Novell exteNd Composer™ Telnet 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

Telnet Connect ユーザガイド
2003 年 1 月
000-000000-000

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、Telnet コンポーネントエディタと呼ばれる exteNd Composer Telnet Connect の使用方法について説明します。Telnet 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、Telnet アプリケーションを統合する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オプションに精通しており、これらを使用したことがあるという前提で説明していきます。また、Telnet 環境、および Telnet または VT シリーズの端末 (VT100 など) 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。

追加のマニュアル

Novell exteNd Composer の完全なマニュアルのセットは、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 Telnet へようこそ」では、Telnet 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ます。

第 2 章「Telnet コンポーネントエディタをお使いになる前に」では、Telnet 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「Telnet 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「Telnet アクションの実行」では、基本的な Telnet アクションの使用方法、および Telnet Connect におけるドラッグアンドドロップの固有な規則について説明します。

第 5 章「高度な Telnet アクション」では、アクションモデルでの Telnet の一般的なコンピューティング問題の解決方法について説明します。

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、Telnet Connect により認識および使用される「ANSI エスケープシーケンスおよびコントロールコード」について説明します。

付録 C では、「Telnet 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 D では、「予約語」について説明します。Telnet 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」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 Telnet User Interface へようこそ

はじめに

『Telnet 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 Telnet 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 Telnet 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、Telnet 環境と、XML に対応させる特定のアプリケーションに慣れ親しんでおく必要があります。

exteNd Composer Connect について

exteNd Composer は、単純なハブと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 ドキュメントを使用して要求を受け付け、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。

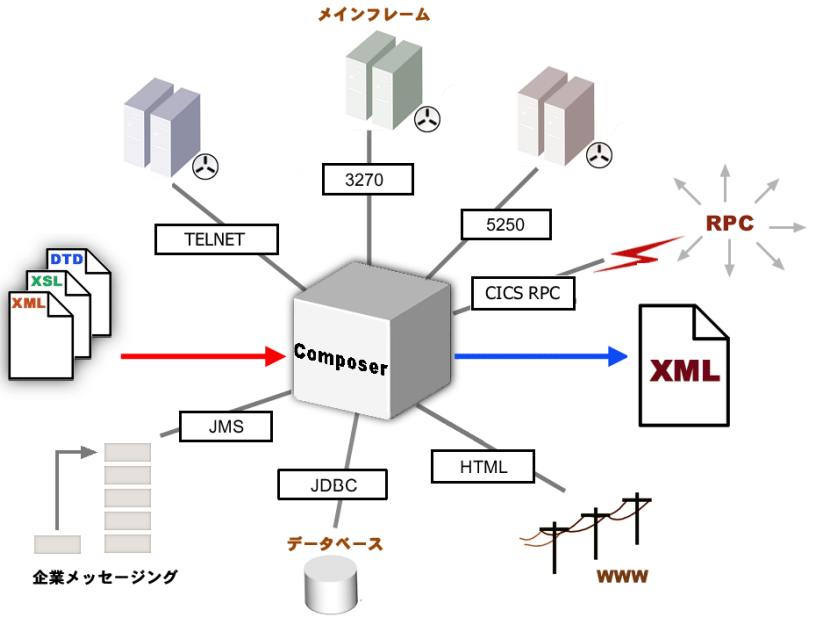
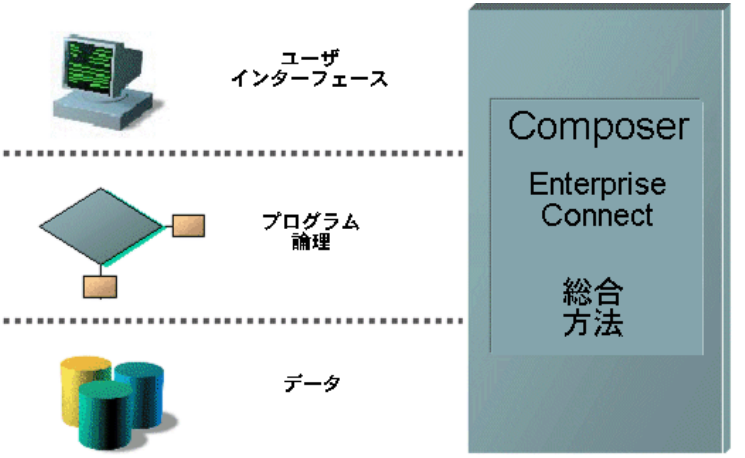


図 1-1

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 (次を参照)。



Telnet とは

Telnet とは、通信プロトコルに対する仕様 (RFC 854) です。Telnet という用語は、ANSI 標準システム上で端末をエミュレートするための一般 TCP/IP プロトコルを指します。UNIX や VAX/VMS (およびその他) に対するアプリケーションの多くは、端末ベースのシステム用に開発されています。これらのシステムでは、Telnet TCP/IP プロトコルを通じてインタフェースをリモートで実行できます。Telnet では、画面をクライアントに送信し、キーデータをそのクライアントから受け付ける端末を模倣することによって、これが実現されます。いわゆる「ダム」端末を通じて行われるこの操作は、すべてのデータがホストコンピュータで処理されることを意味します。Telnet 端末エミュレーションソフトウェアは、ホストコンピュータとの通信中にマイクロコンピュータまたは PC を Telnet タイプの端末であるかのように動作させるために使用できます。

Telnet Connect とは

Telnet 端末ストリームにフッキングすることによって User Interface 統合方法を使用する、Telnet Connect XML に対応した ANSI 端末ベースのシステム。Telnet 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プロセス、エクストラネットプロセス、またはイントラネットプロセスに対して利用可能にすることができます。端末セッションでの場合と同様にアプリケーションを移動したり、XML ドキュメントを使用して問い合わせや更新をキーではなく画面に追い込んだり、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での場合と同様に同じ決断を下したり、要求者に返すことのできる XML ドキュメントにデータや応答を移動するか処理を続行することができます。Telnet 画面は、Telnet 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 Telnet コンポーネントについて

XML Map コンポーネントと同様に、Telnet 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、ホストアプリケーションに (Telnet を通じて) 接続して、画面からの要素を使用してデータを処理し、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。

Telnet 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、Telnet プログラムの「スクリーンスクレーピング」を実行して XML ドキュメントに取得したデータを配置することができます。Telnet 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。

Telnet User Interface コンポーネントエディタを使用して作成できるアプリケーション

Telnet User Interface コンポーネントエディタを使用すると、作成している XML 統合を拡張して、Telnet ベースの端末操作をサポートするビジネスアプリケーションをどれでも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。Telnet 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、個別の情報ソースの情報をマージしてから、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情報が提供されます。

2 Telnet コンポーネントエディタをお使いになる前に

Telnet 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 Telnet 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。

- プログラム用に XML テンプレートを作成する。
- 接続リソースを作成する。
- Telnet コンポーネントを作成する。
- 記録モードを入力し、コンポーネントエディタのネイティブ環境画面で使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、Telnet コンポーネントを使用可能にするために不可欠な最初の手順である、Telnet 接続リソースの作成と設定に焦点を当てます。

Telnet 接続リソースの作成

Telnet コンポーネントを作成する前に、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必要があります。利用可能な接続リソースなしで Telnet 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が尋ねられます。この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。

接続リソースについて

Telnet コンポーネント用に接続リソースを作成すると、選択したホスト環境にライブ Telnet 接続を使用して接続します。接続リソースは、設定後、該当するホストへの接続が必要となり得る Telnet コンポーネントで利用可能になります。

定数駆動型および式駆動型の接続について

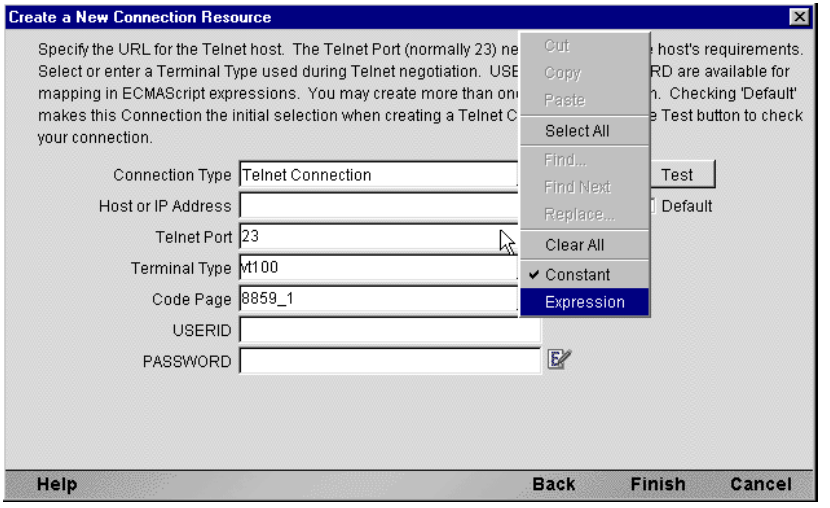
接続パラメータの値は、定数または式としての方法のうちいずれかを使用して指定できます。「定数ベース」のパラメータでは、接続が使用されるたびに［Connection］ダイアログボックスで指定する静的な値を使用します。「式ベース」のパラメータでは、ランタイム時に接続が使用されるたびに「異なる」値となりえるプログラム的な式 (つまり、ECMAScript 式) を使用して該当する値を設定できます。これにより、接続の動作は柔軟になり、ランタイム条件に基づいて変化することができるようになります。

たとえば、Telnet 接続における式駆動型のパラメータの非常に単純な使用の 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に、Deployment Wizard で PROJECT 変数を最終配備環境に適切な値に更新できます。それとは正反対に、アプリケーションサーバで Java ビジネスオブジェクトを照会するカスタムスクリプトを使って、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。

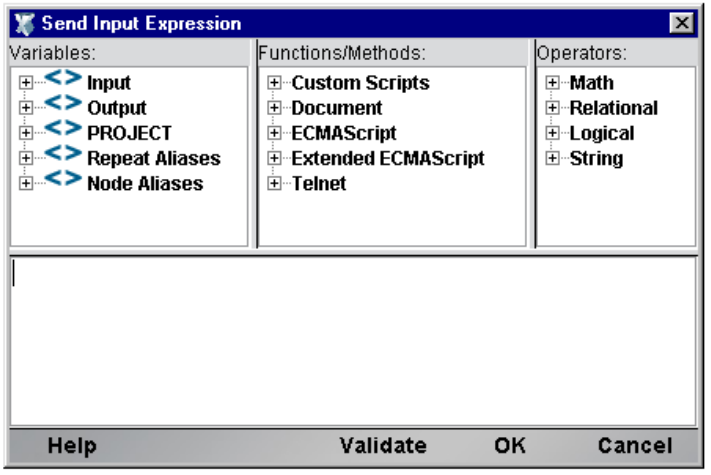
➢ 定数駆動型から式駆動型にパラメータを切り替える

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**］を選択すると、エディタボタンが表示されるか、または有効になります (次を参照)。



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



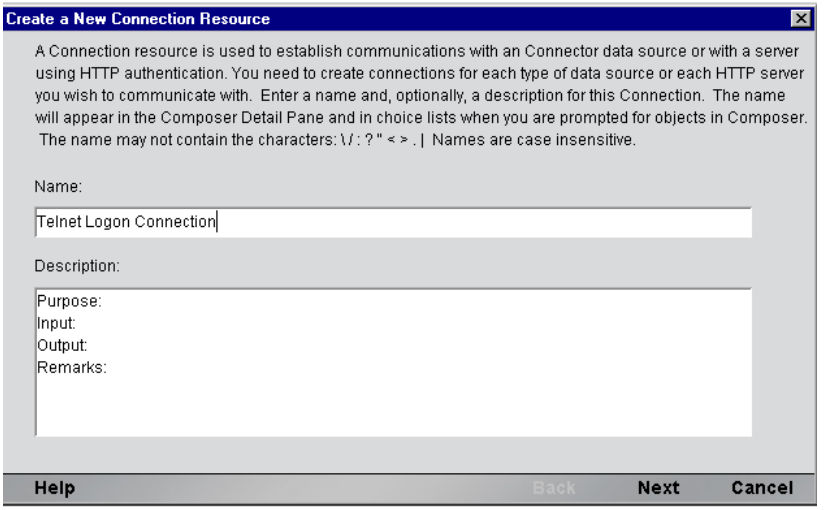
4 ランタイム時に有効なパラメータの値に評価する式を作成します (または、ウィンドウの上部にある選択リストを使用します)。[**OK**]をクリックします。

➢ Telnet 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で［**Connection**］を選択し、マウスを右クリックした後で［**New**］を選択することもできます。

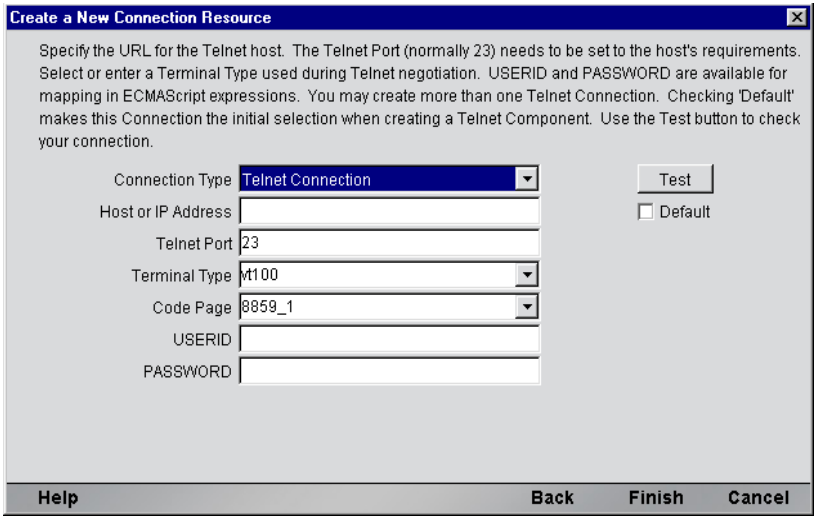
Create a New Connection Resource ウィザードが表示されます。



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 接続タイプのプルダウンメニューから［**Telnet Connection**］を選択します。ダイアログボックスの外観が変わり、Telnet 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別名を入力します。

7 ［**Port**］フィールドに、Telnet 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 ［**Terminal Type**］フィールドで、ホストとのハンドシェーク時に指定する端末のタイプを入力します。プルダウンメニューの値 (現在は、VT100、VT220、または VT320) を 1 つ選択するか、別の端末タイプを手動で入力します。手動で値を入力する際には、小文字で「vt」(vt132 など) と使用します。

注記： 一部のホストでは、「VT220」(およびその他すべて) としてログオンできない場合があります。ホストで認識される端末 (複数可) のタイプが分かっている場合、この欄に許容値を入力して、ホストが正常にハンドシェークできるよう「スプーフ」することができます。

9 ［**Code Page**］フィールドで、コードページを指定します (23 ページ「コードページサポートについて」を参照)。

10 「**ユーザ ID**」および「**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)。マウスを右クリックし、これらのフィールドを式駆動型にする場合は［Expression］を選択します (以前のディスカッションを参照)。

注記 : ユーザ ID とパスワードの情報をこのダイアログボックスに入力すると、ECMAScript グローバル変数の USERID と PASSWORD はこれらの値を指すようになります。その後、これらのグローバルを Send Buffer 式で (または、33 ページ「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。

11 この特定の Telnet 接続を後続の Telnet 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。

12 ［**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。

コードページサポートについて

exteNd Composer 接続リソースのコードページサポートでは、exteNd Composer とその他のホストシステム間で送信された文字を変換する際に使用する文字エンコードスキームを指定できます。exteNd Composer データでは、Unicode 文字エンコード (Java および XML 標準) が使用されます。既存のレガシーホストシステムとその他のホストシステムでは、言語または使用に特定のさまざまな文字エンコードスキーム (つまり、コードページ) を使用します。互いに通信する場合は、これらのシステム間で文字エンコードを変換するためのメカニズムが必要となります。これは、接続リソースのホストシステムで使用されるコードページを指定することによって、exteNd Composer で処理されます。

コンポーネント用の XML テンプレートの作成

接続リソースのほかに、Telnet コンポーネントでは、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持つよう、XML テンプレートをすでに作成していることが必要とされる場合もあります (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むよう設計されます。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よう設計されます。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、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、Telnet セッションで「ダム端末」からホストシステムを通常は照会します。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、Telnet 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。(HTTP を介して着信する) 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されることがあります。ホストから返されたルックアップデータは、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： コンポーネント設計によって別の xObject リソース (カスタムスクリプトやコードテーブルマップなど) が要求される場合は、Telnet コンポーネントを作成する前にこれらのリソースを作成することが推奨されます。詳細については、『exteNd Composer ユーザガイド』を参照してください。

3 Telnet コンポーネントの作成

Telnet コンポーネントを作成する前に

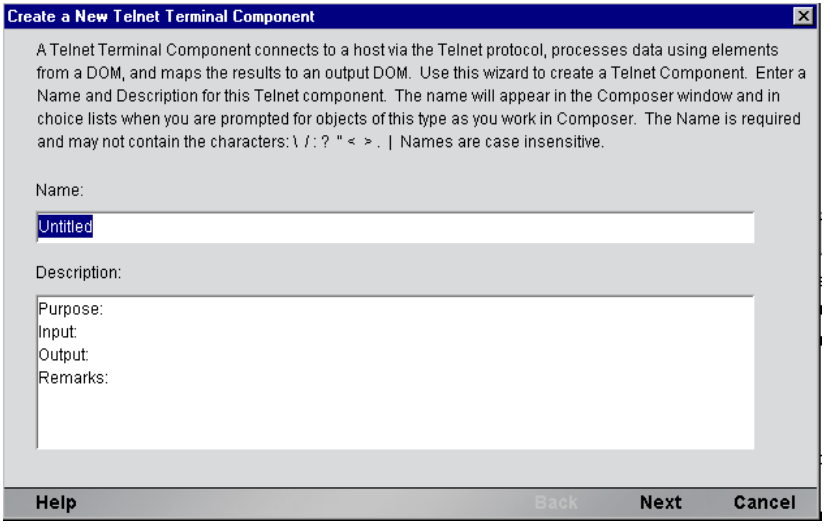
すべての exteNd Composer コンポーネントと同様に、接続リソースが使用可能であることを前提とした場合、Telnet コンポーネントを作成する最初の手順は、コンポーネントで必要な XML テンプレートを準備することです (詳細については、『Composer ユーザガイド』の「新しい XML テンプレートの作成」を参照してください)。

XML テンプレートを指定すると、コンポーネントによって処理される入力および出力を表すテンプレートのサンプルドキュメントを使用して、コンポーネントを作成できます。

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

➢ 新しい Telnet コンポーネントを作成する

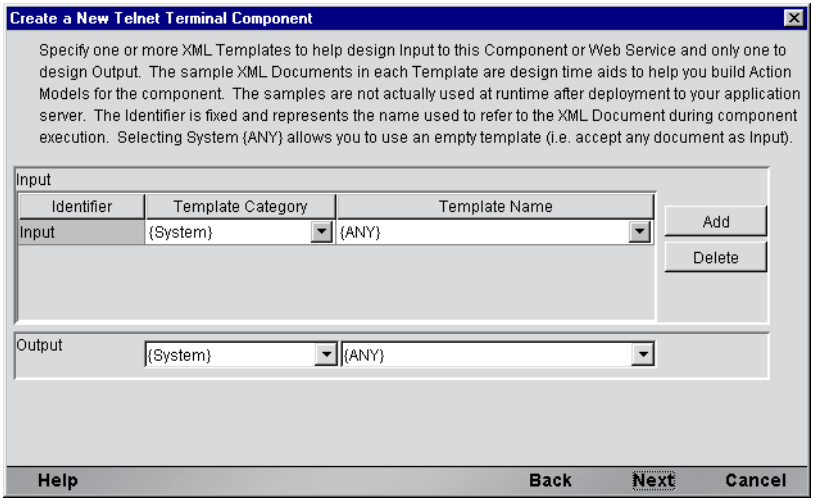
1 ［**File**］>［**New xObject**］>［**Component**］>［**Telnet**］の順に選択します。「Create a New Telnet Component」ウィザードが表示されます。［**Terminal Component**］を選択します。



2 新しい Telnet 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**］をクリックします。New Telnet 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。



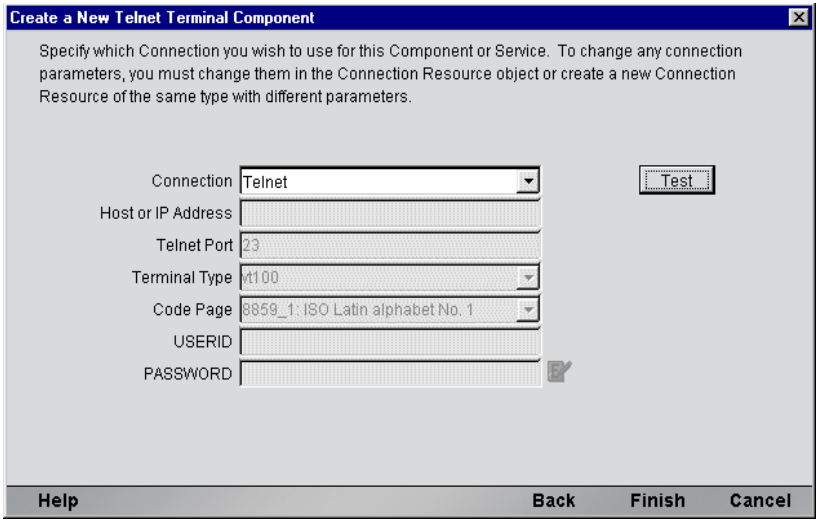
5 「**入力**」テンプレート (複数可) を指定します。デフォルトのカテゴリと異なる場合は、テンプレートカテゴリを選択します。その後、選択したテンプレートカテゴリにある XML テンプレートのリストからテンプレート名を選択します。

6 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。

7 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)。

注記：出力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。

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



9 プルダウンリストで「**接続**」の名前を選択します。Telnet 接続の詳細については、このガイドの第 2 章の「接続リソースの作成」を参照してください。

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

Telnet コンポーネントエディタウィンドウについて

Telnet 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すべての機能が含まれます。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。

しかし、大きな違いが 1 つあり、Telnet コンポーネントエディタには、Telnet 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。いずれのアクションによっても、ネイエィブ環境ペイン内では、この Telnet コンポーネントにより使用された接続リソースで指定したホストとの Telnet エミュレーションセッションが確立されます。

Telnet ネイティブ環境ペインについて

Telnet ネイティブ環境ペインでは、ホスト環境の Telnet エミュレーションが装備されています。このペインでは、「ダム端末」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、Telnet 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。

- Telnet 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 Telnet 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 Telnet 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。

Telnet キーボードサポートについて

Telnet ネイティブ環境ペインでは、特別な端末キーの使用が多数サポートされています。［Terminal Keypad］ダイアログボックス (次を参照) は、［Common Keys］、［NumPad Keys］、［Control Keys］、および［Other Keys］の 4 つのタブで構成されています。各タブには、特定の機能を備えたキーグループが含まれています。

［Expression Builder］ダイアログボックスの［Function/Methods］列で［Telnet］>［Keys］の順に選択すると表示される選択リストを使用すると、追加のキー (F13 から F20 など) を使用することもできます。

➢ 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。テンキーウィンドウには、［Common Keys］、［NumPad Keys］、［Control Keys］、および［Other Keys］という一連のタブがあります。

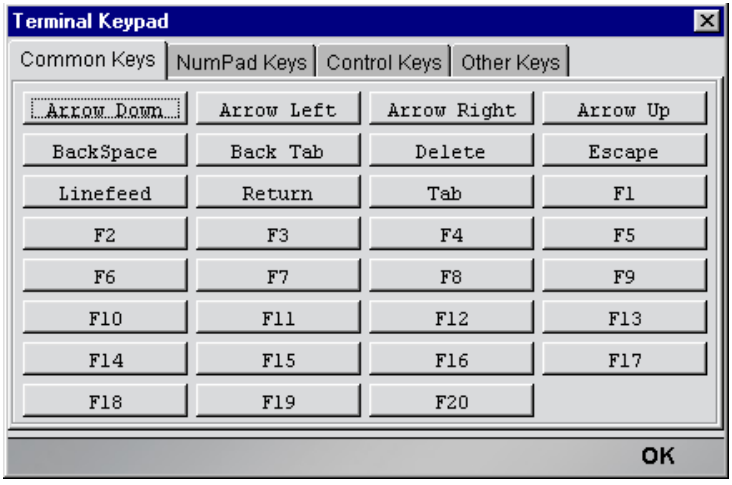
2 該当するタブをクリックして、［Terminal Keypad］で参照するキーを表示します。

3 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合わせます。そのキーに対応する Telnet キーボードが表示されます。クリックしたキーの結果はネイティブ環境ペインで参照できます。

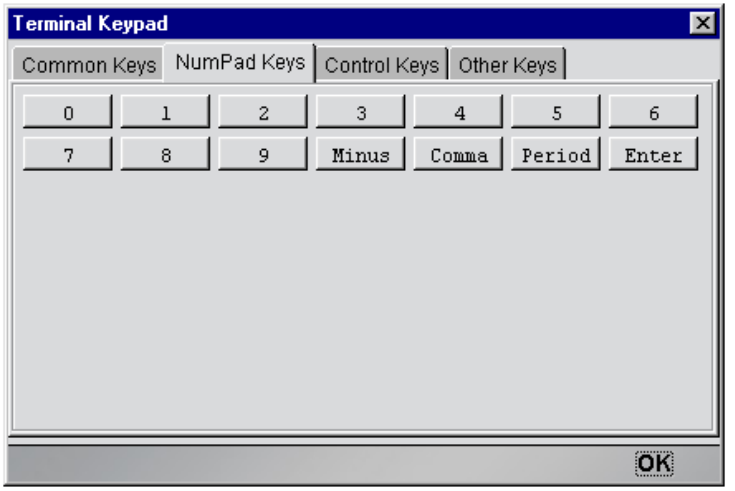
4 ［**OK**］をクリックして、キーパッドを閉じます。キーパッドを再表示するためには、手順 1 を繰り返します。キーパッドを表示すると、最後に使用していたタブが表示されます。

次のページでは、Telnet との通信に使用できる 4 つのタブおよびそれに対応するキーについて説明します。

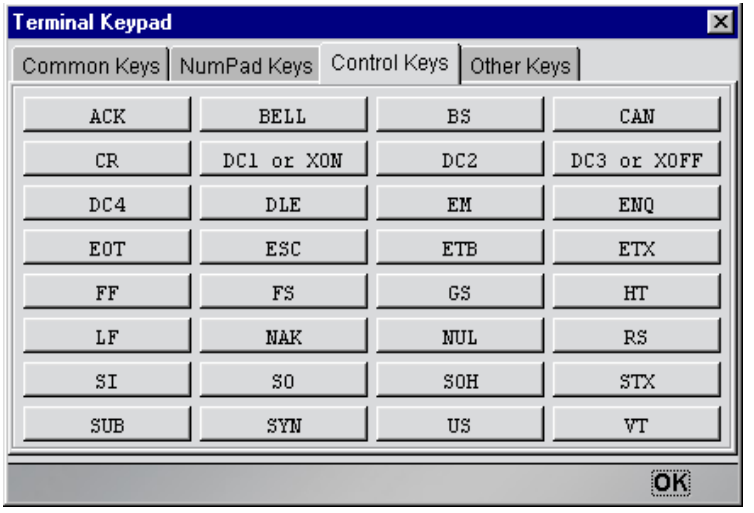
［Common Keys］: 方向キー (［Arrow Down］、［Arrow Left］、［Arrow Right］、［Arrow Up］、［BackSpace］、［BackTab］) や、［Delete］、［Escape］、［Linefeed］、［Return］、［Tab］ が含まれます。［F1］ から［F20］ までのファンクションキーも表示されます。



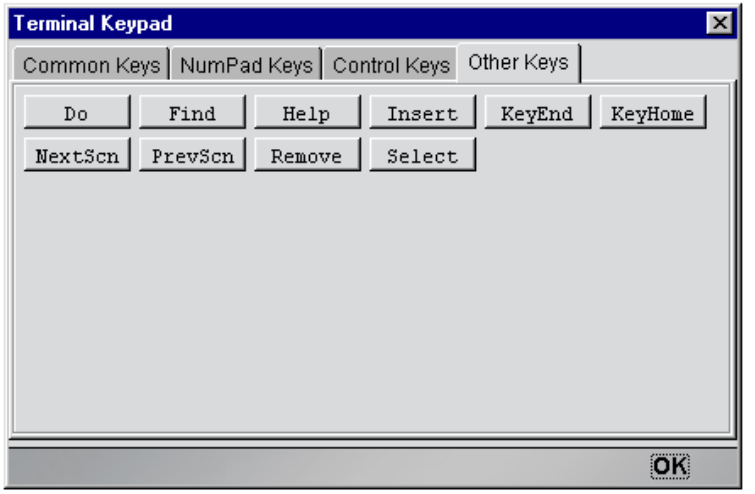
［NumPad Keys］: ［0］ から［9］ までの数字、［Minus］、［Comma］、［Period］、および［Enter］ のキーが含まれます。



［Control Keys］: 特定の機能に関連付けられた 32 個のキーが含まれます。完全なリストについては、付録 B を参照してください。



［Other Keys］:［Help］キーなどの一般的な機能を実行するキーが含まれます。



注記： 特別なキー (印刷以外) および ANSI に対応するキーの完全なリストについては、付録 B を参照してください。

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表示されるエミュレータ画面のバイト列表示で、画面のコンテンツを操作するための方法がいくつかあります。

説明

Telnet コンポーネントは、Telnet「セッション」で文字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。ユーザは、キー入力の形式 (または、カーソルのプロンプトにマップされた XML データ) でホストにデータを送信します。逆に、ホストは単一バイトから画面全体にわたる情報まで、あらゆる情報を含むデータの端末ストリームを送信します。Screen オブジェクトは、現在の画面のデータを表します。これは、24 x 80 ANSI 端末画面では、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届くと、リアルタイムでネイティブ環境ペインに適切な更新が反映されます。これらの更新には、単純なカーソルの位置変更から完全な端末画面の塗り直しまであらゆる内容が含まれます。この意味で、画面のコンテンツは極めて動的になります。

(Check Screen 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、ホストにキー入力を送信し直したり、データをプロンプトにマップする前には、コンポーネントで完全な画面のコンテンツを「認識」または理解する必要はありませんが、画面から DOM にマップする場合にプログラムで Screen オブジェクトにアクセスできると便利です。これを可能にするために、Telnet 用 Connect によって、画面のコンテンツを操作するための ECMAScript 拡張が多数定義されます。これらの拡張については、次の章で詳しく説明します。ここでは、単純な例を取り扱います。たとえば、画面の列位置 20、行 5 に現れる文字列値を取得したいとします。文字列の長さが 10 文字の場合、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 (および、出力 DOM または一時 DOM をターゲットとして使用することによって)、この値を取得できます。

```
Screen.getTextAt( 5, 20, 10 )
```

画面上の行 5、列 20 で始まる 10 文字は、Map アクションのターゲットにマップされます。

詳細な例 (および Screen オブジェクトの完全な API ドキュメント) については、次の章の「Telnet 専用の Expression Builder 拡張」の節を参照してください。

Telnet 固有のメニューバー項目について

［Component］メニュー

［**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**］では、画面を操作する際にアクションの自動作成が有効になり、［**Stop**］では自動作成が終了されます。

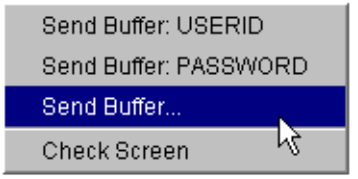
［**Connect/Disconnect**］— このメニューオプションでは、ホストへの接続を制御できます。記録またはアニメーション表示を行う場合は、接続は自動的に確立されます (その結果、［Connection］アイコンが「接続 / 無効」の状態で表示されます)。ただし、このボタンは、記録を行うのではなく、単に Telnet 環境を移動することを目的として接続を確立したい場合に便利です。

Telnet 固有のコンテキストメニュー項目について

Telnet Connect には、この Connect 固有のコンテキストメニュー項目も含まれます。コンテキストメニューを表示するには、適切なペイン (ネイティブ環境ペインまたはアクションペイン) にカーソルを合わせ、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は、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。



4 つのコマンドの機能は、次のとおりです。

［**Send Buffer: USERID**］— このコンポーネントに対する Telnet 接続リソースのユーザ ID に指定した値 (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、アクションモデルで対応する Send Buffer アクションを作成します。

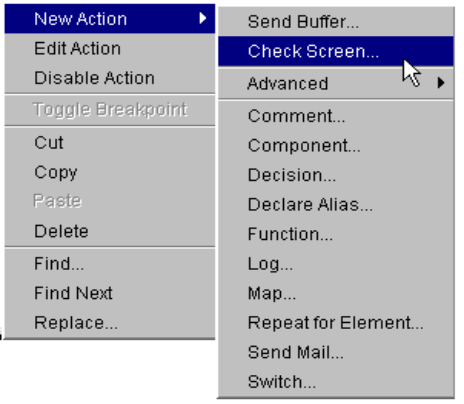
［**Send Buffer: PASSWORD**］— このコンポーネントに対する Telnet 接続リソースのパスワードに指定した値 (存在する場合) に基づいて、パスワード情報をホストに自動的に転送します。また、アクションモデルで対応する Send Buffer アクションを作成します。

［**Send Buffer**］—［Send Buffer］ダイアログボックスを表示し、新しい Send Buffer 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**］— ダイアログボックスを表示せずに、新しい Check Screen アクションを作成します (ツールバーで［Create Check Screen］ボタンをクリックした場合と同じです)。

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)。



コンテキストメニュー項目の機能は、次のとおりです。

［**Send Buffer**］— Send Buffer アクションを作成できます。［Send Buffer Action］ダイアログボックスが表示され、Telnet ホストアプリケーションに送信されるテキストや <Ctrl> キーコマンドを入力できます (このダイアログボックスでは、ECMAScript 式や、入力 DOM の文字列データの場所を表す XPath フラグメントを入力することもできます)。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。

［**Check Screen**］— このコマンドでは、(コンポーネントとホストを同期させるために) 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 アクションに関する詳細な説明については、次の章を参照してください。

Telnet 固有のボタンについて

Telnet Connect には、コンポーネントエディタのメインツールバーに Connect 固有のツールアイコン (または Connect 固有の機能を備えたアイコン、あるいはその両方) が多数含まれています (次を参照)。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 (記録の実行中)

［Record］アイコン (無効な状態)

［Record］ボタンでは、ネイティブ環境ペインを操作する際に、キーボードおよび画面の操作をキャプチャできます。記録された操作は、アクションとしてアクションモデルに配置され、テスト中にこのアクションを「再生」することができます。

［Connection］ボタン

［Connection］(切断された状態)

［Connection］(接続された状態)

［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、コンポーネントの接続状態が切り替わります (コンポーネントに関連付けられている接続リソースの作成中に指定した設定を使用して)。

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ます。その結果、ボタンが「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］ボタンが有効な状態に戻ります。

［Create Check Screen］ボタン

Composer のメインツールバーにある［Create Check Screen］ボタンは、ユーザが最初に端末画面を操作する前にクリックしておく必要があります。これにより、ネイティブ環境ペインで現在表示された画面データを操作することが exteNd Composer に通知されます。このボタンをクリックすると、新しい Check Screen アクションがアクションモデルに挿入されます (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)。

4 Telnet 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。

Telnet 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、非 XML データソースと通信したりするための命令のセットが、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 ドキュメント間で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とサービス内でのデータ転送を実行します。

アクションモデルは、連携したアクションのリストから構成されています。たとえば、あるアクションモデルでは、請求書のデータをディスクから読み取り、データをホストインベントリデータベースから取得し、一時 XML ドキュメントに結果をマップして変換し、変換されたデータを出力 XML ドキュメントにマップします。

このアクションモデルの例は、いくつかのアクションから構成されています。そのアクションは次のとおりです。

- 請求書のドキュメントを開き、Telnet コマンドを実行してホストデータベースから請求書のデータを取得する
- 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

Telnet 専用アクションについて

Telnet Connect には、Telnet 環境専用の Check Screen および Send Buffer という 2 つのアクションが含まれます。

Telnet アクション	説明
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されます) でない限り、実行を続行できないことをコンポーネントに通知します。
Send Buffer	ホストに送信するための文字列をバッファします。文字列は、Map アクションまたはユーザのキー入力、あるいはその両方から作成されます (Send Buffer アクションは、手動で作成できますが、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)。

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) Telnet コンポーネントが、ランタイム時に Telnet 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。

Check Screen アクション

Telnet セッションでは、待ち時間が生じたり、画面データの着信順序が、ホストとアプリケーション間で定義された順序で一定でない可能性があるため、コンポーネントで現在の画面データを操作する前には、特定の状態の端末画面に依存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」が保てるようにします。Check Screen アクションは、アクションモデルのさまざまなポイントで手動で作成して、正しい画面が正しいときに正確に動作するようにします。

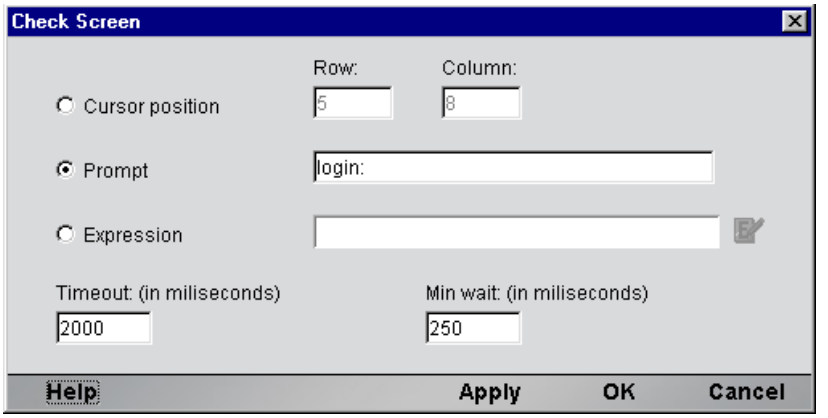
新しい Check Screen アクションを作成するには、次のいずれかの操作を実行できます。

- メインツールバーで［Create Check Screen Action］ボタンをクリックするか、
- 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**］の順に選択するか、
- 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

注記： 記録モードを使用している場合は、ほとんどの場合に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 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 実行許可 (画面の準備状態) 条件を指定する方法に応じて、3 つのラジオボタン (［**Cursor position**］、［**Prompt**］、または［**Expression**］) のいずれかをオンにします (デフォルトは、［Cursor position］です)。次の説明を参照してください。

3 ［**Timeout**］に、タイムアウトの値をミリ秒で指定します (次の説明を参照)。

4 ［**Min wait**］に、Min wait 時間の値をミリ秒で指定します (次の説明を参照)。

5 ［**OK**］をクリックします。

Check Screen アクションについて

Check Screen アクションのダイアログボックスの目的は、次の 2 つです。

- 実行時に画面の状態を判断する基準となる準備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できます。

これらの点は、次に詳しく説明します。Telnet コンポーネントをはじめて作成する前には、次の節をよくお読みになり、内容を十分理解するようにしてください。

準備条件

重要な点として、アクションモデルのアクションは、次の条件を満たすまで実行されません。

1 ホストアプリケーションの準備が整い、

2 すべての画面データが着信済みである (つまり、画面が既知の状態となっている)

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。カーソル位置、プロンプト名、または ECMAScript 式に基づき、準備条件を指定できます。

[Cursor Position]

準備状態を端末のカーソル位置に基づかせることができます。単純に、カーソルの「プロンプト位置」の行数および列数を入力します (ダイアログボックスの［Row］フィールドと［Column］フィールドに示された値は、常にデフォルトで自動的にカーソルの現在の位置になります。通常、手動で数値を入力する必要はありません)。

[Prompt]

現在のプロンプト位置は、端末のエミュレーションウィンドウでのカーソル位置の直前にくる文字列に基づき、指定できます。たとえば、プロンプトで、「Choose one: (A, B, C, D)」と表示される場合、実行許可プロンプトして、「Choose one: (A, B, C, D)」または「(A, B, C, D)」、あるいは単純に「)」を指定できます (プロンプト文字列で示されたデフォルト値は、カーソルを合わせた行に対する現在の画面のコンテンツとなります。デフォルト文字列には、カーソルの前にくるプロンプト行の最初から最後のスペース (スペースがある場合はスペースを含む) まで、すべての文字が含まれます)。

[Expression]

プロンプトの位置またはプロンプトのテキストは、ランタイム時に動的に変更することができます。実行許可条件を柔軟に決定するために、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。ランタイム時に、式で「true」が返された場合、画面は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。

[Timeout]

タイムアウトの値 (ミリ秒単位) は、画面データを着信して、このデータがダイアログボックス上部で指定された準備条件を満たす間、コンポーネントが待機する最長時間を表します。指定のミリ秒数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、Telnet 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幅に異なるため、タイムアウトの値は、慎重に決定する必要があります。「安全な」タイムアウトの値を決定するためには、設計時およびサーバ上ともに、慎重にコンポーネントをテストする必要があります。**

デフォルトのタイムアウトの値は、記録モードを使用しているか、または単純に手動でアクションを作成しているかに応じて異なります。記録モードでは、デフォルトのタイムアウト値は、最後に操作を実行してから新しい画面をロードするまでに経過した実際の時間に基づいて計算された値となります (ダイアログボックスに表示された値は、この「検出されたロード時間」を 2 倍して、最も近い秒数に切り上げた数値です)。(記録モードの代わりに) 手動で Check Screen アクションを作成している場合、デフォルト値は、1500 ミリ秒です。

[Min Wait]

Min Wait 時間 (ミリ秒単位) は、画面バッファを初めて確認する前にコンポーネントが待機しなければならない時間を表します。たとえば、Min Wait に 500 を指定した場合、コンポーネントは、500 ミリ秒間待機してから (指定した条件に基づき) 画面の準備状態を確認します。実行許可条件が満たされた場合、さらに 100 ミリ秒後に、画面が再確認されます。2 番目の確認も適切な場合に限り、コンポーネントの実行が続行されます。そうでない場合は、タイムアウト値 (前を参照) に達するまで、画面が 100 ミリ秒間隔で再確認されます。その時点でも、画面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：すべての Check Screen アクションでは、画面が最低 2 度確認されます。2 度連続で確認が適切とならない場合は、実行許可は発生しません。

Min Wait のデフォルト値は、50 ミリ秒です。ただし、Min Wait 時間には関係なく、画面はタイムアウト時間が切れる際に最終的にもう一度確認され、Min Wait 時間がタイムアウト値より大きい場合でも、画面は 1 度は確認されることになります。

Send Buffer アクション

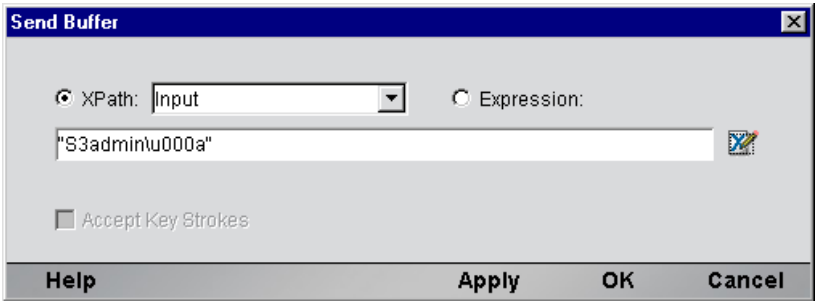
Send Buffer アクションでは、コンポーネントの実行時に 1 度の送信でホストに送信される「キー入力されたデータ」(データが実際にキー入力によって取得されたか、ドラッグアンドドロップのマップによるものか、または Expression Builder で作成した ECMAScript 式によるものかは問わない) をカプセル化します。Send Buffer アクションを実行すると、バッファされたデータが、適切にエスケープされた ANSI バイトストリームの形式でホストに送信されます。

Send Buffer アクションは、何種類かの方法で作成できます。

- 記録モードでは、Check Screen アクションの作成後に、単純に入力していきます。キー入力は、新しい Send Buffer 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すると、コンテキストメニューが表示されます。［**New Action**］、［**Send Buffer**］の順に選択します。
- メインメニューバーの［**Action**］で、［**New Action**］、［**Send Buffer**］の順に選択します。

➢ メニューコマンドを使用して Send Buffer 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右クリックして、コンテキストメニューから［**New Action**］、［**Send Buffer**］の順に選択します (または、前に説明されたように、［Action］メニューを使用します)。［Send Buffer］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 を選択して、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、［**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 (文字列をテキストフィールドに入力することで、) 手動でバッファのコンテンツを指定するには、最初に［**Accept Key Strokes**］チェックボックスをオンにしてから入力していきます。［Expression］ラジオボタンが自動的にオンになり、押したキーすべてがテキスト領域の引用符で囲まれた文字列中に入力されます。コントロールキー (矢印キー、ファンクションキーなど) は、適切なエスケープシーケンスに自動的に変換されます (次の説明を参照)。

5 ［**OK**］をクリックします。

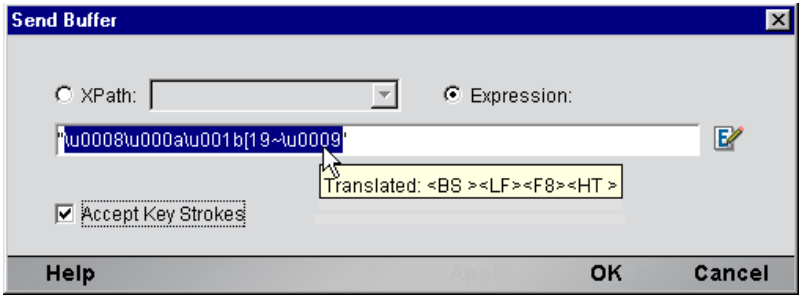
［Send Buffer］ダイアログボックスでのテキストの編集

「Accept Key Strokes」モードの場合、キー入力はすべてエスケープされた文字列リテラル値としてダイアログボックスにキャプチャされるため、<Backspace>、切り取り / 貼り付けなどを使用した通常のテキスト編集は行えません。たとえば、<F11> キーを押すと、削除した前の文字の代わりに、「F11」の値が文字列バッファに追加されます。しかし、この状況を望まないこともあります。

(切り取り、貼り付け、<Backspace> などを使用して) バッファのコンテンツを直接編集するには、最初に［Accept Key Strokes］チェックボックスをオフにしてから、テキストを編集します。キーキャプチャモードに戻るには、［Accept Key Strokes］チェックボックスをオンにします。これで追加のキー入力は、すべてエスケープシーケンスに変換され、既存のテキストに追加されます。

状況によっては、エスケープの値を手動で入力したい場合があります。［Accept Key Strokes］チェックボックスをオフにして、該当する値を現在のテキスト文字列の任意の場所に入力すると、この操作を実行できます。特定のコントロールキーに対するエスケープシーケンスが分からない場合は、テキスト領域の右側にある［Expression］アイコンをクリックしてから (［Expression Builder］ダイアログボックスが表示されます)、［Expression Builder］ダイアログボックス上部の選択リストで適切なコントロールキーエントリをダブルクリックすると検索できます。

特定のエスケープシーケンスの英語の意味を知りたい場合は、単純に目的のエスケープシーケンスを選択 (ハイライト) し、選択項目の上にマウスを合わせます (次を参照)。



エスケープシーケンスの英語の翻訳を示したマウス移動ヘルプボックスが表示されます。たとえば、前の図ではエスケープシーケンス「\u0008」を選択して、選択項目の上にマウスを合わせています。マウス移動ヘルプボックスによって、「\u0008」という組み合わせが、Telnet で <Backspace> (BS) または <Control>-<H> に等しくなることが示されます。

エスケープシーケンスのグループを選択すると、角括弧で囲まれた文字に相当する内容がすべて (マウス移動ヘルプボックスで) 表示されます。たとえば、シーケンス「\u001b[A\u000a\u000d」を選択すると、マウス移動ヘルプには次のように表示されます。

```
< Arrow Up > < LF = CTRL+J > < CR = CTRL+M >
```

特殊 (非印字) キー、およびそれに対応する ANSI のリストは、付録 B「Telnet の対応キーボード」に示されています。

Send Buffer アクションおよび記録モードについて

記録モードでアクションモデルを作成する際には、［Check Screen］ボタンをクリックして、入力していくと、新しい Send Buffer アクションが自動的に作成されます。この結果、［Check Screen］ボタンをクリックしてから入力し (または、要素を入力 DOM から画面上のプロンプト領域にドラッグする)、ホストから次の画面を着信するまで待機してから、［Check Screen］をクリックして入力 (またはドラッグ) していくという操作を繰り返すだけでよいため、アクションモデルの作成が簡単になります。このように、Check Screen アクションと Send Buffer アクションのシーケンスは、非常に簡単で自然に作成できます。

Send Buffer アクションが自動的に作成されると、次のいずれかの操作を実行するまで、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。

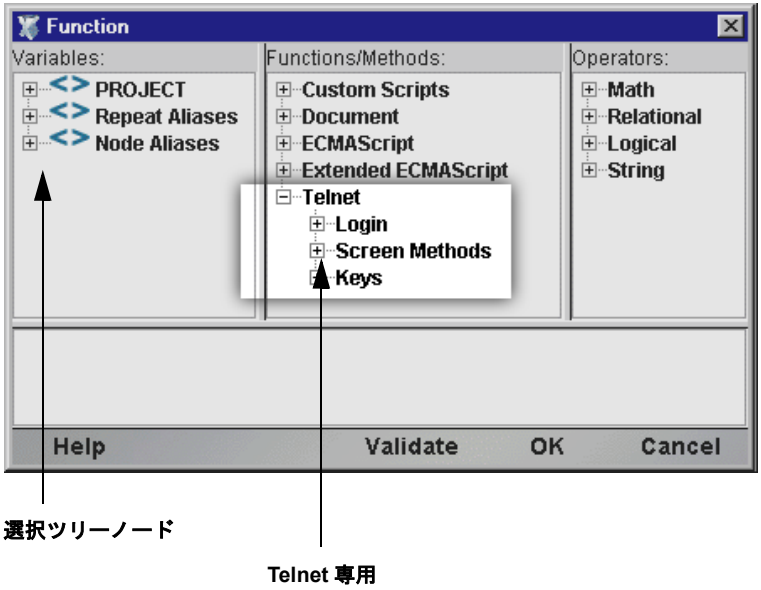
- マウスを右クリックする。
- アクションモデルで新しいアクションを作成していく。
- ネイティブ環境ペインの内外にデータをドラッグする。
- ［Record］ボタンを記録を行わない状態に切り替える。

アクションモデルでキーを表示する方法

Send Buffer アクションを作成すると、リアルタイムでキャプチャされるキー入力は、英数字の値、または (非印字文字の場合は) エスケープされた形式でアクションモデルに表示されます。たとえば、「上向き矢印」は `\u001b[a` に変換されます。ここで、**\u001b** は、ANSI コントロールコードの 2 バイトの 16 進 Unicode 値を表し、**[a** は、上向き矢印に対する残りの ASCII エスケープシーケンスを表します。<Backspace> キーおよび <Delete> キーの入力もエスケープシーケンスとして表されます。そのため、Send Buffer アクションで入力ミスを訂正する場合は、アクションモデルでアクションをダブルクリックして (［Send Buffer］ダイアログボックスが表示されます)、バッファ文字列を手動で編集します。

Telnet 専用の Expression Builder 拡張

Telnet 用 Connect では、Telnet 専用の ECMAScript のグローバルおよびオブジェクト拡張が多数提供されており、Expression Builder の選択リストに表示されています。Telnet 専用項目のリストは、［Telnet］というラベルの付いたノードで表示され、［Login］、［Screen Methods］、および［Keys］という 3 つのチャイルドノードがあります (次の図を参照)。



[Login]

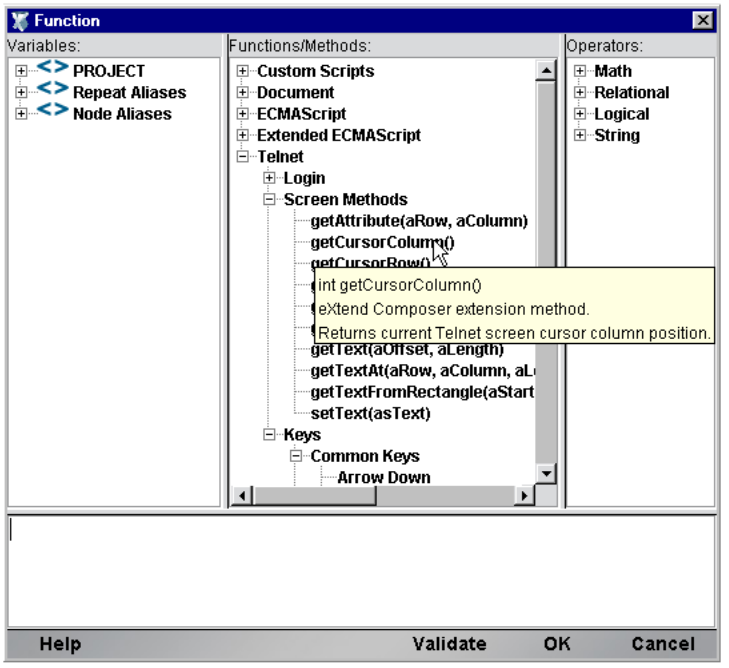
Telnet 接続リソースには、[Expression Builder] ダイアログボックスからアクセスできる「USERID」および「PASSWORD」という 2 つのグローバル変数があります。これらのプロパティ (選択ツリーの [Login] ノードで使用できる) では、接続時にホストシステムで要求される可能性があるユーザ ID およびパスワードの値を指定します。これらの変数は、端末画面にマップできるため、Map アクションで明示的にユーザ情報およびパスワード情報を入力する必要がなくなります。

注記： また、XPath ソースが $PASSWORD として定義された Send Buffer アクションを作成することもできます。

[Screen Methods]

Telnet 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、Telnet 専用の特別な ECMAScript 拡張が表示されます。この ECMAScript 拡張は、Screen オブジェクト、および仮想端末キーボードで「特殊キー」に対応する事前定義されたエスケープシーケンスの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合わせると、マウス移動ヘルプが使用できます (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、シグネチャ、および使用規則を示したメソッドが提供されています。

int getAttribute(*nRow, nColumn*)

このメソッドは、nRow、nColumn によって指定された画面の位置にある文字の「display 属性」値を返します。使用できる display 属性値の完全なリストは、付録 C 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 ) == 1) // if character at 5, 20 is bold
     // do something
```

int getColumnCount(*void*)

このメソッドは、現在の画面に固有の列幅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、この値は画面に応じて変更できます。この値は、コンポーネントの有効期間中一定にならないようにしてください)。プログラムが 24x80 モードの場合、このメソッドは 80 を返します。現在の画面で、固有のサイズに関係なく、行 15 のコンテンツをすべて取得するには、次のようにできます。

```
var myRow = Screen.getTextAt( 15, 1, Screen.getMaxColumn() );
```

int getCursorRow(*void*)

このメソッドは、Telnet エミュレータ画面 (ネイティブ環境ペイン) 内で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、このメソッドは 1 から 24 の値 (1 と 24 を含む) を返します。

int getCursorColumn(*void*)

このメソッドは、Telnet エミュレータ画面 (ネイティブ環境ペイン) 内で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、80x80 モードでは、このメソッドは 1 から 24 の値 (1 と 24 を含む) を返します。

String getPrompt(*void*)

`getPrompt()` メソッドは、列 1 から `getCursorColumn()` までで (getCursorColumn() は含まない)、カーソルの行にあるすべての文字、つまり、行頭からカーソル位置までにあるすべての内容を表す文字列を返します (これは、[Check Screen] ダイアログボックスに表示されたデフォルトのプロンプト文字列と同じです)。例 :

```
var theprompt = Screen.getPrompt();

if (theprompt().toLowerCase().indexOf("password") != -1)

   Screen.setText(PASSWORD);
```

int getRowCount(*void*)

このメソッドは、現在の画面に固有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、この値は画面に応じて変更できます。この値は、コンポーネントの有効期間中一定にならないようにしてください)。プログラムが 24x80 モードの場合、このメソッドは 24 を返します。固有のサイズに関係なく、画面の行をすべてループするには、次のようにできます。

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

String getText(*nOffset, nLength*)

このメソッドは、nOffset によって指定されたバイトオフセットで、Screen オブジェクトで生じる文字 (長さ nLength) の文字列を返します。オフセットは、ゼロではなく、1 を基準とします。そのため、ECMAScript 文字列として 24 x 80 画面を「すべて」取得するには、次のようにします。

```
var wholeScreen = Screen.getText( 1, 24 * 80 );
```

画面バッファの限度を超えて文字データ取得しようとすると、例外が発生します。たとえば、次のような呼び出しは実行できません。

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

String getTextAt(*nRow, nColumn, nLength*)

このメソッドは、現在の画面での、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ゼロ値では、いずれも例外が発生します。

24x80 画面の行 20 を取得するには、次のようにします。

```
var myRow = Screen.getTextAt( 20, 1, 80 );
```

getTextAt() メソッドは、次の「連続するデータの選択」の説明のように作成された、画面の選択に関する Map アクションをドラッグアンドドロップして内部的に使用されます。

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

このメソッドは、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 (\u000a) 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。

void setText(*String*)

setText() メソッドでは、明示的に Send Buffer アクションを作成せずに、プログラムによって画面 (および、ホストアプリケーション) にデータを送信できます (例を参照)。

```
var myPhone = "(203) 225-1800";

if (Screen.getPrompt().indexOf("Phone") != -1)

     Screen.setText( myPhone + "\r" ); // send string + CR
```

［Keys］

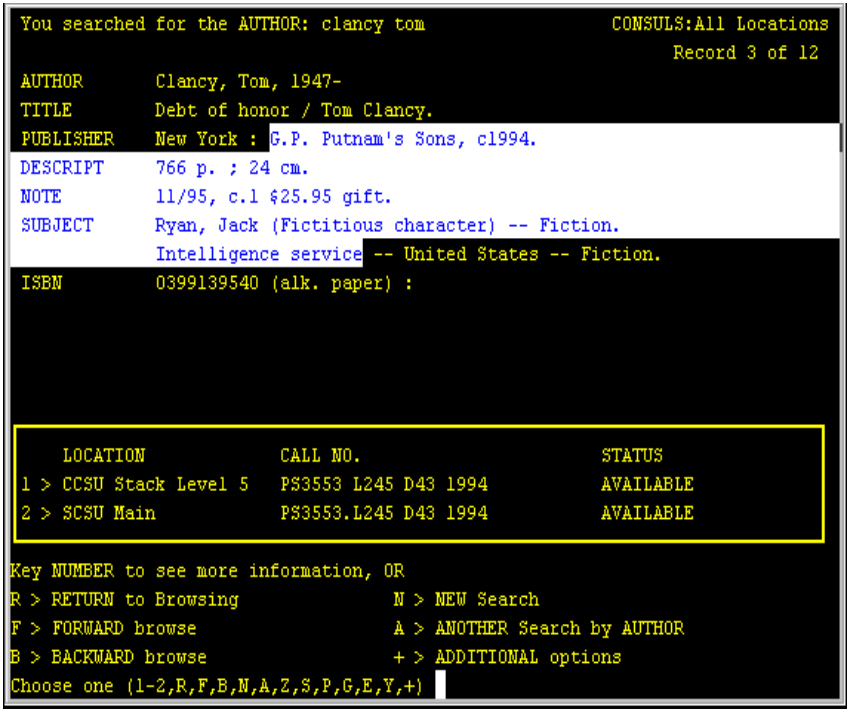
［Expression Builder］ダイアログボックスの Telnet 専用選択ツリーの［Keys］ノードには、［Common Keys］、［NumPad Keys］、［Control Keys］、および［Other Keys］というラベルが付いたチャイルドノードがあります。これらのカテゴリで選択リスト項目をダブルクリックすると、ホストに送信しようとしている任意の非印字文字に対して、ANSI エスケープシーケンスを自動生成できます。選択リスト項目の詳細なコンテンツについては、付録 B を参照してください。

Telnet Connect での画面の選択

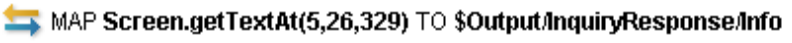
設計時に、端末画面の外にデータをドラッグする目的で端末画面 (ネイティブ環境ペイン内) でデータを選択するには、主に 2 つの方法があります。1 つの方法では、ある画面バッファオフセットから別の画面バッファオフセットに連続するストリームでテキストを選択し、もう 1 つの方法では、任意の画面上で境界ボックスまたは境界部分でテキストを選択します。

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 (マウスを合わせた点) から最後の画面オフセット (マウスを離した点) までにある「すべての」文字列が選択されます (次の図を参照)。選択したテキストは、「黒くハイライト」されます。上から行の一部、3 つの完全な行、行の一部という順に選択されます。



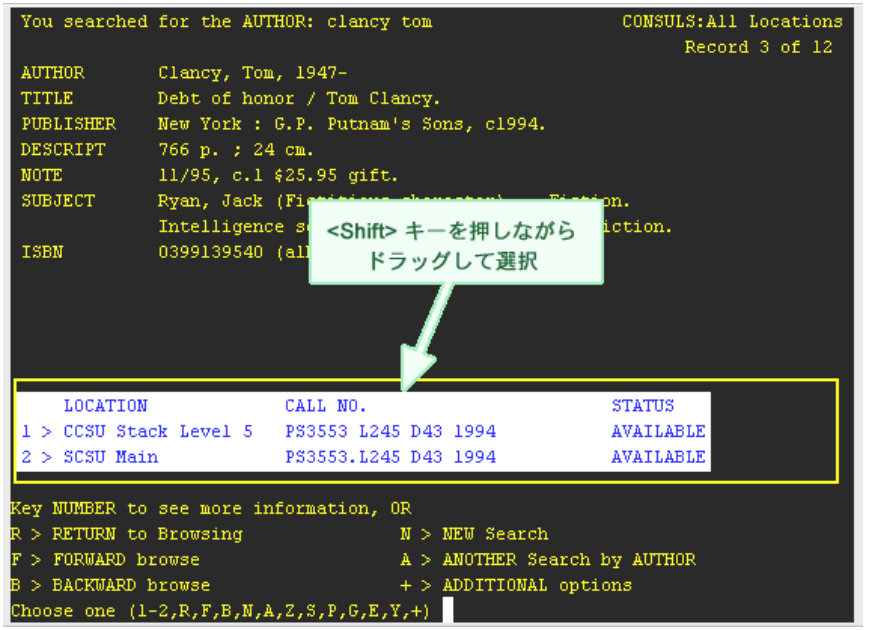
コンポーネントエディタウィンドウのステータス行 (左下) で示すように、前の例での選択内容は、実際に行 5、列 26 で開始し、行 9、列 35 で終了します。この選択内容をネイティブ環境ペインから DOM 内にドラッグすると、Map アクションが次のように生成されます。



`getTextAt()` 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/Inquiry/Response/Info** 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 (黒色で表示された画面領域は、単純に文字列内でスペース文字として表されます)。

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、前の画面では、画面の上から 3 分の 2 の位置に、特定の本が利用できるかどうかを示したボックスがあります。画面上のこの長方形で囲まれた部分のデータのみを (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 <Shift> 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、49 ページで説明した `getTextFromRectangle()` メソッドが使用されます。アクションは、次のようになります。

`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、`getTextAt()` 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)。

サンプルプログラムについて

説明を目的として、次の例では CONSULS プログラムが使用されています。この Telnet プログラムは、コネチカット州立大学の図書館システムによってオンラインで提供され、このプログラムのユーザは、タイトル、著者、および他の条件から本や定期刊行物を検索することができます。

Telnet セッションの記録

Telnet 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 Telnet 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または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、Telnet 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： Telnet コンポーネントを正常に作成するためには、Telnet コマンド、および XML 統合プロジェクトで使用する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。

Telnet 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。

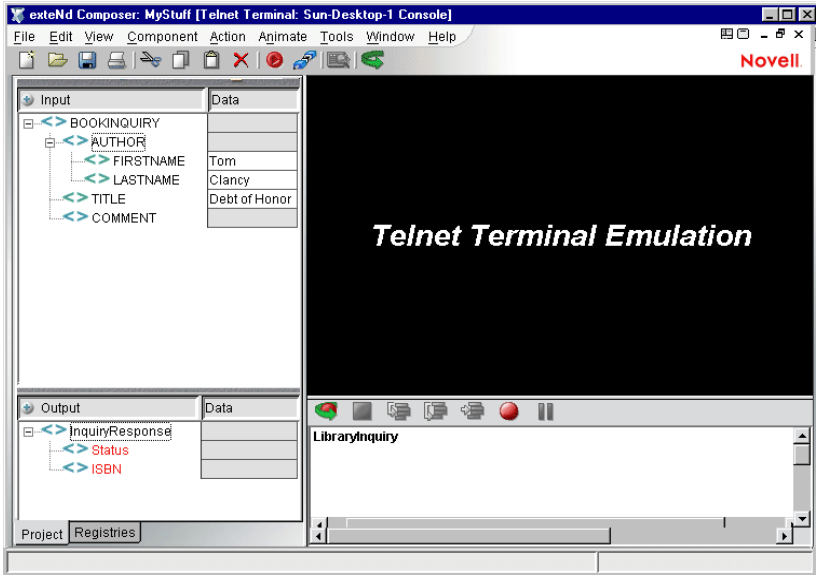
- Check Screen アクションを作成する
- Send Buffer アクションを自動的に作成する
- 入力 DOM 要素を Telnet 画面のプロンプトにドラッグアンドドロップしてマップする
- ネイティブ環境画面から出力 DOM にドラッグアンドドロップしてマップする
- ECMAScript 式を使用して、Screen オブジェクトの要素を操作する

次の例では、まず本のタイトルおよび著者を含む入力 XML ドキュメントから操作を開始します。Web サービスの目標は、CONSULS Telnet アプリケーションを使用して、著者のオンライン検索を行い、指定したタイトルの本がライブラリシステムに存在するかどうかを確認することです。存在する場合、出力 DOM で ISBN (International Standard Book Number) コードを取得します。成功した場合、しなかった場合を問わず、出力 DOM に適切なステータスメッセージを挿入します。

➢ Telnet セッションを記録する

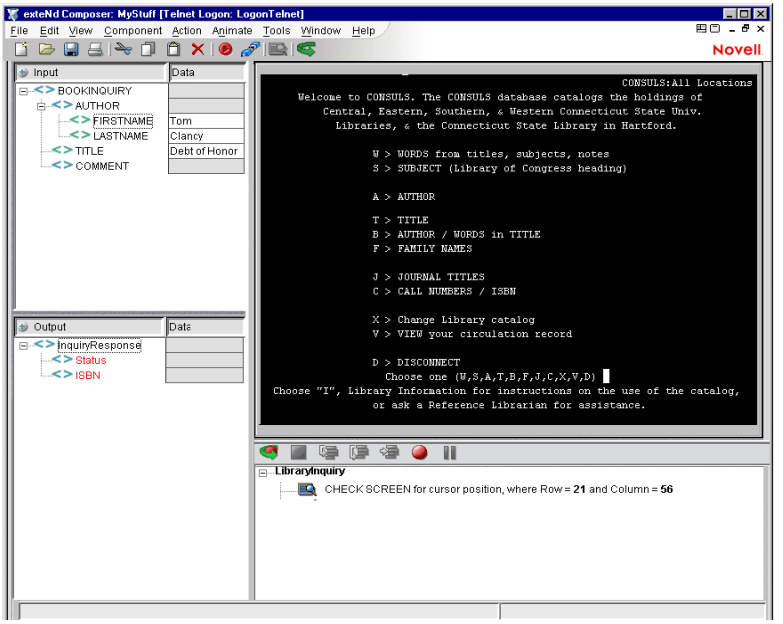
1 前章の 25 ページで説明された手順で、Telnet コンポーネントを作成します。

2 作成すると、Telnet 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［Telnet Terminal Emulation］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)。

注記： この例の残りでは、州立大学図書館システムのオンライン書籍検索サービスからの画面を示します。同様な Telnet サービスがオンラインで数多く利用できますので、好きな Web 検索エンジンを使用して、サービスの IP アドレスを取得してください。

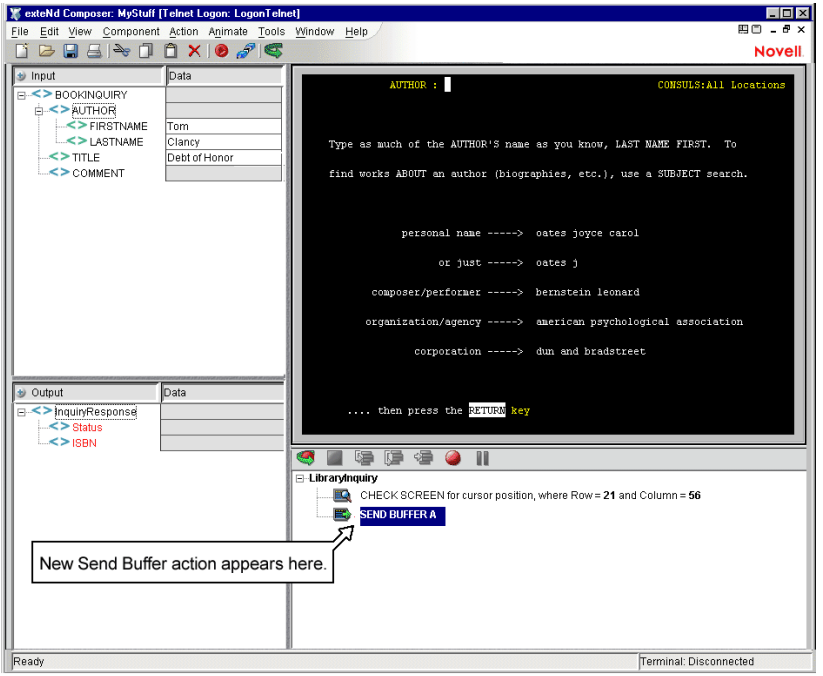


4 ツールバーで［**Create Check Screen Action**］ボタンをクリックします。アクションリストに新しい Check Screen アクションが表示されます。「現在のカーソル位置」(常に、この画面上では 21,56 と予想されますが、将来このコンポーネントの実行時には、この予想は疑問の余地があります) に基づき、デフォルトで実行許可条件になります。CONSULS プログラムの応答時間は、比較的短いため、この Check Screen アクションに対してデフォルトで 1500 ミリ秒となるタイムアウトを暫定的に受け入れます (その場合でも、このタイムアウト値が安全であることを検証するため、コンポーネントを慎重にテストする必要があります)。

5 Telnet 環境ペインの入力画面に文字 **A** (著者 : Author) を入力します。コンポーネントのアクションリストに新しい Send Buffer アクションが自動的に表示されます。入力した「A」は、すでにアクション内に含まれています。

注記： 大抵の場合、Telnet 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する必要があります。

このホストアプリケーションのこの部分では、(<Enter> または <Return> を押さずに) 1 文字を入力するだけで、新しい画面が表示されます。つまり、ホストによって入力した文字がただちに処理されることになります。これは、一般的な Telnet の慣用法で、新しい画面を表示するために、常に <Return> または <Enter> を押す必要があるとは限りません。



「A」への応答として、ホストプログラムによって新しい画面が送信されます (前を参照)。

6 Send Buffer アクションを終了して、次に新しい画面を操作するため、この時点でツールバーで［Check Screen］ボタンをクリックして、コンポーネントが次のアクションと現在の画面を「同期」できるようにする必要があります。［**Create Check Screen Action**］ボタンをクリックします。アクションリストに新しい Check Screen アクションが表示されます。

注記： この時点で、(最初に新しい Check Screen アクションを作成せずに、) 次のコマンドを単純に入力していく場合は、コマンドはアクティブな Send Buffer アクションに追加されます。本質的には、「先読み入力」バッファを作成することになります。ランタイム時には、(連結された 2 セットの画面コマンドを含む) バッファがすべて 1 度に送信されます。先読み入力は、この特定のプログラムでは適切に動作しますが、この方法は他の実際の Telnet プログラムでは正常に動作しない場合があります。そのため、Send Buffer アクションを意図的にオーバーロードする際には注意が必要です。「最良の方法」は、セッション中に表示される新しい画面すべてに対して、新しい Check Screen アクションを作成することです。

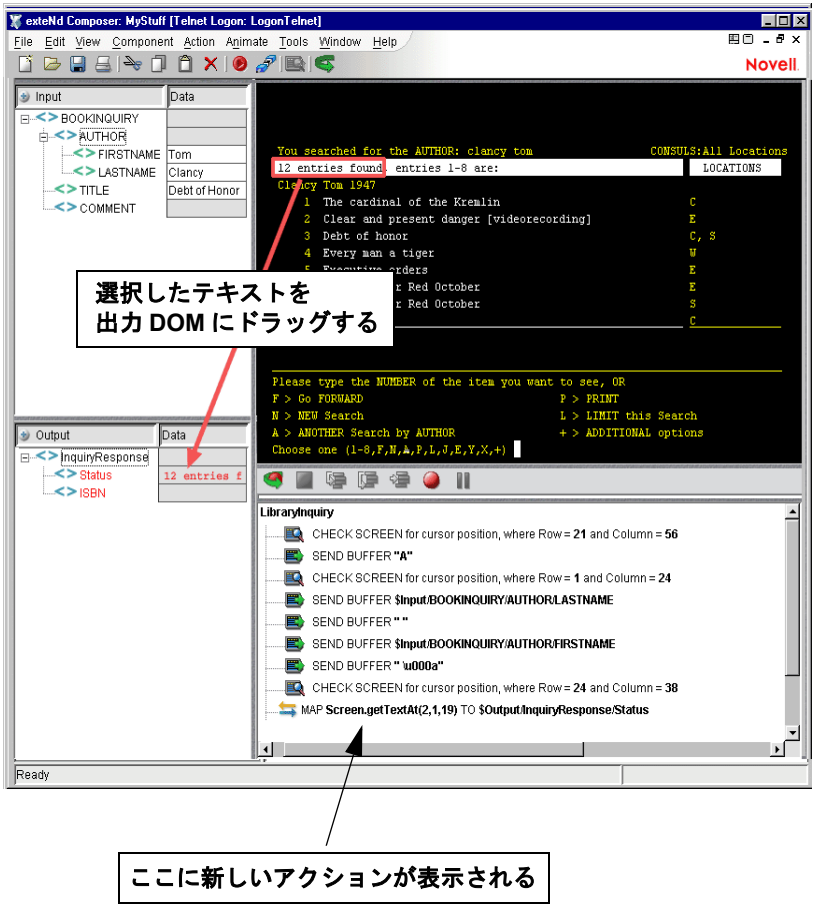
7 ［BOOKINQUIRY］、［AUTHOR］、［LASTNAME］の順に展開されたノードを、入力 DOM からネイティブ環境ペインのカーソル位置にドラッグします。「Clancy」(「」なし) が、プロンプトゾーンに表示され、新しい Send Buffer アクションがアクションモデルに自動的に表示されます。

注記： この Telnet アプリケーションでは、姓、名の順序 (姓と名の間にスペースあり) で著者名を指定することが予想されているため、最初に［LASTNAME］の要素をドラッグしました。

8 キーボードのスペースバーを押します。ネイティブ環境ペインで、「Clancy」にスペース文字が追加されます。また、スペース文字のみを含む新しい Send Buffer アクションが作成されます。

9 ［BOOKINQUIRY］、［AUTHOR］、［FIRSTNAME］の順に展開されたノードの要素を、入力 DOM からネイティブ環境ペインのカーソル位置にドラッグします。「Tom」(「」なし) が、プロンプトゾーンで「Clancy」の後に表示され、新しい Send Buffer 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。

10 端末画面は、<Return> または <Enter> を待っているため、変更されていません (ホストは、入力によって動作していません)。<**Return**> または <**Enter**> を押して、クエリ文字列 (著者名) が完了したことをホストに通知します。**\u000a** を含む新しい Send Buffer アクションが表示され、クエリ結果を反映してネイティブ環境ペインが更新されます。



11 ツールバーで［**Create Check Screen**］ボタンをクリックします。新しい Check Screen アクションが表示され、行 24、列 38 のカーソル位置に基づくデフォルトの実行許可条件が表示されます (行 24 は、一番下の行で、列 38 は、80 列から成る画面の約半分の位置にある列です。前のスクリーンショットを参照)。この場合、デフォルトの Check Screen アクションを変更する必要はありません。

12 ネイティブ環境ペインで、マウスをクリックしてからドラッグして、行 2、列 2 から列 18 までの端末画面のテキストを選択します。

注記： 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、コンポーネントエディタウィンドウのステータス行 (左下隅) に表示されます。

13 マウスのボタンから指を離して、選択したテキストの上にマウスを合わせます。カーソルが指の形になって表示されます。選択内容をクリックして、出力 DOM で［**InquiryResponse**］の［**Status**］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。

14 ［Record］ボタンをクリックして、記録をオフにします。

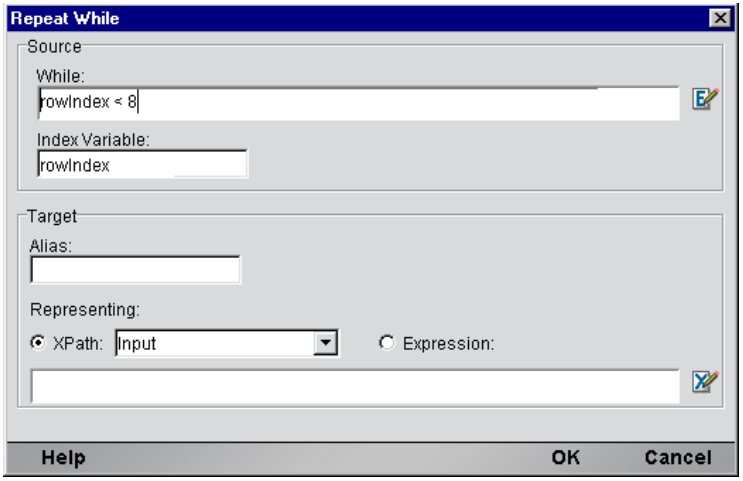
データ検索での複数行のループ

CONSULS の例 (前を参照) での目的は、関心のある本の ISBN (International Standard Book Number) 情報を検索し、出力 DOM にマップすることです。そのため、CONSULS アプリケーションで著者の検索結果が表示されると、画面をスキャンして該当する本のタイトルを検索する必要があります。タイトルが存在する場合、次のアクションとして、対応する行番号を送信し、CONSULS で本の詳細情報 (ISBN を含む) を示す「新しい」画面が表示されるようにします。

単純に端末エミュレータ画面 (前の図を参照) の見た目で、Tom Clancy の『*Debt of Honor*』が、検索結果画面の品目番号 3 として表示されていることを簡単に確認できます。ただし、これは、この特定の検索に対してのみ当てはまります。異なる著者 / タイトルの組み合わせを検索すると、異なる行の位置でヒットが生じる場合があります (または、Tom Clancy がさらに多くの本を出版した場合、『*Debt of Honor*』のリストでの位置は変更される場合があります)。ランタイム時に本の行の位置を判断するには、端末画面の行 4 から行 11 を繰り返して、入力 DOM で［BOOKINQUIRY］の［TITLE］ノードに保存されている文字列を検索する必要があります。前の例を踏まえて、この操作を行う方法は次の例のとおりです。

➢ 1 度に 1 行ずつデータ項目を検索する

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**］、［**Advanced**］、［**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。

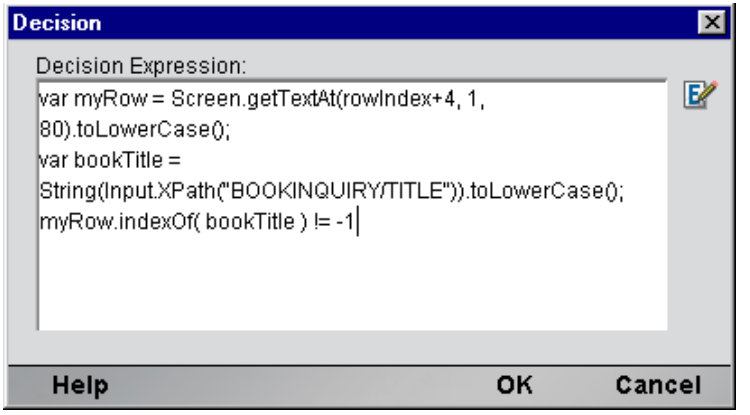


2 ［**While**］テキスト入力ボックスで、このループに適用するループの終了条件を表す式を入力します。この場合、条件は、インデックス変数 `rowIndex` のチェックです。全部で画面データの 8 行をチェックします。

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、`rowIndex`)。

4 画面から単一の値 (本 1 冊) を取得しているだけなので、ダイアログボックスでオプションの［Target］部分を記入する必要はありません。そのため、単純に［**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 この例では、指定の行内で特定の文字列を検索しています。文字列が検索された場合、複数のアクションを実行してからループを抜け出します。Decision アクション内で、行の解析および文字列の検索を実行します。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**］>［**Decision**］を選択して、Decision アクションを作成します。［Decision］ダイアログボックスが表示されます。



6 ［**Decision Expression**］に決定式を入力します。この例では、3 行の式は、次のようになります。

```
var myRow = Screen.getTextAt(rowIndex+4, 1, 80).toLowerCase();
var bookTitle =
       String(Input.XPath("BOOKINQUIRY/TITLE")).toLowerCase();
myRow.indexOf( bookTitle ) != -1
```

1 番目の行では、Screen オブジェクトの `getTextAt()` メソッド (45 ページを参照) を使用して、`rowIndex + 4` で 80 文字のデータ (つまり、24x80 の端末画面での完全な 1 行) を取得します。画面データの検索は、行 4 で始まり、行 11 まで続く必要があるため、インデックス変数にオフセットを追加します (インデックス変数自体は、0 から 7 までの値を持ちます。`rowIndex` が 8 になると、ループが終了します)。

前のコードで 2 番目の行は、単純に小文字の文字列として入力 DOM から本のタイトルを取得します (検索で大文字と小文字が区別されないようにするため、クエリ文字列およびターゲットオブジェクト文字列のいずれも、強制的に小文字にします)。

コードの最後の行は、実際の「条件チェック」です。これは、引数文字列がメソッドを呼び出す文字列の下位文字列でない場合に、–1 を返すコアの ECMAScript 文字列メソッド `indexOf()` に依存します。

7 Decision アクションの TRUE 分岐で、新しい **Send Buffer** アクションを作成します (マウスを右クリックしてから、コンテキストメニューで［**New Action**］>［**Send Buffer**］の順に選択します)。［Send Buffer］ダイアログボックスが表示されます。

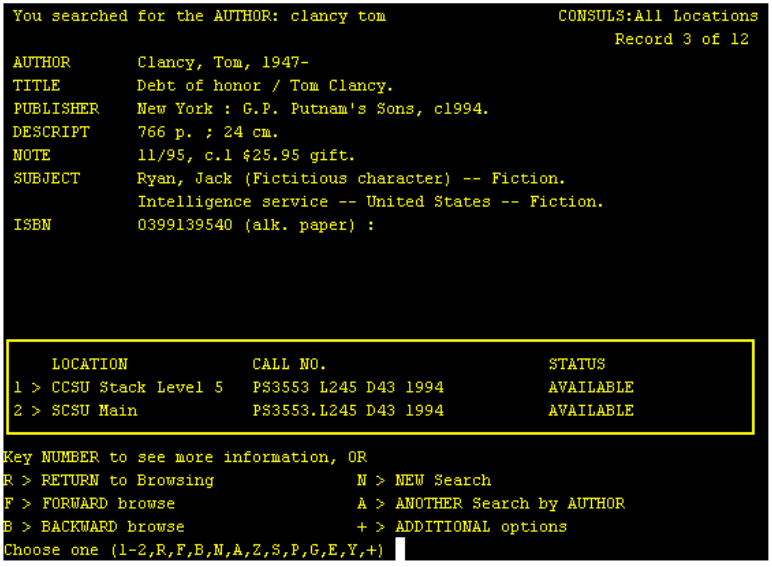
8 ［**Expression**］ラジオボタンをオンにしてから、テキスト編集領域に ECMAScript 式を入力します。ここでの入力例は、次のとおりです。

```
var item = Screen.getTextAt( rowIndex + 4, 1,10);
var regex = new RegExp("\\d+");
item.match(regex)[0];
```

1 番目の行は、`getTextAt()` メソッドを使用して、「ヒット」の行でデータの最初の 10 文字を取得します。この文字列内では、本の CONSULS 行数 (例 : 3) を表す数字の最初の下位文字列を取得する必要があります。この下位文字列を抽出する 1 つの方法として、ECMAScript 文字列メソッド `match()` を使用して、正規表現オブジェクトを引数として実行します。成功すると、このメソッドは、0 番目の項目が一致したテキストとなる配列を返します。正規表現は、円記号、d、プラス記号の順で構成され、「1 行に 1 桁または複数桁の文字」を意味します。

注記 : RegExp コンストラクタは、文字列引数をとり、「リテラル」円記号として表示される円記号は、「円記号でエスケープされる必要があります」。

これらの ECMAScript 行の最終的な結果は、ターゲット行 (つまり、「3」) で本のタイトルの前にくる数字は、Send Buffer アクションを介してホストアプリケーションに送信されます。数字「3」には改行を付ける必要はありません。ホストアプリケーションは、この数字を受信すると、ただちに指定された本に関する詳細情報を示した新しい画面を送信し直します (次を参照)。



9 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**］の順に選択して、新しい Check Screen アクションを作成します。［Check Screen］ダイアログボックスが表示されます。

10［**Expression**］ラジオボタンをオンにしてから、テキスト編集領域に「true」と入力します。［**Min wait**］の値を、(この場合は) 経験上十分であると判明している 100 に設定します。

注記：「true」および 100 の組み合わせは、100 ミリ秒以内に送信された任意の画面データを「自動的に受け入れる」ことを意味します。

11 新しい Function アクションを作成します (マウスを右クリックして、［**New Action**］>［**Function**］の順に選択します)。このアクションで、ページ上の最初の ISBN 番号 (存在する場合) を取得し、ECMAScript グローバルに保存します。

使用する式は、次のとおりです。

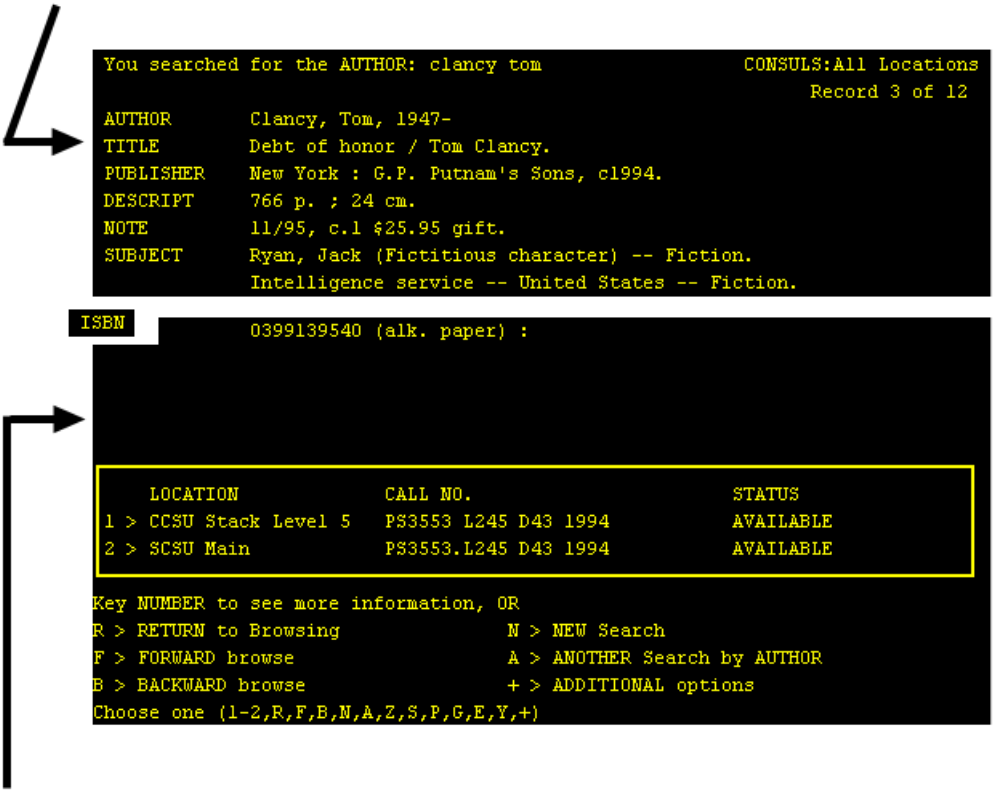
```
this.isbn = "Not found"; // set up global

var screen = Screen.getText( 1, 24 * 80 ); // fetch whole screen

if (screen.indexOf('ISBN') != -1) // if 'ISBN' occurs, get it

    this.isbn = lTrim( screen.split('ISBN')[1] ).split(' ')[0];
```

前の式で、最初の行は、単純に ECMAScript グローバル変数を宣言して初期化します (ECMAScript グローバル変数は、成功すると、有効な ISBN 値で上書きされます)。

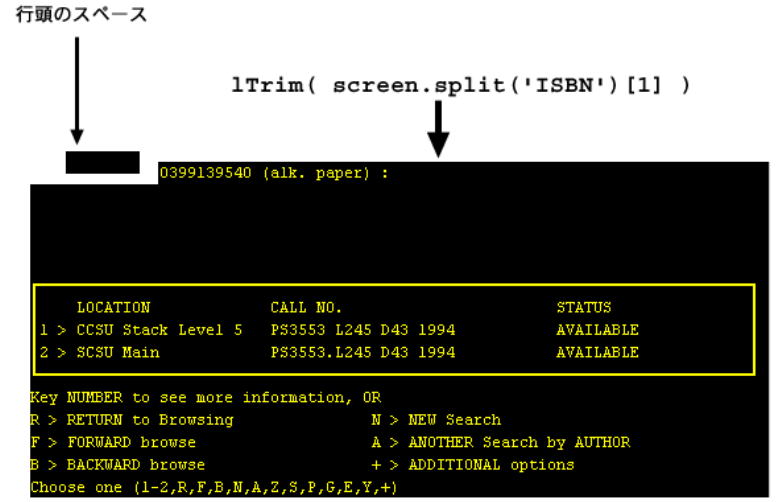
コードの 2 番目の行は、画面バッファ全体を文字列として取得し、ローカル変数 `text` 内に配置します (ここでは、24x80 モードであると想定します)。

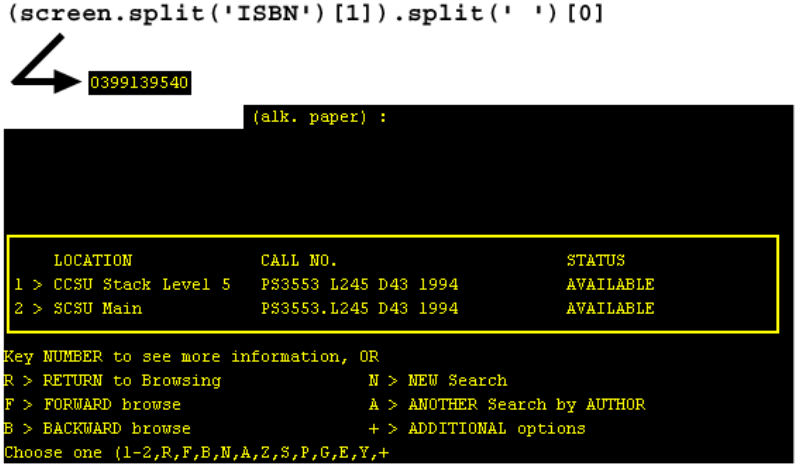
3 番目の行は、画面バッファをチェックして、画面バッファ内に「ISBN」が生じるかどうか確認します。生じた場合、「ISBN」を区切り記号として使用して、バッファを下位文字列の配列へと分割します。インデックス 1 での配列メンバーには、ISBN 数字が含まれ、画面の一部に当たる情報 (および、おそらく 1 つまたは複数の行頭のスペース文字を含む) が含まれます。スペースが区切り文字である場合、split メソッドをもう一度使用して、文字列を下位文字列の配列に分割し、さらにカスタム ECMAScript 関数 `lTrim()` を使用して、行頭のスペースを削除します。この最後の配列で 0 番目の項目が、対象となる ISBN 文字列です (次の一連の図を参照)。

```
screen.split('ISBN')[0]
```




```
screen.split('ISBN')[1]
```

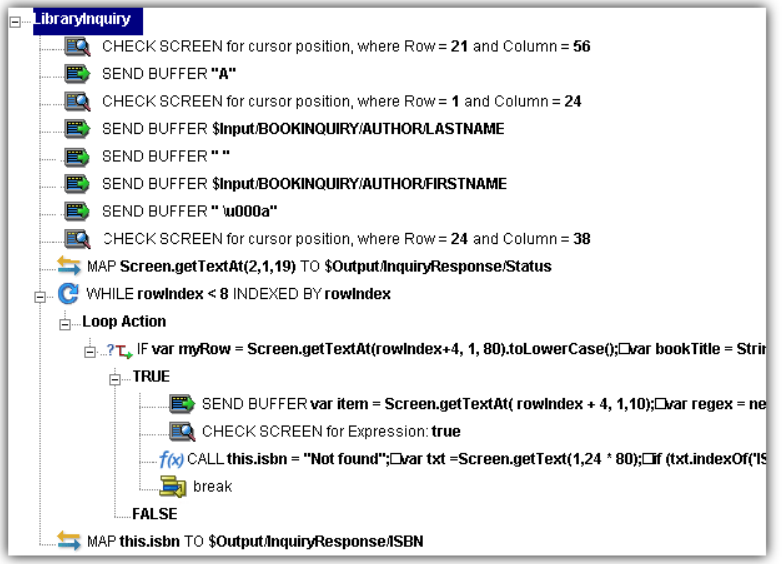




12 該当する情報が見つかった場合、品目全体を繰り返す必要はありません。そのため、「**Break アクション**」を作成して、ループを抜け出します (マウスを右クリックして、[New Action]、[Break] の順に選択します)。

13 `this.isbn` を、出力 DOM で［**InquiryResponse**］の［**ISBN**］ノードにマップする「**Map アクション**」を作成します。

完全なアクションモデルは、次のとおりです。



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、Telnet コンポーネントの編集には、特別な注意が必要です。Telnet 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、次の推奨事項が適用されます。

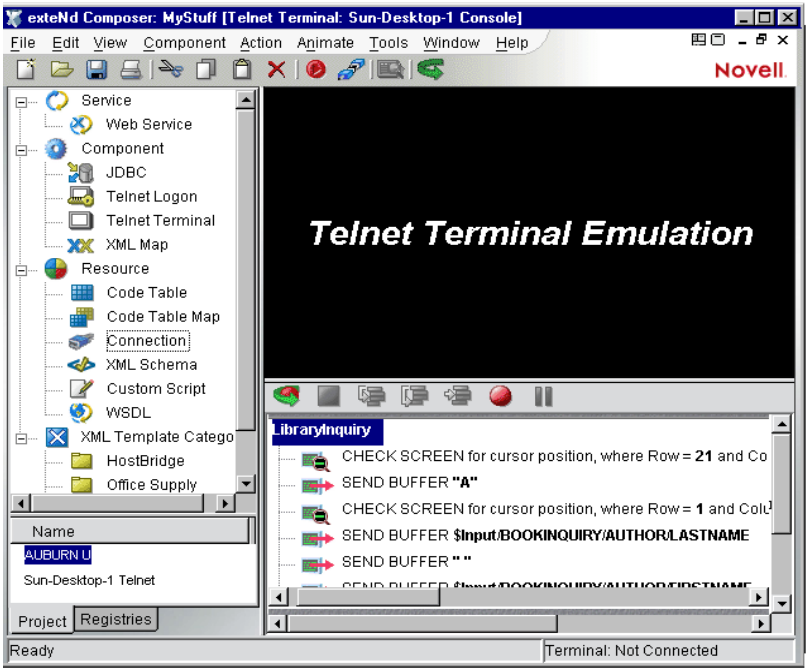
- アクションモデルで、[Cut]、[Copy]、または [Paste]、あるいはこれらすべてを使用してアクションを削除、移動、または複製する際には、細心の注意を払います。「記録」セッション中に自動的に作成されたアクションでは、編集処理で簡単に見落とされるデータ従属性が頻繁に作成されます。
- ドラッグアンドドロップを使用して、アクションモデルに新しい Map アクションを追加する必要がある場合は、アクションペインのツールバーで [Start Animation] ボタンをクリックし、アクションモデルの目的の行に進んでから、アニメーションを一時停止して記録モードをオンにします。この時点では、安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

既存のアクションの変更

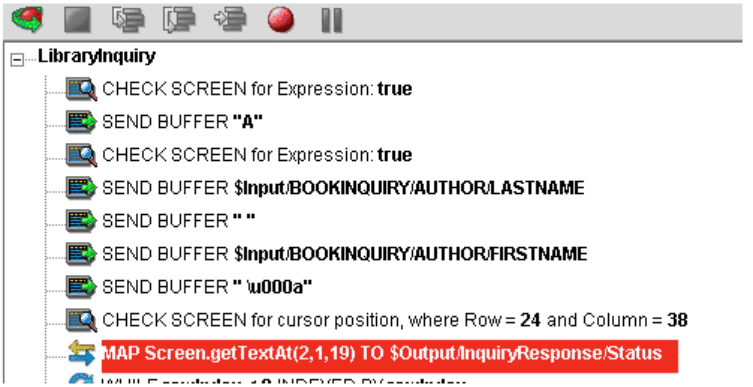
次の手順では、以前に記録したセッションで既存のアクションを変更する方法を説明します。

➢ 以前に記録されたアクションモデルで、既存のアクションを変更する

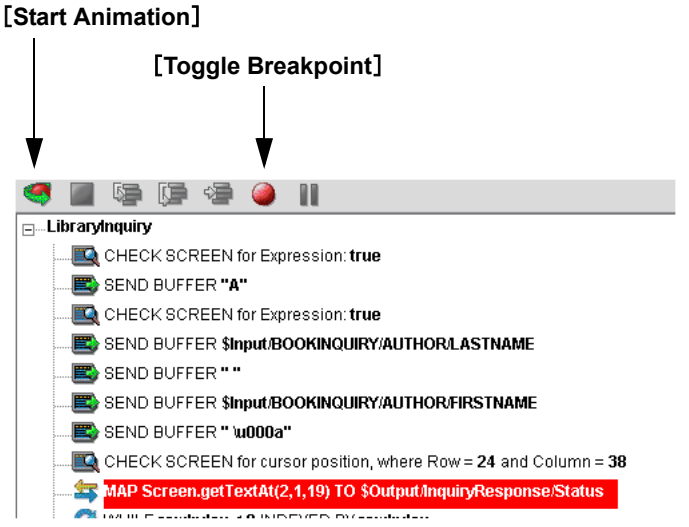
1 編集するアクションモデルを含むコンポーネントを開きます。Telnet コンポーネントエディタウィンドウにコンポーネントが表示されます。



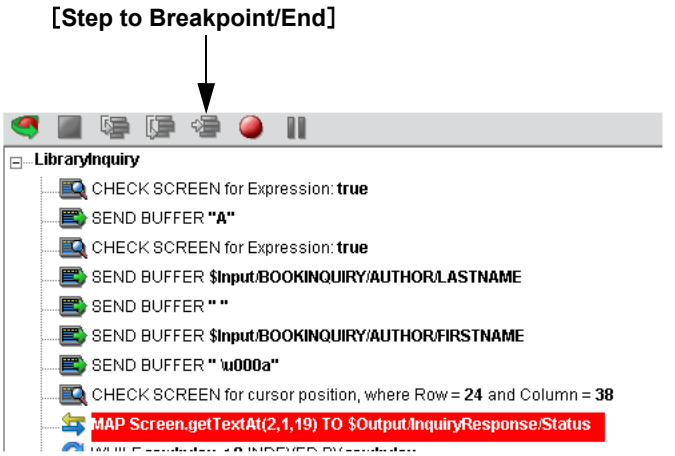
2 編集を行うアクションモデルでアクションに移動し、アクションを選択します。



3 ［**Toggle Breakpoint**］ボタンをクリックします (または、<**F2**> を押します)。選択したアクションが赤色になります。

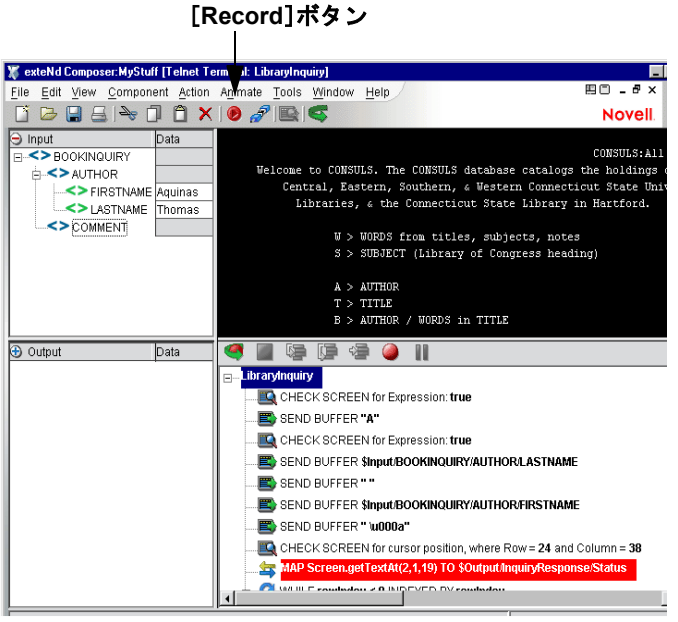


4 ［**Start Animation**］ボタンをクリックします。(アクションペインのツールバーで) アニメーションツールが有効になります。



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 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。



7 アクションモデルに加える追加のドラッグアンドドロップ(または他の操作)を行います。

8 記録をオフにします(［**Record**］ボタンを切り替えます)。

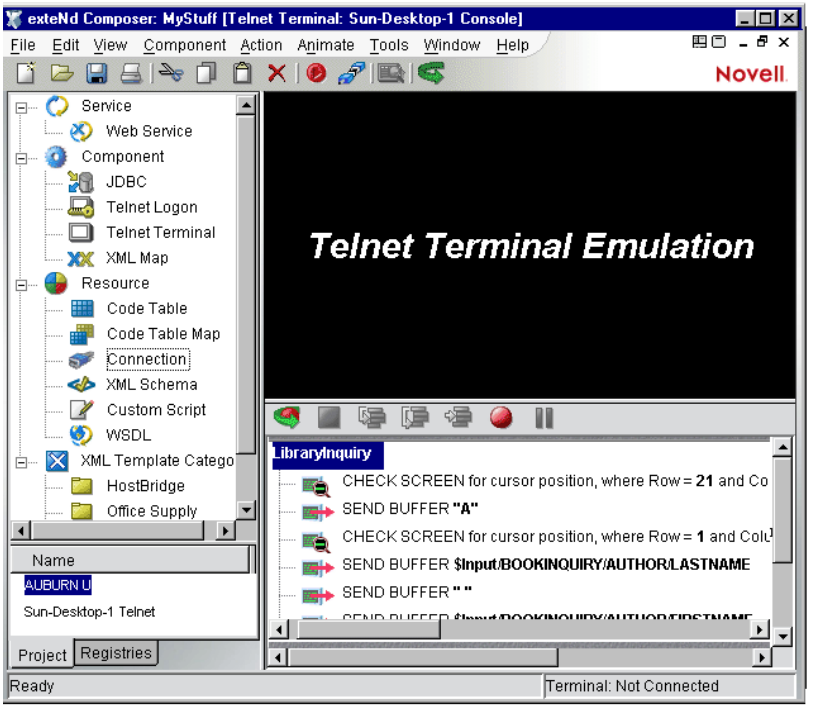
9 コンポーネントをテストします。

新しいアクションの追加

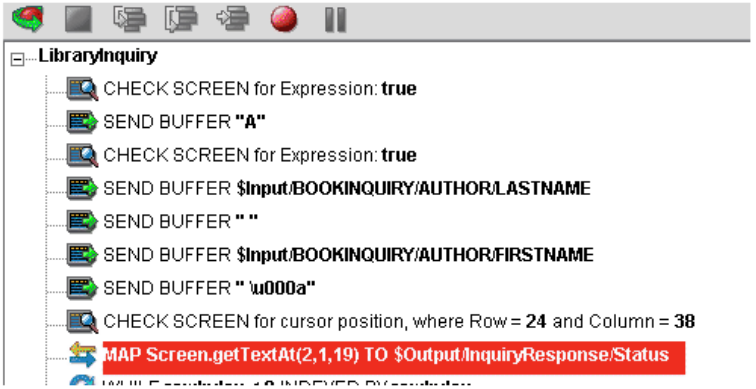
次の手順では、以前に記録したセッションで新しいアクションを追加する方法を説明します。

➢ 新しいアクションを以前に記録したアクションモデルに追加する

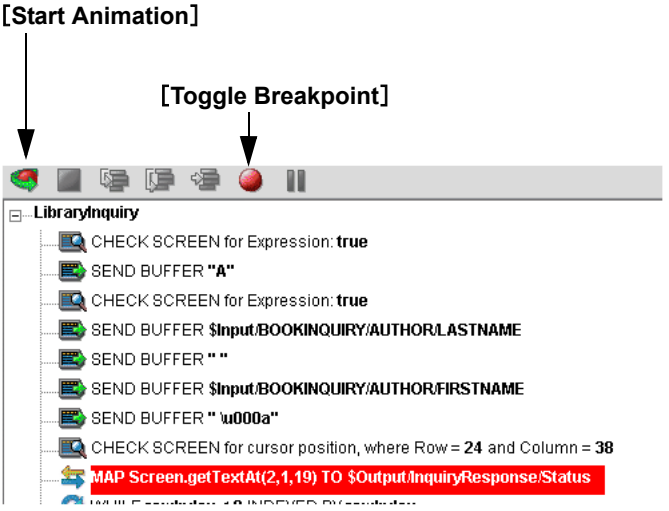
1 アクションを追加するアクションモデルを含むコンポーネントを開きます。Telnet コンポーネントエディタウィンドウにコンポーネントが表示されます。



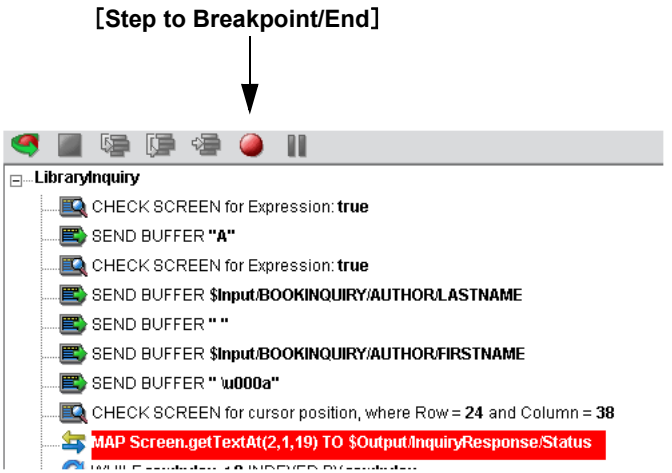
2 アクションモデルで追加を行うアクションに移動し、アクションを選択します。



3 ［**Toggle Breakpoint**］ボタンをクリックします (または、<**F2**> を押します)。選択したアクションが赤色になります。

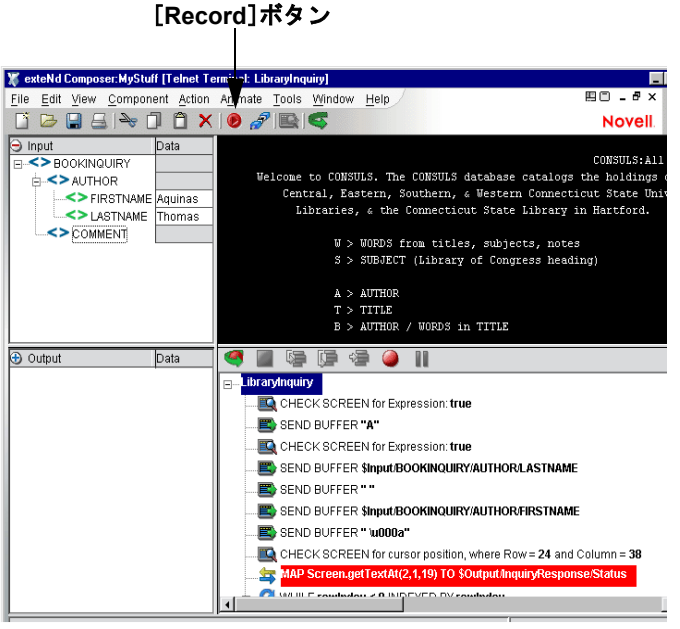


4 ［**Start Animation**］ボタンをクリックします。(アクションペインのツールバーで) アニメーションツールが有効になります。



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 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。



7 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。

8 記録をオフにします (［**Record**］ボタンを切り替えます)。

9 コンポーネントをテストします。

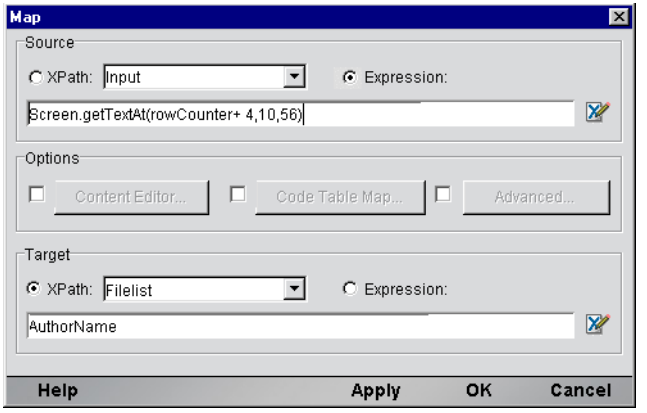
Alias アクションの追加について

別名であるループに Map アクションを追加している場合、次の操作を実行します。

➢ **Alias アクションを以前に記録したアクションモデルに追加する**

1 コンポーネントを開きます。

2 ［**Action**］メニューから、［**New Action/Advanced**］、［**Map**］の順に選択します。［Map］ダイアログボックスが表示されます。



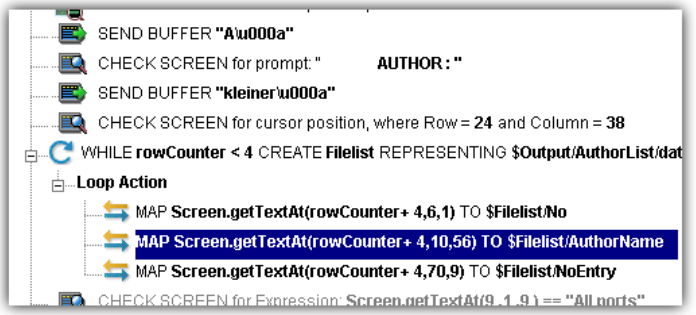
3 ［Source］で［Expression］をオンにすると、ドロップダウンボックスがグレー表示になります。

4 情報を入力するか、［**Expression Builder**］ボタンをクリックして新しい式を作成します。

5 別名によって表される XPath を作成します。ドロップダウンリストから別名をクリックします。

6 ［**OK**］をクリックします。

7 選択した行の下に新しいアクションが挿入されます (次の画面では、新しい行がハイライトされ、挿入されたことが示されています)。

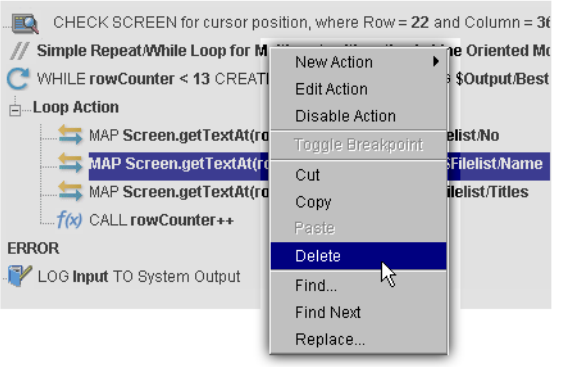


アクションの削除

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を説明します。

➢ **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、マウスを右クリックし、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。

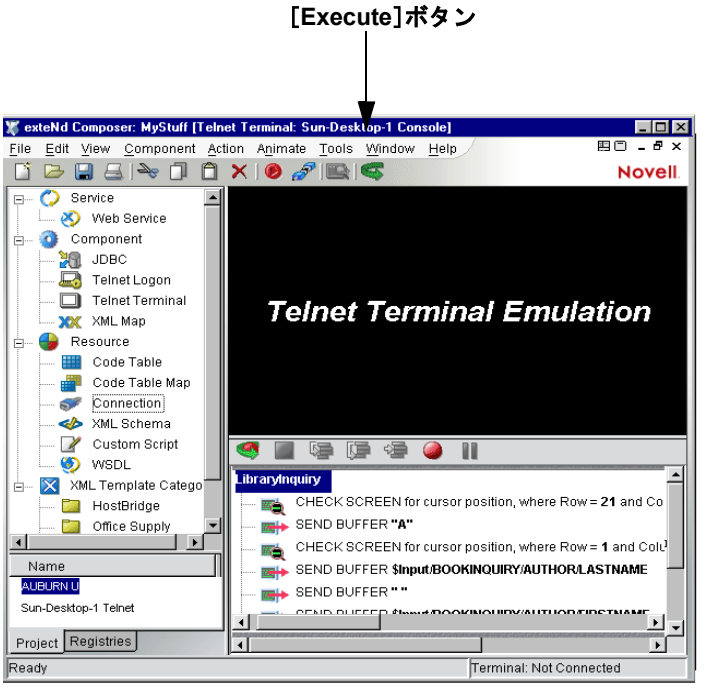


Telnet コンポーネントのテスト

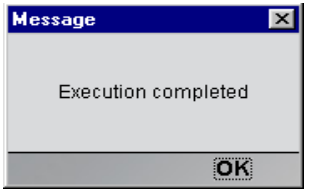
Composer には、コンポーネントを簡単にテストできるアニメーションツールが含まれています。Telnet コンポーネントエディタのツールバーには、[Execute] ボタンがあり、このボタンを使用するとアクションモデル全体を実行して、コンポーネントが意図したように動作するか検証できます。新規に作成した Telnet コンポーネントをテストして、すべての Check Screen アクションでのタイムアウト値が適切であり、Send Buffer アクションおよび他のアクションが意図したように動作するか確認することが大切です。

Telnet コンポーネントを実行する

1 Telnet コンポーネントを開きます。Telnet 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アクションが実行されます。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、意図したようにすべてのデータが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。

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。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。

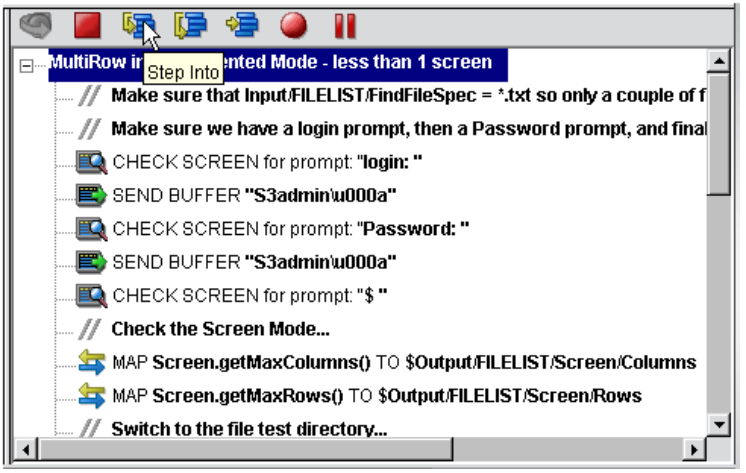
➢ アニメーションツールを使用して Telnet コンポーネントを実行する

1 Telnet コンポーネントを開きます。Telnet 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ためには、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、またはキーボードの <**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。

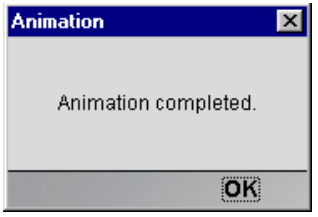


4 ［**Step Into**］ボタンをもう一度クリックします。Check Screen アクション (前を参照) が実行され、次のアクションが選択されます。

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 実行が完了すると、次のメッセージが表示されます。



信頼性の高い Telnet 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 Telnet コンポーネントを作成する上で役に立ちます。

- 常に、Send Buffer アクションの後に Check Screen アクションを続けます。
- Check Screen アクションでは、絶対カーソル位置が指定の画面に対して常に一定であると確信している場合に限り、(カーソル位置に基づく) デフォルトの実行許可条件を受け入れます。多くの場合、カスタム式を記述する方がより安全です。
- 記録中にプロンプトに基づく Check Screen アクションを作成する迅速で、正確な方法は、カーソル直前にある (カーソル位置まで、カーソル位置は含まない) 該当する文字をハイライト (選択) してから、マウスを右クリックして［Check Screen］を選択することです。これにより、ハイライトしたプロンプトに基づき、Check Screen アクションが自動的に作成されます。
- ［Prompt］(［Check Screen］ダイアログボックス内) でカスタムプロンプト文字列を入力する場合は、プロンプト文字列内に表示される引用符を忘れずにエスケープしてください。
- 日付、時刻などの変数情報に基づく Check Screen 実行許可条件は使用しないようにします。
- Min Wait 設定を使用して、指定の期間待機する以外には何も実行されない Check Screens は、使用しないようにします。この方法は、適切に動作するように思われますが、重大なパフォーマンスのボトルネックが発生する場合があります。
- Check Screen アクションで使用するデフォルトのタイムアウト値は、設計セッション中に実際の応答時間から計算されます。この結果、いくらかの影響が生じます。まず、デフォルトのタイムアウト値は、ロード関連のアプリケーションに対して大きくしなければならない場合があります。次に、Check Screen アクションを削除すると、それ以降の実行で同期がタイムアウトになる可能性があります。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。
- 塗り直し中に画面の途中は一定のままだが、最初と最後の行が変更する場合など、不完全な実行許可条件が有効になると、2 つの Check Screen アクションを作成してから、それらを式に基づく 1 つのアクションに結合する必要があります。

Telnet コンポーネントエディタでの他のアクションの使用

Check Screen アクションおよび Send Buffer アクションに加えて、Telnet コンポーネントエディタでは、その他すべてのアクションを使用できます。［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 4-1

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 を指定します。
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。
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	コンポーネントに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	要素のデータをある 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 またはその他 (Decision アクション) のチェーンを排除するときに使用できる、基本的に便利なアクションです。

表 4-2 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 and Repeat**］の順にサブメニューを選択すると利用できます。

表 4-2

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 となる場合は、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。
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 4-3

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。

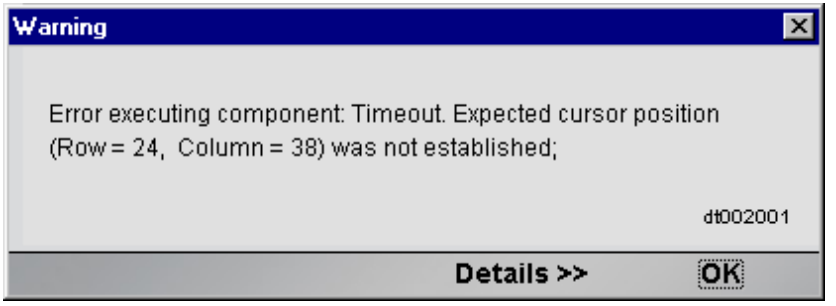
Data Exchange アクション	説明
XML Interchange	外部 XML ドキュメントをコンポーネントの DOM に読み込んだり、外部 XML ドキュメントにコンポーネントの DOM を書き込んだりします。読み込み / 書き込みメソッドには、ファイル、FTP、HTTP、および HTTPS プロトコルを使用した Get、Put、Post、および Post with Response が含まれます。

表 4-4

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

Telnet コンポーネントをテストする際には、Check Screen アクションまたは Send Buffer アクション、あるいはその両方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。次に説明する Check Screen エラーは、すべてタイムアウトエラーであることを認識しておくことが大切です。次に説明するエラーのいずれかが発生した場合、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、「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。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タイムアウト値を大きくすることです)。タイムアウト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。

次に、一般的なエラーメッセージおよびその意味について説明します。

「Expected cursor position (Row = {0}, Column = {1}) was not established」

このエラーメッセージは、タイムアウト時間が切れる際にカーソルが予期される位置になかったため、Check Screen が失敗したことを意味します。おそらく、予期しない何らかの方法でホストアプリケーションが変更されたか、プロンプト行が動的に変更されているなどです。また、ホストの負荷が重い、または接続不良といった理由から、Check Screen が単純に「タイムアウトになった」という可能性もあります (前の説明を参照)。指定の Check Screen アクションに対して、タイムアウト値を大きくします。それでも解決しない場合 (または、問題が不適切な実行許可条件の選択に関すると考えられる場合) は、固定したカーソルの座標以外の条件に基づき、Check Screen 実行許可条件を再度記述するようにします。たとえば、プロンプト文字列を指定したり、式を使用して何らかの方法で画面のコンテンツを検証します。

「Expected prompt text {0} was not established」

このエラーメッセージは、タイムアウト時間が切れる前にプロンプトが指定の (予期される) プロンプト文字列と同一でなかったため、Check Screen が失敗したことを意味します。プロンプト行が、予期しない何らかの方法で動的に変更されている可能性があります。または、負荷が重いあるいは他の要因によって、単純に、ホストの応答時間が予期せず長くなった可能性もあります (前の説明を参照)。ホストの待ち時間が問題であると考えられる場合、Check Screen アクションのタイムアウト値を大きくします。それ以外の場合は、Check Screen 実行許可条件を再度記述して、ハードコード化されたプロンプト値以外の条件に基づかせます。たとえば、何らかの方法でプロンプトを検証する式を指定します。

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

このエラーメッセージは、Check Screen 実行許可条件が ECMAScript 式に基づき、実行時に式が *false* を返し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 アクションのタイムアウト値を大きくするか (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。

Send Buffer エラー

文字列「\u001b[?1;2c」(または、これに類似した文字列) を含むアクションモデル上部に、自動生成された Send Buffer アクションが表示された場合、Telnet 接続リソースの設定ダイアログボックスの［Terminal Type］フィールドで端末タイプが指定されていない (または、おそらく認識できない端末タイプを指定した) ことを意味します。

一般的に、Send Buffer エラーの発生はまれです。ただし、Send Buffer には複数画面に渡るコマンド (「先読み入力」バッファと呼ばれる) が含まれるため、ご注意ください。そのようなアクションは、簡単に誤って作成されます。オーバーロードした Send Buffer を使用したアクションモデルは、アニメーション時にアクションを処理する際には正常に動作するかもしれませんが、コンポーネント全体の実行時には、画面での同期問題のため、失敗する場合があります。ここで問題を回避する方法は、Send Buffer アクションすべてに対して、常に対応する Check Screen アクションがあるか確認することです。

接続プールを使用し、不正なユーザ ID またはパスワードを使用してログオンしようとした場合、接続インスタンスは使用できなくなり、プールのメンバーは次の接続要求でスキップされます。「Logon connection in pool <Pool name> was discarded for User ID <User ID>」というエラーメッセージがサーバログに送信されます。運用前のテスト中またはパフォーマンスの問題が生じた場合、あるいはその両方で、この種のメッセージを確認する必要があります。

「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 Check Screen アクションおよび Send Buffer 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、「Expected」の後のテキストを選択して、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 にあたる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)。

5 高度な Telnet アクション

Telnet ベースの計算は、他の計算 (他の端末ベースの処理を含む) と次のような多くの重要な点で異なります。

- データはまとめてではなく、一度に 1 文字ずつ着信します。
- 着信データに特定の構造はなく、データの着信順序が一定でない可能性があります。
- 画面の更新内容は、画面の一部だけ (1 文字だけ) または画面全体の場合があります。
- データセットの取得に、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、「ページを進める」コマンドを複数回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) Telnet 処理の自動化は困難になります。この章の目的は、eXtend アクションモデルに関連する一般的な Telnet 計算において、(問題となる可能性がある) 状況の対処方法をいくつか提案することです。

この章の内容をよく理解するには、第 4 章「Telnet アクションの実行」をすでにお読みになって、(Repeat While アクションを利用したループなど) アクションモデルプログラミングの構成に慣れていることが必要です。また、ECMAScript の使用経験も多少必要になります。

画面にまたがるデータセット

Telnet 計算では、よく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。「4 ページの 1 ページ目」のような行が画面に含まれる場合は、(「Telnet 専用の Expression Builder 拡張」ですでに説明した ECMAScript 画面 - オブジェクトメソッドの 1 つを使用して) この行が発生する場所で画面を検査したり、使用されているすべての画面で反復するループを構成したりすることは簡単です。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。そのような場合、判断の方法は、(たとえば) 画面の上下に表示されていて最後の画面になると［Back］(またはその他のメッセージでは［End］) に変わる［More］コマンドの存在だけの可能性があります。その他、「レコード」の数の合計がわかっていて、「1 画面あたり」に表示されるレコードの数を (見た目で) 判断でき、それによって情報が含まれる画面の合計を計算できる場合もあります。

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 Telnet アプリケーションの実装により異なります。アプリケーションによっては、空白のレコードが表示されるまで画面を繰り返す必要があるものや、同じ方法が適当でないものがあります。

このうち 2 つの条件を組み合わせたアクションモデルの例について後ほど詳しく説明します。

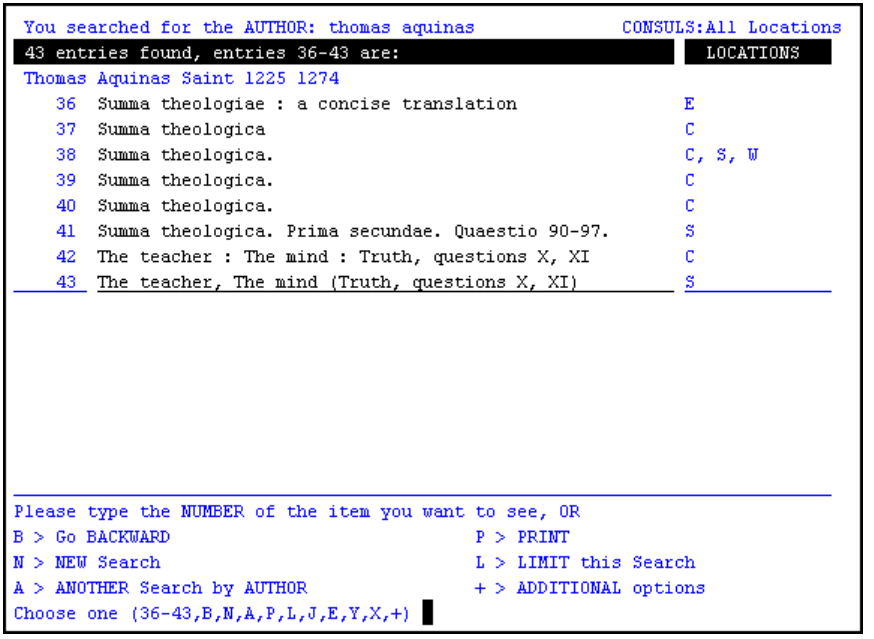
余分なデータの処理

Telnet 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。全画面の表示がそうして維持されています。

次の 2 つのスクリーンショットについて考えます。上のスクリーンショットには、6 画面にわたる情報を返すクエリの最後から 2 番目の画面に当たる情報が表示されています。黒くハイライトされているステータス行 (上から 2 行目) に「43 entries found, entries 33–40 are:」とあり、続けて行のエントリが表示されていることに注目してください。合計のデータセットが 43 レコードあり、最後から 2 番目の画面が 40 番目のレコードで終わっているため、次の (最後の) 画面にはレコード 41 からが表示されると予想できます。最後の画面はその次のスクリーンショットに表示されているような画面になります。画面にはレコード 36 から 43 が表示されています。つまり、「前の」画面から 5 つのレコード (36 から 40) が含まれています。多くの場合、この余分なデータはキャプチャしないようにします。問題は、このような余分なデータを検出、削除する方法です。

ECMAScript には、重複がないようにリストを維持する簡単で便利な方法が備えられています。コツは、生の (初期化されていない) オブジェクトを作成して、プロパティとしてレコード名を添付することです。オブジェクトについて、同一の名前で 2 つのプロパティを持つことはできないため、プロパティ名としてレコード名を割り当てると、そのオブジェクトのプロパティリストが重複のないレコード名のリストになります。

```
You searched for the AUTHOR: thomas aquinas              CONSULS:All Locations
43 entries found, entries 33-40 are:                      LOCATIONS
Thomas Aquinas Saint 1225 1274
   33  Summa contra gentiles.                                      W
   34  Summa contra gentiles.                                      E
   35  The Summa contra gentiles of Saint Thomas Aquinas,          S
   36  Summa theologiae : a concise translation                    E
   37  Summa theologica                                            C
   38  Summa theologica.                                           C, S, W
   39  Summa theologica.                                           C
   40  Summa theologica.                                           C

Please type the NUMBER of the item you want to see, OR
F > Go FORWARD       A > ANOTHER Search by AUTHOR      + > ADDITIONAL options
B > Go BACKWARD      P > PRINT
N > NEW Search       L > LIMIT this Search
Choose one (33-40,F,B,N,A,P,L,J,E,Y,X,+)
```



簡単な例を使用すると理解しやすくなります。いくつかの項目が複数回リストに表示されている項目の配列があるとします。

```
var myArray = new Array( "Tom","Amy","Greg","Tom","Amy");
```

この配列の重複する項目を削除するために、生のオブジェクトにプロパティを割り当てることができます (プロパティ名は配列の値と等しくなります)。

```
var myObject = new Object(); // create a bare object
for (var i = 0; i < myArray.length; i++) // loop over array
{
        var arrayMember = myArray[ i ]; // fetch array member
        myObject[ arrayMember ] = true; // create the property
}

 // Now obtain all property names
 // in a new, unduplicated array:
```

```
var uniqueValues = new Array();
var n = 0; // counter
for (var propertyName in myObject) // enumerate property names
       uniqueValues[ n++ ] = propertyName;

// Now 'uniqueValues' contains just "Tom","Amy","Greg"
```

このコツを、後で説明する Telnet アプリケーションの例で活用します。

複数の画面でループする例

これまで取り上げてきた複数の条件を組み合わせる Telnet コンポーネントの例について考えます。ホストアプリケーションは、ある大学の図書館システムで本を検索するサービスです。この例では、著者名を指定する入力ドキュメントを使用します。その名前を基に、著者名によって利用可能な本のタイトルすべてのライブラリを照会し、出力 DOM への結果をキャプチャします。出力ドキュメントには、タイトルに重複がないリストが含まれるようにします。

この例では次の内容について説明します。

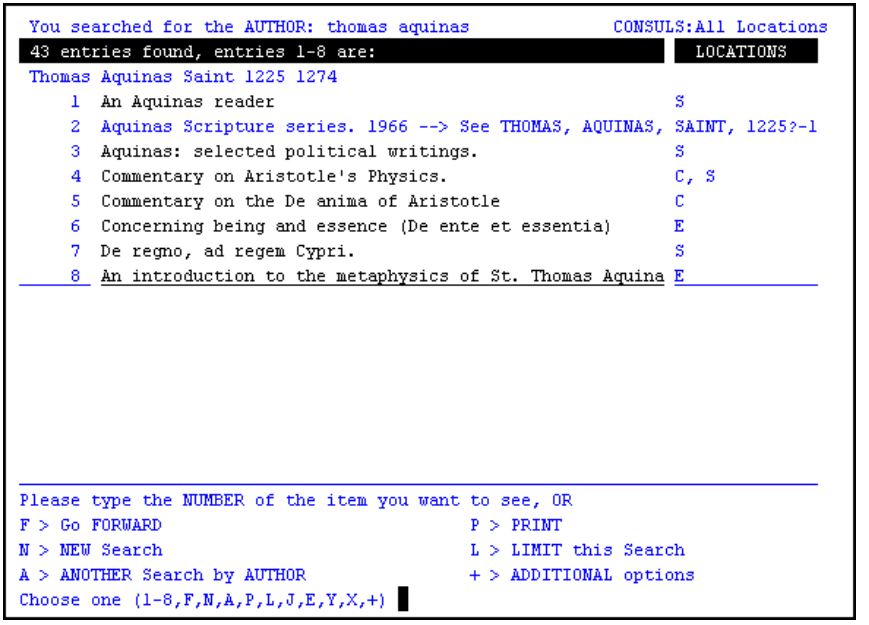
- 事前に画面の数がわからない場合に複数の画面からデータを「スクレーピング」する方法
- 発生したレコードの重複を防ぐ方法
- 出力 DOM のノードをプログラムによって作成する方法
- 空白のレコードまたは最後の画面に到達した場合に、メインループを終了する

アクションモデルのメインループに関する論理は、次のようにまとめることができます (擬似コード)。

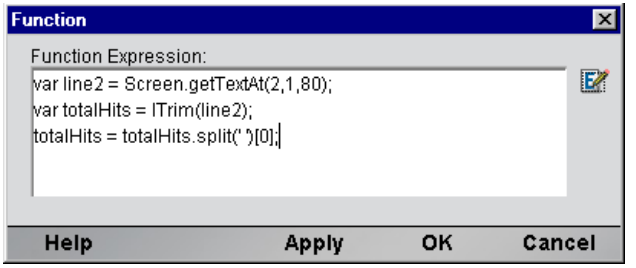
```
Determine the number of records-per-screen
While (true) // enter a "forever" loop
    Fetch a record
┌   IF Record is Valid   // i.e., not blank
|        Write data to Output DOM
|        IF Screen has been completely processed
|            ┌ IF this is not the final screen
|            |          Fetch next screen
|            └ ELSE BREAK   // final screen processed
└   ELSE BREAK     // blank record reached
```

この例のアクションモデルにおける最初の部分は、前の例 (「Telnet アクション」の章) の「Telnet セッションの記録」で作成したアクションとほぼ同様です。ただし、この例では著者名として Thomas Aquinas を使用する点が例外です。初期アクションは、「Thomas Aquinas」という著者名検索を実行するために必要な Check Screen アクションおよび Send Buffer アクションのみです。

検索結果の初期画面は次のようになります。



2 行目の初めに、見つかったレコード数 (エントリ) が表示されています。Function アクションを使用すると、この情報をキャプチャできます。



この 3 行のスクリプトにより、ローカル変数 line2 にある 2 行目がすべて取得され、行頭のスペースは削除され、スペース文字がある行は分割されます (結果配列の 0 番目のメンバーを変数 totalHits にキャプチャします)。それから、Map アクションを使用して「total hits (ヒット総数)」の数を出力 DOM に書き込みます。

この時点で、「ヒット総数」の数をメインループの基本として使用できます。ただし、「ヒット総数」の情報がすべての Telnet ホストによって最初の応答画面に報告されるわけではないため、図の問題からこの段階は省略します。しかし、この特定のアプリケーションにより 1 画面あたりのレコード数 (2 行目) が報告される機能は活用します。繰り返しになりますが、便利な ECMAScript のプログラミングを使用して、1 画面あたりのレコード数に関する情報をランタイム時に動的に取得することも可能です。また、画面を目で調べた後、この値をハードコード化することもできます。

注記： そのうち、1 画面あたりのレコード数をハードコード化するか、ランタイム論理を適用するか判断が必要になります。Telnet アプリケーションでは、重要な画面の特徴を 1 つ 1 つ動的に判断することは困難です。ホストアプリケーションの動作に関する事前の知識の中には、最終的なアクションモデルでほぼ絶対的なものもあります。

ECMAScript 変数 booksPerScreen に、1 画面あたりのレコード数を保存します。この例では、1 画面あたり 8 つのレコードがあります。

メインループの設定

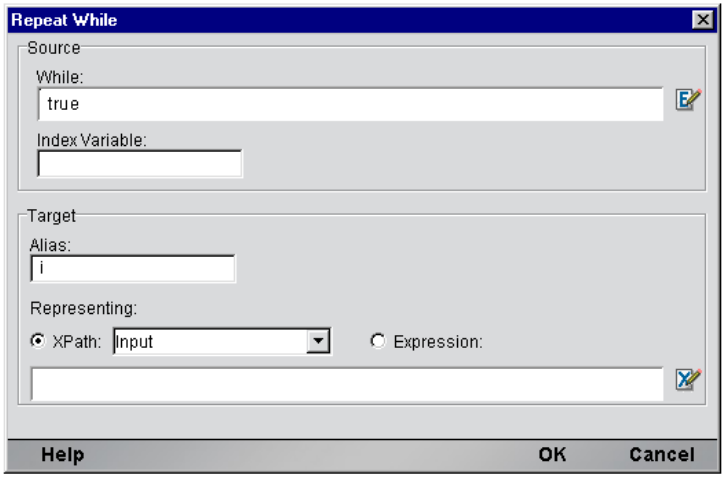
メインループを作成する前に、出力 DOM でノードを作成する際に使用するインデックス変数を設定する必要があります。このインデックス (bookNumber) は、*1* から始まり出力に本のタイトルをキャプチャするたびに 1 増加します。このインデックスが *0* ではなく *1* から始まる理由は、DOM ノードでは 1 からインデックス処理が行われるためです。bookNumber を使用して、ノードにインデックスを作成します。

また、(Function アクションで) ECMAScript 式を使用して、空白の ECMAScript オブジェクトを作成します。

```
var bookTable = new Object();
```

すでに説明したように、本のタイトルをこのオブジェクトのプロパティ名として保存することにより、レコードのリストを「重複なく」保存できます (「余分なデータの処理」を参照)。

ループを作成するには、アクションモデルで Repeat While アクションを使用します (マウスを右クリックして［**New Action**］>［**Advanced**］>［**Repeat While**］の順に選択します)。ダイアログボックスの設定は、次のようになります。



［While］条件を *true* に設定すると、実質的には無限のループを作成することになります。ループの終了条件は 2 重になっています。

- 空白のレコード (すべてのスペース文字) が見つかった場合、ループは終了します。
- 現在の画面が前の画面と同一の場合、ループは終了します。

後の条件は、無限ループを終了するための適切かつ安定した方法で、さらに、ライブラリ - クエリアプリケーションだけでなく、通常幅広い Telnet アプリケーションに適用できます。

インデックス変数 `i` は、ゼロから `booksPerScreen - 1` のサイクルで変化するもので、2 つの役割があります。

1. 新しい画面を取得する場合 (つまり、値が `booksPerScreen - 1` になった場合) に通知する
2. レコードを取得する際に「行オフセット」の基礎となる

画面のキャッシュ

ループ前に追加の設定コードを使用して、現在画面をキャッシュに保存します。ループが開始される前すぐに、次の Function アクションステートメントを含めます。

```
previousScreen = Screen.getTextAt(1,1,Screen.getColumnCount() *
Screen.getRowCount());
```

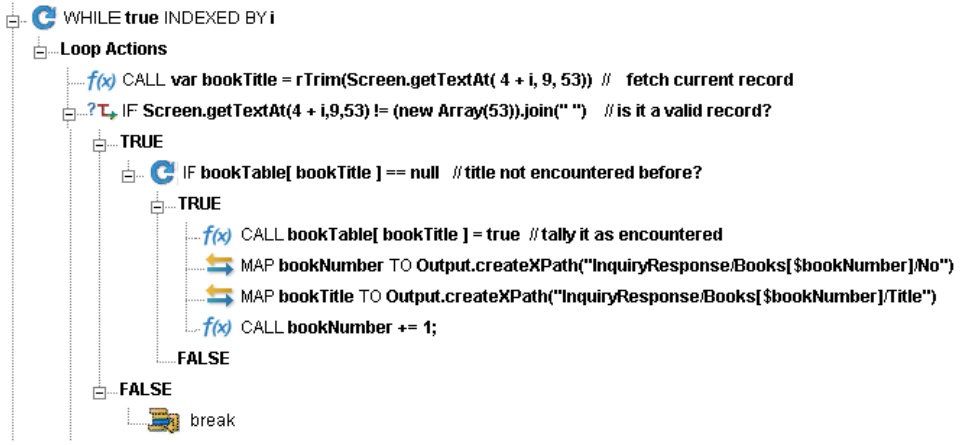
変数 previousScreen により、最後に表示された画面のコンテンツがキャッシュされるため、新たに取得された画面を確認することができます。新たに取得された画面に、処理された画面と同一のコンテンツが含まれる場合、最後の画面に到達したこと (そしてループを終了する必要があること) がわかります。

メインループ

ここでは、アクションモデルのメインループにより実際に行われることを理解します。

前半部分

ループの最初の部分を次の図に示します。図の内容は、実際の動作とほぼ同様です。



ループ内の最初のアクションは Function アクションで、列 9 の行 `4 + i` から始まる 53 文字を取得します。目的の行は 4 から 11 まで (11 を含む) で、これはホストによって品目が報告されるゾーンです。`i` はゼロから 7 まで変化するため、コードで「4 + i」を行オフセットとして使用できます。

レコードの取得後、処理を実行する前に検証を行います。レコードが取得されるゾーンが空白でない場合に限り、ループが続行されます。次の決定式を含む Decision アクションを使用します。

```
Screen.getTextAt( 4 + i, 9, 53) != (new Array(53) ).join(" ")
```

式の右側のステートメントは、「新しい、53 文字長の空白の配列を作成し、区切り文字としてスペース文字 1 文字を使用し配列のメンバーをともに結合することによって文字列に変換する」という意味です。各配列のメンバーは null であるため、実質的にはこれにより行に 53 のスペース文字を含む文字列が作成されます。この文字列と画面の文字列を比較して、空白のレコードが見つかったかどうか判断できます。

Decision アクションの TRUE 分岐では、取得した本のタイトルがすでに見つかっているものかどうか即時に確認します (重複を避けます)。本のタイトルは bookTable オブジェクトにプロパティ名として保存する方法を使用しているため (前の説明を参照)、次の式に対して Decision アクションを実行するだけで既出の本かどうか確認できます。

```
bookTable[ bookTitle ] == null
```

このステートメントが true の場合、bookTable オブジェクトには bookTitle の文字列に一致するプロパティ名がない、ということになります。その場合は、先に進んでマップ操作を実行できます (そうでない場合は、不成立のため動作が繰り返されます)。

この判断に関する TRUE 分岐では、bookTable[bookTitle] に *true* とマークを付けます。bookTable に新しい null でないプロパティが割り当てられます。次に、インデックス番号および本のタイトルを出力 DOM の新しいノードにマップします。次の式を適用することにより、マップできます。

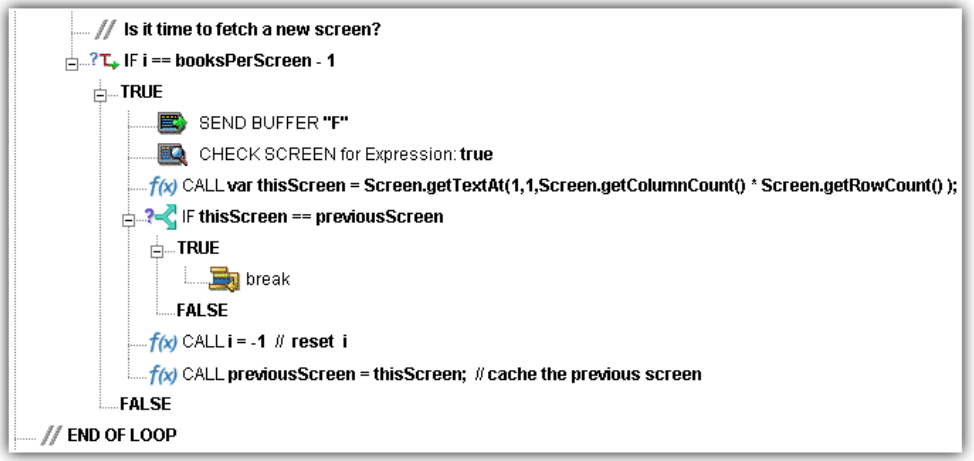
```
Output.createXPath("InquiryResponse/Books[$bookNumber]/Title")
```

マップの際、bookNumber で作成中のインデックスを使用して、**InquiryResponse/Books** に **Title** という要素名を持つ新しいノードインスタンスを作成することができます。

これで、bookNumber が増えます。

後半部分

ループの後半部分では、新しい画面を取得するかどうかを確認します。取得する場合は、[Send Buffer] コマンドを実行して、ホストに対しページを次の画面に進めるよう命令します。

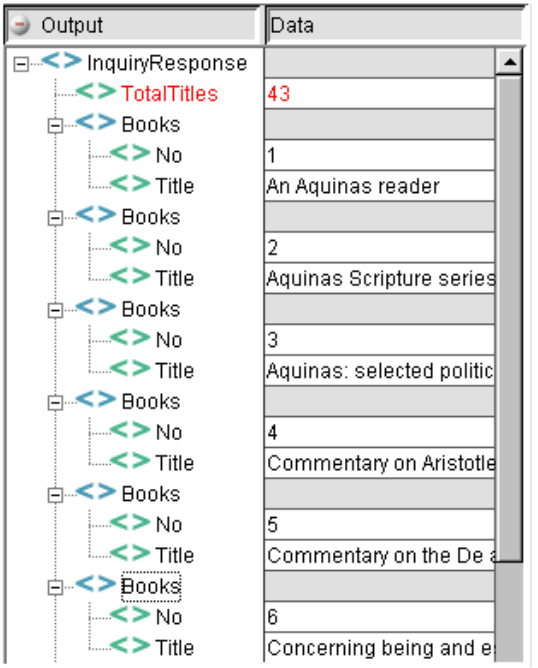


新しい画面が取得されると、すぐにそのコンテンツを文字列変数 `thisScreen` にキャプチャします。そして、`thisScreen` と `previousScreen` を比較する Decision アクションを実行します。2 つの値が等しい場合は、Break アクションを使用してループを終了します。等しくない場合は、式が不成立のため実行を続けます。

注記： 図に表示されている Check Screen アクションに対する Min Wait 時間の設定に注意してください。Min Wait 時間が短く実行許可条件が true の場合、画面を誤ってスキップし、本来よりも早くループを終了してしまう可能性があります。

実行中の場合は、`i` (行のインデックス変数) をリセットし、次のラウンドの準備で `previousScreen` に `thisScreen` を入力します。

ループが終了すると、出力 DOM は次の図ように表示されます。



DOM に、この著者に対して見つかったすべてのタイトルが順番に表示されます。最後の画面のデータに、以前の画面と重複する情報がかなり含まれている場合でも、DOM には重複するタイトルは含まれません。

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ミリ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 一時DOMの要素に`endTime - startTime`をマッピングする**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アクションに使用されていることがわかります(150 ミリ秒以下で実行される Check Screen アクションはほとんどありません)。そこで、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 アクションの Min Wait 値および Timeout 値の調整に左右されます。

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、次の章の「ログオンコンポーネント」をお読みください。

6 ログオンコンポーネント、接続、および接続プール

Telnet セッションパフォーマンスについて

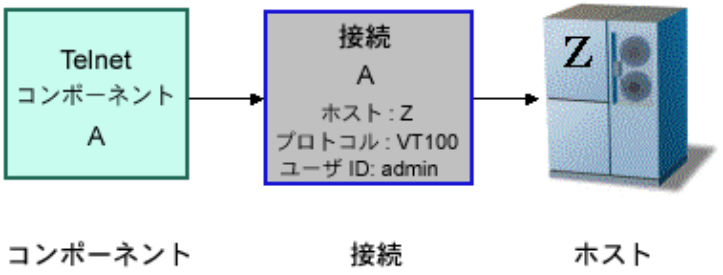
通常の Telnet コンポーネントは、exteNd Composer 内のテストワークステーションで適切に動作しますが、運用アプリケーションサーバ環境に配備すると、このコンポーネントをカプセル化するサービスの動作が負荷により遅くなる場合があります。これは、珍しいことではなく、複数のユーザをサポートしたデータベースシステムで過去によく見られた問題に共通しています。この問題では、通常標準的な端末セッションプログラムで必要となる手順に要した時間を追跡することができます。プログラムの実際の実行を除くと、この手順は次のようになります。

1. ホストへの接続を安全にする
2. ユーザを認証する
3. メニューシステムを使用して、トランザクションを起動できるポイントへ移動する
4. トランザクション終了時に、ユーザをサインオフし、接続を切断する

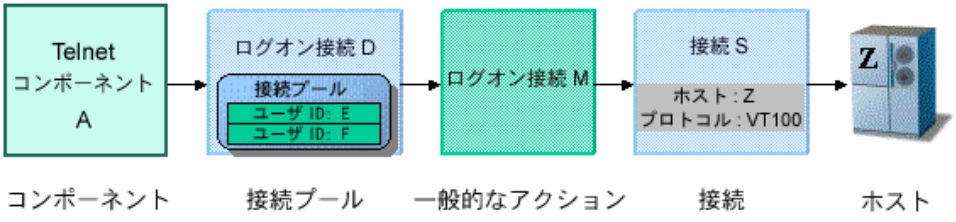
個々のプログラムに対する負荷は小さい場合でも、多数の Web サイトまたはアプリケーションサーバ環境、あるいはその両方で一般的にみられるように、プログラムの負荷が重くなると Telnet コンポーネントとセッションのオーバーヘッドの 1 対 1 の関係に問題が生じる場合があります。exteNd Composer では、2 つの特別なオブジェクトを用意することで、繰り返しセッションのオーバーヘッドを最小限に抑えています。このオブジェクトは、「Telnet ログオン接続」、およびこれに対応する「Telnet ログオンコンポーネント」と呼ばれる特別な接続リソースタイプです。

接続プールアーキテクチャ

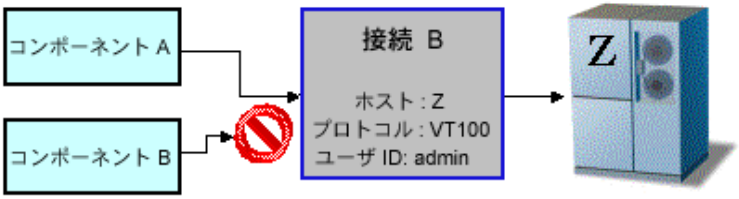
Telnet 用 Connect をインストールすると、Telnet 接続および Telnet ログオン接続 (これ以降、ログオン接続と呼びます) という 2 種類の接続リソースが接続の作成ウィザードに追加されます。Telnet 接続とは、実際の「接続」で、Telnet コンポーネントでの使用時にホストシステムとのセッションを確立できます。



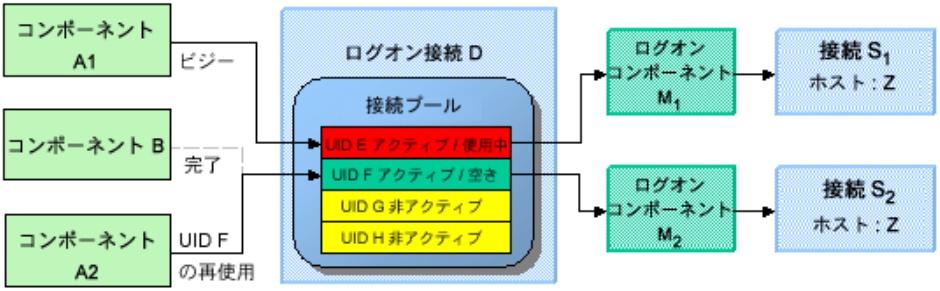
逆に、ログオン接続は Telnet 接続とは異なり、使用可能なユーザ ID のプールを定義し、Telnet ログオンコンポーネント (これ以降、ログオンコンポーネントと呼びます) を使用し、個々のユーザ ID に対して接続およびナビゲーションに関連するアクションを実行します。実際には、ログオンコンポーネントにより、通常の Telnet パスワードを使用して接続が確立されます。ログオンコンポーネントについては、後に説明しますが、接続プールを確立するには、ログオン接続とログオンコンポーネントを同時に使用する必要があることに注意してください。



通常、Telnet コンポーネントにより、単一のユーザ ID およびパスワードを使用して定義した接続がアクティブになると、その接続のユーザ ID/ パスワードは、同じ接続定義を使用するコンポーネントの別のインスタンス、または別のコンポーネントでは使用できなくなります。



ログオン接続では、追加のユーザ ID を作成して新しい接続を確立することで、他のコンポーネントが終了するまでのシリアル待機時間を減らし、さらに可能な場合は接続を再使用してセッションのオーバーヘッドを回避することで、パフォーマンス上の利点が提供されます。

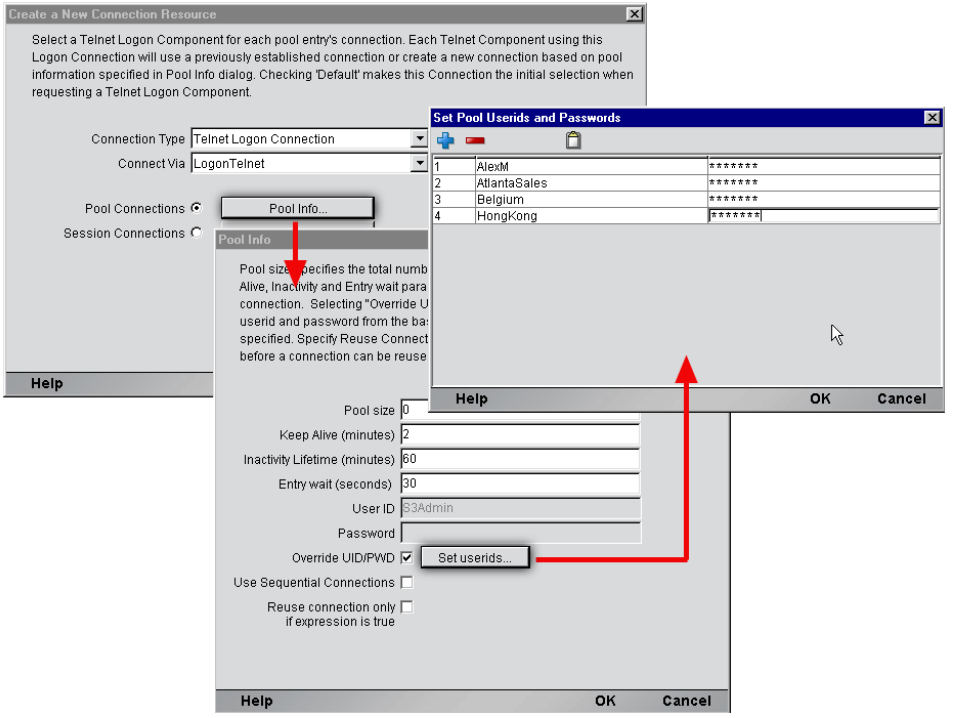


上の図では、アクティブな Telnet コンポーネントには、それぞれ独自のユーザ ID、ログオンコンポーネントの独自のインスタンス、および接続リソースの独自のインスタンスがあることに注意してください。また、同じ接続リソースを使用してシリアルログオンのオーバーヘッドを繰り返し引き起こす、複数の Telnet コンポーネントを実行すると、単一のログオン接続により提供されたユーザ ID/ ログオンコンポーネント / 接続インスタンスを再使用できることにもご注意ください。さらに、設計時にユーザは、1 つのログオン接続オブジェクト、1 つのログオンコンポーネントオブジェクト、および 1 つの接続オブジェクトのみを作成できます。ログオン接続では、ランタイムにユーザ ID/ パスワードそれぞれに対して個々のインスタンスが作成されます。

ログオン接続、ログオンコンポーネント、接続を組み合わせたものが、接続プールとなります。追加の接続プールを定義して、展開する必要があるかどうか決断を下す際の重要な要素として、1 つまたは複数の Telnet コンポーネントで、ログオンコンポーネントの異なる起動画面を使用する必要が生じた際とします。接続プールを使用して効率性とパフォーマンスを向上させるには、Telnet 端末コンポーネント、ログオン接続、およびログオンコンポーネントで、起動画面が適切に管理されていることを確認してください。

Telnet 接続について

ログオン接続は、Telnet 接続リソースのように実際の接続オブジェクトではなく、ユーザ ID/ パスワードに関連付けられた接続管理パラメータを持つユーザ ID/ パスワードのプールです。重要なパラメータの 1 つは、初期ログオンタスクと起動画面へのメニューナビゲーションを行うユーザ ID/ パスワードすべてに対して、ログオンコンポーネントを使用することです。



ログオン接続では、ログオンコンポーネントの指定に加えて、次のユーザ ID プール機能が提供されています。

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/ 接続が確立されると、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対して、タイムアウト時間を設定して待機し、空き接続を使用できるようにする。

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。

Telnet コンポーネントからの複数のインスタンス、または異なる Telnet 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 Telnet 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、Telnet ホスト、ポート、および端末タイプを共有する) 必要があります。

2 すべての Telnet 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、「Telnet ログオンコンポーネントについて」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

ご使用の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をサポートしている場合、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。この操作を実行するには、次の手順に従います。

- ログオンコンポーネントで使用す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］には 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**］設定はオフのままにします。

こ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。

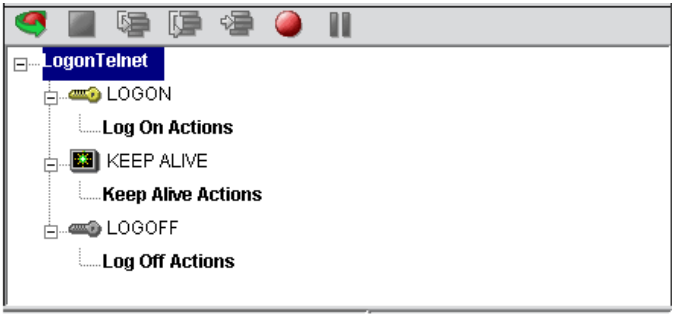
Telnet ログオンコンポーネントについて

ログオンコンポーネントは、複数の Telnet コンポーネントで使用する接続が管理できるよう、アクションモデルが設計された特別なコンポーネントです。ログオンコンポーネントは、2 つの主な違いを除き、Telnet コンポーネントとほぼ同じです。

1 ログオンコンポーネントのアクションモデルは、Logon アクション、KeepAlive アクション、および Logoff アクションから構成される接続タスクにより編成、実行されます。

2 ログオンコンポーネントは、別のコンポーネントやサービスによっては実行されず、ログオン接続によって実行されます。

3 ログオンコンポーネントは、ログオン接続と同時に使用する必要があり、また同時にのみ使用できます。



ログオンコンポーネントの接続タスクをログオン接続と同時に使用すると、パフォーマンスにおいてさらに 3 つの利点があります。

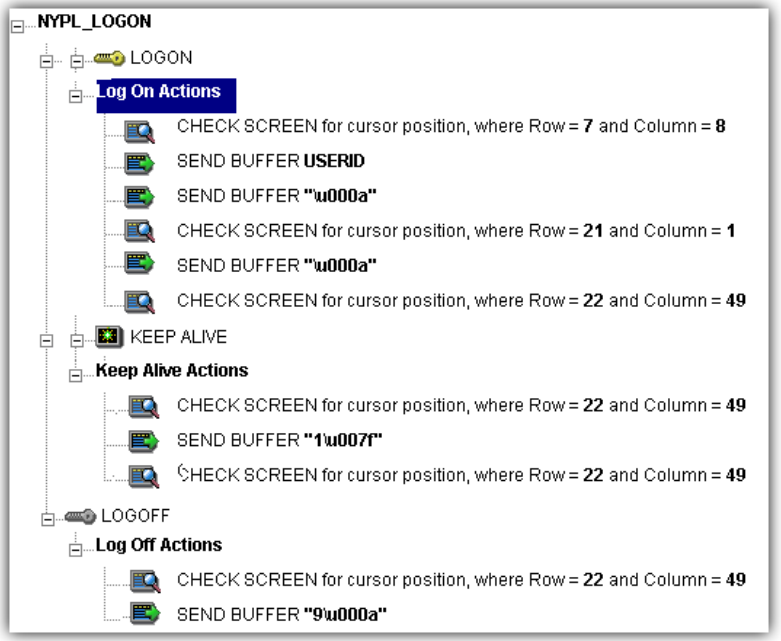
- **LOGON** アクションは、ホスト環境全体で移動し、プールからのユーザ ID によりホストへの接続が最初にアクティブになると、ホストシステム内の希望の起動画面で停止します。その後、接続を再使用する Telnet コンポーネントでは、コンポーネントが起動画面にすでに存在し、新しいセッションを使用した場合と同様に起動画面に移動する際のオーバーヘッドが発生しないため、パフォーマンス上の利点が得られます。
- **KEEPALIVE** アクションは、ホストが標準のタイムアウト時間内に使用されない場合、接続が切断されないようにし、Logon アクションと同じ起動画面で接続された状態にします。
- **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。

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 Telnet 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 Telnet 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 Telnet コンポーネントが動作しなくなります。

Logon アクションは、ログオン接続を使用しない Telnet コンポーネントと同様の方法で作成されます。記録機能を使用し、必要なアクションを作成してユーザ ID やパスワードなどのサインオン情報と、起動画面に表示する初期メニューの選択項目を入力します。その他の留意点として、ログオン接続プールからユーザ ID と

パスワードを使用することが重要です。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。この 2 つの変数を使用して、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マップします。



起動画面は、ログオン接続で提供されたユーザ ID/ パスワードのプールを使用した Telnet コンポーネントすべてに対する共通の実行ポイントとなる必要があります。起動画面に達するには、通常の Telnet コンポーネントでアクションを作成する場合と同じ方法でアクションを作成します。ログオンコンポーネントでは、LOGON アクションは、新しい接続が確立された際に一度だけ実行されます。

Logon アクションは、適切な構造でなければならず、上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。最後の Check Screen アクションでは、画面が接続に達する前に Telnet コンポーネントに制御が移らないよう保証されます。それ以外の場合は、Telnet コンポーネントが無効な画面で開始して例外が発生し、おそらく操作が無効になる場合があります。また、ユーザにより定義された特別な確認 (画面、式、およびカーソル位置) も行われます。Check Screen では、途中で Telnet コンポーネントから制御が渡されないようにしている点が重要です。

3 つのエントリのユーザ ID/ パスワードが完全に使用され、コンポーネントの実行時に 15 回再使用された場合、目的のプログラム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、LOGON アクションは

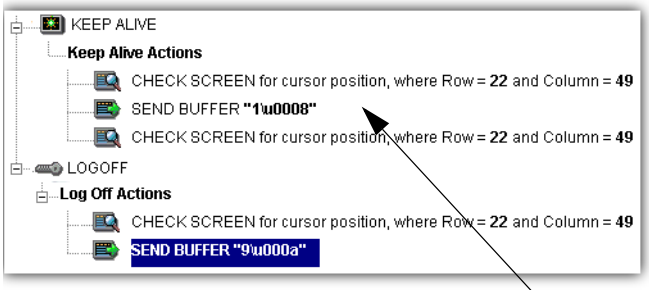
1 回だけ実行されるため、ホストへのログオンは 3 つだけになります。

注記： 可能な場合は、Try/On Error アクションを使用して、回復可能なログオンエラーを捕捉します。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、Telnet の Composer Enterprise Server Console で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ます。詳細については、この章の「プールセクションの管理」を参照してください。

KEEPALIVE アクション

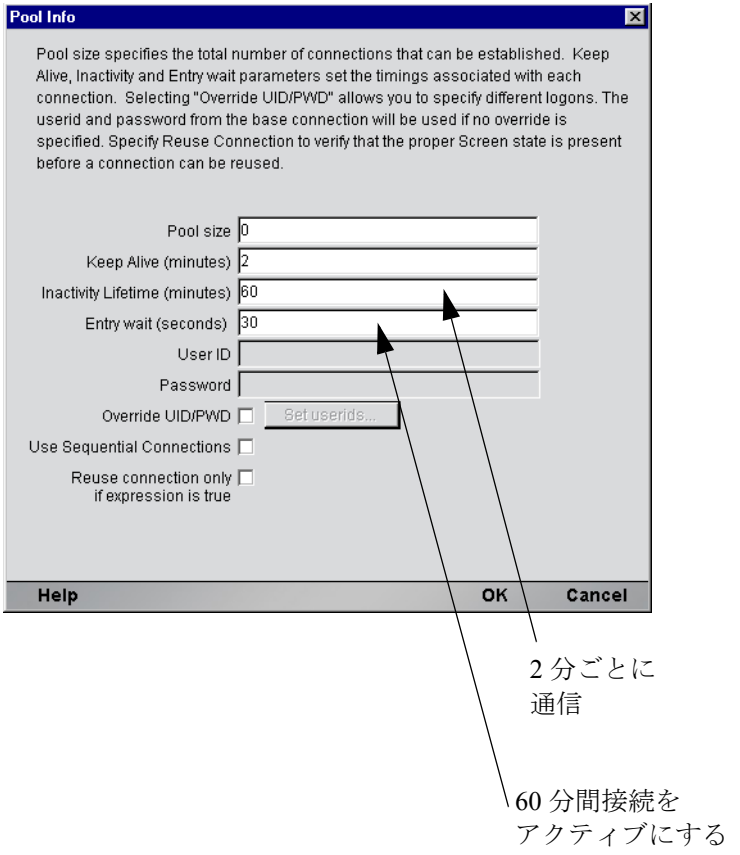
KEEPALIVE 見出しには、アクションのうちアクティビティティを作成し、ログオンコンポーネントで使用する接続上で生じたホストと通信するアクションを配置します。KEEPALIVE アクションでは、通常 <ENTER> キーなどのキーがホストに送信されます。ただし、キーの送信後に画面が起動画面とは異なる画面に変更された場合、ログオンコンポーネントの KEEPALIVE セクションで起動画面に戻る必要があります。起動画面に戻らない場合は、次のコンポーネントが不正な画面で表示され、エラーが発生します。

Telnet 接続では、文字を入力してから <Backspace> キーを使用して入力した文字を消去できます (次の画面を参照)。この結果、バッファアクションが作成され、ホストに送信されます。このアクションでは、ホストとの通信が維持でき、端末の接続状態を保つことができます。



このエスケープシーケンスは、「1」と「Backspace」に対応します。

ログオン接続の［Pool Info］ダイアログボックスでは、KEEPALIVE アクションを実行する頻度を制御します。ログオン接続プールで、空き接続を 60 分間アクティブに維持するが、アクティビティティから 2 分後にホストにより接続が切断されるよう指定した場合、キーボードアクションを指定して (例 : <ENTER> キーを送信する)、接続がアクティブであることをホストに通知できます。

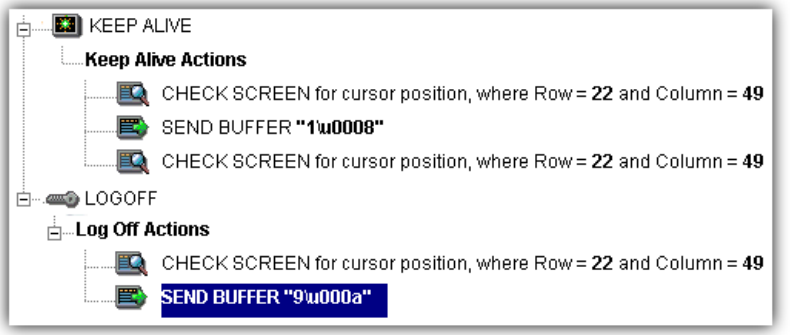


KEEPALIVE アクションは、ログオン接続の［Pool Info］ダイアログボックスで定義した KeepAlive の間隔後に限り、複数回実行できます。

注記： KeepAlive アクションを実行しても、ログオン接続で非活動ライフタイムクロックはリセットされません。Telnet コンポーネントを実行した場合のみ、非活動ライフタイムがリセットされます。

KeepAlive 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も処理されなければなりません。［Keep Alive］セクションでは、接続が切断されないようにするほかにも、実行完了時に起動画面が存在するよう確認する必要があります。最初の Check Screen では、接続が使用できるが使用されていない間、ホストから予期しない画面が届いていないことが確認されます。そして再び最後の Check Screen では、KeepAlive アクションの実行後、途中で次の Telnet コンポーネントに接続が解放されないようにします (次の画面を参照)。



LOGOFF アクション

基本的に、Logoff アクションは、ユーザ ID を適切にホストシステム外に移動させます。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)、画面の式条件を満たさない場合、または接続が Telnet サーバコンソールを介して切断された場合に、1 つの接続に対して一度だけ実行されます。

LOGOFF アクションのパフォーマンスの最大化

Logoff アクションは、安定していなければなりません。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、ユーザ ID がプールで解放されます。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、停止状態が続いた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 KeepAlive アクションに関しては、ログオフの開始時に適切な画面となるよう保証する方法は、Check Screen から開始することです。

ログオンコンポーネントの実行

ログオンコンポーネントを作成するために、システムで新しい接続が要求されるごとに、最初の使用可能なユーザ ID/ パスワードの組み合わせが新しい接続に関連付けられます。その後、目的の起動画面に達するまで、Logon アクションが実行されます。この時点で、Telnet コンポーネントの実行が開始します。実行が終了すると、同じログオンコンポーネントを使用した別の Telnet コンポーネントの実行が同じ起動画面から開始します。

別のコンポーネントの実行が開始しない場合、ログオン接続の［Pool Info］ダイアログボックスの［Inactivity Lifetime］と［KeepAlive］の設定で定義されたアクティブな空き状態が、接続によって入力されます。KeepAlive 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、KeepAlive 間隔の終了時に KeepAlive アクションが実行され、KeepAlive 間隔が最初から開始します。非活動ライフタイムおよび KeepAlive の間隔は、ログオン接続の［Pool Info］ダイアログボックスで定義されます。

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。非アクティブな状態が続いたため、ログオンコンポーネントに関連付けられたログオン接続の期限が切れた場合、ログオンコンポーネントの実行は中止されます。他の方法でログオンコンポーネントの実行を停止するには、サーバの Telnet コンソールを使用する必要があります。

注記 : 不正なユーザ ID やパスワードを使用して接続しようとした場合、接続インスタンスは使用できなくなり、プールのメンバーは次の接続要求でスキップされます。「Logon connection in pool <Pool name> was discarded for User ID <User ID>」というエラーメッセージがサーバログに送信されます。

接続プールの作成

概要

Telnet 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する必要があります。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て、接続がホストで開始し、ホストにアクセスする Telnet コンポーネントに戻るよう示す必要があります。接続プールを作成する標準的な手順は、次のとおりです。

- ホスト接続を作成する
- 接続を使用するログオンコンポーネントを作成する
- ログオンコンポーネントを使用するログオン接続を作成する
- ログオン接続を使用する 1 つまたは複数の Telnet コンポーネントを作成する

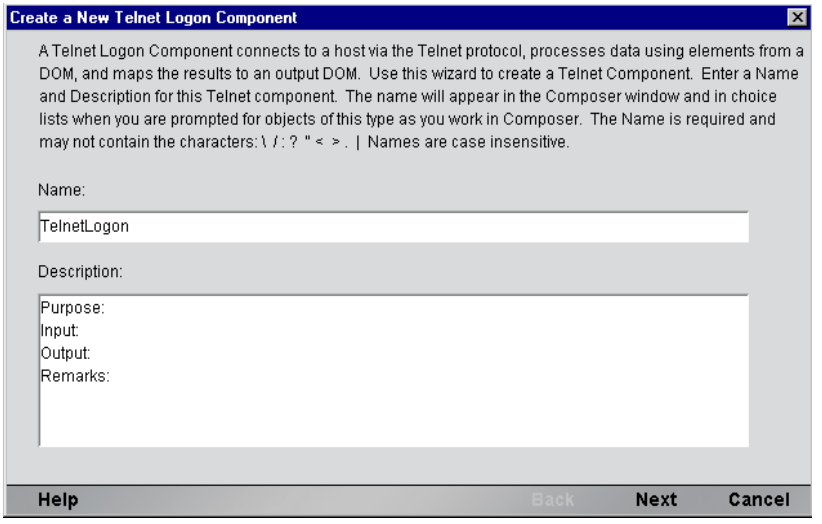
接続の作成

この手順は簡単です。このガイドの第 2 章での説明に従って新しい接続リソースを作成します。ログオン接続で後に定義されたユーザ ID とパスワードを使用している場合でも、接続で同様にユーザ ID とパスワードを定義する必要があります。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

➢ **Telnet ログオンコンポーネントを作成する**

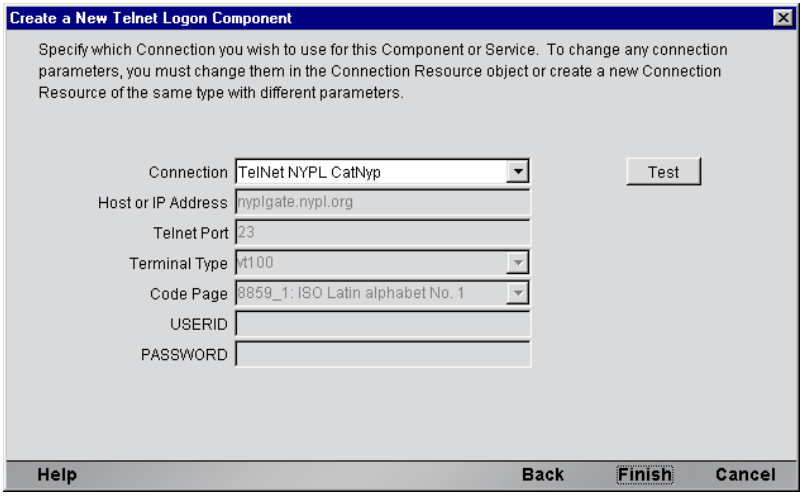
Composer の［**File**］メニューから、［**New xObject**］、［**Component**］、［**Telnet Logon**］の順に選択します。新しい xObject ウィザードのヘッダ情報パネルが表示されます。



4 ［**Name**］に、接続オブジェクトの名前を入力します。

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

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 ドロップダウンリストで「**接続**」を選択します。

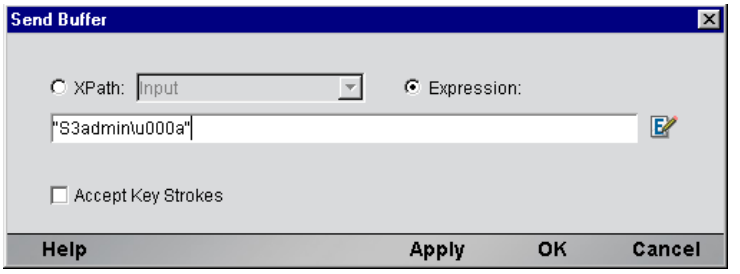
8 ［**Finish**］をクリックすると、ログオンコンポーネントエディタが表示されます。

9 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。完了すると、記録をオフにします。「KEEPALIVE」にカーソルを合わせてから、記録をオンにします。完了すると、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。完了すると、記録をオフにします。

10 このガイドの第 4 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。

11 このガイドの第 4 章の「Telnet 専用の Expression Builder 拡張」の節で説明されたユーザ ID とパスワードを入力して特別な「USERISD」変数と「PASSWORD」変数を使用する、LOGON Map アクションを編集します。

アクションモデルの KEEPALIVE セクションで、必要な SEND Buffer アクションを作成します (簡単な方法として、既存の SEND key アクションをコピーして、貼り付けてから、送信されたキーコードを修正します)。



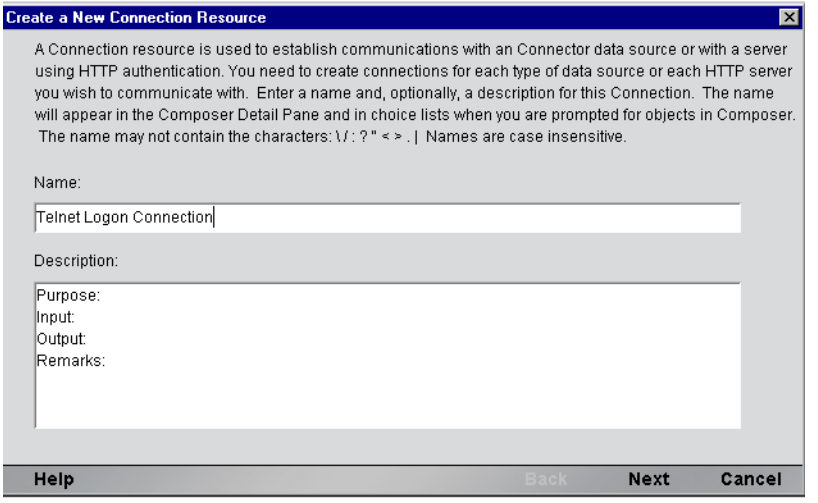
12 ホストを適切に終了するために、LOGOFF アクションを記録します。

13 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

➢ Telnet ログオン接続を作成する

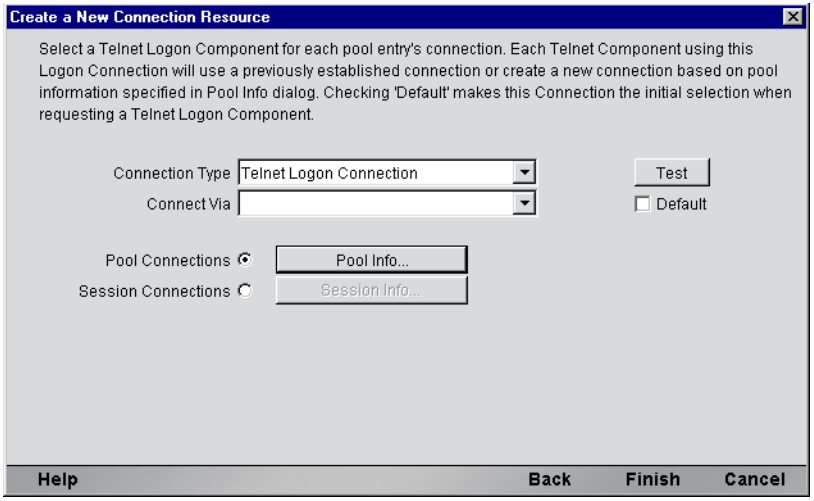
1 Composer の［File］メニューから、［**New xObject**］、［**Resource**］、［**Connection**］の順に選択するか、アイコンをクリックすることもできます。新しい xObject ウィザードのヘッダ情報パネルが表示されます。



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。

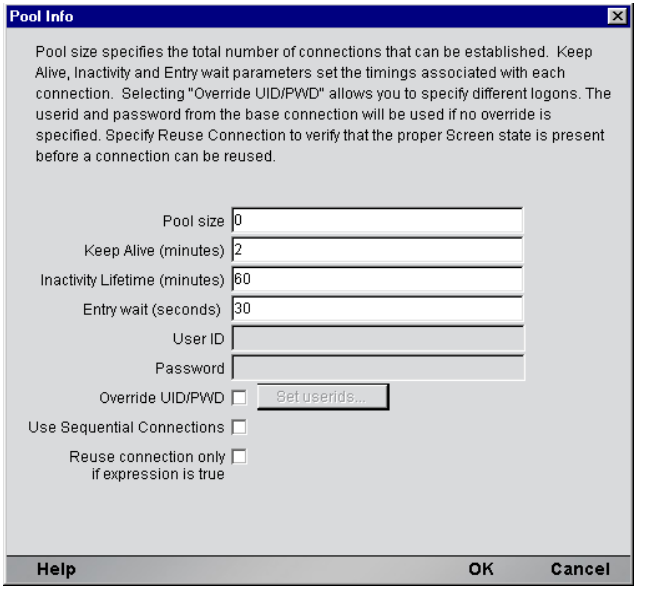
4 ［**Next**］をクリックすると、「**接続情報**」パネルが表示されます。



5 ［**Connection Type**］には、ドロップダウンリストで［Telnet 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。



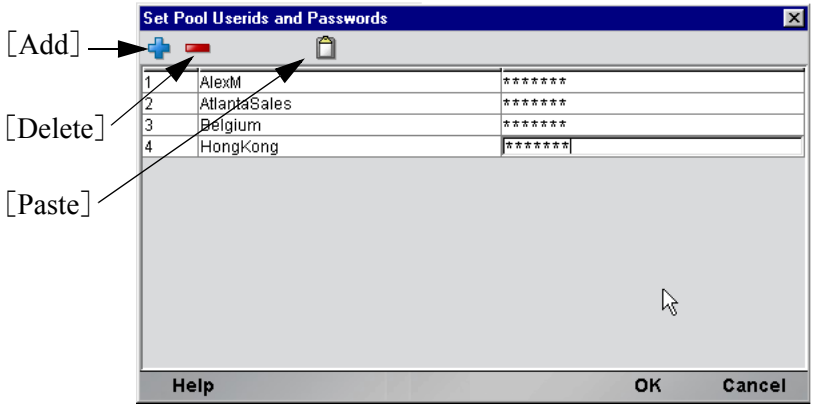
8 ［**Pool Size**］に数値を入力します。ここで入力した数値は、このプールで使用できる接続の総数を表します。各接続に対して、後にユーザ ID とパスワードの組み合わせを指定するよう求められます。

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、Telnet 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 KEEPALIVE 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タイムアウト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、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るまでに、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。

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使用されている場合、Telnet 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 USERID/PASSWORD］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、少なくともユーザ ID とパスワードの 2 列が選択されているか確認してください。1 番目と 2 番目の列にはデータが入力されている必要があり、その他の列はすべて無視されます。画面に表示された最初の数値列は、自動生成されます。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。表を開き、ただちに［Paste］ボタンを押します。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 プールサイズは、入力した行数に応じて更新されます。

14 順番にすべての接続を確立する場合は、オプションで［**Use Sequential Connection**］をオンにできます (一部のシステムでは、画面に同時にアクセスできないよう画面が矛盾する場合があります)。

15 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。このコントロールでは、新しい Telnet コンポーネントでアクティブな空き接続を再使用するたびに、起動画面が存在することを確認する式を入力できます。［**OK**］をクリックして、接続情報パネルに戻ります。この章の「Telnet 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。

注記： 特定の画面が存在するかどうか確認する場合に使用するカスタムスクリプトのサンプルを次に示します。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;
    }
}
```

16 ［OK］をクリックして、接続情報パネルに戻ります。

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

Telnet ログオン接続のパフォーマンスの最大化

ログオン接続リソースでは、Telnet コンポーネントが前の Telnet コンポーネントによって無効な画面に残された接続で実行を開始しないようにするために、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、Telnet 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と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。

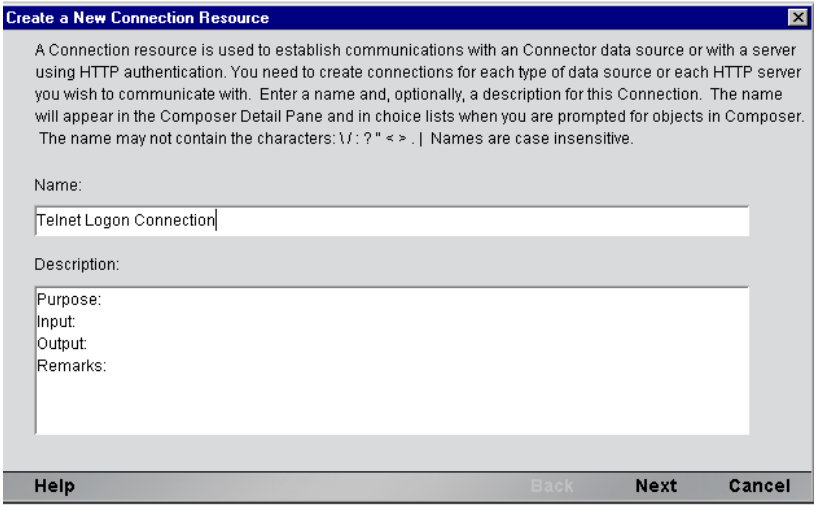
動的に作成したドキュメント / 要素と静的に作成したドキュメント / 要素の違い

一部の Composer アプリケーションでは、ユーザは XML ドキュメントに各種のコントロール、監査、メタデータを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となる場合もあれば、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

➢ **Telnet ログオン接続を作成する**

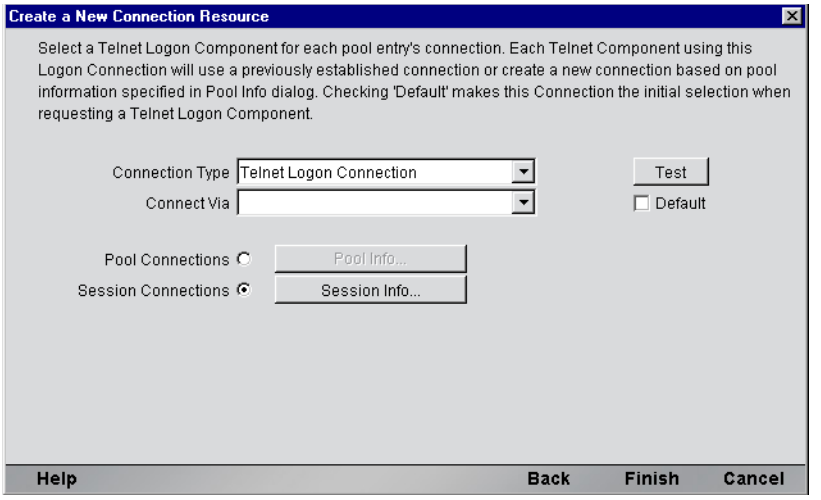
Composer の［**File**］メニューから、［**New xObject**］、［**Resource**］、［**Connection**］の順に選択するか、アイコンをクリックすることもできます。新しい xObject ウィザードのヘッダ情報パネルが表示されます。



18 ［**Name**］に、接続オブジェクトの名前を入力します。

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

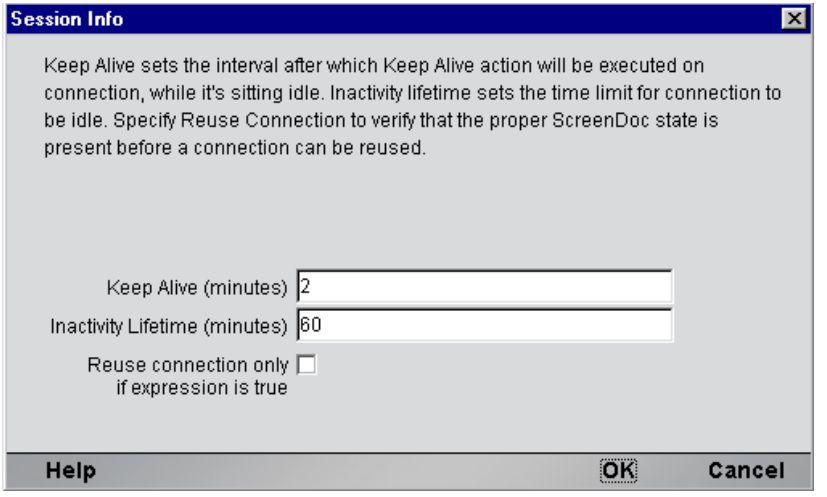
20 ［**Next**］をクリックすると、「**接続情報**」パネルが表示されます。



21 ［Connection Type］には、ドロップダウンリストで［Telnet Logon Connection］を選択します。

22 ［**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。

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



24 ［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、Telnet 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 KEEPALIVE 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。

25 ［Inactivity Lifetime (minutes)］の数値は、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るまでに、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たものです。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に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。

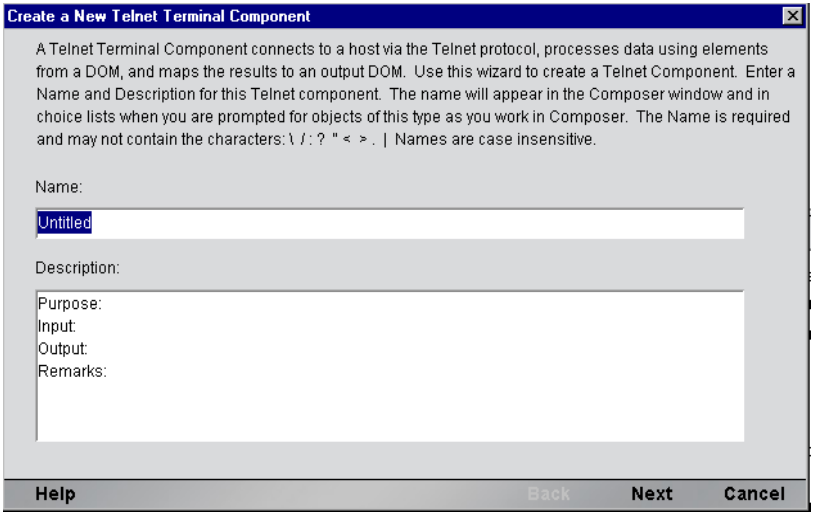
26 ［Reuse connection only if expression is true］をオンにする場合は、チェックボックスをクリックします。そうすると、［Expression］フィールドが自動的に開き、式アイコンをクリックすると［Reuse connection only if the expression is true］ダイアログボックスを表示できます。

Telnet 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 Telnet コンポーネントを作成する準備が整いました。唯一の違いが新しい xObject ウィザードで指定した接続となる、ほぼ通常の Telnet コンポーネントを作成できます。

➢ Telnet 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**］、［**Telnet**］の順に選択します。新しい xObject ウィザードのヘッダ情報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**］をクリックします。接続情報パネルが表示されます。

7 ［**Logon Connection**］で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 このガイドの他の箇所で説明されたように、コンポーネントを作成します。

Telnet 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、起動画面は接続を使用する Telnet 端末コンポーネントに渡されます。その後、(実行終了時に) Telnet 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。Telnet 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 Telnet 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 Telnet 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。

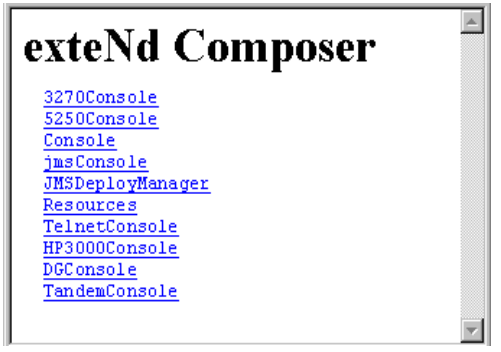
プールの管理

接続プールは、Telnet コンソール画面を使用して管理できます。

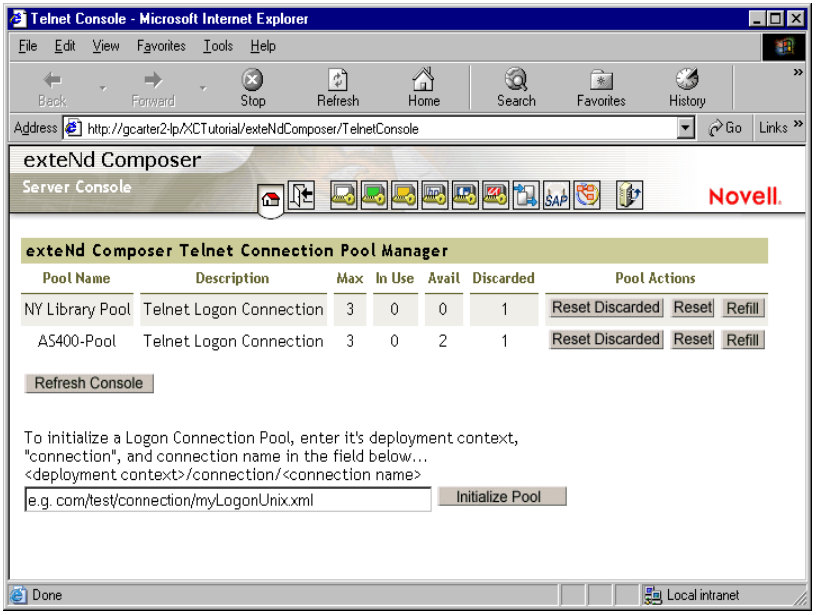
➢ コンソールへのアクセス方法

1 お使いの Web ブラウザで、http://localhost/SilverMaster35 にアクセスしサーバにログオンします。この例では、Novell サーバを使用します。

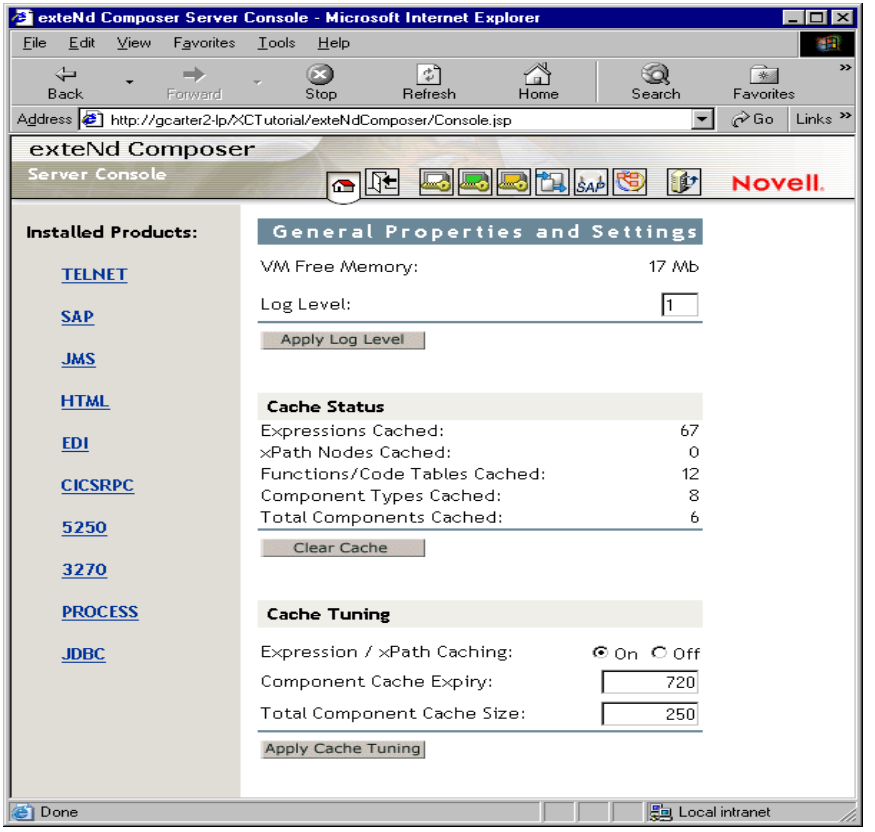
2 ［exteNd Composer］をクリックすると、インストール済みの接続のリストが表示されます。



3 ［Telnet］をクリックすると、コンソール画面が表示されます。フィールドにパスと名前を入力して、［**SUBMIT**］ボタンをクリックします。Telnet Connect の接続がすでに存在している場合は、接続プールの管理画面に表が表示されます。詳細については、該当するサーバガイドを参照してください。



exteNd Composer コンソールの左側の［Installed Products］リストで［Telnet］を選択して、［exteNd Composer Telnet Connection Pool Manager］を表示させることもできます。



接続プールの管理および配備サービス

接続プールの管理画面には、Telnet 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、接続名、接続の説明、プールでの最大接続数、最大使用接続数、使用可能な接続数、破棄された接続数が表に一覧表示されます。その他には、接続をリセットできるボタンが 3 つの列で表示されます。

［Reset Discarded］ボタンは、破棄されてから表に反映された接続をリセットします。［Reset Pool］ボタンは、使用可能な接続で、破棄されてから表に反映された接続をリセットします。［Refill Pool］ボタンは、プールを最大接続数まで再読み込みします。

表の下には、[Refresh] ボタンがあり、このボタンをクリックすると接続プールの現在の状態が示されます。この下のフィールドでは、配置 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できます。この操作は、配置 jar を抽出しないと動作しません。終了すると［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 1 人以上のユーザが単一のリソースやリソースのセットにアクセスできることです。テーブルからユーザ ID とパスワードを選択すると、ログオンコンポーネントで接続が確立されます。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。

Telnet 用 Connect では、不正なユーザ ID、パスワードのタイムアウトなど理由を問わず、正常に確立できなかった接続をすべて破棄し、その他の接続を再使用することで、他の接続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、Telnet 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。Telnet コンポーネントでは、正しい Send Attention 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。

ラインタイム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、下の関数と同様の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。

```
function checkValidReleaseScreen(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 // Write error messages to
                // System.out and System.err:
```

```
        {
    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;
        }
}
```

この関数を実行すると、画面テキストが確認され、最後の画面が正しくない場合「false」が返されます。画面に「MENU」や「APLS」が含まれ、「COMMAND UNRECOGNIZED」や「UNSUPPORTED FUNCTION」が含まれない場合は、「true」が返されます。

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 Telnet の画面表示を XML ドキュメントとして表す、Telnet (および 5250) コンポーネントエディタウィンドウにある特別な DOM。

Send Buffer

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

TCP/IP

Transmission Control Protocol/Internet Protocol (伝送制御プロトコル / インターネットプロトコル) の略語。

Telnet

ANSI 標準システム上で端末をエミュレートする際に使用する通信プロトコル (TCP/IP) に対する仕様 (RFC854)。

VT100

VAX 端末、モデル 100。この端末クラスで使用する特定の ANSI エンコードも表します。

先読み入力

キーボードバッファを複数画面に渡るコマンドでプリロードする方法。

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

「ダム端末」のランタイムの動作をデスクトップ (またはその他の) コンピュータで模倣する方法。

ネイティブ環境ペイン

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

B Telnet の対応キーボード

Telnet の一般キー	
Arrow Down	\u001b[B
Arrow Left	\u001b[D
Arrow Right	\u001b[C
Arrow Up	\u001b[A
BackSpace	\u0008
Back Tab	\u001bOP\u0009
Delete	\u007f
Escape	\u001b
Linefeed	\u000a
Return	\u000d
Tab	\u0009

Telnet のファンクションキー F1 - F20	
F1	\u001bOP
F2	\u001bOQ
F3	\u001bOR
F4	\u001bOS
F5	\u001b[15~
F6	\u001b[17~
F7	\u001b[18~
F8	\u001b[19~
F9	\u001b[20~
F10	\u001b[21~
F11	\u001b[23~
F12	\u001b[24~
F13	\u001b[25~
F14	\u001b[26~
F15	\u001b[28~
F16	\u001b[29~
F17	\u001b[31~
F18	\u001b[32~
F19	\u001b[33~
F20	\u001b[34~

Telnet の NumPad キー	
0	\u001bOp
1	\u001bOq
2	\u001bOr
3	\u001bOs
4	\u001bOt
5	\u001bOu
6	\u001bOv
7	\u001bOw
8	\u001bOx
9	\u001bOy
Minus	\u001bOm
Comma	\u001bOl
Period	\u001bOn
Enter	\u001bOM
Telnet のコントロールキー	
ACK	\u0006 (<Ctrl>+<F>)
BELL	\u0007 (<Ctrl>+<G>)
BS	\u0008 (<Ctrl>+<H>)
CAN	\u0018 (<Ctrl>+<X>)
CR	\u000d (<Ctrl>+<M>)
DC1 or XON	\u0011 (<Ctrl>+<Q>)
DC2	\u0012 (<Ctrl>+<R>)
DC3 or XOFF	\u0013 (<Ctrl>+<S>)

DC4	\u0014（<Ctrl>+<T>）
DLE	\u0010（<Ctrl>+<P>）
EM	\u0019（<Ctrl>+<Y>）
ENQ	\u0005（<Ctrl>+<E>）
EOT	\u0004（<Ctrl>+<D>）
ESC	\u001b（<Ctrl>+<[>）
ETB	\u0017（<Ctrl>+<W>）
ETX	\u0003（<Ctrl>+<C>）
FF	\u000c（<Ctrl>+<L>）
FS	\u001c（<Ctrl>+<\>）
GS	\u001d（<Ctrl>+<]>）
HT	\u0009（<Ctrl>+<I>）
LF	\u000a（<Ctrl>+<J>）
NAK	\u0015（<Ctrl>+<U>）
NUL	\u0000 （<Ctrl>+ スペースバー）
RS	\u001e（<Ctrl>+<~>）
SI	\u000f（<Ctrl>+<O>）
SO	\u000e（<Ctrl>+<N>）
SOH	\u0001（<Ctrl>+<A>）
STX	\u0002（<Ctrl>+<B>）
SUB	\u001a（<Ctrl>+<Z>）
SYN	\u0016（<Ctrl>+<V>）
US	\u001f（<Ctrl>+<?>）
VT	\u000b（<Ctrl>+<K>）

Telnet のその他のキー	
Do	\u001b[29~
Find	\u001b[1~
Help	\u001b[28~
Insert	\u001b[2~
KeyEnd	\u001b[F
KeyHome	\u001b[H
NextScn	\u001b[6~
PrevScn	\u001b[5~
Remove	\u001b[3~
Select	\u001b[44~

C Telnet 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。

数値	属性
0	標準の表示
1	太字モード
2	フェイント
3	強調
4	下線 (一重のみ)
5	点滅モード
7	リバースビデオモード
8	非表示
30	黒色の前景
31	赤色の前景
32	緑色の前景
33	黄色の前景
34	青色の前景
35	マゼンタ色の前景

36	シアン色の前景
37	白色の前景
40	黒色の背景
41	赤色の背景
42	緑色の背景
43	黄色の背景
44	青色の背景

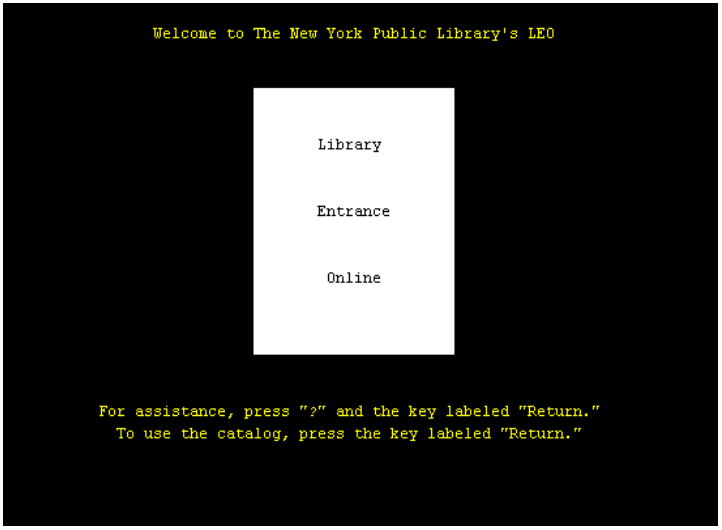
数値	属性
45	マゼンタ色の背景
46	シアン色の背景
47	白色の背景

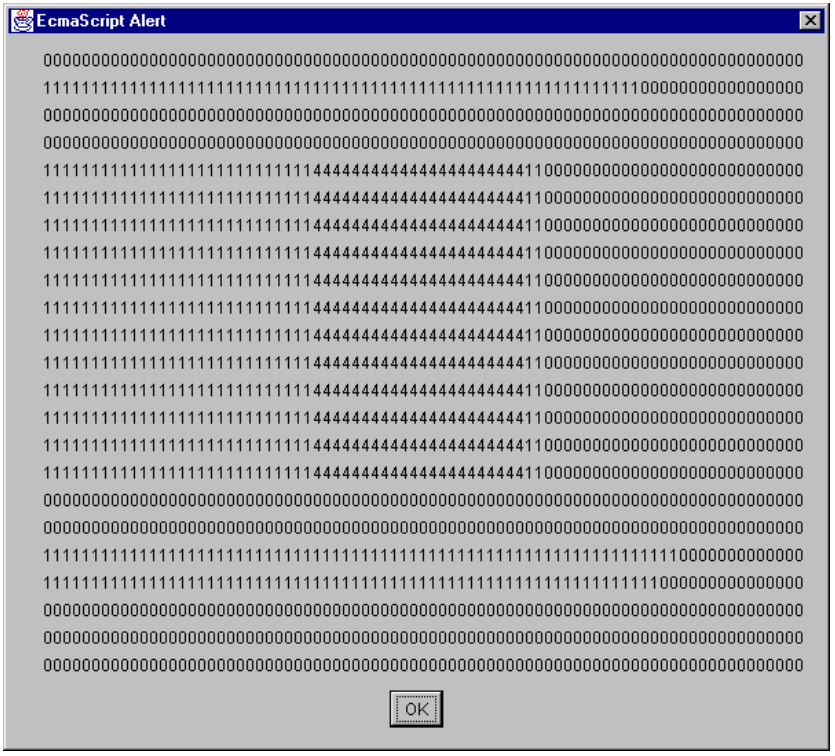
すべての文字属性を一度に表示

`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.getRowCount(); i++, attribs += "\n")
          for (var k = 1; k <= myScreen.getColumnCount(); k++)
               attribs += myScreen.getAttribute(i,k);
     // display the results:
     alert( attribs );
}
```

Telnet 画面および画面への `showAttributes()` 関数の適用結果は、次の図のとおりです。




EcmaScript Alert
OK

D 予約語

次の用語は、Telnet 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
- getCursorColumn
- getColumnCount
- getPrompt
- getRowCount
- getText
- getTextAt
- getTextFromRectangle
- setText

E Java コードページ

文字エンコーディングについて

exteNd Composer の文字エンコーディング変換機能は、使用されている Java VM と密接な関係があります。サポートされる文字エンコーディングは、Java 2 プラットフォームの実装によって異なります。Sun の Java 2 Software Development Kit, Standard Edition, v. 1.2.2 for Windows or Solaris および Java 2 Runtime Environment, Standard Edition, v. 1.2.2 for Solaris では文字エンコーディングがサポートされます。文字エンコーディングは、次の Sun の Web ページで参照できます。

http://java.sun.com/products//jdk/1.2/docs/guide/internat/encoding.doc.html

Sun's Java 2 Runtime Environment, Standard Edition, v. 1.2.2 for Windows には、米国版と国際版の 2 種類があります。国際版 (lib\i18n.jar ファイルを含む) では、両方の表にあるすべての文字エンコーディングがサポートされます。

索引

記号

A

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

ゆ

よ

る

れ

ろ