イベントタイプを設定する場合

- {tpsend | tprecv}=TPEEVENT
  イベントタイプを設定することを示します。
- tpsend(event) | tprecv(event)
  イベントタイプを設定する関数を指定します。
  tpsend(event)：tpsend 関数にイベントタイプを設定します。
  tprecv(event)：tprecv 関数にイベントタイプを設定します。
- イベントタイプ
  tpsend 関数または tprecv 関数に設定する，イベントタイプを設定します。
  イベントタイプの設定を省略した場合は，TPEV_SVCERR を仮定します。

(c) 出力データを設定する場合

- マルチノード機能で使用する関数名(node_id) | マルチノード機能で使用する関数名(sv_name)
  出力データを設定する関数名と，出力データの種別を設定します。
  node_id：出力データとしてノード識別子を設定します。
  sv_name：出力データとしてサーバ名を設定します。
  設定できる関数名と，それに対応する出力データの種別を，次に示します。

関数名	出力データの種別
dc_adm_get_nd_status_next 関数	node_id
dc_adm_get_sv_status_next 関数	sv_name
dc_adm_get_nodeconf_next 関数	node_id
dc_adm_get_node_id 関数	node_id

- ノード識別子 | サーバ名
  設定するノード識別子，またはサーバ名を設定します。
  関数リターン値ファイル内のノード識別子とサーバ名の設定順序は，UAP で発行するマルチノード機能の関数の発行順序と対応します。
  設定を省略した場合や設定に誤りがあった場合，その行は無視されます。したがって，ノード識別子やサーバ名はないものとして処理され，その個数はカウントされません。
  関数リターン値ファイルに設定したノード識別子やサーバ名の個数より，UAP で発行した対応する関数の個数が多い場合，余分の関数のリターン値として DCADMER_NO_MORE_ENTRY が返されます。ただし，dc_adm_get_node_id 関数は除きます。

(4) 定義例

(a) C言語の場合

```
dc_jnl_ujput  = 0,
dc_dam_open   = DCDAMER_PROTO,
#dc_trn_begin = DC_OK,
dc_dam_read   = -1600,
tpsend        = TPEEVENT,
tpsend(event) = TPEV_DISCONIMM,
dc_adm_get_nd_status_next(node_id) = ND01
        :
        :
dc_logprint   = DC_OK;
```

(b) COBOL言語の場合

```
CBLDCJNL(UJPUT)  = 0,
#CBLDCTRN(BEGIN) = 905,
CBLDCDAM(READ)   = 1600,
        :
        :
CBLDCLOG(PRINT)  = 1905;
```

(5) 注意事項

- 定義解析では，関数とリターン値の正当性や，関係についてはチェックしません。
- 関数名（プログラム名）として，サポートしない関数，リターン値を返さない関数，およびオフライン環境でDAMファイルにアクセスする関数の名称を指定した場合，書式不正と見なされます。
- 重複して指定（同じ名称の関数の指定や，同じ機能の関数とプログラムの指定），できません。重複して指定した場合は，先に指定したりターン値が設定されたりします。
- 定義解析中に書式エラーを検出した場合は，エラーメッセージを出力して解析を続行します。書式エラーのあった定義内容の有効・無効の判断は，次のようになります。

書式エラーの内容	定義内容	有効時の仮定
文末に','または';'なし	有効	文末に','を仮定
その他	無効	－

（凡例）

－：該当しません。

- 要求コードには，マニュアル「OpenTP1 プログラム作成リファレンス COBOL 言語編」に記載されている要求コードを指定します。ただし，次に示す機能については，要求コードを次のように設定してください。

機能	要求コード
TAM テーブルのレコード削除	DELT
TAM レコードのレコード入力	READ
TAM レコードの更新/出力	WRIT

- 正常終了したときに状態コードを返す COBOL 言語プログラムのリターン値は，次のように設定してください。

  CBLDCADM(STATUS)

  ユーザサーバの状態コードをリターン値として設定します。

  CBLDCTAM(GST)

  次に示す値を設定します。

  1：RO（オープン状態）を返す場合

  2：RC（クローズ状態）を返す場合

  3：HL（論理閉塞状態）を返す場合

  4：HO（障害閉塞状態）を返す場合

  （例）CBLDCADM(STATUS)=1　と設定した場合，次を返します。

  リターン値=0

  ユーザサーバ状態コード=1

  CBLDCTAM(GST)=3　と設定した場合，次を返します。

  リターン値=0

  TAM テーブル状態=HL（論理閉塞状態）

11.1.4　連続実行コマンドの設定

決めておいた順序でコマンドを連続して実行する場合，あらかじめ連続実行コマンド定義ファイルを作成し，実行したい順序でコマンドを設定しておきます。途中に end サブコマンドを指定した場合は，オフラインテスタは終了し，以降のコマンドは実行しません。

連続実行コマンド定義ファイルには，サービス実行中にオフラインテスタからの問い合わせに対して応答するコマンド（read サブコマンドなど）も指定できます。

指定がなければ入力待ちになります。

定義方法や記述形式は，オフラインテスタ環境定義と同じです。

(1)　形式

```
コマンド名　〔コマンド引数 … 〕　{,|;}
```

(2) 機能

オフラインテスタに連続して実行させるコマンドを定義します。

(3) オペランド

- コマンド名
  コマンド名として指定できるのは，次の値です。
  - call
  - end
  - ps
  - read
  - start
  - stop
  - write

  そのほかのコマンドを指定した場合は，コマンドエラーのメッセージが出力され，コマンドを無視して処理が続行されます。
- コマンド引数
  指定したコマンドに対する，コマンド引数を指定します。

(4) 定義例

```
call  ser1 sppsub1 a_data,
call  ser2 mcfsub b_data+c_data,
call  ser3 sppsub2 d_data,
read  rtn_data,
#call ser1 sppsub1 b_data,
         :
         :
end;
```

(5) 注意事項

- 定義解析中に書式エラーを検出した場合は，エラーメッセージを出力して定義解析が続行されます。書式エラーのあった定義内容の有効・無効の判断は，次のようになります。

書式エラーの内容	定義内容	有効時の仮定
文末に','または';'なし	有効	文末に','を仮定
その他	無効	−

（凡例）

−：該当しません。

- コマンドは，実際に cmdauto サブコマンドが入力されたあとに，個々のコマンドを実行する時点でチェックされます。

11.1.5　スタブの生成

スタブは，RPC 環境や XATMI 環境，および TxRPC 環境でサービスを使用できる UAP（SPP と MHP）に必要です。

RPC および XATMI インタフェースの UAP の場合，スタブは RPC（XATMI）インタフェース定義を格納した，**RPC(XATMI)インタフェース定義ファイル**からスタブゼネレータで作成します。TxRPC インタフェースの UAP の場合，スタブやサーバ UAP のテンプレートなどは，**インタフェース定義言語ファイル**から OpenTP1 のコマンドで作成します。作成したスタブは C 言語のコンパイラで翻訳して，サーバ UAP のオブジェクトファイルに結合させます。

オフラインテスタで使用するスタブの作成方法は，実業務用の UAP の場合と同じです。スタブの作成方法については，マニュアル「OpenTP1 プログラム作成リファレンス C 言語編」を参照してください。

11.2　ユーザが作成するファイル

オフラインテスタを使用するために，ユーザが作成するファイルの一覧を次の表に示します。

表 11–2　ユーザが作成するファイルの一覧

ファイル種別		用途と内容	作成タイミング	削除者	削除タイミング
サービス要求データファイル	RPC 要求データファイル	RPC インタフェースのクライアント UAP シミュレート機能使用時，サービス要求先に渡すデータを格納します。	サービス要求前	ユーザ	任意
	XATMI 要求データファイル	XATMI インタフェースのクライアント UAP シミュレート機能使用時，サービス要求先に渡すデータを格納します。	サービス要求前	ユーザ	任意
	TxRPC 要求データファイル	TxRPC インタフェースのクライアント UAP シミュレート機能使用時，サービス要求先に渡すデータを格納します。	サービス要求前	ユーザ	任意
サービス応答データファイル	RPC 応答データファイル	RPC インタフェースのサーバ UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	シミュレート対象の SPP の起動時	ユーザ	任意
	XATMI 応答データファイル	XATMI インタフェースのサーバ UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	シミュレート対象の SPP の起動時	ユーザ	任意
	TxRPC 応答データファイル	TxRPC インタフェースのサーバ UAP シミュレート機能使用時，サービス結果として応答するデータを格納します。	シミュレート対象の SPP の起動時	ユーザ	任意
XATMI 受信データファイル		tprecv 関数で受信するデータを格納します。	サービス要求前	ユーザ	任意
MCF 受信メッセージファイル		MCF シミュレート機能使用時，MHP に渡すメッセージを格納します。	サービス要求前	ユーザ	任意
運用コマンド結果データファイル		運用コマンドシミュレート機能使用時，実行結果として UAP に返すデータを格納します。	サービス要求前	ユーザ	任意
ユーザファイル		DAM，TAM 機能使用時，ユーザファイルとして使用します。	オフラインテスタ起動前	ユーザ	任意

注

ユーザが作成するファイルは，次のファイルを除いたすべてのファイルで，オンラインテスタのものが使用できます。

・XATMI 受信データファイル

・MCF 受信メッセージファイル（非同期型受信メッセージファイル）

・運用コマンド結果データファイル

11.2.1　サービス要求データファイル

(1)　RPC 要求データファイル

RPC インタフェースのクライアント UAP シミュレート機能使用時，サービスに対応したサービス関数に渡すデータを格納します。一つのファイルには，一つのデータを作成します。

(a)　ファイルの構造

(b)　ファイルの内容

項目	位置	長さ（バイト）	内容
データ長	0	4	サービス関数に渡すデータの長さを指定します （1～DCRPC_MAX_MESSAGE_SIZE の指定値）。
応答領域長	4	4	サービス関数に渡す応答領域長を指定します （1～DCRPC_MAX_MESSAGE_SIZE の指定値）。
データ	8	n	サービス関数に渡すデータを指定します。

(c)　注意事項

- サービス関数の引数との関係を次に示します。

```
サービス関数　（in, in_len, out, out_len）
                1.  2.      3.
```

  1. データ
  2. データ長
  3. 応答領域長

- RPC 要求データファイルは，オンラインテスタのファイルも使用できます。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。

(2)　XATMI 要求データファイル

XATMI インタフェースのクライアント UAP シミュレート機能使用時，サービスに対応したサービス関数に渡すデータを格納します。一つのファイルには，一つのデータを作成します。

(a) ファイルの構造

(b) ファイルの内容

項目	位置	長さ（バイト）	内容
呼び出し種別	0	8	サービスを呼び出す際に，呼び出し元の関数の種別を指定します。 call：tpcall 関数からの呼び出し acall：tpacall 関数からの呼び出し connect：tpconnect 関数からの呼び出し
タイプ	8	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE
サブタイプ	16	16	サブタイプを 16 文字以下の文字列で指定します。ただし，タイプに'X_OCTET'を指定した場合は，サブタイプはヌル文字とします。
フラグ	32	4	サービス関数に渡す次のフラグを，16 進数で指定します。 0x00000000L：0 0x00000004L：TPNOREPLY 0x00000008L：TPNOTRAN 0x00000100L：TPNOCHANGE 0x00000800L：TPSENDONLY 0x00001000L：TPRECVONLY
データ長	36	4	サービス関数に渡すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	40	n	サービス関数に渡すデータを指定します。

(c) 注意事項

- サービス関数の引数との関係を次に示します。

```
void tpservice(svcinf)
     TPSVCINFO *svcinf;

     struct TPSVCINFO {
               char name[32];     ……1.
               char *data;        ……2.
               long len;          ……3.
               long flags;        ……4.
               int cd;            ……5.
          }
```

1.サービス名

2. タイプ，サブタイプにマッピングされたデータが格納されているアドレス

3. data で示すデータの長さ

4. フラグ（指定したフラグをビット列で格納）

5. 会話記述子（0 を格納）

- XATMI 要求データファイルは，オンラインテスタのファイルも使用できます。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- 応答データ領域は，応答データ内のバッファタイプ，サブタイプに従って再度割り当てられます。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

(3) TxRPC 要求データファイル

TxRPC インタフェースのクライアント UAP シミュレート機能使用時，サービスに対応したサービス関数に渡すデータを格納します。一つのファイルには，一つのデータを作成します。

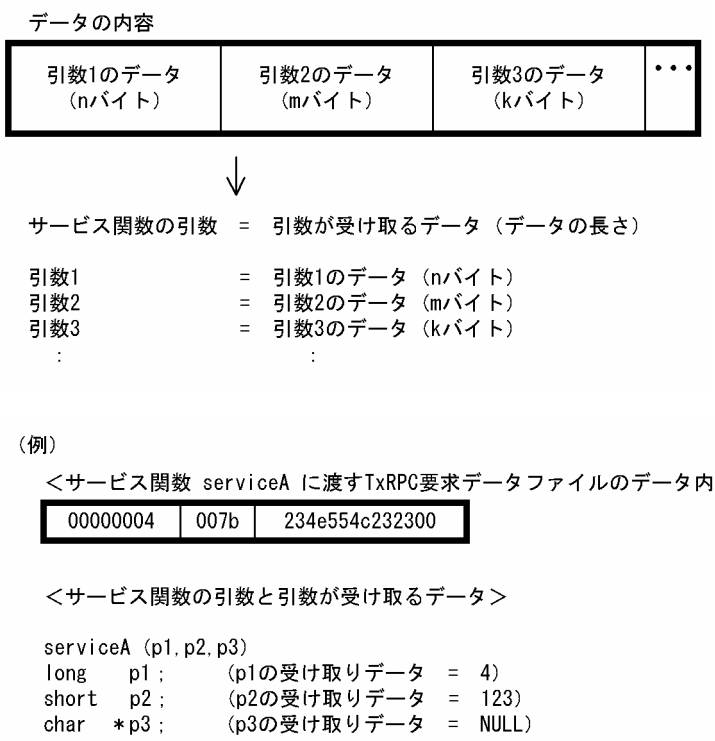
(a) ファイルの構造

(b) ファイルの内容

項目	位置	長さ（バイト）	内容
メジャーバージョン	0	2	txidl コマンドのインタフェース定義で指定した，メジャーバージョンの値を指定します。 省略する場合は，0 を指定します。
マイナーバージョン	2	2	txidl コマンドのインタフェース定義で指定したマイナーバージョンの値を指定します。 省略する場合は，0 を指定します。
データ長	4	4	データ部に指定するデータの長さを指定します （0～DCRPC_MAX_MESSAGE_SIZE の指定値-16）。
データ	8	n	サービス関数に渡す引数データを指定します。ただし，引数にアドレスを設定する場合は，アドレスの指す領域の内容そのものをデータに設定してください。 アドレスがヌルの場合は，文字列"#NULL##"をデータに設定してください。

(c) 注意事項

- TxRPC 要求データファイルのデータ内容と，サービス関数の引数と引数が受け取るデータの関係を次に示します。



- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- データ内容に誤りがある場合の UAP の動作は保証しません。

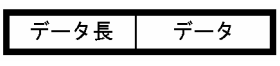
11.2.2 サービス応答データファイル

(1) RPC 応答データファイル

RPC インタフェースのサーバ UAP シミュレート機能使用時，シミュレート対象の SPP にサービスを要求した際に，サービスの要求元へ返す応答データを格納します。

一つのファイルには，一つのデータを作成します。

(a) ファイルの構造



(b) ファイルの内容

項目	位置	長さ（バイト）	内容
データ長	0	4	サービス要求元へ返すデータの長さを指定します（0～2147483647）。
データ	4	n	サービス要求元へ返すデータを指定します。

(c) 注意事項

- サービス要求元のサービス要求関数（dc_rpc_call 関数）の引数との関係を次に示します。

```
dc_rpc_call(……,in,in_len,out,out_len)
                          1.
```

1. データ

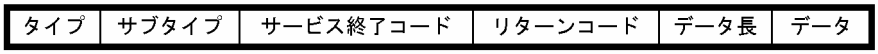
- RPC 応答データファイルは，オンラインテスタのファイルも使用できます。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。

(2) XATMI 応答データファイル

XATMI インタフェースのサーバ UAP シミュレート機能使用時，シミュレート対象の SPP にサービスを要求した際に，サービスの要求元へ返す応答データを格納します。

一つのファイルには，一つのデータを作成します。

(a) ファイルの構造



(b) ファイルの内容

項目	位置	長さ（バイト）	内容
タイプ	0	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE
サブタイプ	8	16	サブタイプを 16 文字以下の文字列で指定します。ただし，タイプに'X_OCTET'を指定した場合は，サブタイプはヌル文字とします。
サービス終了コード	24	4	tpreturn 関数の rval で指定する次のどちらかの値を，16 進数で指定します。この値は，tperrno 領域に設定されます。 0x04000000L：TPSUCCESS

項目	位置	長さ（バイト）	内容
サービス終了コード	24	4	0x20000000L：TPFAIL
リターンコード	28	4	tpreturn 関数の rcode で指定する値を 16 進数で指定します。この値は，tpurcode 領域に設定されます。
データ長	32	4	サービス要求元に返すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	36	n	サービス要求元に返すデータを指定します。

(c) 注意事項

- サービス終了関数（tpreturn 関数）の引数との関係を次に示します。

```
tpreturn(rval, rcode, data, len, ……)
          1.    2.     3.   4.
```

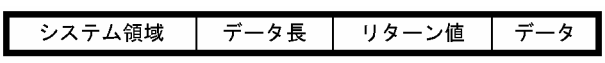
  1. サービス終了コード
  2. リターンコード
  3. タイプ，サブタイプで確保されたバッファに格納したデータ
  4. データ長

- XATMI 応答データファイルは，オンラインテスタのファイルも使用できます。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

(3) TxRPC 応答データファイル

TxRPC インタフェースのサーバ UAP シミュレート機能使用時，シミュレート対象の SPP にサービスを要求した際に，サービスの要求元へ返す応答データを格納します。一つのファイルには，一つのデータを作成します。

(a) ファイルの構造

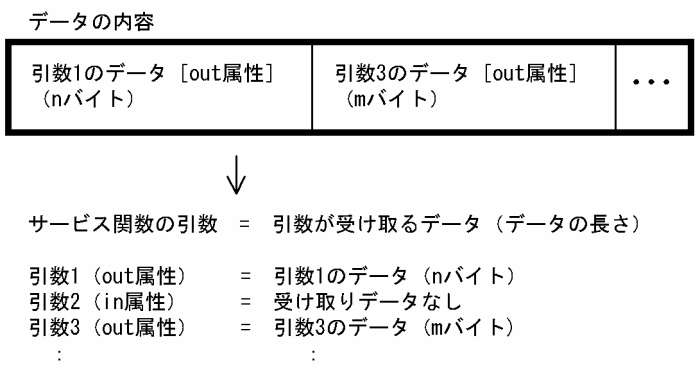


(b) ファイルの内容

項目	位置	長さ（バイト）	内容
システム領域	0	12	オフラインテスタが使用する領域です。 使用しないでください。
データ長	12	4	データ部に指定するデータの長さと，リターン値の長さの合計を指定します（0～DCRPC_MAX_MESSAGE_SIZE の指定値-16）。
リターン値	16	m	サービス関数のリターン値を指定します。データの型とサイズは txidl コマンドのインタフェース定義に指定した値となります。サービス関数が void 型の場合，リターン値は指定できません。
データ	16+m	n	クライアントに返す引数データを指定します。引数データには，txidl コマンドのインタフェース定義のパラメタ宣言に[out]属性で指定した引数だけを指定します。 引数にアドレスを設定する場合は，アドレスの指す領域の内容そのものをデータに設定してください。アドレスがヌルの場合は，文字列"#NULL##"をデータに設定してください。

(c) 注意事項

- TxRPC 応答データファイルのデータ内容と，サービス関数の引数と引数が受け取るデータの関係を次に示します。



（例）

＜サービス関数 serviceA が返すTxRPC応答データファイルのデータ内容＞

00000004	007b	234e554c232300

＜サービス関数の引数と引数が受け取るデータ＞

```
serviceA (p1,p2,p3,p4)
long    p1 ;   [out属性]   (p1の受け取りデータ  =  4)
short   p2 ;   [out属性]   (p2の受け取りデータ  =  123)
short   p3 ;   [in属性]    (p3の受け取りデータ  =  なし)
char   *p4 ;   [out属性]   (p4の受け取りデータ  =  NULL)
```

- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。

- データ内容に誤りがある場合の UAP の動作は保証しません。

11.2.3 XATMI 受信データファイル

tprecv 関数で，UAP が受け取るメッセージを格納します。一つのファイルには複数のデータを作成でき，順番に tprecv 関数へ渡します。

(1) ファイルの構造

(2) ファイルの内容

項目	位置	長さ（バイト）	内容
共用領域	0	36	XATMI 送信データファイルとの共用領域です。 空白，またはヌル文字を指定します。
タイプ	36	8	バッファタイプとして，次の文字列のどれかを指定します。 • X_OCTET • X_COMMON • X_C_TYPE
サブタイプ	44	16	サブタイプを 16 文字以下の文字列で指定します。 ただし，タイプに'X_OCTET'を指定した場合は，サブタイプはヌル文字とします。
イベントフラグ	60	4	tprecv 関数に渡す次の文字列のどれかを，16 進数で指定します。 0x00000000L：0 0x00000001L：TPEV_DISCONIMM 0x00000002L：TPEV_SVCERR 0x00000004L：TPEV_SVCFAIL 0x00000008L：TPEV_SVCSUCC 0x00000020L：TPEV_SENDONLY
データ長	64	4	tprecv 関数に渡すデータの長さを指定します（0～524288）。データがない場合は 0 を指定します。 0 を指定すると，タイプ，サブタイプの指定は無視されます。
データ	68	n	tprecv 関数に渡すデータを指定します。

(3) 注意事項

- メッセージ受信関数（tprecv 関数）の引数との関係を次に示します。

```
tprecv(……,data,len,……,revent)
          1.   2.        3.
```

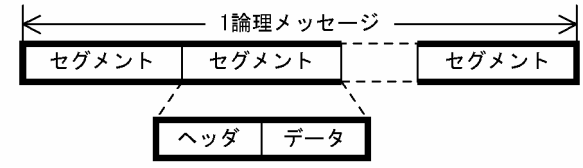
  1. タイプ，サブタイプで確保されたバッファに格納したデータ
  2. データ長
  3. イベントフラグ

- XATMI 受信データファイルは，オンラインテスタのファイルは使用できません。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- データ長に満たないデータを指定した場合は，エラーとなります。また，データ長を超えた部分のデータは無視されます。
- 受信データは実行単位に作成します。一つのサービスで複数回 tprecv 関数を発行する場合は，その数だけデータを作成しておく必要があります。データの数より tprecv 関数の発行回数が多い場合は，tpreturn 関数のデータが受信されたものとし，それ以降は，エラーとなります。

  XATMI 受信データファイルは，サービス単位にオープン，クローズします。
- タイプ，サブタイプを指定した場合のデータ長とデータは，スタブで定義したデータ構造（構造体）と同じでなければなりません。また，スタブで定義したデータ構造は，バウンダリ調整が行われます（合計長は 4 の整数倍となります）。そのため，ファイルで指定するデータはその調整部分を考慮して作成する必要があります。

11.2.4 MCF 受信メッセージファイル

1 論理メッセージは，一つまたは複数のセグメントで構成されています。1 セグメントはセグメント情報の入ったヘッダ部と，メッセージデータであるデータ部で構成されています。



セグメントには，次の五つの種別があります。

- 単セグメント

  1 論理メッセージが 1 セグメントで構成されている場合のセグメント
- 先頭セグメント

  1 論理メッセージが複数セグメントで構成されている場合の先頭セグメント
- 中間セグメント

1 論理メッセージが複数セグメントで構成されている場合の中間セグメント

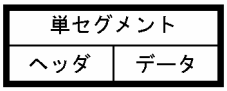
- 最終セグメント
  1 論理メッセージが複数セグメントで構成されている場合の最終セグメント
- ヘッダセグメント
  二つのメッセージを連結する場合に，先頭に付けるセグメント

なお，セグメントの種別は，ヘッダ部に指定します。

MCF 関数（dc_mcf_receive 関数，dc_mcf_recvsync 関数，または dc_mcf_sendrecv 関数）で，UAP が受け取るメッセージを格納します。一つのファイルには，一つの論理メッセージを作成します。ヘッダセグメントを使用すると，二つのメッセージを連結できます。

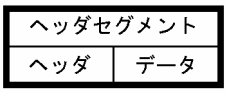
(1) ファイルの構造

● 1 論理メッセージ 1 セグメントのとき



● 1 論理メッセージ複数セグメントのとき

● ヘッダセグメントのとき



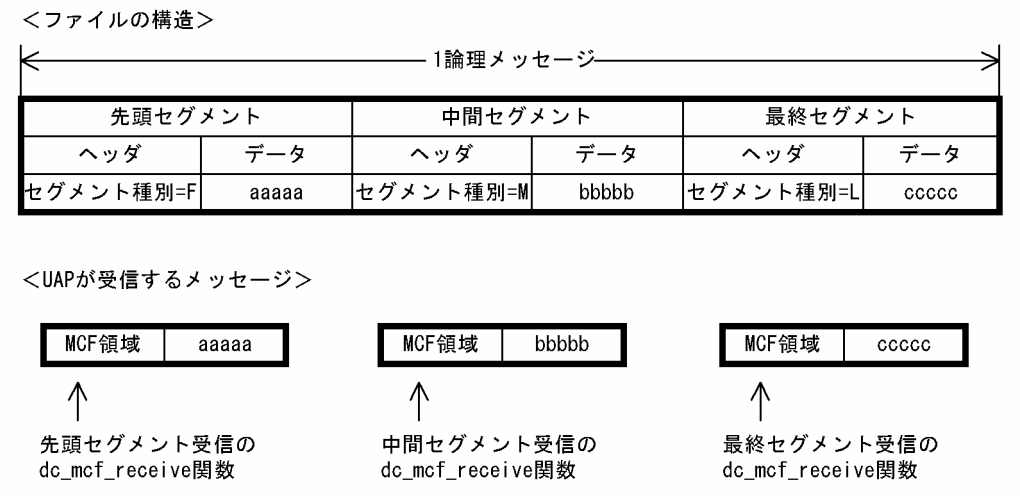
(2) ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	入出力論理端末名	0	9	MCF 関数でやり取りするための論理端末名を指定します（最後のヌル文字を含みます）。 複数セグメントのときは，各セグメントで同じものを指定します。
	マップ名称	9	9	マップ名称を指定します（最後のヌル文字を含みます）。複数セグメントのときは，各セグメントで同じものを指定します。 この指定はマップ名称を返す関数でだけ有効です。
	予備	18	9	ヌル文字を指定します。
	セグメント種別	27	1	次の文字のどれかを指定します。 F：先頭セグメントのとき M：中間セグメントのとき

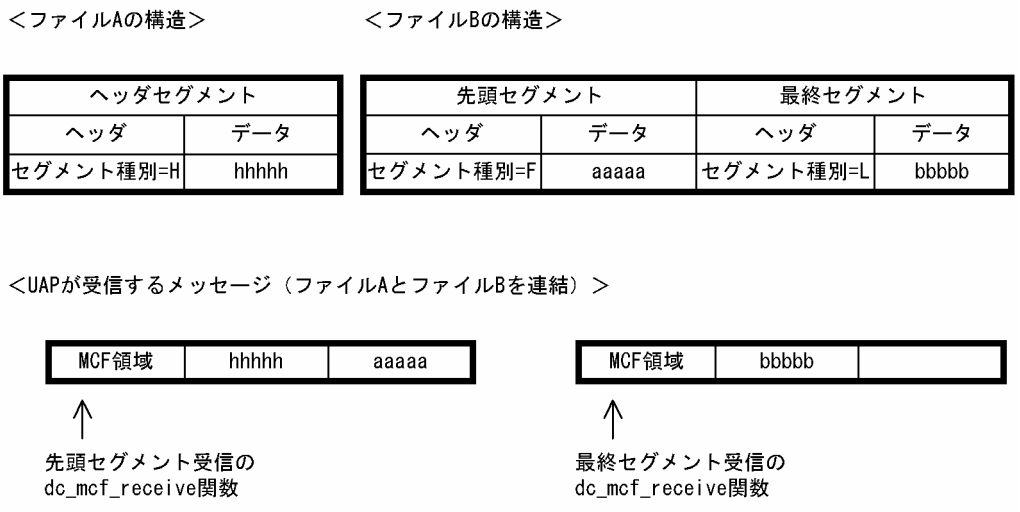
項目		位置	長さ（バイト）	内容
ヘッダ	セグメント種別	27	1	L：最終セグメントのとき O：単セグメントのとき H：ヘッダセグメントのとき
	メッセージ長	28	4	メッセージの長さを指定します（0～2147483647）。
データ	メッセージ	32	n	セグメントの実体（データ）を，メッセージ長で指定した長さで指定します。

(3) 注意事項

- MCF 受信メッセージファイルの構造と，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。



- ヘッダセグメントを連結すると，別ファイルに作成したデータを先頭セグメント，または単セグメントと合成して UAP に渡せます。ヘッダセグメントと，MCF 関数での UAP からのメッセージ受信要求との関係を次に示します。



- MHP へのサービス要求時にセグメントヘッダに指定したセグメント種別と，実行時のファイルの種別との関係を次に示します。

● 1 論理メッセージ 1 セグメントの場合

セグメント種別に F，M，および L を指定した場合には，O を指定した場合と同様に扱い，エラーとしません。

セグメント種別	ファイルの扱い
F	MCF 受信メッセージファイルとして扱います。
M	
L	
O	
H，X	ファイル不正として，ファイル名を問い合わせます。

（凡例）

X：F，M，L，O，および H 以外の指定

● 1 論理メッセージ複数セグメントの場合

セグメント種別に L，H，および O を指定したメッセージを受信すると，メッセージ完了と判断して以降のセグメントを無視します。セグメント種別に F を指定した場合と M を指定した場合は，同等に扱います。

セグメント種別			MHP が受信するセグメント
先頭セグメント	中間セグメント	最終セグメント	
F	M	L	F，M，L
F	L	M	F，L※1
F	O	L	F，O※2
M	M	L	M，M，L
L	M	F	L※3
O	O	O	O※3
F	L	M	F，L※1
F	M	H	受信しません※4。
X	M	L	
F	X	L	
F	M	X	
H	F	L	H※3
H	O	O	H※3

（凡例）

X：F，M，およびL以外の指定

注※1

Mは無視されます。

注※2

Lは無視されます。

注※3

中間セグメント以降は無視されます。

注※4

ファイル不正として，ファイル名を問い合わせます。

● ヘッダセグメントによるファイルの連結の場合

セグメント種別にHを指定した場合にだけ連結します。そのほかのセグメント種別の場合は，連結するファイルの指定は無視します。

セグメント種別（連結の組み合わせ）	ファイルの扱い
H＋（Fで始まるファイル）	連結したMCF受信メッセージファイルとして扱います。
H＋（Mで始まるファイル）	
H＋（Lで始まるファイル）	
H＋（Oで始まるファイル）	
H＋（Xで始まるファイル）	
F，M，L，およびO＋（不特定の種別で始まるファイル）	＋以降のファイルを無視します。
X＋（不特定の種別で始まるファイル）	ファイル不正として，ファイル名を問い合わせます。

（凡例）

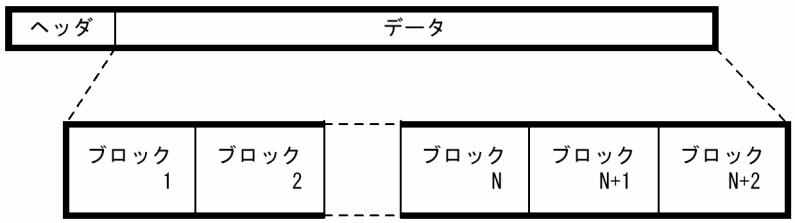
X：F，M，L，O，およびH以外の指定

ファイル名には'+'，空白，およびタブを使用できません。また，ファイル名として，'ps'または'end'は使用できません。

11.2.5 DAMファイル

DAMサービスシミュレート機能使用時，オフラインテスタ用のDAMファイルデータを格納します。DAMファイルは，エディタ，またはオフラインテスタが提供する関数（dc_dam_create関数）を使用したプログラムを作成・実行することで作成します。

(1) ファイルの構造



(2) ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	ファイル名	0	64	該当する DAM ファイルの名称を指定します。内容はチェックしません。
	ブロック長	64	4	1 ブロックの長さを指定します（0～32760）。
	総ブロック数	68	4	データ部にあるブロックの総数を指定します（1～2147483647）。
	未使用	72	2	ヌル文字を指定します
	閉塞状態	74	2	次のどれかを指定します。 0x0000：非閉塞状態（通常状態） 0x0001：論理閉塞状態 0x0002：障害閉塞状態
	予備	76	20	ヌル文字を指定します。
データ	ブロック	96	n	任意のデータをブロック単位で指定します。

(3) 注意事項

ヘッダ部の総ブロック数と実際のブロック数が等しいかどうかはチェックされません。実際のブロック数が少ないと，データのアクセス時にエラーになります。

11.2.6 TAM ファイル

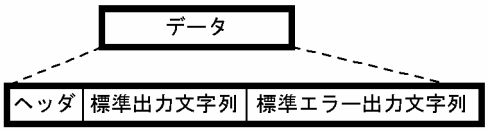
TAM サービスシミュレート機能使用時，オフラインテスタ用の TAM データを格納します。TAM ファイルは，TAM データファイルを基に，オフラインテスタのコマンド（utftamcre コマンド）で作成します。コマンドについては，「13.1 テストで使用する運用コマンド」を参照してください。

TAM データファイルは，TP1/FS/Table Access で使用する本来の TAM ファイルと同じ方法で作成します。また，実業務用の TAM ファイルをそのまま使用してもかまいません。実業務用の TAM ファイルの作成方法は，マニュアル「OpenTP1 運用と操作」を参照してください。

11.2.7　運用コマンド結果データファイル

運用コマンドシミュレート機能使用時に，コマンドの実行結果として UAP に返すデータを格納します。一つのファイルには，一つのデータを格納します。

(1)　ファイルの構造



(2)　ファイルの内容

項目		位置	長さ（バイト）	内容
ヘッダ	運用コマンドの結果コード	0	4	dc_adm_call_command 関数の引数 stat に設定する値を指定します。
	標準出力文字列長	4	4	標準出力文字列の長さ（ヌル文字を含みます）を指定します（0～2147483647）。
	標準エラー出力文字列長	8	4	標準エラー出力文字列の長さ（ヌル文字を含みます）を指定します（0～2147483647）。
標準出力文字列		12	n	dc_adm_call_command 関数の引数 outmsg に設定するデータを指定します（最後のヌル文字を含みます。ヌル文字が付けられていない場合は文字列の最後の文字をヌル文字に置き換えます）。 標準出力文字列長が 0 の場合は指定しても無視されます。
標準エラー出力文字列		−	n	dc_adm_call_command 関数の引数 errmsg に設定するデータを指定します（最後のヌル文字を含みます。ヌル文字が付けられていない場合は文字列の最後の文字をヌル文字に置き換えます）。 標準エラー出力文字列長が 0 の場合は指定しても無視されます。

（凡例）

−：該当しません。

(3)　注意事項

- 運用コマンド結果データファイルは，オンラインテスタのものは使用できません。
- 標準出力文字列と標準エラー出力文字列の最後には，ヌル文字を付けてください。ヌル文字が付いていない場合は，文字列の最後の文字がヌル文字に置き換えられます。また，文字列長に 0 を指定した場合は，文字列を指定しても無視されます。
- ファイル名には'+'を使用できません。また，ファイル名として，'ps'または'end'は使用できません。
- DML の SEND 文で運用コマンドを発行する場合は，データ部を次のように指定します。

標準出力文字列長

0 を指定します。

標準エラー出力文字列長

0 を指定します（標準エラー出力を受け取らない指定の場合）。

11.3　ファイルの作成

ここでは，テスタファイルを簡単に作成するためのテストデータ定義ファイルの作成，およびオフラインテスタが作成するファイルについて説明します。

11.3.1　テストデータ定義ファイルの作成

テストデータ定義ファイルを作成すると，テスタファイル作成機能で簡単にテスタファイルを作成できます。

テストデータ定義ファイルの名称は任意です。テストデータ定義ファイルから作成できるテスタファイルを次に示します。

- RPC 要求データファイル
- XATMI 要求データファイル
- TxRPC 要求データファイル
- RPC 応答データファイル
- XATMI 応答データファイル
- TxRPC 応答データファイル
- XATMI 受信データファイル
- MCF 受信メッセージファイル
- 運用コマンド結果データファイル

(1)　形式

```
#コメント          ……1.
start テスタファイル識別子 テスタファイル種別 出力先ファイル名 ……2.
キーワード = 入力データ          ……5.
キーワード = 入力データ
sep          ……3.
キーワード = 入力データ
     :              :
     :              :
キーワード = 入力データ
end          ……4.
```

なお，形式中の太字の数字は，「(3)　説明」の番号と対応しています。

(2) 機能

テスタファイルに設定するテストデータを定義すると，テスタファイル作成コマンドでテスタファイルを作成できます。

なお，定義ファイルでの 1 行の長さは，改行コードを含めて 512 バイトまでです。

(3) 説明

1.コメント文

コメントを記述します。

- **コメント**

  コメントを，1 行で記述します。

2.start 文

一つのテスタファイルの入力データの開始を宣言します。各テスタファイルの入力データごとに，必ず記述します。

一つのテストデータ定義ファイル内に複数のテスタファイルの入力データを作成した場合，1 テスタファイルの入力データの最後に end 文を記述します。

- **テスタファイル識別子　　～〈14 文字以内の英数字〉**

  テストデータ定義ファイル中に記述した各テスタファイルのデータを区別するための識別子を指定します。1 テストデータ定義ファイルで，一意の名称を指定します。

- **テスタファイル種別**

  テスタファイルの種別を指定します。指定できるテスタファイル種別を次に示します。

  RRQ：RPC 要求データファイル

  XRQ：XATMI 要求データファイル

  TRQ：TxRPC 要求データファイル

  RRT：RPC 応答データファイル

  XRT：XATMI 応答データファイル

  TRT：TxRPC 応答データファイル

  XRV：XATMI 受信データファイル

  NRV：MCF 受信メッセージファイル

  COM：運用コマンド結果データファイル

- **出力先ファイル名　　～〈パス名〉**

  入力データを基に作成する，テスタファイルの名称を指定します。

  一つのテストデータ定義ファイル内に，複数のテスタファイル種別の入力データを作成する場合は，ファイル種別ごとに異なった出力先ファイル名を指定します。

テスタファイル種別が異なる入力データで出力先ファイル名を重複して指定すると，指定したファイルはテストデータが追加された状態で作成されます。この場合，エラーにはなりませんが，作成されたテスタファイルがテストで使用できないことがあります。

また，すでにあるファイル名を指定した場合は，テストデータがそのファイルに追加された状態で作成されます。

3.sep 文

一つのファイルに複数のデータを記述するテスタファイルを作成する場合に，一つのデータの区切りを指定します。ただし，オフラインテスタの場合は，一つのファイルに複数のデータを記述しても先頭のデータだけが有効となり，二番目以降のデータは無視されます。

次のテスタファイルを作成する際に指定できます。

- XATMI 受信データファイル
- 運用コマンド結果データファイル

4.end 文

一つのテスタファイルの入力データの終了を宣言します。各テスタファイルの入力データごとに，必ず記述します。

5.入力データ定義文

各テスタファイルの入力データを定義します。

入力データには，あらかじめ設定できる情報が決められている**固定情報データ**と，ユーザが任意の情報を設定できる**ユーザデータ**（キーワードが data の場合）とがあります。一つのテストデータ内では，固定データはすべてユーザデータの前に記述してください。

なお，一つのテストデータ内では，入力データを重複して指定できません。ただし，運用コマンド結果データファイルでは，標準出力文字列データと標準エラー出力文字列データを設定するため，ユーザデータを 2 回指定します。

- キーワード

  各テスタファイル特有のデータを区別するための，キーワードを指定します。

  キーワードの前後の空白文字やタブコードは無視されます。

- 入力データ

  キーワードに対応する入力データを指定します。

  入力データの前後の空白文字やタブコードは無視されます。

  固定情報データを指定する場合の入力データの形式は，「(5)　テスタファイルのキーワードに対応する入力データの形式」の表を参照してください。

(4)　入力データにユーザデータを指定する場合に必要な設定

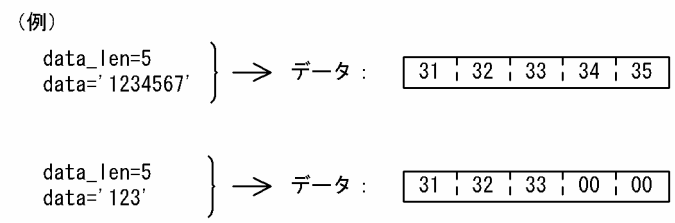
ユーザデータを指定する場合の，入力データの形式を次に示します。

(a)　ユーザデータ長の設定

ユーザデータ全体のデータ長は，固定情報データとして次の形式で設定します。

data_len=バイト数

ユーザデータとして設定したデータが，バイト数に設定したデータ長を超えた場合は，データを切り捨ててメッセージを出力します。データがデータ長に満たない場合は，それ以上のデータは設定しません。



(b) ユーザデータの初期化

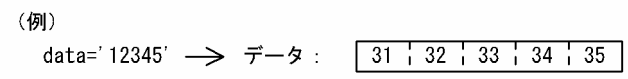
ユーザデータは，テスタファイル作成コマンドで，指定されたユーザデータ長分を初期化します。

(c) 文字データの設定

文字データは，次の形式で設定します。

data='データ'

文字データで記述した場合は，データの最後にヌル文字を付けません。



(d) バイナリデータの設定

バイナリデータは，次の形式で設定します。

data=データ

データは 10 進数と 16 進数で記述できます。記述方法を次に示します。

- **10 進数**：数値をそのまま設定します。
- **16 進数**：数値の前に'0x'を付けます。

(例)

```
data=5 → データ：10進数の5
data=0x05 → データ：16進数の5
```

データは，int 型で設定されます。

(e) 16 進コード形式データの設定

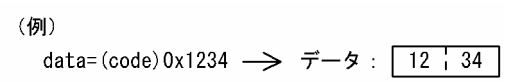
16 進コード形式のデータは，次の形式で設定します。

data=(code)0x データ

データの部分には，16 進コードで，n バイトのデータを 2n けたで記述します。1 行の最大長を超えないかぎり，記述できるデータのバイト数の制限はありません。

データ 1 バイトには，0x00 ～ 0xff の値を記述します。

(code)の指定がない場合，データはバイナリデータの 16 進数の記述と見なします。



(f)　特殊文字の設定

改行コード，タブコード，ヌル文字，' ' '（一重引用符）および'¥'は，文字データ中で特殊文字として扱います。これらの文字を記述する場合は，次の形式で記述してください。

記述する文字	記述形式
改行コード	¥n
タブコード	¥t
ヌル文字	¥0
' ' '（一重引用符）	¥'
'¥'	¥¥

(g)　ファイルからのデータ読み込みの設定

ファイルから読み込んだデータをユーザデータとして使用する場合は，次の形式で設定します。

data=(file)ファイルのパス名

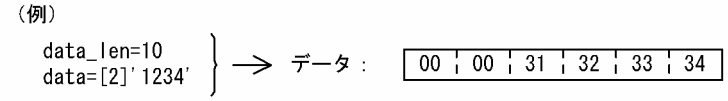
(例)

data=(file)/tmp/datafile → データ：/tmp/datafile のデータを設定

(h)　データの開始位置の設定

データを任意の位置から設定する場合は，次の形式で設定します。

data=［ユーザデータ先頭からのオフセット］データ



(i)　複数データ形式の設定

ユーザデータに複数のデータ型を使用する場合は，次の形式で設定します。

```
data=データ
    =データ
       :
       :
```

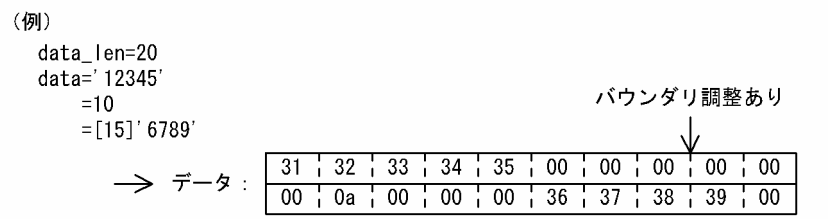
（例）

```
data=0x00000001    → 1番目のデータ
    ='ABCDEF'      → 2番目のデータ
```

(j)　バウンダリの調整

複数のデータ型の記述で前後のデータ型が異なる場合，テスタファイル作成コマンドで自動的に前のデータとのバウンダリを調整してデータを設定します。ただし，次の場合にはバウンダリを調整しません。

- ファイルからユーザデータを読み込む場合
- ユーザデータの開始位置を設定した場合
- 16 進コード形式でデータを設定した場合



(5)　テスタファイルのキーワードに対応する入力データの形式

各テスタファイルのキーワードと，それに対応する入力データの形式の一覧を，以降の表に示します。

指定する情報については，「11.2　ユーザが作成するファイル」の各テスタファイルの説明を参照してください。

表 11-3　RPC 要求データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
out_len	応答領域長	dc_rpc_call 関数に指定する応答領域長を，10 進数または 16 進数で設定します。data より前で設定します。
data_len	データ長	dc_rpc_call 関数でサーバ UAP に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	dc_rpc_call 関数でサーバ UAP に渡すユーザデータを設定します。

表 11-4　XATMI 要求データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
call_kind	呼び出し種別	サービス要求する関数種別として，次の文字列のどれかを設定します。

キーワード	指定する情報	説明
call_kind	呼び出し種別	• call • acall • connect data より前で設定します。
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET • X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
flag	フラグ	サービス関数に渡すフラグとして，次の文字列を設定します。複数のフラグを設定する場合は，'|'で区切ります。 • 0 • TPNOREPLY • TPNOTRAN • TONOCHANGE • TPSENDONLY • TPRECVONLY data より前で設定します。
data_len	データ長	tpcall 関数，tpacall 関数，または tpconnect 関数でサーバ UAP に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	tpcall 関数，tpacall 関数，または tpconnect 関数でサーバ UAP に渡すユーザデータを設定します。

表 11-5　TxRPC 要求データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
version	バージョン番号	txidl コマンドのインタフェース定義で指定したバージョン番号を 10 進数または 16 進数で設定します。data より前で設定します。バージョン番号は省略できます。省略時は，0 が設定されたと見なします。設定できる範囲は 0～65535 です。 （例） version=　　バージョンは，0.0 となります。 version=2　　バージョンは，2.0 となります。 version=3.2　　バージョンは，3.2 となります。
data_len	データ長	サーバ UAP に渡すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	サーバ UAP に渡すユーザデータを設定します。

表 11-6　RPC 応答データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
data_len	データ長	サービス終了時にクライアント UAP に返すユーザデータ長を，10 進数または 16 進数で設定します。data より前で設定します。
data	データ	サービス終了時にクライアント UAP に返すユーザデータを設定します。

表 11-7　XATMI 応答データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET • X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
rval	サービス終了コード	サービス終了コードとして，次の文字列のどれかを設定します。 • TPSUCCESS • TPFAIL data より前で設定します。
rcode	リターンコード	リターンコードを 10 進数または 16 進数で設定します。data より前で設定します。
data_len	データ長	サービス終了時にクライアント UAP に返すユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	データ	サービス終了時にクライアント UAP に返すユーザデータを設定します。

表 11-8　TxRPC 応答データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
data_len	データ長	クライアント UAP に渡すユーザデータ長を，10 進数または 16 進数で設定します。 data より前で設定します。
svc_rtn	リターン値	クライアント UAP に渡すリターン値を，10 進数または 16 進数で設定します。 data より前で設定します。
data	データ	クライアント UAP に渡すユーザデータを設定します。

表 11-9　XATMI 受信データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
buff_type	タイプ	バッファタイプとして，次の文字列のどれかを設定します。 • X_OCTET

キーワード	指定する情報	説明
buff_type	タイプ	• X_COMMON • X_C_TYPE data より前で設定します。
sub_type	サブタイプ	サブタイプを 16 文字以内の文字列で設定します。data より前で設定します。 （例）sub_type=subtype01
event	イベントフラグ	tprecv 関数に渡すイベントフラグとして，次の文字列のどれかを設定します。 • 0 • TPEV_DISCONIMM • TPEV_SVCERR • TPEV_SVCFAIL • TPEV_SVCSUCC • TPEV_SENDONLY data より前で設定します。
data_len	データ長	tprecv 関数に渡すユーザデータ長を 10 進数または 16 進数で設定します。 data より前で設定します。
data	データ	tprecv 関数に渡すユーザデータを設定します。
sep	sep 文	複数サービス分のデータを記述する場合に，1 サービス分のデータの最後に記述します。ただし，最終データのあとには記述しません。

注

複数サービス分のデータを設定する場合は，buff_type 以下のデータを繰り返し設定します。

表 11-10　MCF 受信メッセージファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
termname	入出力論理端末名称	dc_mcf_receive 関数に渡す入出力論理端末名称を，8 文字以内の文字列で設定します。data より前で設定します。
mapname	マップ名	dc_mcf_receive 関数に渡すマップ名を，8 文字以内の文字列で設定します。 data より前で設定します。
seg_kind	セグメント種別	dc_mcf_receive 関数に渡すセグメント種別として，次の文字のどれかを設定します。 • F • M • L • O • H 複数セグメント分のデータがある場合は，次のどれかの順序で設定します。 • F…M…L • F…F…L • M…M…L • L

キーワード	指定する情報	説明
seg_kind	セグメント種別	• H • O data より前で設定します。
data_len	メッセージ長	dc_mcf_receive 関数に渡すセグメントのユーザデータ長を 10 進数または 16 進数で設定します。data より前で設定します。
data	メッセージ	dc_mcf_receive 関数に渡すセグメントのユーザデータを設定します。

注

複数セグメント分のデータを設定する場合は，seg_kind 以下のデータを繰り返し設定します。

表 11–11　運用コマンド結果データファイルのキーワードと対応する入力データの形式

キーワード	指定する情報	説明
status_code	運用コマンドの結果コード	運用コマンドが返す結果コードを，10 進数で設定します。data より前で設定します。
outsize	標準出力文字列長	運用コマンドが標準出力に出力するメッセージ長を，10 進数または 16 進数で設定します。data より前で設定します。
errsize	標準エラー出力文字列長	運用コマンドが標準エラー出力に出力するメッセージ長を，10 進数または 16 進数で設定します。data より前で設定します。
data	標準出力文字列	運用コマンドが標準出力に出力するメッセージを，文字データで設定します。
data	標準エラー出力文字列	運用コマンドが標準エラー出力に出力するメッセージを，文字データで設定します。
sep	sep 文	複数コマンド分のデータを記述する場合に，1 コマンド分のデータの最後に記述します。ただし，最終データのあとには記述しません。

11.3.2　オフラインテスタが作成するファイル

オフラインテスタ使用時に，オフラインテスタが作成するファイルの一覧を次の表に示します。

表 11–12　オフラインテスタが作成するファイルの一覧

ファイル種別	用途と内容	作成タイミング	削除者	削除タイミング
XATMI 送信データファイル	tpsend 関数で送信するデータを格納します。	tpsend 関数発行時	ユーザ	任意
一時記憶データファイル	MCF シミュレート機能使用時，UAP の dc_mcf_tempput 関数で更新し dc_mcf_tempget 関数で受け取るデータを格納します。	左記の関数内※1	オフラインテスタ※2	dc_mcf_contend 関数発行時

ファイル種別	用途と内容	作成タイミング	削除者	削除タイミング
トレースファイル	オフラインテスタトレース情報を取得します。	オフラインテスタ（UAP）の最初のトレース取得時	ユーザ	任意

注※1

ディレクトリ/tmp 下に，dc_mcf_receive 関数で取得した論理端末名称をファイル名にして作成されます。ただし，ディレクトリ/tmp 下に同じ名称のファイルがある場合は，作成されません。

注※2

dc_mcf_contend 関数を発行する UAP を動作させない場合は，ユーザが任意のタイミングで削除します。

12 テストの実行

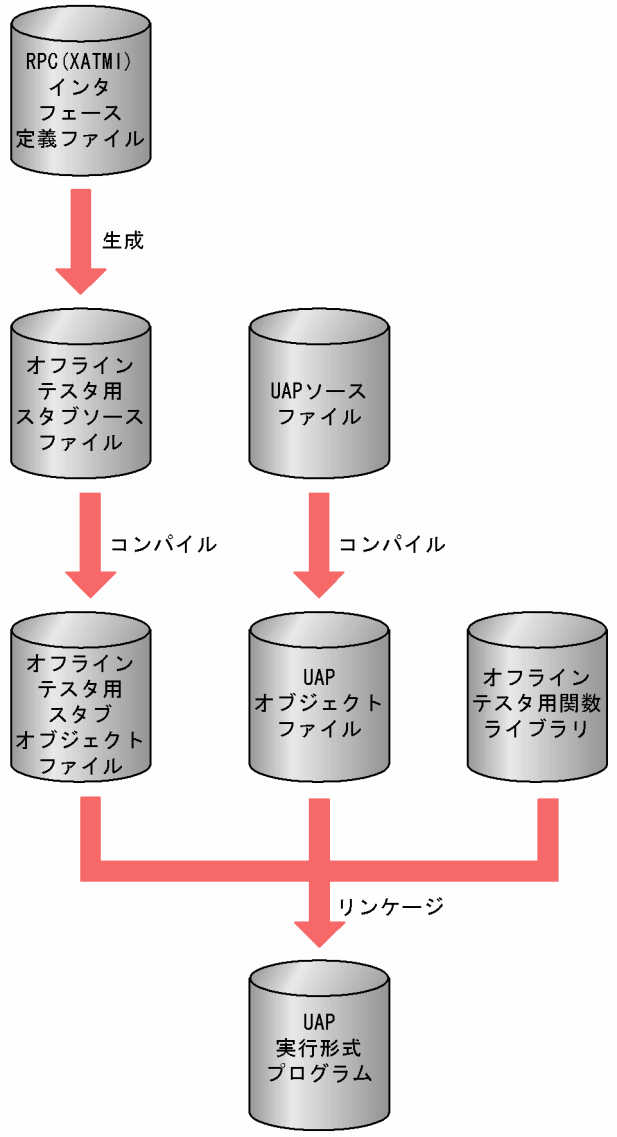
オフラインテスタでの，テストの実行方法について説明します。

12.1.1　UAP 実行形式プログラムの作成

(1)　RPC・XATMI インタフェースの UAP 実行形式プログラムの作成

RPC・XATMI インタフェースの UAP 実行形式プログラムの作成手順を次の図に示します。

図 12–1　RPC・XATMI インタフェースの UAP 実行形式プログラムの作成手順



RPC・XATMI インタフェースの UAP 実行形式プログラムの作成に必要なスタブのソースプログラムは，RPC（XATMI）インタフェース定義ファイルを使用して，OpenTP1 の stbmake コマンドで作成します。stbmake コマンドについては，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。

スタブ生成の例を次に示します。

（例）RPC インタフェース定義ファイルからのスタブ生成

```
stbmake spp1stb.def
            1.
```

1. RPC インタフェース定義ファイル名

この例の場合に生成されるソースファイル名は spp1stb_sstb.c となります。

（例）XATMI インタフェース定義ファイルからのスタブ生成

```
stbmake -x spp1stb.def
              1.
```

1. XATMI インタフェース定義ファイル名

この例の場合に生成されるソースファイル名は spp1stb_stbx.c，ヘッダファイル名は spp1stb_stbx.h となります。

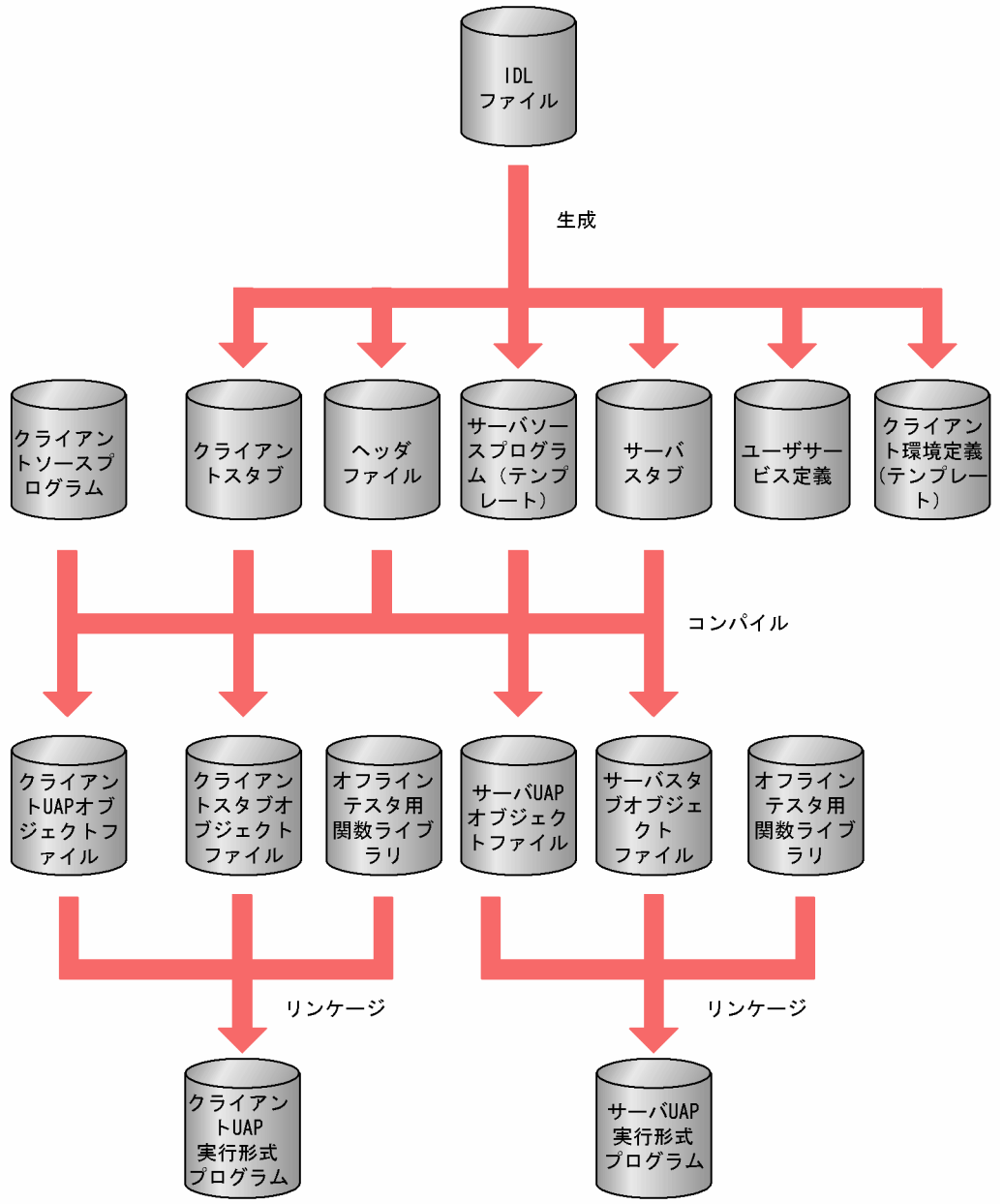
スタブの生成後，スタブと UAP（C 言語，または COBOL 言語）をコンパイルします。コンパイルに必要な OpenTP1 のヘッダファイルは，OpenTP1 本体が提供しているものを使用します。

コンパイル終了後，スタブのオブジェクトファイルと UAP のオブジェクトファイルを，オフラインテスタが提供するシミュレート関数ライブラリとリンケージします。

(2) TxRPC インタフェースの UAP 実行形式プログラムの作成

TxRPC インタフェースの UAP 実行形式プログラムの作成手順を次の図に示します。

図 12-2　TxRPC インタフェースの UAP 実行形式プログラムの作成手順



TxRPC インタフェースの UAP 実行形式プログラムの作成に必要な，クライアントスタブやサーバスタブのソースプログラム，およびサーバ UAP のテンプレートなどは，**インタフェース定義言語ファイル（IDL ファイル）**を使用して，OpenTP1 の **txidl** コマンドで作成します。txidl コマンドについては，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。

スタブおよびテンプレート生成の例を次に示します。

（例）IDL ファイルからのスタブ生成

```
txidl spp1.idl
          1.
```

1. インタフェース定義言語ファイル名

   この例の場合に生成されるファイルは次の六つとなります。

・spp1_cstub.c（クライアントスタブソース）
・spp1_sstub.c（サーバスタブソース）
・Cspp1（クライアント用ユーザサービス定義）
・Sspp1（サーバ用ユーザサービス定義）
・spp1.h（ヘッダファイル）
・spp1.c（サーバソースプログラムテンプレート）

ファイルの生成後，テンプレートを基に UAP をコーディングして，スタブと UAP（C 言語）をコンパイルします。コンパイルに必要な OpenTP1 のヘッダファイルは，OpenTP1 本体が提供しているものを使用します。UAP を作成する手順については，マニュアル「OpenTP1 プログラム作成リファレンス C 言語編」を参照してください。

コンパイル終了後，スタブのオブジェクトファイルと UAP のオブジェクトファイルを，オフラインテスタが提供するシミュレート関数ライブラリとリンケージします。この場合，クライアント UAP はクライアントスタブのオブジェクトファイルを，サーバ UAP はサーバスタブのオブジェクトファイルをリンケージしてください。

12.2　オフラインテスタの開始と終了

オフラインテスタは，utfstart コマンドで開始させます。utfstart コマンドには，オフラインテスタの実行条件を定義したオフラインテスタ環境定義ファイルの名称と，オプションパラメタを指定します。

オフラインテスタを起動すると，オフラインテスタ環境定義ファイルで指定したサービスグループが起動されます。すべての UAP が main 関数を実行してサービス開始関数（dc_rpc_mainloop 関数，または dc_mcf_mainloop 関数）を発行した時点で，コマンド入力を促すプロンプト'?>'が表示され，オフラインテスタのサブコマンドを入力できる状態になります。

オフラインテスタの起動時は，複数のサービスグループが一度に起動されるので，複数の UAP が並行に動作することがあります。

サブコマンドが入力できる状態で end サブコマンドを入力すると，オフラインテスタは終了します。

12.3 UAP の起動と停止

オフラインテスタは，OpenTP1 の代わりに UAP（サービスグループ）の起動・停止を制御しています。オフラインテスタ起動時には，オフラインテスタ環境定義で起動抑止を指定していないすべての UAP を起動します。また，オフラインテスタの起動完了後に，start サブコマンドでまだ起動していない UAP や，障害などで停止した UAP を起動できます。

オフラインテスタの終了時には，起動されているすべての UAP を停止させます。

また，stop サブコマンドで個別に UAP を停止できます。

12.4　サービスの要求

サービスを要求するには，次の二つの方法があります。

- dc_rpc_call 関数やそのほかの関数などによるプログラム内からのサービス要求
- call サブコマンド入力によるサービス要求

call サブコマンドでサービス要求する場合は，UAP の起動後（サービスグループの開始後）にコマンドを入力します。

12.5　テスタファイルの作成

テスタファイルの作成は，utffilcre コマンドで実行します。

テストデータ定義ファイルを使用してテスタファイルを作成する場合の手順は，オンラインテスタと同じです。作成手順については，「11.3.1　テストデータ定義ファイルの作成」を参照してください。

12.6　コマンドの連続実行

連続実行コマンド機能は，cmdauto サブコマンドを入力して実行します。cmdauto サブコマンドには，コマンド引数として連続実行コマンドファイルの名称を指定します。

連続実行コマンドファイルには，ユーザの応答待ち時に入力するコマンドも指定できます。なお，ファイル内のコマンドにエラーがあると，コマンドを無視するか，ユーザの応答待ちになります。

連続コマンド実行中に stop サブコマンドの実行以外で UAP プロセス（デバッガプロセス）が終了すると，連続実行コマンド機能を続行するか，中断するかの応答待ちになります。また，途中でサブコマンドの指定が足りなくなると，オフラインテスタからの問い合わせがあります。

12.7　デバッガの連動

オフラインテスタ環境定義でデバッガとの連動を指定しておくと，UAP はデバッガの制御下で動作します。テスト対象の UAP のソースファイルがあるディレクトリをデバッガに指示するなどの，デバッガへのパラメタが必要な場合は，あらかじめオフラインテスタ環境定義で指定しておきます。

デバッガとの連動は，UAP の main 関数から実行されます。ユーザは，デバッガに制御がわたって初期設定後，プログラム開始のコマンドを入力して実際にプログラムを開始させます。プログラムの実行が終了したあとは，デバッガを終了させます。なお，デバッガの再開始はできません。

デバッガとして使用できるのは次の二つです。

- dbx
- cbltd（COBOL2002）

デバッガの使用については，各デバッガの使用方法に従ってください。

12.8　オフラインテスタトレース情報の編集

オフラインテスタトレース情報は，オフラインテスタ起動時のオプションで，情報の出力可否や出力先，内容を選択してトレースファイルに取得します。トレースファイルに取得したトレース情報は，utftrcpic コマンドでサービス単位，サービスグループ単位に分けて出力できます。

トレースファイルのオープンなど，トレース情報を取得するための準備は，dc_rpc_open 関数内でされます。そのため，dc_rpc_open 関数より前で発行した関数のトレース情報は取得できません。また，DAM ファイルアクセス用の次のシミュレート関数のトレース情報も取得できません。

- dc_dam_create 関数
- dc_dam_get 関数
- dc_dam_iclose 関数
- dc_dam_iopen 関数
- dc_dam_put 関数

COBOL 言語の UAP で，要求コードや DML などの指定を誤った際に，API のトレース情報を出力しないことがあります。この場合，メッセージ KFCA20016-E，または KFCA20018-E が出力されます。また，DML の記述を誤った際に，エラー情報が出力されて（COBOL コンパイラが出力）異常終了することがあります。

main 関数の実行中など，UAP が並行して動作しているときのトレース情報は，トレース情報の行単位で混在している場合があります。これを防ぐためには，サービスグループの起動時点をずらす必要があります。

12.9　テスト実行中の注意事項

オフラインテスタでテストを実行する際の注意事項を示します。

12.9.1　オフラインテスタに関する注意事項

(1)　テスタプロセス異常終了時の後処理

オフラインテスタは，プロセス間を制御するために，パイプ機能および共用メモリ機能を使用しています。

割り込みキーを入力するなど，不測の事態でオフラインテスタを異常終了させた場合，使用中だった共用メモリ領域や一時ファイルが割り当てられたままになります。

そのままでもオフラインテスタを再度実行できますが，資源効率に問題がある場合は，ユーザが削除してください。

オフラインテスタが使用する一時ファイルに付けるファイル名は，次のとおりです。

- shmxxxx（/tmp ディレクトリの下に作成）
- cpixxxx（/tmp ディレクトリの下に作成）
- ppixxxx（/tmp ディレクトリの下に作成）
- tttttttt（/tmp ディレクトリの下に作成）
- aaaaaaaaxxxx（/tmp ディレクトリの下に作成）

(凡例)

xxxx
: 実行時のプロセス ID の 16 進表示です。

tttttttt
: dc_mcf_receive 関数で先頭セグメントを受信したときに返された論理端末名称と同じ名称です（最大 8 文字)。

aaaaaaaa
: オフラインテスタ環境定義ファイルで指定した IST テーブル名称と同じ名称です（最大 8 文字)。

(例)

- shm4e7
- cpi3e9
- ppi3e8
- termnalA

また，オフラインテスタの異常終了のタイミングによっては，UAP プロセスおよびデバッガプロセス（デバッガの連動指定時）が残ることがあります。この場合は，ユーザが kill コマンドでプロセスを終了させてください。

(2) オフラインテスタの上限値

オフラインテスタ使用時の上限値を，次の表に示します。

表 12-1 オフラインテスタの上限値

項目	項目の詳細	上限値	上限値を超えたときの処置
UAP 起動待ち時間	オフラインテスタ起動時または start サブコマンド実行時，UAP プロセスを生成してから UAP プロセスが起動を開始する（dc_rpc_open 関数を発行する）までの時間	60 分	エラーメッセージを出力後，プロセスを強制終了させます※1。
UAP 停止待ち時間	オフラインテスタ終了時または stop サブコマンド実行時，UAP プロセスに終了を要求してから終了するまでの時間。デバッガの連動を指定している UAP では，デバッガプロセスが終了するまでの時間	10 分	UAP プロセス，およびデバッガプロセスを強制終了させます。
コマンド行長さ	オフラインテスタのサブコマンドのコマンド行の長さ。または連続実行コマンドファイルの定義行の長さ	254 バイト	エラーメッセージを出力後，コマンドを拒否します。
定義行長さ	オフラインテスタ環境定義ファイルの定義行の長さ，または関数リターン値ファイルの定義行の長さ	510 バイト	エラーメッセージを出力後，その行を無視して定義解析を続行します※2。
パス名情報長さ	オフラインテスタ環境定義やコマンドで指定する，ディレクトリ名やパス名の長さ	255 バイト	エラーメッセージを出力後，その指定を無効にします。
関数リターン値の定義数	関数リターン値ファイル内の関数リターン値の定義数	200	エラーメッセージを出力後，以降の行を無視して処理を続行します。
DAM ファイル数	一つの UAP 内で，dc_dam_open 関数または dc_dam_create 関数でオープンした DAM ファイルの数	200	エラーメッセージを出力後，dc_dam_open 関数または dc_dam_create 関数をエラーリターンさせます。
TAM ファイル数	一つの UAP 内で，dc_tam_open 関数でオープンした TAM ファイルの数	200	エラーメッセージを出力後，dc_tam_open 関数をエラーリターンさせます。
dc_rpc_call 関数の発行数	一つの UAP 内で，dc_rpc_poll_any_replies 関数を発行しないで，DCRPC_NOWAIT 指定の dc_rpc_call 関数を発行した数	200	エラーメッセージを出力後，dc_rpc_call 関数をエラーリターンさせます。

項目	項目の詳細	上限値	上限値を超えたときの処置
同期型メッセージ送受信関数の発行数	一つのサービス内で，dc_mcf_sendrecv 関数，および dc_mcf_recvsync 関数を発行した数	100	エラーメッセージを出力後，dc_mcf_sendrecv 関数または dc_mcf_recvsync 関数をエラーリターンさせます。

注※1

デバッガの連動を指定している UAP は，対象外です。

注※2

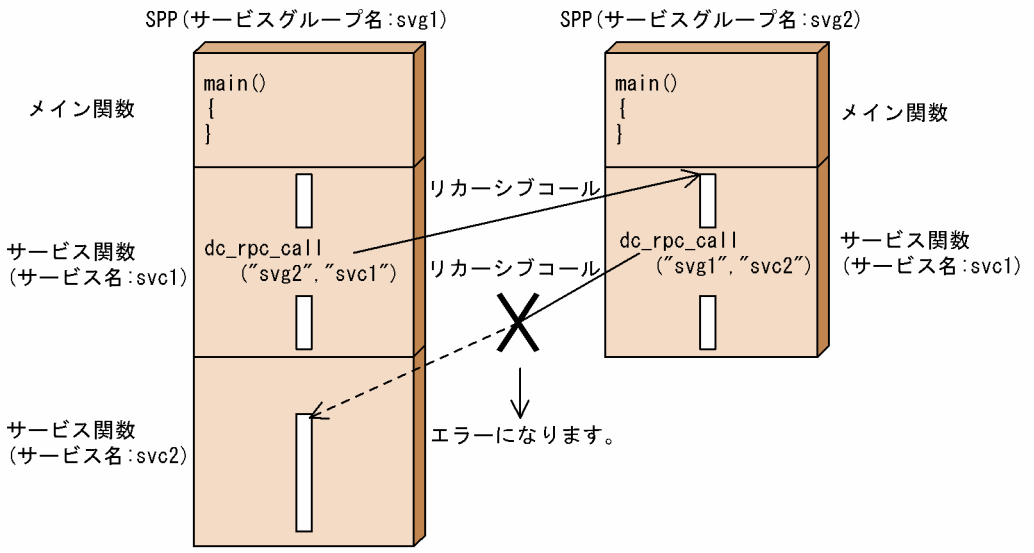
定義解析終了後，オフラインテスタの起動を続行するかどうかの応答待ちになります。

(3)　サービスグループのリカーシブコール

オフラインテスタでは，サービス関数内で dc_rpc_call 関数を発行して，サービスをネストして実行できます。ただし，別サービスであっても，同じサービスグループ内のサービスをネスト内で 2 回以上コールできません。

オフラインテスタでのリカーシブコールの例を，次の図に示します。

図 12-3　オフラインテスタでのリカーシブコール



(4)　サービスを受け付ける前後で発行できない関数

オフラインテスタでは，サービスを受け付ける前やサービスを受け付けたあと（dc_rpc_mainloop 関数，または dc_mcf_mainloop 関数の発行前かリターン後）に次の関数を発行すると，エラーメッセージが出力されてエラーリターンします。

- dc_rpc_call 関数
- dc_adm_call_command 関数
- dc_mcf_open，dc_mcf_close，dc_mcf_mainloop 以外の MCF 関数

(5) ユーザオウンコーディング関数

オフラインテスタでは，ユーザオウンコーディングに関する関数のうち，dc_mcf_svstart 関数はサポートしません。この関数を含む UAP をテストする場合，ユーザは同じ名称のダミーの関数を作成して，リンケージしてください。

(6) DAM ファイルアクセス関数での TAM テーブルへのアクセス

オフラインテスタでは，DAM ファイルアクセス関数で TAM テーブルにアクセスできません。アクセスした場合は，その動作は保証しません。

(7) トランザクション内外の処理

オフラインテスタでは，トランザクション内外を意識した処理はサポートしません。

(8) tpsend 関数でのイベントの受け取り

XATMI インタフェースの会話型のサービス要求では，tpsend 関数でイベントを受け取ることはできません。UAP のイベントを確認する場合は，関数リターン値ファイルを使用してください。

(9) IST テーブルへのアクセス

オフラインテスタの IST シミュレーション機能では，IST テーブルの内容を一時ファイルに格納して，参照や更新の処理をしています。このため，本来なら発生しないファイルアクセスエラーが発生する場合があります。

この場合は，エラーメッセージが出力され，ファイルアクセスエラーが発生した関数がエラーリターンします。リターン値は，その関数が返すエラーリターン値のどれかです。

12.9.2 ファイルに関する注意事項

(1) DAM ファイル・TAM ファイルの排他

DAM ファイルや TAM ファイルの排他は，ファイル単位で行います。このため，本来なら並行に動作してもデッドロックしない UAP 間（同じ DAM ファイルの別ブロックをブロック排他する UAP 間など）で，デッドロックが発生することがあります。

この場合，次のように処理して対処してください。

- オフラインテスタ環境定義の指定で排他を抑止します。
- utfstart コマンドの-c オプションの指定で更新を抑止します。
- オフラインテスタ開始後に start サブコマンドで UAP を順に起動して，UAP を並行動作させないようにします。

(2) DAM ファイルの一括処理ブロック数

オフラインテスタでは，dc_dam_create 関数および dc_dam_iopen 関数発行時に設定される一括処理ブロック数の値に関係なく，1 ブロック単位で処理します。ただし，0 より小さい値では処理されません。

(3) DAM ファイル・TAM ファイルのクローズ処理

dc_dam_open 関数または dc_tam_open 関数を発行したあとは，必ず dc_dam_close 関数または dc_tam_close 関数を発行してください。

dc_dam_close 関数または dc_tam_close 関数を発行しないままサービスグループが終了した場合，そのサービスを再度実行すると，DAM（TAM）ファイルの二重オープンエラーや，排他エラーが発生することがあります。

この場合は，stop サブコマンドで該当するサービス（グループ）を停止し，start サブコマンドで再度起動してください。

(4) COBOL UAP での TAM ファイルの排他

COBOL 言語の UAP では，TAM ファイルは排他しません。COBOL 言語を使用して UAP を作成する場合は，オフラインテスタ環境定義の TAM ファイルの定義で，排他の抑止を指定してください。排他の抑止を指定しないと，TAM ファイルにアクセスするサービスを再度実行した際に，排他エラーが発生することがあります。

この場合は，stop サブコマンドで該当するサービス（グループ）を停止し，start サブコマンドで再度起動してください。

12.9.3 UAP に関する注意事項

(1) UAP の無限ループ

オフラインテスタでは，タイマ監視をしていないため，無限ループなどによって UAP からの応答がなくなった場合，オフラインテスタの応答もなくなることがあります。

この場合は，ユーザが別ウィンドウで kill コマンドを入力し，UAP プロセスを強制停止させてください。

なお，kill コマンドで応答のなくなった UAP 以外のプロセスを強制停止させた場合の動作は保証しません。

13 運用コマンド

オフラインテスタの運用コマンド，サブコマンドの使用方法について説明します。

13.1　テストで使用する運用コマンド

テストで使用する運用コマンドの一覧を，次の表に示します。

表 13–1　テストで使用する運用コマンドの一覧

コマンド名	機能
utfdamcre	オフラインテスタ用 DAM ファイルを作成します。
utffilcre	テスタファイルを作成します。
utfstart	オフラインテスタを開始します。
utftamcre	オフラインテスタ用の TAM ファイルを作成します。
utftrcpic	ファイルからオフラインテスタトレース情報を取り出します。

13.1.1　utfdamcre（オフラインテスタ用 DAM ファイルの作成）

(1)　名称

オフラインテスタ用 DAM ファイルの作成

(2)　形式

```
utfdamcre ブロック長 ブロック数 DAMファイル名　〔入力ファイル名〕
```

(3)　機能

DAM データファイルを入力して，オフラインテスタ用の DAM ファイルを作成します。

(4)　コマンド引数

- ブロック長　　～((セクタ長 × n − 8))
  DAM ファイルのブロック長を指定します。
- ブロック数　　～((1～2147483647))
  作成する DAM ファイルのブロック数を指定します。DAM ファイルの大きさは，（ブロック長 × ブロック数 ＋ 96）バイトとなります。
- DAM ファイル名　　～〈パス名〉
  作成する DAM ファイル名を指定します。
- 入力ファイル名　　～〈パス名〉

DAM ファイルに出力するデータを格納したファイル名を指定します。この指定を省略した場合は，DAM ファイルにはヌルデータが出力されます。

(5) 注意事項

- utfdamcre コマンド実行時にエラーが発生した場合，DAM ファイルが割り当て状態のままとなります。そのため，utfdamcre コマンドを再度実行する前に，rm コマンドを使用して DAM ファイルを削除する必要があります。
- utfdamcre コマンドで指定したブロック数と入力ファイルのブロック数が異なる場合は，次のようになります。

  指定したブロック数　＞　入力ファイルのブロック数の場合

  DAM ファイルの最後までヌルデータのブロックを出力します。

  指定したブロック数　＜　入力ファイルのブロック数の場合

  入力ファイルからのブロック入力を中止して，KFCA20789-W のメッセージを出力して utfdamcre コマンドを終了します。

13.1.2 utffilcre（テスタファイルの作成）

(1) 名称

テスタファイルの作成

(2) 形式

```
utffilcre -e テストデータ定義ファイル名
```

(3) 機能

指定したテストデータ定義ファイルから，テスタファイルを作成します。

(4) オプション

- -e　テストデータ定義ファイル名　　～〈パス名〉

  作成するテスタファイルの入力データを定義した，テストデータ定義ファイルの名称を指定します。

13.1.3　utfstart（オフラインテスタの開始）

(1)　名称

オフラインテスタの開始

(2)　形式

```
utfstart〔-s〕〔-l〕〔-i〕〔-f〕〔-g〕〔-d〕〔-c〕オフラインテスタ環境定義ファイル名
```

(3)　機能

オフラインテスタ環境定義ファイルの内容に従って，オフラインテスタを開始します。

(4)　オプション

- -s

  サービス関数名とリターン情報を，オフラインテスタトレース情報として標準出力へ出力します。

  -i オプションと同時に指定した場合，この指定は無視されます。

- -l

  サービス関数名やリターン情報のほか，関数の引数情報をオフラインテスタトレース情報として標準出力へ出力します。

  -i オプションと同時に指定した場合，この指定は無視されます。

- -i

  オフラインテスタトレース情報の出力を抑止します。

- -f

  オフラインテスタトレース情報を，標準出力とトレースファイルへ出力します。トレースファイルがすでにある場合は，最後のデータの後ろに追加して出力します。トレースファイルがない場合は，オフラインテスタがトレースファイルを作成して出力します。

  -g オプションと同時に指定した場合，この指定は無視されます。

- -g

  オフラインテスタトレース情報を，標準出力とトレースファイルへ出力します。トレースファイルがすでにある場合は，トレースファイルを再度作成し，ファイルの先頭からトレース情報を出力します。トレースファイルがない場合は，オフラインテスタがトレースファイルを作成して出力します。

- -d

  関数の引数情報がデータ領域（バッファなど）のとき，データ領域の内容をすべて標準出力へ出力します。

  この指定を省略した場合は，20 バイト分だけを出力します。

  このオプションは，-l オプションを指定した場合にだけ有効です。

- -c
  DAM サービスと TAM サービスで，DAM ファイル，および TAM ファイルを更新しません。
  この指定を省略した場合は，DAM ファイル，および TAM ファイルを更新します。

(5) コマンド引数

- **オフラインテスタ環境定義ファイル名　～〈パス名〉**
  テスト環境を定義した，オフラインテスタ環境定義ファイルの名称を指定します。

(6) 注意事項

すべてのオプションを省略した場合は，-l を仮定します。

13.1.4 utftamcre（オフラインテスタ用 TAM ファイルの作成）

(1) 名称

オフラインテスタ用 TAM ファイルの作成

(2) 形式

```
utftamcre -r レコード長 -l キー領域長 -k キー開始位置 -m 最大レコード数
          〔-t〕〔-u ハッシュエントリ使用率〕〔-s〕
          〔-d TAMデータファイル名〕TAMファイル名
```

(3) 機能

TAM データファイルを入力して，オフラインテスタ用の TAM ファイルを作成します。

(4) オプション

- **-r　レコード長　～((1～2147483647))**
  TAM ファイルのレコード長を指定します。
- **-l　キー領域長　～((1～2147483647))**
  キーの長さを指定します。
- **-k　キー開始位置**
  レコードの先頭を 0 とした，キーの開始位置までのオフセットを指定します。
  -s オプションを指定した場合，-k オプションの指定が 0 以外の場合はエラーになります。また，TAM ファイルの管理部のレコード長は，**（レコード長 － キー領域長）**になります。
- **-m　最大レコード数　～((1～2147483647))**

TAM テーブル内の最大レコード数を指定します。

- -t
  TAM テーブルをツリー構造で作成します。
  省略時にはハッシュ構造で作成します。ただし，その場合は必ず-u オプションを指定してください。
- -u　ハッシュエントリ使用率　　～((1～100))
  ハッシュ域として使用するインデクスの使用率を指定します。
  -t オプションを指定した場合は，-u オプションを指定するとエラーになります。
- -s
  レコードの内容からキー領域を削除する場合に指定します。
- -d　TAM データファイル名　　～((255))
  TAM データファイルの名称を指定します。最大 255 文字まで指定でき，この文字数を超えるとエラーになります。また，TAM ファイル名と TAM データファイル名が同じ場合はエラーになります。
  なお，指定したファイル名が同じかどうかは，単純にファイル名を比較して判断されます。

(5)　コマンド引数

- TAM ファイル名　　～〈パス名〉
  このコマンドで作成する TAM ファイルの名称を指定します。

(6)　注意事項

- TAM データファイルのデータ長が（**レコード長 × 最大レコード数**）を超える場合は，エラーになります。
- TAM データファイルのデータ長が，-r オプションで指定したレコード長で割り切れない場合は，あまりのデータ長分のデータは切り捨てられ，TAM ファイルには格納されません。

13.1.5　utftrcpic（トレース情報の取り出し）

(1)　名称

トレース情報の取り出し

(2)　形式

```
utftrcpic トレースファイル名 サービスグループ名
        〔サービス名〔データファイル名〕〕
```

(3) 機能

オフラインテスタトレース情報を，キーを基にトレースファイルから取り出し，標準出力へ出力します。

(4) コマンド引数

- トレースファイル名　　～〈パス名〉
  オフラインテスタトレース情報を取得した，トレースファイルの名称を指定します。
- サービスグループ名　　～〈1～31 文字の識別子〉
  キー情報として，取り出したいトレース情報を含むサービスグループの名称を指定します。
- サービス名　　～〈1～31 文字の識別子〉
  キー情報として，取り出したいトレース情報を含むサービスの名称を指定します。
  この指定を省略した場合は，トレース情報はサービスグループ単位で取り出します。
- データファイル名　　～〈パス名〉
  キー情報として，サービス実行時のデータファイル名を特定したい場合に，そのデータファイルの名称を指定します。

(5) 出力形式

```
18:41:01 関数=dc_dam_read[ CBLDCDAM(READ) ]                                  ] 1.
         ファイル記述子(IN)=00000008                                          ]
         DAMキーの内容(IN)=先頭の相対ブロック番号 最後の相対ブロック番号      ] 2.
                           00000000                00000000                   ]
         DAMキーの数(IN)=00000001                                             ]
         入力データ(OUT)=                                                     ]
           00000000 00000000 00000000 00000000 00000000                       ] 3.
           00000000 00000000 00000000 00000000 00000000                       ]
         入力バッファ長(IN)=000001f8                                          ]
         オプションフラグ(IN)=00000009                                        ]
                              DCDAM_REFERENCE[ 入力要求種別:R ]               ] 4.
                              DCDAM_NOEXCLUSIVE[ 排他要求種別:N ]             ]
         戻り値=DC_OK(000000)[ 00000 ]                                        ] 5.
```

1. 時刻と関数の情報
   - サービスグループを起動した時刻（時：分：秒）
   - C 言語の関数名
   - COBOL 言語の機能名
   - 要求コード
   - DML 文名
2. 引数の情報
   (IN)は，UAP が関数の引数で指定した内容の表示を，(OUT)は，関数が UAP に返す内容の表示を意味します。文字列領域のアドレスがヌル文字の場合は，'引数名(OUT)=NULL'を表示します。

3. データとデータ長の情報

   データの内容を表示する場合，データ長分だけを 1 行 40 バイトで表示します。

   異常時の形式について，次に示します。

   (例)

   データのアドレスがヌル文字の場合

   データ名(IN)=NULL

   データ長が 0 の場合

   データ名(IN)=

4. オプションフラグの情報

   - オプションフラグ名
   - COBOL 言語フラグ名
   - COBOL 言語フラグ種別

   指定を誤っていた場合は，誤ったフラグのコードがそのまま表示されます。

   COBOL 言語のフラグ名とフラグ種別は'[***]'と表示されます。

   表示例を次に示します。

   (例)

   オプションフラグ(IN)=00000001

   DCDAM_FILE_EXCLUSIVE［排他種別：B］

   00000006[***]

5. リターン値の情報

   - C 言語リターン値定義名
   - C 言語リターン値 10 進数表示
   - COBOL 言語リターン値 10 進数表示

（出力例）

```
KFCA20001-I  プロセスを生成しました。サービスグループ名=svg2（xdb）

15:18:56  関数＝dc_rpc_open(svg2) [CBLDCRPC(OPEN)]
          オプションフラグ( IN )=00000000
                                 DCNOFLAGS

KFCA20000-I  オフラインテスタを起動しました。Tue  May  31  15:18:56  1994
             戻り値=DC_OK(000000)   [00000]

15:18:56  関数=dc_rpc_mainloop(svg2)   [CBLDCRSV(MAINLOOP)]
          オプションフラグ(IN)=00000000
                               DCNOFLAGS

?>call    svg2  svc5  xd03km

15:19:18  サービス開始(svc5)
            バッファタイプ(IN)=X_OCTET
            バッファ長(IN)=00000000
            データ(IN)=NULL
            オプションフラグ(IN)=00000000
                                 DCNOFLAGS
```

1.

2.

3.

（続く）

（続き）

```
15:19:18  関数=tpalloc                                                    ┐
          バッファタイプ(IN)=X_OCTET                                      │
          バッファサブタイプ(IN)=NULL                                     │
          バッファ長(IN)=0000008c                                         │
          戻り値=DC_OK(ADDRESS)                                           │
                                                                          │
                                                                          │
15:19:18  関数=tprealloc                                                  │
          バッファ長(IN)=00000096                                         │
          戻り値=DC_OK(ADDRESS)                                           │
                                                                          │
15:19:18  関数=tpcall                                                     │
          サービス名(IN)=svc1                                             │
          送信バッファタイプ(IN)=X_OCTET                                  │
          送信バッファ長(IN)=0000008c                                     │
          データ(IN)=                                                     │
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  00000000  00000000  00000000             │ 3.
            00000000  00000000  00000000  00000000  00000000             │
            00000000  00000000  0000                                     │
          応答バッファタイプ(IN)=X_OCTET                                  │
          応答バッファ長(IN)=0000008c                                     │
          オプションフラグ(IN)=00000008                                   │
                               TPNOTRAN                                   │
                                                                          │
read(svg2:svc5:crm_rtn)?>read  xd04km                                     ┘ 4.
          応答バッファタイプ(OUT)=X_OCTET
          応答バッファ長(OUT)=00000000
          データ(OUT)=
          サービス関数の戻り値(OUT)=999999
          戻り値=DC_OK(000000)

15:19:27  関数=tpreturn                                                   ┐
          終了コード(IN)=TPSUCCESS                                        │
          リターン値(IN)=000000                                           │
          バッファ長(IN)=00000000                                         │
          データ(IN)=NULL                                                 │
          オプションフラグ(IN)=00000000                                   │
                               DCNOFLAGS                                  │
15:19:27  サービス終了(svc5)                                              ┘
                                                                          ┐ 5.
?>end                                                                     ┘

15:19:31  戻り値(svg2)=DC_OK(000000)    [00000]                           ┐
                                                                          │
15:19:31  関数=dc_rpc_close(svg2) [CBLDCRPC(CLOSE)]                       │ 6.
          オプションフラグ(IN)=00000000                                   │
                               DCNOFLAGS                                  ┘
```

1. オフラインテスタ開始時のメッセージと，SPP 起動時に取得したトレース情報
2. サブコマンド入力時に取得したトレース情報（テスト開始）
3. サービス実行時に取得したトレース情報
4. テスタファイル読み込み時に取得したトレース情報
5. サブコマンド入力時に取得したトレース情報（テスト終了）
6. オフラインテスタ終了時，SPP 停止時に取得したトレース情報

(6) 注意事項

- オフラインテスタトレース情報を取り出す範囲は，サービスの開始から終了までです。
- サービスの開始から終了までの間に，read または write サブコマンドの入力待ちかファイル名の入力待ちが発生した場合は，その応答内容によってトレース情報の取り出し範囲が変わります。

  応答内容と取り出し範囲の関係を次に示します。

入力待ち種別	応答内容	トレース情報の取り出し範囲
read または write サブコマンドの入力待ち	read または write サブコマンドを入力	コマンド入力以降のトレース情報も取り出します。
	ps サブコマンドを入力	コマンド入力情報とコマンド実行結果は取り出しません。
	end サブコマンドを入力	コマンド入力以降のトレース情報は取り出しません。
	不正なコマンドを入力	コマンド入力情報とエラーメッセージは取り出しません。
ファイル名の入力待ち	ps サブコマンドを入力	コマンド入力情報とコマンド実行結果は取り出しません。
	end サブコマンドを入力	コマンド入力以降のトレース情報は取り出しません。
	ps または end サブコマンド以外を入力	コマンド入力以降のトレース情報も取り出します。

13.2　テストで使用するサブコマンド

テストで使用するサブコマンドの一覧を，次の表に示します。

表 13–2　テストで使用するサブコマンドの一覧

コマンド名	機能
call	サービスを要求します。
cmdauto	コマンドを連続実行します。
end	オフラインテスタを終了します。
ps	テスト状況を表示します。
read	オフラインテスタにテスタファイル名を連絡します。
start	サービスグループを起動します。
stop	サービスグループを停止します。
write	オフラインテスタにテスタファイル名を連絡します。

13.2.1　call（サービスの要求）

(1)　名称

サービスの要求

(2)　形式

```
call サービスグループ名 サービス名{RPC要求データファイル名|
                                XATMI要求データファイル名|
                                TxRPC要求データファイル名|
                                MCF受信メッセージファイル名
                                 〔 + MCF受信メッセージファイル名〕}
```

(3)　機能

サービスグループ名に対応した SPP，または MHP プロセスを起動し，サービス名に対応したサービス関数を実行します。

(4)　コマンド引数

- サービスグループ名　　～〈1～31 文字の識別子〉
  起動するサービスを含むサービスグループの名称を指定します。
- サービス名　　～〈1～31 文字の識別子〉

起動するサービスの名称を指定します。

- RPC 要求データファイル名　　～〈パス名〉

  RPC インタフェースの SPP へサービス要求する場合，最初のサービス関数で受け取るデータを格納した RPC 要求データファイルの名称を指定します。

- XATMI 要求データファイル名　　～〈パス名〉

  XATMI インタフェースの SPP へサービス要求する場合，最初のサービス関数で受け取るデータを格納した XATMI 要求データファイルの名称を指定します。

- TxRPC 要求データファイル名　　～〈パス名〉

  TxRPC インタフェースの SPP へサービス要求する場合，最初のサービス関数で受け取るデータを格納した TxRPC 要求データファイルの名称を指定します。

- MCF 受信メッセージファイル名　　～〈パス名〉

  MHP へサービス要求する場合，UAP が MCF 関数で受け取るデータを格納した MCF 受信メッセージファイルの名称を指定します。

  連結構成にする場合は，連結する MCF 受信メッセージファイルの名称を'+'で続けて指定します。

(5) 注意事項

- サービスグループ名はオフラインテスタ環境定義で，サービス名はユーザサービス定義で定義しておく必要があります。
- 各テスタファイルにアクセスできない場合や，ファイルの内容が誤っている場合，次のプロンプトが表示され，ファイル名の入力待ち状態になります。MCF 受信メッセージファイルの連結指定時に，片方のファイルだけでエラーが発生した場合は，エラーの発生したファイル名だけでなく，連結指定でのファイル名を指定してください。

  〈形式〉

```
file(group1:service1)?>
       1.      2.
```

  1. サービスグループ名
  2. サービス名

13.2.2 cmdauto（コマンドの連続実行）

(1) 名称

コマンドの連続実行

(2) 形式

```
cmdauto 連続実行コマンドファイル名
```

(3) 機能

連続実行コマンドファイルの内容に従って，コマンドを連続して実行します。

(4) コマンド引数

- 連続実行コマンドファイル名　　～〈パス名〉
  連続して実行するコマンドを記述した，連続実行コマンドファイルの名称を指定します。

13.2.3 end（オフラインテスタの終了）

(1) 名称

オフラインテスタの終了

(2) 形式

```
end
```

(3) 機能

起動されているサービスグループを停止させ，オフラインテスタを終了します。

(4) 注意事項

このコマンドは，各サービスグループの dc_rpc_mainloop 関数を正常リターンさせます。10 分経過しても UAP プロセス（デバッガ連動指定時はデバッガプロセス）が終了しない場合，UAP プロセス（デバッガプロセス）を強制終了します。ただし，read サブコマンド，またはファイル名の入力待ち状態でこのコマンドを入力すると，そのサービスグループは，オフラインテスタが dc_rpc_close 関数を発行してから，正常終了します。

13.2.4 ps（テスト状況の表示）

(1) 名称

テスト状況の表示

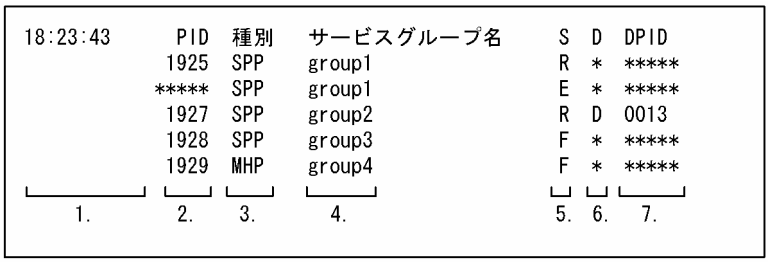
(2) 形式

```
ps
```

(3) 機能

オフラインテスタ下のプロセスの状態を表示します。

(4) 出力形式



1. ps サブコマンドを実行した時刻（時：分：秒）
2. UAP プロセス ID
   プロセスが停止状態のときは'*****'を表示します。
3. サービスグループの種別コード
   - SPP：SPP を示します。
   - MHP：MHP を示します。
4. サービスグループ名
5. プロセスの状態
   - R：このサービスグループのプロセスが起動状態であることを示します。
   - E：このサービスグループのプロセスが停止状態であることを示します。
   - F：サーバ UAP シミュレート機能で，シミュレート対象として指定（起動・停止の対象外）していることを示します。
6. デバッガ連動指定の有無
   - D：デバッガの連動を指定していることを示します。
   - ＊：デバッガの連動を指定していないことを示します。
7. デバッガのプロセス ID
   プロセスが停止状態のときは'*****'を表示します。

13.2.5　read（テスタファイル名の連絡）

(1)　名称

テスタファイル名の連絡

(2)　形式

```
read テスタファイル名〔 + MCF受信メッセージファイル名〕
```

(3)　機能

各種シミュレート機能で必要になったテスタファイル名を，オフラインテスタに連絡します。

(4)　コマンド引数

- テスタファイル名　　～〈パス名〉

  オフラインテスタに連絡する，テスタファイルの名称を指定します。

  どのテスタファイル名の連絡が必要なのかは，次のプロンプト中に表示されます。

  〈形式〉

  ```
  read(group1:service1:rpc_rtn)?>
       1.     2.      3.
  ```

  1. サービスグループ名
  2. サービス名（サービス外では表示しません）
  3. テスタファイルの種別
     - rpc_rtn：サービス応答データファイル
     - crm_rtn：XATMI 応答データファイル
     - trp_trn：TxRPC 応答データファイル
     - crm_rcv：XATMI 受信データファイル
     - mcf_msg：MCF 受信メッセージファイル
     - adm_cmd：運用コマンド結果データファイル

- MCF 受信メッセージファイル名　　～〈パス名〉

  テスタファイル名を MCF 受信メッセージファイル名で連結構成にする場合に，'+'に続けて連結する MCF 受信メッセージファイルの名称を指定します。

13.2.6　start（サービスグループの起動）

(1)　名称

サービスグループの起動

(2)　形式

```
start {SPP|MHP} サービスグループ名
```

(3)　機能

次のような UAP を再起動させます。

- オフラインテスタ開始時にサービスグループ起動の抑止を指定した UAP
- テスト中に異常終了した UAP

(4)　コマンド引数

- SPP | MHP
  起動するサービスグループの種別を指定します。
  - SPP：SPP を示します。
  - MHP：MHP を示します。
- サービスグループ名　　～〈1～31 文字の識別子〉
  起動するサービスグループの名称を指定します。
  サービスグループ名はオフラインテスタ環境定義で定義しておく必要があります。

13.2.7　stop（サービスグループの停止）

(1)　名称

サービスグループの停止

(2)　形式

```
stop {SPP|MHP} サービスグループ名
```

(3)　機能

起動中の UAP を停止させます。

(4) コマンド引数

- SPP | MHP
  停止するサービスグループの種別を指定します。
  - SPP：SPP を示します。
  - MHP：MHP を示します。
- サービスグループ名　～〈1～31 文字の識別子〉
  停止するサービスグループの名称を指定します。
  サービスグループ名はオフラインテスタ環境定義で定義しておく必要があります。

13.2.8 write（テスタファイル名の連絡）

(1) 名称

テスタファイル名の連絡

(2) 形式

```
write テスタファイル名
```

(3) 機能

各種シミュレート機能で必要になったテスタファイル名を，オフラインテスタに連絡します。

(4) コマンド引数

- テスタファイル名　～〈パス名〉
  オフラインテスタに連絡する，テスタファイルの名称を指定します。
  どのテスタファイル名の連絡が必要なのかは，次のプロンプト中に表示されます。
  〈形式〉

```
write(group1:service1:crm_snd)?>
      1.     2.      3.
```

1. サービスグループ名
2. サービス名（サービス外では表示しません）
3. テスタファイルの種別
   crm_snd：XATMI 送信データファイル

14 関数のシミュレーション内容

オフラインテスタが提供するシミュレート関数の内容と，関数のリターン値について説明します。

14.1　関数のシミュレーション内容の一覧

ここでは，関数のシミュレーション内容と，関数シミュレート時の注意事項について説明します。

14.1.1　関数のシミュレーション内容

シミュレート関数での，OpenTP1 の提供する関数のシミュレーション内容一覧を，次の表に示します。

表 14–1　OpenTP1 の提供する関数のシミュレーション内容一覧

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
システム運用の管理(adm)	dc_adm_call_command 関数 〔CBLDCADM(COMMAND)〕	運用コマンドの発行	○	○	運用コマンド結果データファイルから，データを入力して返します。
	dc_adm_complete 関数 〔CBLDCADM(COMPLETE)〕	ユーザサーバの開始完了報告	○	○	−
	dc_adm_status 関数 〔CBLDCADM(STATUS)〕	ユーザサーバの状態報告	○	○	正常時は，リターン値として DCADM_STAT_START_NORMAL を，リターンコードとして 0 を返します。
	dc_adm_get_nd_status_begin 関数	OpenTP1 ノードのステータス取得処理開始	○	○	関数リターン値ファイルに設定したノード識別子の個数を取得します。
	dc_adm_get_nd_status_next 関数	OpenTP1 ノードのステータス取得	○	○	関数リターン値ファイルに設定したノード識別子を取得します。正常時は，C 言語のリターン値として DCADM_STATUS_NORMAL を返します。
	dc_adm_get_nd_status_done 関数	OpenTP1 ノードのステータス取得処理終了	○	○	−
	dc_adm_get_nd_status 関数	OpenTP1 ノードのステータス取得	○	○	正常時は，リターン値として DCADM_STATUS_NORMAL を返します。

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
システム運用の管理(adm)	dc_adm_get_node_id 関数	システム共通定義からの自ノード識別子取得	○	○	関数リターン値ファイルに設定したノード識別子を取得します。
	dc_adm_get_sv_status_begin 関数	サーバのステータス取得処理開始	○	○	関数リターン値ファイルに設定したサーバ名の個数を取得します。
	dc_adm_get_sv_status_next 関数	OpenTP1 ノード上のサーバのステータス取得	○	○	関数リターン値ファイルに設定したサーバ名を取得します。正常時は，C言語のリターン値としてDCADM_STATUS_NORMAL を返します。
	dc_adm_get_sv_status_done 関数	サーバのステータス取得処理終了	○	○	－
	dc_adm_get_sv_status 関数	指定したサーバのステータス取得	○	○	正常時は，リターン値としてDCADM_STATUS_NORMAL を返します。
	dc_adm_get_nodeconf_begin 関数	ノード識別子取得開始	○	○	関数リターン値ファイルに設定したノード識別子の個数を返します。
	dc_adm_get_nodeconf_next 関数	関数発行元が属するマルチノードエリア，または指定されたサブエリアの全ノード識別子取得	○	○	関数リターン値ファイルに設定したノード識別子を返します。
	dc_adm_get_nodeconf_done 関数	ノード識別子取得終了	○	○	－
DAM ファイルサービス(dam)	dc_dam_close 関数 〔CBLDCDAM(CLOS)〕	DAM ファイルのクローズ	○	○	DAM ファイルをクローズします。
	dc_dam_create 関数 〔CBLDCDMB(CRAT)〕	物理ファイルの割り当て	×	×	DAM ファイルを生成し，そのファイル記述子を返します。
	dc_dam_end 関数 〔CBLDCDAM(END)〕	回復対象外ファイルの使用終了宣言	○	○	－
	dc_dam_get 関数 〔CBLDCDMB(GET)〕	物理ファイルのブロック読み込み	×	×	DAM ファイルから指定ブロックを指定バッファに読み込みます。

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
DAM ファイルサービス (dam)	dc_dam_hold 関数 〔CBLDCDAM(HOLD)〕	DAM ファイルの論理閉塞	○	○	DAM ファイルのヘッダに閉塞のステータスを設定し，閉塞状態にします。
	dc_dam_iclose 関数 〔CBLDCDMB(CLOS)〕	物理ファイルのクローズ	×	×	DAM ファイルをクローズします。
	dc_dam_iopen 関数 〔CBLDCDMB(OPEN)〕	物理ファイルのオープン	×	×	DAM ファイルをオープンし，そのファイル記述子を返します。
	dc_dam_open 関数 〔CBLDCDAM(OPEN)〕	DAM ファイルのオープン	○	○	DAM ファイルをオープンし，そのファイル記述子を返します。また，ファイル排他の指定がある場合はファイルを排他します。
	dc_dam_put 関数 〔CBLDCDMB(PUT)〕	物理ファイルのブロック書き込み	×	×	DAM ファイルの指定ブロックにバッファの内容を書き込みます。
	dc_dam_read 関数 〔CBLDCDAM(READ)〕	DAM ファイルのブロック読み込み	○	○	DAM ファイルの指定ブロックを指定バッファに読み込みます。また，ブロック排他の指定がある場合はファイルを排他します。
	dc_dam_start 関数 〔CBLDCDAM(STRT)〕	回復対象外ファイルの使用開始宣言	○	○	－
	dc_dam_status 関数 〔CBLDCDAM(STAT)〕	DAM ファイル状態表示	○	○	DAM ファイルの状態を返します。
	dc_dam_release 関数 〔CBLDCDAM(RLSE)〕	DAM ファイルの閉塞解除	○	○	DAM ファイルのヘッダが閉塞のステータスの場合，これをリセットし，閉塞を解除します。
	dc_dam_rewrite 関数 〔CBLDCDAM(REWT)〕	DAM ファイルのブロック更新	○	○	DAM ファイルの指定ブロックに指定バッファの内容を書き込みます。
	dc_dam_write 関数 〔CBLDCDAM(WRIT)〕	DAM ファイルの出力	○	○	DAM ファイルの指定ブロックに指定バッファの内容を書き込みます。

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
共用テーブルサービス(ist)	dc_ist_close 関数 〔CBLDCIST(CLOS)〕	IST テーブルのクローズ	○	○	IST テーブルをクローズします。
	dc_ist_open 関数 〔CBLDCIST(OPEN)〕	IST テーブルのオープン	○	○	IST テーブルをオープンし，その記述子を返します。
	dc_ist_read 関数 〔CBLDCIST(READ)〕	IST テーブルからのレコード読み込み	○	○	IST テーブルから指定レコードを指定バッファに読み込みます。
	dc_ist_write 関数 〔CBLDCIST(WRIT)〕	IST テーブルへのレコード書き込み	○	○	IST テーブルへ指定されたレコードを書き込みます。
ユーザジャーナルの取得(jnl)	dc_jnl_ujput 関数 〔CBLDCJNL(UJPUT)〕	UAP 履歴情報の取得	○	○	−
資源の排他制御(lck)	dc_lck_get 関数 〔CBLDCLCK(GET)〕	資源の排他要求	○	○	−
	dc_lck_release_all 関数 〔CBLDCLCK(RELALL)〕	全資源の排他解除要求	○	○	−
	dc_lck_release_byname 関数 〔CBLDCLCK(RELNAME)〕	資源名称指定による排他解除要求	○	○	−
メッセージログ管理(log)	dc_logprint 関数 〔CBLDCLOG(PRINT)〕	ログメッセージ出力要求	○	○	−
メッセージ制御機能(mcf)	dc_mcf_execap 関数 〔CBLDCMCF(EXECAP)〕 <SEND>	アプリケーションの起動	○	○	−
	dc_mcf_mainloop 関数 〔CBLDCMCF(MAINLOOP)〕	MCF サービス開始	○	○	オフラインテスタにサービスの開始を報告します。該当する MHP へのサービス要求があると，サービス関数を実行し，再度サービス要求待ちになります。オフラインテスタの終了などで UAP の終了要求を受け取ると，リターンします。
	dc_mcf_receive 関数	メッセージ受信	○	○	MCF 受信メッセージファイルからセグメント

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
メッセージ制御機能(mcf)	〔CBLDCMCF(RECEIVE)〕 <RECEIVE>	メッセージ受信	○	○	を入力し，メッセージ受信領域に格納します。また，トランザクション通番をカウントアップします。
	dc_mcf_reply 関数 〔CBLDCMCF(REPLY)〕 <SEND>	応答メッセージの送信	○	○	−
	dc_mcf_rollback 関数 〔CBLDCMCF(ROLLBACK)〕 <ROLLBACK>	部分回復	○	○	この関数発行後の処理を別トランザクションとして動作するように指定したとき，トランザクション通番をカウントアップします。
	dc_mcf_send 関数 〔CBLDCMCF(SEND)〕 <SEND>	メッセージ送信	○	○	−
	dc_mcf_open 関数 〔CBLDCMCF(OPEN)〕	MCF 通信サービス使用のための準備・初期化	○	○	−
	dc_mcf_close 関数 〔CBLDCMCF(CLOSE)〕	MCF 通信サービス使用のための環境を消去	○	×	−
	dc_mcf_sendrecv 関数 〔CBLDCMCF(SENDRECV)〕 <SEND>	同期型メッセージ送受信	○	○	最終セグメントのトレースを出力後，MCF 受信メッセージファイルからセグメントを入力し，メッセージ受信領域に格納します。
	dc_mcf_recvsync 関数 〔CBLDCMCF(RECVSYNC)〕 <RECEIVE>	同期型メッセージ受信	○	○	MCF 受信メッセージファイルからセグメントを入力し，メッセージ受信領域に格納します。
	dc_mcf_sendsync 関数 〔CBLDCMCF(SENDSYNC)〕 <SEND>/<ENABLE>/<DISABLE>	同期型メッセージ送信	○	○	−
	dc_mcf_tempget 関数 〔CBLDCMCF(TEMPGET)〕 <RECEIVE>	継続問い合わせ応答用一時記憶データ受け取り	○	○	一時記憶データファイルからデータを入力し，メッセージ受信領域に格納します。ファイルがな

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
メッセージ制御機能(mcf)	dc_mcf_tempget 関数 〔CBLDCMCF(TEMPGET)〕 <RECEIVE>	継続問い合わせ応答用一時記憶データ受け取り	○	○	い場合は，ヌル文字を格納します。
	dc_mcf_tempput 関数 〔CBLDCMCF(TEMPPUT)〕 <SEND>	継続問い合わせ応答用一時記憶データ更新	○	○	一時記憶データファイルを更新します。ファイルがない場合は，ファイルを作成して更新します。
	dc_mcf_contend 関数 〔CBLDCMCF(CONTEND)〕 <DISABLE>	継続問い合わせ応答終了	○	○	一時記憶データファイルを削除します。
	dc_mcf_regster 関数	UOC 関数アドレス登録	○	○	−
	dc_mcf_resend 関数 〔CBLDCMCF(RESEND)〕	メッセージの再送信	○	○	−
	dc_mcf_commit 関数 〔CBLDCMCF(COMMIT)〕	同期点取得	○	○	トランザクション通番をカウントアップします。
リモートプロシジャコール(rpc)	dc_rpc_call 関数 〔CBLDCRPC(CALL)〕	遠隔サービス呼び出し	○	○	オフラインテスタにサービス関数の実行を要求します。 DCRPC_NOWAIT 指定の場合，リターン値としてディスクリプタ（正の整数）を返します。 DCRPC_NOREPLY 指定の場合，指定したサービス（関数）には応答長として 0 を返します。
	dc_rpc_close 関数 〔CBLDCRPC(CLOSE)〕	UAP 終了処理	○	×	−
	dc_rpc_mainloop 関数 〔CBLDCRSV(MAINLOOP)〕	SPP サービス開始	○	○	オフラインテスタにサービスの開始を報告します。該当する SPP へのサービス要求があると，サービス関数を実行し，再度サービス要求待ちになります。オフラインテスタの終了などで UAP の終了要求を受け取ると，リターンします。

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
リモートプロシジャコール(rpc)	dc_rpc_open 関数 〔CBLDCRPC(OPEN)〕	UAP 開始処理	○	○	共用メモリを割り当てたあと，UAP の開始をオフラインテスタに報告します。
	dc_rpc_poll_any_replies 関数 〔CBLDCRPC(POLLANYR)〕	dc_rpc_call 関数（DCRPC_NOWAIT 指定）の応答を受信	○	○	flags が DCNOFLAGS の場合，まだ応答受信されていない dc_rpc_call 関数（DCRPC_NOWAIT 指定）の中で最初に発行した dc_rpc_call 関数のディスクリプタを返します。 flags が DCRPC_SPECIFIC_MSG の場合，DC_OK を返します。SPP 内で正常終了した dc_rpc_call 関数が発行されていない場合，DCRPC_PROTO を返します。
	dc_rpc_discard_further_replies 関数 〔CBLDCRPC(DISCARDF)〕	dc_rpc_call 関数（DCRPC_NOWAIT 指定）の応答を取り消し	○	×	dc_rpc_call 関数（DCRPC_NOWAIT 指定）で返したすべてのディスクリプタを取り消します。
	dc_rpc_get_callers_address 関数 〔CBLDCRPC(GETCLADR)〕	クライアントのノードアドレス通知	○	×	クライアントのアドレスとして，ADDRESS（固定値）を返します。
	dc_rpc_set_service_prio 関数 〔CBLDCRPC(SETSVPRI)〕	サービス要求のスケジュールプライオリティ設定	○	×	−
	dc_rpc_get_service_prio 関数 〔CBLDCRPC(GETSVPRI)〕	サービス要求のスケジュールプライオリティ値の取得	○	×	dc_rpc_set_service_prio 関数で設定したスケジュールプライオリティの値を返します。
	dc_rpc_set_watch_time 関数 〔CBLDCRPC(SETWATCH)〕	サービス応答待ち時間更新	○	○	サービス応答待ち時間を更新します。
	dc_rpc_get_watch_time 関数 〔CBLDCRPC(GETWATCH)〕	サービス応答待ち時間参照	○	×	dc_rpc_set_watch_time 関数で設定した値を参

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
リモートプロシジャコール(rpc)	dc_rpc_get_watch_time 関数 〔CBLDCRPC(GETWATCH)〕	サービス応答待ち時間参照	○	×	照します。未発行時には 180 を返します。
TAM ファイルサービス(tam)	dc_tam_close 関数	TAM テーブルのクローズ	○	○	TAM テーブルファイルの排他を解除し，クローズします。
	dc_tam_delete 関数 〔CBLDCTAM(ERS,ERSR)〕	TAM テーブルからレコード削除	○	○	キー値で指定されたレコードを TAM テーブルから削除します。TAM テーブルファイルの内容も変更します。
	dc_tam_get_inf 関数 〔CBLDCTAM(GST)〕	TAM テーブルの情報取得	○	○	指定 TAM テーブルファイルに対するオープン要求が関数発行元プロセスから発行されていれば DCTAM_STS_OPN を，発行されていなければ DCTAM_STS_CLS を返します。
	dc_tam_open 関数	TAM テーブルのオープン	○	○	テーブル ID での指定 TAM テーブルファイルをオープンし，そのファイル記述子をテーブル ID として返します。TAM テーブル排他の指定がある場合，TAM テーブルファイルを排他します。
	dc_tam_read 関数 〔CBLDCTAM(FxxR,FxxU)〕	TAM テーブルのレコード検索	○	○	共用メモリ上の TAM テーブル（管理部，インデクス部）から該当するインデクスを検索し，そのインデクスに対応するレコードを TAM テーブルファイルから読み込みます。レコード排他の指定がある場合，TAM テーブルファイルを排他します。
	dc_tam_read_cancel 関数	TAM テーブルのレコード検索を取り消し	○	○	指定レコードを含む TAM テーブルファイル

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
TAM ファイルサービス (tam)	dc_tam_read_cancel 関数	TAM テーブルのレコード検索を取り消し	○	○	に対する排他を解除します。
	dc_tam_rewrite 関数	TAM テーブルのレコード検索前提の更新	○	○	指定バッファの内容を TAM テーブル上の指定レコードに書き込みます。
	dc_tam_write 関数 〔CBLDCTAM(MFY,MFYS,STR)〕	TAM テーブルのレコード更新・追加	○	○	共用メモリ上の TAM テーブル（管理部，インデクス部）から該当するインデクスを検索し，そのインデクスに対応する，TAM テーブルファイル上のレコードに，指定バッファの内容を書き込みます。
トランザクション制御 (trn)	dc_trn_begin 関数 〔CBLDCTRN(BEGIN)〕	トランザクションの開始	○	○	トランザクション通番をカウントアップします。
	dc_trn_chained_commit 関数 〔CBLDCTRN(C-COMMIT)〕	トランザクションのコミット（連鎖モード）	○	○	トランザクション通番をカウントアップします。
	dc_trn_chained_rollback 関数 〔CBLDCTRN(C-ROLL)〕	トランザクションのロールバック（連鎖モード）	○	○	トランザクション通番をカウントアップします。
	dc_trn_info 関数 〔CBLDCTRN(INFO)〕	現在のトランザクションの情報をリターン	○	○	関数リターン値ファイルの指定がなければ，0 を返します。
	dc_trn_unchained_commit 関数 〔CBLDCTRN(U-COMMIT)〕	トランザクションのコミット（非連鎖モード）	○	○	－
	dc_trn_unchained_rollback 関数 〔CBLDCTRN(U-ROLL)〕	トランザクションのロールバック（非連鎖モード）	○	○	－
TX インタフェース (tx_～)	tx_begin 関数 〔TXBEGIN〕	トランザクションの開始	○	○	トランザクション通番をカウントアップし，TXINFO 情報を初期化します。
	tx_close 関数 〔TXCLOSE〕	リソースマネジャ集合のクローズ	○	○	－

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
TX インタフェース(tx_～)	tx_commit 関数 〔TXCOMMIT〕	トランザクションのコミット	○	○	連鎖モードの場合，トランザクション通番をカウントアップします。
	tx_info 関数 〔TXINFORM〕	現在のトランザクションの情報をリターン	○	○	関数リターン値ファイルの指定がなければ，0 を返します。
	tx_open 関数 〔TXOPEN〕	リソースマネジャ集合のオープン	○	○	－
	tx_set_commit_return 関数 〔TXSETCOMMITRET〕	commit_return 特性の設定	○	○	－
	tx_set_transaction_control 関数 〔TXSETTRANCTL〕	transaction_control 特性の設定	○	○	transaction_control 特性を設定します。
	tx_set_transaction_timeout 関数 〔TXSETTIMEOUT〕	transaction_timeout 特性の設定	○	○	－
	tx_rollback 関数 〔TXROLLBACK〕	トランザクションのロールバック	○	○	連鎖モードの場合，トランザクション通番をカウントアップし，transaction_state 特性を設定します。
XATMI インタフェース(tp～)	tpalloc 関数	タイプトバッファの割り当て	○	○	引数の type 型で指定されたバッファを割り当て，そのポインタを返します。
	tpfree 関数	タイプトバッファの解放	○	×	tpalloc 関数，tprealloc 関数で割り当てたバッファを解放します。
	tprealloc 関数	タイプトバッファのサイズの変更	○	○	tpalloc 関数，tprealloc 関数で割り当てたバッファのサイズを変更します。
	tptypes 関数	タイプトバッファの情報の取得	○	○	tpalloc 関数，tprealloc 関数で割り当てたバッファのタイプとサブタイプを返します。
	tpservice 関数	サービス関数のテンプレート	○	×	サービス関数を呼び出す直前にトレース情報を取得します。

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	機能	トレース情報の取得可否	リターン値の設定可否	関数の処理
XATMI インタフェース(tp～)	tpreturn 関数	サービス関数からのリターン	○	○	リターン情報を設定して，サービス要求元にリターンします。
	tpadvertise 関数	サービス名の広告	○	○	－
	tpunadvertise 関数	サービス名の広告の取り消し	○	○	－
	tpacall 関数	サービスの非同期要求	○	○	オフラインテスタにサービス関数の実行を要求します。呼び出し結果は tpgetrply 関数で返します。
	tpcall 関数	サービスの同期要求	○	○	オフラインテスタにサービス関数の実行を要求します。
	tpcancel 関数	サービスのキャンセル	○	○	tpacall 関数で呼び出したサービスの応答をキャンセルします。
	tpgetrply 関数	サービスからの非同期応答の受信	○	○	サービス関数の実行結果を返します。
	tpconnect 関数	会話型サービスとのコネクションの確立	○	○	オフラインテスタにサービス関数の実行を要求します。実行結果は tprecv 関数で返します。
	tpdiscon 関数	会話型サービスとのコネクションの切断	○	○	受信待ち（tprecv 関数）があれば終了させ，tpdiscon 関数受け付け後の tpsend 関数，tprecv 関数を受け付けないようにします。
	tprecv 関数	会話型サービスからのメッセージ受信	○	○	XATMI 受信データファイルからデータを入力します。
	tpsend 関数	会話型サービスへのメッセージ送信	○	○	XATMI 送信データファイルにデータを出力します。
オンラインテスタ(uto)	dc_uto_test_status 関数 〔CBLDCUTO(T-STATUS)〕	ユーザサーバのテスト状態報告	○	○	非テストモード状態を返します。

（凡例）

○：取得または設定できます。

×：取得または設定できません。

－：処理しません。

14.1.2　関数シミュレート時の注意事項

関数シミュレート時の注意事項を次に示します。

1. 関数発行元の UAP の種別，トランザクション状態，および main 関数の内か外かはチェックされません。
2. 関数の発行順序は，オフラインテスタの動作に影響するものだけがチェックされます。
3. 引数の内容はチェックされません。ユーザがトレース情報を確認してチェックしてください。
4. COBOL 言語のプログラムで，インタフェースコードや要求コードに設定誤りがある場合は，該当するプログラムのトレースは取得されません。ただし，エラーメッセージは出力されます。
5. dc_trn_～関数と，tx_～関数は混在して使用できません。また，混在して使用してもチェックされません。
6. オフラインテスタは，トランザクションの回数（以後，トランザクション通番と呼びます）をカウントしています。トランザクション通番は，一部のシミュレート関数発行時と，call サブコマンド実行時にカウントアップします。トランザクション通番は，tx_info 関数のトレース情報で参照できます。トランザクション通番をカウントアップするシミュレート関数については，「14.1.1　関数のシミュレーション内容」を参照してください。

14.2　シミュレート関数リターン値の一覧

シミュレート関数のリターン値の一覧（ただし，0，DC_OK，DCMCFRTN_00000，TX_OK を除きます）を，次の表に示します。

表 14–2　シミュレート関数のリターン値一覧

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
システム運用の管理(adm)	dc_adm_call_command 関数 〔CBLDCADM(COMMAND)〕	DCADMER_STATNOTZERO DCADMER_PARAM DCADMER_MEMORY_OUT DCADMER_MEMORY_ERR DCADMER_MEMORY_OUTERR DCADMER_PROTO	01801 01802 01803 01804 01805 01807※1
	dc_adm_complete 関数 〔CBLDCADM(COMPLETE)〕	DCADM_STAT_START_NORMAL DCADMER_PROTO DCADMER_PARAM	00000 01830※1 01831
	dc_adm_status 関数 〔CBLDCADM(STATUS)〕	DCADMER_PROTO DCADMER_PARAM	01830※1 01831
	dc_adm_get_nd_status_begin 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※2 ―
	dc_adm_get_nd_status_next 関数	DCADM_STAT_START_NORMAL DCADMER_PROTO DCADMER_PARAM DCADMER_NO_MORE_ENTRY	― ―※1，※3 ― ―
	dc_adm_get_nd_status_done 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※3 ―
	dc_adm_get_nd_status 関数	DCADM_STAT_START_NORMAL DCADMER_PROTO DCADMER_PARAM	― ―※1，※2 ―
	dc_adm_get_node_id 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※2 ―
	dc_adm_get_sv_status_begin 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※2 ―
	dc_adm_get_sv_status_next 関数	DCADM_STAT_START_NORMAL DCADMER_PROTO DCADMER_PARAM	― ―※1，※3

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
システム運用の管理(adm)	dc_adm_get_sv_status_next 関数	DCADMER_NO_MORE_ENTRY	― ―
	dc_adm_get_sv_status_done 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※3 ―
	dc_adm_get_sv_status 関数	DCADM_STAT_START_NORMAL DCADMER_PROTO DCADMER_PARAM	― ―※1，※2 ―
	dc_adm_get_nodeconf_begin 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※2 ―
	dc_adm_get_nodeconf_next 関数	DCADMER_PROTO DCADMER_PARAM DCADMER_NO_MORE_ENTRY	―※1，※3 ― ―
	dc_adm_get_nodeconf_done 関数	DCADMER_PROTO DCADMER_PARAM	―※1，※3 ―
DAM ファイルサービス(dam)	dc_dam_close 関数 〔CBLDCDAM(CLOS)〕	DCDAMER_PROTO DCDAMER_BADF DCDAMER_PARAM_FLAGS	01600※1 01603 01611
	dc_dam_create 関数 〔CBLDCDMB(CRAT)〕	DCDAMER_NOMEM DCDAMER_OPENED DCDAMER_PARAM_FLAGS DCDAMER_FILEER DCDAMER_PNUMER DCDAMER_EXIST DCDAMER_IOER DCDAMER_OPENNUM DCDAMER_ACCESS DCDAMER_LBLNER DCDAMER_LBNOER DCDAMER_LFNOVF	01607 01608 01611 01614 01615 01617 01620 01627 01628 01630 01631 01635
	dc_dam_end 関数 〔CBLDCDAM(END)〕	DCDAMER_PROTO DCDAMER_PARAM_FLAGS	01600※1 01611
	dc_dam_get 関数 〔CBLDCDMB(GET)〕	DCDAMER_BADF DCDAMER_BUFER DCDAMER_SEQER DCDAMER_PARAM_FLAGS	01603 01604 01605 01611

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
DAM ファイルサービス (dam)	dc_dam_get 関数 〔CBLDCDMB(GET)〕	DCDAMER_IOER DCDAMER_EOF	01620 01637
	dc_dam_hold 関数 〔CBLDCDAM(HOLD)〕	DCDAMER_PROTO DCDAMER_UNDEF DCDAMER_PARAM_LFNAME DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_LHOLDED DCDAMER_OHOLDED	01600※1 01601 01610 01611 01620 01625 01626
	dc_dam_iclose 関数 〔CBLDCDMB(CLOS)〕	DCDAMER_BADF DCDAMER_PARAM_FLAGS	01603 01611
	dc_dam_iopen 関数 〔CBLDCDMB(OPEN)〕	DCDAMER_NOMEM DCDAMER_OPENED DCDAMER_PARAM_FLAGS DCDAMER_FILEER DCDAMER_PNUMER DCDAMER_NOEXIST DCDAMER_IOER DCDAMER_OPENNUM DCDAMER_ACCESS DCDAMER_LFNOVF	01607 01608 01611 01614 01615 01619 01620 01627 01628 01635
	dc_dam_open 関数 〔CBLDCDAM(OPEN)〕	DCDAMER_PROTO DCDAMER_UNDEF DCDAMER_EXCER DCDAMER_OPENED DCDAMER_PARAM_LFNAME DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_LHOLD DCDAMER_OHOLD DCDAMER_OPENNUM DCDAMER_ACCESS	01600※1 01601 01602 01608 01610 01611 01620 01621 01622 01627 01628
	dc_dam_put 関数 〔CBLDCDMB(PUT)〕	DCDAMER_BADF DCDAMER_BUFER DCDAMER_SEQER DCDAMER_PARAM_FLAGS DCDAMER_IOER	01603 01604 01605 01611 01620

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
DAM ファイルサービス (dam)	dc_dam_put 関数 〔CBLDCDMB(PUT)〕	DCDAMER_EOF	01637
	dc_dam_read 関数 〔CBLDCDAM(READ)〕	DCDAMER_PROTO DCDAMER_EXCER DCDAMER_BADF DCDAMER_BUFER DCDAMER_BNOER DCDAMER_PARAM_KEYNO DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_LHOLD DCDAMER_OHOLD	01600※1 01602 01603 01604 01606 01609 01611 01620 01621 01622
	dc_dam_start 関数 〔CBLDCDAM(STRT)〕	DCDAMER_PROTO DCDAMER_PARAM_FLAGS DCDAMER_STARTED	01600※1 01611 01647
	dc_dam_status 関数 〔CBLDCDAM(STAT)〕	DCDAMER_PROTO DCDAMER_UNDEF DCDAMER_PARAM_LFNAME DCDAMER_PARAM_FLAGS DCDAMER_PARAM_ERROR DCDAMER_IOER	01600※1 01601 01610 01611 01612 01620
	dc_dam_release 関数 〔CBLDCDAM(RLSE)〕	DCDAMER_PROTO DCDAMER_UNDEF DCDAMER_PARAM_LFNAME DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_NOLHOLD DCDAMER_NOOHOLD	01600※1 01601 01610 01611 01620 01623 01624
	dc_dam_rewrite 関数 〔CBLDCDAM(REWT)〕	DCDAMER_PROTO DCDAMER_BADF DCDAMER_BUFER DCDAMER_BNOER DCDAMER_PARAM_KEYNO DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_LHOLD DCDAMER_OHOLD	01600※1 01603 01604 01606 01609 01611 01620 01621 01622

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
DAM ファイルサービス (dam)	dc_dam_rewrite 関数 〔CBLDCDAM(REWT)〕	DCDAMER_BUFOV	01641
	dc_dam_write 関数 〔CBLDCDAM(WRIT)〕	DCDAMER_PROTO DCDAMER_EXCER DCDAMER_BADF DCDAMER_BUFER DCDAMER_BNOER DCDAMER_PARAM_KEYNO DCDAMER_PARAM_FLAGS DCDAMER_IOER DCDAMER_LHOLD DCDAMER_OHOLD DCDAMER_BUFOV	01600※1 01602 01603 01604 01606 01609 01611 01620 01621 01622 01641
共用テーブルサービス(ist)	dc_ist_close 関数 〔CBLDCIST(CLOS)〕	DCISTER_PROTO DCISTER_BADID DCISTER_PARAM_FLAGS	—※1 — —
	dc_ist_open 関数 〔CBLDCIST(OPEN)〕	DCISTER_PROTO DCISTER_UNDEF DCISTER_OPENED DCISTER_PARAM_TBLNAME DCISTER_PARAM_FLAGS	—※1 — — — —
	dc_ist_read 関数 〔CBLDCIST(READ)〕	DCISTER_PROTO DCISTER_BADID DCISTER_BUFER DCISTER_RNOER DCISTER_NOMEM DCISTER_PARAM_KEYNO DCISTER_PARAM_FLAGS	—※1 — — — — — —
	dc_ist_write 関数 〔CBLDCIST(WRIT)〕	DCISTER_PROTO DCISTER_BADID DCISTER_BUFER DCISTER_RNOER DCISTER_NOMEM DCISTER_PARAM_KEYNO DCISTER_PARAM_FLAGS DCISTER_BUFOV	—※1 — — — — — — —

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
ユーザジャーナルの取得(jnl)	dc_jnl_ujput 関数 〔CBLDCJNL(UJPUT)〕	DCJNLER_PARAM DCJNLER_SHORT DCJNLER_PROTO	01101 01102 01105※1
資源の排他制御(lck)	dc_lck_get 関数 〔CBLDCLCK(GET)〕	DCLCKER_PARAM DCLCKER_OUTOFTRN	00401 00455※1
	dc_lck_release_all 関数 〔CBLDCLCK(RELALL)〕	DCLCKER_PARAM DCLCKER_OUTOFTRN	00401 00455※1
	dc_lck_release_byname 関数 〔CBLDCLCK(RELNAME)〕	DCLCKER_PARAM DCLCKER_OUTOFTRN	00401 00455※1
メッセージログ管理(log)	dc_logprint 関数 〔CBLDCLOG(PRINT)〕	DCLOGER_PARAM_ARGS DCLOGER_COMM	01900 01901※1
メッセージ制御機能(mcf)	dc_mcf_execap 関数 〔CBLDCMCF(EXECAP)〕 <SEND>	DCMCFER_PROTO DCMCFRTN_71002 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72005 DCMCFRTN_72016 DCMCFRTN_72024 DCMCFRTN_72026 DCMCFRTN_72041 DCMCFRTN_72108	70901※1, ※4 71002 72000 72001 72005 72016 72024 72026 72041 72108
	dc_mcf_mainloop 関数 〔CBLDCMCF(MAINLOOP)〕	DCMCFER_INVALID_ARGS DCMCFER_PROTO DCMCFER_FATAL	70900 70901※1, ※5 70902
	dc_mcf_receive 関数 〔CBLDCMCF(RECEIVE)〕 <RECEIVE>	DCMCFER_PROTO DCMCFRTN_71000 DCMCFRTN_71001 DCMCFRTN_71002 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72013 DCMCFRTN_72016 DCMCFRTN_72024 DCMCFRTN_72025 DCMCFRTN_72036	70901※1, ※4 71000 71001 71002 72000 72001 72013 72016 72024 72025 72036

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
メッセージ制御機能(mcf)	dc_mcf_reply 関数 〔CBLDCMCF(REPLY)〕 <SEND>	DCMCFER_PROTO DCMCFRTN_71002 DCMCFRTN_72000 DCMCFRTN_72005 DCMCFRTN_72016 DCMCFRTN_72017 DCMCFRTN_72026 DCMCFRTN_72041 DCMCFRTN_72047	70901※1, ※4 71002 72000 72005 72016 72017 72026 72041 72047
	dc_mcf_rollback 関数 〔CBLDCMCF(ROLLBACK)〕 <ROLLBACK>	DCMCFER_PROTO DCMCFRTN_72000 DCMCFRTN_72027	70901※1, ※4 72000 72027
	dc_mcf_send 関数 〔CBLDCMCF(SEND)〕 <SEND>	DCMCFER_PROTO DCMCFRTN_71002 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72005 DCMCFRTN_72016 DCMCFRTN_72017 DCMCFRTN_72020 DCMCFRTN_72024 DCMCFRTN_72026 DCMCFRTN_72041	70901※1, ※4 71002 72000 72001 72005 72016 72017 72020 72024 72026 72041
	dc_mcf_open 関数 〔CBLDCMCF(OPEN)〕	DCMCFER_INVALID_ARGS DCMCFER_PROTO	70900 70901※1
	dc_mcf_close 関数 〔CBLDCMCF(CLOSE)〕	－	－
	dc_mcf_sendrecv 関数 〔CBLDCMCF(SENDRECV)〕 <SEND>	DCMCFER_PROTO DCMCFRTN_71002 DCMCFRTN_71108 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72005 DCMCFRTN_72013 DCMCFRTN_72016 DCMCFRTN_72024 DCMCFRTN_72026	70901※1, ※4 71002 71108 72000 72001 72005 72013 72016 72024 72026

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
メッセージ制御機能(mcf)	dc_mcf_sendrecv 関数 〔CBLDCMCF(SENDRECV)〕 <SEND>	DCMCFRTN_72036 DCMCFRTN_72041	72036 72041
	dc_mcf_recvsync 関数 〔CBLDCMCF(RECVSYNC)〕 <RECEIVE>	DCMCFER_PROTO DCMCFRTN_71001 DCMCFRTN_71108 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72013 DCMCFRTN_72016 DCMCFRTN_72024 DCMCFRTN_72025 DCMCFRTN_72036 DCMCFRTN_73001	70901※1, ※4 71001 71108 72000 72001 72013 72016 72024 72025 72036 73001
	dc_mcf_sendsync 関数 〔CBLDCMCF(SENDSYNC)〕 <SEND>/<ENABLE>/ <DISABLE>	DCMCFER_PROTO DCMCFRTN_71002 DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72005 DCMCFRTN_72016 DCMCFRTN_72024 DCMCFRTN_72026 DCMCFRTN_72041	70901※1, ※4 71002 72000 72001 72005 72016 72024 72026 72041
	dc_mcf_tempget 関数 〔CBLDCMCF(TEMPGET)〕 <RECEIVE>	DCMCFER_PROTO DCMCFRTN_72000 DCMCFRTN_72013 DCMCFRTN_72016 DCMCFRTN_72036 DCMCFRTN_72106	70901※1, ※4 72000 72013 72016 72036 72106
	dc_mcf_tempput 関数 〔CBLDCMCF(TEMPPUT)〕 <SEND>	DCMCFER_PROTO DCMCFRTN_71103 DCMCFRTN_72000 DCMCFRTN_72013 DCMCFRTN_72016 DCMCFRTN_72035 DCMCFRTN_72106	70901※1, ※4 71103 72000 72013 72016 72035 72106
	dc_mcf_contend 関数	DCMCFER_PROTO	70901※1

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
メッセージ制御機能(mcf)	〔CBLDCMCF(CONTEND)〕 <DISABLE>	DCMCFRTN_72000 DCMCFRTN_72016	72000 72016
	dc_mcf_regster 関数	DCMCFER_INVALID_ARGS DCMCFER_PROTO	－ ＿※1
	dc_mcf_resend 関数 〔CBLDCMCF(RESEND)〕	DCMCFER_PROTO DCMCFRTN_72000 DCMCFRTN_72001 DCMCFRTN_72011 DCMCFRTN_72016 DCMCFRTN_72017 DCMCFRTN_72024 DCMCFRTN_72047	70901※1, ※4 72000 72001 72011 72016 72017 72024 72047
	dc_mcf_commit 関数 〔CBLDCMCF(COMMIT)〕	DCMCFER_PROTO DCMCFRTN_72000 DCMCFRTN_72016	70901※1, ※4 72000 72016
リモートプロシジャコール(rpc)	dc_rpc_call 関数 〔CBLDCRPC(CALL)〕	DCRPCER_INVALID_ARGS DCRPCER_PROTO DCRPCER_MESSAGE_TOO_BIG DCRPCER_REPLY_TOO_BIG DCRPCER_NO_SUCH_SERVICE_GROUP DCRPCER_NO_SUCH_SERVICE DCRPCER_SERVICE_CLOSED DCRPCER_SYSERR_AT_SERVER DCRPCER_SYSER	00301 00302※1, ※5 00308 00309 00310 00311 00312 00316 00318
	dc_rpc_close 関数 〔CBLDCRPC(CLOSE)〕	－	－
	dc_rpc_mainloop 関数 〔CBLDCRSV(MAINLOOP)〕	DCRPCER_INVALID_ARGS DCRPCER_PROTO DCRPCER_FATAL	00301 00302※1, ※5 00303
	dc_rpc_open 関数 〔CBLDCRPC(OPEN)〕	DCRPCER_INVALID_ARGS DCRPCER_PROTO DCRPCER_FATAL	00301 00302 00303
	dc_rpc_poll_any_replies 関数 〔CBLDCRPC(POLLANYR)〕	DCRPCER_INVALID_ARGS DCRPCER_PROTO DCRPCER_REPLY_TOO_BIG DCRPCER_NO_SUCH_SERVICE	00301 00302※1, ※6 00309 00311

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
リモートプロシジャコール(rpc)	dc_rpc_poll_any_replies 関数 〔CBLDCRPC(POLLANYR)〕	DCRPCER_SERVICE_CLOSEDDCRPCER_SYSERR_AT_SERVER DCRPCER_NO_BUFS_AT_SERVER DCRPCER_ALL_RECEIVED	00312 00316 00318 00321
	dc_rpc_discard_further_replies 関数 〔CBLDCRPC(DISCARDF)〕	－	－
	dc_rpc_get_callers_address 関数 〔CBLDCRPC(GETCLADR)〕	－	－
	dc_rpc_set_service_prio 関数 〔CBLDCRPC(SETSVPRI)〕	－	－
	dc_rpc_get_service_prio 関数 〔CBLDCRPC(GETSVPRI)〕	DCRPCER_PROTO	00302※1
	dc_rpc_set_watch_time 関数 〔CBLDCRPC(SETWATCH)〕	DCRPCER_INVALID_ARGS DCRPCER_PROTO	00301 00302※1
	dc_rpc_get_watch_time 関数 〔CBLDCRPC(GETWATCH)〕	DCRPCER_PROTO	00302※1
TAM ファイルサービス(tam)	dc_tam_close 関数	DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN	－ －※1 －
	dc_tam_delete 関数 〔CBLDCTAM(ERS,ERSR)〕	DCTAMER_PARAM_KEY DCTAMER_PARAM_KNO DCTAMER_PARAM_BFA DCTAMER_PARAM_BFS DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN DCTAMER_NOREC DCTAMER_LOCK DCTAMER_MEMORY DCTAMER_IO	01702 01703 01704 01705 01708 01721※1 01726 01731 01736 01769 01770
	dc_tam_get_inf 関数 〔CBLDCTAM(GST)〕	DCTAMER_PARAM_TBL DCTAMER_PARAM_FLG DCTAMER_UNDEF DCTAMER_PROTO	01701 01708 01710 01721※1

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
TAM ファイルサービス (tam)	dc_tam_open 関数	DCTAMER_PARAM_TBL DCTAMER_PARAM_FLG DCTAMER_UNDEF DCTAMER_PROTO DCTAMER_NOLOAD DCTAMER_OPENED DCTAMER_LOCK DCTAMER_OPENNUM DCTAMER_IO	－ － － －※1 － － － － －
	dc_tam_read 関数 〔CBLDCTAM(FxxR,FxxU)〕	DCTAMER_PARAM_KEY DCTAMER_PARAM_KNO DCTAMER_PARAM_BFA DCTAMER_PARAM_BFS DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN DCTAMER_IDXTYP DCTAMER_NOREC DCTAMER_LOCK DCTAMER_MEMORY DCTAMER_IO	01702 01703 01704 01705 01708 01721※1 01726 01729 01731 01736 01769 01770
	dc_tam_read_cancel 関数	DCTAMER_PARAM_KEY DCTAMER_PARAM_KNO DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN DCTAMER_NOREC DCTAMER_MEMORY	－ － － －※1 － － －
	dc_tam_rewrite 関数	DCTAMER_PARAM_KEY DCTAMER_PARAM_KNO DCTAMER_PARAM_DTA DCTAMER_PARAM_DTS DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN DCTAMER_NOREC DCTAMER_MEMORY DCTAMER_IO	－ － － － － －※1 － － － －

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
TAM ファイルサービス(tam)	dc_tam_write 関数 〔CBLDCTAM(MFY,MFYS,STR)〕	DCTAMER_PARAM_KEY DCTAMER_PARAM_KNO DCTAMER_PARAM_DTA DCTAMER_PARAM_DTS DCTAMER_PARAM_FLG DCTAMER_PROTO DCTAMER_NOOPEN DCTAMER_NOREC DCTAMER_EXKEY DCTAMER_LOCK DCTAMER_NOAREA DCTAMER_MEMORY DCTAMER_IO	01702 01703 01706 01707 01708 01721※1 01726 01731 01735 01736 01763 01769 01770
トランザクション制御(trn)	dc_trn_begin 関数 〔CBLDCTRN(BEGIN)〕	DCTRNER_PROTO	00905※1
	dc_trn_chained_commit 関数 〔CBLDCTRN(C-COMMIT)〕	DCTRNER_PROTO	00905※1
	dc_trn_chained_rollback 関数 〔CBLDCTRN(C-ROLL)〕	DCTRNER_PROTO	00905※1
	dc_trn_info 関数 〔CBLDCTRN(INFO)〕	1	00001 00908
	dc_trn_unchained_commit 関数 〔CBLDCTRN(U-COMMIT)〕	DCTRNER_PROTO	00905※1
	dc_trn_unchained_rollback 関数 〔CBLDCTRN(U-ROLL)〕	DCTRNER_PROTO	00905※1
TX インタフェース(tx_～)	tx_begin 関数 〔TXBEGIN〕	TX_PROTOCOL_ERROR	TX_PROTOCOL_ERROR※1
	tx_close 関数 〔TXCLOSE〕	－	－
	tx_commit 関数 〔TXCOMMIT〕	TX_PROTOCOL_ERROR	TX_PROTOCOL_ERROR※1
	tx_info 関数 〔TXINFORM〕	TX_PROTOCOL_ERROR	TX_PROTOCOL_ERROR※1
	tx_open 関数 〔TXOPEN〕	TX_ERROR	TX_ERROR※1

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
TX インタフェース(tx_～)	tx_set_commit_return 関数 〔TXSETCOMMITRET〕	TX_EINVAL TX_NOT_SUPPORTED TX_PROTOCOL_ERROR	TX_EINVAL TX_NOT_SUPPORTED TX_PROTOCOL_ERROR※1
	tx_set_transaction_control 関数 〔TXSETTRANCTL〕	TX_EINVAL TX_PROTOCOL_ERROR	TX_EINVAL TX_PROTOCOL_ERROR※1
	tx_set_transaction_timeout 関数 〔TXSETTIMEOUT〕	TX_EINVAL TX_PROTOCOL_ERROR	TX_EINVAL TX_PROTOCOL_ERROR※1
	tx_rollback 関数 〔TXROLLBACK〕	TX_PROTOCOL_ERROR	TX_PROTOCOL_ERROR※1
XATMI インタフェース(tp～)	tpalloc 関数	TPEINVAL TPENOENT TPESYSTEM TPEPROTO	－ － － －※1
	tpfree 関数	－	－
	tprealloc 関数	TPEINVAL TPESYSTEM TPEPROTO	－ － －※1
	tptypes 関数	TPEINVAL TPEPROTO	－ －※1
	tpreturn 関数	－	－
	tpadvertise 関数	TPEINVAL TPEPROTO	－ －※1
	tpunadvertise 関数	TPEINVAL TPEPROTO	－ －※1
	tpacall 関数	TPEINVAL TPEPROTO TPENOENT TPEITYPE TPETRAN	－ －※1，※7，※8，※9 － － －
	tpcall 関数	TPEINVAL TPEPROTO TPENOENT TPEITYPE TPEOTYPE	－ －※1，※7，※8，※9 － － －

機能名称	関数名 〔プログラム名 (要求コード)〕 <DML>	C 言語でのリターン値	COBOL 言語でのリターンコード
XATMI インタフェース (tp～)	tpcall 関数	TPETRAN TPESVCFAIL TPESVCERR	－ － －
	tpcancel 関数	TPEBADDESC TPETRAN TPEPROTO	－ － －※1，※7，※8
	tpgetrply 関数	TPEBADDESC TPEOTYPE TPESYSTEM TPEPROTO TPESVCFAIL TPESVCERR	－ － － －※1，※7，※8，※9，※10 － －
	tpconnect 関数	TPEINVAL TPENOENT TPEITYPE TPETRAN TPEPROTO	－ － － － －※1，※7，※8，※9
	tpdiscon 関数	TPEBADDESC TPEPROTO	－ －※1，※7，※8，※11
	tprecv 関数	TPEINVAL TPEOTYPE TPEBADDESC TPEPROTO	－ － － －※1，※7，※8，※12
	tpsend 関数	TPEINVAL TPEBADDESC TPEPROTO	－ － －※1，※7，※8，※13
オンラインテスタ (uto)	dc_uto_test_status 関数 [CBLDCUTO(T-STATUS)]	DCUTOER_PROTO DCUTOER_PARAM_FLAGS DCUTOER_PARAM_ADDS	02701※1 02757 02758

(凡例)

－：リターン値（リターンコード）がない場合です。

注

XATMI インタフェースの場合，C 言語でのリターン値は tperrno に返す値を示します。

注※1

dc_rpc_open 関数が未発行の場合です。

注※2

dc_adm_get_nd_status_begin 関数，dc_adm_get_sv_status_begin 関数，または dc_adm_get_nodeconf_begin 関数が発行済みの場合です。

注※3

dc_adm_get_nd_status_begin 関数，dc_adm_get_sv_status_begin 関数，または dc_adm_get_nodeconf_begin 関数が未発行の場合です。

注※4

メイン関数で発行した場合です。

注※5

dc_mcf_mainloop 関数または dc_rpc_mainloop 関数が発行済みの場合です。

注※6

非同期応答型の dc_rpc_call 関数が未発行の場合です。

注※7

tpreturn 関数の発行後の場合です。

注※8

サービスパラダイムの異なるサービス環境での発行の場合です。

注※9

サービスグループのリカーシブコールの場合です。

注※10

tpacall 関数が未発行の場合です。

注※11

コネクションのオリジネータでない場合です。

注※12

コネクションの属性が TPSENDONLY の場合です。

注※13

コネクションの属性が TPRECVONLY の場合です。

14.3　シミュレート機能未サポート関数一覧

「14.1.1　関数のシミュレーション内容」で示しているように，オフラインテスタのシミュレート関数を使用すると OpenTP1 の提供する関数のシミュレーションができます。ただし，以降の表で示している OpenTP1 の提供する関数については，オフラインテスタでのシミュレート関数が未サポートです。したがって，これらの関数を UAP で実行しても，トレース情報の取得や関数引数の変更などの処理を実行しないで，以降の表に示すリターン値を返します。また，関数リターン値ファイルのリターン値の設定もできません。

シミュレート機能未サポート関数の一覧を，C 言語の場合と COBOL 言語の場合に分けて，以降の表に示します。

表 14-3　シミュレート機能未サポート関数一覧（C 言語）

機能名称	関数名	OpenTP1 提供関数の機能	リターン値
リモートプロシジャコール（rpc）	dc_rpc_call_to 関数	通信先を指定した遠隔サービスの呼び出し	0
	dc_rpc_get_error_descriptor 関数	エラーが発生した非同期応答型 RPC 要求の記述子の取得	1
	dc_rpc_discard_specific_reply 関数	特定の処理結果の受信の拒否	DC_OK
	dc_rpc_service_retry 関数	サービス関数のリトライ	DC_OK
	dc_rpc_get_gateway_address 関数	ゲートウェイのノードアドレスの取得	DC_OK
	dc_rpc_cltsend 関数	CUP への一方通知	DC_OK
リモート API 機能（rap）	dc_rap_connect 関数	rap リスナーとのコネクションの確立	DC_OK
	dc_rap_disconnect 関数	rap リスナーとのコネクションの解放	DC_OK
性能検証用トレース（prf）	dc_prf_utrace_put 関数	ユーザ固有の性能検証用トレースの取得	DC_OK
	dc_prf_get_trace_num 関数	性能検証用トレース取得通番の通知	DC_OK
メッセージ送受信（mcf）	dc_mcf_ap_info 関数	アプリケーション情報通知	DCMCFRTN_00000
	dc_mcf_ap_info_uoc 関数	UOC へのアプリケーション情報通知	DCMCFRTN_00000
	dc_mcf_timer_set 関数	ユーザタイマ監視の設定	DC_OK

機能名称	関数名	OpenTP1 提供関数の機能	リターン値
メッセージ送受信（mcf）	dc_mcf_timer_cancel 関数	ユーザタイマ監視の取り消し	DC_OK
DAM ファイルサービス（dam）	dc_dam_bseek 関数	物理ファイルのブロックの検索	関数の引数に指定した，相対ブロック番号をリターンします。
	dc_dam_dget 関数	物理ファイルからブロックの直接入力	504
	dc_dam_dput 関数	物理ファイルへブロックの直接出力	504

表 14-4　シミュレート機能未サポート関数一覧（COBOL 言語）

機能名称	プログラム名(要求コード)	OpenTP1 提供関数の機能	ステータスコード
リモートプロシジャコール（rpc）	CBLDCRPC('GETERDES')	エラーが発生した非同期応答型 RPC 要求の記述子の取得	00000
	CBLDCRPC('DISCARDS')	特定の処理結果の受信の拒否	00000
	CBLDCRPC('SVRETRY')	サービスプログラムのリトライ	00000
	CBLDCRPC('GETGWADR')	ゲートウェイのノードアドレスの取得	00000
リモート API 機能（rap）	CBLDCRAP('CONNECT')	rap リスナーとのコネクションの確立	00000
	CBLDCRAP('DISCNCT')	rap リスナーとのコネクションの解放	00000
ジャーナルデータの編集（jnl）	CBLDCJUP('CLOSERPT')	jnlrput 出力ファイルのクローズ	00000
	CBLDCJUP('OPENRPT')	jnlrput 出力ファイルのオープン	00000
	CBLDCJUP('RDGETRPT')	jnlrput 出力ファイルからジャーナルデータの入力	00000
性能検証用トレース（prf）	CBLDCPRF('PRFPUT')	ユーザ固有の性能検証用トレースの取得	00000
	CBLDCPRF('PRFGETN')	性能検証用トレース取得通番の通知	00000
メッセージ送受信（mcf）	CBLDCMCF('APINFO')	アプリケーション情報通知	00000
DAM ファイルサービス（dam）	CBLDCDMB('BSEK')	物理ファイルのブロックの検索	00000
	CBLDCDMB('DGET')	物理ファイルからブロックの直接入力	00000

機能名称	プログラム名(要求コード)	OpenTP1 提供関数の機能	ステータスコード
DAM ファイルサービス（dam）	CBLDCDMB('DPUT')	物理ファイルへブロックの直接出力	00000
XATMI インタフェース（tp～）	TPCALL	リクエスト型/レスポンス型サービスの呼び出しと応答の受信	TPOK※
	TPACALL	リクエスト型/レスポンス型サービスの呼び出し	TPOK※
	TPGETRPLY	リクエスト型/レスポンス型サービスからの非同期応答の受信	TPOK※
	TPCANCEL	リクエスト型，またはレスポンス型サービスのキャンセル	TPOK※
	TPCONNECT	会話型サービスとのコネクションの確立	TPOK※
	TPDISCON	会話型サービスとのコネクションの切断	TPOK※
	TPRECV	会話型サービスからのメッセージの受信	TPOK※
	TPSEND	会話型サービスへのメッセージの送信	TPOK※
	TPADVERTISE	サービス名の広告	TPOK※
	TPUNADVERTISE	サービス名の広告の取り消し	TPOK※
	TPSVCSTART	サービスルーチンの開始	TPOK※
	TPRETURN	サービスルーチンからのリターン	ステータスコードはありません。

注※

実行結果を示すリターン値が設定されるデータ領域（TP-STATUS）に，TPOK を設定します。

15 UAP トレースの使用方法

UAP トレースの使用方法について説明します。

15.1 UAP トレースの取得

UAP から呼び出した OpenTP1 の関数に関する情報を取得する機能を **UAP トレース**といいます。OpenTP1 では，UAP トレースを UAP トレースデータファイル，またはプロセスの固有領域に取得しています。

UAP が異常終了したときに UAP トレースを使用すると，UAP で発行したライブラリ関数の履歴を編集出力できます。編集出力した結果を基に，UAP が異常終了した原因を解析できます。

UAP トレースは，次のような UAP で使用できます。

- 異常終了した UAP
- dcstop -df コマンドで強制停止した UAP
- dcsvstop -df コマンドで強制停止した UAP
- prckill コマンドで強制停止した UAP

15.1.1 UAP トレースを取得する単位

UAP トレースは，**UAP プロセス単位**に取得しています。UAP プロセスごとに取得した UAP トレースデータファイル，または退避コアファイルを基に，UAP トレースを編集出力します。

UAP トレースを取得する対象の UAP は，**SUP**，**SPP**，および **MHP** です。

15.1.2 トレース領域の定義

UAP トレースで使用する領域の大きさは，ユーザサービス定義の uap_trace_max オペランドで指定します。ユーザサービス定義については，マニュアル「OpenTP1 システム定義」を参照してください。

15.1.3 取得する情報

UAP トレースでは，UAP から OpenTP1 のライブラリ関数を呼ぶときに，引数に設定した各種情報が取得されます。このうち，関数がリターンしたときの情報を，関数からの**出口情報**といいます。また，UAP からの関数呼び出しで OpenTP1 関数に入ったときの情報を，関数への**入り口情報**といいます。

オンラインテスタ（TP1/Online Tester）を使用している場合は，UAP トレースデータには実行したすべての関数の入り口情報と出口情報が取得されます。

また，オンラインテスタ（TP1/Online Tester）使用時に全入出力データの取得を指定した場合は，入出力データも取得されます。

15.2　UAP トレースの編集出力

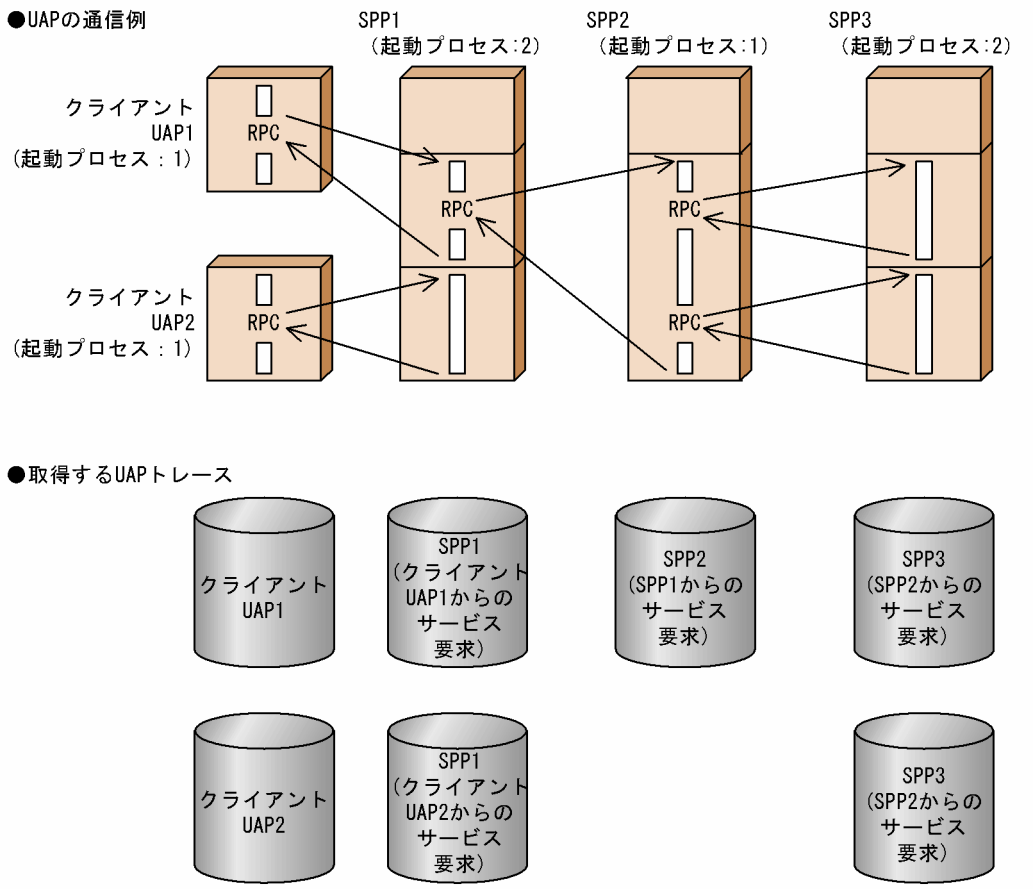
UAP トレースを編集出力する方法について説明します。

15.2.1　UAP トレースを編集出力する単位

UAP トレースを編集出力するのは，**UAP プロセス単位**です。複数の UAP プロセスに渡る処理の場合でも，異常終了した UAP プロセスで実行していたトランザクションブランチの情報だけを出力します。

UAP の通信形態と取得する UAP トレースの例を次の図に示します。

図 15–1　UAP の通信形態と取得する UAP トレースの例



15.2.2　UAP トレースを編集出力する方法

UAP トレースを編集出力するには，次の二つの方法があります。

(1) 自動でファイルに編集出力する

UAP プロセスごとに OpenTP1 が取得する異常終了情報を格納するファイルを，**退避コアファイル**といいます。UAP の異常終了時に退避コアファイルがあれば，UAP トレースはファイルに自動で編集出力されます。UAP トレースが編集出力されるファイルを，**UAP トレース編集出力ファイル**といいます。

退避コアファイルと UAP トレース編集出力ファイルのディレクトリとファイル名を，次の表に示します。

表 15–1　退避コアファイルと UAP トレース編集出力ファイルのディレクトリとファイル名

名称	ディレクトリ	ファイル名
退避コアファイル	$DCDIR/spool/save/※1	サーバ名 n※2
UAP トレース編集出力ファイル	$DCDIR/spool/save/	サーバ名 n.uat※2

注※1

プロセスサービス定義の prc_coresave_path オペランドを指定している場合，退避コアファイルは，prc_coresave_path オペランドに指定したディレクトリに退避します。

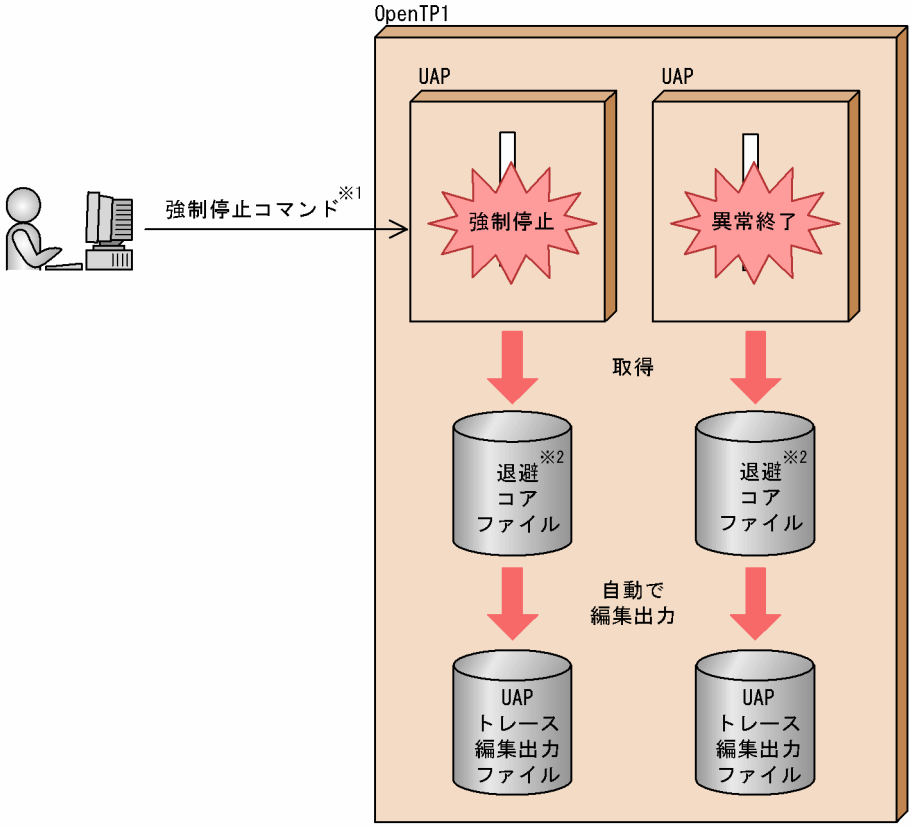
注※2

n：退避コアファイルの通番（1～3）

ただし，OpenTP1 の強制停止時（dcsvstop -df コマンドの実行時，または実時間監視タイムアウトになったとき）に出力される退避コアファイルには，通番は付きません。

UAP トレースを自動でファイルに編集出力する場合の概要を，次の図に示します。

図 15-2　UAP トレースを自動でファイルに編集出力する場合の概要



注※1

次に示すコマンドが該当します。
- 'dcsvstop -df'コマンド
- 'prckill'コマンド
- 'dcstop -df'コマンド

注※2

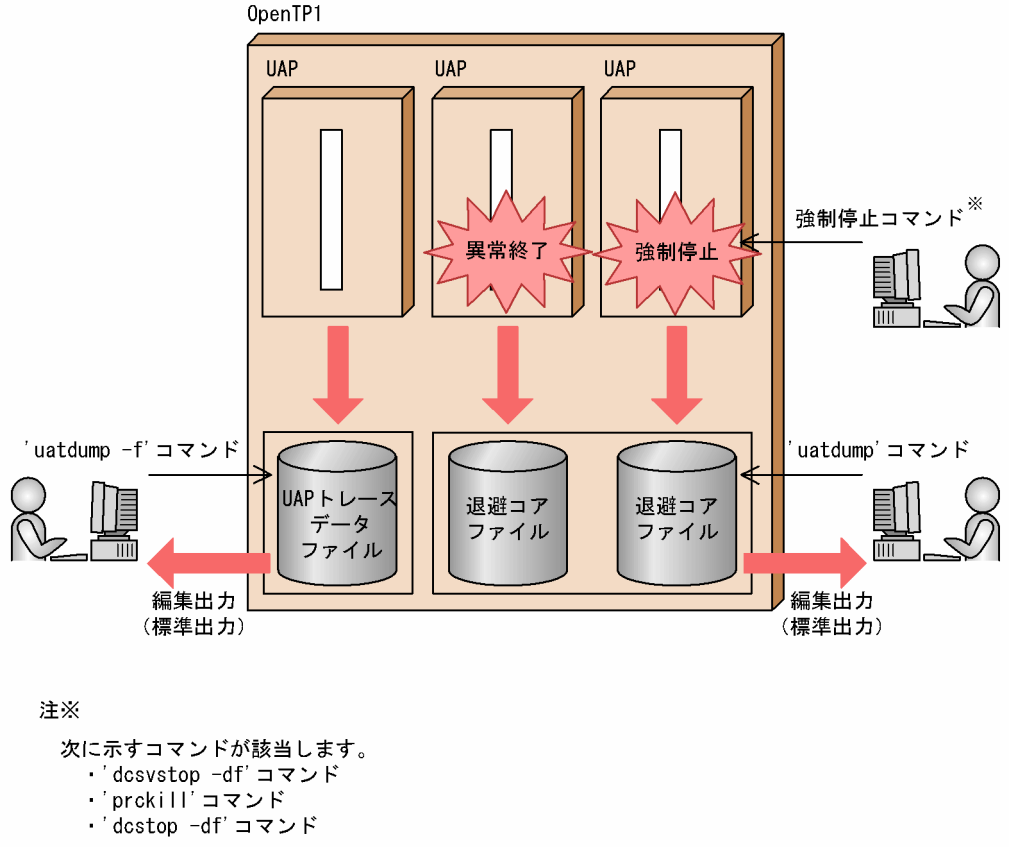
uap_trace_file_putオペランドにYを指定した場合，退避コアファイルではなくUAPトレースデータファイルが自動で編集出力されます。
uap_trace_file_putオペランドは，次のどれかの定義で指定します。
- システム共通定義
- ユーザサービスデフォルト定義
- ユーザサービス定義

(2)　コマンドで標準出力に編集出力する

uatdump コマンドを入力すると，UAP トレースは標準出力に編集出力されます。uatdump コマンドの使用方法については，「15.2.3　uatdump（UAP トレースの編集出力）」を参照してください。

UAP トレースをコマンドで標準出力に編集出力する場合の概要を，次の図に示します。

図 15-3　UAP トレースをコマンドで標準出力に編集出力する場合の概要



15.2.3　uatdump（UAP トレースの編集出力）

(1)　名称

UAP トレースの編集出力

(2)　形式

```
uatdump {〔退避コアファイル名〕 | -f 〔UAPトレースデータファイル名〕}
```

(3)　機能

指定した UAP トレースデータファイル，または退避コアファイルを編集して，標準出力に出力します。

オンラインテスタ（TP1/Online Tester）を使用しているノードでは，実行したすべての関数の出口情報と入り口情報を編集出力します。ただし，テスタ情報の出力は抑止するため，テスタ情報直後のデータが欠落する場合があります。

(4) オプション

- -f UAP トレースデータファイル名 ～<パス名>

  uap_trace_file_put オペランドに Y を指定して取得した UAP トレースデータファイルを編集出力するときに，そのファイル名をパス名で指定します。

  引数の指定を省略すると，UAP トレースデータファイル名には，コマンドを実行するカレントディレクトリ下の dcuat.map が仮定されます。

(5) コマンド引数

- 退避コアファイル名 ～〈パス名〉

  異常終了した UAP プロセスの退避コアファイルを，パス名で指定します。

  引数の指定を省略すると，退避コアファイル名には，コマンドを実行するカレントディクトリ下の core が仮定されます。

(6) 出力メッセージ

メッセージ ID	内容	出力先
KFCA03100-E	メモリ不足が発生しました。	標準エラー出力
KFCA03101-E	オプションフラグが不正です。	標準エラー出力
KFCA03102-E	指定したファイルはありません。	標準エラー出力
KFCA03103-E	指定したファイルにトレースデータがありません。	標準エラー出力
KFCA03104-W	トレースデータに不正な種別コードがあります。	標準エラー出力
KFCA03105-I	ヘルプメッセージ。	標準出力
KFCA03110-E	指定したファイルのトレースデータが破壊されています。	標準エラー出力

(7) 出力形式

uatdump コマンドの出力形式については，「15.2.4 UAP トレースの編集出力形式」を参照してください。

(8) 注意事項

uap_trace_file_put オペランドに Y を指定した場合，次の注意事項があります。

- 取得した UAP トレースデータファイルを編集出力するときは-f オプションを指定してください。

  -f オプションを指定しないと，UAP トレースが編集できないため，コマンドがエラーになります。
- uatdump コマンドに-f オプションを指定して実行した場合，UAP への影響を抑えるため，UAP トレースデータファイルを UAP と同期を取らずに参照します。このため，UAP が動作している最中に uatdump コマンドを実行すると，タイミングによっては，最新の情報を取得できないことがあります。

- uatdump コマンドに-f オプションを指定して実行した場合，UAP と同期を取らずに UAP トレースデータファイルを参照するため，KFCA03104-W や KFCA03110-E メッセージが出力されることがあります。この場合，再度，-f オプションを指定して uatdump コマンドを実行してください。
- UAP が動作している最中に uatdump コマンドを実行すると，UAP の性能に影響をおよぼします。

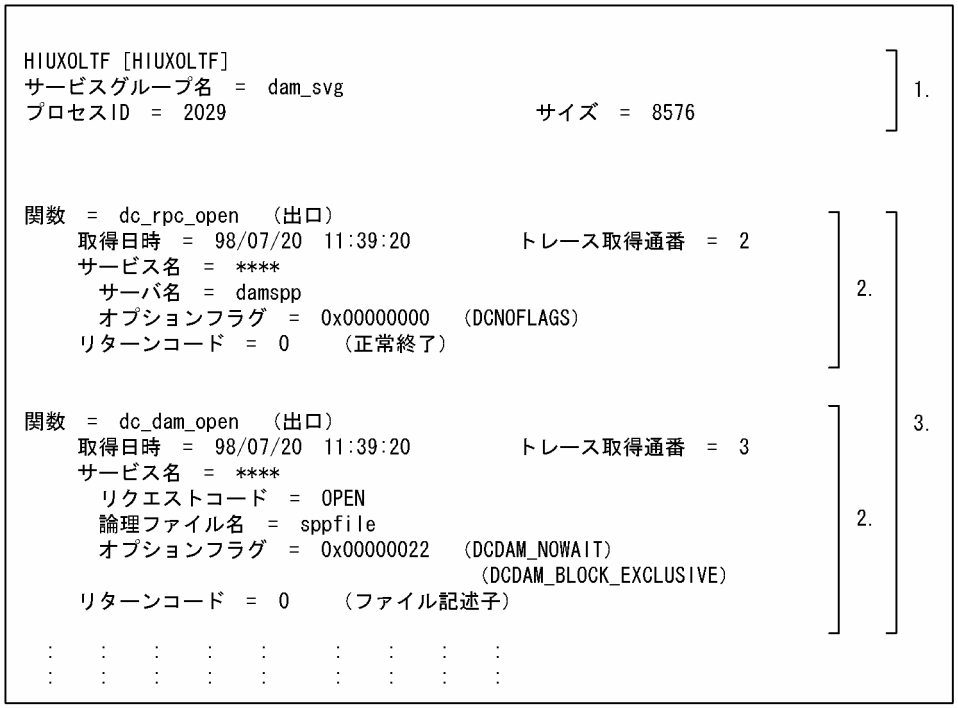
uap_trace_file_put オペランドは，次のどれかの定義で指定します。

- システム共通定義
- ユーザサービスデフォルト定義
- ユーザサービス定義

15.2.4 UAP トレースの編集出力形式

UAP トレースを自動でファイルに編集出力した場合，および uatdump コマンドで標準出力に編集出力した場合の編集出力形式を，次に示します。

(1) 出力形式



(凡例)

1. UAP トレースヘッダ
2. UAP トレースデータ

   オンラインテスタを使用している場合は，入り口情報と出口情報が交互に出力されます。出口情報の場合は（出口），入り口情報の場合は（入口）と表示されます。

3. OpenTP1 関数の呼び出し情報の出力領域

出力領域に出力される内容は，発行した関数によって異なります。

(説明)

サービスグループ名

起動していたサービスがあるサービスグループ名

SUP，MHP の場合はアスタリスクが表示されます。

プロセス ID

該当の UAP トレースを取得したプロセスのプロセス ID

サイズ

UAP トレース情報領域の大きさ（10 進数　単位：バイト）

関数

発行した OpenTP1 関数

取得日時

UAP トレースを取得した日時

西暦下 2 けた/月/日 △ 時:分:秒の形式

トレース取得通番

UAP トレースデータを取得したときに付けた一連の番号（最大 6 けた）

サービス名

起動していたサービス名（最大 32 文字）

SUP，MHP の場合はアスタリスクが表示されます。

リターンコード

OpenTP1 関数の実行結果

(2) 出力例

```
HIUXOLTF[HIUXOLTF]
サービスグループ名 = dam_svg
プロセスID = 2029                          サイズ = 8576



関数 = dc_rpc_open （出口）
   取得日時 = 98/07/20 11:39:20          トレース取得通番 = 2
   サービス名 = ****
     サーバ名 = damspp
     オプションフラグ = 0x00000000（DCNOFLAGS）
   リターンコード = 0  （正常終了）

関数 = dc_dam_open （出口）
   取得日時 = 98/07/20 11:39:20          トレース取得通番 = 3
   サービス名 = ****
     リクエストコード = OPEN
```

```
     論理ファイル名 = sppfile
     オプションフラグ = 0x00000022 (DCDAM_NOWAIT)
                                (DCDAM_BLOCK_EXCLUSIVE)
   リターンコード = 0   (ファイル記述子)

関数 = XATMIサービス関数開始情報
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 4
   サービス名 = REFSVC_A
     ノード名 = 2c3gfm01
     サービス名 = REFSVC_A
     受信タイプ名 = X_C_TYPE
     受信サブタイプ名 = sub1
     ------ 受信データ ------
     00008c    534b492d 50415241 44494345 00000000   SKI- PARA
     DICE ....
     00009c    00000000 00000000 00000000 00000000   .... ....
     .... ....
     受信データ長 = 104
     フラグ = 0x00000000
     ディスクリプタ = 0

関数 = tpconnect  (出口)
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 5
   サービス名 = REFSVC_A
     サービス名 = REFSVC_C
     送信タイプ名 = X_C_TYPE
     送信サブタイプ名 = sub1
     ------ 送信データ ------
     00006c    534b492d 50415241 44494345 00000000   SKI- PARA
     DICE ....
     00007c    00000000 00000000 00000000 00000000   .... ....
     .... ....
     送信データ長 = 50
     フラグ = 0x00001000 (TPRECVONLY)
   リターンコード = 1390287197

関数 = tprecv  (出口)
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 6
   サービス名 = REFSVC_A
     ディスクリプタ = 1390287197
     受信タイプ名 = X_C_TYPE
     受信サブタイプ名 = sub1
     ------ 受信データ ------
     000050    4e4f5254 482d464c 49474854 00000000   NORT H-FL
     IGHT ....
     000060    00000000 00000000 00000000 00000000   .... ....
     .... ....
     受信データ長 = 104
     フラグ = 0x00000000
     イベント = 0x0008   (TPEV_SVCSUCC)
     tperrno = 22 (TPEEVENT)
     tpurcode = 0x00000000
   リターンコード = -1

関数 = tpreturn  (出口)
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 7
   サービス名 = REFSVC_A
     リターンバリュー = 0x04000000        ユーザリターン値 = 22
```

```
     送信タイプ名 = X_C_TYPE
     送信サブタイプ名 = sub1
     ------ 送信データ ------
     000054    4e4f5254 482d464c 49474854 00000000   NORT H-FL
     IGHT ....
     000064    00000000 00000000 00000000 00000000   .... ....
     .... ....
     送信データ長 = 50
     フラグ = 0x00000000


関数 = XATMIサービス関数終了情報
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 8
   サービス名 = REFSVC_A
     ノード名 = 2c3gfm01
     サービス名 = REFSVC_A
     送信タイプ名 = X_C_TYPE
     送信サブタイプ名 = sub1
     ------ 送信データ ------
     00008c    4e4f5254 482d464c 49474854 00000000   NORT H-FL
     IGHT ....
     00009c    00000000 00000000 00000000 00000000   .... ....
     .... ....
     送信データ長 = 104

関数 = XATMIサービス関数開始情報
   取得日時 = 98/07/20 11:39:25           トレース取得通番 = 9
   サービス名 = REFSVC_B
     ノード名 = 2c3gfm01
     サービス名 = REFSVC_B
     受信タイプ名 = X_C_TYPE
     受信サブタイプ名 = sub1
     ------ 受信データ ------
     00008c    534b492d 50415241 44494345 00000000   SKI- PARA
     DICE ....
     00009c    00000000 00000000 00000000 00000000   .... ....
     .... ....
     受信データ長 = 104
     フラグ = 0x00000c00 (TPSENDONLY)
                         (TPCONV)
     ディスクリプタ = 1028921693
   :  :  :  :  :   :  :  :  :
   :  :  :  :  :   :  :  :  :
```

索引

W

X

あ

い

う

え

お

か

く

け

こ

さ

し

す

せ

そ

た

ち

て