Novell exteNd Composer™ T27 Connect

4.2

www.novell.com

ユーザガイド

Novell®

保証と著作権

米国 Novell, Inc.
1800 South Novell Place
Provo, UT 85606

www.novell.com

T27 Connect ユーザガイド
2003 年 5 月

オンラインマニュアル： この製品およびその他の Novell 製品のオンラインマニュアルや更新情報については、www.novell.com/documentation を参照してください。

Novell の商標

Novell および jBroker は Novell, Inc. の登録商標です。また Novell exteNd は Novell, Inc. の商標です。

サードパーティ商標

Sun Microsystems, Inc. Sun, Sun Microsystems, Sun Logo Sun、Sun のロゴ、Sun Microsystems、JavaBeans、Enterprise JavaBeans、JavaServer Pages、Java Naming and Directory Interface、JDK、JDBC、Java、HotJava、HotJava Views、Visual Java、Solaris、NEO、Joe、Netra、NFS、ONC、ONC+、OpenWindows、PC-NFS、SNM、SunNet Manager、Solaris sunburst design、Solstice、SunCore、SolarNet、SunWeb、Sun Workstation、The Network Is The Computer、ToolTalk、Ultra、Ultracomputing、Ultraserver、Where The Network Is Going、SunWorkShop、XView、Java WorkShop、Java Coffee Cup のロゴ、Visual Java、および NetBeans は、米国およびその他の国の Sun Microsystems, Inc. の商標ならびに登録商標です。

目次

このガイドについて

目的

このガイドでは、T27 コンポーネントエディタと呼ばれる exteNd Composer T27 Connect の使用方法について説明します。T27 コンポーネントエディタは、exteNd Composer で個別にインストールされるコンポーネントエディタです。

対象読者

このガイドの対象読者は、exteNd Composer を使用して、T27 アプリケーションを統合する Web サービスやコンポーネントを作成する開発者およびシステムインテグレータです。

前提条件

このガイドでは、読者が exteNd Composer の開発環境および配備ツールに精通しており、これらを使用したことがあるという前提で説明を進めます。また、T27 環境、および T27 を利用したアプリケーションの作成方法または使用方法を理解しておく必要もあります。このガイドをお読みになる上で、UTS、3270、5250、VT シリーズ端末 (VT100 など) といった他のメインフレーム端末エミュレータに関する理解も多少必要になります。

追加のマニュアル

Novell exteNd Composer の完全なマニュアルのセットについては、Novell マニュアルの Web サイト (**http://www.novell.com/documentation-index/index.jsp**) を参照してください。

構成

このガイドは、次のように編成されています。

第 1 章「exteNd Composer および T27 User Interface へようこそ」では、T27 Connect およびコンポーネントエディタの定義および概要について説明し、さらにこれらを使用して作成できるアプリケーションのタイプについても説明します。

第 2 章「T27 コンポーネントエディタをお使いになる前に」では、T27 コンポーネントを作成するために必要な準備について説明します。

第 3 章「T27 コンポーネントの作成」では、コンポーネントエディタの異なる部分について説明します。

第 4 章「T27 アクションの実行」では、基本的な T27 アクションの使用方法、および T27 Connect 固有の機能について説明します。

第 5 章「アクションでの T27 コンポーネント」では、アクションモデルでサンプルアプリケーションを使用して、T27 コンポーネントおよびアクションを使用する方法について示します。

第 6 章「ログオンコンポーネント、接続、および接続プール」では、共有接続を使用してパフォーマンス向上させる方法について説明します。

付録 A は、用語集です。

付録 B では、「T27 属性」およびその表示の意味、さらに `getattribute( )` の使用方法について説明します。

付録 C「予約語」では、T27 Connect でのみ使用される予約語のリストを示します。

このガイドで使用する表記規則

このガイドで使用する表記規則は、次のとおりです。

手順での「太字」フォントは、次のアクション項目を示します。

- メニューの選択
- フォームの選択
- ダイアログボックス項目

太字の **Sans Serif** フォントは、次の項目に使用します。

- Uniform Resource Identifier
- ファイル名
- ディレクトリおよび部分的なパス名

「斜体」のフォントは、次の項目を示します。

- 入力する変数情報
- 新出の技術用語
- 他の Novell 出版物のタイトル

「モノスペース」フォントは、次の項目を示します。

- メソッド名
- コードの例
- システム入力
- オペレーティングシステムオブジェクト

1 exteNd Composer および T27 Connect へようこそ

はじめに

『T27 Connect ガイド』へようこそ。このガイドは、exteNd Composer の全機能 (Connect コンポーネントエディタを除く) の使用方法が詳しく説明されている『exteNd Composer ユーザガイド』に付属しています。『Composer ユーザガイド』をご覧になっていない場合は、このガイドを使用する前に読んで内容を確認してください。

exteNd Composer には、Connect ごとに異なるコンポーネントエディタが用意されています。各コンポーネントエディタの特殊な機能は、これと同じような別のガイドで説明されています。

exteNd Composer を使用しており、XML Map コンポーネントエディタに精通している場合は、このガイドに従って T27 コンポーネントエディタを簡単に使用することができます。

作業を始める前に、まず T27 Connect を既存の exteNd Composer にインストールしておく必要があります。また、この Connect で作成されたサービスを exteNd Composer Enterprise Server 環境で実行するには、この Connect 用のサーバ側ソフトウェアが Composer Enterprise Server にインストールされている必要があります。

注記：このコンポーネントエディタを正しく使用するには、T27 環境および XML を使用する特定のアプリケーションについて理解を深めておく必要があります。

exteNd Composer Connect について

exteNd Composer は、単純なハブ & スポークアーキテクチャに基づいて構築されています (図 1-1)。ハブは、XML ドキュメントから要求を受け入れ、XML に対応したアプリケーション上でこのようなドキュメントやインタフェースで変換プロセスを実行し、XML 応答ドキュメントを返す強力な XML 変換エンジンです。スポーク (つまり Connect) は、XML 対応でないデータのソースを「XML に対応させる」プラグインモジュールで、データをハブに送信して XML として処理し

ます。これらのデータソースには、レガシー COBOL/ アプリケーションから、HTML ページに対するメッセージキューまで何でも使用できます。

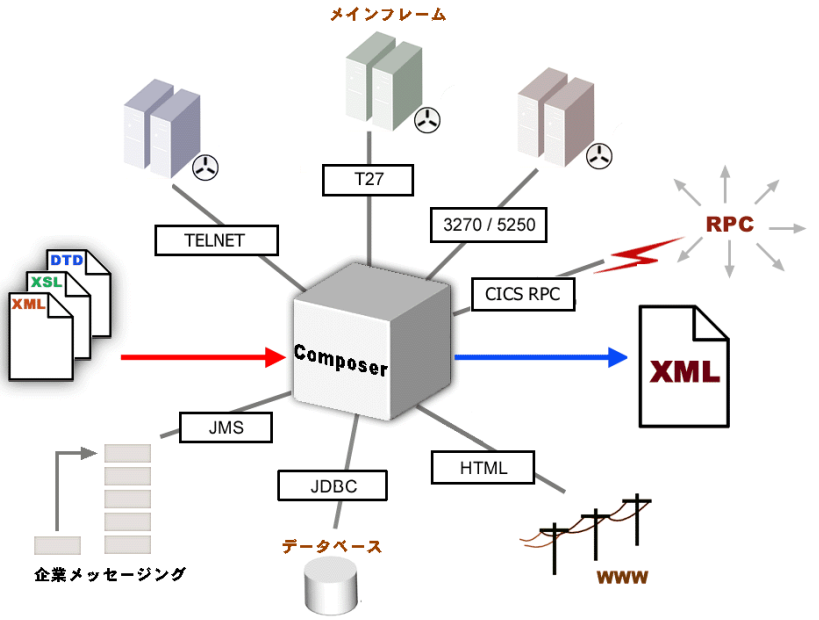
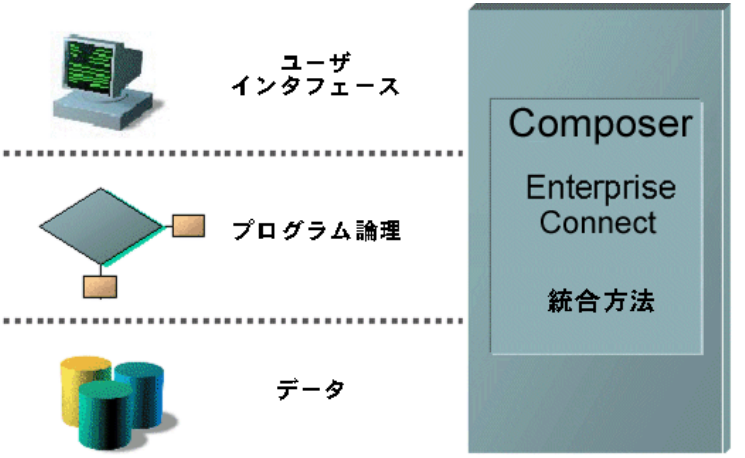


図 1-1

exteNd Composer Connect は、情報ソースを XML に対応させるために各製品で使用されている統合方法に従って分類できます。統合方法は、インターネットベースのコンピュータアーキテクチャに対する現在のシステム設計において使用される主要な区分を反映したものです。exteNd Composer では、B2B のニーズとレガシーアプリケーションのアーキテクチャに応じて、ユーザインタフェースレベル、プログラム論理レベル、またはデータレベルでビジネスシステムを統合できます (次の図を参照してください)。



T27 Connect とは

端末データストリームにフッキングすることによってUser Interfaceの統合方法を使用する、T27 Connectの XML に対応した Unisys ホストシステムデータです。

T27 の端末は、A Series、V Series、および ClearPath™ NX など人気の高い Unisys メインフレームモデルと通信するために使用されます。1980 年代に個人用コンピュータが広く普及するまで、企業では重要な情報の保存とアクセスについて、このような大容量のメインフレームシステムにかなり依存していました。

T27 Connect を使用すると、レガシーアプリケーションやそれらのビジネス論理をインターネット、エクストラネット、またはイントラネットに Web サービスとして利用できるようにすることができます。T27 Connect コンポーネントエディタを使用すると、端末セッションでの操作と同じように単純にアプリケーション内を移動することによって Web サービスを作成できます。また、キーではなく XML ドキュメントを使用して問い合わせや更新を画面に表示したり、アプリケーション画面から返されたメッセージを使用して端末と同様に同じ決断を下したり、要求者に返すことができる、あるいは処理を続行できる XML ドキュメントにデータや応答を移動したりできます。T27 の画面は、T27 コンポーネントエディタのネイティブ環境ペインに表示されます。

exteNd Composer の T27 コンポーネントについて

XML Map コンポーネントと同様、T27 コンポーネントは 2 つの異なる XML テンプレート (つまり、要求 XML ドキュメントと応答 XML ドキュメント) 間でデータをマップ、変換、および転送するために設計されています。ただし、Unisys T27 ホストアプリケーションに接続し、画面からの要素を使用してデータを処理して

からその結果を出力 DOM にマップします。その後、統合アプリケーションに適切な方法で、出力 DOM に基づいて処置を取ることができます。本質的には、ホストシステム自体を変更せずに、ホストシステムからデータをキャプチャしたり、ホストシステムにデータをプッシュすることが可能です。

T27 コンポーネントでは、単純なデータ操作 (XML ドキュメントからホストプログラムへのデータのマップや転送など) を実行したり、取得したデータを XML ドキュメントに配置して T27 トランザクションの「スクリーンスクレーピング」を実行したりすることができます。T27 コンポーネントには XML Map コンポーネントの全機能が備わっており、XSL の処理、メールの送信、および HTTP プロトコルを使用した XML ドキュメントのポストと受信を実行できます。

T27 Connect を使用して作成できるアプリケーション

exteNd Composer および T27 Connect は、次のタイプのアプリケーションに適用できます。

1. サプライチェーンアプリケーションなどのビジネス間の Web サービス処理
2. Web ブラウザからのセルフサービスアプリケーションなどのコンシューマとビジネス間の処理
3. 異種システムからの情報が組み合わされ、結合される企業アプリケーションの統合

基本的に、T27 コンポーネントエディタを使用すると、作成中の任意の XML 統合を拡張して、T27 ベースの端末操作をサポートする任意のビジネスアプリケーションを含めることができます (詳細については、『exteNd Composer ユーザガイド』を参照してください)。

たとえば、定期的に更新されるデータベースから製品の説明、画像、価格、および在庫情報を取得して Web ブラウザに表示するアプリケーションがあるとします。T27 コンポーネントエディタを使用すると、動作中のシステムから最新の製品情報を取得したり、データベースから静的情報 (画像など) を取得し個別の情報ソースから情報をマージして、ユーザに対して表示できます。これにより、内部ユーザと外部ユーザの両方に同じ最新の情報が提供されます。

2 T27 コンポーネントエディタをお使いになる前に

T27 コンポーネントの作成に使用される一般的な手順

T27 コンポーネントを作成する方法はいくつもありますが、単純なコンポーネントの作成に使用される一般的な手順は次のとおりです。

- プログラム用の XML テンプレートを作成する
- T27 接続リソースを作成する
- T27 コンポーネントを作成する
- 記録モードに設定し、コンポーネントエディタのネイティブ環境画面で利用可能な端末エミュレーションを使用してプログラムに移動する
- 必要に応じて入力ドキュメントデータを画面にドラッグアンドドロップする
- 画面の結果を出力ドキュメントにドラッグアンドドロップする
- 記録を停止する

この章では、このプロセスの最初の 2 つについて説明します。

コンポーネントに対する XML テンプレートの作成

必須というわけではありませんが、コンポーネントを設計するためのサンプルドキュメントを利用できるように、T27 コンポーネントでは XML テンプレートの作成が必要な場合もあります (詳細については、『exteNd Composer ユーザガイド』の第 5 章「XML テンプレートの作成」を参照してください)。

多くの場合、入力ドキュメントは、端末オペレータがプログラムにインタラクティブに入力する可能性のあるデータを含めるために設計されます。同様に、出力ドキュメントは、オペレータの入力の結果として画面に返されるデータを受信するために設計されます。たとえば、一般的なビジネスシナリオでは、端末オペレータは、アイテムの価格や購入が可能であるかどうかについて興味のある顧客から電話による要求を受け取ります。オペレータは通常、プロンプトが表示されると、端末に情報 (パーツ番号など) を入力することによって T27 端末セッショ

ンからホストシステムに照会します。その後まもなく、ホストは端末画面にデータを返すことによって応答し、オペレータはこの情報を顧客にリレーします。このセッションは、T27 コンポーネントを使用する exteNd Composer Web サービスで実行できます。要求されたパーツ番号は、XML 入力ドキュメントのデータ要素として表すことができます。ホストから返された照会データは、コンポーネントの「出力」ドキュメントに表示されます。このデータは、次に Web ページに出力されたり、または XML などとして別のビジネスプロセスに送信されたりします。

注記： コンポーネントの設計により、カスタムスクリプトまたはコードテーブルのマップなど xObject リソースが必要になる場合があります。その場合、T27 コンポーネントを作成する前に、これらのオブジェクトを作成することをお勧めします。詳細については、『exteNd Composer ユーザガイド』を参照してください。

T27 接続リソースの作成

XML テンプレートを作成した後、次の手順はホストプログラムにアクセスするための接続リソースを作成することです。利用可能な接続リソースなしで T27 コンポーネントを作成しようとすると、ダイアログボックスが表示され、接続リソースを作成するかどうか尋ねられます。ダイアログボックスで［Yes］を選択すると、適切なウィザードが開始します。

接続リソース

T27 コンポーネントの接続リソースを作成する場合、ストレート接続、ログオン接続、およびマルチブリッジ接続の 3 つの選択があります。一般的に、ホスト環境に接続する場合、ストレートな T27 接続を使用します。ログオン接続は接続のプールに使用されます (詳細については、このガイドの第 6 章で説明します)。マルチブリッジ接続は、ホストに戻る接続の数を最小化するゲートウェイのサーババージョンであり、追加されたセキュリティも含まれています。マルチブリッジ接続は、Novell およびサードパーティのビジネスパートナーを介して特別に有効にする必要があります。アプリケーションでマルチブリッジ接続を使用する必要がある場合は、exteNd テクニカルサポートにお問い合わせください。

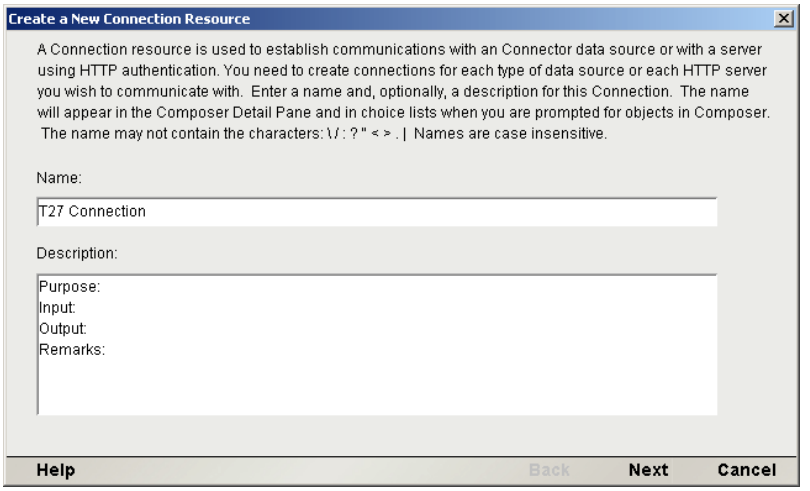
T27 接続リソースは、設定後、ホストへの接続が必要な任意の数の T27 コンポーネントで利用可能になります。

➢ T27 接続リソースを作成する

1 Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、［**Connection**］の順に選択します。

注記： または、Composer ウィンドウのカテゴリペインの［**Resource**］で、［**Connection**］を選択し、マウスを右クリックして［**New**］を選択することもできます。

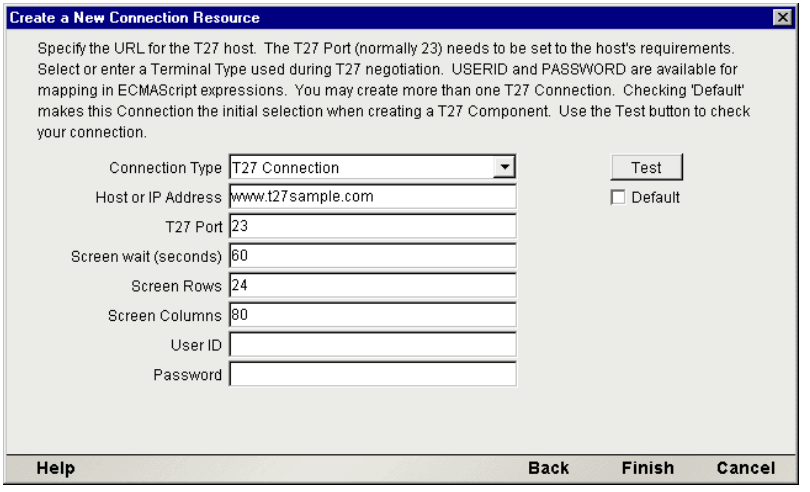
Create a New Connection Resource ウィザードが表示されます。



2 ［**Name**］に、接続オブジェクトの名前を入力します。

3 オプションとして、［**Description**］に説明テキストを入力します。

4 ［**Next**］をクリックします。ウィザードの 2 番目のパネルが表示されます。



5 プルダウンメニューから［**T27 Connection**］を選択します。ダイアログボックスの外観が変わり、T27 接続の作成に必要なフィールドだけが表示されます。

6 ［**Host or IP Address**］フィールドに、接続先のマシンの物理 (IP) アドレスまたはホスト名の別名を入力します。

7 ［**Port**］フィールドに、T27 ポートの番号を入力します。デフォルトのポート番号は 23 です。

8 ［**Screen Wait (seconds)**］フィールドに、T27 端末コンポーネントが［Check Screen Action］ペインで次の画面を待つ時間を秒単位で入力します (デフォルト値として設定されます)。

9 ［**Screen Rows**］フィールドに、1 画面あたりのデフォルトの行数を指定します。

10 ［**Screen Columns**］フィールドに、1 画面あたりのデフォルトの列数を指定します。

11 ［**UserID**］および［**Password**］に、それぞれユーザ ID とパスワードを入力します。これらは、接続の確立中にホストに対して実際には送信されず、単にここで定義されます (パスワードは暗号化されます)。これらのフィールドを式駆動型にする場合は、マウスを右クリックして［**Expression**］を選択します。

注記 : このダイアログボックスでユーザ ID とパスワードの情報を入力すると、ECMAScript グローバル変数の USERID と PASSWORD でこれらの値が指定されるようになります。そうすると、これらの変数を Set Screen Text 式で (または、第 3 章の「ネイティブ環境ペインのコンテキストメニュー」で説明されているように) 使用できます。

12 この特定の T27 接続を後続の T27 コンポーネントのデフォルトの接続にする場合は、［**Default**］チェックボックスをオンにします。

13 ［**Finish**］をクリックします。新しく作成されたリソース接続オブジェクトが、Composer 接続リソースの詳細ペインに表示されます。

定数駆動型および式駆動型の接続

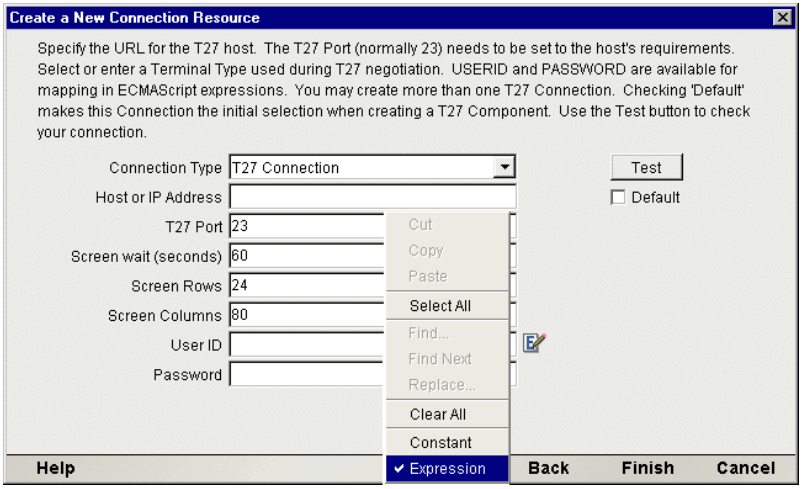
接続パラメータの値は、定数または式のいずれかの方法で指定できます。「定数ベース」のパラメータでは、接続が使用されるたびに［Connection］ダイアログボックスで指定する静的な値を使用します。「式ベース」のパラメータでは、ランタイム時に接続が使用されるたびに「異なる」値となりえるプログラム的な式 (つまり、ECMAScript 式) を使用して該当する値を設定できます。これにより、接続の動作は柔軟になり、ランタイム条件に基づいて変えることができます。

たとえば、T27 接続で式駆動型のパラメータを使用する非常に単純な例の 1 つは、ユーザ ID とパスワードを PROJECT 変数 (例 : PROJECT.XPath("USERCONFIG/MyDeployUser")) として定義することです。このようにすると、プロジェクトを配備する際、最終的な配備環境に適した値に Deployment ウィザードで PROJECT 変数を更新できます。正反対に、アプリケーションサーバで Java ビジネスオブジェクトをクエリするカスタムスクリプトを利用して、使用するユーザ ID とパスワードを決定することもできます。

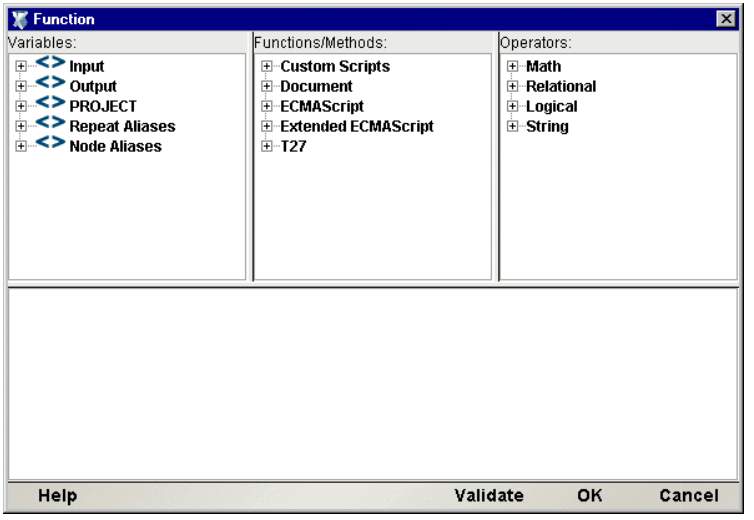
➢ 定数駆動型から式駆動型にパラメータを切り替える

1 変更するパラメータフィールドでマウスを右クリックします。

2 コンテキストメニューから［**Expression**］を選択して、［Expression Editor］ボタンを表示するか、有効にします。次の図を参照してください。



3 ［**Expression Editor**］ボタンをクリックします。［Expression Editor］が表示されます。



4 ランタイム時に有効なパラメータの値を評価する式を作成します（ または、ウィンドウの上部にある選択リストを使用します）。［**OK**］をクリックします。

3 T27 コンポーネントの作成

T27 コンポーネントの作成

前の章で説明したように、T27 コンポーネントを作成する前に、コンポーネントで必要な XML テンプレートを最初に準備しておく必要があります (詳細については、『Composer ユーザガイド』の「新しい XML テンプレートの作成」を参照してください)。コンポーネントを作成する際には、これらのテンプレートのサンプルドキュメントを使用して、コンポーネントにより処理される入力および出力を表します。

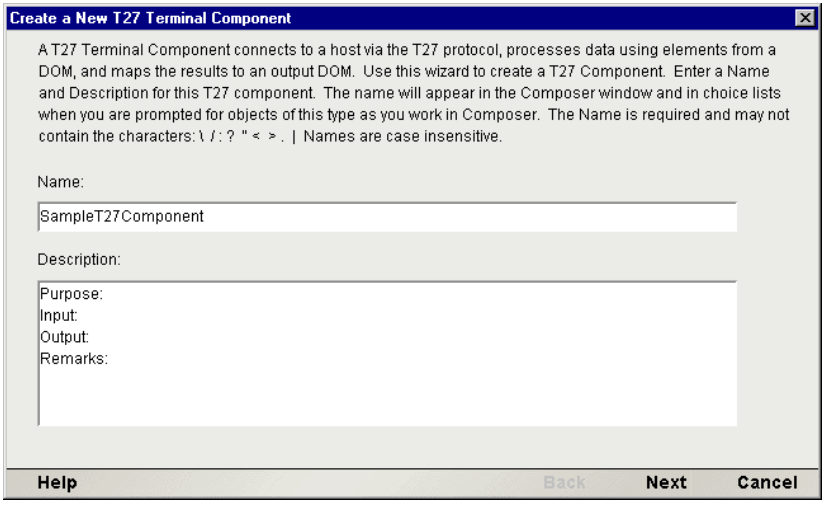
また、T27 コンポーネントの作成プロセスの一環として、コンポーネントで使用する T27 接続を指定する必要があります (新しく作成することもできます)。T27 接続リソースの作成については、前の章を参照してください。

➢ **新しい T27 コンポーネントを作成する**

1 ［**File**］>［**New xObject**］>［**T27 Terminal**］の順に選択します。

注記 : または、Composer ウィンドウのカテゴリペインの［**Component**］で、［**T27 Terminal**］を選択し、マウスを右クリックして［**New**］を選択することもできます。

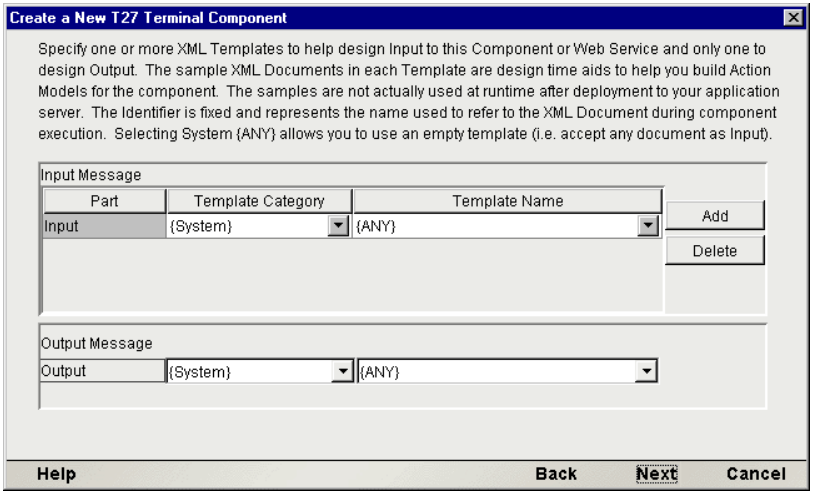
2 「Create a New T27 Component」ウィザードが表示されます。



3 新しい T27 端末コンポーネントの「名前」を入力します。

4 オプションとして、「説明」テキストを入力します。

5 ［**Next**］をクリックします。New T27 Component ウィザードの XML プロパティ情報パネルが表示されます。



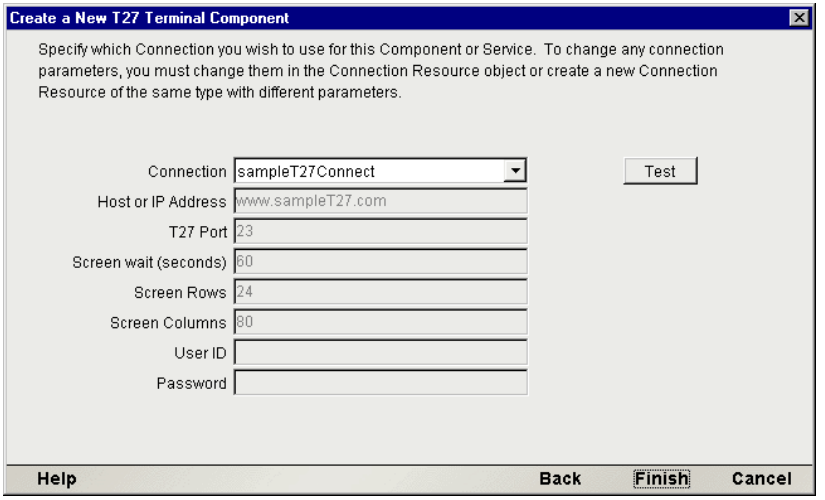
6 「入力」テンプレート (複数可) を指定します。デフォルトのカテゴリと異なる場合は、テンプレートカテゴリを選択します。その後、選択したテンプレートカテゴリにある XML テンプレートのリストからテンプレート名を選択します。

注記： テンプレートとして **{System}{ANY}** を選択すると、定義済みの構造が含まれない入力 XML テンプレートまたは出力 XML テンプレートを指定できます。詳細については、『Composer ユーザガイド』の「テンプレートを使用しない出力 DOM の作成」を参照してください。

7 入力 XML テンプレートをさらに追加するには、［**Add**］をクリックして、それぞれにテンプレートカテゴリとテンプレート名を選択します。この手順を必要なだけ繰り返します。入力 XML テンプレートを「削除する」には、エントリを選択して［**Delete**］をクリックします。

8 出力として使用する XML テンプレートを選択します (出力 DOM の名前は「Output」です)。

9 ［**Next**］をクリックします。Create a New T27 Component ウィザードの接続情報パネルが表示されます。



10 プルダウンリストで「接続」の名前を選択します。T27 接続の詳細については、第 2 章の「T27 接続リソースの作成」を参照してください。

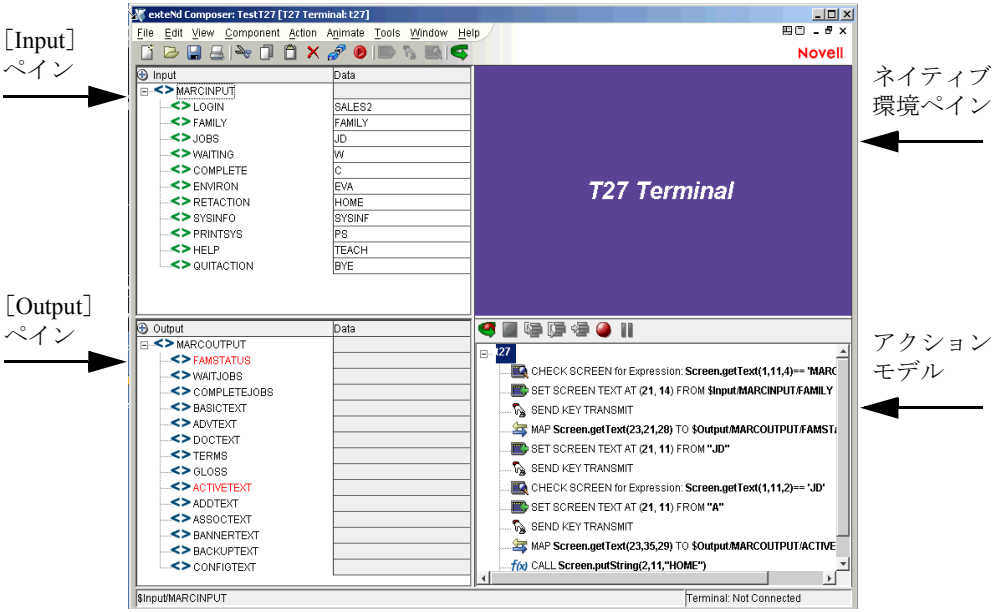
11 ［**Finish**］をクリックします。コンポーネントが作成され、T27 コンポーネントエディタが表示されます。

T27 コンポーネントエディタウィンドウについて

T27 コンポーネントエディタには、exteNd Composer の XML Map コンポーネントエディタの機能がすべて含まれています。たとえば、入力 XML ドキュメントと出力 XML ドキュメントのマッピングペインや、アクションペインも含まれています。

ただし、大きな違いが 1 つあり、T27 コンポーネントエディタには、T27 エミュ

レータを特徴とするネイティブ環境ペインも含まれます。この画面は、メインツールバーで［Connection］アイコンをクリックするか、ツールバーで［Record］ボタンをクリックして記録を開始するまで黒色で表示されます。いずれの操作でも、ネイティブ環境ペイン内に T27 エミュレーションセッションが確立されます。このセッションのホストは、T27 コンポーネントにより使用される接続リソースで指定したホストです。



T27 ネイティブ環境ペインについて

T27 ネイティブ環境ペインでは、ホスト環境の T27 エミュレーションが提供されます。このペインでは、Unisys メインフレームに接続された端末の画面を操作する場合とまったく同じようにネイティブ環境ペインを操作して、T27 セッションをリアルタイムで実行できます。また、次の操作を行うこともできます。

- T27 画面フィールドに対する入力として、入力 XML ドキュメント (または他の使用可能な DOM) からのデータを使用する。たとえば、SKU 番号を入力 DOM から T27 画面の「パーツ番号」フィールドにドラッグして、ホストを照会し、そのパーツ番号に関連付けられているデータ (説明や価格など) を返すことができます。
- 返された T27 画面からデータをマップして、出力 XML ドキュメント (または、Temp、MyDom などの他の使用可能な DOM) に配置する。

- ヘッダ情報および詳細情報 (複数の品目から成るフォームなど) を、ECMAScript 式または関数を使用してネイティブ環境ペインから XML ドキュメントにマップする。

T27 のキーボードサポート

T27 ネイティブ環境ペインでは、Clear Home、Local、Previous Page、Specify、Forms Mode Toggle、Next Page、Receive、Transmit などの、いくつかの特別なアテンションキーの使用がサポートされています。各アテンションキーの機能は、ホストアプリケーションにより異なります。これらのキーは、PC キーボードにマップされます (次を参照)。

表 1-1:

T27 キー	PC キー
Clear Home	Ctrl+Home
Local	F10 または F6
Previous Page	PageUp
Specify	F9 または F5
Forms Mode、Toggle	Esc、Alt+S、または O
Next Page	PageDown
Receive	F11 または F7
Transmit	F12 または F8

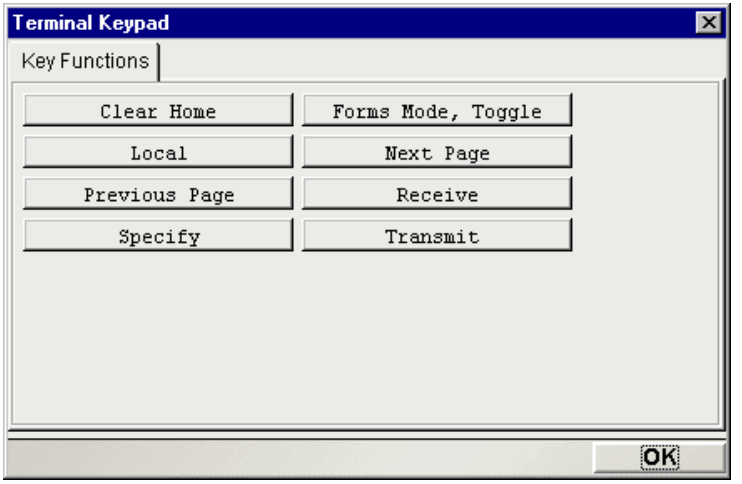
T27 コンポーネントを作成する際、キーボードのキーを直接使用するか、［View］メニューのキーボードツールバーを使用できます。

➢ フローティングテンキーの使用方法

1. Composer メニューから［**View/Terminal Keypad**］を選択します。フローティングテンキーが表示されます。

2. 呼び出すキーをクリックします。ヘルプが必要な場合は、キーの上にマウスを移動させます。マウスを合わせたキーに該当する T27 テンキーが表示されます。ここでは、ネイティブ環境ペインでキーをクリックした場合の結果を確認できます。

3 ［**OK**］をクリックして、テンキーを閉じます。テンキーを再表示するには、手順 1 を繰り返します。



Screen オブジェクトについて

Screen オブジェクトは、ネイティブ環境ペインに示されたエミュレータ画面のバイト配列表記で、画面のコンテンツを操作するためのメソッドが備わっています。

説明

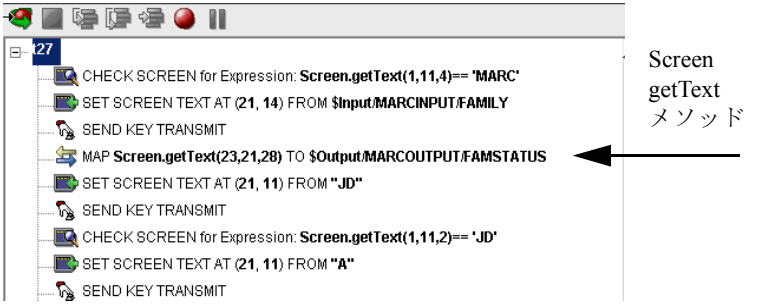
T27 コンポーネントは、T27「セッション」でブロックモードの端末データストリームを介してホスト環境との通信を行います。データのブロックは、基本的に画面を表します。ホストでは、コンポーネントに表示される画面ブロックを送信します。この画面は、ユーザ (最終的にはユーザの作成するコンポーネント) により編集されます。編集された画面ブロックは、アテンションキーを押した後の処理のためにホストに送り返されます。Screen オブジェクトは、現在の画面のデータブロックを表します。たとえば、24 x 80 の端末画面では、これは 1,920 バイトのデータになります。

動作

文字データがホストから着信すると、ネイティブ環境ペインに対して適切な更新がリアルタイムで行われます。このような更新には、単なるカーソルの位置変更から、端末画面の完全な再表示まで、あらゆる内容が含まれます。その意味では、画面のコンテンツは非常に動的です。

(Set Screen Text アクションを使用して) 現在の画面のコンテンツを操作したいということを exteNd Composer に通知すると、ECMAScript によってコンポーネントにアクセス可能になる「Screen オブジェクト」に画面バッファがパッケージ化されます。

多くの場合、コンポーネントでは、ホストにキー入力を送信し直したり、プロンプトにデータをマップしたりする前に、完全な画面のコンテンツを「認識」または理解している必要はありません。しかし、画面から DOM への送信をマップする場合は、Screen オブジェクトへのプログラム的なアクセスがあると便利なことがあります。これを実現するため、T27 Connect では、画面コンテンツを操作するための多数の ECMAScript 拡張を定義しています。このような拡張については、次の章で詳細に説明し、ここでは、単純な例を取り扱います。たとえば、画面の列 21、行 23 の位置に現れる文字列値を取得するとします。この文字列の長さが 28 文字の場合、(出力 DOM または一時 DOM をターゲットとして使用して) Map アクションのソースとして次の ECMAScript 式を使用することによって、この値を取得できます。



前の例では、XPATH /MARCOUTPUT/FAMSTATUS で、画面の行 23、列 21 で始まる 28 文字が出力 DOM にマップされます。

Map アクションおよび Screen メソッドの詳細については、第 4 章の T27 固有の Expression Builder 拡張に関する節で説明されています。

T27 固有のツールバーボタン

exteNd Composer に精通している場合、T27 Connect ではコンポーネントエディタのメインツールバーに多数の Connect 固有のツールアイコンが表示されることがわかります。これらのアイコンは、次のとおりです。

［Record］ボタン

［Record］アイコン (通常の状態)

［Record］アイコン (記録が進行中)

［Record］アイコン (無効な状態)

［Record］ボタンを使用すると、ネイティブ環境ペインを操作する場合に、キーボードや画面での操作をキャプチャできます。記録された操作は、アクションモデルにアクションとして配置され、その後、テスト中に「再生」することが可能です。

［Connection］ボタン

［Connection］(切断されている状態)

［Connection］(接続されている状態)

［Connection］(接続されているが無効な状態)

Composer のメインツールバーにある［Connection］ボタンでは、(コンポーネントに関連付けられた接続リソースの作成時に指定した設定を使用して) コンポーネントの接続の状態が切り替わります。

注記： **記録またはアニメーション表示を行う場合、接続は自動的に確立され、その場合、ボタンは「接続されているが無効」な状態で表示されます。記録をオフにすると、［Connection］ボタンは有効な状態に戻ります。**

［Set Screen Text］ボタン

exteNd Composer のメインツールバーにある［Set Screen Text］ボタンを使用すると、Screen オブジェクトにデータを送信することを示すことができます。このボタンをクリックすると、［Set Screen Text］ダイアログボックスが表示され、新しい Set Screen Text アクションを作成できるようになります (このアクションタイプの詳細については、次の章を参照してください)。

［Send Key］ボタン

アクションモデルに Send Key アクションを追加する場合に、Composer のメインツールバーの［Send Key］ボタンを押します (このアクションタイプについての詳細は、次の章を参照してください)。さまざまな T27 アテンションキーの詳細については、前の「T27 のキーボードサポート」の節を参照してください。

［Create Check Screen］ボタン

Composer のメインツールバーの［Create Check Screen］ボタンを使用すると、端末画面が予想される状態であるかどうか確認できます。このボタンをクリックすると、［Check Screen］ダイアログボックスが表示され、新しい Check Screen アクションを作成できるようになります (このアクションタイプの詳細については、次の章を参照してください)。

T27 固有のメニューバー項目

［Component］メニュー

T27 Connect の［Component］ドロップダウンメニューには、［Start/Stop Recording］および［Connect/Disconnect］(現在の状態に応じて異なります) という 2 つの項目が追加されています。

［**Start/Stop Recording**］- このメニューオプションは、ホストプログラムと通信する場合にアクションの自動作成を管理します。［**Start**］は、画面を操作する際にアクションの自動作成を有効にし、［**Stop**］は、アクションの作成を終了します。

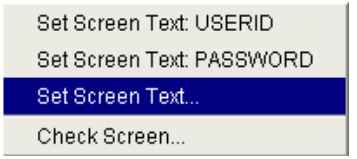
［**Connect/Disconnect**］- このメニューオプションでは、ホストへの接続を制御できます。記録またはアニメーション表示を行う場合、接続は自動的に確立されます (その結果、［Connection］アイコンは「接続されているが無効」な状態で表示されます)。ただし、このメニュー項目は、記録を行うのではなく、単に T27 環境を移動することを目的として接続を確立したい場合に便利です。

T27 固有のコンテキストメニュー項目

T27 Connect には、この Connect に固有なコンテキストメニュー項目も含まれています。コンテキストメニューを表示するには、ネイティブ環境ペインまたはアクションペインにカーソルを置き、マウスを右クリックします。

ネイティブ環境ペインのコンテキストメニュー

ネイティブ環境ペインでマウスを右クリックすると、コンテキストメニューが表示されます。記録モードになっていない場合、メニュー項目はグレー表示されます。記録モードでは、コンテキストメニューは次のようになります。



これら 4 つのコマンドの機能は、次のとおりです。

［**Set Screen Text: USERID**］- このコンポーネントの T27 接続リソースでユーザ ID に対して指定した値 (存在する場合) に基づいて、ユーザ ID 情報をホストに自動的に送信します。また、対応する Set Screen Text アクションをアクションモデルで作成します。

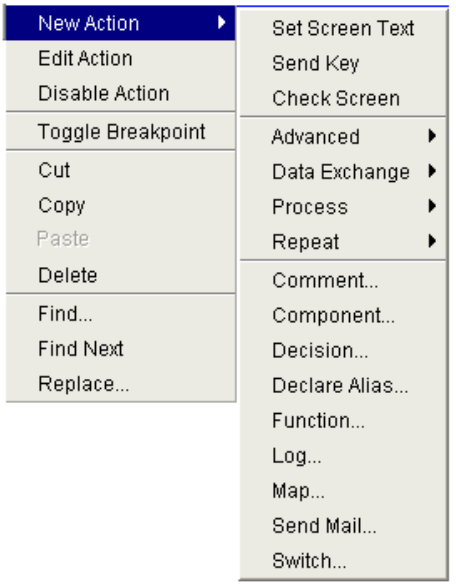
［**Set Screen Text: PASSWORD**］- このコンポーネントの T27 接続リソースでパスワードに対して指定した値 (存在する場合) に基づいて、パスワード情報をホストに自動的に転送します。また、対応する Set Screen Text アクションをアクションモデルで作成します。

［**Set Screen Text...**］- 新しい［Set Screen Text］ダイアログボックスを作成し、新しい Set Screen Text アクションを作成できるようにします (このコマンドの使用の詳細については、次の章を参照してください)。

［**Check Screen...**］-［Check Screen］ダイアログボックスを表示し、新しい Check Screen アクションを作成できるようにします (詳細については、次の章を参照してください)。

アクションペインのコンテキストメニュー

アクションペインの任意の場所にマウスを合わせて右クリックすると、コンテキストメニューが表示されます (次を参照)。



T27 固有の機能であるコンテキストメニュー項目は、次のとおりです。

［**Set Screen Text**］- Set Screen Text アクションを作成して、データをホストに送信できます。ダイアログボックスが開き、ホストに送信する内容を指定したり、情報を受け取る画面の位置を決定したりできます (このコマンドの使用の詳細については、次の章を参照してください)。

［**Send Key**］- Send Key アクションを作成できます。さまざまな T27 アテンションキーの詳細については、前の「T27 のキーボードサポート」の節を参照してください (Send Key アクションの詳細については、次の章を参照してください)。

［**Check Screen**］- コンポーネントが処理を続行する前に、適切な画面が存在することを確認するために使用する新しい Check Screen アクションを作成できます。ダイアログボックスが表示され、さまざまな実行許可条件や、タイムアウト値を指定できます (Check Screen アクションの詳細については、次の章を参照してください)。

4 基本的な T27 アクションの実行

アクションについて

「アクション」は、プログラミングステートメントに類似しており、パラメータの形式で入力を受け付け、特定のタスクを実行します。『Composer ユーザガイド』のアクションに関する章を参照してください。

T27 コンポーネントエディタ内では、XML ドキュメントを処理したり、XML 以外のデータソースと通信したりするための一連の命令がアクションモデルの一部として作成されます。アクションモデルは、ホストと XML ドキュメント間のすべてのデータマッピング、データ変換、データ転送、およびコンポーネントおよびサービス内のデータ転送を実行します。

アクションモデルは、連係するアクションのリストから構成されています。たとえば、ファイルから請求書に関するデータを読み取り、出力 XML ドキュメントに保存する前に何らかの方法でデータを変換するアクションモデルを構築できます。

このようにアクションモデルは、複数のアクションから構成されています。たとえば、次のようなアクションが含まれます。

- 入力として SKU 番号を含む XML ドキュメントを使用し、Unisys ホストにある在庫データベースからその SKU に関する請求書のデータを取得する T27 トランザクションを実行する
- 結果を一時 XML ドキュメントにマップする
- コードテーブルを使用して数値コードを変換する
- 結果を出力 XML ドキュメントにマップする

T27 固有のアクションについて

前の章で説明したように、T27 Connect には T27 環境に固有な 3 つのアクション、Set Screen Text、Send Key、および Check Screen が含まれています。

T27 アクション	説明
Set Screen Text	ユーザは、ホストに転送するデータおよび受信時の画面の位置を指定できます。文字列は、Map アクションまたはユーザのキー入力から作成されるか、ECMAScript 式から構築されます。Set Screen Text アクションは手動でも作成できますが、ほとんどの場合、ユーザが画面に入力したり、現在のプロンプトにデータをマップしたりすると自動生成されます。
Send Key	T27 に固有なアテンションキーをホストシステムに送信します。Send Key アクションはアクションを選択して手動で作成することもできますが、ユーザがマップされたキーのいずれかを押すか、T27 のキーパッドから選択する際に自動的に作成されます。
Check Screen	コンポーネントとホストアプリケーションの同期を保つことができます。このアクションでは、ユーザが指定したタイムアウトの値に応じて、画面が特定の状態 (Check Screen 設定ダイアログボックスで指定します) になるまで実行できないことをコンポーネントに通知します。

これらのアクションの目的は、(配備されたサービスで実行されている) T27 コンポーネントが、ランタイム時に T27 セッションで発生する端末 / ホストの通信を複製できるようにすることです。次に、これらのアクションの使用法と意味をさらに詳しく説明します。

Set Screen Text アクション

Set Screen Text アクションでは、コンポーネントの実行時に 1 度の送信でホストに送信される「キー入力されたデータ」(データが実際にキー入力によって取得されたか、ドラッグアンドドロップによりマップされたものか、または Expression Builder で作成した ECMAScript 式によるもの) がカプセル化されます。Set Screen Text アクションを実行すると、データがホストシステム画面に表示されます。ただし、Send Key アクションを使用して何らかのアテンションキーが送信されるまで、データはホストに送信されません。

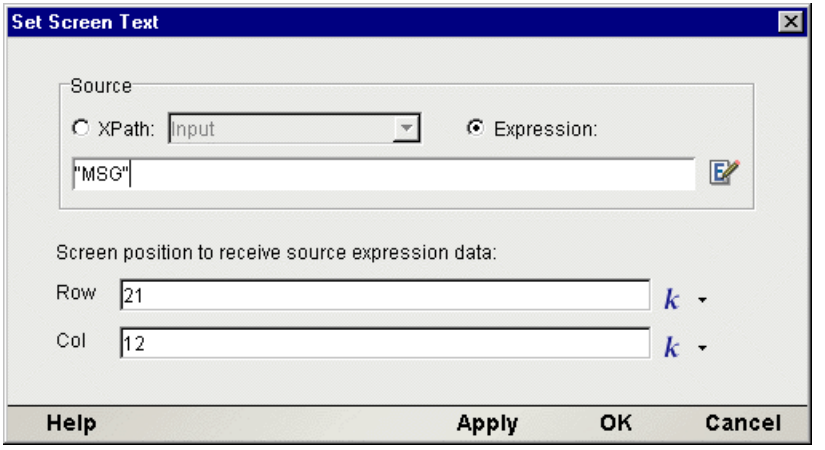
Set Screen Text アクションは、次のように複数の方法で作成できます。

- 記録モードでは、ナビゲーション環境ペインに入力するだけです。キー入力は、新しい Set Screen Text アクションに自動的にキャプチャされます。
- アクションモデルの任意の場所を右クリックして、コンテキストメニューを表示します。［**New Action**］および［**Set Screen Text**］を選択します。
- メインメニューバーで、［**Action**］、［**New Action**］、［**Set Screen Text**］の順に選択します。

- 記録モードの状態で、ネイティブ環境ペインにカーソルを置き、右クリックして［**Set Screen Text**］を選択します。

➢ メニューコマンドを使用して Set Screen Text アクションを作成する

1. アクションモデルの任意の場所をマウスで右クリックして、コンテキストメニューから［**New Action**］、［**Set Screen Text**］の順に選択します (または、前に説明したように［Action］メニューを使用します)。［Set Screen Text］ダイアログボックスが表示されます。



2. DOM 要素のコンテンツをバッファにマップするには、［**XPath**］ラジオボタンをオンにしてからプルダウンリストで DOM を選択し、テキスト領域に適切な XPath ノード名を入力します (または、右側の［Expression］アイコンをクリックして、Expression Builder を使用してノード名を作成します)。
3. ECMAScript を使用してバッファのコンテンツを指定するには、前の画面に表示されているように［**Expression**］ラジオボタンをオンにしてから、［Expression Builder］ダイアログボックスを使用して、文字列を返す ECMAScript 式を作成します。
4. データを受信する行を指定するには、フィールドに値を入力します。デフォルトでは、入力した数字が定数になります。**k** (定数) の横にある下向き矢印を使用すると、定数の入力と ECMAScript 式を切り替えることができます。
5. データを受信する列を指定するには、フィールドに値を入力します。デフォルトでは、入力した数字が定数になります。**k** (定数) の横にある下向き矢印を使用すると、定数の入力と ECMAScript 式を切り替えることができます。
6. ［**OK**］をクリックします。

注記： セッションの記録中に Set Screen Text アクションが自動的に作成されると、アテンションキーを押すか［Send Key］ダイアログボックスからいずれかを選択するまで、その後のキー入力はすべてバッファにキャプチャされます。

Send Key アクション

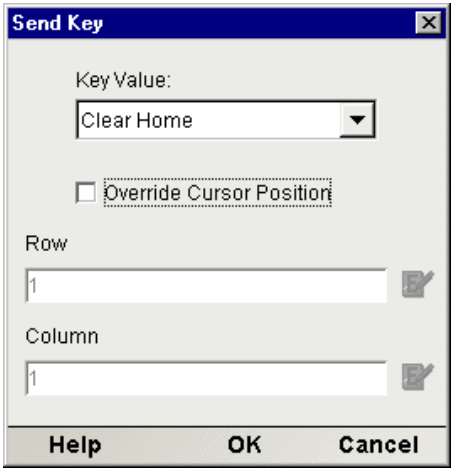
Send Key アクションでは単純に、アテンションキーがホストに送信されます。このアクションは通常、ホストに転送したい情報が確実に送信されるように Set Screen Text アクションに続きます。Send Key アクションを実行すると、Set Screen Text アクションで指定されたデータが実際にホストに送信されます。もちろん、Send Key アクションの中にはスタンドアロンのものもあり、それらは特定の情報を受信したり、画面をクリアしたり、別の領域に移動したりする場合にいつでも押すことができます。

Send Key アクションは、次のように複数の方法で作成できます。

- 記録モードで、アテンションキーとして割り当てられている PC のキーのいずれかを押し、現在のカーソル位置でアテンションキーを実行します (これらのキーについては、前の章を参照してください)。
- ドロップダウンメニューから［**View**］、［**Terminal Keypad**］の順に選択し、アテンションキーを押して［**OK**］をクリックして現在のカーソル位置でアテンションキーを実行します。
- メインツールバーで［Send Key］アイコンをクリックして、［Send Key］ダイアログボックスを表示します。
- アクションモデルで、Send Key の前に実行するアクションにカーソルを合わせてマウスを右クリックし、コンテキストメニューを表示します。［**New Action**］、［**Send Key**］の順に選択し、ダイアログボックスを表示します。
- Send Key の前に実行するアクションにカーソルを合わせて、メインメニューバーから［**Action**］、［**New Action**］、［**Send Key**］の順に選択して、ダイアログボックスを表示します。

➢ メニューコマンドを使用して Send Key アクションを作成する

1. アクションモデルの任意の場所をマウスで右クリックして、コンテキストメニューから［**New Action**］、［**Send Key**］の順に選択します (または、前に説明したように［Action］メニューを使用します)。［Send Key］ダイアログボックスが表示されます。



2 ［Key Value］ドロップダウンリストから、ホストに送信するアテンションキーを選択します。各アテンションキーの機能は、ホストアプリケーションにより異なります。

3 現在の行 / 列ではなく他の位置でキーを実行する場合は、［Override Cursor Position］ボックスをオンにします。［Row］および［Column］フィールドが使用できるようになります。

4 キーを転送する行を指定するには、フィールドに値を入力します。デフォルトでは、入力した数字が定数になります。別の方法として、［Expression Builder］をクリックして、ECMAScript 式の形式で行を入力することもできます。

5 キーを転送する列を指定するには、フィールドに値を入力します。デフォルトでは、入力した数字が定数になります。別の方法として、［Expression Builder］をクリックして、ECMAScript 式の形式で列を入力することもできます。

6 ［**OK**］をクリックします。

Check Screen アクション

T27 セッションでは待ち時間が生じたり、画面データがホストとアプリケーション間で定義された任意の順序で着信する可能性があるため、コンポーネントで現在の画面データを操作する前に、端末画面が特定の状態にあると確証できることが不可欠となります。Check Screen アクションでは、コンポーネントとホストとの「同期」を保つことができます。正しい画面が正しい状態のときに正確に動作するように、Check Screen アクションをアクションモデルのさまざまな場所に手動で作成します。

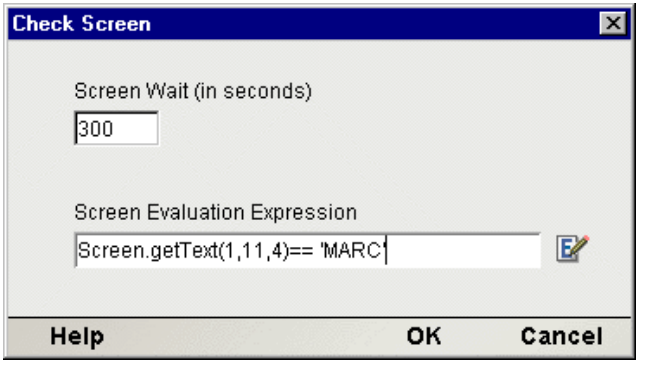
新しい Check Screen アクションを作成するには、次のいずれかの操作を実行できます。

- メインツールバーで［Create Check Screen Action］ボタンをクリックする
- アクションリスト内でマウスを右クリックして、コンテキストメニューから［**New Action**］、［**Check Screen**］の順に選択する
- コンポーネントエディタのメインメニューバーで、［**Action**］、［**New Action**］、［**Check Screen**］の順に選択する
- 記録モードの状態で、ネイティブ環境ペインにカーソルを置き、右クリックして［**Check Screen**］を選択する

注記： 記録モードでは、ほとんどの場合ツールバーボタンを使用します。

➢ メニューコマンドを使用して Check Screen アクションを作成する

1. その後に新しい項目を表示するアクションモデルのアクション項目にカーソルを置き、マウスを右クリックします。コンテキストメニューから［**New Action**］、［**Check Screen**］の順に選択します (または、前の説明のようにメインメニューバーで［Action］メニューを使用します)。［Check Screen］ダイアログボックスが表示されます。



2. ［**Screen Wait**］に、画面が待機する時間を秒単位で指定します (後の説明を参照)。
3. ［**Screen Evaluation Expression**］に、評価式を直接入力するか、［Expression Builder］アイコンをクリックして作成します (後の説明を参照)。
4. ［**OK**］をクリックします。

Check Screen アクションについて

アクションモデルでは、ホストアプリケーションの準備ができ、すべての画面データが着信する (つまり画面の状態が既知の状態となる) まで、アクションの実行は進まないことに注意してください。

コンポーネントでは、何らかの方法で現在の画面の準備が整ったことを「認識」する必要があります。Check Screen アクションを使用すると、準備が整ったという条件を指定できます。

Check Screen アクションのダイアログボックスでは、次の 2 つの条件を指定できます。

- プログラムを同期するための待機時間を指定する
- 画面の準備が実行時に整っているかどうか判断するための条件として使用される式を指定する

Screen Wait

［Screen Wait］の値 (秒単位) は、画面データが着信してから式で指定された準備条件が満たされるまでコンポーネントが待機する最長時間を表します。指定された時間が経過するまでに使用可能な画面データが準備条件を満たさない場合は、例外がスローされます。

注記： **もちろん、T27 セッションにおける待ち時間は、アプリケーション、接続、または画面によっても大きく異なるため、［Screen Wait］の値は慎重に決定する必要があります。［Screen Wait］の「安全な」値を決定するには、設計時およびサーバ上で慎重にコンポーネントをテストする必要があります。**

［Screen Wait］のデフォルト値は、T27 接続リソースの設定時に入力した値により決定されます。

Expression

実行可能条件を判断するために、Check Screen アクションのダイアログボックスで［Expression］ラジオボタンをオンにして、関連するテキストフィールドに ECMAScript 式を入力できます。ここで作成される式によって、通常は Screen オブジェクトバッファのある場所で特定のデータの存在が確認されます。ランタイム時に式で「true」が返されると画面の準備が整っているものとみなされますが、逆の場合はあてはまりません。式の例 : `Screen.getText(1,11,4) == "MARC"`

式については後で詳しく説明します。

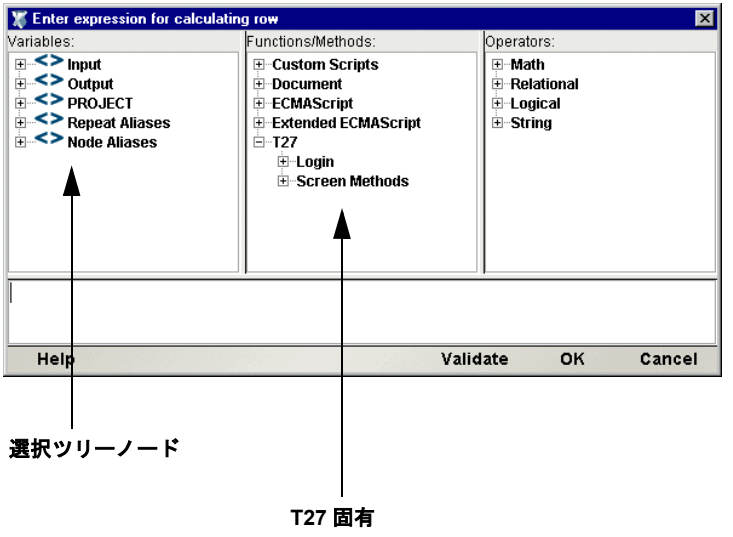
記録モードでのアクションの使用

コンポーネントに対してアクションモデルを作成する簡単な方法は、記録モードを使用することです。この方法でアクションモデルを構築する場合、要素を入力するか、入力 DOM から画面上の適当なフィールドにドラッグすると、新しい Set Screen Text アクションが自動的に作成されます。その後は、有効なアテンションキーを送信し、ホストから次の画面が着信するのを待ち、Check Screen アクションを追加して正しい画面にいることを確認し、再び処理を開始するという手順を繰り返し行うだけです。この方法を利用すると、Set Screen Texts、Send Keys、および Check Screens アクションのシーケンスを非常に簡単で自然に作成できます。

記録モードでの作業については、第 5 章の「T27 セッションの記録」の節で詳しく説明します。

T27 固有の Expression Builder 拡張

T27 Connect では、T27 固有の ECMAScript の変数およびオブジェクト拡張が複数提供されており、Expression Builder の選択リストに表示されます。T27 に固有な項目のリストは［T27］とラベルの付いたノードで表示され、［Login］および［Screen Methods］という 2 つのチャイルドノードがあります (次の図を参照)。



ログイン

T27 接続リソースには、［Expression Builder］ダイアログボックスからアクセスできる USERID および PASSWORD という 2 つのグローバル変数があります。これ

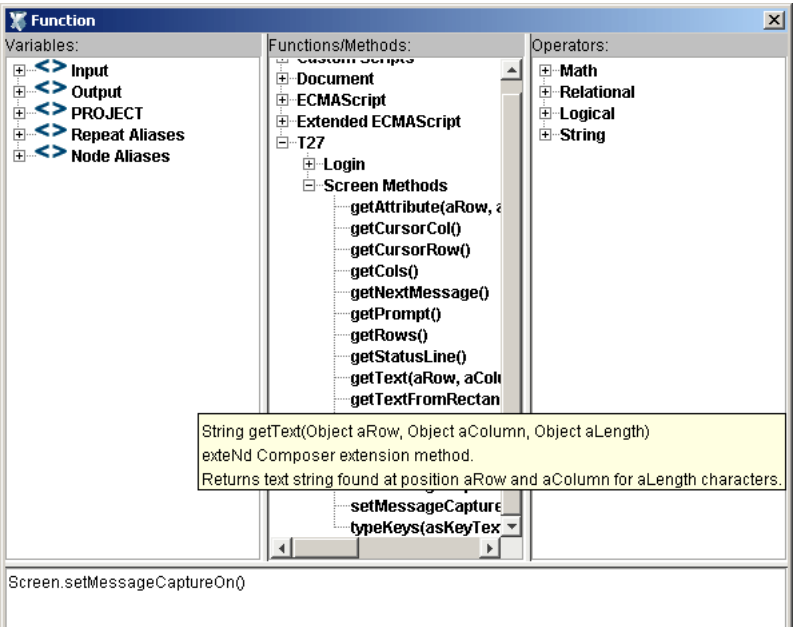
らのプロパティ (T27 選択ツリーの［Login］ノードでアクセスできます) により、接続時にホストシステムで要求される可能性があるユーザ ID およびパスワードの値が指定されます。これらの変数は端末画面にマップできるため、ユーザは Map アクションで明示的にユーザ情報およびパスワード情報を入力する必要がなくなります。

注記： また、XPath ソースが $PASSWORD として定義されている Set Screen Text アクションを作成することもできます。

Screen メソッド

T27 コンポーネントで、Map アクションまたは Function アクションから［Expression Builder］ウィンドウにアクセスした場合、ウィンドウ上部の選択リストには T27 に固有の特別な ECMAScript 拡張が表示されます。この ECMAScript 拡張は、Screen オブジェクトのさまざまなメソッドから構成されています。

マウスを特定の選択ツリー項目の上に移動すると、マウス移動ヘルプを使用できます (図を参照)。



さらに、ダイアログボックスの左下隅にある［Help］をクリックするとさらに詳細なオンラインヘルプを取得できます。

Screen オブジェクトでは、次の名前、署名、および使用規則を持つメソッドが提供されています。

getAttribute(*nRow, nColumn*)

戻り値のデータ型 : int

このメソッドは、aRow、aColumn によって指定された画面位置にある文字の「表示属性」値を返します。使用できる表示属性値の完全なリストは、付録 B「T27 の表示属性」に示されています。このメソッドの使用例は、次のとおりです。

```
if (Screen.getAttribute( 5, 20 ) == 34) // if character at row 5, col
20 is protected and bold
... // do something
```

getCursorCol(*void*)

戻り値のデータ型 : int

このメソッドは、T27 エミュレータ画面 (ネイティブ環境ペイン) 内のカーソルの現在の列位置を返します。列位置は、ゼロではなく 1 を基準とします。つまり、24x80 モードでは、このメソッドにより 1 から 80 までの値が返されます。

getCursorRow(*void*)

戻り値のデータ型 : int

このメソッドは、T27 エミュレータ画面 (ネイティブ環境ペイン) 内のカーソルの現在の行位置を返します。行位置は、ゼロではなく 1 を基準とします。つまり、24x80 モードでは、このメソッドにより 1 から 24 までの値が返されます。

getCols(*void*)

戻り値のデータ型 : int

このメソッドは、現在の画面本来の横のサイズを返します (ホストプログラムの実行中にモードが変更される可能性があるため、この値は画面によって変わる場合があります。コンポーネントの有効期間中、この値が一定であるとは仮定しないでください)。プログラムが 24x80 モードの場合、このメソッドにより 80 が返されます。本来のサイズに関係なく画面の列をすべてループするには、次のように入力します。

```
for (var i = 1; i <= Screen.getCols(); i++)
{
      var myCol = Screen.getTextAt( i, 1, Screen.getCols() );
      // do something with myCol
}
```

getNextMessage(*void*)

戻り値のデータ型 : string

変数に配置された場合、このメソッドの結果によって、次にキャプチャされるメッセージを表す文字列が返されます。このメソッドで値が返されるようにするには、setMessageCaptureOn() メソッド(次を参照) を設定する必要があります。これらのメソッドに加えて、hasMoreMessages() および setMessageCaptureOff() という他の 2 つのメッセージメソッドがあります。次に、この 4 つのメソッドが一緒に使用される方法の例を示します。

```
function msgChecker(theScreen)
{
  theScreen.setMessageCaptureOn();
  while (theScreen.hasMoreMessages())
    {
       alert(theScreen.getNextMessage());
    }
  theScreen.setMessageCaptureOff();
}
```

getPrompt(*void*)

戻り値のデータ型 : string

変数に配置された場合、このメソッドの結果によって、列 1 から getCursorCol() まで (ただし getCursorCol() の値は含みません) のすべての文字、つまり行頭から現在のカーソル位置にあるすべての文字を表す文字列が返されます。次に例を示します。

```
var prompt=Screen.getPrompt();
alert(prompt);
```

注記 : 返されたこの文字列は、実際にホストのプロンプトである場合とそうでない場合があります。

getRows(*void*)

戻り値のデータ型 : int

このメソッドは、現在の画面本来の縦のサイズを返します (ホストプログラムの実行中にモードが変更される可能性があるため、この値は画面によって変わる可能性があります。コンポーネントの有効期間中、この値が一定であるとは仮定しないでください)。プログラムが 24x80 モードの場合、このメソッドにより 24 が返されます。本来のサイズに関係なく画面の行をすべてループするには、次のように入力します。

```
for (var i = 1; i <= Screen.getRows(); i++)

{

      var myRow = Screen.getText( i, 1, Screen.getRows() );

      // do something with myRow

}

var wholeScreen = Screen.getText( 1, 1 + 24 * 80 ); // ERROR!
```

getStatusLine(*void*)

戻り値のデータ型 : string

変数に配置された場合、このメソッドの結果によって、ネイティブ環境ペイン下部の黒いステータス行を表す ECMAScript 文字列が返されます。このステータス行は、Check Screen アクションの後でのみ有効になります。

画面の現在のステータスを示す警告を作成する場合、次のような Function アクションを作成できます。

```
var screenStatus = Screen.getStatusLine( );

alert(screenStatus);
```

getText(*nRow, nColumn, nLength*)

戻り値のデータ型 : String

このメソッドは、現在の画面で指定した行と列の位置から始まる文字 (長さ nLength) のシーケンスを表す ECMAScript 文字列を返します。nRow および nColumn は、ゼロではなく、1 を基準とします。これらのパラメータの値がゼロの場合、いずれも例外が発生します。

24x80 画面の行 20 に存在する最初の半分の文字列を変数に取得するには、次のように入力します。

```
var myRow = Screen.getText( 20, 1, 40 );
```

getText() メソッドは、画面の選択に関連したドラッグアンドドロップによる Map アクション (後の「連続するデータの選択」で説明します) および Check Screen アクションのいずれの場合も内部的に使用されます。

注記 : 関数の引数によって選択されたデータの量が画面の行の最後を超える場合、改行またはその他の特別な文字は文字列には挿入されません。

getTextFromRectangle(*nStartRow, nStartColumn,nEndRow, nEndColumn*)

getTextFromRectangle() メソッドは、下位文字列 (1 行につき 1 つ) で構成された単一の文字列を返します。また、下位文字列とは、パラメータとして指定された左上および右下の行 / 列の座標で定義された境界ボックス内にあるすべての文字で成り立っています。たとえば、24x80 モードでは、次のように実行すると、画面を 4 等分したうちの左上の部分を取得できます。

```
var topLeftQuadrant = Screen.getTextFromRectangle(1,1,12,40);
```

getTextFromRectangle() メソッドは、<Shift> を使用した選択方法によって作成された、長方形画面の選択部分に関するドラッグアンドドロップを使用した Map アクションで、内部的に使用されます (次の「長方形領域の選択」を参照)。

このメソッドによって返された文字列には、下位文字列間に改行区切り記号が含まれます。つまり、各行でデータの最後には改行が含まれます。そのため、返された文字列の全体的な長さは、行数と列数を乗算して、行数を加算した値となります。たとえば、Screen.getTextFromRectangle(1,1,4,4).length は、20 になります。

hasMoreMessages(*void*)

前に説明されている getNextMessage() メソッドを使用してより多くのメッセージが取得できる場合、hasMoreMessages() メソッドは true を返します。このメソッドは、前に説明されている getNextMessage() メソッドの中のその他のメッセージメソッドと共に示されています。

putString(*nRow, nColumn, textString*)

putString() メソッドを使用すると、明示的に Set Screen Text アクションを作成することなくプログラムによって画面上にある特定の行 / 列にデータを送信できます (例を参照)。

```
var goHome = "HOME";

Screen.putString(2,14, goHome); // send string to screen
```

putStringInField(*nFieldNumber, textString*)

putStringInField() メソッドを使用すると、明示的に Set Screen Text アクションを作成することなくプログラムによって画面上にある特定のフィールドにデータを送信できます。たとえば、MARC システムには通常、行 2 の Action: フィールド、行 21 の Choice: フィールドという 2 つのフィールドがあります。下の例では、putString と同じ結果が現れます。

```
var goHome = "HOME";

Screen.putStringInField( 1, goHome); // send string to screen
```

setMessageCaptureOff(*void*)

`setMessageCaptureOff()` メソッドは、メッセージキャプチャ機能をオフにします (下の `setMessageCaptureOn()` を参照)。

setMessageCaptureOn(*void*)

`setMessageCaptureOn()` メソッドはメッセージキャプチャ機能をオンにして、呼び出し側が取得できるようにすべてのホストメッセージが保存されるようにします。このメソッドは、前に説明されている `getNextMessage()` メソッドの中のその他のメッセージメソッドと共に示されています。

typeKeys(*String keys*)

`typeKeys( Str)` メソッドを使用すると、文字列で指定したキー入力を画面上でエミュレートできます。指定された文字列は、画面上の現在のカーソル位置に配置されます。次のテキストを含む関数では、SendKey アクションと同じ結果が現れます。

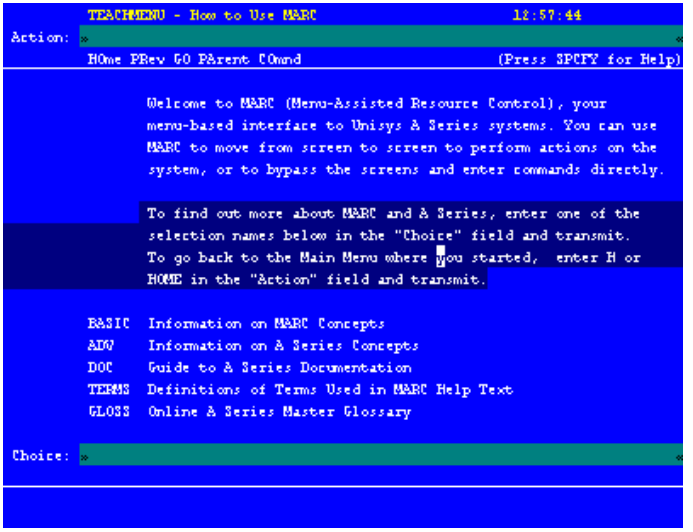
```
Screen.typekeys( "[Transmit]")
```

T27 Connect での複数行の画面選択

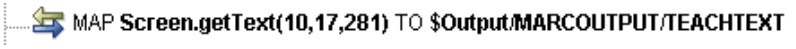
記録モードでは、DOM にドラッグするために連続する複数行にまたがるデータを選択できます。

連続するデータの選択

<Shift> キーを押さずに複数行でデータをドラッグすると、最初の画面オフセット (マウスをクリックした点) から最後の画面オフセット (マウスを離した点) までの文字列すべてが選択されます (次の図を参照)。選択されたテキストは「黒くハイライト」されます。上から行の一部、2 つの完全な行、行の一部という順に選択されます。



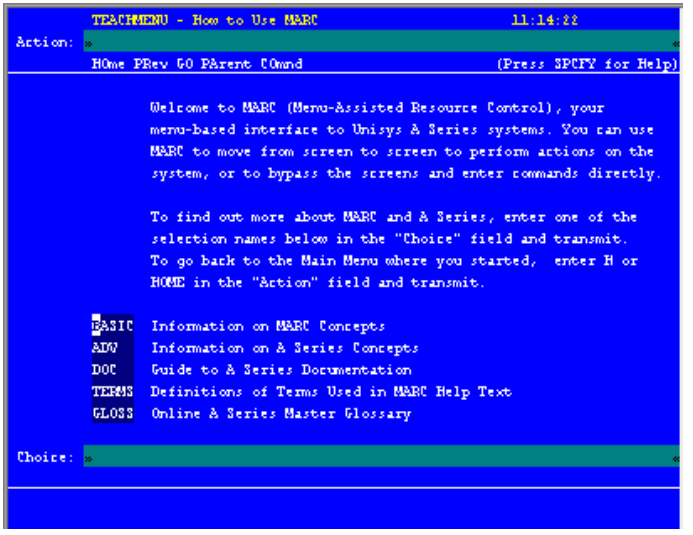
ドラッグすると、左下隅にあるコンポーネントエディタウィンドウのステータス行に、選択した開始行と終了行および列が報告されます。選択内容をネイティブ環境ペインの外にある DOM にドラッグすると、Map アクションが生成されます。



`getText()` メソッドが使用されていることに注意してください。つまり、キャプチャされた画面文字によって 1 つの文字列が形成され、**Output/MarcOutput/Teachtext** にマップされます。改行または他の特殊文字は文字列に挿入されません (画面上で濃い青でハイライトされた空白領域は、文字列では単純にスペース文字として表示されます)。

長方形領域の選択

場合によっては、前で説明した選択は、使用したくない場合があります。特定の状況では、画面データは独自の境界を持ったゾーンに分類できます。たとえば、前に表示されている画面では、定義および多数の空白を含めることなく左下の 5 つの用語のみを (画面外にドラッグする目的で) キャプチャできます。それには、最初に <Ctrl> キーを押してから、選択する画面部分でマウスを横にドラッグします。選択した領域がハイライトされ、適切な行 / 列の開始ポイントおよび終了ポイントが、コンポーネントエディタのウィンドウのステータス行に表示されます (次を参照)。



この例では、長方形内で選択された領域をネイティブ環境ペインから DOM 内にドラッグすると、その結果 Map アクションで、前に説明した `getTextFromRectangle()` メソッドが使用されます。結果のアクションは、次のようになります。

```
MAP Screen.getTextFromRectangle(15,11,19,15) TO $Output/MARCOUTPUT/TEACH
```

`getTextFromRectangle()` によって返された文字列は、長方形の右端でラップされるため、このメソッドは、`getText()` とは異なる方法で動作します。改行は、`getTextFromRectangle()` の API 記述で説明されたように、ラップポイントで挿入されます (前を参照)。

5 アクションでの T27 コンポーネント

サンプルトランザクション

このガイドでは、サードパーティにより説明を目的として提供された MARC (Menu-Assisted Resource Control) と呼ばれるメニュー駆動型のシステムインタフェースを例として使用します。ここで、スクリーンキャプチャで示されたトランザクションは、T27 端末のオペレータで一般的に使用されるトランザクションのタイプを表しています。

T27 セッションの記録

T27 コンポーネントは、大部分のアクションモデルが自動的に作成される点で、他のコンポーネントとは異なります。この現象は、ライブ T27 セッションの一部として、ネイティブ環境ペインでホストと通信する際に起こります。Composer では、アクションモデルで自動生成された一連のアクションとして通信を記録します。通常、他の exteNd Composer コンポーネント (JDBC コンポーネントなど) では、アクションモデルでアクションを手動で作成してから、マップ、ログ、変換、通信、およびコンポーネントやサービスで必要とされるその他のタスクを実行する必要があります。これとは逆に、T27 コンポーネントを作成する場合は、ホストへの要求およびホストからの応答を「記録」し、これが最終的にアクションモデルでアクションとして処理されます。さらに、他のコンポーネントと同様に、アクションモデルに標準のアクション (Map、Log、Function など) を追加できます。

注記： T27 コンポーネントを正常に作成するためには、XML 統合プロジェクトで使用するホストアプリケーションの仕様を理解しておく必要があります。

T27 コンポーネントを作成する際に必要な共通タスクは、次の例のとおりです。

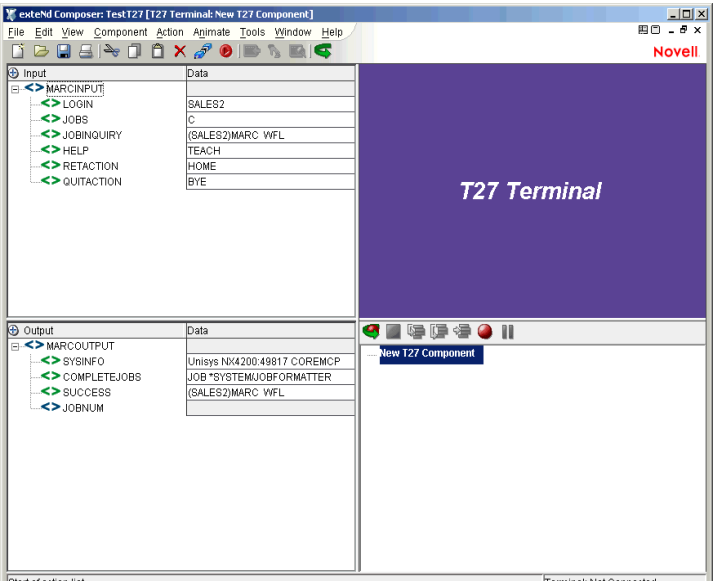
- Set Screen Text アクションを自動的に作成する
- Send Key アクションを自動的に作成する
- Check Screen アクションを自動的に作成する

- 入力 DOM 要素を T27 画面のプロンプトにドラッグアンドドロップしてマップする
- ネイティブ環境画面から出力 DOM にドラッグアンドドロップしてマップする
- ECMAScript 式を使用して、Screen オブジェクトの要素を操作する

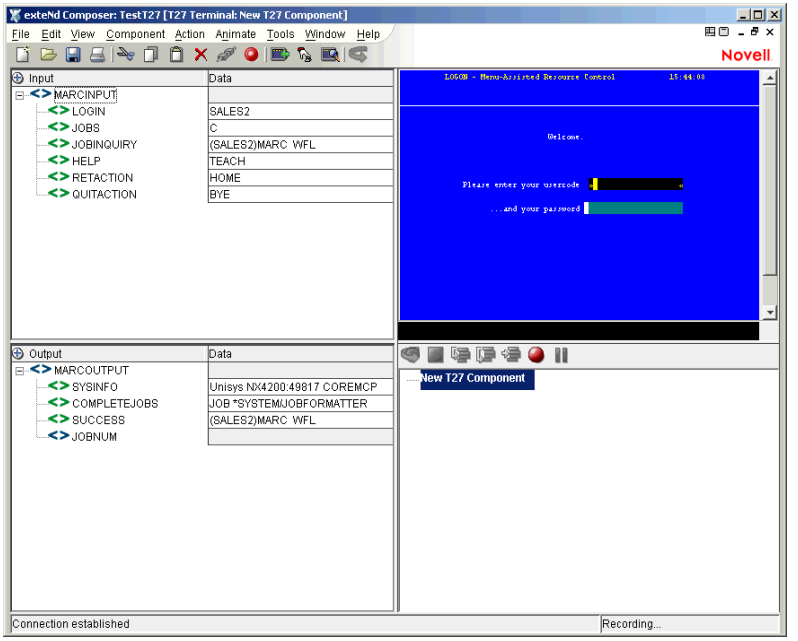
次の例では、MARC システムで処理するいくつかのトランザクションコマンドを含む入力 XML ドキュメントから操作を開始します。この特定のコンポーネントの目的は、MARC でこれらのコマンドを処理し、最後に完了するジョブを特定して、出力 DOM にそのジョブの名前を配置することです。

➢ T27 セッションを記録する

1. 第 3 章「T27 コンポーネントの作成」で説明されている手順に従って、T27 コンポーネントを作成します。
2. T27 コンポーネントを作成すると、T27 コンポーネントエディタウィンドウが開き、ネイティブ環境ペインの中央に「T27 Terminal」という語句が表示され、ホストとの接続が確立されていないことが示されます。



3. ［**Record**］ボタンをクリックします。自動的にコンポーネントの接続リソースで選択したホストに接続されます。ネイティブ環境ペインに入力画面が表示されます (次を参照)。

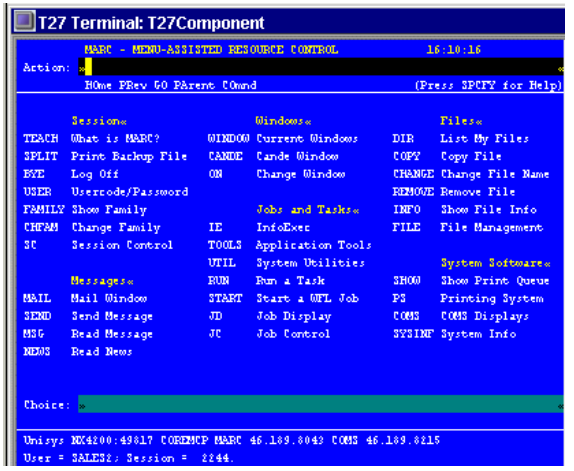


4 ［MARCINPUT］、［LOGIN］の順に展開されたノードを、入力 DOM からネイティブ環境ペインの最初のフィールドにドラッグします (フィールド内での正確な配置については特に心配する必要はありません)。ユーザコードとして「SALES2」(「」なし) が表示され、新しい Set Screen Text アクションがアクションモデルに自動的に表示されます。

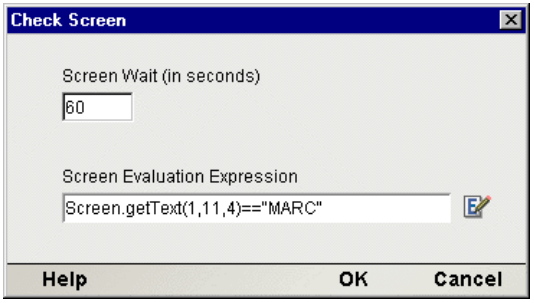
5 もう一度、入力 DOM から［MARCINPUT］、［LOGIN］ノードをドラッグして、今度はネイティブ環境ペインの 2 番目のフィールドに配置します (「SALES2」は、この画面では保護されたフィールドであるため、パスワードフィールドには表示されません)。2 番目の Set Screen Text アクションがアクションモデルに自動的に表示されます。2 つのアクションをよく確認すると、これらのアクションでは、行 / 列の配置のみが異なっていることに気付きます。

6 <F12> キーを押します。Send Key Transmit アクションがアクションモデルに追加されます。

7 Set Screen Text アクションおよび Send Key アクションに応じて、T27 画面でホームメニューが再び表示されます。



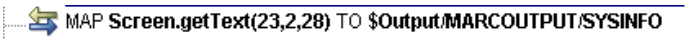
8 第 4 章の「Check Screen アクション」で説明したように、正しい画面になっていることを確認してから、操作を続行することをお勧めします。その場合には、まず左上隅の「MARC」という単語をカーソルでドラッグします。左下隅のコンポーネントエディタウィンドウのステータス行に、単語の始まりと終わりを表す行と列の位置が示されていることに注意してください。マウスの右ボタンを使用して、[Check Screen] アイコンをクリックします。getText メソッドが自動的に表示され、「MARC」という単語が画面上の目的の場所に表示されることが確認されます。



MARC プログラムの応答時間は、比較的短いため、この Check Screen アクションに対してデフォルトで 60 秒となる画面待機時間を暫定的に受け入れます (その場合でも、このタイムアウト値が安全であることを検証するため、コンポーネントを慎重にテストする必要があります)。［**OK**］をクリックして、アクションモデルに Check Screen アクションを入力します。

9 次に、ネイティブ環境パネルから出力DOMに何らかの項目をマップし直します。この例では、どのようなタイプの Unisys メインフレームを操作しているかを判明しようとしています。この情報は、エミュレータ画面の左下隅に示されます。「Unisys」という単語から「MARC」という単語の始まりまでカーソルをクリックしてドラッグします。クリックしてドラッグすると、選択した領域での画面上の行 / 列の座標がステータス行に表示されます。

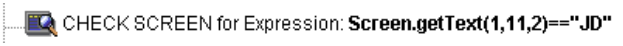
選択した情報を出力 DOM の［SYSINFO］ノードにドラッグします。新しい Map アクションがアクションモデルに表示されます。



10 正しい画面であることを検証した後に、正しいシステムを使用して、コマンドの一部を入力していきます。T27 環境ペインで［Choice:］フィールドの任意の場所をクリックして、「**JD**」と入力します。

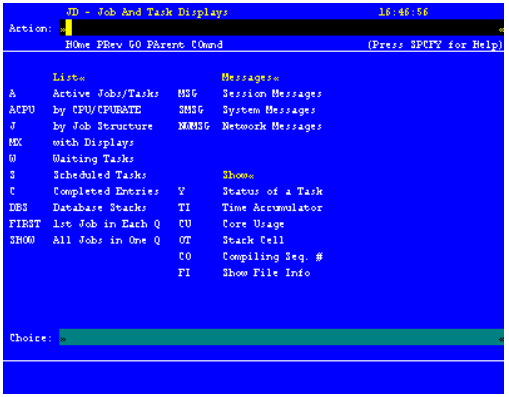
注記：ほとんどの場合、Unisys コマンドでは大文字と小文字が区別され、通常はすべて大文字で入力します。

コンポーネントのアクションリストに新しい Set Screen Text アクションが自動的に表示されます。フィールドに対する画面座標とともに、「JD」という文字がすでにアクション内に表示されていることに注意してください。



11 T27 接続では、［Transmit］キーを押すまでどのような動作も発生しないため、<F12> をもう一度押して、モデルに別の Send Key Transmit アクションを追加します。

12 送信した「JD」コマンドに対して、ホストプログラムにより、新しいメニュー画面が送信されます (次を参照)。



13 ここでも、目的の画面が表示されていることを確認することをお勧めします。この作業を実行するには、画面上部で「JD」という文字を選択し、右クリックして別の Check Screen アクションを作成して、画面が続行できる状態であるかどうかを判断します。この例では、次のように式を設定します。

```
Screen.getText(1,11,2) == "JD"
```

［**OK**］をクリックすると、Check Screen アクションがアクションリストに表示されます。

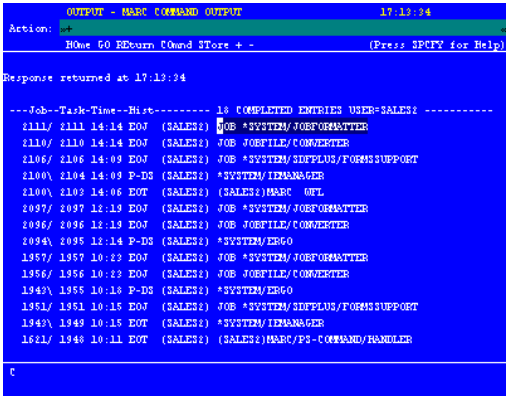
14 ［MARCINPUT］、［JOBS］の順に展開されたノードを、入力 DOM からネイティブ環境ペインの［Choice:］フィールドにドラッグします。フィールドに「C」(「」なし) が表示され、新しい Set Screen Text アクションがアクションモデルに自動的に表示されます。このメニューでは、「C」は「Completed Jobs」の省略形です。

15 <F12> を押して、別の Send Key Transmit を追加します。

16 別の Check Screen アクションを追加して、正しいメニューが表示されていることを確認します。今回は、「OUTPUT」という単語を確認します。

```
Screen.getText(1,11,6) == "OUTPUT"
```

17 ネイティブ環境ペインで、行 8 の端末画面のテキストを、列 35 からテキストの末尾まで選択してから、マウスをクリックしてドラッグすることによって、最初に完了したジョブの名前を選択します。テキストが選択された状態で表示されます。

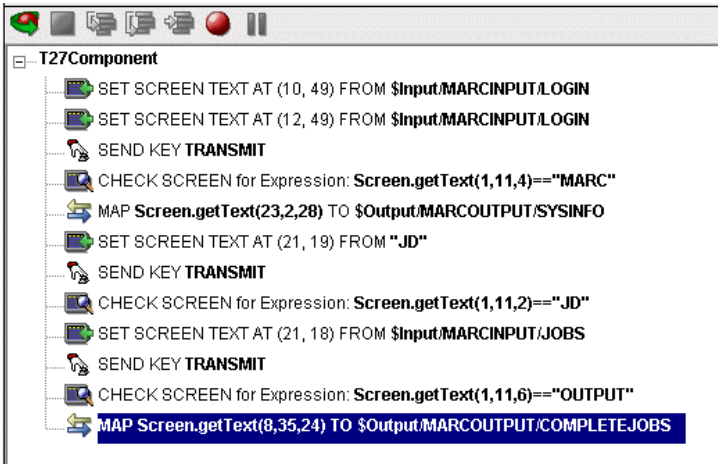


18 マウスのボタンから指を放して、選択したテキストの上にマウスを合わせます。選択内容をクリックして、出力 DOM で［**MARCOUTPUT**］の［**COMPLETEJOBS**］ノードにドラッグします。選択したテキストは、希望の場所で DOM に挿入され、新しい Map アクションがアクションモデルで自動生成されます。

19 ［Record］ボタンをクリックして、記録をオフにします。

20 コンポーネントを保存します。

ここまでで説明したすべての手順に正しく従った場合、完全なアクションモデルは次のようになります。



この簡単なアクションモデルについては、さらに手順を追って説明しますが、その前に既存のアクションモデルを編集する方法を学ぶ必要があります。

以前に記録したアクションモデルの編集

以前に記録したアクションモデルを編集する必要が生じる場合があります。他のコンポーネントを使用する場合とは異なり、T27 コンポーネントの編集には、特別な注意が必要です。T27 コンポーネントを実行すると、コンポーネントが適切に動作するために、特定の画面およびデータが一定の時間帯に表示されるようなアクションのシーケンスが繰り返されます。そのため、コンポーネントを編集する際には、アクションモデルのシーケンスが以前に記録したホストプログラムの実行シーケンスと矛盾しないよう (シーケンスを壊さないよう) 注意する必要があります。

一般的に、正常に編集を行うには、次の推奨事項に従います。

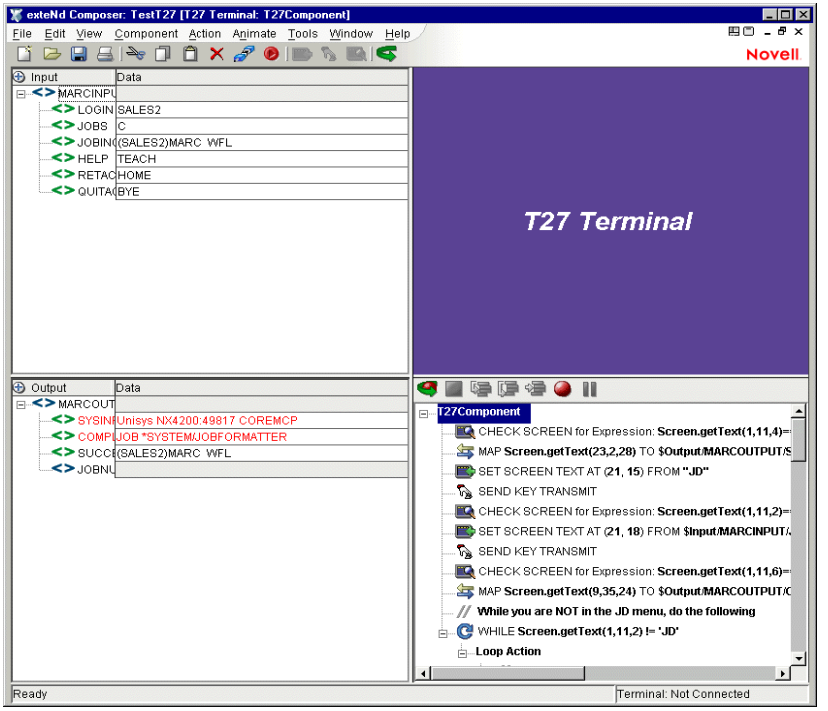
- アクションモデルで、[Cut]、[Copy]、または [Paste]、あるいはこれらすべてを使用してアクションを削除、移動、または複製する際には、細心の注意を払います。「記録」セッション中に自動的に作成されたアクションでは、編集処理で簡単に見落とされるデータの従属性が頻繁に作成されます。
- ドラッグアンドドロップを使用して、アクションモデルに新しい Map アクションを追加する必要がある場合は、アクションペインのツールバーで [Start Animation] ボタンをクリックし、アクションモデルの目的の行に進んでから、アニメーションを一時停止して記録モードをオンにします。この時点では、安全に画面の内外にドラッグできます。この手順に従うと、アクションモデルでホストとの同期がずれたり、以前にマップした DOM データと矛盾したりすることを防ぐことができます。

既存のアクションの編集または既存のアクションへの追加

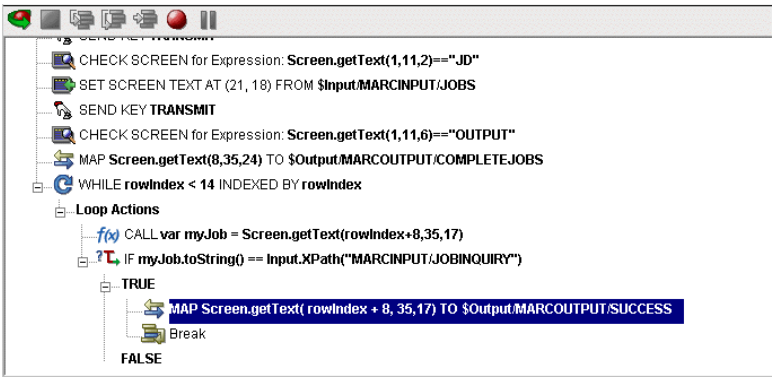
次の手順では、既存のアクションを変更したり、以前に記録したセッションに新しいアクションを追加したりする方法について説明します。

➢ 以前に記録したアクションモデルで、既存のアクションを変更する

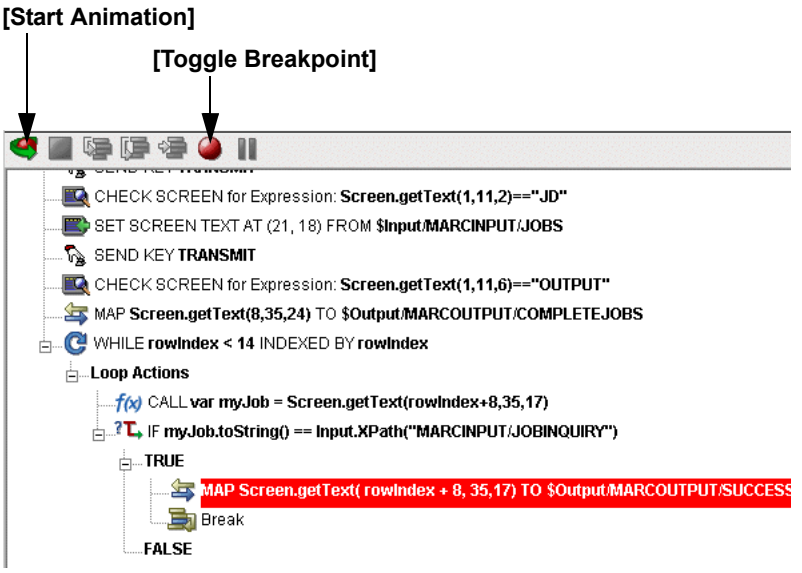
1 編集するアクションモデルが含まれるコンポーネントを開きます。T27 コンポーネントエディタウィンドウにコンポーネントが表示されます。



2 アクションモデルで、編集を行うアクション、または後ろに新しいアクションを追加するアクションに移動し、そのアクションを選択します。

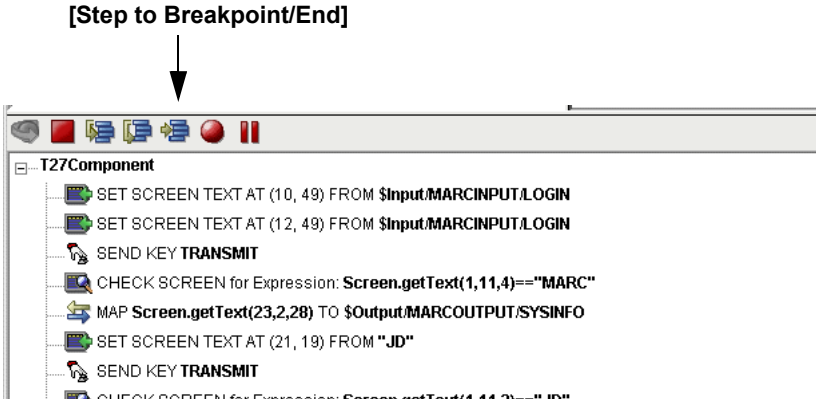


3 ［**Toggle Breakpoint**］ボタンをクリックします (または、<**F2**> を押します)。選択されたアクションが赤になります (アニメーションについては、下で詳しく説明します)。

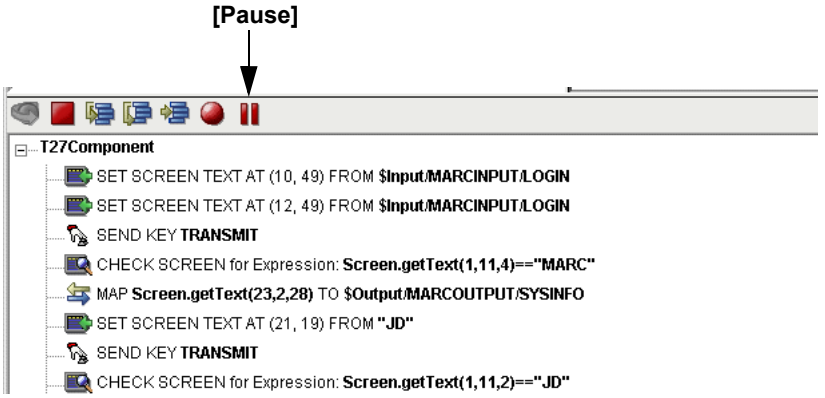


4 ［Start Animation］ボタンをクリックします。(アクションペインのツールバーで) アニメーションツールが有効になります。

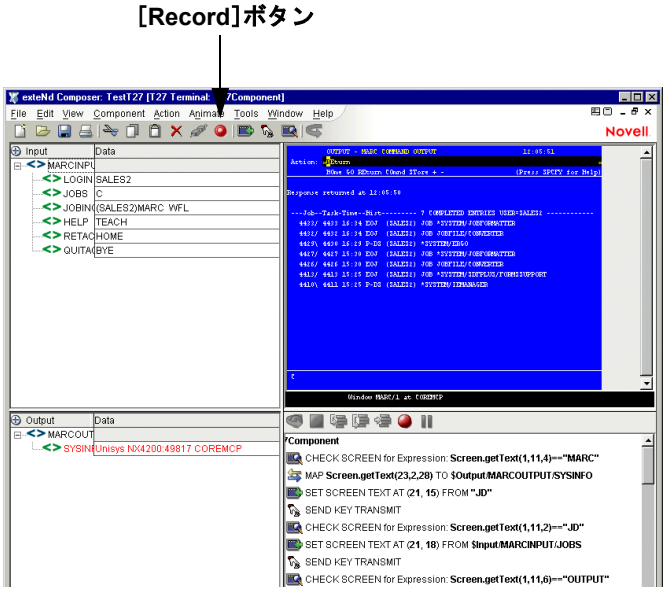
5 ［Step to Breakpoint/End］ボタンをクリックします。アクションモデルで、アクションモデルの最初から前の手順 3 で設定したブレークポイントまでのアクションがすべて実行されます。



6 ［Pause］ボタンを押します。



7 コンポーネントエディタのツールバーで、［**Record**］ボタンをクリックします。



8 アクションを右クリックして［**Edit Action**］を選択することによって、アクションを編集して、現在の行に必要な変更を加えます。または、新しいアクションを追加する場合は、Composer のドラッグアンドドロップ機能を使用して、画面と動作する新しい Map アクションを追加します。選択した行のすぐ下に新しいアクションが追加されます。

9 記録をオフにします (［**Record**］ボタンを切り替えます)。

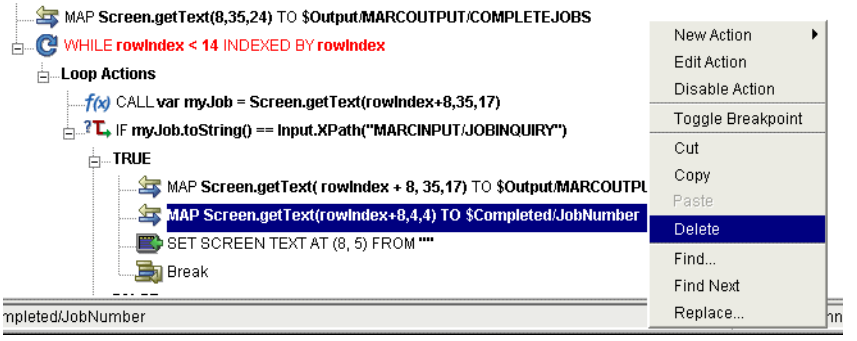
10 コンポーネントをテストします。

アクションの削除

次の手順では、以前に記録したセッションでアクションを削除する方法について説明します。

➢ 以前に記録したアクションモデルでアクションを削除する

削除するアクションの行を選択して、マウスの右ボタンをクリックします。メニューから［**Delete**］を選択します。行を選択して、キーボードの <Delete> ボタンを押すこともできます。



次の節で説明されているように、記録モードを使用しないで既存のアクションモデルに追加することもできます。

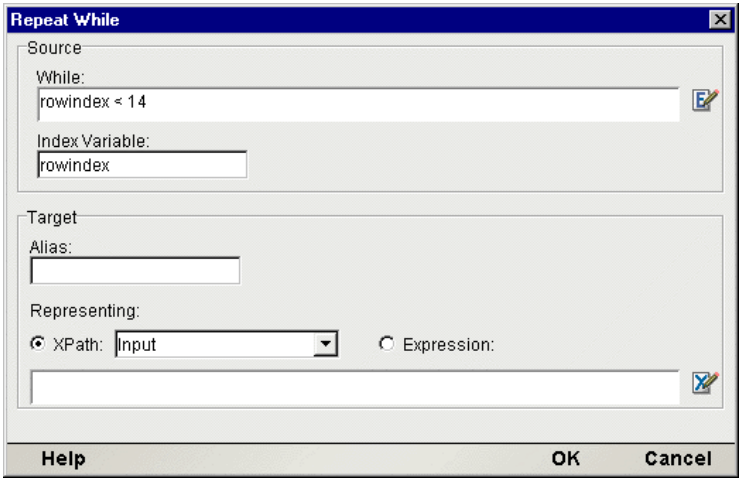
データ検索での複数行のループ

MARC の例 (前を参照) での目的は、完了したジョブを検索し、そのジョブの名前を出力 DOM にマップすることでした。このジョブの名前は、別のコンポーネントまたは Web サービスに対する入力として使用できます。

端末エミュレータ画面の［OUTPUT］メニューを簡単に目で確認することによって、例でマップしたジョブだけではなく複数のジョブが完了していることが簡単に確認できます。すべてのジョブのジョブ番号、名前、および完了時間を確認すると仮定します。この処理を実行するには、端末画面の行 8 から行 21 までを繰り返して、キーの値を出力 DOM に配置する必要があります。前の例を基準として、次の手順ではこの操作を行う方法を示します。

➢ 複数行の値を一度にマップする

1 アクションモデルの一番下に新しいRepeat Whileアクションを追加します(マウスを右クリックしてから、［**New Action**］、［**Repeat**］、［**Repeat While**］の順に選択します)。［Repeat While］ダイアログボックスが表示されます。



2 ［**While**］テキスト入力ボックスで、このループに適用するループの終了条件を表す式を入力します。この場合、条件は、インデックス変数 `rowIndex` のチェックです。全体で 14 行の画面データを確認することになります (ゼロは最初のデータとして計算されるため、0 ～ 13 回確認する必要があります)。

3 ［**Index Variable**］テキスト入力領域で、インデックス変数の名前を入力します (この場合は、`rowIndex` です)。

4 ターゲット情報を入力する必要はありません。［**OK**］をクリックします。新しい Repeat While アクションがコンポーネントのアクションモデルに追加されます。

5 次に、ループを実行するにつれて取得される値を保持するための変数を設定する必要があります。ジョブ番号から始めます。［Loop Action］で、マウスを右クリックして、コンテキストメニューから［**New Action**］>［**Function**］の順に選択します。次の ECMAScript 式を入力します。

```
var jobnum = Screen.getText(rowindex+8,4,5)
```

これにより、変数 `jobnum` が Screen オブジェクトの `getText()` メソッドの値 (第 4 章を参照) に等しくなるよう設定され、列 4 から始まる `rowIndex + 8` で 5 文字のデータが取得されます。画面データのこの検索は行 8 で始まり行 21 まで続くため、`rowIndex` 変数に 8 のオフセットを追加します。

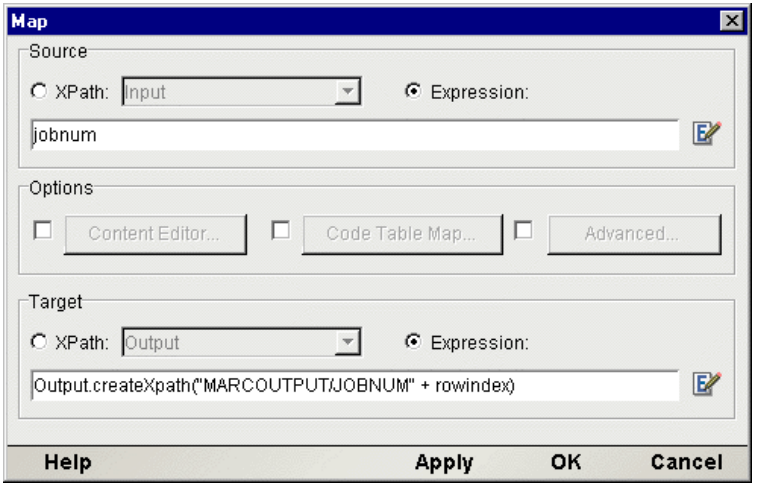
6 タスク番号、完了時間、およびジョブ名を保持する 3 つの変数を作成します。次のように ECMAScript 式を使用します。

```
var tasknum = Screen.getText(rowindex+8,10,4)

var timecomp = Screen.getText(rowindex+8,15,5)

var jobname = Screen.getText(rowindex+8,35,35)
```

7 次に、これらの変数を出力 DOM にマップして、exteNd ECMAScript 拡張 createXpath を使用して新しいノードを作成します。最初のマップアクションを作成するには、最初の変数作成ステートメント (jobnum) を選択して、右クリックします。コンテキストメニューから［**New Action**］、［**Map**］の順に選択します。［Map Action］ダイアログボックスが表示されます。

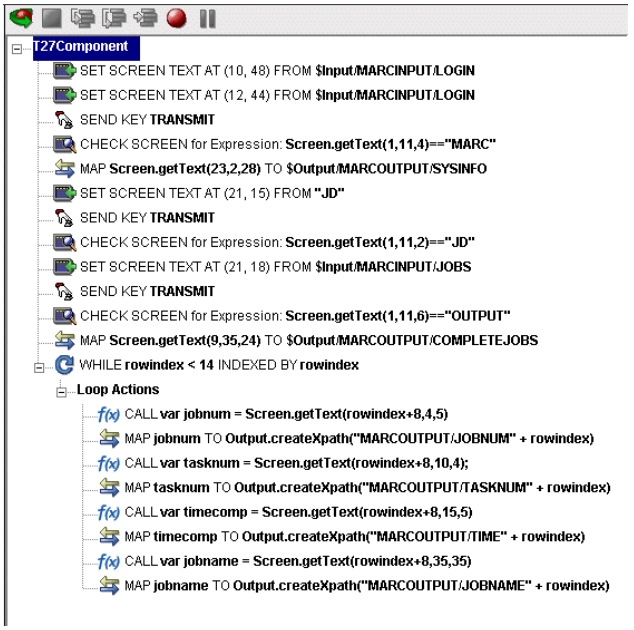


作成された Map アクションは非常にシンプルなものです。［**Expression**］ラジオボタンが選択され、ソースが ECMAScript 式であることを示しています。新しく作成された変数 (jobnum) をソースとして使用します。

ターゲットに対しては、出力 DOM で新しいノードを作成する必要があります。このため、ECMAScript 式を入力していることを示す必要があります。createXpath メソッドを使用して、rowindex の番号を nodename に付加し、各エントリに対して固有の名前が与えられるようにします。

8 作成した 4 つの各変数に対して、このマッピングプロセスを繰り返します。

完全なアクションモデルは、次のとおりです。

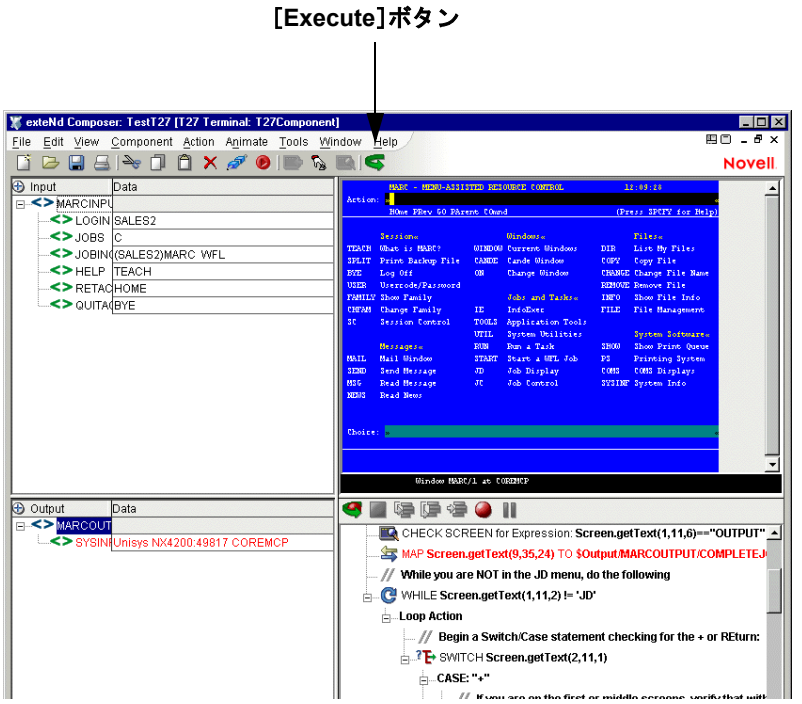


T27 コンポーネントのテスト

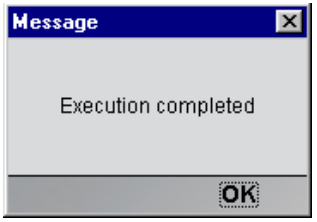
Composer には、コンポーネントを簡単にテストできるアニメーションツールが含まれています。T27 コンポーネントエディタのツールバーには、［Execute］ボタンがあり、このボタンを使用するとアクションモデル全体を実行して、コンポーネントが意図したとおりに動作するかを検証できます。テストを行う際には、すべての Check Screen アクションで画面待機時間の値に細心の注意を払い、値が適切であるか、また Set Screen Text アクションおよび他のアクションが意図したとおりに動作するか確認してください。

➢ T27 コンポーネントを実行する

1 T27コンポーネントを開きます。T27コンポーネントエディタウィンドウが表示されます。



2 ［**Execute**］ボタンを選択します。アクションモデルのすべてのアクションが順番に実行されます。コンポーネントが正常に実行された場合、次のようなメッセージが表示されます。



3 ［**OK**］をクリックします。

コンポーネントを実行した後は、DOM のコンテンツをもう一度確認し、すべてのデータが予想どおりに適切にマップされたか確認する必要があります。すべてのデータ要素を表示するには、［**View**］メニューで［**Expand XML Documents**］を選択します。これにより、DOM ツリーのペアレント、チャイルド、データ要素などがすべて展開され、コンポーネントの実行結果を簡単に確認できます。実行時に問題がある場合は、アニメーションツールを使用して、問題が生じている場所を特定できます。この処理については、次の節で説明します。

アニメーションツールの使用

アクションモデルには、1 つまたは複数のブレークポイントを設定して、アクションモデルの特定のセクションをテストできるアニメーションツールがあります。Toggle/Breakpoint ツールについては、前の節「既存のアクションの編集または既存のアクションへの追加」で簡単に説明しましたが、ここではさらに詳細に説明します。これらのツールを使用すると、適切に動作するアクションをすべて実行して、問題の生じたアクションで停止してから、問題のアクションを 1 つずつトラブルシューティングすることができます。

アニメーションツール機能の簡単な例は、次のとおりです。すべてのアニメーションツールおよびその機能の詳細については、『exteNd Composer ユーザガイド』を参照してください。

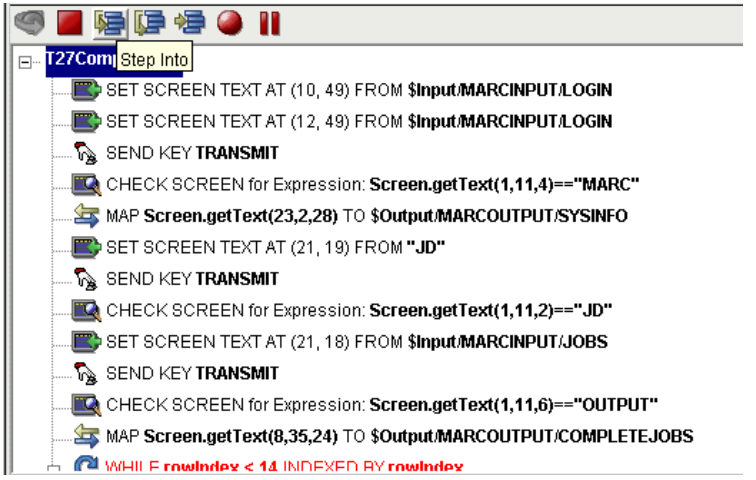
➢ アニメーションツールを使用して T27 コンポーネントを実行する

1. T27 コンポーネントを開きます。T27 コンポーネントエディタウィンドウにコンポーネントが表示されます。

   注記 : アニメーションモードと記録モードは、コンポーネントで「互いに排他的なモード」です。アニメーション中に記録を行うには、アニメーションを一時停止または停止してから、記録モードをオンにする必要があります。

2. アクションモデルのツールバーで［**Start Animation**］ボタンをクリックするか、キーボードの <**F5**> キーを押します。ツールバーのツールがすべてアクティブになり、ホストとの接続が確立され、ネイティブ環境ペインがアクティブになります。

3. ［**Step Into**］ボタンをクリックします。最初の Check Screen アクションが選択されます。

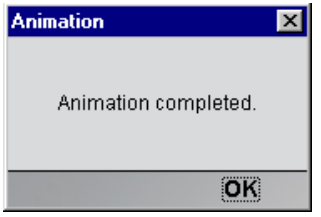


4 ［**Step Into**］ボタンをもう一度クリックします。Check Screen アクション (前を参照) が実行され、次のアクションが選択されます。

5 ［**Step Into**］ボタンを繰り返しクリックして、アクションを 1 つずつ実行します。

6 希望に応じて他のボタン (［Step Over］、［Run To Breakpoint］、［Pause］など) をクリックして、コンポーネントの実行を制御します。アクションの行でマウスをクリックして、<F2> を押すか、［Set Breakpoint］ボタンを使用すると、ブレークポイントを実行中いつでも設定できます。

7 アニメーションが完了すると、次のメッセージが表示されます。



画面にまたがるデータセット

T27 ベースの計算は、他の計算 (他の端末ベースの処理を含む) と次のような点で異なります。

- データセットの取得に、Unisys ホストと通信の繰り返しが必要な場合があります。たとえば、1 つのクエリで多くの画面にわたるデータを取得する場合、複数の「ページを進める」コマンドを使用して取得する必要があります。
- 複数の画面にまたがる情報は、最後の画面にその一部が複製される場合が (よく) あります。

これらの点のために、(アクションモデルを利用する) T27 処理の自動化は困難になります。次に、これらの問題に対する対処方法およびその例について説明します。

複数画面

T27 計算では、頻繁に複数の画面にまたがるデータセットをキャプチャする必要があります。その場合、大部分の Unisys ホストでは、［Action］フィールドに「+」記号が表示され、データが次の画面に続いていることが示されます。

しかし、何ページの画面にデータが存在しているか明白でない場合があります。一般的に、判断の方法は、最後の画面になると［Return］に変わる、［Action］フィールドの + 記号の存在だけであるといえます。

ここで重要なことは、(可能性として) クエリが複数の画面にわたる情報になる場合、Repeat/While アクションを使用して情報が含まれているすべての画面を繰り返したり、情報が含まれている画面が終了した場合に停止したりする準備をしておく必要がある、ということです。繰り返しを停止する場合、それを判断する特別なカスタム論理を作成する必要があります。論理は、次の 1 つまたは複数の方法により異なる可能性があります。

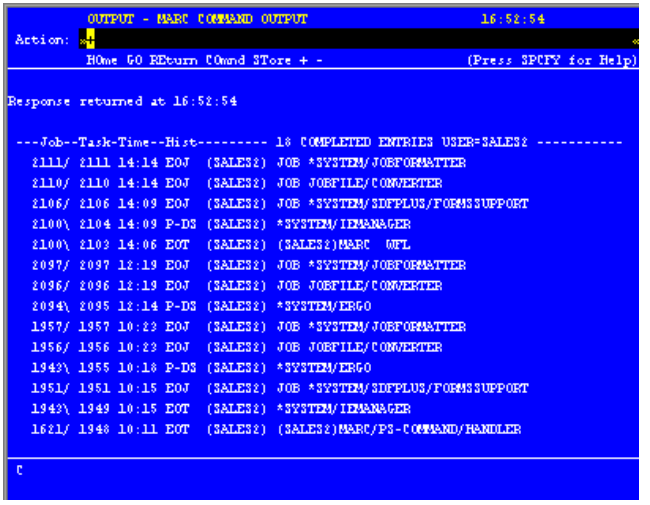
- 情報を「スクレーピングする」ことにより画面の総数 (可能な場合は最初の画面も) を判断する。
- 「レコードの総数」(この情報がわかる場合) を 1 画面あたりのレコード数 (事前にこの数がわかっている場合) で割り、1 を加える。
- 1 画面ずつ表示して、空白のレコードが検出された場所で区切る。
- 特別な文字列 (［End］または［Go Back］など) が検出されるまで 1 画面ずつ表示する。
- 2 つの同一画面が続けて表示されるまで 1 画面ずつ表示する。

明らかに、使用する方法は適用するホストアプリケーションの実装により異なります。

余分なデータの処理

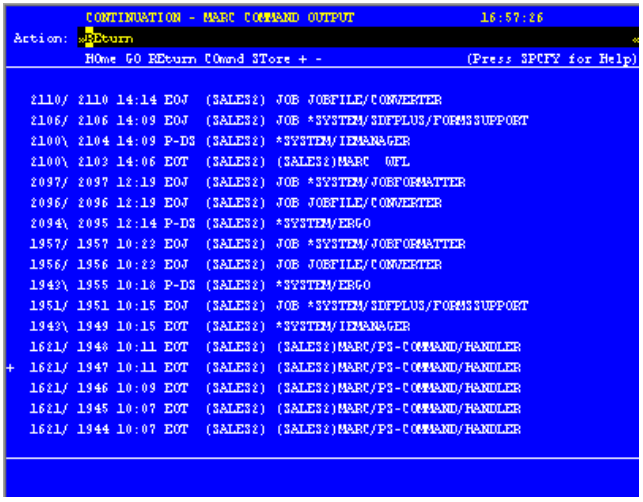
T27 ホストアプリケーションでは、通常、複数画面セットの最後の画面に前の画面からのデータが「当てられて」います。全画面の表示がそのように維持されています。

次の 2 つのスクリーンショットについて考えます。最初のスクリーンショットには、2 画面にわたる情報を返すコマンドを送信した後の、最初の画面に当たる情報が表示されています。最初の画面で、［Action］フィールドの + 記号によって、データがさらに続くことが示されている点に注意してください。



［Transmit］キー (または <F12>) を押すと、2 番目の画面が表示されます。この 2 番目の (また、この場合では最後の) 画面では、いくつかの注目すべき点があります。

- ［Action:］フィールドで、+ 記号の代わりに「REturn」が使用されています。ここで［Transmit］キーを送信すると、［Job］および［Task Display Menu］が返されます。
- 2 番目の画面には、ジョブ番号 2111/2111 を除いては最初の画面と同じレコードが表示されます。ジョブ番号 2111/2111 が表示されないのは、2 番目の画面では 17 行のジョブしか表示できないため、4 つの 1621 ジョブに対してスペースを確保する必要があったためです (最初の画面には 3 行のヘッダ情報があったため、データは 14 行しかありませんでした)。この画面の大部分では、余分なデータが表示されています。
- 画面には、別の + 記号が表示されており、今度は画面上で最後から 4 番目のジョブが示されています。MARC では、リストが分割される場所およびデータの重複がなくなっている場所を確認するための便利な方法が利用できます。



多くの場合、この種の余分なデータはキャプチャしないようにします。幸い、MARC を使用すると、リスト内でデータが新しくなっている最初の列の左側に + 記号を配置することによって、余分なレコードを簡単に検出して、削除できます。この方法は、ECMAScript に加えて、重複のないリストを維持するための簡単で便利な方法として使用できます。この操作を実行するための基本的な手順は、次のとおりです。

- 画面の名前を確認する Repeat/While ループを入力します。
- 画面が続行するかどうかに基づいて Switch ステートメントを作成します。
- 各 Switch ステートメント内で Repeat While ループを入力して、各レコードを取得して、変数に配置します (前の例を参照)。
- ループが完了したら、Transmit キーを送信して次の画面に進みます。

2 つの画面のデータをスクロールして、無駄なデータを削除するアクションモデルの全体像は次のとおりです。

注記 : この大きなサイズの画面に対処するために、このアクションモデルではこの章の「T27 セッションの記録」で終了した箇所が取り上げられています。


MAP Screen.getText(9,35,24) TO $Output/MARCOUTPUT/COMPLETEJOBS
While you are NOT in the JD menu, do the following
WHILE Screen.getText(1,11,2) != 'JD'
Loop Actions
Begin a Switch/Case statement checking for the + or REturn:
SWITCH Screen.getText(2,11,1)
CASE: "+"
If you are on the first or middle screens, verify that with a Check Screen:
CHECK SCREEN for Expression: Screen.getText(1,11,6)=="OUTPUT"
There are 21 rows of data on the first screen, with 8 rows of header info.
Since we start counting at 0, we need to iterate 13 times (21-8)
WHILE rowindex <= 13 INDEXED BY rowindex
Loop Action
Remember to always account for the 8 rows of header info on iteration
CALL var jobname = Screen.getText(rowindex+8,35,35)
MAP jobname TO Output.createXpath("MARCOUTPUT/JOBNAME" + rowindex)
Another check screen check here, and act on it once we determine...
CHECK SCREEN for Expression: Screen.getText(1,11,6)=="OUTPUT"
IF Screen.getText(1,11,6)=="OUTPUT"
TRUE
If we are on the right screen, we want to get to the next screen by Transmitting.
SEND KEY TRANSMIT
CHECK SCREEN for Expression: Screen.getText(1,11,4)=="CONT"
FALSE
LOG "Wrong screen. Verify Screen Wait times." TO System Output using Log Level 5
Break
CASE: "R"
Verify that we are on the Next Screen
CHECK SCREEN for Expression: Screen.getText(1,11,4)=="CONT"
WHILE rowindex < 21 INDEXED BY rowindex
Loop Action
Find out where non-redundant data starts by looking for the + sign
IF Screen.getText(rowindex,1,1)=="+"
TRUE
CALL newrow=rowindex
FALSE
LOG "Continue" TO System Output using Log Level 5
CHECK SCREEN for Expression: Screen.getText(1,11,4)=="CONT"
There are 21 rows of data in the second screen, including 4 rows of header info
Subtract the row number where new data starts from the total number of rows on the second screen.
WHILE rowindex <= (20 - newrow) INDEXED BY rowindex
Loop Action
CALL var jobnum = Screen.getText(rowindex+newrow,4,5)
MAP jobnum TO Output.createXpath("MARCOUTPUT/JOBNUM" + rowindex+newrow)
CALL var tasknum = Screen.getText(rowindex+newrow,10,4);
MAP tasknum TO Output.createXpath("MARCOUTPUT/TASKNUM" + rowindex+newrow)
CALL var timecomp = Screen.getText(rowindex+newrow,15,5)
MAP timecomp TO Output.createXpath("MARCOUTPUT/TIME" + rowindex+newrow)
CALL var jobname = Screen.getText(rowindex+newrow,35,35)
MAP jobname TO Output.createXpath("MARCOUTPUT/JOBNAME" + rowindex+newrow)
SEND KEY TRANSMIT
CHECK SCREEN for Expression: Screen.getText(1,11,2)=="JD"
Default
LOG "Unknown Screen " TO System Output using Log Level 5

信頼性の高い T27 コンポーネントを作成するためのヒント

次のヒントは、信頼性の高い T27 コンポーネントを作成する上で役立ちます。

- 常に、Send Key アクションの後には、Set Screen Text アクションを続けます。
- 常に、Send Key アクションの後には、Check Screen アクションを続けます。
- Check Screen アクションで使用する［Screen Wait］のデフォルト値は、はじめて接続リソースを作成する際に設定されることに注意してください。デフォルトの画面待機時間の値を変更するには、接続リソースのプロパティを変更する必要があります。
- ロードに対応したアプリケーションに対して、画面待機時間の値を大きくしなければならない場合があることにも注意してください。慎重にテストを行うと、この種の問題が明らかになります。
- 以前に記録したアクションモデルを編集する場合は、注意が必要です。単一の Set Screen Text アクションを削除または編集すると、アクションモデルの全体がコースから外れる可能性があります (そうなることは間違いないでしょう)。

T27 コンポーネントエディタでの他のアクションの使用

Set Screen Text アクション、Send Key アクション、および Check Screen アクションに加えて、すべての標準の Composer アクションも使用できます。［Action］メニューには、基本的なアクションおよび高度なアクションの両方のリストが表示されます (次の表を参照)。

表 5-1

基本的なアクション	説明
Comment	アクションモデルを記録します。特に、アクションモデルに［Decisions］または［Repeats］、あるいはその両方が使用されている場合、コメントを使用して処理を明確にすることができます。
Component	別のコンポーネントを実行し、呼び出されたコンポーネントで受け渡しするランタイム DOM を定義します。
Decision	指定した条件に基づいて、アクションの 2 つのセットから 1 つを実行できます。コンポーネントの実行で指定した条件がどのように解決されるかによって、True または False へのパスの分岐を処理します。
Declare Alias	特定の XPath 式に独自のカスタムラベルを適用できます (特定のアクションモデルのスコープ内でのみ有効です)。

Function	ECMAScript スクリプト関数または以前に作成したカスタムスクリプトのいずれかを実行します。カスタムスクリプトは、Composer のカスタムスクリプトリソースエディタを使用して作成できます。
Log	コンポーネントで指定されているさまざまなログファイルに情報を書き込みます。ログのタイプには、システム出力、システムログ、およびユーザログの 3 種類があります。
Map	要素データをあるXML DOMから別のXML DOMへ転送し、オプションで変換します。
Send Mail	コンポーネントの実行中、指定した電子メールアドレスに自動的に電子メールを送信します。
Switch	入力値と大文字小文字の値との一致に基づいて、プログラムの制御をアクションの特定のブロックに分岐させることができます。これは、長く、読み取りが困難な if/else (Decision アクション) のチェーンを排除する場合に使用できる便利なアクションです。

次の表のアクションは、コンポーネントエディタの［**Action**］メニューで、［**Advanced**］、［**Data Exchange**］、［**Process and Repeat**］の順にサブメニューを選択すると利用できます。

表 5-2

高度なアクション	説明
Apply Namespaces	NameSpace プリフィックスを上書きしたり、新しい NameSpace プリフィックスを宣言したり、または NameSpace 全体を無視したりする方法を提供します。
Raise Error	条件を評価し、true の場合は ERROR と呼ばれるグローバル変数に式のコンテンツを記述します。単独で使用された場合は、例外をスローしてコンポーネントを停止し、サービスに制御を返します。Try On Error アクションの Execute 分岐内で使用された場合は、評価され、On Error 分岐でアクションに制御が渡されます。
Simultaneous Components	2 つまたはそれ以上のコンポーネントを同時に (つまり、マルチスレッド方式で) 実行できるようにします。
Transaction	非コンテナ管理サービスの一部として配備されるコンポーネントで User Transaction コマンド (開始、コミット、およびロールバックなど) を呼び出したり、コンテナ管理 EJB 配備の一部となるコンポーネントで setRollbackOnly を呼び出したりできます。
Try On Error	一連のアクションを実行することで、エラーを生成するアクションに応答します。Try On Error アクションは、本質的にエラートラップおよび解決を行うアクションです。

高度なアクション	説明
XSLT Transform	XSL ファイルの指示に従って XML ファイルを変換します。出力は、一般的に Web ブラウザに XML ファイルを表示するために使用されます。

表 5-3

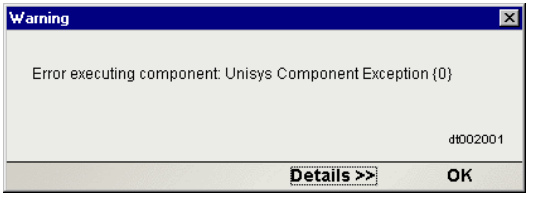
Data Exchange アクション	説明
UR_/File Read	XML でないファイル形式を Composer に読み込むことができます。
UR_/File Write	ファイルを XML 以外の別の形式で書き込むことができます。
WS Interchange	WSDLリソースで定義されたメッセージおよび操作を使用してWebサービスを実行します。
XML Interchange	外部 XML ドキュメントをコンポーネントの DOM に読み込んだり、外部 XML ドキュメントにコンポーネントの DOM を書き込んだりします。読み込み / 書き込みメソッドには、ファイル、FTP、HTTP、および HTTPS プロトコルを使用した Get、Put、Post、および Post with Response が含まれます。

表 5-4

Repeat アクション	説明
Break	Repeat for Element、Repeat for Group、または Repeat While ループの実行を停止し、ループ外で次のアクションの実行を続行します。
Continue	Repeat for Element、Repeat for Group、または Repeat While ループで現在のループ反復の実行を停止し、次の反復で同じループの一番上から続行します。
Declare Group	複数回発生する要素に基づきグループを作成して、グループに名前を付けることができます。グループは、Repeat for Group アクションで使用されます。
Repeat for Element	DOM ツリーに指定した要素が発生するごとに 1 つまたは複数のアクションを繰り返します。Repeat for Element アクションでは、複数回発生する要素に基づき、ループを作成できます。
Repeat for Group	グループの各メンバーに対して 1 つまたは複数のアクションを繰り返します。Repeat For Group アクションでは、データを再作成して、データを集約計算できます。
Repeat While	ループを作成することで、1 つまたは複数のアクションを繰り返します。Repeat While アクションでは、処理ループを任意の有効な ECMAScript 式に基づかせることができます。

エラーおよびメッセージの処理

T27 コンポーネントをテストする際には、Set Screen Text アクション、Send Key アクション、または Check Screen アクション、あるいはこれらすべてに関するエラーが表示されることがあります。結果は、次のようなダイアログボックスで示されます。



この節では、考えられるエラーの条件およびそれに対処する方法について説明します。

Check Screen エラー

実行時に発生するエラーの大部分は、Check Screen アクションに関連します。一般的に、Check Screen エラーは、タイムアウトエラーであり、Check Screen 設定ダイアログボックスで指定した実行許可条件が「画面待機のタイムアウト時間内に満たされなかった」ことを意味します。エラーダイアログボックスの［**Details**］ボタンをクリックすると、この内容が確認されます。そのため、ホストの応答の遅れが実際の問題であるかどうか最初に判断する必要があります (その場合の解決策は、問題の Check Screen アクションに対する画面待機時間の値を大きくすることです)。画面待機時間の値を大きくしてもエラーが引き続き発生する場合は、エラーの原因が Check Screen アクションでの不正または不適切な実行許可条件によるものと確信できます。

「Screen Check Expression {0} was evaluated as false」

このエラーメッセージは、Check Screen 実行許可条件に対して使用した ECMAScript 式で、実行時に *false* が返された場合に発生します。ここでも、この種のエラーは、単純にホストの応答の遅れ (タイムアウト) によって、トリガされることを認識しておくことが大切です。ホストの応答が遅い場合、ECMAScript 式が「タイムアウトの時点で画面バッファに存在する内容すべて」によって評価されます。データが着信しない (または不十分なデータである) 場合、式が評価され、結果として「false」が返されます。

この種の問題を解決するためには、この Check Screen アクションの画面待機時間の値を大きくするか (ホストの待ち時間が問題と考えられる場合)、ECMAScript 式での論理を変更します。

Set Screen Text エラー

一般的に、Composer で Set Screen Text アクションから生成されるエラーの数は多くありません。この理由は、任意の内容を自由に画面に送信することが許可されているため、発生するエラーは、アプリケーション側の問題である可能性が高くなるからです。Unisys ホストは、受け入れる入力に対して非常に選別的です。Set Screen Text アクションで送信するテキストがホストで予期されたテキストと異なる場合、ホスト側のエラーが送信され、残りの Composer アクションモデルは、期待どおりに続行されません。ここで問題を回避するには、Set Screen Text アクションと Send Key アクションのすべての組み合わせに対して、常に対応する Check Screen アクションが存在していることを確認します。

「不正な」アクションの検索

アクションモデルが大きい (数十、数百の Set Screen Text アクション、Send Key アクション、および Check Screen アクションを含む) 場合、単純にエラーの原因となるアクションを検索することが困難になります。問題のアクションを検索する 1 つの方法は、次のとおりです。

1. エラーのダイアログボックスで、テキストの一部を選択して、コピーします (必要な場合は、［Details］ボタンをクリックして、完全なエラーの説明を表示します。カーソルの座標といった関連するテキストを選択してから、<Control>-<C> を使用してコピーします)。
2. アクションモデル内をクリックします。
3. <Control>-<F> を使用して、検索を開始します。
4. エラーテキストを検索ダイアログボックスに貼り付けます。
5. 検索を実行します。

もちろん、同一の実行許可条件に基づく複数の Check Screen アクションがある場合、前の方法が必ずしも役に立つとは限りません。その場合、アクションモデルの中間点にブレークポイントを設定し、コンポーネントを実行します。エラーが発生しない場合は、元のブレークポイントからアクションリストの最後までで中間となる場所にブレークポイントを移動します(エラーが発生する場合は、アクションリストの最上部から下に 4 分の 1 の場所にブレークポイントを設定します)。コンポーネントをもう一度実行します。ブレークポイントの場所を変え続け、毎回最後のブレークポイントとアクションリストの最上部または最下部の距離が「半分」となる場所に適宜設定します。このようにすると、問題となるアクションの場所を迅速に絞り込むことができます (この「バイナリ検索」方法を使用すると、128 個のアクションを含むアクションモデルをわずか 7 回でデバッグできます)。

パフォーマンスについて

タイミングの呼び出しについて個別のアクション (またはアクションのブロック) を調整することにより、アクションモデルの動作を秒単位で調整することができます。

➢ アクションの時間を調整する

1. アクションモデルをクリックし、時間を調整するアクションのすぐ前に、新しい **Function** アクションを挿入します (マウスを右クリックして、［**New Action**］>［**Function**］の順に選択します)。
2. Function アクションで、次の形式で ECMAScript 式を入力します。

```
startTime = Number(new Date)
```

3. 時間を調整するアクションのすぐ「後」に、新しい **Function** アクションを挿入します。
4. Function アクションで、次の形式で ECMAScript 式を入力します。

```
endTime = Number(new Date)
```

5. `endTime - startTime` を一時 DOM の要素にマップする Map アクションを作成します (マウスを右クリックして、［**New Action**］>［**Map**］の順に選択します)。
6. コンポーネントを実行します (メインツールバーの［**Execute**］ボタンをクリックします)。

アクションモデルのプロファイルを詳しく行うと、実行時間の大部分が Check Screen アクションに費やされていることがわかります。そこで、次の 2 つの考慮事項が生じます。

- ECMAScript の式 (Map アクションまたは Function アクション、あるいはその両方) がコンポーネント全体のパフォーマンスに与える影響はほとんどありません。
- コンポーネント全体のパフォーマンスは、Check Screen アクションで画面待機時間の値を慎重に調整することで変化します。

最後に、配備されたサービスをアプリケーションサーバでテストするまで (そして信頼性が証明されるまで)、テストが本当に完了したわけではないことに注意してください。

共有する接続に関するその他のパフォーマンスの最適化については、次の章「ログオンコンポーネント」を参照してください。

6 ログオンコンポーネント、接続、および接続プール

この節では、配備されたサービスのパフォーマンスを最大化するために設計された、T27 Connect で使用できる特定の機能について説明します。

T27 端末セッションパフォーマンスについて

バックエンド接続を使用する任意のサービスの全体的なパフォーマンスは通常、接続を確立してホストとの対話を開始するために費やされる時間に基づきます。待機時間を考えると、接続を取得することは「高価」な作業です。これに対応する手段として、「接続プール」があります。これは、中間プロセス (アプリケーションサーバ自体、またはサーバに関連付けられていないメモリ常駐のバックグラウンドプロセス) で、事前に確立、認証された接続を維持して、クライアントアプリケーションまたはエンドユーザの間の接続の「分配」を監視するスキームです。

接続プールによって、接続を開いたり、ホストへの認証を行う際の待ち時間が解決されます。ただし、端末ベースのアプリケーションでは、セッションの最初の正式なアプリケーション画面に到達するために、メニュー選択や設定画面の移動によってドリルダウンすることによって、かなりの時間が費やされる可能性があります。このため、プールを通じて接続が再使用されたとしても、セッションを通じたオーバーヘッドによってパフォーマンスに重要な影響が及ぼされる可能性があります。

Composer では、端末セッションの特定のドリルダウンポイントでオープン接続を維持できる特別な種類のコンポーネント (「ログオンコンポーネント」と呼ばれます) によって管理される、接続プールを提供することによってこの問題に対処しています。このコンポーネントによって、クライアントはセッションの適切なポイントで即時、トランザクションを開始できます。

ログオンコンポーネントが必要な場合

ログオンコンポーネントは、複数の状況で役立ちます。

- システム内の複数のセキュリティチャレンジに基づいた複数階層のプールが必要である場合 (たとえば、ユーザはネットワークに接続するためのログオン認証情報、メインフレームに接続するための別のログオン認証情報、またデータベースに接続するためのログオン認証情報など複数の認証情報が必要である場合があります)。連続したログイン要件によって、複数ログオンコンポーネントの使用が必要とされる可能性があります。
- サービスでステートフルな「セッションベース」の接続が必要である場合
- 接続プールを通じたパフォーマンス上の利点を必要とする場合。

負荷下でのパフォーマンスが優先度の高い問題ではなく、接続の必要性が比較的複雑ではない場合、ログオンコンポーネントを使用する必要がまったくない可能性もあります。ただし、設計時にデスクトップマシンでサービスをテストすることによって、パフォーマンスが適切であるかどうかを確認することはできません。

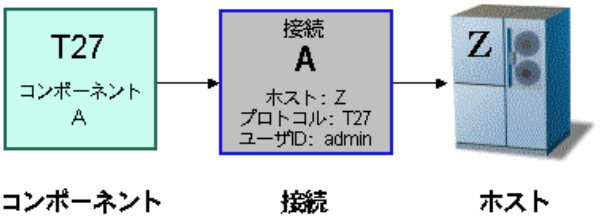
T27 コンポーネントエディタで作成されたコンポーネントおよびサービスは設計時では素早く実行されるように感じられます (たとえば、アニメーションモードなど)。ただし、サーバで 1 秒ごとに何十ものまたは何百もの要求が受信されるような負荷下にある実際の状況を考えると、全体的なトランザクション時間においてセッションオーバーヘッドが重要な要因となる可能性があります。この章で説明されている特別なパフォーマンス拡張機能を使用する必要があるかどうかを判断する唯一の方法は、実際に考えられる「最悪の状態」を想定したテスト状況下で、サーバに対して負荷テストを実行することです。

接続プールアーキテクチャ

T27 Connect をインストールすると、3 種類の接続リソースが接続の作成ウィザードに追加されます。

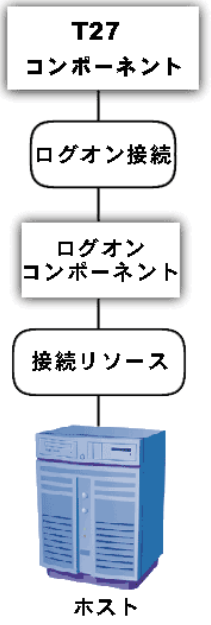
- T27 接続
- T27 マルチブリッジ接続
- T27 ログオン接続 (以下、「ログオン接続」と呼びます)

T27 マルチブリッジ接続は、ホストに戻る接続の数を最小化するサーババージョンであり、追加されたセキュリティも含まれています。T27 接続とは、実際の端末「接続」で、T27 コンポーネントでの使用時にホストシステムとのセッションを確立できます。この接続タイプを、このガイドでは使用しています。



T27 接続リソースは、必要なときにホストと個々に接続できるように設計されています。接続はジャストインタイムで行われ、クライアントが完了したらすぐに切断されます。再使用されることはありません。

一方で、ログオン接続は異なります。ユーザ ID およびパスワードのプールが定義され、それぞれ独自の接続を行えます。ログオン接続はまた、間接層としての役割も果たし、クライアントはホストプログラムで開始する必要のある正確なポイント (正確な画面) でホストに接続できます。このエントリポイントロケーション動作はログオン「コンポーネント」によって可能になります (ログオン接続では、実際の接続を取得するために常にログオンコンポーネントが使用されます)。アーキテクチャは次の図に示されています。



ホストに到達するために、「接続リソース」は常に必要とされます (これは、T27 コンポーネントを使用する任意の Composer サービスに当てはまります)。内容を簡潔にするために、この図では接続リソースがホストに直接接続されていますが、実際には、セキュリティ上の理由からその間に委任階層が存在する可能性があります。

「ログオンコンポーネント」には、ホストプログラムで目的の画面を検索するために設計されたアクション (アクションモデル) が含まれています。このドリルダウンロケーションは、このログオンコンポーネントを使用する任意のアップストリームプロセスにおいて、トランザクションの有効なエントリポイントとなります。ログオンコンポーネントは、「物理的」な接続 (接続リソースによって表されています) と論理階層 (T27 コンポーネントで表されています) との間の仲介役であると考えられます。

T27 コンポーネント (図の上部) でログオンコンポーネントを使用するには、「ログオン接続」リソースの援助を登録する必要があります。ログオン接続は、T27 コンポーネントおよびログオンコンポーネント間の架け橋としての役割を果たします。

前に説明されたさまざまなオブジェクトの種類および役割は、次の表で簡単に説明されています。

オブジェクト	役割
T27 接続リソース	ホストシステムとの接続の確立を許可します。
ログオンコンポーネント	アクションモデルに Logon、Keep Alive、および Logoff のアクションブロックが含まれる特別なタイプのコンポーネント。このコンポーネントは、ホストプログラムの特定の起動画面で接続を維持できます。
ログオン接続	ユーザIDおよびパスワードのプールを指定のログオンコンポーネントタイプと関連付ける特別なタイプの接続リソース。ランタイム時には、要求に応じてクライアントプロセスに対する接続が 1 つの接続あたり 1 つのログオンコンポーネントで確立 (および再使用) されます。関連付けられたログオンコンポーネントのために (前を参照)、プールの各接続によってホストプログラムの指定のポイント (特定の起動画面) にアクセスできるようになっています。
T27 端末コンポーネント	特定の T27 処理 (またはトランザクション) に対するビジネス論理を備えたアクションモデルを含みます。

プールでのログオン接続の役割

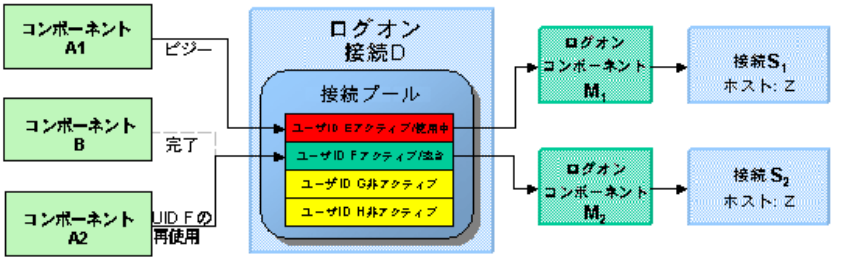
ログオン接続は、「プール」(ランタイム時での接続インスタンスおよびログオンコンポーネントインスタンスの共有) を管理するという点で通常の「ホスト直接」の接続リソースとは異なります。

Composer サービスの場合、プールによってランタイム時に (オープン) 接続の再使用が可能となるだけではなく、配備されたサービスの有効な帯域幅も増加します。通常の接続リソースを使用する T27 コンポーネントを設計した簡単なケースを例にとって考えてみましょう。接続リソースを作成する場合、ランタイム時にコンポーネントがホストにログインできるように、使用するリソースのユーザ ID およびパスワードを指定する必要があります。実際に実行しているコンポーネントのインスタンスでその接続が使用されている場合、そのコンポーネントのその他のインスタンスは、同じ認証情報を使用しているホストにはログインできません。サービスの帯域幅は接続されている 1 つのインスタンスに制限されています。

一方で、ログオン接続の場合、コンポーネントの複数のインスタンスが待機することなくホストにアクセスできるように (それぞれ独自の接続を持ちます)、多数のホスト接続を「ライブ」状態で維持できます。スループットは大幅に増加します。

下の図には、3 つのコンポーネントインスタンス (T27 端末コンポーネント A の 2 つのインスタンスおよび T27 端末コンポーネント B の 1 つのインスタンス) がサーバ上で実行するランタイムケースが示されています。コンポーネント A のインスタンス 1 は、接続を確保するためにユーザ ID「E」を使用しています。このコンポーネントには、独自の専用インスタンスであるログオンコンポーネント M および接続 S があります。

端末コンポーネント B は実行を終了して、(UID「F」によって定義された認証情報を通じて確立された) 接続を破棄しています。接続プールが有効であるため、コンポーネント B のダウンストリームリソース (ログオンコンポーネントインスタンス M2 および接続インスタンス S2) は単純に破棄されず、そのまま残ります。その結果、端末コンポーネント A2 は以前端末コンポーネント B によって維持されていた M2/S2 リソースインスタンスを取得 (再使用) できます。



この図では、ログオン接続 D は 4 つの UID (ユーザ ID または認証情報 : A から F) に基づく 4 つの接続と関連付けられています。1 つは使用中で別のもの (UID「F」) はライブ状態ですが使用中ではなく、残りの 2 つは非アクティブですが使用可能な状態です (有効な UID が割り当てられているため、これらの 2 つの接続はいつでもライブ状態にできます)。

必要なプールの数

異なる複数の T27 コンポーネントが同じ接続プールを使用することは可能です。また、異なるコンポーネントが異なるプールを使用することも可能です。これは、異なるログオン接続を意味します。

サービスで必要なログオン接続リソースの数 (プール数) を判断する上で重要な要因は、プロジェクトのさまざまなコンポーネントによって必要とされる異なる起動画面 (またはエントリポイント画面) の数です。端末コンポーネント A がホストアプリケーションのある特定の起動画面で作業を開始する必要があるものの、異なる画面で開始する必要のある別のコンポーネント、端末コンポーネント B も設計されていると仮定します。コンポーネント A およびコンポーネント B は個別のログオン接続が必要であり、そのログオン接続は個別のログオンコンポーネントを指定します (任意の接続プールでは、プールの各ユーザが同じ画面で開始する必要があるように Composer オブジェクトが共有されます)。

プールに必要な構成要素

ログオン接続、ログオンコンポーネント、および接続リソースの組み合わせは、接続プールの基礎を構成します。ホスト層から開始して、チェーンが増加してゆきます。

- 「接続リソース」は、Unisys ホストとの接続を確立するために必要な最も基本的なパラメータを定義します。接続プールが有効な場合、このオブジェクトのランタイムインスタンスはライブな状態のまま存続し、再使用されます。
- 「ログオンコンポーネント」は、ホストプログラムの特定のエントリポイントに到達するために必要な一連の手順 (アクション) を定義します (ランタイム時には、ホストプログラムの特定の画面ロケーションに到達して、すぐに使用できる状態を維持するためにこのコンポーネントのインスタンスでは実際にこれらの手順が実行されます)。接続プールが有効な場合、このオブジェクトのインスタンスはライブな状態のまま存続し、再使用されます。
- 「ログオン接続」は、接続「プール」を定義するために必要なすべての情報が含まれた特別なタイプのリソースです。このリソースは、プール管理情報をカプセル化するように設計されており、直接ホスト接続は確立されません。その代わりに、これらの役割をログオン接続に委任していいます (ログオン接続は、これらの役割を適切な接続リソースに委任します)。

プールの実装方法

プールに必要なさまざまな構成要素を作成するために、次の基本的な手順に従います (各手順については、この後の節で詳細に説明します)。

1 最初に、このガイドの第 2 章の説明に従って、基本的な T27 接続リソースを作成します。

2 次に、手順 1 で定義された接続リソースを使用するログオンコンポーネントを作成します。このプロセスの一環として、ホストプログラムの特定のポイントに移動するように設計されたアクションモデルを作成します。

3 (手順1で定義されたリソースを通じて)基本的な接続を行うために(手順2で作成した) ログオンコンポーネントに依存する特別なタイプの接続リソースであるログオン接続リソースを作成します。

4 最後に、T27 端末コンポーネントを作成して、それを手順 3 のログオン接続リソースに関連付けます。

これらの手順は、「接続プールの作成」でのディスカッションを始めとして、後で詳細に説明されています。ただし、この節に進む前に、ログオンコンポーネント (およびログオン接続) の背後にある設計原理について理解する必要があります。ログオンコンポーネントを使用することなくログオン接続を作成することはできないため、ログオンコンポーネントから開始します。

T27 ログオンコンポーネント

ログオンコンポーネントは特別なタイプのコンポーネントです。このコンポーネントにはアクションモデルがありますが、接続リソースとしても使用できます。このタイプのコンポーネントのアクションモデルは、複数の T27 接続コンポーネントによって使用される接続を管理するように設計されています。大部分において、ログオンコンポーネントは T27 端末コンポーネントと同じです。異なる点は次のとおりです。

1 ログオンコンポーネントでは、アクションモデルは接続管理タスクを基準として構成されています。これらのタスクは、Logon アクション、KeepAlive アクション、および Logoff アクションなどの特別なアクションを通じて実装されます。

2 ログオンコンポーネントは別のコンポーネントまたはサービスによって直接呼び出しされません。呼び出しはログオン接続が制御します。

注記 : ログオンコンポーネントは、ログオン接続と組み合わせて使用する必要があり、また組み合わせてのみ使用できます。

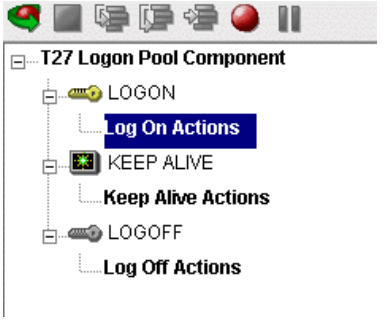
ログオンコンポーネントを直接呼び出す代わりに、(たとえば) コンポーネントアクションを使用して、ログオンコンポーネントをログオン接続と呼ばれる特別な接続リソースと関連付けます。T27 端末コンポーネントは、ログオン接続を通じて実行されます。また、ログオン接続はログオンログオンコンポーネントを実行します。

Logon アクション、Keep Alive アクション、および Logoff アクション

ログオンコンポーネントには、全体的なパフォーマンスにおいて重要な要因となる複数の画面管理機能が用意されています。これらの機能は、Logon アクション、Keep Alive アクション、および Logoff アクションに関連して実装されます。

- 「Logon アクション」は、ホスト環境内を移動し、ホストシステムの目的の「起動画面」で停止します。接続は、プールからのユーザ ID を使用してアクティブにされます。その後で接続を再使用する T27 端末コンポーネントでは、コンポーネントがすでに起動画面に存在し、新しいセッションを使用した場合と同様に起動画面に移動する際のオーバーヘッドが発生しないため、パフォーマンス上の利点が得られます。
- 「Keep Alive アクション」は、2 つの重要なタスクを実行します。まず最初に、Keep Alive アクションは、ホストがホストによって定義された標準のタイムアウト時間内に使用されない場合に接続を切断しないようにします。次に、これらのアクションは、接続が切断されることを防ぐ (最も重要なタスク) ために必要な Keep Alive アクションを実行した後でも、接続が常にホストの起動画面に配置されることを確実にします。
- 「Logoff」アクションは、接続の終了時に、ユーザ ID によってプールから確立された接続すべてに対して、指定した方法でホスト環境を終了します。

これらのアクションおよびその意味については、後で詳細に説明します。今の段階では、ログオンコンポーネントを最初に作成したときにこれらの 3 つのアクションが自動的に作成されるということを覚えておいてください。下に表示されているアクションモデルの (空の) Logon、Keep Alive、および Logoff アクションブロックに注意してください。



LOGON アクション

LOGON グループに配置したアクションは、主にホストのセキュリティ画面にサインインしてから、ホストのメニューシステムを使用して起動画面に移動するアクションを指します。この起動画面で、各 T27 コンポーネントのアクションモデルが開始します。ログオンコンポーネントを使用した T27 コンポーネントでは、共通の画面で実行が開始できなければなりません。それ以外の場合は、移動のオーバーヘッドを回避することでパフォーマンスは向上されず、さらに重要な点として、不適切な T27 コンポーネントが動作しなくなります。

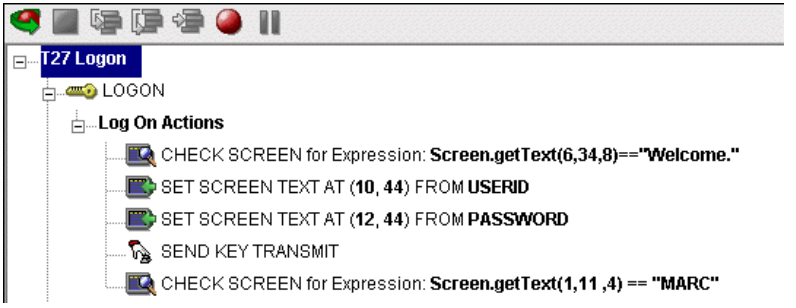
Logon アクションブロックの下には、通常の T27 端末コンポーネントと同じ方法でアクションを作成できます。つまり、ユーザ ID やパスワード (および起動画面に到達するための初期のメニュー選択) などのサインオン情報を入力するために必要なアクションを (リアルタイムで) 作成するために記録機能を使用します。

注記：この場合、ログオン接続プールからユーザ ID およびパスワードを使用することに注意してください (下の「プール接続を使用したログオン接続の作成」の説明を参照)。そのためには、USERID と PASSWORD と呼ばれる 2 つの特別なシステム変数を、画面上の適切なフィールドにマップする必要があります。これらの 2 つの変数を指定することによって、exteNd Composer では次のアクティブな空きプールスロットから値を自動的に検索して、使用できます。

起動画面は、ログオン接続で提供されたユーザ ID プールを使用する T27 端末コンポーネントすべてに対する共通の実行ポイントです。ログオンコンポーネントの Logon アクション (新しい接続が確立されたときに一度だけ実行されます) によって、呼び出し側コンポーネントである T27 端末コンポーネントはホストプログラムの指定された画面で実行を開始します。

ログオンコンポーネントを使用したパフォーマンスの最大化

Logon アクションは、適切な構造でなければならず、下の画面に示すように常に Check Screen アクションで開始および終了する必要があります。



Logon ブロックの最後の Check Screen アクションでは、目的の画面が接続に達する前に T27 コンポーネントに制御が移らないことが保証されます。この処理が行われない場合、T27 コンポーネントが無効な画面で開始して例外が発生し、トランザクションが無効になる可能性があります。

注記： 最後の Map Screen をスキップするログオンコンポーネントをアニメーション表示している場合、これは通常の設計時の動作です。運用環境におけるログオンコンポーネントのアクションは、T27 端末コンポーネントでは常にインタリーブ方法で実行されます。最初から最後までログオンコンポーネントをアニメーション表示すると、イベントの異常なシーケンスが実際には作成されます。この結果、2 つの Check Screen が連続して処理されますが、このことは許可されていません。

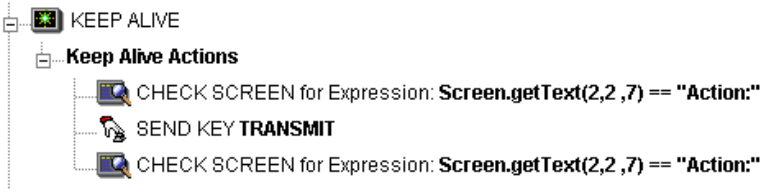
接続の再使用だけではなく起動画面の再使用の結果として、パフォーマンス上の利点が示されます。たとえば、3 つのエントリのユーザ ID プールが完全に使用され、(最終的に) コンポーネントの実行時に 15 回再使用された場合、目的のトランザクションを実行するメニュー項目に移動する際のオーバーヘッドは、3 回だけ発生することになります。同様に、新しい接続がアクティブになった際 (再使用された際ではなく) に、ログオンコンポーネントの上部の Logon アクションは 1 回だけ実行されるため、ホストへのログオンは 3 つだけになります。これは、トランザクション数の多い運用設定で最大のパフォーマンスを取得する上で重要です。

注記： 可能な場合は、Try/On Error アクションを使用して、回復可能であると思われる潜在的なログオンエラーをトラップします。それ以外の場合は、失敗したログオンを確立するために使用したユーザ ID がプールから破棄されるため、T27 の exteNd Composer Enterprise Server Console を使用して破棄された接続を手動でリセットするまで、プールサイズは小さくなり続けます。詳細については、この章の「プールの管理」を参照してください。

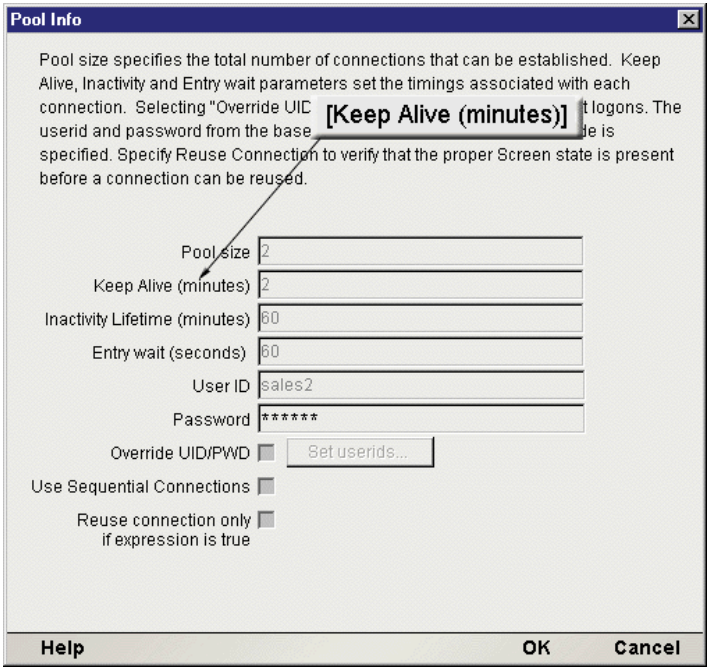
Keep Alive アクション

KEEP ALIVE ブロックとは、再使用できるように接続をそのままの状態で維持するために必要な方法で、「ホストを ping」するアクションを配置する場所です。

Keep Alive アクションには通常、指定した間隔で <Transmit> などのアテンションキーをホストに送信する処理が関わります。ただし、アテンションキーの送信後に画面が起動画面とは異なる画面に変わった場合、Keep Alive セクションでログオンコンポーネントを起動画面に戻す必要があります。起動画面に戻らない場合は、次のコンポーネントが間違った画面で表示され、エラーが発生します。



［Logon Connection］設定ダイアログボックスの［Pool Info］ダイアログボックス (下の「プール接続を使用したログオン接続の作成」の説明を参照) は、Keep Alive アクションを実行する頻度を制御する場所です。ログオン接続プールで、空き接続を 3 分間アクティブに維持するように指定したものの、非アクティブな状態が 2 分続くとホストにより接続が切断される場合、キーボードアクションが 30 秒間隔で発生するように指定することによって、接続がまだアクティブであることをホストに通知することができます。



Keep Alive アクションは、ログオン接続の［Pool Info］ダイアログボックスで定義されている Keep Alive パラメータによって定義された間隔で、複数回実行されます。

Inactivity Lifetime パラメータ (［Pool Info］ダイアログボックスの Keep Alive のすぐ下) によって、接続が T27 端末コンポーネントによって実際に使用されていない場合、その接続を切断する前に Composer で待機する必要のある時間が伝えられます。

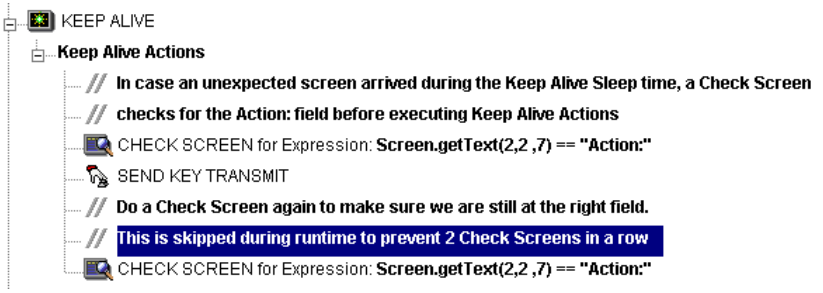
注記： ログオンコンポーネントの「Keep Alive アクション」の実行によって、非活動ライフタイムのクロックがログオン接続でリセットされることはありません。非活動ライフタイムがリセットされるのは、T27 端末コンポーネントを実行した場合のみです。つまり、ライブ接続が実際には使用されなかった場合 (Keep Alive アクションによって単にライブな状態になっている場合)、［Pool Info］ダイアログボックスの［Inactivity Lifetime］の値に基づいてタイムアウトします。ただし、タイムアウトする前に接続が (T27 コンポーネントによって) 使用されると、タイマーはその時点でリセットされます。

Keep Alive ブロック内の最後のアクションは空の「有効」なナビゲーションアクションでなければなりません。この最後のアクションをユーザが無効にした場合、アニメーションは、発生する 2 つの連続した空のナビゲーションアクションのために正しく機能しません。たとえば、ログオンのアクションおよび Keep Alive の最初のアクションが無効である場合、エラーが発生します。

Keep Alive アクションを使用したパフォーマンスの最大化

Check Screen アクションは、［Keep Alive］セクションの開始時と終了時に発生する必要があります。

［Keep Alive］セクションは、接続が切断されないようにするだけではなく、実行が完了したときに起動画面が存在することも確実にする必要があります。このため、最初の Check Screen では、接続が使用できる状態であるが使用されていない間に、ホストから予期しない画面が到着していないことが確認されます。最後の Check Screen では、接続が次の T27 コンポーネントへ早い段階でリリースされないようにします。一般的な Keep Alive ブロックについては、後の説明を参照してください。



Logoff アクション

基本的に、Logoff アクションによって、タイムアウト後にユーザ ID がホストシステム外に適切に移動されます。

Logoff アクションは、接続がタイムアウトになった場合 (つまり、非活動ライフタイムの期限が切れた場合)、または接続が T27 サーバコンソールを介して切断された場合のみ、1 つの接続に対して一度だけ実行されます。

「最良実施」の観点から考えると、Logoff アクションを安全で安定した状態にすることは不可欠です。Logoff アクションの実行中に例外が発生した場合、exteNd Composer でホストとの接続が切断され、UserID がプールに解放されます。ただし、ユーザ ID はホストでアクティブのままになる可能性があります。ログオンをコード化してこの状況に対処しない限り、(非活動のために) ホストでユーザ ID が停止されるまで、プールでそのユーザ ID に次にログオンしようとしても、ログオンが失敗する可能性があります。ログオンに失敗すると、プールからユーザ ID が破棄され、プールサイズと全体的なパフォーマンスが低減する可能性があります。Logon アクションと Keep Alive アクションに関しては、ログオフの終了時に適切な画面となるよう保証する方法は、Check Screen で終了することです。

ログオンコンポーネントのライフサイクル

ログオン接続プールからユーザ ID がアクティブにされるたびに、それに対応するログオンコンポーネントのインスタンスが作成され、そのユーザ ID と関連付けられます。その後、目的の起動画面に到達するまで Logon アクションが実行されます。この時点で、T27 端末コンポーネントの実行が開始します。実行が終了したら、同じログオンコンポーネントを使用した別の T27 端末コンポーネントの実行が同じ起動画面から開始します。

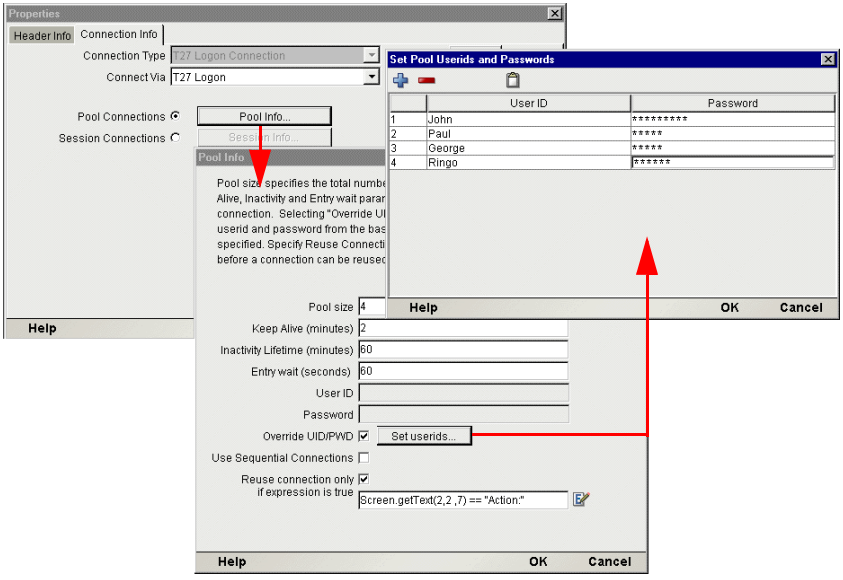
その他のコンポーネントでその接続が要求されていない場合、その接続インスタンスはログオン接続の［Pool Info］ダイアログボックスの［Inactivity Lifetime］および［KeepAlive］の設定によって定義されたアクティブで空の状態 (「アイドル状態」) に入ります。Keep Alive の間隔 (例 : 2 分) が非活動ライフタイム (例 : 120 分) より短い場合は、ホストがタイムアウトになり接続が切断されないよう、適切な間隔 (2 分) で Keep Alive アクションが実行され、Keep Alive 間隔が新規に開始します。

ログオンコンポーネントの実行期間は、ログオンコンポーネントが使用されるログオン接続に応じて異なります。ログオン接続プールで 1 つのエントリがアクティブの場合は、ログオンコンポーネントの 1 つのインスタンスが、ライブな状態でメモリ内に存在します。非活動の状態が続いたために、残りの最後のプールエントリの期限が切れた場合、ログオンコンポーネントインスタンスはスコープから離れます (実行は中止されます)。ログオンコンポーネントの実行を停止する他の唯一の方法は、サーバの T27 コンソールを使用することです。

T27 ログオン接続について

ログオン接続は、T27 接続リソースのような実際の接続オブジェクトではなく、ログオンコンポーネントへのポインタです (ログオンコンポーネントは、従来の接続リソースまたはより介入的なログオン接続 / ログオンコンポーネントのペアのいずれかを通じてホストに接続します)。ログオン接続によって、接続プールを記述するために必要な情報がカプセル化されます。この中には、ユーザ ID およびパスワード、また破棄された接続に対する再試行の間の時間間隔に関連するプール設定などが含まれます。ログオン接続の別の機能は、接続が行われる対象であるすべてのユーザ ID に対して、同じログオンコンポーネントの異なるインスタンスが使用されることを確認することです。

ログオン接続用にユーザ ID のプールを設定する場合に使用するダイアログボックスは、次の図で示されています。矢印は、続きのダイアログボックスを表示するボタンを指しています。



各ログオン接続は、指定のログオンコンポーネントに関連付けられています。さらに、ログオン接続では、次のユーザ ID プール機能が利用できます。

1. ユーザ ID が必要となった場合に、クライアントが安全に接続できるよう事前に複数のユーザ ID の仕様を許可する。
2. ユーザ ID/ 接続が確立されたら、ユーザ ID/ 接続の再使用を許可してユーザ認証と接続の切断が繰り返し行われないようにする。

3 ホストシステムによりサポートされている場合、単一のユーザ ID で複数の接続の使用を許可する。

4 接続がアクティブでない間にホストがタイムアウトにならないよう、接続をアクティブ状態に保つ。

5 アクティブなプールから接続を削除する時間を指定する。

6 完全にアクティブなプールによって空き接続が提供されるまで使用するタイムアウト時間を設定する。

7 ログオン接続で使用するログオンコンポーネントの状態に依存するエラーの処理を指定する。

コンポーネントとログオンの多対一のマッピング

T27 端末コンポーネントの複数のインスタンスまたは異なる T27 端末コンポーネントで、同じログオン接続を使用するには、次の条件を満たす必要があります。

1 すべての T27 端末コンポーネントでは、同じ接続リソースを使用する (その結果、Unisys ホスト、ポート、およびデータエンコードパラメータを共有する) 必要があります。

2 すべての T27 端末コンポーネントには、ホストシステム内にコンポーネントの実行開始場所となる共通の起動画面が必要です (詳細については、「ログオンコンポーネントの作成」を参照してください)。

シングルサインオンを使用した接続プール

ホストシステムのセキュリティで、単一のユーザ ID からの複数のログインがサポートされている場合、この単一のユーザ ID をプールするような状況が生じる可能性があります。これを実現するには、次の手順に従います。

- ログオンコンポーネントで使用される接続リソースでユーザ ID/ パスワードを指定します。
- ログオン接続の［Pool Info］ダイアログボックスで、［Pool Size］に 1 より大きい値を指定します。
- ログオン接続の［Pool Info］ダイアログボックスで、［**Override the UID/PWD**］設定をオフのままにします。

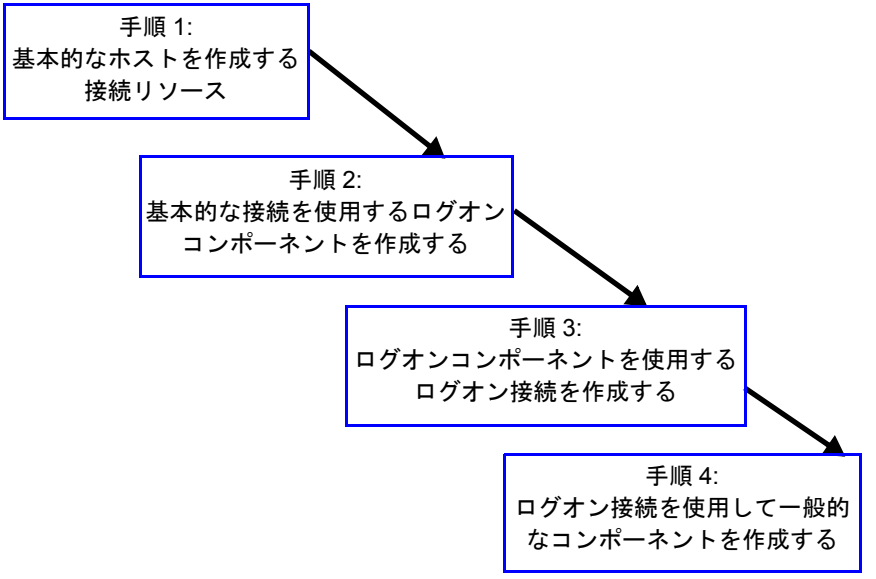
これらの手順を実行すると、各プールスロットで、接続オブジェクトに含まれるユーザ ID とパスワードが使用され、プールからのユーザ ID は使用されません。

接続プールの作成

概要

T27 端末コンポーネントを作成する場合は、最初に必要な接続オブジェクトを作成しなければなりません。同様に、接続プールを構成するオブジェクトを作成する場合は、最初に必要なオブジェクトを作成しなければなりません。これは、(実質的に) ホストで開始し、ホストにアクセスする T27 端末コンポーネントに戻ることを意味します。

接続プールを作成する一般的な手順は、次のとおりです。



基本的な T27 接続の作成

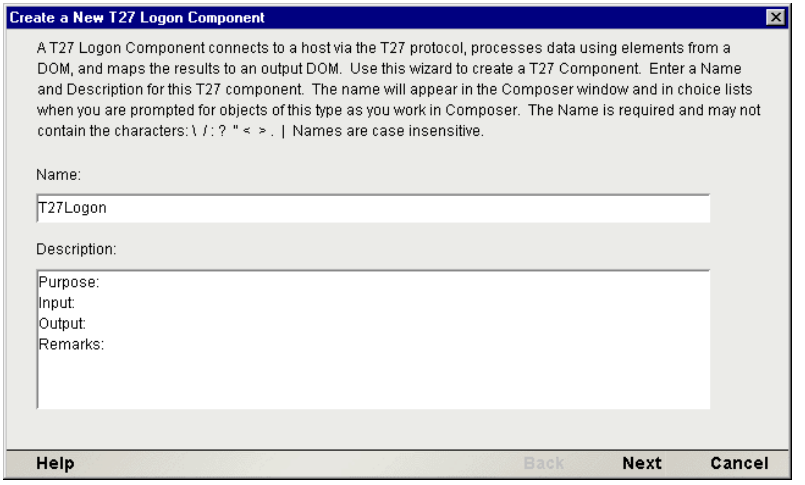
この手順は簡単です。20 ページ「T27 接続リソースを作成する」の説明に従って、新しい接続リソースを作成します。ログオン接続で定義されたユーザ ID とパスワードを後で使用しますが、接続でも同様にユーザ ID とパスワードを定義します。この操作は、次の手順でログオンコンポーネントを定義する場合に必要となります。他の方法としては、単純に既存の接続リソースを使用することもできます。

ログオンコンポーネントの作成

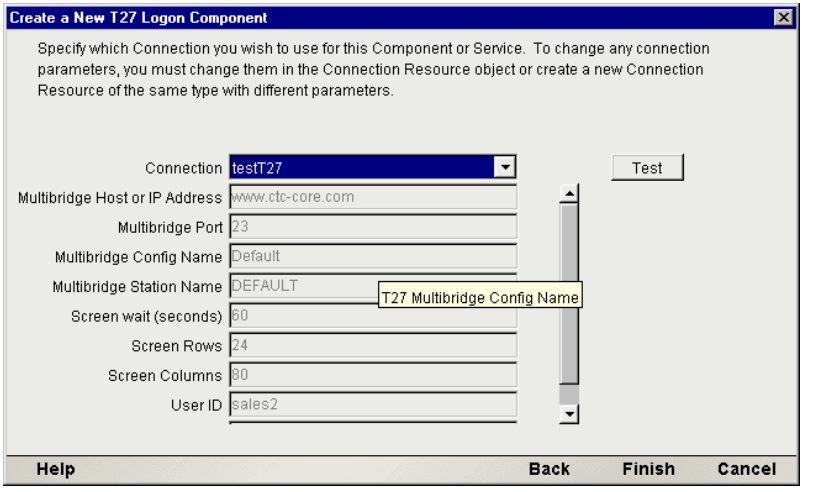
➢ T27 ログオンコンポーネントを作成する

1. Composer の［**File**］メニューから、［**New xObject**］、［**Component**］、［**T27 Logon**］の順に選択します。

   New xObject ウィザード［Header Info］パネルが表示されます。

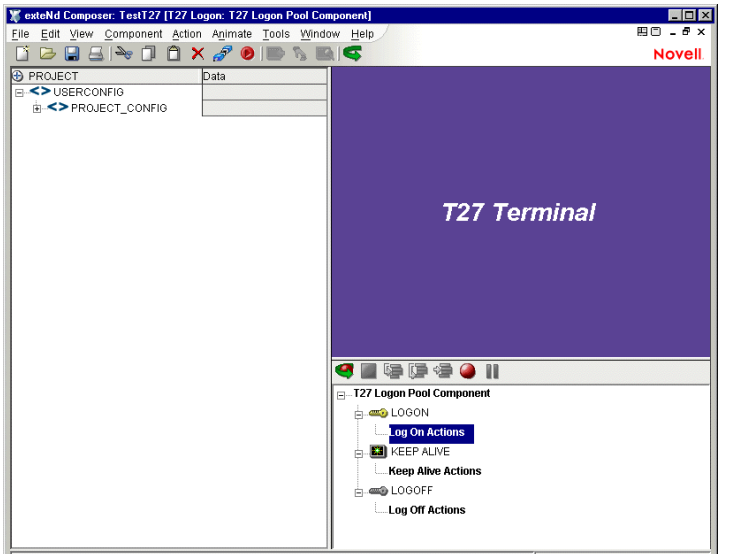


2. 接続オブジェクトの「名前」を入力します。
3. オプションとして、［**Description**］に説明テキストを入力します。
4. ［**Next**］をクリックすると、接続情報パネルが表示されます。



5 ［**Connection**］ドロップダウンリストから接続を選択します。

6 ［**Finish**］をクリックすると、ログオンコンポーネントエディタが表示されます。

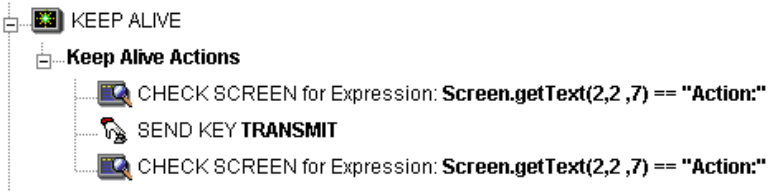


注記： アクションを記録するには、次の手順に従います。「LOGON」上にカーソルを合わせてから、記録をオンにします。完了後、記録をオフにします。「Keep Alive」にカーソルを合わせてから、記録をオンにします。完了後、記録をオフにします。「Logoff」にカーソルを合わせてから、記録をオンにします。完了後、記録をオフにします。

7 このガイドの第 5 章で説明されている記録方法と同じ方法でホストにログインして、起動画面に移動するために、Logon アクションを記録します。

8 このガイドの第 4 章の「T27 固有の Expression Builder 拡張」という節で説明されている特別な USERID 変数および PASSWORD 変数を代わりに使用するためにユーザ ID およびパスワードを入力する Logon Map アクションを編集します。

9 アクションモデルの KEEPALIVE セクションで、必要な Check Screen アクションおよび Send Key アクションを作成します (既存のアクションをコピーし、適切なアクションを選択して、貼り付けてから、必要に応じて修正する方法が一番簡単です)。



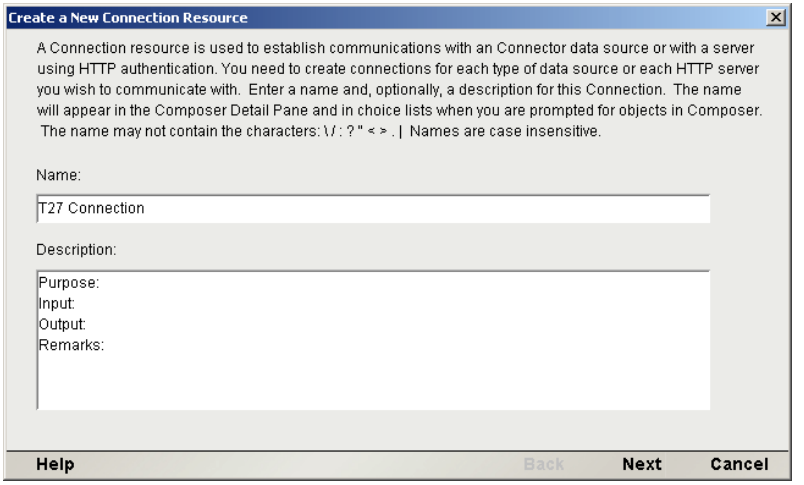
10 ホストを適切に終了するために、LOGOFF アクションを記録します。

11 ログオンコンポーネントを保存して、閉じます。

プール接続を使用したログオン接続の作成

➢ T27 ログオン接続を作成する

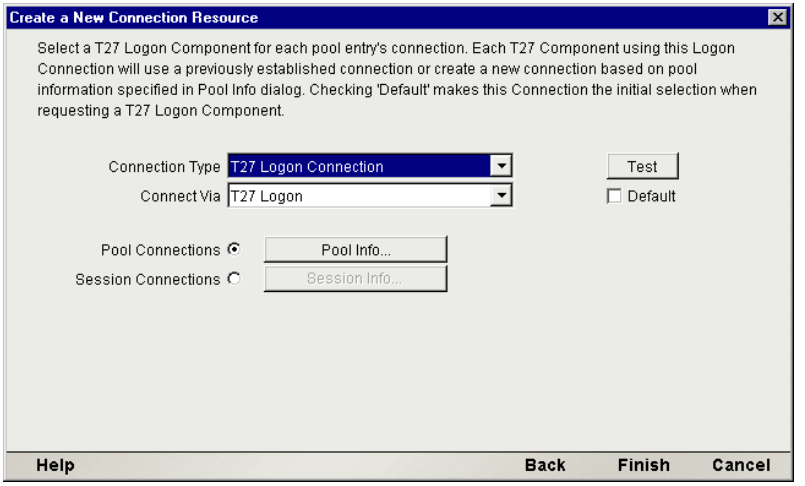
1 Composer の［File］メニューから、［**New xObject**］、［**Resource**］、［**Connection**］の順に選択するか、アイコンをクリックすることもできます。New xObject ウィザード［Header Info］パネルが表示されます。



2 接続オブジェクトの「名前」を入力します。

3 オプションとして、［**Description**］に説明テキストを入力します。

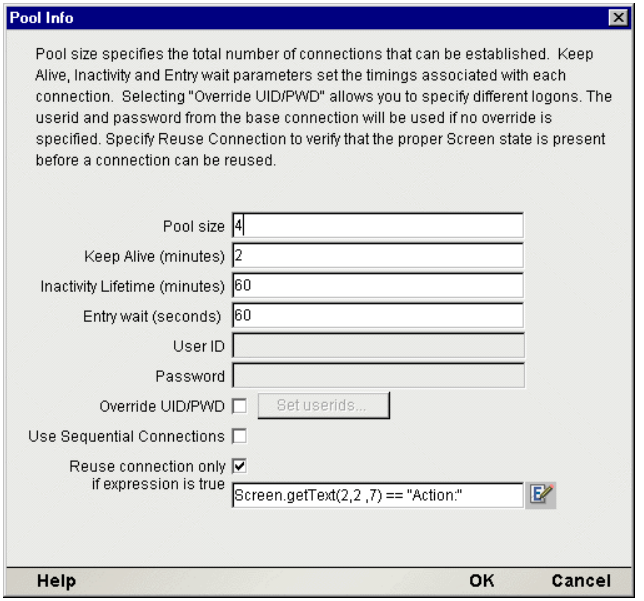
4 ［**Next**］をクリックすると、「接続情報」パネルが表示されます。



5 ［**Connection Type**］には、ドロップダウンリストから［T27 Logon Connection］を選択します。

6 ［**Logon Via**］コントロールで、作成したログオンコンポーネントを選択します。

7 ［**Pool Info**］ボタンをクリックすると、［Pool Info］ダイアログボックスが表示されます。



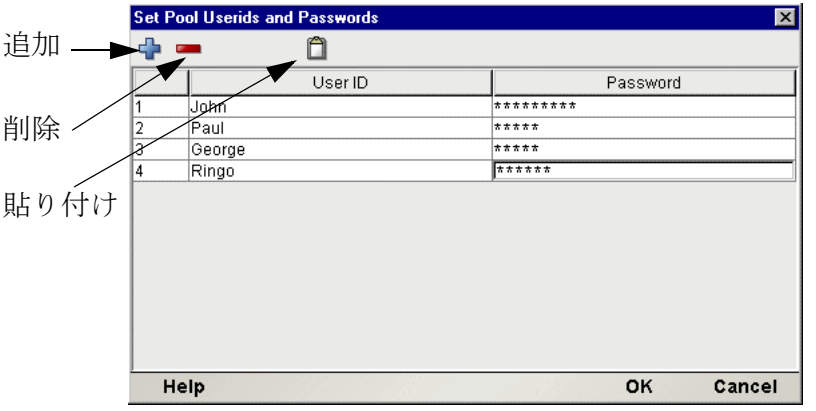
8 ［**Pool Size**］に数値を入力します。これは、このプールで使用できるようにする接続の総数を表します。各接続に対して、後にユーザ ID とパスワードの組み合わせを指定するように求められます。

9 ［**KeepAlive**］に間隔を入力します。ここで入力した数値 (分単位) は、アクティブな空き接続が存在する際に (つまり、T27 コンポーネントで使用されていない状態)、関連するログオンコンポーネントで Keep Alive アクションを実行する頻度を表します。ここで入力した数値は、アクティブでない接続に対してホストで定義された画面待機時間より小さくする必要があります。

10 ［**Inactivity Lifetime**］に値を入力します。この数値は、アクティブな空き接続を、切断してから接続プールでアクティブでない部分に戻すまでに、使用可能な状態で維持する時間を分単位で表します。接続がプールで非アクティブな状態に戻ってから再びアクティブになると、ログインしてホスト画面を移動する際にオーバーヘッドが発生することに注意してください。

11 ［**Entry Wait**］に時間を秒単位で入力します。この時間は、すべてのプールエントリがアクティブであり使用されている場合、T27 コンポーネントが空き接続に対して待機する時間を表します。この時間に達すると、アプリケーションサーバに例外がスローされます。

12 ［**Override UID/PWD**］をオンにすると、接続プールで使用するユーザ ID/ パスワードの組み合わせを指定できます。このチェックボックスがオンの場合、［Set Userids］ボタンがアクティブになります。ボタンをクリックすると、［Set USERIDs and PASSWORDS］ダイアログボックスが表示されます。



ツールバーには、空白の行を追加する［Add］、ハイライトした行を削除する［Delete］、およびスプレッドシートの情報をコピーして表に貼り付ける［Paste］の 3 つのアイコンがあります。詳細については、次の注記を参照してください。

注記： データをより迅速に入力する別の方法は、スプレッドシートからデータをコピーして、表に貼り付けることです。その場合、2 列が選択されていることを確認してください。最初の列にはユーザ ID が含まれており、2 番目の列にはパスワードが含まれている必要があります。スプレッドシートを開き、この 2 列と行を必要な数だけコピーします。表を開いたら、ツールバーにある［Paste］アイコンをただちに押します。同じ方法で、Microsoft Word® ドキュメントの表からデータをコピーすることもできます。

13 指定したプールのサイズになるまで **USERID/PASSWORD** の組み合わせの入力を繰り返し、［**OK**］をクリックします。プールサイズは、入力した行数に応じて調整されます。

14 ［**OK**］をクリックして［Set User IDs and Passwords］ダイアログボックスを閉じ、［Pool Info］ダイアログボックスに戻ります。

15 Composer で、［Set User IDs and Passwords］ダイアログボックスでユーザ ID が表示されている順番で接続を確立するには、オプションで［**Use Sequential Connections**］チェックボックスをオンにします。接続は、番号順に作成されます。

16 オプションで、［**Reuse connection only if expression is true**］コントロールをオンにできます。このコントロールを使用すると、起動画面のいくつかのテストに基づいて true または false を評価する ECMAScript 式を入力できます。この式の目的は、新しい T27 コンポーネントがアクティブな空の接続を使用しようとするたびに、起動画面が正しいものであることを確認することです。Composer サービスに関連していない状況では、起動画面がホストによって別の画面で置き換えられる可能性があります。たとえば、システム ABEND がホストに存在する場合、ログオンコンポーネントの起動画面はシステムメッセージ画面によって置き換えられることがあります。

注記： この式の作成方法の詳細については、このガイドの第 5 章の「エラーおよびメッセージの処理」の説明を参照してください。また、「T27 ログオン接続のパフォーマンスの最大化」も参照してください。

特定の画面が存在するかどうかを確認する場合に使用するサンプルカスタムスクリプトは、次のとおりです。画面が存在しない場合、スクリプトにより、不正な画面のためログオン接続が解放されたことを示すメッセージがコンソールに記述されます。この関数は、［Pool Info］ダイアログボックスの［Reuse connect only if expression is true］コントロールから呼び出されます。

```
function checkValidLaunchScreen(ScreenDoc)
{
    var screenText = ScreenDoc.XPath("SCREEN").item(0).text
    if((screenText.indexOf( "MENU") != -1 || screenText.indexOf("APLS") != -1) &&
      (screenText.indexOf("COMMAND UNRECOGNIZED") == -1 ||
        screenText.indexOf("UNSUPPORTED FUNCTION") == -1))
    {
       return true;
    }
    else
    {
        java.lang.System.out.println("Warning - Releasing logon connection at bad screen");
        java.lang.System.err.println("Warning - Releasing logon connection at bad screen");
       return false;
    }
}
```

17 ［**OK**］をクリックして、接続情報パネルに戻ります。

18 ［**Finish**］をクリックすると、ログオン接続が保存されます。

T27 ログオン接続のパフォーマンスの最大化

T27 コンポーネントが前の T27 コンポーネントによって無効な画面に残された可能性のある接続で実行を開始しないようにするために、ログオン接続リソースでは、接続自体で起動画面が存在することを確認できます。この操作を実行するには、ログオン接続の［Pool Info］ダイアログボックスで、［Reuse connection only if expression is true］オプションを使用します。ここで画面テストを指定すると、T27 コンポーネントの実行が完了するごとに画面の確認が行われます。テストに失敗した場合、exteNd Composer はただちにホストから切断し、その時点まで使用していたユーザ ID はホストに残されます。先に説明したように、最終的にホストでこのユーザが停止されますが、ユーザが停止される前にもう一度ユーザ ID にアクセスした場合は、ユーザ ID がプールから破棄される可能性があります。したがって、プールサイズが小さくなり、その結果全体的なパフォーマンスも低減します。

［Reuse connection only if true］オプションを使用する別の理由は、このオプションでは画面に対して詳細なテストを実行して、正しい起動画面であるかどうかを確認できるためです。Map Screen アクションでは、画面の確認を実行しますが、端末データストリーム内のフィールド数のみが確認されます。大抵の場合は、これで十分ですが、異なる 2 つの画面で同じフィールド数になる可能性もあります。この場合、式に基づき、画面の内容を検査するテストを使用すると、さらに厳密な結果が得られます。常にこの機能を使用することをお勧めします。

動的に作成したドキュメント / 要素と静的に作成したドキュメント / 要素の違い

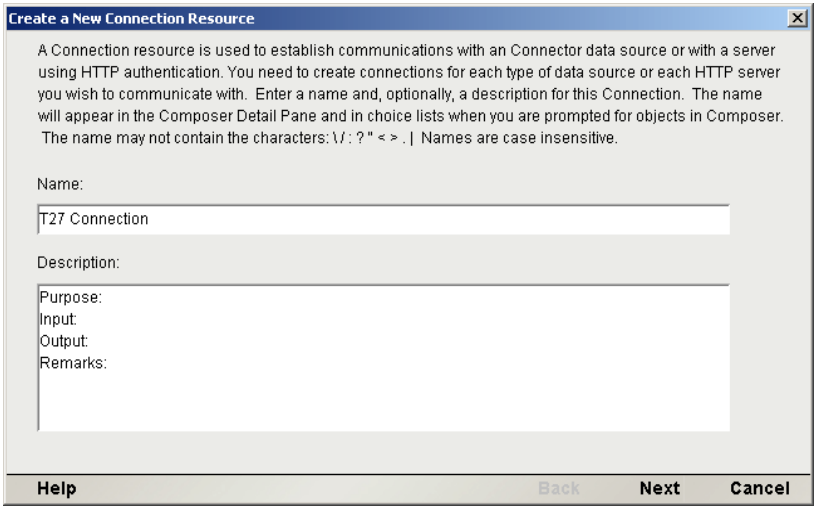
一部の Composer アプリケーションでは、ユーザが各種のコントロール、監査、またはメタデータ、あるいはこれらすべてを XML ドキュメントに配置しなければならない場合があります。このドキュメントは、処理中の (つまり、情報元から作成された) 実際のエレメント / ドキュメントの追加になる場合もあれば、そうではない場合もあります。このドキュメントの構造およびデータが複数 (100 以上) の Map アクションにより、動的に作成された場合、コンポーネントのパフォーマンスおよびサービス全体に障害が発生する可能性があります。パフォーマンスを向上するには、事前に動的な内容を作成しないで、ドキュメント構造の一部を作成します。その後、そのドキュメント構造を XML Interchange アクションを使用してサービスにロードし、動的な内容に対して Map アクションを保持します。この結果、パフォーマンスが最大 30% まで向上される場合があります。

セッション接続を使用したログオン接続の作成

場合によっては、プールを使用することなく、ログオン接続によって許容されるセッションパラメータに対して、特別な制御レベルを必要とする可能性があります。この場合、次に説明されている手順に従います。

➢ T27 ログオン接続を作成する

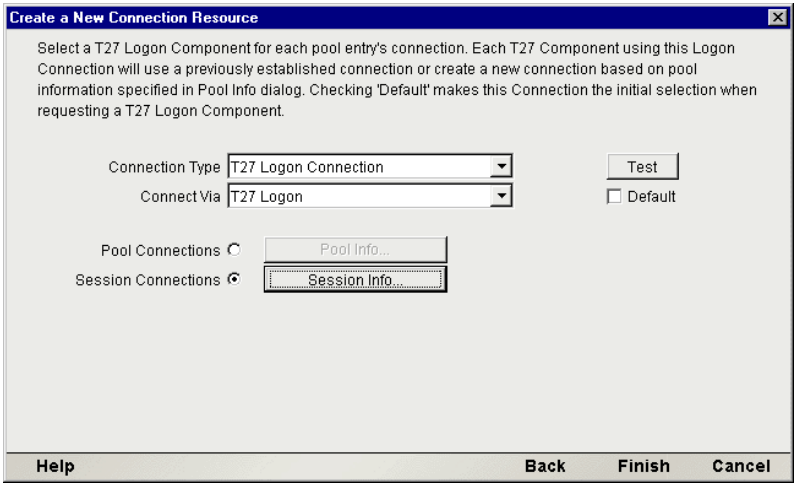
1 Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、［**Connection**］の順に選択するか、アイコンをクリックすることもできます。New xObject ウィザード［Header Info］パネルが表示されます。



2 接続オブジェクトの「名前」を入力します。

3 オプションとして、［**Description**］に説明テキストを入力します。

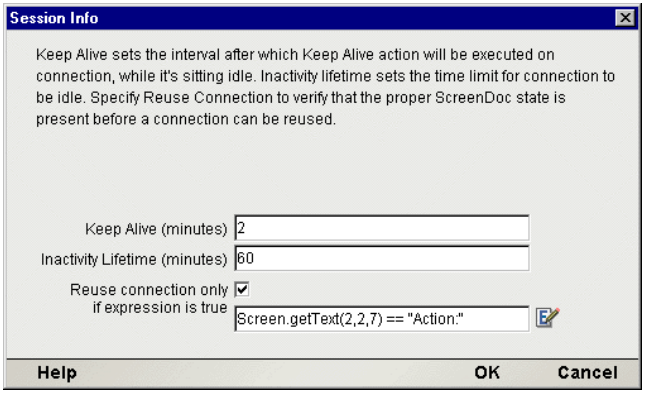
4 ［**Next**］をクリックすると、［**Connection Info**］パネルが表示されます。



5 ［Connection Type］には、ドロップダウンリストから［T27 Logon Connection］を選択します。

6 ［**Connect Via**］コントロールで、作成したログオンコンポーネントを選択します。

7 ［**Session Connections**］ラジオボタン、［Session Info］ボタンの順にクリックします。



8 ［Keep Alive (minutes)］の数値は、アクティブな空き接続が存在する際に (つまり、T27 端末コンポーネントで使用されていない状態)、関連するログオンコンポーネントで Keep Alive アクションを実行する頻度を (分単位で) 表したものです。ここで入力した数値は、アクティブでない接続に対してホストで定義されたタイムアウト時間より小さくする必要があります。

9 ［Inactivity Lifetime (minutes)］の数値は、接続を切断してから接続プールのアクティブでない部分に戻すまでに、アクティブな空き接続を使用可能な状態で維持する時間を (分単位で) 表します。接続がプールで非アクティブな状態に戻ってから再びアクティブになると、ログインしてホスト画面を移動する際にオーバーヘッドが発生することにご注意ください。

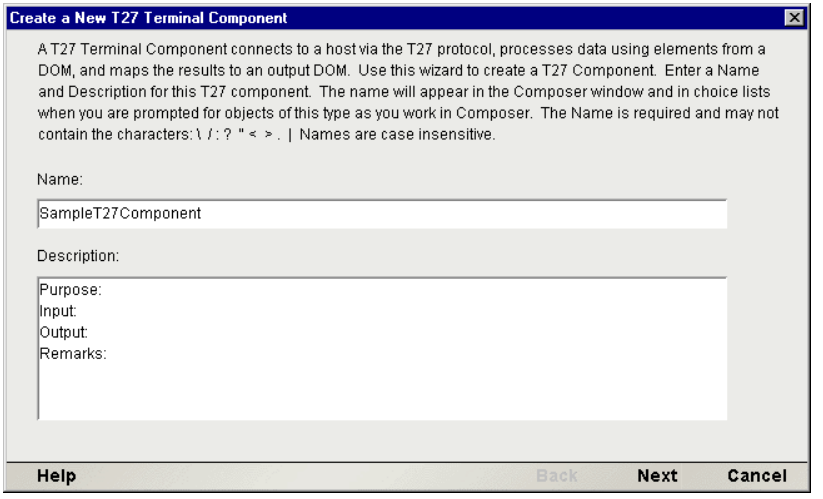
10 ［Reuse connection only if expression is true］をオンにする場合は、チェックボックスをクリックします。［Expression］フィールドが自動的に開き、式アイコンをクリックすると［Reuse connection only if the expression is true］ダイアログボックスを表示できます。

プールされた接続を使用する T27 コンポーネントの作成

この時点で、接続プールを使用する T27 コンポーネントを作成する準備が整いました。ほとんどの場合、通常の T27 コンポーネントの場合と同じようにコンポーネントを作成でき、唯一の違いは、New Component ウィザードの接続パネルで指定する接続です (通常の T27 接続の代わりにログオン接続を指定します)。

➢ T27 コンポーネントを作成する

1 Composer の［File］メニューから、［**New xObject**］、［**Component**］、［**T27**］の順に選択します。New xObject ウィザード［Header Info］パネルが表示されます。



2 コンポーネントの「名前」を入力します。

3 オプションとして、[**Description**] に説明テキストを入力します。

4 [**Next**] をクリックすると、XML プロパティ情報パネルが表示されます。

5 [**Input and Output Templates**] で、コンポーネントに対して必要な入力テンプレートおよび出力テンプレートを選択します。

6 [**Next**] をクリックすると、接続情報パネルが表示されます。

7 作成したログオン接続を選択して、[**Next**] をクリックします。コンポーネントエディタが表示されます。

8 25 ページ「新しい T27 コンポーネントを作成する」の説明に従って、コンポーネントを作成します。

T27 端末コンポーネントのパフォーマンスの最大化

ログオンコンポーネントの Logon アクションによって起動画面に達すると、この起動画面は接続を使用する T27 端末コンポーネントに渡されます。その後、(実行終了時に) T27 端末コンポーネントで画面ハンドラが起動画面に戻されます。T27 コンポーネントが起動画面に達せずに終了した場合 (つまり、不正な画面で接続をプールに解放した場合)、その接続を使用するその後の T27 コンポーネントすべてで例外が発生し、接続が使用できなくなる可能性があります。また、全体的なパフォーマンスが低下したり、処理中のコンポーネント内でデータの整合性に問題が生じる可能性もあります。

このためにも、起動画面が存在し、起動画面を確認する Check Screen アクションが T27 コンポーネントで実行する最後のアクションになるようにします。コンポーネントで、独立してコンポーネントの実行を終了する設定のパスが多数あるような場合は注意が必要です。そのような場合は、パスがそれぞれ Check Screen アクションで終了するようにしてください。

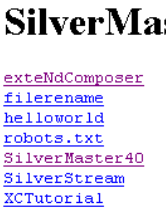
プールの管理

exteNd Composer コンソールの使用

T27 接続プールは、T27 コンソール画面を使用して管理できます。

➢ コンソールへのアクセス方法

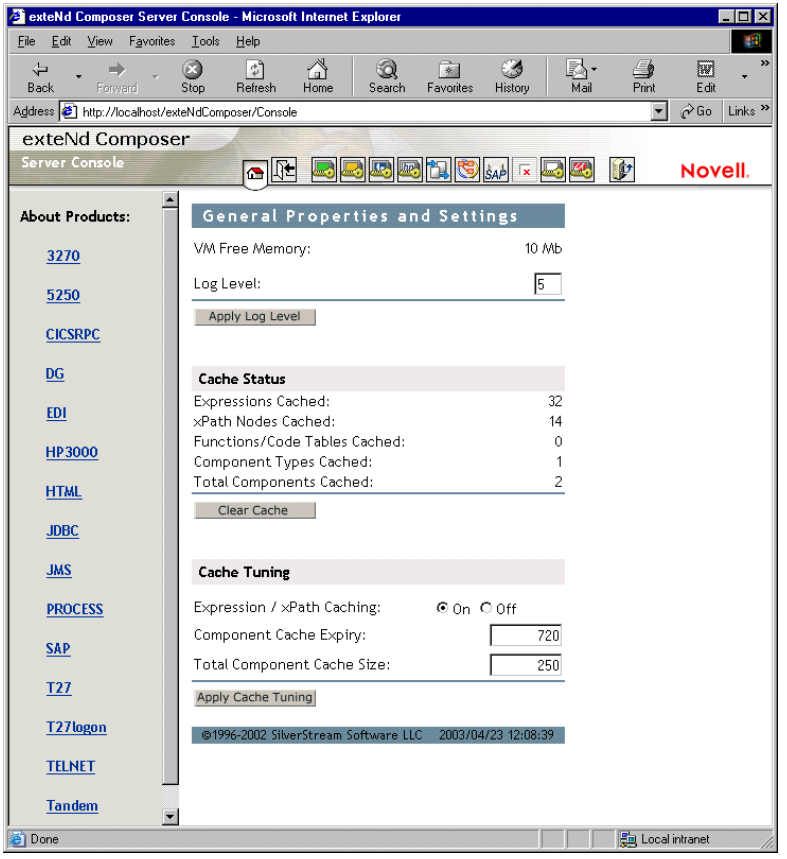
1 Novell exteNd Application Server を使用している場合、**http://localhost/SilverMaster40** (または使用しているバージョンに適切なパス) を使用して Web ブラウザ経由でサーバにログオンします。この例では、Novell exteNd App Server 4.0 が使用されています。



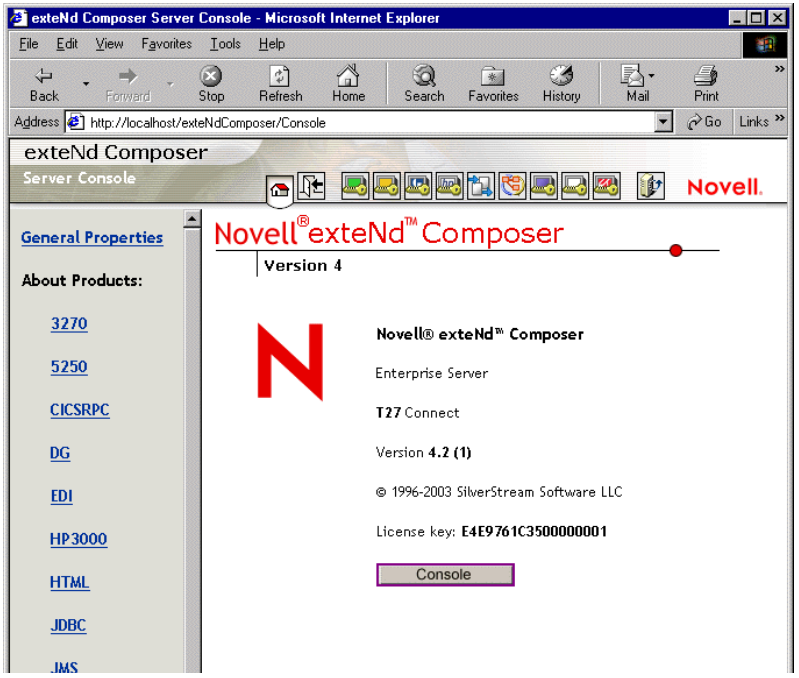
注記： exteNd App Server を使用していない場合、次の形式の URL を入力します。

http://<hostname>:<port>/exteNdComposer/Console

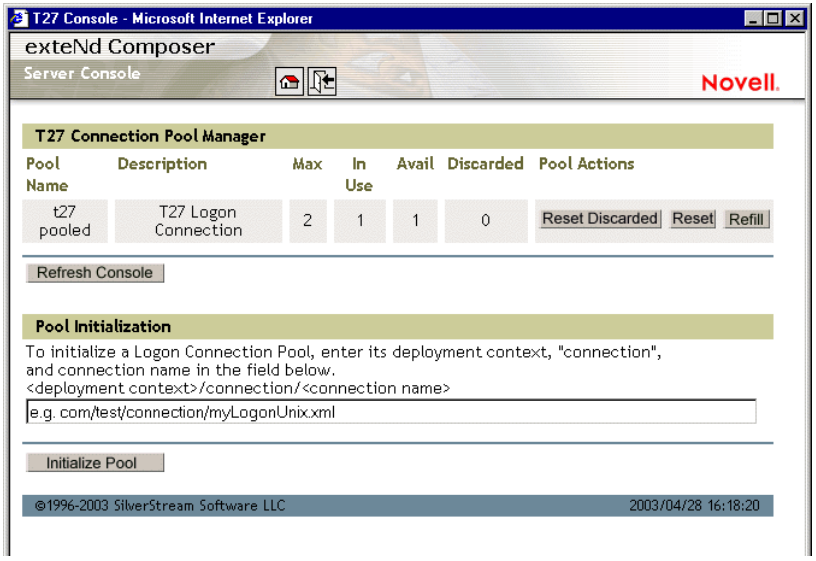
2 ［**exteNd Composer**］リンクをクリックします。メインコンソールページが表示されます。



3 左の(ナビゲーション)フレームのT27をクリックすると、[T27 Console General Properties]画面が表示されます。



4 ［Console］ボタンをクリックします。ブラウザポップアップウィンドウ (T27 接続プール管理画面) が表示されます。



5 ログオン接続プールを初期化するには、画面下部のテキストフィールドに配備コンテキスト、単語「connection」、および実際の接続名を入力します (前の図を参照してください)。次に、[**Initialize Pool**] ボタンをクリックします。

注記： 詳細については、該当する Composer Enterprise Server のガイドを参照してください。

6 オプションで [**Refresh Console**] ボタンをクリックして、ビューを更新します。

接続プール管理および配備済みのサービス

接続プール管理画面には、T27 Connect を使用した接続の現在の状態が表示されます。画面には、プール名、接続の説明、プールでの最大接続数、使用中の接続数、使用可能な接続数、および破棄された接続数が表に一覧表示されます。また、接続プールに関連するさまざまなアクションを実行できるいくつかのボタンがあります。これらのボタンについては、下の表で説明しています。

表 1-2:

ボタン名	アクション
Reset Discarded	破棄された接続をリセットします。この内容はその後テーブルに反映されます。
Reset (Pool)	使用可能な接続および破棄された接続をリセットします。この内容はその後テーブルに反映されます。
Refill (Pool)	プールを最大接続数まで再読み込みします。
T27 Connection Pool Manager コンソールのその他のボタン	
Refresh Console	接続プールの現在のステータスを表示します。
Initialize Pool	配備された lib ディレクトリへの相対パスを入力して、ログオン接続プールを初期化します。この操作は、配備された jar を抽出しないと動作しません。終了後、[SUBMIT] ボタンをクリックします。

接続破棄の動作

接続プールにおけるパフォーマンス上の利点は、一度に複数のユーザが特定のリソースやリソースセットにアクセスできることです。接続の確立は、表からユーザ ID およびパスワードを選択するログオンコンポーネントで開始されます。接続に失敗すると、このユーザ ID とパスワードに対して接続が破棄され、接続を確立するまで別の接続を確立しようとします。接続に失敗しても、正常な接続の確立が妨げられるわけではありません。

T27 Connect では、不正なユーザ ID、タイムアウトしたパスワードなど理由を問わず、確立できなかった接続をすべて破棄し、他の接続を再使用することによって、確立できなかった接続以外に同じ問題が発生しないようにします。接続が使用不可能と判断された場合、T27 用 Connect で、「Logon connection in pool <Pool name> was discarded for User ID <User ID>」というメッセージがシステムログに作成されます。

画面の同期

画面を同期すると、プールのユーザに特別な影響が生じます。ユーザが元の状態に戻る画面を使用せずに接続から離れた場合、次のユーザは、予期しない状態の画面でセッションを開始することになるため、エラーが発生します。このような状況にならないために、ユーザが接続プールで指定できる画面式が用意されています。T27 コンポーネントでは、Send Key アクションが最後のアクションとして実行され、正しいログオン画面でセッションを終了することが重要です。

注記： 接続を切断する前に、T27 端末コンポーネントが起動画面が表示されるまで待機するように、最後のアクションは空の Check Screen アクションでなければなりません (Send Key アクションを作成すると自動的にこの処理は行われますが、この場合でも最後のアクションは Check Screen である必要があります)。

ラインタイム時に、ユーザセッションの終了時に不正な画面があるかどうか確認するには、次に示されている関数と類似した関数を実行するコンポーネントのアクションモデルの最後に、Function アクションを含めます。

```
if((Screen.getText(1,11,4)== "MARC" || Screen.getText(2,2,7) ==
"Action:" ) &&

(Screen.getTextFromRectangle(1,1,24,80).indexOf("COMMAND
UNRECOGNIZED") == -1 ||
Screen.getTextFromRectangle(1,1,24,80).indexOf("UNSUPPORTED
FUNCTION") == -1))
          {
              java.lang.System.out.println("OK to exit");
          }
         // Otherwise, write error messages to Sys.out
      else
        {
              java.lang.System.out.println("Warning - Releasing
logon connection at bad screen");
        }
```

この特別な例では、この関数によって画面テキストの MARC ヘッダまたは Action: フィールドが確認され、また「COMMAND UNRECOGNIZED」または「UNSUPPORTED FUNCTION」の単語が見つからないことも確認されます。結果によっては、ログにエラーが記述されます。

A 用語集

ANSI

American National Standards Institute。

Check Screen

コンポーネントに対して、画面が特定の状態になるまで処理が実行されてはならないという信号を送るアクション。ユーザ指定のタイムアウト値により異なります。

ECMAScript

European Computer Manufacturers Association の標準 No. 262 に準拠した、Java スクリプトに類似した言語。

Screen オブジェクト

現在の T27 の画面表示を表します。

Send Key

T27 に固有のアテンションキーまたはファンクションキーを押すことを示すアクション。

Set Screen Text

画面または画面に入力されたキーへのマップがある場合に、アクションモデルに表示されるアクション。

T27

Burroughs Corporation により最初に開発され、後に Unisys により購入された端末。A Series、V Series、ClearPath™ NX などのメインフレームコンピュータと通信するために使用されます。

Unisys

コンピュータベースの情報システムおよびそれに関連する製品やサービスの設計者、製造業者、および販売業者。T27 メインフレーム端末は、1986 年に Unisys の一部となった Burroughs Corporation により最初に開発されました。A Series、V Series、および ClearPath™ NX などのメインフレームのコンピュータモデルで T27 端末エミュレーションが実行されます。

接続プール

中間プロセス(アプリケーションサーバ自体、またはサーバに関連付けられていないメモリ常駐のバックグラウンドプロセス) で、事前に確立、認証された接続を維持して、クライアントアプリケーションまたはエンドユーザの間の接続の「分配」を監視する手段。

ダム端末

より強力なホストコンピュータと通信するために必要な最低限の CPU、メモリ、または記憶デバイス以上の能力を持たないコンピュータ端末。

端末エミュレーション

パーソナルコンピュータで、T27 といった (特定のブランドの) 端末のような動作を可能にするプログラム。その結果、コンピュータは端末としてメインフレーム (ホスト) コンピュータに表示され、カーソル移動、画面のクリアといった機能に対して同じエスケープシーケンスおよび他のアテンションキーが受け入れられます。

ネイティブ環境ペイン

実際の T27 端末セッションのエミュレーションを表示する、T27 コンポーネントエディタ内のペイン。

B T27 の表示属性

`Screen.getAttribute()` メソッドでは、次の表に示されている値のうちの 1 つが返されます。これは画面の特定の場所で表示されている文字に関する現在の属性の状態を表します。下に表示されている属性は単に最も一般的な属性ということだけであり、下に説明されている内容の任意の組み合わせが発生する可能性は理論的に十分考えられます。基本的に、underlined、bold、blinking、reverse の文字は標準の整数を返します。これはその後、フィールドが secure、protected、selected、または vertical、あるいはこれらすべてであるかを示す 16 進数の数字に加算されます。

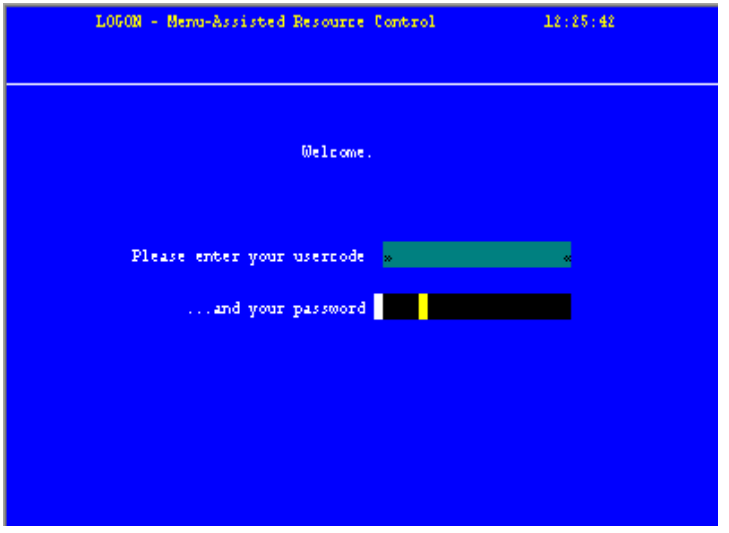
番号	属性
0	standard (入力可 - 例 : エントリフィールド)
16 (0X10)	secure (入力可 - 例 : パスワード)
32 (0X20)	protected (入力不可)
33 (0X20)+1	protected および underlined
34 (0X20)+2	protected および bold
36 (0X20)+4	protected および blinking
40 (0X20)+8	protected および reverse
48 (0X10)+(0X20)	secure および protected
64 (0X40)	selected
80 (0X40) + (0X10)	selected および secure
96 (0X20)+(0X40)	selected および protected
98 (0X20)+(0X40)+2	selected、protected、および bold
0X100	vertical

すべての文字属性を一度に表示

Screen.getAttribute() メソッドを使用すると、すべての属性 (画面のすべての場所) を一度にキャプチャする関数を簡単に作成できます。たとえば、次の ECMAScript 関数は、設計時に画面の属性を警告ダイアログボックスに表示するために使用できます。

```
function showAttributes( myScreen )
{
     var attribs = new String(); // create empty string
     // Iterate over all rows and columns:
     for (var i = 1; i <= myScreen.getRows(); i++, attribs += "\n" )
          for (var k = 1; k <= myScreen.getCols(); k++)
               attribs += " " + myScreen.getAttribute(i,k);
}
// display the results:
alert(showAttributes( Screen ));
```

次の図は、T27 の画面を示しています。



下の図は、showAttributes() 関数を画面に適用した結果を示しています (右 / 左の余白がページの境界を越えたため、図は途中で切れています)。

```
34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 48 48 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 48
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 33
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
```

C 予約語

次の用語は、T27 Connect 用 exteNd Composer の予約語であるため、ユーザ作成の変数、メソッド、またはオブジェクトのラベルとして使用することはできません。

- USERID
- PASSWORD
- PROJECT
- Screen
- getAttribute
- getCols
- getCursorCol
- getCursorRow
- getNextMessage
- getPrompt
- getRows
- getStatusLine
- getText
- getTextFromRectangle
- hasMoreMessages
- putString
- putStringInField
- setMessageCaptureOff
- setMessageCaptureOn
- typeKeys

索引

T

U

あ

え

お

か

き

こ

し

へ

ま

ゆ

よ

ろ