HP UFT Testing Extensibility

ソフトウェア・バージョン：11.50

開発者ガイド

ドキュメント・リリース日：2010 年 12 月（英語版）
ソフトウェア・リリース日：2010 年 12 月（英語版）

ご注意

保証

HP製品、またはサービスの保証は、当該製品、およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載で追加保証を意図するものは一切ありません。ここに含まれる技術的、編集上の誤り、または欠如について、HPはいかなる責任も負いません。

ここに記載する情報は、予告なしに変更されることがあります。

権利の制限

機密性のあるコンピューターソフトウェアです。これらを所有、使用、または複製するには、HPからの有効な使用許諾が必要です。商用コンピューターソフトウェア、コンピューターソフトウェアに関する文書類、および商用アイテムの技術データは、FAR12.211および12.212の規定に従い、ベンダーの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

著作権について

商標について

Adobe®およびAcrobat®は、Adobe Systems Incorporated (アドビシステムズ社) の登録商標です。

Intel®，Pentium®およびIntel® Xeon™ は、米国およびその他の国におけるIntel Corporationまたはその子会社の商標または登録商標です。

Javaは、Oracle Corporationおよびその関連会社の登録商標です。

Microsoft®，Windows®，Windows NT®および Windows® XPは、米国におけるMicrosoft Corporationの登録商標です。

Oracle®は、カリフォルニア州レッドウッド市のOracle Corporationの米国登録商標です。

Unix®は、The Open Groupの登録商標です。

SlickEdit®は，SlickEdit Inc.の登録商標です。

ドキュメントの更新情報

このマニュアルの表紙には、以下の識別情報が記載されています。

- ソフトウェアバージョンの番号は、ソフトウェアのバージョンを示します。
- ドキュメントリリース日は、ドキュメントが更新されるたびに変更されます。
- ソフトウェアリリース日は、このバージョンのソフトウェアのリリース期日を表します。

更新状況、およびご使用のドキュメントが最新版かどうかは、次のサイトで確認できます。

http://support.openview.hp.com/selfsolve/manuals

このサイトを利用するには、HP Passportへの登録とサインインが必要です。HP Passport IDの登録は、次のWebサイトから行なうことができます。

http://h20229.www2.hp.com/passport-registration.html (英語サイト)

または、HP Passportのサインインページの [**New users - please register**] をクリックします。

適切な製品サポートサービスをお申し込みいただいたお客様は、更新版または最新版をご入手いただけます。詳細は、HPの営業担当にお問い合わせください。

サポート

次のHPソフトウェアサポートのWebサイトを参照してください。

http://support.openview.hp.com

このサイトでは、HPのお客様窓口のほか、HPソフトウェアが提供する製品、サービス、およびサポートに関する詳細情報をご覧いただけます。

HPソフトウェアオンラインではセルフソルブ機能を提供しています。お客様のビジネスを管理するのに必要な対話型の技術サポートツールに、素早く効率的にアクセスできます。HPソフトウェアサポートのWebサイトでは、次のようなことができます。

- 関心のあるナレッジドキュメントの検索
- サポートケースの登録とエンハンスメント要求のトラッキング
- ソフトウェアパッチのダウンロード
- サポート契約の管理
- HP サポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェアカスタマーとの意見交換
- ソフトウェアトレーニングの検索と登録

一部のサポートを除き、サポートのご利用には、HP Passportユーザーとしてご登録の上、サインインしていただく必要があります。また、多くのサポートのご利用には、サポート契約が必要です。HP Passport IDを登録するには、次のWebサイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html (英語サイト)

アクセスレベルの詳細については、次のWebサイトをご覧ください。

http://support.openview.hp.com/access_level.jsp

目次

Testing Extensibility へようこそ

HP Unified Functional Testing（UFT）は，機能テストと回帰テストの自動化をサポートするソリューションです。UFT では，Java，.NET，Oracle などさまざまなテクノロジ環境や開発環境に対応し，各環境で開発するアプリケーションの用語やビジネス・ロジックを反映したテストの作成をサポートする方法として，アドインを使用します。

HP QuickTest Professional Testing Extensibility は，UFT バージョン 11.50 で使用可能な SDK パッケージです。SDK とは，各種テクノロジ環境や開発環境への対応を目的に，UFT 11.50 GUI テスト機能を拡張するパッケージです。アプリケーション開発では，重要なテスト対象となるテクノロジ・インフラストラクチャ，オブジェクト，操作，ビジネス・プロセスを特定し，それに関する豊富な知識をもとに，UFT を使用して有効なアプリケーション・テストを作成します。

さらに，QuickTest Professional Testing Extensibility を作成することによって，ソフトウェアの各バージョンですぐにテストの自動化機能を活用できます。

注：

UFT Testing Extensibility で UFT GUI テスト機能を拡張するには，COM のプログラミング作業が必要になります。Testing Extensibility を使用してユーザ環境に応じた UFT サポートを独自開発する方法だけでなく，HP プロフェッショナル・サービスにサポートを依頼することもできます。

プロフェッショナル・サポートのサポートを依頼せず，ユーザが Testing Extensibility を使用して開発作業を行うには，次に示すスキルと知識が必要です。

- UFT の主要機能に関する知識
- UFT オブジェクト・モデルに関する知識
- COM プログラミングに関する知識と経験
- C++ プログラミングと XML に関する知識

➤ サポートの開発に使用するテクノロジと環境で使用されるインフラストラクチャと実装に関する知識

HP プロフェッショナル・サービスは，UFT サポートの開発において，ユーザ環境に合わせた情報とサポートを提供します。また，サポートの実装サービスも提供しています。詳細については，AskQMsoftwarePS@hp.com にお問い合わせください。

本章の内容

QuickTest Professional Testing Extensibility SDK について

QuickTest Professional Testing Extensibility SDK には次の内容が含まれます。

- UFT GUI テスト機能を拡張し，ユーザ環境で開発するアプリケーションをサポートする API
- Testing Extensibility ドキュメント・セット（API Reference と本開発者ガイド）
- Testing Extensibility を実装するためのサンプル・アプリケーション

このガイドについて

このガイドでは，QuickTest Professional Testing Extensibility SDK をインストールおよびセットアップする方法と，UFT GUI テスト を拡張してユーザ環境で開発したアプリケーションをサポートする方法について説明します。

このガイドを使用するには UFT 機能に関する知識が必要です。QuickTest Professional Testing Extensibility API Reference（オンライン・ヘルプ形式）も参照してください。また，このドキュメントに併せて『HP Unified Functional Testing ユーザーズ・ガイド』と『HP Unified Functional Testing Object Model Reference』も参照してください。このガイドは，UFT メイン・ウィンドウから［**ヘルプ**］＞［**HP Unified Functional Testing 文書ライブラリ**］を選択することによってオンラインで参照できます。

本書の情報，例，画面キャプチャは，特に UFT GUI テストで作業するものに的を絞っています。ただし，Testing Extensibility を使用して開発した このガイドの内容のほとんどはコンポーネントにも適用できます。

ビジネス・コンポーネントおよびスクリプト・コンポーネントは，HP Business Process Testing の一部で，アプリケーションのテストにキーワード駆動型の方法論が使用されます。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

対象読者

このガイドは，独立系ソフトウェア・ベンダのプログラマなどを対象に，UFT GUI テスト 機能を拡張することによって既存の UFT アドインではサポートされないテクノロジやアプリケーションをサポートする方法について説明します。

プロフェッショナル・サポートのサポートを依頼せず，Testing Extensibility を使用して開発作業を行うには，次に示すスキルと知識が必要です。

- UFT の主要機能
- UFT オブジェクト・モデル
- COM プログラミング（豊富な知識と経験）
- C++ プログラミング
- XML

さらに，ユーザ環境のアプリケーションで重要なテスト対象となるテクノロジ・インフラストラクチャ，操作，ビジネス・プロセスに関する豊富な知識が必要になります。

技術的な知識が十分にある方は，まずチュートリアル（105 ページ「QuickTest Testing Extensibility の開始方法」）をご覧ください。ほかのガイドはリファレンスとして活用できます。

注 : HP プロフェッショナル・サービスは，UFT GUI テスト サポートの開発において，ユーザ環境に合わせた情報とサポートを提供します。また，サポートの実装サービスも提供しています。詳細については，AskQMsoftwarePS@hp.com にお問い合わせください。

Unified Functional Testing 文書ライブラリ

Unified Functional Testing 文書ライブラリでは，UFT ドキュメントに 1 箇所からアクセスできます。

Unified Functional Testing 文書ライブラリ には, 次のいずれかの方法でアクセスできます。

- UFT で［**ヘルプ**］>［**HP Unified Functional Testing 文書ライブラリ**］を選択します。
- UFT の［スタート］メニューから，［**すべてのプログラム**］>［**HP Software**］>［**HP Unified Functional Testing**］>［**ドキュメント**］>［**HP Unified Functional Testing ヘルプ**］を選択します。
- 選択した UFT ウィンドウおよびダイアログ・ボックスをクリックするか，F1 キーを押します。
- UFT テスト・オブジェクト，メソッド，またはプロパティの上にカーソルを置いて F1 キーを押すと，説明，構文，例が表示されます。

その他のオンライン・リソース

トラブルシューティング&ナレッジベース: 問題の自己解決が可能な技術情報を検索できる、HPソフトウェアサポートWebサイトのトラブル シューティングのページにアクセスできます。［**ヘルプ**］>［**トラブルシューティング&ナレッジ ベース**］を選択します。このWebサイトのURLは、http://support.openview.hp.com/troubleshooting.jsp です。

HP ソフトウェアサポート：HP ソフトウェアオンラインではセルフソルブ機能を提供しています。また、ユーザディスカッションフォーラムへの書き込みや検索、サポート要求の送信、パッチや更新されたドキュメントのダウンロードなどを行なうこともできます。［**ヘルプ**］>［**HPソフトウェア サポート**］を選択します。このWebサイトのURLは http://support.openview.hp.com/ です。

一部のサポートを除き、サポートのご利用には、HP Passportユーザーとしてご登録の上、サインインしていただく必要があります。また、多くのサポートのご利用には、サポート契約が必要です。

アクセスレベルの詳細については、次のWebサイトをご覧ください。

http://support.openview.hp.com/access_level.jsp

HP Passport IDを登録するには、次のWebサイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html (英語サイト)

HPソフトウェアWebサイト：HPソフトウェアWebサイトにアクセスします。このサイトでは、HPソフトウェア製品に関する最新の情報をご覧いただけます。新しいソフトウェアのリリース、セミナー、展示会、カスタマーサポートなどの情報も含まれています。［**ヘルプ**］＞［**HP ソフトウェア Web サイト**］を選択します。この Web サイトの URL は、http://support.openview.hp.com です。

HP Software は、新しい情報を提供する目的で、製品の文書を継続的に更新しています。

更新状況、およびご使用のドキュメントが最新版かどうかは、HP Software 製品マニュアル (http://support.openview.hp.com/selfsolve/manuals)で確認できます。

第 1 章

QuickTest Professional Testing Extensibility の概要

QuickTest Professional Testing Extensibility SDK を使用すると，QuickTest 自動テスト機能を拡張して，それぞれの環境内のオブジェクトを認識および操作できるように QuickTest を設定できます。

誰からの支援もなしに Testing Extensibility を使用するには，次に示すスキルと知識が必要です。

- QuickTest の主要機能に関する知識
- QuickTest Professional オブジェクト・モデルに対する全体的な理解
- COM プログラミングに関する知識と経験
- C++ プログラミングと XML に関する知識
- サポートの開発に使用するテクノロジと環境で使用されるインフラストラクチャと実装に関する知識

HP プロフェッショナル・サービスは，QuickTest サポートの開発において，ユーザ環境に合わせた情報とサポートを提供します。また，サポートの実装サービスも提供しています。詳細については，AskQMsoftwarePS@hp.com にお問い合わせください。

本章の内容

QuickTest Professional Testing Extensibility について

QuickTest Professional Testing Extensibility SDK には，環境に合わせて QuickTest の全機能を活用するために実装できる一連の API が用意されています。QuickTest Testing Extensibility を実装するには，QuickTest Professional とテスト対象アプリケーション間のインタフェースをとるシングルトンの **Testing Agent** を作成します。

QuickTest ユーザの観点から見ると，QuickTest がインストールされているコンピュータ上に Testing Agent をインストールし登録すると，Testing Agent は QuickTest アドインとして機能します。たとえば，QuickTest は，アドインまたはサポートされている環境のリストを表示するすべてのダイアログ・ボックスで環境の名前を表示します。また，QuickTest は，各アドインで使用できるテスト・オブジェクト・クラスのリストを表示するダイアログ・ボックスで，Testing Agent によって定義されたテスト・オブジェクト・クラスのリストを表示します。

Testing Agent はテスト対象アプリケーションと通信し，QuickTest にアプリケーション内のオブジェクトに関して必要な情報を提供します。QuickTest は環境ごとに 1 つの Testing Agent と通信します。同時にテストするアプリケーションのインスタンスが複数ある場合，Testing Agent は複数のインスタンスを処理する必要があります。

QuickTest Professional は COM インタフェースを介して通信するため，Testing Agent は COM オブジェクトである必要があります。ただし，Testing Agent とテスト対象アプリケーション間の通信は，使用しているテクノロジに最も合った方法で実装できます。

Testing Extensibility インタフェースの詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

Testing Agent と QuickTest の間でやりとりされる一部の情報は，あらかじめ定義されている XML スキーマに準拠する必要があります。QuickTest Testing Extensibility が使用する XML スキーマの詳細については，次のスキーマ（『QuickTest Testing Extensibility API Reference』を参照）を参照してください。**TestingEnvironment**，**ActiveScreen**，**Filter**，**ExternalParent**，**Description**，**AppDescription**，**AddElements**，**Suppression**。

Testing Extensibility SDK のインストール

QuickTest Professional Testing Extensibility SDK を使用すると，それぞれの環境で QuickTest テスト機能をサポートする Testing Agent を設計できます。この SDK のインストールには，次のものが含まれています。

- Testing Agent の作成に使用できる API
- QuickTest Professional Testing Extensibility ドキュメント・セット（API Reference と本開発者ガイド）
- サンプル・アプリケーションおよび QuickTest Testing Extensibility（これらのアプリケーションに対して QuickTest テスト機能を拡張するために実装されています）

  アプリケーションとそれに付随する QuickTest Testing Extensibility のソースもインストールされます。サンプル・アプリケーション用の Testing Agent の実装は，QuickTest Testing Extensibility の実装方法の例を示します。API メソッドの使用例についても，これらのソース・ファイルを参照できます。

  サンプル・アプリケーションとその Testing Agent は，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples** フォルダにインストールされます。

QuickTest Professional Testing Extensibility SDK をコンピュータにインストールするには，SDK アーカイブの **TEA SDK for QTP 11.zip** からファイルを抽出します。サンプル・プログラムを含め，この SDK のインストールには，90MB のハード・ディスクの空き領域が必要です。

Testing Extensibility の実装方法について

QuickTest Testing Extensibility を使用すると，すべての QuickTest 機能の完全なサポートをそれぞれの環境に合わせて実装できます。

QuickTest Testing Extensibility の実装は，次の段階で構成されます。

1 Testing Agent の計画

- テスト・オブジェクト・モデルの定義この段階で，それぞれの環境でサポートされているコントロールとビジネス・プロセスに基づいて，サポートするオブジェクトと操作を決定します。テストで生成される用語とオブジェクト階層を定義することもできます。

➤ Testing Agent として機能する COM オブジェクトの開発とパッケージ化の方法を決定します。

2 Testing Agent の開発

QuickTest Extensibility Agent type library 内の Testing Extensibility によって定義されるインタフェースを実装する COM オブジェクトを作成します。QuickTest は，インタフェース・メソッドを呼び出して，テスト対象アプリケーションに関する情報を取得します。詳細については，17 ページ「QuickTest Professional Testing Extensibility のタイプ・ライブラリとインタフェース」を参照してください。

Testing Extensibility インタフェースは，段階的実装用に設計されています。最も基本的な QuickTest 機能，つまり，オブジェクトの識別とテストの実行をサポートするインタフェースを実装することから始めます。Testing Agent が正常に QuickTest と通信することを確認したら，より高度な QuickTest 機能（オブジェクト・リポジトリへのオブジェクトの追加，テスト・オブジェクトの認識プロパティの実行時の値の取得など）のサポートを開発します。

これらの必須の機能を開発したら，オプションで追加の QuickTest オプションのサポートを開発できます。次の機能を有効にできます。

➤ スマート認識

➤ オブジェクト・スパイ

➤ ネイティブ（実行時オブジェクト）のプロパティとメソッドへのアクセス

➤ ActiveScreen キャプチャ

➤ テストの記録

➤ アプリケーション内のオブジェクトの強調表示

3 Testing Agent のデプロイ

QuickTest がインストールされているコンピュータ上の任意の場所に Testing Agent をインストールします。Testing Agent をインストールしたら，登録します。QuickTest Professional Testing Extensibility SDK には，Testing Agent を登録（および登録解除）できる C 関数が用意されています。

QuickTest が開くと，登録されている Testing Agent が確認され，アドイン・マネージャにそれらの環境が表示されます。QuickTest ユーザは環境を選択し，Testing Agent の COM オブジェクトを作成することでその環境のサポートをロードするように QuickTest を設定できます。［アドイン マネージャ］ダイアログ・ボックスの詳細については，『HP Unified Functional Testing アドイン・ガイド』を参照してください。

QuickTest Professional Testing Extensibility のタイプ・ライブラリとインタフェース

次の表に，**QuickTest Extensibility Agent** type library の必須とオプションのインタフェースについてまとめます。これらのインタフェースを Testing Agent に実装します。

インタフェース	サポートされる QuickTest 機能	重要度レベル
ITestable	➤ オブジェクトの識別 ➤ テストの実行 ➤ テスト・オブジェクトの認識プロパティの実行時の値の取得 ➤ オブジェクトの学習（**ISpyable** の実装も必要になります） ➤ エラーの報告	必須
ITestable2	スマート認識	オプション
ISpyable	オブジェクト・スパイ	推奨
IRunTimeObjectSupplier	ネイティブのプロパティと操作へのアクセス	オプション
IRecordable	記録	推奨
IHWNDSupplier	ActiveScreen と強調表示	オプション IActiveScreenSupplier または IRectangleSupplier を実装する場合に，必要になります。
IRectangleSupplier	強調表示	オプション
IActiveScreenSupplier	ActiveScreen	オプション
ITestEvents	テスト・イベント通知	オプション

記録機能をサポートするために，Testing Agent は，**QuickTest Extensibility** type library 内の QuickTest が提供する API メソッドを呼び出します。Testing Agent が記録に使用するために QuickTest が公開するインタフェースは，**IRecorder** と **IRecordSuppressor** です。

第 2 章

QuickTest Testing Agent の計画

QuickTest Testing Extensibility を実装する最初の段階は，その実装方法を決定し，サポートするすべてのオブジェクトと操作を明確に定義することです。

本章の内容

テスト・オブジェクト・モデルの定義

Testing Agent を計画する最初のステップは, テクノロジが定義する環境を明確に記述することです。QuickTest がテストするオブジェクト，テストにとって重要なオブジェクトのプロパティ，ユーザがオブジェクト上で実行できる操作を識別します。

➤ テクノロジに基づいて，アプリケーションを構成するオブジェクトを表す一連のテスト・オブジェクト・クラスを定義します。

➤ テスト･オブジェクトが QuickTest テスト内で持つと想定される親子関係を定義することで，テスト・オブジェクト・クラスの階層を決定します。できる限り浅く単純な階層を定義します。

テスト・オブジェクト・クラスとテスト・オブジェクト・メソッドの定義

テスト・オブジェクトには，認識プロパティとテスト・オブジェクト・メソッドがあります。テスト・オブジェクトには，一般的なテスト・オブジェクト・メソッドを持つ単純なものと，ビジネス・ロジックを表す固有のテスト・オブジェクト・メソッドを持つ複雑なものがあります。

ユーザがアプリケーションで実行する操作を反映するテスト・オブジェクトとテスト・オブジェクト・メソッドを定義します。たとえば，アプリケーションにエディット・ボックスのセルから構成されるテーブルが含まれている場合，1 つのセルを表す各テスト・オブジェクトの集まりでなく，テーブルを表す 1 つのテスト・オブジェクトを使用します。テーブル・テスト・オブジェクト上の操作は，指定した行，カラム，セルで実行するか，テーブル全体で実行できます。

テスト・オブジェクト・クラスを定義する場合，次のものを指定できます。

- **テスト・オブジェクト・クラス名**と**テスト・オブジェクト・クラス属性**：汎用タイプ，基本クラス，フィルタ・レベルなどを含みます。
  - **汎用タイプ**：テスト・オブジェクト・クラスの汎用タイプ。たとえば，WinEdit および WebEdit テスト・オブジェクトの場合，汎用タイプは Edit です。汎用タイプは，オブジェクトのフィルタ処理時（［オブジェクトをリポジトリに追加］ダイアログ・ボックスや［ステップでオブジェクトを選択］ダイアログ・ボックスなど），およびキーワード・ビューの［**注釈**］カラムの注釈の文字列の作成時に使用する類似のテスト・オブジェクト・クラスのグループを作成するのに使用されます。詳細については，99 ページ「テスト・オブジェクト・クラスの QuickTest 汎用タイプへの割り当て」を参照してください。

    使用可能な汎用タイプのリストについては，次の場所にあるレジストリの場所のキーのリストを参照してください。**HKEY_CURRENT_USER\Software\Mercury Interactive\QuickTest Professional\MicTest\SummaryData\GenTypes**
  - **基本クラス**：新しいテスト・オブジェクト・クラスが拡張するテスト・オブジェクト・クラス。あるテスト・オブジェクト・クラスが別のものを拡張する場合，基本クラスの認識プロパティとテスト・オブジェクト・メソッドが，新しいテスト・オブジェクト・クラスに使用できるものとして，QuickTest のキーワード・ビューとエディタに表示されます。ただし，新しいテスト・オブジェクト・クラスで各認識プロパティと各テスト・オブジェクト・メソッドのサポートは実装する必要があります。

- **フィルタ・レベル**：そのコンテナがオブジェクト・リポジトリに追加されたときに，このタイプのオブジェクトが標準設定で子孫として学習されるかどうかを指定します。つまり，［オブジェクト タイプの選択］ダイアログ・ボックスでこのタイプのオブジェクトが標準設定で選択されるかどうかです。

  テストに表示する必要がないテスト・オブジェクト・クラスを定義する場合，このクラスを学習プロセスからフィルタで除外するように定義して，ユーザが手動で削除する必要がないようにできます。

- **Helpinfo**：キーワード・ビューまたはエディタでこのタイプのオブジェクトに対して F1 キーを押すと，このコンテキスト・ヘルプ・トピックが開きます。

- QuickTest がこのテスト・オブジェクト・クラスに使用する**アイコン**・ファイル（オプション）。指定可能なファイルは，**.dll**，**.ico** のいずれかです。定義しない場合は，汎用アイコンが使用されます。

- **テスト・オブジェクト・メソッド**：このクラスのテスト・オブジェクト上で実行できるメソッドで，各メソッドの次の情報を含みます。

  - **引数**：引数のタイプと方向（In または Out）を含みます。各引数について，その引数が必須かどうかを指定できます。必須でない場合，標準設定値を指定できます。

  - **メソッドの説明**：QuickTest でキーワード・ビュー，エディタ，ステップ・ジェネレータのツールヒントとして表示されます。

  - **注釈の文字列**：QuickTest でキーワード・ビューとステップ・ジェネレータの［**注釈**］カラムに表示されます。

  - **Helpinfo**：キーワード・ビューまたはエディタでこのメソッドに対して F1 キーを押すか，ステップ・ジェネレータでこのメソッドに対して［**操作ヘルプ**］ボタンをクリックすると，コンテキスト・ヘルプ・トピックが開きます。

  - **戻り値のタイプ**

  - **ExposureLevel**：このメソッドが QuickTest を使用している Automation Engineer にのみ使用可能か，Business Process Testing を使用している Subject Matter Expert でも使用可能かを示します。

注 : QuickTest Extensibility Agent タイプ・ライブラリに定義されている必要なすべてのインタフェース・メソッドを実装している限り，Testing Extensibility のすべてのテスト・オブジェクトで，QuickTest の共通のすべてのメソッドとプロパティを実行できます。QuickTest の共通のメソッドとプロパティの定義，またはその実装をオーバーライドしないでください。QuickTest によってサポートされている共通のメソッドとプロパティのリストについては，『HP Unified Functional Testing Object Model Reference』を参照してください。

- **標準設定のテスト・オブジェクト・メソッド :** このクラスのオブジェクトのステップを作成する際に，キーワード・ビューとステップ・ジェネレータで標準で選択されるテスト・オブジェクト・メソッド。
- **認識プロパティ :** 認識プロパティとは，QuickTest テスト内のチェックポイントで確認できるテスト・オブジェクトの特性です。

注 : QuickTest では，内部で **Index**，**Location**，**CreationTime** という名前を使用しているため，テスト・オブジェクトの認識プロパティでは使用しないでください。

認識プロパティのサブセットは，アプリケーション内のオブジェクトの一意な認識に使用されるオブジェクトの記述を構成します。これらは記述プロパティです。

また，次の指定もできます。

- このクラスのオブジェクトのテスト・オブジェクト記述を構成する認識プロパティのセット。

注 : 記述プロパティは，時間が経っても変わらないプロパティである必要があります。アプリケーションの実行中に変化するプロパティ，またはアプリケーションの異なる実行セッションで異なる値を持つプロパティは，使用しないでください。

- チェックポイントおよび出力値ステップ（このクラスのオブジェクト用に QuickTest で作成される）に使用できる認識プロパティ
- QuickTest の［チェックポイントのプロパティ］ダイアログ・ボックスで標準設定で選択される認識プロパティ（このクラスのオブジェクト用にチェックポイントを作成する場合）

詳細については，31 ページ「テスト環境を QuickTest に認識させる方法」を参照してください。

テスト・オブジェクト・クラス，テスト・オブジェクト・メソッド，プロパティの命名

各テスト・オブジェクト・クラスには，一意な名前を付ける必要があります。オブジェクトのタイプとオブジェクトが属しているテスト環境を示す，わかりやすい名前を使用してください。これにより，QuickTest ユーザがテストを理解しやすくなります。

テスト・オブジェクト・クラス名でテスト環境を示すには，環境に属しているすべてのテスト・オブジェクト・クラス名に共通のプレフィックスを追加します。たとえば，Foo テクノロジ・アプリケーションをサポートするために定義されたテスト環境には，FooButton，FooCheckBox，FooBar などを設定できます。

テスト・オブジェクト・メソッドの名前は，メソッドが実行する操作を明確に示す必要があります。テスト・オブジェクト・メソッドの引数の名前には，どのような値を入力する必要があるかが明確にわかる名前を付けてください。

テスト・オブジェクト・クラス，テスト・オブジェクト・メソッド，テスト・オブジェクト・メソッドの引数，テスト・オブジェクトの認識プロパティの名前には，次の規則が適用されます。

- 名前に非英語文字を含めることはできません
- 名前の最初の文字は必ず英字にする必要があります。また，スペースや次の記号を含めることはできません：! @ # $ % ^ & * ( ) + = [ ] \ { } | ; ` : " , / < > ?

オブジェクトのテスト・オブジェクト・メソッドには，予約されているテスト・オブジェクト・メソッド名を使用しないでください。これらのテスト・オブジェクト・メソッドのリストについては，25 ページ「予約済みのテスト・オブジェクト・メソッド」を参照してください。

定義するテスト・オブジェクト・メソッドには，テスト・オブジェクト・クラスの名前を使用しないでください。

QuickTest Professional のタイプおよび関数との整合性を維持するために，テスト・オブジェクト・クラス名とテスト・オブジェクト・メソッド名には Pascal 記法を使用してください。

予約済みのテスト・オブジェクト・メソッド

QuickTest は，すべてのテスト・オブジェクトについて，次に示すすべてのテスト・オブジェクト・メソッドをサポートします。テスト・オブジェクトに対してこれらのメソッドをサポートするために，テスト環境でこれらのメソッドを定義する必要はありません。また，これらのテスト・オブジェクト・メソッド名は QuickTest 用に予約されていて，ユーザが定義するテスト・オブジェクト・メソッドには使用できません。これらの名前では，大文字と小文字は区別されません。

CaptureBitmap	GetProperty	Highlight	SetProperty
Check	GetROProperty	Init	SetTOProperty
CheckProperty	GetTOProperties	Output	ToString
ChildObjects	GetTOProperty	QueryValue	WaitProperty
get_Exist			

Testing Agent 特性の計画

実装を開始する前に，本ガイドを読んで，QuickTest に提供する必要がある情報について学習してください。その後，次の決定を行います。

➤ Testing Agent をどんな言語で実装するか？

COM をサポートしている任意の言語を使用できます。

➤ Testing Agent をテスト対象アプリケーションとどのようにして通信させるか？

テスト対象アプリケーションのテクノロジに合った任意の手法を使用できます。

Testing Agent は，実行中のアプリケーションからテスト対象オブジェクトおよびオブジェクト間の関係に関する情報を取得します。このエージェントは，アプリケーション内の可視なオブジェクト上で操作を実行します。

アプリケーションに複数のインスタンスがある可能性がある場合，Testing Agent は複数のインスタンスを処理する必要があります。

➤ Testing Agent はどのようにして起動するか？

［アドイン マネージャ］ダイアログ・ボックスで QuickTest ユーザが該当する環境を選択した場合，QuickTest が開くときに Testing Agent を起動します。ただし，QuickTest より前にテスト対象アプリケーションが開いている場合，Testing Agent をもっと前に起動させたいと思うかもしれません。1 つの解決策は，Testing Agent をシングルトン・オブジェクトとして実装し，アプリケーションに実行させることです。こうすると，QuickTest またはアプリケーションのいずれか最初に実行されたものによって，Testing Agent が作成されます。さらに，アプリケーションまたは QuickTest が使用している限り，有効な状態を維持します。これにより，テスト対象アプリケーションまたは QuickTest を閉じて再度開く場合の同期の問題が回避されます。

注意: Testing Agent をプロセス外で動作するように実装して，Testing Agent が QuickTest のコンテキストで動作しないようにする必要があります。

➤ QuickTest がアプリケーションをテストしていない場合に，不要なアクティビティが発生していないことをどのように確認するか？

ユーザは，アプリケーション上でテストを実行しない場合，QuickTest とは無関係にアプリケーションを実行できます。この場合，Testing Agent を開かないか，開いても不要な操作は実行しないようにします。

Testing Agent をアクティブにするかどうかを示すフラグを実装することもできます。フラグを設定するタイミングを決定する 1 つの方法は，QuickTest がコンピュータにインストールされているかどうかを確認することです。QuickTest がインストールされているかどうかを確認する方法の詳細については，90 ページ「QuickTest がインストールされているかどうかの確認」を参照してください。もう 1 つの方法は，QuickTest が通信を開始しない限り，Testing Agent が不要な操作を実行しないように設計することです。3 番目の選択肢は，ユーザがアプリケーションを開くときに，テストが必要かどうかを指定できるようにすることです。

➤ Testing Agent はユーザにどのようにして提供されるか？

Testing Agent は，ソフトウェア・インストールの一環として，または別個に QuickTest ユーザに提供できます。

第 3 章

基本的な QuickTest Testing Agent の開発

QuickTest Testing Extensibility の実装は，QuickTest Professional Testing Extensibility SDK で定義されているインタフェースを実装する COM オブジェクトを作成することによって行います。このオブジェクトは，Testing Agent として機能し，QuickTest とテスト対象アプリケーション間の通信を処理します。

本章の内容

QuickTest Testing Agent の開発について

Testing Agent の実装方法を決定し，第 2 章「QuickTest Testing Agent の計画」の説明に従って Testing Agent がサポートするオブジェクト・モデルを決定したら，QuickTest Testing Extensibility の実装を開始できます。最初のステップは，**QuickTest Extensibility Agent** type libraryで定義されている **ITestable** インタフェースを実装することです。このインタフェースにより，最も基本的な QuickTest 機能のサポートが提供されます。

ITestable インタフェースは，Testing Extensibility SDK で必須の唯一のインタフェースです。このインタフェースを実装すると，QuickTest によってオブジェクトを認識し，それぞれの環境で開発されたアプリケーションでテストを実行できるようになります。また，テスト・オブジェクトの認識プロパティの実行時の値の取得とテスト・オブジェクトのオブジェクト・リポジトリへの追加をサポートできるようになります。

ITestable インタフェースを実装したら, QuickTest Professional Testing Extensibility SDK の追加インタフェースも実装できます。これらにより，記録，ActiveScreen，オブジェクト・スパイ，強調表示などの QuickTest のより高度な機能もサポートされます。詳細については，57 ページ「QuickTest Testing Agent の機能拡張」を参照してください。

次の表は，各 Itesttable メソッドが使用されている QuickTest の主要機能を示します。

QuickTest 機能	API メソッド
テスト環境の認識	**GetTestingEnvironment**（ハンドシェイク）
オブジェクト認識	**BuildDescription**，**FindObjectId**，**FindObjectId2**，**GetDisplayName**，**GetParent**，**GetChildren**
テストの実行	**Run**
オブジェクト認識プロパティの値の取得	**GetElementType**，**GetProperties**，**GetTabularData**，**GetTabularAttributes**
意味のあるエラー・メッセージの提供	**GetLastError**
オブジェクトの学習	**LearnChildObjects**，**CompareObjectIds**

このインタフェースの詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

注 : ITestable2 インタフェースは，**ITestable** を継承し，**FindObjectId2 メソッドを宣言しています。**QuickTest スマート認識をサポートするには，**FindObjectId** でなく **FindObjectId2** を実装してください。詳細については，40 ページ「FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て」を参照してください。

エージェントを次の段階に従って開発し，各段階の開発後にテストを実行します。

1 Testing Agent と QuickTest 間のハンドシェイクを開発します。

Testing Agent と QuickTest 間の初期ハンドシェイクのみ実装します。これにより，QuickTest でテスト環境が認識できるようになるため，環境で定義されたオブジェクトと操作を使用して QuickTest テストを作成できます。

2 オブジェクトを認識し，テストを実行するためのサポートを作成します。

QuickTest でいくつかのオブジェクトを認識しテストを実行するために必要なメソッドを実装します。

3 認識プロパティの実行時の値を取得するためのサポートを作成します。

QuickTest でテスト対象アプリケーションから認識プロパティの実行時の値を取得するために必要なメソッドを実装します。

4 オブジェクトを学習するためのサポートを作成します。

QuickTest でオブジェクトを学習し，それらのオブジェクトをオブジェクト・リポジトリに追加するために必要なメソッドを実装します。

5 作成したサポートを環境内のすべてのオブジェクトを対象とするように拡張します。

環境内のすべてのオブジェクトを対象とするように, テスト環境の定義を完成します。環境内のすべてのテスト・オブジェクトと操作のサポートを実装します。**ITestable** インタフェースで残りのメソッドを実装します。

開始する前に

本項では, Testing Agent を開発する前に必要な基本的なプログラミング情報について説明します。

- Testing Agent をプロセス外で動作するように実装して，Testing Agent が QuickTest のコンテキストで動作しないようにする必要があります。
- Testing Agent を開発すると，プロジェクトは QuickTest Professional Testing Extensibility SDK で提供されるタイプ・ライブラリを参照します。これらのライブラリは，**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK \Ifs** にあります。Microsoft Visual Studio を使用して Testing Agent を開発している場合，次の Windows コマンドを使用してタイプ・ライブラリを登録します。regtlib <.TLB **ファイルの完全パス>**。ライブラリは次のとおりです。
  - **QuickTest Extensibility Agent** type library（**AutInterface.tlb** および **AutInterface.idl** ファイル）。このライブラリは必須です。

- **QuickTest Extensibility** type library（**QtpInterface.tlb** および **QtpInterface.idl** ファイル）。このタイプ・ライブラリは，記録をサポートする場合にのみ必要となります。詳細については，71 ページ「記録のサポートの開発」を参照してください。
- **QuickTest Extensibility 仮想オブジェクト**・タイプ・ライブラリ（**TEAVirtualObject.tlb** および **TEAVirtualObject.idl**）。このタイプ・ライブラリは，テスト・オブジェクト・メソッドからテスト・オブジェクトを返せるようにする場合にのみ必要となります。詳細については，95 ページ「テスト・オブジェクト・メソッドからのテスト・オブジェクトの戻し」を参照してください。

- QueryInterface または QueryService を使用して実装したインタフェースは公開できます。
- すべての COM アプリケーションと同様に，Testing Agent のすべてのメソッドは，retval に加えて HRESULT も返します。QuickTest で定義された HRESULT 戻り値定数とエラー・コードを使用するには，**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\include\TestError.h** をインクルードしてください。
- QuickTest Testing Extensibility API メソッドの使用方法を示す例については，サンプル・アプリケーションとその Testing Agent のソース・ファイルを参照してください。これらのファイルは，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\Samples** フォルダにインストールされています。
  - Hours Report Sample の Testing Agent は，Testing Extensibility がサポートできるすべての QuickTest 機能の C++ 実装です。Hours Report Sample ファイルには，サンプルを実行する方法と QuickTest を使用してサンプルをテストする方法（サンプルの Testing Agent を使用する場合と使用しない場合の両方）について説明している，**Readme** ファイルが含まれています。
  - QuickID サンプル・アプリケーションの Testing Agent は，Testing Extensibility の基本的なメソッドのみの C++ 実装です。QuickID ファイルには，サンプルを実行する方法と QuickTest を使用してサンプルをテストする方法（サンプルの Testing Agent を使用する場合と使用しない場合の両方）について説明している，**Readme** ファイルが含まれています。QuickID サンプルの目的は，Testing Agent の実装時に作業の開始方法を示すことです。付録 C「QuickTest Testing Extensibility の開始方法」では，このアプリケーションの Testing Agent を開発する手順を順を追って説明します。
  - Maps サンプル・アプリケーションの Testing Agent は，C# でのエージェントの作成方法を示します。Maps サンプル・ファイルには，サンプルの実行方法を説明する **Readme** ファイルが含まれています。

Maps サンプル・アプリケーションを実行するには，.NET Framework 3.5 SP1 以降がインストールされているコンピュータとインターネット接続が必要です。

注：

- これらのサンプル Testing Agent では，サンプル・アプリケーションでのテキスト/テキスト領域チェックポイントまたは出力値の作成はサポートされていません。
- Hours Report Sample と QuickID サンプルを使用して作成したテストでは，プロパティ値に正規表現が使用されている場合，QuickTest はオブジェクトを認識できません。

テスト環境を QuickTest に認識させる方法

テスト環境とは，環境内で開発したアプリケーションで QuickTest が認識する必要がある一連のテスト・オブジェクトとそのプロパティ，メソッドのことです。このテスト環境でオブジェクトの認識をサポートするには，QuickTest に環境を記述したすべての情報を提供する必要があります。

テスト環境には，登録済みの Testing Agent でサポートされているほかの環境と区別する一意の名前を付ける必要があります。この名前（**PackageName** または**環境 ID** とも呼ばれます）を使用して，Testing Agent を QuickTest コンピュータ上に登録します。詳細については，89 ページ「Testing Agent の登録」を参照してください。

また，環境の表示名（**AddinName** とも呼ばれます）を定義します。この名前は，QuickTest のアドインまたはサポートされている環境のリストを表示するウィンドウに表示されます。

QuickTest が開くと，登録されている Testing Agent が確認され，アドイン・マネージャにそれらの環境の表示名が表示されます。QuickTest ユーザは環境を選択し，環境の Testing Agent COM オブジェクトを作成するように QuickTest を設定できます。

QuickTest は，Testing Agent との初期ハンドシェイク中にテスト環境の基本的な記述を要求します。この情報を提供するには，**ITestable** インタフェースで **GetTestingEnvironment** メソッドを実装します。

GetTestingEnvironment の実装

QuickTest が Testing Agent を作成すると，**GetTestingEnvironment** を呼び出してテスト環境の基本的な記述を取得します。**GetTestingEnvironment** を実装して，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従ってテスト環境の情報を XML 形式で返すようにする必要があります。

QuickTest を使用して Testing Agent をデプロイする場合，テスト環境の XML ファイルを **<QuickTest Professional インストール・フォルダ>\dat\Extensibility\TEA** フォルダにインストールする必要があります。

QuickTest Testing Extensibility テスト・オブジェクトのメソッドとアイコンを ALM/Quality Center で使用可能にするには（たとえば，テストまたはビジネス・コンポーネントのキーワード・ビューで作業するか，ALM/Quality Center で［リソース ビューア］を使用する場合），環境の XML ファイルを **<QuickTest Add-in for ALM/QC インストール・フォルダ>\dat\Extensibility\TEA** フォルダにもインストールする必要があります。**GetTestingEnvironment** の最も簡単な実装方法は，ファイルを読み取り，内容を返すことです。

詳細については，19 ページ「テスト・オブジェクト・モデルの定義」および 98 ページ「テスト環境の XML に関する追加情報」を参照してください。

GetTestingEnvironment の構文の詳細については，『QuickTest Testing Extensibility API Reference 』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

QuickTest がテスト環境を認識することのテスト

少数のテスト・オブジェクト・クラスを使用して最小限のテスト環境を定義し，**GetTestingEnvironment** メソッドを実装したら，QuickTest と Testing Agent との初期通信をテストできます。

QuickTest がテスト環境を認識することをテストするには，次の手順で行います。

1 Testing Agent を QuickTest がインストールされているコンピュータにインストールします。

2 C 関数 **RegisterTeaAut** を呼び出すプログラムを作成し，プログラムを実行して，Testing Agent を QuickTest コンピュータに登録します。詳細については，89 ページ「Testing Agent の登録」を参照してください。

3 QuickTest を開きます。QuickTest が登録済みの Testing Agent を認識し，その環境名を［アドイン マネージャ］ダイアログ・ボックスに表示します（［アドイン マネージャ］ダイアログ・ボックスが開かない場合は，『HP Unified Functional Testing アドイン・ガイド』の手順を参照してください）。

注：環境名が［アドイン マネージャ］ダイアログ・ボックスに表示されない場合は，Testing Agent が正しく登録されていることを確認してください。

---

4 環境のチェック・ボックスを選択し，［**OK**］をクリックします。QuickTest が Testing Agent を実行し，**GetTestingEnvironment** を呼び出して，テスト環境に関する情報を取得します。

5 QuickTest で，テスト環境で定義されたテスト・オブジェクト上でテスト・オブジェクト・メソッドを実行するテストを作成します。

テストを作成するには，次のいずれかの方法を使用できます。

- オブジェクト・リポジトリ・エディタで新しいテスト・オブジェクトを定義し，キーワード・ビューまたはステップ・ジェネレータのステップでこれらのテスト・オブジェクトを使用します。
- エディタでテストを作成し，プログラム的記述を使用してテスト・オブジェクトを定義します。

詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

6 テスト環境で行った定義が正しく機能することをテストします。次の項目を確認します。

- 定義したテスト・オブジェクト・クラスを表すのに使用されるアイコン（キーワード・ビューとステップ・ジェネレータ内）。
- 各テスト・オブジェクトで使用できるテスト・オブジェクト・メソッド。エディタのステートメント補完機能，キーワード・ビューの［**操作**］カラム，ステップ・ジェネレータの［操作］ボックスを確認します。
- 各テスト・オブジェクト・メソッドに必要な引数の数とタイプ。エディタのステートメント補完機能，キーワード・ビューの［**値**］カラム，ステップ・ジェネレータの引数テーブルを確認します。
- キーワード・ビューの［**注釈**］カラムのステップ・サマリには，テスト環境定義の **Documentation** 要素の文字列が表示されます。
- キーワード・ビュー，エディタ，ステップ・ジェネレータのテスト・オブジェクト・メソッドのツールヒントには，テスト環境定義の **Description** 要素の文字列が表示されます。
- テスト環境定義の次の追加詳細は，ステップ・ジェネレータに反映する必要があります。
  - **ReturnValueType**
  - **HelpInfo**（このステップのコンテキスト・ヘルプ・トピック）
  - **IsMandatory** 属性の値が **true** である引数には，ステップ・ジェネレータで必須のマークを付ける必要があります。

テスト・オブジェクト・モデルのサポートの開発

環境内に存在するオブジェクトのタイプの認識に加えて，Testing Agent では，QuickTest が個々のオブジェクトとアプリケーション内の異なるオブジェクトの階層を認識および識別できるようにする必要があります。

Testing Agent では，ユーザがテストに含める可能性があるユーザ・インタフェース・オブジェクトを QuickTest が識別およびアクセスできるようにする必要があります。オブジェクトは，記述またはオブジェクト ID（実行時 ID とも呼ばれます）で識別できます。

Testing Agent は，テスト対象アプリケーション内の各オブジェクトに対して記述とオブジェクト ID を提供する必要があります。**BuildDescription** および **FindObjectId2**（または **FindObjectId**）メソッドを実装して，オブジェクト ID をテスト・オブジェクト記述に割り当てる必要があり，逆もまた同様です。また，テスト・オブジェクト名を返す **GetDisplayName** メソッドも実装する必要があります。

QuickTest では，すべてのテスト・オブジェクトをある階層に属しているものとして認識します。たとえば，ブラウザには，編集フィールドといくつかのボタンを含むフレームが含まれている場合があります。これにより，コンテキストがオブジェクト認識とグループ関連オブジェクトの両方に追加され，テストが理解しやすくなります。

QuickTest は，任意のテスト・オブジェクトからテスト・オブジェクト階層をスキャンできる必要があります。このために，**GetParent** および **GetChildren** メソッドを実装する必要があります。これらのメソッドは，QuickTest が指定する任意のテスト・オブジェクトの該当する情報を返せる必要があります。

詳細については，次を参照してください。

➤ 36 ページ「オブジェクト ID」

➤ 36 ページ「オブジェクトの記述」

➤ 38 ページ「スマート認識」

➤ 38 ページ「外部の親の概念について」

➤ 39 ページ「BuildDescription の実装：オブジェクト ID の記述への割り当て」

➤ 39 ページ「GetDisplayName の実装：オブジェクト名の戻し」

➤ 40 ページ「FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て」

➤ 40 ページ「GetParent の実装」

➤ 41 ページ「GetChildren の実装」

オブジェクト ID

オブジェクト ID とは，ある時点でシステム内のユーザ・インタフェース・オブジェクトを一意に識別する値です。オブジェクト ID は，オブジェクトに直接アクセスするために使用されます。指定可能な ID は，数字，ポインタ値，ハッシュ・キー，hWnd，データベース文字列，などのアプリケーション固有の参照です。オブジェクト ID は一時的なもので，テスト・セッションごとに異なる可能性があります。オブジェクト ID は，実行時にオブジェクトにアクセスするのに使用されるため，オブジェクトをすばやく簡単に取得できるように設計する必要があります。

オブジェクトの記述

テスト・オブジェクト記述は，オブジェクト名と認識プロパティのセットから構成されています。これらの認識プロパティは，組み合わせると，特定のユーザ・インタフェース・オブジェクトを一意に認識できるものです。テスト・オブジェクト記述は，同じ親に属している異なるテスト・オブジェクトを一意に識別する必要があります。テスト・オブジェクト記述は，QuickTest テストとともに格納され，特定のオブジェクトの識別に使用されます。

テスト・オブジェクト名は，QuickTest のユーザにとって意味がある必要があり，テクノロジまたは環境に関連する用語を使用しているのが望ましいといえます。QuickTest はこの名前を，キーワード・ビュー，エディタ，オブジェクト・リポジトリに表示します。こ

の名前は，オブジェクトの認識には用いられないので，アプリケーション内で一定である必要はありません。

たとえば，言語によって異なるテスト・オブジェクト名を付けることもできます。QuickTest ユーザは，ある言語で動作するアプリケーションのテストを作成する際に，その言語の名前を持つテスト・オブジェクトを作成することができます。ユーザは，別の言語で同じアプリケーションのテストを実行できます。テスト内のテスト・オブジェクトの名前は元の言語のままですが，QuickTest はテスト・オブジェクトを記述に基づいて正しく認識できます。

記述には，オブジェクトの一意な識別に使用される次の項目を含んでいる必要があります。

- テスト・オブジェクト・クラス。
- 時間が経っても変わらないプロパティ。アプリケーションの実行中に変化するプロパティ，またはアプリケーションの異なる実行セッションで異なる値を持つプロパティは，使用しないでください。

たとえば，「**show picture**」というラベルを表示するチェック・ボックスについて考えてみてください。このチェック・ボックスには，次のプロパティがあります。

- **label** プロパティには，画面に表示されるテキストが含まれています。これは言語に依存するため，テスト・オブジェクト記述に使用するプロパティとしては適していません。
- ブール・プロパティの **selected** は，チェック・ボックスが選択されているかどうかを示します。これは値が時々で変化するため，テスト・オブジェクト記述に使用するプロパティとしては適していません。
- このチェック・ボックスの **Name** プロパティは，**show_pic** です。このプロパティはこの特定のチェック・ボックスを一意に識別するため，チェック・ボックスのテスト・オブジェクトの記述に使用できます。

オブジェクトの記述は，「**AppDescription Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。

記述には，テスト・オブジェクト・クラス，テスト・オブジェクト記述の一部であるすべての認識プロパティの名前と値，テスト・オブジェクト名を含める必要があります。記述ではまた，選択したオブジェクトのすべての親を，Testing Agent がサポートしている階層のルートにまで遡って再帰的に記述する必要があります。記述する各親オブジェクトについて，前述のすべての情報を提供する必要があります。

テスト環境の XML では，記述に含めたすべての認識プロパティについて，ForDescription 属性を true に設定します。

詳細については，39 ページ「BuildDescription の実装：オブジェクト ID の記述への割り当て」を参照してください。

スマート認識

場合によっては，テストに格納されている記述がアプリケーション内のオブジェクトを一意に識別しないことがあります。これは，たとえば，ユーザがオブジェクト・リポジトリ内のテスト・オブジェクトの記述プロパティの記述値を手動で変更した場合，またはテストの作成後にアプリケーションを変更した場合に起こる可能性があります。QuickTest ユーザは，テストとともに格納した記述がオブジェクトを一意に認識しなくなっても，QuickTest がアプリケーション内のオブジェクトを認識できるようにスマート認識を設定できます。

格納された記述に一致するオブジェクトがアプリケーション内に複数ある場合にスマート認識をサポートするには，**FindObjectId2** を実装して，複数のオブジェクト ID を返すようにします。

詳細については，次を参照してください。

➤ 40 ページ「FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て」

➤ 45 ページ「スマート認識のサポートのテスト」

外部の親の概念について

テスト対象アプリケーションには，複数の環境のオブジェクトを含めることができます。Testing Agent がサポートするオブジェクトが，ほかの環境のオブジェクトに含まれていることがあります。たとえば，アプリケーションが Web ブラウザ内で実行されていたり，使用しているコントロールが Java オブジェクトに含まれていることがあります。

Testing Agent がオブジェクトの記述を提供する場合，その中には，同じ環境に属するすべての先祖と子孫が含まれています。QuickTest が環境のテスト・オブジェクトをアプリケーションの階層全体に統合できるようにするには，**外部の親**に関する情報を提供する必要があります。

外部の親とは，Testing Agent の階層の最上位のテスト・オブジェクトを含むテスト環境の外にあるオブジェクトです。つまり，テスト・オブジェクトの記述に含まれている最上位の先祖の親オブジェクトです。たとえば，コントロールが Web ブラウザに含まれてい

て，テスト内のステップに階層全体を反映させる場合，Web ブラウザをコントロールの外部の親として指定します。

テスト・ステップに Testing Agent の階層のみを反映させるか，トップレベル・オブジェクトがテスト環境に属している場合は，外部の親を指定する必要はありません。

外部の親は，「**ExternalParent Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。

BuildDescription の実装：オブジェクト ID の記述への割り当て

QuickTest では，オブジェクトの実行時 ID があり，将来使用するために格納しておく対応するテスト・オブジェクト記述が必要な場合, Testing Agent の **BuildDescription** メソッドを呼び出します。

詳細については，36 ページ「オブジェクトの記述」を参照してください。

このメソッドの詳細については，『QuickTest Testing Extensibility API Reference』を参照してください。

GetDisplayName の実装：オブジェクト名の戻し

「**AppDescription Schema**」（『QuickTest Testing Extensibility API Reference』を参照）に従ってテスト・オブジェクト記述を定義する場合，記述にはテスト・オブジェクト名が含まれます。**GetDisplayName** メソッドを実装して，QuickTest がメソッドに渡すオブジェクト ID に基づいてこの名前を返します。

QuickTest は，テスト・オブジェクト名をキーワード・ビュー，エディタ，オブジェクト・リポジトリに表示します。QuickTest ユーザがアプリケーション内でこのテスト・オブジェクトが表すオブジェクトを識別できるように，わかりやすく明確な名前を使用してください。

テスト・オブジェクト名は，オブジェクト認識には使用されません。ただし，同じ親内のテスト・オブジェクト間では一意である必要があります。**GetDisplayName** が同じ親オブジェクトの別の子に使用されている名前を返した場合，QuickTest ではインデックスを付加して一意にします。たとえば，MyButton_1 のようにします。

FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て

QuickTest がテスト内に格納されている記述に基づいてアプリケーション内のオブジェクトを検索する必要がある場合（テストの実行時など），QuickTest は格納されている記述を使用して **FindObjectId2** を呼び出します。このメソッドは，記述に一致するすべてのオブジェクトのオブジェクト ID を返します。これにより，QuickTest では，スマート認識を使用して該当するオブジェクトを特定できます（QuickTest ユーザがそのように設定している場合）。詳細については，38 ページ「スマート認識」を参照してください。

要求されたオブジェクトが見つからない場合，タイムアウトに到達するまで QuickTest は FindObjectId2 を繰り返し呼び出し続け，オブジェクトのロードに数秒かかる場合に対応します。

QuickTest が **FindObjectId2** に渡す記述には，QuickTest ユーザが定義した正規表現を含めることができます。必要に応じて，正規表現を処理する **FindObjectId2** メソッドを実装して，記述に一致するオブジェクトを検索する必要があります。QuickTest での正規表現の使用方法の詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

FindObjectId2 は，**ITestable2 インタフェースで定義されます。FindObjectId2** は，**ITestable.FindObjectId を置換します。FindObjectId2** は，複数の一致を検索する機能と正規表現を処理する機能を追加します。

ITestable2 を実装しない場合，QuickTest は **ITestable::FindObjectId** を呼び出します。この場合，追加した機能は使用できません。**ITestable2** を実装する場合，QuickTest は，常に **ITestable::FindObjectId** が存在するかどうかを確認せずに，**ITestable2::FindObjectId2** を呼び出します。

このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』を参照してください。

GetParent の実装

QuickTest は，**GetParent** を呼び出して，メソッド呼び出しに指定されているオブジェクトを含むオブジェクトを取得します。**GetParent** は，そのオブジェクトがテスト・アプリケーション環境の一部である場合，包括するオブジェクトのオブジェクト ID を返します。包括するオブジェクトが環境の一部でない場合は，外部の親の XML を返します。

このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』を参照してください。

GetChildren の実装

QuickTest は，**GetChildren** を呼び出して，メソッド呼び出しに指定されているオブジェクトが直接含んでいるオブジェクトのオブジェクト ID を取得します。QuickTest は，**GetChildren** を再帰的に呼び出して，オブジェクトの子孫のツリー全体を取得することがあります。メソッドに渡すオブジェクトには，アプリケーションのメンバまたは外部の親を指定できます。

GetChildren は，オブジェクト ID の配列を返します。指定したオブジェクトに子がない場合，**GetChildren** は NULL を返します。

GetChildren メソッドはフィルタ引数も受け取り，QuickTest では，この引数を使用してオブジェクトの子のサブセットのみを要求できます。**GetChildren** メソッドは，そのプロパティ値がフィルタに指定されているすべての値に一致するオブジェクトのすべての子を返す必要があります。

QuickTest は，さまざまな状況で **GetChildren** メソッドを呼び出して，テスト・オブジェクト階層に関する情報を取得します。**GetChildren** メソッドは，テストに ChildObjects() 操作が含まれている場合にも呼び出されます。たとえば，QuickTest テストで次のステップを実行する場合，QuickTest は **GetChildren** メソッドを呼び出して，電卓アプリケーションのボタンを返します。

```
Set desc2 = Description.Create()
desc2("micclass").Value = "WinButton"
Set container2 = Window("Calculator")
Set childObjects2 = container2.ChildObjects(desc2)
MsgBox childObjects2.Count
```

このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』を参照してください。

テストを実行するためのサポートの開発

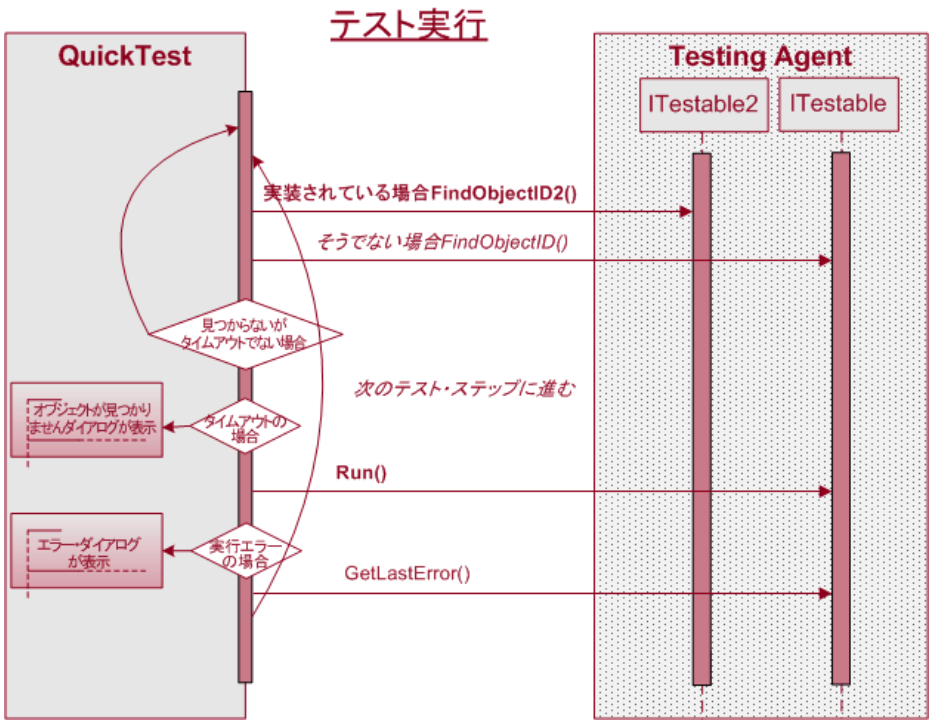
QuickTest がテスト・ステップを実行する場合，最初に **FindObjectId2** を呼び出して，テスト・オブジェクトの格納されている記述に基づいて Testing Agent から該当するオブジェクト ID を取得します（**FindObjectId2** が Testing Agent によって実装されていない場合は，QuickTest が **FindObjectId** を呼び出します）。オブジェクトが見つからないか，**FindObjectId2** から複数のオブジェクト ID が返される場合, QuickTest はスマート認識を使用してアプリケーション内のオブジェクトを認識できます。QuickTest は，QuickTest ユーザが該当するテスト・オブジェクトのスマート認識を設定している場合にのみ，スマート認識を使用します。

次に，QuickTest は，オブジェクト ID，オブジェクト上で実行するテスト・オブジェクト・メソッド，そのテスト・オブジェクト・メソッドの引数を使用して，**Run** メソッドを呼び出します。Testing Agent がテスト対象アプリケーションで該当する操作を実行できるように，**Run** メソッドを実装する必要があります。

QuickTest が意味のあるエラー・メッセージを QuickTest ユーザに提供できるようにするには，**GetLastError** メソッドを実装する必要があります。

次の図は，テストの実行時に QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest は，Testing Agent によって **ITestable2** インタフェースに実装されている **FindObjectId2** を呼び出します。**FindObjectId2** が実装されていない場合，QuickTest は，Testing Agent によって **ITestable** インタフェースに実装されている **FindObjectId** を呼び出します。

この図は，**GetLastError** メソッドの使用方法も示しています。詳細については，44 ページ「GetLastError メソッドの実装：エラーの報告」を参照してください。



Run メソッドの実装

それぞれの環境で開発されたアプリケーションでのテストの実行をサポートするには，**Run** メソッドを実装して，各テスト・オブジェクト・クラスの関連するすべてのテスト・オブジェクト・メソッドをサポートする必要があります。このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

GetLastError メソッドの実装：エラーの報告

Run メソッドが成功以外の結果を返すか，例外をスローする場合，QuickTest は **GetLastError** メソッドを呼び出します。このメソッドを実装して，QuickTest がエラー・メッセージ・ボックスに表示できる文字列を返すようにしてください。理想的には，エラー・メッセージをテストを実行するコンピュータのロケールにします。このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

QuickTest では，一般的ないくつかのエラー（オブジェクトが見つからない，オブジェクトが一意でないなど）について HRESULT エラー・コードを使用します。これらのコードは，**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\include]TestError.h** ファイルで定義されています。メソッドがこれらのエラー・コードの 1 つを返し，**GetLastError** メソッドがエラー・メッセージ文字列を提供しない場合，QuickTest はこのメッセージの標準設定で定義されたエラー・メッセージを表示します。

ほかのエラー・コードについては，**GetLastError** メソッドがエラー・メッセージ文字列を提供しない場合，QuickTest は［**特定できないエラー**］というメッセージを表示します。

テストを実行するためのサポートのテスト

QuickTest がアプリケーション内のオブジェクトを認識するのに必要なメソッドを実装およびテストしたら，Testing Agent の基本的な機能をテストできます。

テストの実行のサポートをテストするには，次の手順で行います。

1. Testing Agent と QuickTest が閉じていることを確認します。
2. 更新済みの Testing Agent をインストールします。
3. QuickTest を開き，Testing Agent が起動されていることを確認します（詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 2 から 4 を参照してください）。
4. テスト環境で定義したタイプのオブジェクトを含むアプリケーションを実行します。
5. QuickTest で，サポートを設計したテスト・オブジェクト上でテスト・オブジェクト・メソッドを実行するテストを作成します。テストの作成の詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 5 を参照してください。
6. 作成したテストを実行し，テストがアプリケーション上で正しくテスト・オブジェクト・メソッドを実行していることを確認します。

7 実行結果のレポートで，テスト・オブジェクト・クラスに使用したアイコンがテスト環境の XML で定義したものであることを確認します。

スマート認識のサポートのテスト

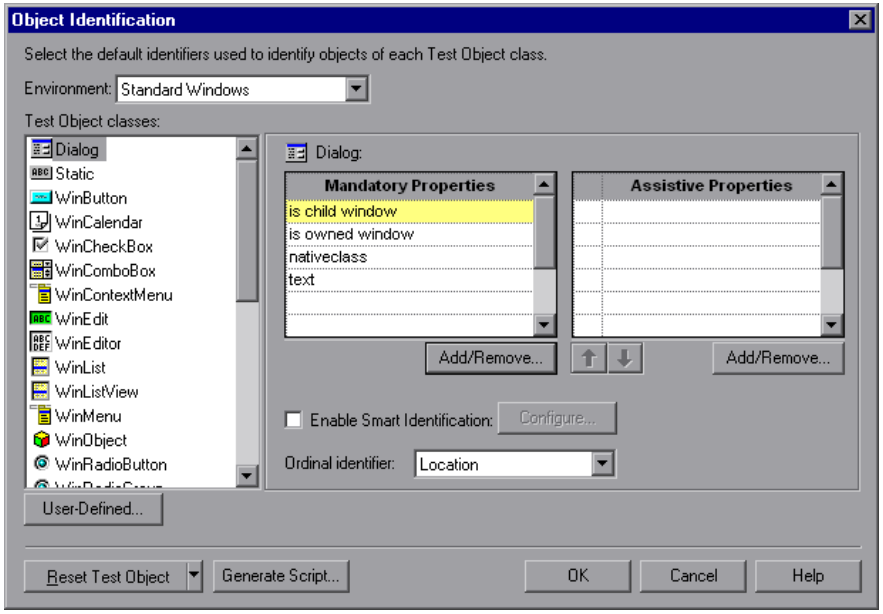
QuickTest がアプリケーション内のオブジェクトを認識するのに必要なメソッドを実装およびテストしたら，スマート認識に関する Testing Agent のサポートをテストできます。

スマート認識のサポートをテストするには，次の手順で行います。

1 QuickTest を開き，Testing Agent が起動されていることを確認します（詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 2 から 4 を参照してください）。

2 スマート認識を設定して，複数のオブジェクトがオブジェクトの記述に一致する場合に，オブジェクト記述の一部でない認識プロパティを使用して，必要なオブジェクトを認識するようにします。

a QuickTest で，［**ツール**］＞［**オブジェクトの認識**］を選択します。［オブジェクトの認識］ダイアログ・ボックスが開きます。



b ［**環境**］リストで，環境を選択します。

テスト環境の XML で定義されたテスト・オブジェクト・クラスが，［**テスト オブジェクト クラス**］リストにアルファベット順に表示されます。

c ［**テスト オブジェクト クラス**］リストで，スマート認識を設定するテスト・オブジェクト・クラスを選択します。

d ［**スマート認識を有効にする**］チェック・ボックスを選択し，［**設定**］をクリックします。［スマート認識プロパティ］ダイアログ・ボックスが開きます。複数のオブジェクトがオブジェクト記述に一致する場合に，テスト・オブジェクト・クラスのオブジェクトの認識に使用できる認識プロパティを追加します。

詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

3 スマート認識を設定したテスト・オブジェクト・クラスの複数のオブジェクトを含むアプリケーションを実行します。これらのオブジェクトの少なくとも 2 つが，テスト・オブジェクト記述を構成する認識プロパティに対してすべて同じ値を共有しているが，スマート認識用に選択した認識プロパティに対しては異なる値を持っていることを確認します。

4 QuickTest で，アプリケーション内の類似のテスト・オブジェクトの 1 つでステップを実行するテストを作成します。スマート認識に使用されるプロパティの値を定義して，アプリケーション内の正しいオブジェクトの認識に使用できるようにします。

テストを作成するには，次のいずれかの方法を使用できます。

- オブジェクト・リポジトリ・エディタで新しいテスト・オブジェクトを定義し，キーワード・ビューまたはステップ・ジェネレータのステップでこれらのテスト・オブジェクトを使用します。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。
- エディタでテストを作成し，プログラム的記述を使用してテスト・オブジェクトを定義します。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。
- テストを記録するためのサポートを作成後にこのテストを実行する場合，テストを記録してから，テスト・オブジェクト（オブジェクト・リポジトリ内）の記述プロパティの値を手動で変更し，アプリケーション内の複数のオブジェクトと一致するようにできます。

5 テストを実行します。QuickTest は，記述プロパティに基づいて，アプリケーション内のテストに記述されているオブジェクトに一致する複数のオブジェクトを検索します。このため，QuickTest はスマート認識メカニズムを使用して，該当するオブジェクトを認識しステップを実行します。スマート認識がいつ使用されたかを示す実行結果を確認することで，これを確認できます。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

認識プロパティの実行時の値を取得するためのサポートの開発

QuickTest でテストを実行する場合，ステップを実行してそれが失敗しないことを確認するだけでは，通常不十分です。テストのさまざまな段階でテスト・オブジェクトのさまざまな認識プロパティの値を確認して，アプリケーションが期待どおりに機能していることを確認する必要もあります。

このため，QuickTest では，さまざまな認識プロパティの実行時の値を取得できる必要があります。QuickTest は，標準のさまざまなテスト・オブジェクト・メソッド（**GetROProperty** など）で認識プロパティの実行時の値を使用します。認識プロパティの実行時の値は，チェックポイントと出力値の作成機能など，さまざまな基本機能を実行する際にも必要になります。

QuickTest がアプリケーションから認識プロパティの実行時の値を取得できるようにするには，**GetElementType** メソッドと **GetProperties** メソッドを実装する必要があります。

記録セッション時のチェックポイントの作成をサポートするには，**ISpyable** インタフェースも実装する必要があります。詳細については，65 ページ「記録セッション中のチェックポイントの作成のサポート」を参照してください。

環境にテーブルが含まれている場合，**GetTabularData** と **GetTabularAttributes** を実装すると，QuickTest がテーブル関連データを取得できるようにできます。これらのメソッドの実装に加えて，QuickTest がテーブルとして処理するテスト・オブジェクト・クラスを指定する必要があります。これを行うには，テスト環境の XML で該当するテスト・オブジェクト・クラスの **SupportsTabularData** 属性を true に設定します。

環境内のオブジェクトが COM オブジェクトである場合，テスト環境の XML で定義した認識プロパティの実行時の値を取得できるだけでなく，COM 実行時オブジェクトのネイティブの操作およびプロパティにアクセスできるように QuickTest を設定できます。詳細については，62 ページ「ネイティブ（実行時オブジェクト）のプロパティとメソッドにアクセスするためのサポートの開発」を参照してください。

GetElementType と GetProperties の実装

指定したオブジェクトのテスト・オブジェクト・クラスを返すには，**GetElementType** メソッドを実装します。指定したプロパティの値を含む配列を入力プロパティ名配列と同じ順序で返すには，**GetProperties** メソッドを実装します。

これらのメソッドの構文の詳細については,『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

QuickTest テストでは，ユーザは，プロパティ名として指定した任意の文字列を使用して **GetROProperty** メソッドを実行するステップを作成できます。このテスト・ステップを実行すると，QuickTest が同じ文字列を使用して **GetProperties** を呼び出します。このため, **GetProperties** では, 認識できないプロパティに対して空の値を返す必要があります。

注：

- **IRectangleSupplier** を実装すると，QuickTest では，**GetRectangle** 関数から返される値から **abs_x**，**abs_y**，**width**，**height** の各プロパティの値を推測し，これらのプロパティを **GetProperties** 呼び出しに渡しません。
- 特定のテスト・オブジェクト・クラス上でビットマップ・チェックポイントをサポートするには，そのテスト・オブジェクト・クラスの **IRectangleSupplier** 認識プロパティを実装するか，**width** および **height** の各認識プロパティを実装する必要があります。

  テスト・オブジェクト・クラスのビジュアル関係識別子機能をサポートするには，そのテスト・オブジェクト・クラスの **IRectangleSupplier** 認識プロパティを実装するか，**abs_x**，**abs_y**，**width**，**height** の各認識プロパティを実装する必要があります。

  これらのいずれかの機能に対して後者のオプションを選択する場合，テスト環境の XML のこのテスト・オブジェクト・クラスの定義に **width** および **height**（および **abs_x**，**abs_y**）の各認識プロパティを含め，**GetProperties** メソッドを実装して，関連する値を返す必要があります。

GetTabularAttributes と GetTabularData の実装

指定したテーブルのサイズ，最初と最後の可視の行数を返すには，**GetTabularAttributes** メソッドを実装します。

テーブル・カラムのタイトルを含む配列，および指定した行のセル・データの 2 次元の配列を返すには，**GetTabularData** メソッドを実装します。

QuickTest ユーザがテーブル・チェックポイントを作成するか，テーブル出力値ステップを追加すると，QuickTest は **GetTabularAttributes** を呼び出して，可視の行数を取得し，**GetTabularData** を呼び出します。

これらのメソッドの構文の詳細については,『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

プロパティ値を取得するためのサポートのテスト

QuickTest がテスト・オブジェクトの認識プロパティの実行時の値を取得するのに必要なメソッドを実装したら，QuickTest テストの **GetROProperty** メソッドを使用して，Testing Agent のこの機能をテストできます。

プロパティ値を取得するためのサポートをテストするには，次の手順で行います。

1 44 ページ「テストを実行するためのサポートのテスト」の手順 1 から 5 を繰り返して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを使用して QuickTest テストを作成します。

2 ステップをテストに追加し，環境のオブジェクトで **GetROProperty** を呼び出して，取得した認識プロパティの値をメッセージ・ボックスに表示します。

GetROProperty メソッドの詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』および『HP Unified Functional Testing Object Model Reference』を参照してください。

3 作成したテストを実行し，メッセージ・ボックスに正しい認識プロパティの値が表示されることを確認します。

チェックポイントを作成するためのサポートのテスト

QuickTest がテスト・オブジェクトの認識プロパティの実行時の値を取得するのに必要なメソッドを実装したら, Testing Agent によるチェックポイントのサポート方法をテストできます。

チェックポイントを作成するためのサポートをテストするには，次の手順で行います。

1 チェックポイントを作成するオブジェクトが見えるように，正しいコンテキストでアプリケーションが開いていることを確認します。

2 44 ページ「テストを実行するためのサポートのテスト」の手順 1 から 5 を繰り返して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを使用して QuickTest テストを作成します。

3 チェックポイントを追加するステップを右クリックし，［**標準チェックポイントの挿入**］を選択します。［チェックポイントのプロパティ］ダイアログ・ボックスが表示されます。

テスト環境の XML で **ForVerification** 属性を指定した認識プロパティがダイアログ・ボックスに表示されていることを確認します。

テスト環境の XML で **ForDefaultVerification** 属性を指定した認識プロパティが標準設定で選択されていることを確認します。

4 チェックポイントの詳細を定義し，［**OK**］をクリックしてチェックポイントを挿入します。チェックポイントのオプションの詳細については,『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

5 作成したテストを実行し, チェックポイントを正しく実行していることを確認します。実行結果のレポートを確認し，テスト・オブジェクト・クラスに使用したアイコンがテスト環境の XML で定義したものであることを確認します。

テーブル・チェックポイントを作成するためのサポートをテストするには, 手順 3 のテーブル・オブジェクトを使用して上記のテストを実行し,［チェックポイントのプロパティ］ダイアログ・ボックスにテーブルの内容とプロパティが正しく表示されることを確認します。

出力値ステップのサポートのテスト

QuickTest がテスト・オブジェクトの認識プロパティの実行時の値を取得するのに必要なメソッドを実装したら, Testing Agent による出力値ステップのサポート方法をテストできます。

出力値ステップのサポートをテストするには，次の手順で行います。

1 値を出力するオブジェクトが見えるように，正しいコンテキストでアプリケーションが開いていることを確認します。

2 44 ページ「テストを実行するためのサポートのテスト」の手順 1 から 5 を繰り返して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを使用して QuickTest テストを作成します。

3 出力値ステップを追加するステップを右クリックし，［**出力値の挿入**］を選択します。［データベース出力値のプロパティ］ダイアログ・ボックスが開きます。

テスト環境の XML で **ForVerification** 属性を指定した認識プロパティがダイアログ・ボックスに表示されていることを確認します。

4 出力値の詳細を定義し，［**OK**］をクリックして出力値ステップを追加します。出力値のオプションの詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

5 作成したテストを実行し，テスト実行結果の［実行時データ テーブル］に正しい値が含まれていることを確認します。

テーブル出力値ステップのサポートをテストするには，手順 3 のテーブル・オブジェクトを使用して上記のテストを実行し，［出力値のプロパティ］ダイアログ・ボックスと実行結果の［実行時データ テーブル］にテーブルの内容とプロパティが正しく表示されることを確認します。

オブジェクトを学習するためのサポートの開発

QuickTest ユーザが［**オブジェクトの追加**］ツールを使用してオブジェクトをオブジェクト・リポジトリに追加し，オブジェクトをポイントすると，QuickTest はこのオブジェクトを学習し，オブジェクト・リポジトリに追加します。指定したオブジェクトにほかのオブジェクトが含まれている場合，それらのオブジェクトも指定したオブジェクトの子孫として学習できます。QuickTest ユーザは，子孫として学習する必要があるオブジェクトのタイプを指定できます。

ユーザがポイントするオブジェクトのオブジェクト ID の取得をサポートするには，**ISpyable** インタフェースから **GetElementFromPoint** メソッドを実装する必要があります。詳細については，59 ページ「オブジェクト・スパイのサポートの開発」を参照してください。

QuickTest が環境内のオブジェクトを学習し，それらをオブジェクト・リポジトリに追加できるようにするには，**LearnChildObjects** および **CompareObjectIds** メソッドを実装する必要があります。

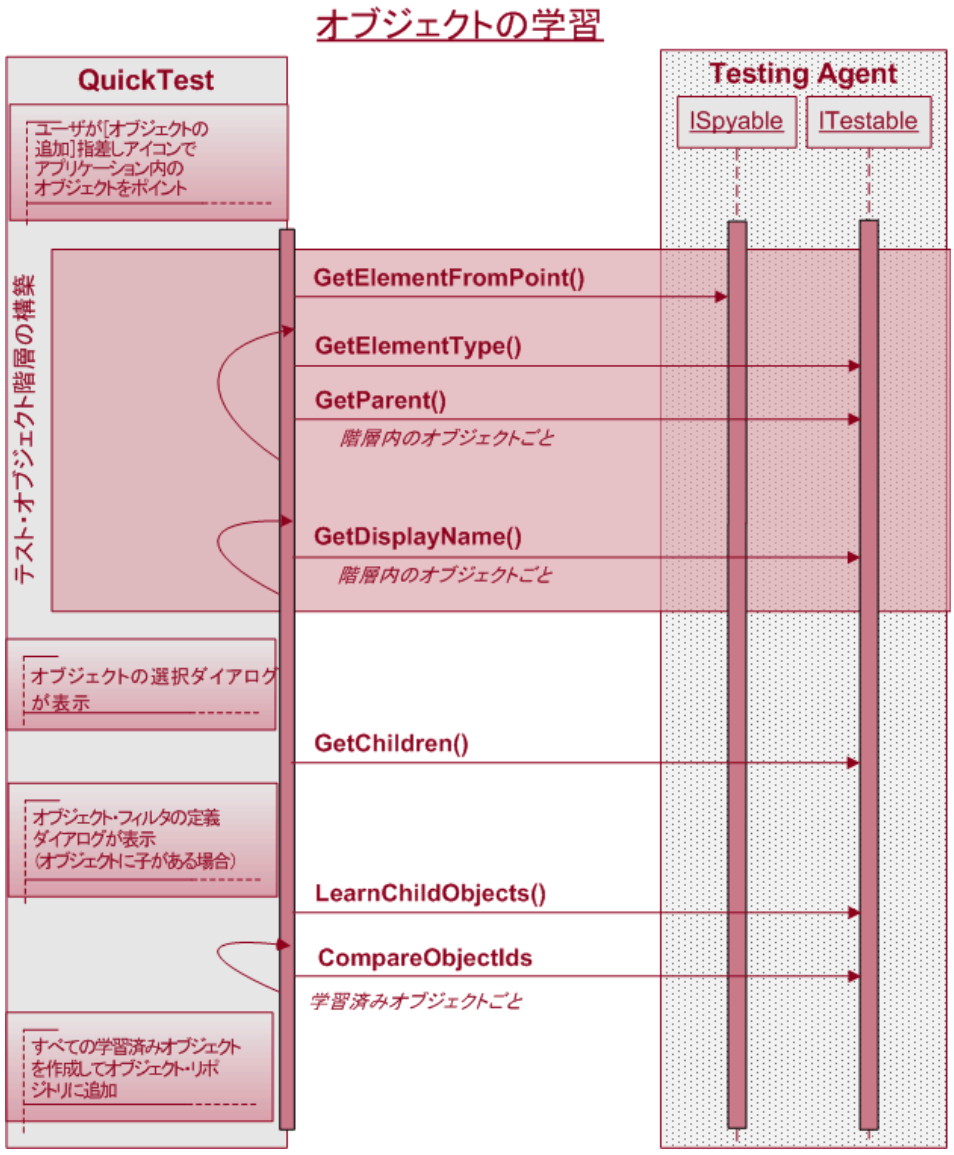
イベントのオブジェクト学習チェーンについて

オブジェクトを学習する場合，QuickTest は最初にテスト・オブジェクト階層を構築します。次に，ユーザが複数のオブジェクトに関連付けられている場所をクリックすると，［オブジェクトの選択］ダイアログ・ボックスが開きます。ユーザが必要なオブジェクトを選択した後，QuickTest は空のフィルタ引数（つまり，XML ルート要素の **<Filter/>** のみ含む）を使用して，**GetChildren** メソッドを呼び出します。これには，Testing Agent が指定したオブジェクトの直接のすべての子を返す必要があります。

指定したオブジェクトに子がある場合，QuickTest で［オブジェクト フィルタの定義］ダイアログ・ボックスが開き，ユーザは学習する子孫のタイプを選択できます。

次に，QuickTest は **LearnChildObjects** メソッドを呼び出して，指定したオブジェクトのオブジェクト ID と記述，そのすべての先祖と子孫を取得します。

次の図は，オブジェクトの学習時に QuickTest が呼び出すメソッドの一般的なシーケンスを示します。この図は，Testing Agent が実装する必要があるメソッドの目的，これらのメソッドが属しているインタフェースを示しています。たとえば，QuickTest は，**ISpyable** インタフェースのメンバである **GetElementFromPoint** を呼び出します。指定したオブジェクトのオブジェクト ID の取得後，QuickTest は，階層内の各オブジェクトに対して **GetElementType** と **GetParent** を呼び出します。**GetElementType** と **GetParent** は，**ITestable** インタフェースのメンバです。


オブジェクトの学習
QuickTest
Testing Agent
ISpyable
ITestable
ユーザが[オブジェクトの追加]指差しアイコンでアプリケーション内のオブジェクトをポイント
テスト・オブジェクト階層の構築
GetElementFromPoint()
GetElementType()
GetParent()
階層内のオブジェクトごと
GetDisplayName()
階層内のオブジェクトごと
オブジェクトの選択ダイアログが表示
GetChildren()
オブジェクト・フィルタの定義ダイアログが表示(オブジェクトに子がある場合)
LearnChildObjects()
CompareObjectIds
学習済みオブジェクトごと
すべての学習済みオブジェクトを作成してオブジェクト・リポジトリに追加

LearnChildObjects の実装

指定したオブジェクトの全階層を構成するすべてのオブジェクトのオブジェクト ID を含む配列を返すには，**LearnChildObjects** メソッドを実装します。この中には，オブジェクトの先祖（サポートしているテスト環境のスコープ内にある最上位のオブジェクトまで），指定したオブジェクト自身，オブジェクトのすべての子孫（存在する場合）が含まれます。

また，このメソッドは **learnChildrenXml** 引数を設定する必要があり，**externalParent** 引数を設定することもできます。

このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

CompareObjectIds の実装

メソッドに渡される 2 つのオブジェクト ID が同じオブジェクトを参照しているかどうかを確認するには，**CompareObjectIds** メソッドを実装します。

QuickTest がオブジェクトのすべての子孫をオブジェクト・リポジトリに追加する場合，QuickTest は各オブジェクトのオブジェクト ID を既存の ID（同じ親に属している）と比較して，そのオブジェクトが以前に追加されていないことを確認します。オブジェクト ID が単純な文字列または整数である場合，QuickTest はメソッドを呼び出さずにこれらを比較します。それ以外の場合，Testing Agent が比較を実行するための **CompareObjectIds** メソッドを呼び出します。このメソッドを実装して，両方のオブジェクト ID が同じオブジェクトを参照している場合に true を返すようにします。

このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

オブジェクトを学習するためのサポートのテスト

ITestable インタフェースですべてのメソッド，**ISpyable** インタフェースで **GetElementFromPoint** メソッドを実装したら，Testing Agent の基本的な機能の包括的テストを実行できます。

このテストを実行するには，QuickTest の［**オブジェクトの追加**］オプションを使用してオブジェクトをオブジェクト・リポジトリに追加し，これらのオブジェクトを使用してテストを作成し，アプリケーションでテストを実行します。

オブジェクトを学習するためのサポートをテストするには，次の手順で行います。

1 44 ページ「テストを実行するためのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを含むアプリケーションを実行します。

2 QuickTest で，次のいずれかを行います。

- ➤［オブジェクト リポジトリ］ウィンドウで，［**オブジェクト**］＞［**ローカルへオブジェクトを追加**］を選択するか，［**ローカルへオブジェクトを追加**］ツールバー・ボタンをクリックします。

- ➤ オブジェクト・リポジトリ・マネージャで，［**オブジェクト**］＞［**オブジェクトの追加**］を選択するか，［**オブジェクトの追加**］ツールバー・ボタンをクリックします。

QuickTest と［オブジェクト リポジトリ］ウィンドウまたはオブジェクト・リポジトリ・マネージャが最小化され，ポインタが指さし型に変わります。

3 オブジェクト・リポジトリに追加するオブジェクトをクリックします。

4 クリックした場所が複数のオブジェクトに関連付けられている場合は，［オブジェクトの選択］ダイアログ・ボックスが開きます。リポジトリに追加するオブジェクトを選択し，［**OK**］をクリックします。

5 選択したオブジェクトにほかのオブジェクトが含まれている場合，［オブジェクト フィルタの定義］ダイアログ・ボックスが開きます。［**すべてのオブジェクト タイプ**］を選択し，［**OK**］をクリックします。

選択したオブジェクトがそのすべての子孫とともにオブジェクト・リポジトリに追加されます。[オブジェクト リポジトリ] ウィンドウが更新され，新規テスト・オブジェクト，それらのプロパティと値が表示されます。

また，オブジェクト・リポジトリに新規オブジェクトの親オブジェクトが存在しない場合は，QuickTest によって親オブジェクトが追加されます。

詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

6 オブジェクト・リポジトリで，オブジェクトが正しく追加されていることを確認します。オブジェクトの親と子で，階層が正しく定義されていることを確認します。テスト・オブジェクト・クラスとアイコンがテスト環境の XML で定義したものであることを確認します。正しい認識プロパティが記述プロパティとしてリストされていることを確認します。

7 オブジェクト・リポジトリに追加したオブジェクト上で操作を実行するテストを作成します。このオブジェクトに対して正しいテスト・オブジェクト・メソッドを使用できることを確認します。

8 作成したテストを実行し，正しく実行されることを確認します。

第 4 章

QuickTest Testing Agent の機能拡張

第 3 章,「基本的な QuickTest Testing Agent の開発」の説明に従って，**ITestable** インタフェースですべてのメソッドを実装したら，QuickTest の追加機能をサポートするように Testing Agent を拡張できます。

本章の内容

QuickTest Testing Agent の機能拡張について

必須の **ITestable** インタフェースの実装が完了したら，SDK 内のその他のインタフェースを実装して，QuickTest のより高度な機能（記録，ActiveScreen，強調表示，チェックポイントの作成など）をサポートできます。これらのインタフェースは，**QuickTest Extensibility Agent** type libraryで定義されています。

次の表は，QuickTest の各機能に対して実装する必要があるインタフェースと各インタフェースに含まれているメソッドを示します。

サポートされる QuickTest 機能	実装するインタフェース	API メソッド
オブジェクト・スパイ	**ISpyable**	**GetElementFromPoint**
ネイティブのプロパティと操作へのアクセス	**IRunTimeObjectSupplier**	**GetRunTimeObject**
記録	**IRecordable**	**BeginRecording** **EndRecording**
ActiveScreen	**IActiveScreenSupplier**, **IHWNDSupplier**	**GetActiveScreen** **GetHWND**
強調表示	**IRectangleSupplier**, **IHWNDSupplier**	**GetRectangle** **GetHWND**
ビットマップ・チェックポイント	**IRectangleSupplier** または **ITestable**（**GetProperties** : **width** および **height** プロパティ）	**GetRectangle** または **GetProperties**（**width** および **height**）
ビジュアル関係識別子	**IRectangleSupplier** または **ITestable**（**GetProperties** : **abs_x**，**abs_y**，**width**，**height** プロパティ）	**GetRectangle** または **GetProperties**（**abs_x**，**abs_y**，**width**，**height**）
テスト・イベント通知（QuickTest による Testing Agent への通知）	**ITestEvents**	**BeginRun** **EndRun** **PauseRun** **ResumeRun** **TestHasBeenClosed** **Disconnect**

これらのインタフェースの詳細については,『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

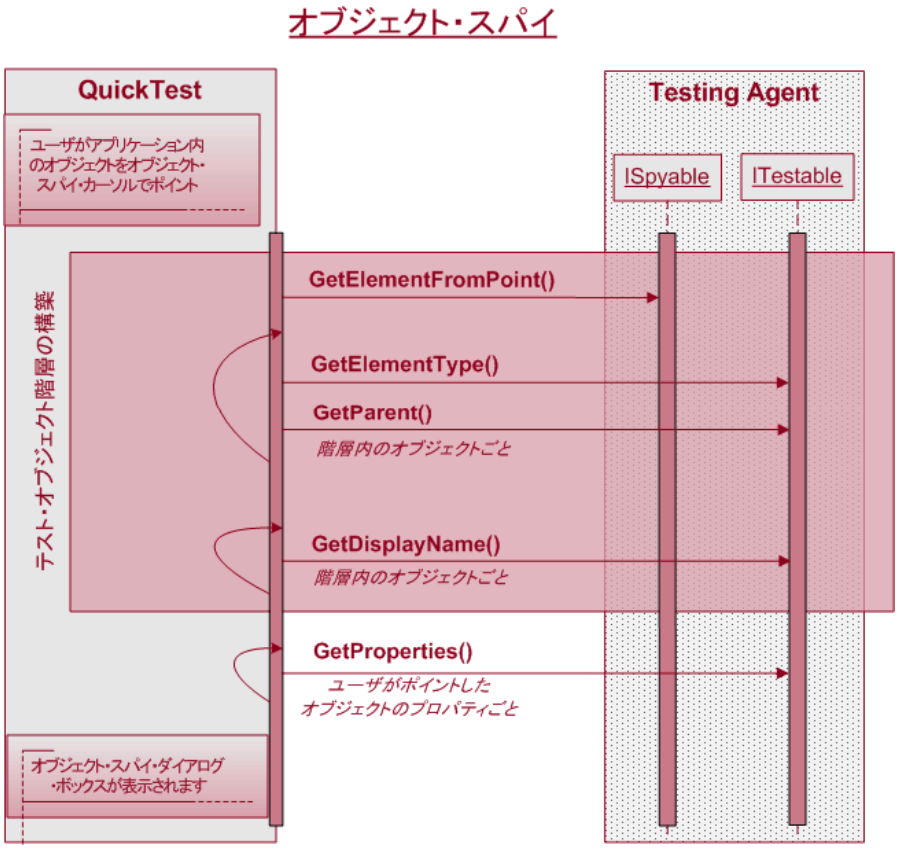
開発は段階的に実行し，一度に 1 つの QuickTest 機能のサポートを追加し，各段階の後にテストを行います。

オブジェクト・スパイのサポートの開発

オブジェクト・スパイの指差しメカニズムを使用すると，QuickTest ユーザは，開いているアプリケーション内の任意の可視なオブジェクトのサポートされているプロパティとメソッドを表示できます。この QuickTest 機能をサポートするには，Testing Agent が，QuickTest が指定する画面座標にあるアプリケーション内のオブジェクトのオブジェクト ID を提供する必要があります。これを行うには，**ISpyable** インタフェースで **GetElementFromPoint** メソッドを実装します。

オブジェクト・スパイでは，環境内のオブジェクトが COM オブジェクトである場合，アプリケーション内のオブジェクトのネイティブの操作およびプロパティを表示することもできます。詳細については，62 ページ「ネイティブ（実行時オブジェクト）のプロパティとメソッドにアクセスするためのサポートの開発」を参照してください。

次の図は，オブジェクト・スパイの使用時に QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest が **GetElementFromPoint**（Testing Agent によって **ISpyable** インタフェースに実装されている）を呼び出し，次に **ITestable** インタフェースから **GetElementType** を呼び出します。



GetElementFromPoint の実装

QuickTest ユーザがポインタでクリックした場所の指定された x 座標と y 座標にある可視なオブジェクトのオブジェクト ID を返すには，**GetElementFromPoint** メソッドを実装します。このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

オブジェクト・スパイのサポートのテスト

GetElementFromPoint メソッドを実装したら，環境内のオブジェクトを識別するオブジェクト・スパイの機能をテストできます。

オブジェクト・スパイのサポートをテストするには，次の手順で行います。

1 Testing Agent と QuickTest が閉じていることを確認します。

2 更新済みの Testing Agent をインストールします。

3 QuickTest を開き，Testing Agent が起動されていることを確認します（詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 2 から 4 を参照してください）。

4 テスト環境で定義したタイプのオブジェクトを含むアプリケーションを実行します。

5 QuickTest で，オブジェクト・スパイを開き，指差しボタンをクリックします。アプリケーション内の環境からオブジェクトをクリックします。

6 QuickTest がオブジェクトを正しく識別することを確認します。次の内容を確認してください。

- ➤ オブジェクト・スパイが正しいテスト・オブジェクト階層を表示している。
- ➤ クリックしたテスト・オブジェクトが正しい親の下に表示されている。
- ➤ QuickTest がテスト・オブジェクト・クラスを正しく識別し，このテスト・オブジェクト・クラスの正しいアイコンを表示している（テスト環境の XML で定義済みの場合）。
- ➤ オブジェクト・スパイがテスト環境の XML で定義したすべてのテスト・オブジェクト・メソッドおよび認識プロパティを表示している。

ネイティブ（実行時オブジェクト）のプロパティとメソッドにアクセスするためのサポートの開発

テスト環境の XML では，認識プロパティとテスト・オブジェクト・メソッドを定義します。Testing Agent で適切なサポートを提供すると，QuickTest はこれらのプロパティとメソッドを認識，学習，記録します。

環境内のオブジェクトが COM オブジェクトである場合，QuickTest が COM 実行時オブジェクトのネイティブのメソッドとプロパティにもアクセスできるようにできます。この機能をサポートするには, Testing Agent が QuickTest にオブジェクトの IDispatch ポインタを提供する必要があります。

これを行うには，**IRunTimeObjectSupplier** インタフェースで **GetRunTimeObject** メソッドを実装します。また，テスト環境の XML の関連するテスト・オブジェクト・クラスの定義に **Object** プロパティを含める必要があります。詳細については，62 ページ「GetRunTimeObject の実装と Object プロパティの定義」を参照してください。

ネイティブのプロパティと操作（実行時オブジェクトのプロパティとメソッドとも呼ばれます）にアクセスするためのサポートを実装すると，次のことが実現します。

- QuickTest がネイティブのプロパティとメソッドをオブジェクト・スパイに表示します。
- QuickTest ユーザは, ステップ・ジェネレータのネイティブの操作にアクセスできます。
- QuickTest ユーザは，**Object** プロパティを使用してプロパティと操作にアクセスできます。

QuickTest でネイティブのプロパティと操作にアクセスする方法の詳細については,『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

GetRunTimeObject の実装と Object プロパティの定義

指定したオブジェクト ID に対応する実行時 COM オブジェクトの IDispatch ポインタを返すには，**GetRunTimeObject** メソッドを実装します。このメソッドの構文の詳細については,『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

GetRunTimeObject は，**IRunTimeObjectSupplier** インタフェース内のただ 1 つのメソッドです。環境内のオブジェクトが COM オブジェクトである（または COM オブジェクトで表されるか，インタフェースされる）場合にのみ，このインタフェースを実装します。

GetRunTimeObject を実装する場合，テスト環境の XML 内の関連するテスト・オブジェクト・クラスに対して **Object** プロパティを定義する必要もあります。**Operation** 要素の **Object** プロパティは，次のように定義します。

```
<Operation Name="Object" PropertyType="Property_Get">
        <ReturnValueType>
            <Type VariantType="IDispatch"/>
        </ReturnValueType>
</Operation>
```

テスト環境の XML の構文については，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

ネイティブのプロパティと操作にアクセスするためのサポートのテスト

GetRunTimeObject メソッドを実装したら，オブジェクト・スパイとテストで，ネイティブのプロパティと操作にアクセスするためのサポートをテストできます。

ネイティブのプロパティと操作にアクセスするためのサポートをテストするには，次の手順で行います。

1. Testing Agent と QuickTest が閉じていることを確認します。
2. 更新済みの Testing Agent をインストールします。
3. QuickTest を開き，Testing Agent が起動されていることを確認します（詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 2 から 4 を参照してください）。
4. Testing Agent がネイティブの操作とプロパティへのアクセスをサポートしているオブジェクトを含むアプリケーションを実行します。
5. QuickTest で，オブジェクト・スパイを開き，指差しボタンをクリックします。アプリケーション内のオブジェクトをクリックします。
6. オブジェクト・スパイに正しいネイティブのプロパティと操作が表示されることを確認します。

7 QuickTest で，Object プロパティを使用してネイティブの操作にアクセスするテストを作成します。テストの作成の詳細については，32 ページ「QuickTest がテスト環境を認識することのテスト」の手順 5 を参照してください。

8 作成したテストを実行し，テストがアプリケーション上で正しく操作を実行していることを確認します。

記録セッション中のチェックポイントの作成のサポート

記録セッション中のチェックポイントの作成のサポートを開発するには，次のメソッドを実装します。

- ユーザがアプリケーションでポイントするオブジェクトを QuickTest が識別できるようにするメソッド
- QuickTest が認識プロパティの実行時の値を取得できるようにするメソッド

59 ページ「オブジェクト・スパイのサポートの開発」では，ユーザがアプリケーションでポイントするオブジェクトを QuickTest が参照するために必要な，**GetElementFromPoint** メソッドを実装しました。

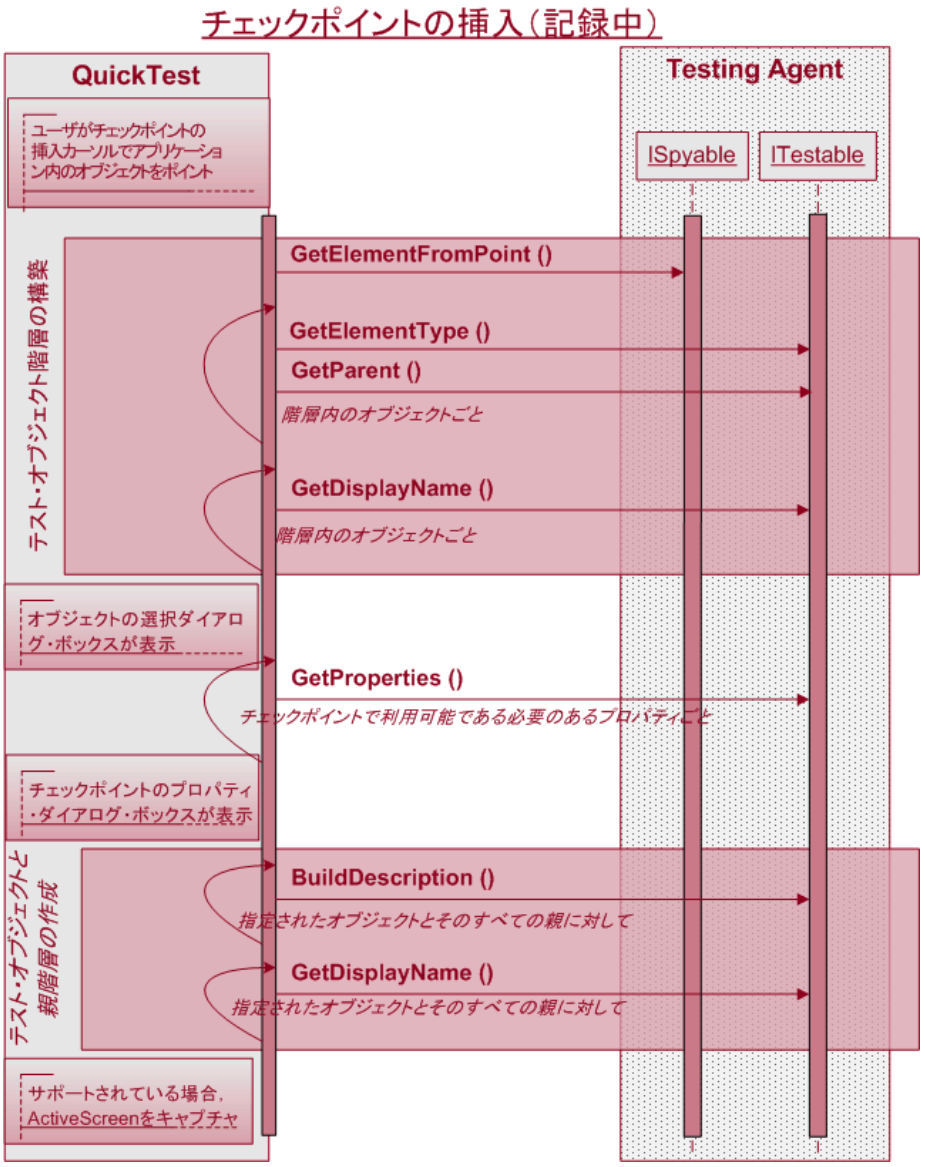
35 ページ「テスト・オブジェクト・モデルのサポートの開発」では，QuickTest がオブジェクトを認識し，テスト・オブジェクト階層を構築するのに必要なメソッドを実装しました。

47 ページ「認識プロパティの実行時の値を取得するためのサポートの開発」では，テスト・オブジェクトの認識プロパティの実行時の値を取得するのに必要なメソッドを実装しました。

これらの項で説明したメソッドを実装することで，記録セッション中のチェックポイントの作成のサポートを提供しました。この機能をサポートするために，**IRecordable** インタフェースを実装する必要はありません。

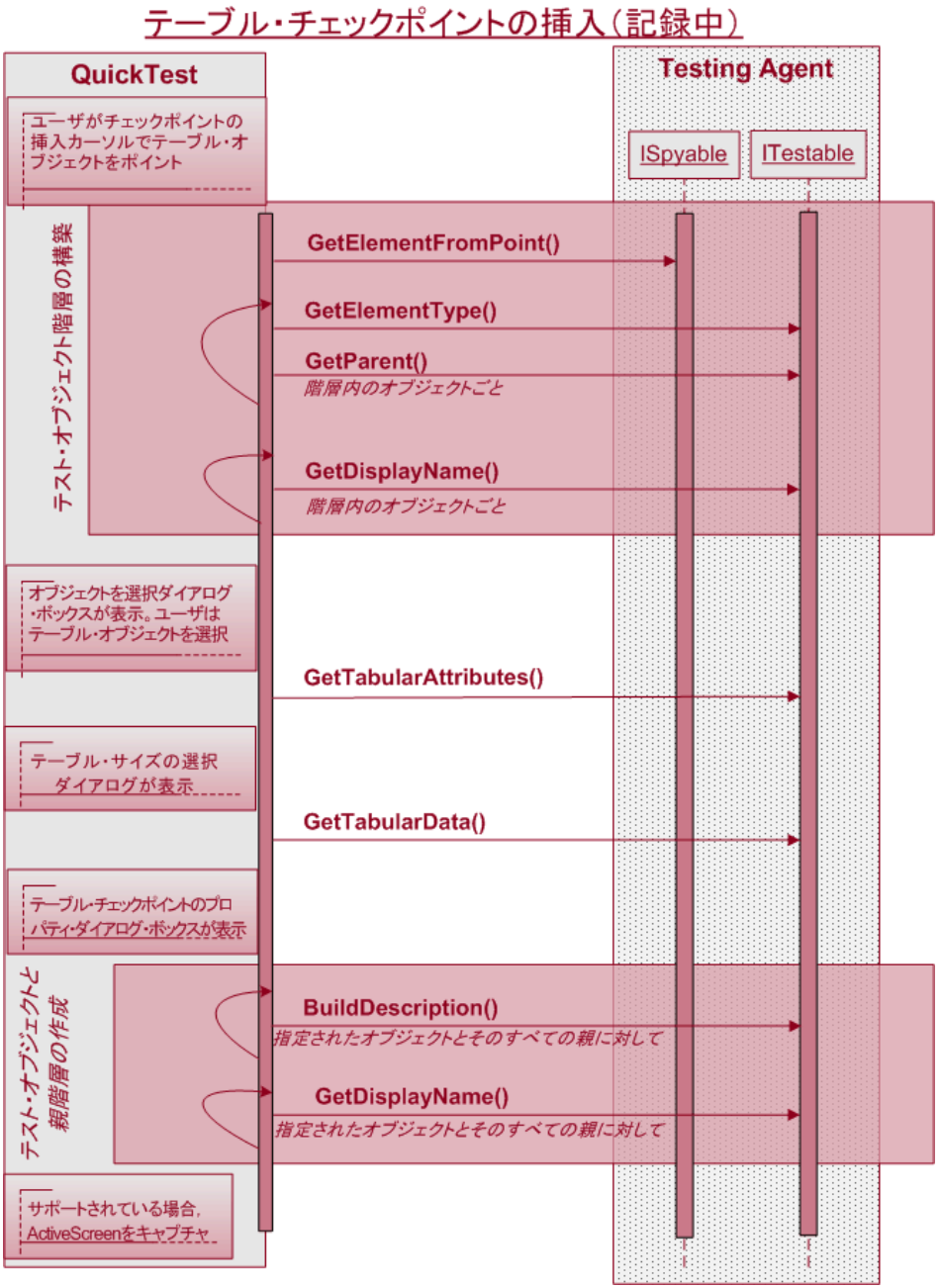
ヒント：ActiveScreen オプションのサポートを作成する場合，ActiveScreen キャプチャはチェックポイントとともに保存できます。詳細については，80 ページ「画面キャプチャのサポートの開発」を参照してください。

次の図は，記録セッション中にユーザがチェックポイントを作成するときに，QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest が最初に **ISpyable** インタフェースから **GetElementFromPoint** を呼び出し，次に **ITestable** インタフェースからメソッドを呼び出すのがわかります。QuickTest は，**GetProperties** を呼び出して認識プロパティの実行時の値を取得します。これらの値は，テスト環境の XML の定義に従ってチェックポイントで確認する必要があります。これらの値は，テストの実行時に取得される値と比較されます。


チェックポイントの挿入（記録中）
QuickTest
Testing Agent
ISpyable
ITestable
ユーザがチェックポイントの挿入カーソルでアプリケーション内のオブジェクトをポイント
テスト・オブジェクト階層の構築
GetElementFromPoint ()
GetElementType ()
GetParent ()
階層内のオブジェクトごと
GetDisplayName ()
階層内のオブジェクトごと
オブジェクトの選択ダイアログ・ボックスが表示
GetProperties ()
チェックポイントで利用可能である必要のあるプロパティごと
チェックポイントのプロパティ・ダイアログ・ボックスが表示
テスト・オブジェクトと親階層の作成
BuildDescription ()
指定されたオブジェクトとそのすべての親に対して
GetDisplayName ()
指定されたオブジェクトとそのすべての親に対して
サポートされている場合，ActiveScreenをキャプチャ

記録セッション中のテーブル・チェックポイントの作成

次の図は，記録セッション中にユーザがテーブル・チェックポイントを作成するときに，QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest が最初に **ISpyable** インタフェースから **GetElementFromPoint** を呼び出し，次に **ITestable** インタフェースからメソッドを呼び出します。テーブルとその内容に関して必要な情報を取得するために，QuickTest は **GetTabularAttributes** と **GetTabularData** を呼び出します。


テーブル・チェックポイントの挿入（記録中）
QuickTest
Testing Agent
ISpyable
ITestable
ユーザがチェックポイントの挿入カーソルでテーブル・オブジェクトをポイント
テスト・オブジェクト階層の構築
GetElementFromPoint()
GetElementType()
GetParent()
階層内のオブジェクトごと
GetDisplayName()
階層内のオブジェクトごと
オブジェクトを選択ダイアログ・ボックスが表示。ユーザはテーブル・オブジェクトを選択
GetTabularAttributes()
テーブル・サイズの選択ダイアログが表示
GetTabularData()
テーブル・チェックポイントのプロパティ・ダイアログ・ボックスが表示
テスト・オブジェクトと親階層の作成
BuildDescription()
指定されたオブジェクトとそのすべての親に対して
GetDisplayName()
指定されたオブジェクトとそのすべての親に対して
サポートされている場合，ActiveScreenをキャプチャ

記録セッション中にチェックポイントを作成するためのサポートのテスト

記録セッション中のチェックポイントの作成をサポートするのに必要なメソッドを実装したら，Testing Agent のこの機能をテストできます。

チェックポイントを作成するためのサポートをテストするには，次の手順で行います。

1 61 ページ「オブジェクト・スパイのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを含むアプリケーションを実行します。

2 QuickTest で記録セッションを開始します。［**デザイン**］＞［**チェックポイント**］＞［**標準チェックポイント**］を選択するか，［**チェックポイントまたは出力値の挿入**］ボタンをクリックします。QuickTest ウィンドウが最小化され，ポインタが指差し型に変わります。

3 テスト対象アプリケーション内のオブジェクトをクリックします。［チェックポイントのプロパティ］ダイアログ・ボックスが表示されます。

テスト環境の XML で **ForVerification** 属性を指定した認識プロパティがダイアログ・ボックスに表示されていることを確認します。

テスト環境の XML で **ForDefaultVerification** 属性を指定した認識プロパティが標準設定で選択されていることを確認します。

4 適合するチェックポイントの詳細を指定します。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。［**OK**］をクリックすると，チェックポイントがテストに挿入されます。

5 作成したテストを実行し，チェックポイントを正しく実行していることを確認します。実行結果のレポートを確認し，テスト・オブジェクト・クラスに使用したアイコンがテスト環境の XML で定義したものであることを確認します。

テーブル・チェックポイントを作成するためのサポートをテストするには，次の手順で行います。

1 61 ページ「オブジェクト・スパイのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを含むアプリケーションを実行します。

2 QuickTest で記録セッションを開始します。［**デザイン**］＞［**チェックポイント**］＞［**標準チェックポイント**］を選択するか，［**チェックポイントまたは出力値の挿入**］ボタンをクリックします。QuickTest ウィンドウが最小化され，ポインタが指差し型に変わります。

3 テスト対象アプリケーション内のテーブル・オブジェクトをクリックします。

注 : テーブル・オブジェクトとは，テスト環境の XML でそのテスト・オブジェクト・クラスの **SupportsTabularData** 属性が true に設定されているオブジェクトです。

［テーブル チェックポイントのプロパティ］ダイアログ・ボックスが開きます。［テーブルの内容］タブで，テーブルのカラムとセルのデータがダイアログ・ボックスに正しく表示されていることを確認します。

［プロパティ］タブで，テスト環境の XML で **ForVerification** 属性を指定した認識プロパティがダイアログ・ボックスに表示され，テスト環境の XML で **ForDefaultVerification** 属性を指定した認識プロパティが標準設定で選択されていることを確認します。

注 : ［プロパティ］タブは，テーブル・オブジェクトの任意の認識プロパティに **ForVerification** または **ForDefaultVerification** 属性を指定した場合にのみ表示されます。

---

4 適合するチェックポイントの詳細を指定します。詳細については,『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。［**OK**］をクリックすると，チェックポイントがテストに挿入されます。

5 作成したテストを実行し, チェックポイントを正しく実行していることを確認します。実行結果のレポートを確認し，テスト・オブジェクト・クラスに使用したアイコンがテスト環境の XML で定義したものであることを確認します。

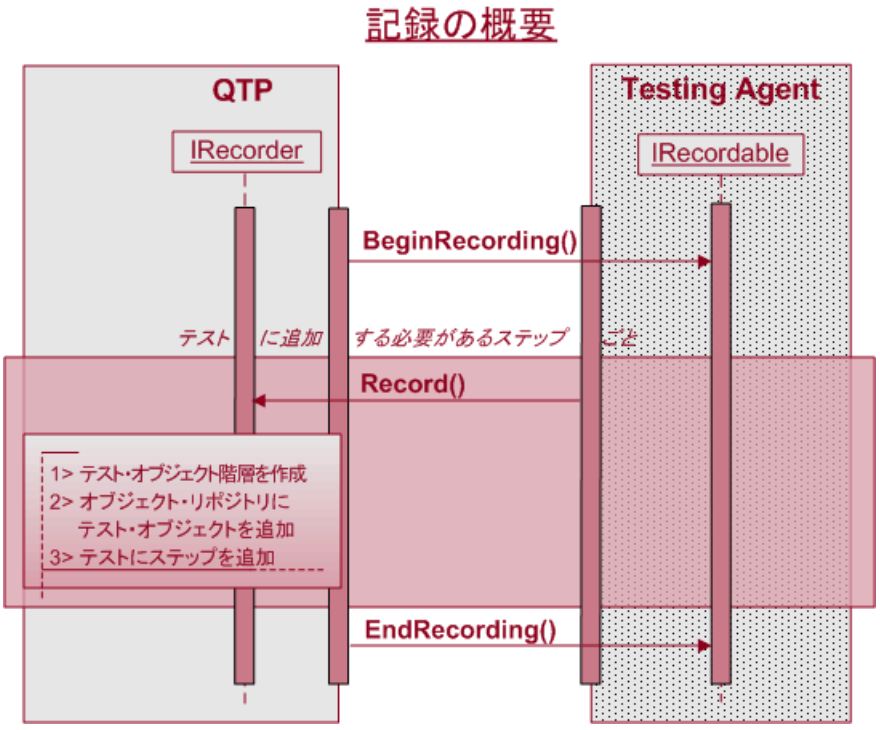
記録のサポートの開発

QuickTest でオブジェクト・リポジトリにオブジェクトを追加してテストを作成する方法の 1 つは，記録を使用することです。QuickTest ユーザは記録セッションを開始し，アプリケーション上で操作を実行して，記録を停止します。記録セッション中に，QuickTest はユーザのすべての操作をテストのステップとして記録し，関連するすべてのオブジェクトをオブジェクト・リポジトリに追加します。

QuickTest の記録機能をサポートするには，Testing Agent は次の操作を実行する必要があります。

- **IRecordable** インタフェースで **BeginRecording** および **EndRecording** メソッドを実装し，QuickTest が QuickTest ユーザによる記録セッションの開始と停止の時点を Testing Agent に通知できるようにします。
- **Record** コールバック・メソッドを呼び出します。このメソッドは，QuickTest に記録セッション中の記録する必要があるすべての操作を通知するために，QuickTest が **IRecorder** インタフェースで提供します。
- **IRecordSuppressor** インタフェースで **Suppress** および **UnSuppress** メソッドを呼び出し，必要に応じてネイティブの QuickTest イベント記録を抑制して，不要なステップがテストに追加されないようにします。

次の図は，テストの記録時に呼び出されるメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest が **IRecordable** インタフェースから **BeginRecording** および **EndRecording** メソッドを呼び出して，QuickTest ユーザが記録セッションを開始または停止した時点を Testing Agent に通知します。記録セッション中に，Testing Agent がユーザの操作をテストのステップとして記録する必要があると判断した場合，Testing Agent は **Record** メソッドを呼び出して，QuickTest にステップを記録するように指示します。



詳細については，次を参照してください。

➤ 73 ページ「ステップの記録」

➤ 74 ページ「QuickTest のネイティブの記録の抑制」

BeginRecording および EndRecording の実装

QuickTest は，QuickTest ユーザが記録セッションを開始すると，**BeginRecording** メソッドを呼び出します。**BeginRecording** メソッドは, **IRecorder** および **IRecordSuppressor** インタフェースを公開する QuickTest Testing Extensibility オブジェクトへの参照を受け取ります。記録セッションがアクティブである間, Testing Agent は **IRecorder::Record** コールバック・メソッドを使用して，QuickTest に記録する必要があるすべての操作を通知します。詳細については，73 ページ「ステップの記録」を参照してください。

記録セッションが開始すると, Testing Agent は **IRecordSuppressor::Suppress** メソッドを呼び出して，すでに開いているプロセスとウィンドウ上でネイティブの記録を抑制できます。詳細については，74 ページ「QuickTest のネイティブの記録の抑制」を参照してください。

QuickTest は，QuickTest ユーザが記録セッションを停止すると，**EndRecording** メソッドを呼び出します。このメソッドは，QuickTest Testing Extensibility オブジェクトへの参照が無効になったので, Testing Agent が **IRecorder** および **IRecordSuppressor** インタフェースのメンバを使用できないことを Testing Agent に通知します。

これらのメソッドの構文の詳細については,『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help \QTP_TestExt_Reference.chm**）を参照してください。

ステップの記録

QuickTest 記録セッションがアクティブである間，Testing Agent は，アプリケーション上で行われ，テストでステップとして記録する必要があるすべての操作に対して，**IRecorder** インタフェースの **Record** コールバック・メソッドを呼び出す必要があります。テスト対象アプリケーションの動作にとって重要で，実際のユーザが実行するビジネス・プロセスを反映する操作のステップを記録します。アプリケーションで行われる下位レベルのイベント（mouseover やスクロール・ダウンなど）のステップは記録しないでください。

たとえば, テスト環境の XML でいくつかのクリックとキーストロークを必要とする複雑な操作を定義する場合，Testing Agent を実装し，すべてのクリックとキーの押し下げに対してでなく，操作全体に対してのみその完了後に **Record** メソッドを呼び出してください。**Record** メソッドを呼び出すと，QuickTest はステップを記録するよう指示され，テスト・オブジェクト，テスト・オブジェクト・メソッド，ステップに含める引数が提供されます。

Testing Agent は，アプリケーションが実行された操作に反応することを許可する前に，**Record** メソッドを呼び出す必要があります。これにより，QuickTest はステップの実行前にアプリケーションの状態に関する情報を収集できるようになります。QuickTest は，その後テストの実行時にこの情報を使用できます。

また，QuickTest がアプリケーションからの情報を収集中にアプリケーションの状態が変わらないように，Testing Agent は，**Record** メソッドから戻るまでアプリケーションをブロックする必要があります。**Record** メソッドから戻るまで待機している間，QuickTest が呼び出すメソッドを実行できるように Testing Agent は使用可能である必要があります。

Record メソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

QuickTest のネイティブの記録の抑制

ユーザがアプリケーション上で操作を実行し，Testing Agent が記録すべき対応ステップを報告する場合，QuickTest がアプリケーション上で検出するほかのイベントの余分なステップを記録しないように，QuickTest に指示する必要もあります。このネイティブの記録が抑制されない場合，冗長なステップがテストに追加されます。

Testing Agent は，QuickTest が無視すべきで記録する必要がないイベントを識別できるようにする必要があります。これは，次のいずれかまたは両方の方法で実現できます。

- **静的抑制：**テスト環境の XML で，QuickTest がそのイベントを無視する必要があるオブジェクトを指定します。詳細については，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』）を参照してください。静的抑制の定義は，QuickTest が開いていて，実行時に変更されない限り，有効になります。

  テスト環境の XML では，そのイベントを無視する次の 2 つのタイプのオブジェクトを指定できます。

  - ActiveX（無視する ActiveX コントロールの ProgID のリストを定義します）
  - 標準 Windows（無視するウィンドウのウィンドウ・クラス名のリストを定義します）

➤ **動的抑制：**記録処理中に，Testing Agent がそのイベントを無視するオブジェクトを指定します。これを行うために，Testing Agent が **IRecordSuppressor** インタフェースの **Suppress** および **UnSuppress** メソッドを呼び出します。**Suppress** および **UnSuppress** メソッドでは, Testing Agent が環境 ID とイベントを無視するプロセスとウィンドウのリストを提供します。このリストには，アプリケーションのプロセス ID とウィンドウ・ハンドルを含めることができます。特定のウィンドウ上のイベントを無視する場合，その子ウィンドウ上のイベントも無視されます。

動的抑制メソッドを呼び出しても，テスト環境の XML で定義された静的抑制には影響しません。

Testing Agent を実装し，**Suppress** および **UnSuppress** メソッドを呼び出して，QuickTest の記録セッションの開始時，プロセスまたはウィンドウの作成または削除時に，リストを更新します。

テストを記録するためのサポートのテスト

QuickTest の記録機能をサポートするには，**IRecordable** インタフェースを実装して，アプリケーション上の操作を記録し，このアプリケーション上での QuickTest のネイティブの記録を抑制するようにTesting Agent を設計します。このサポートを開発したら，環境のオブジェクトを含むアプリケーション上でテストを記録するプロセスをテストできます。

テストを記録するためのサポートをテストするには，次の手順で行います。

1 61 ページ「オブジェクト・スパイのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを含むアプリケーションを実行します。

2 QuickTest で，［**記録**］ボタンをクリックするか，［**記録**］＞［**記録**］を選択して記録セッションを開始します。詳細については,『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。アプリケーション上で操作を実行します。

3 キーワード・ビューで，次のことを確認します。

- ➤ QuickTest が実行したステップを正しく記録している。
- ➤ 冗長なステップが記録されていない。
- ➤［**注釈**］カラムのステップ・サマリにテスト環境の XML で定義した **Documentation** 要素が反映されている。

4 オブジェクト・リポジトリで，次のことを確認します。

- ➤ 操作を実行したオブジェクトがオブジェクト・リポジトリに正しく追加される。
- ➤ オブジェクトに使用したアイコンがテスト環境の XML で定義したアイコンである（定義した場合）。

5［**停止**］ボタンをクリックするか，［**記録**］＞［**停止**］を選択します。

6 キーワード・ビューで，環境のオブジェクトを含むステップの［**操作**］カラムをクリックします。次のことを確認します。

- ➤ 使用可能な操作のリストに，このタイプのオブジェクトに定義したすべてのテスト・オブジェクト・メソッドが含まれている。
- ➤ 定義したテスト・オブジェクト・メソッドまたは認識プロパティのツールヒントに，テスト環境の XML の **Description** 要素で定義した文字列が表示される。
- ➤ テスト環境の XML でテスト・オブジェクト・メソッドの引数を定義した場合，キーワード・ビューの［**値**］カラムに各引数のセルが含まれている。

7 作成したテストを実行し，正しく実行されることを確認します。

オブジェクトを強調表示するためのサポートの開発

QuickTest ユーザは，オブジェクト・リポジトリ内のオブジェクトを選択し，テスト対象アプリケーションでこのオブジェクトを強調表示するように QuickTest を設定できます。これにより，ユーザは QuickTest が確実にテスト・オブジェクトをアプリケーション内の正しいオブジェクトに関連付けるようにできます。オブジェクトが見えるよう正しいコンテキストでアプリケーションが開いている必要があります。そうでない場合，オブジェクトは強調表示されません。

QuickTest ユーザがオブジェクト・スパイのポインタをテスト・オブジェクト上に移動すると，QuickTest ではアプリケーション内のテスト・オブジェクトを強調表示します。これにより，ユーザはアプリケーション内のさまざまなテスト・オブジェクトを視覚的に区別できます。

アプリケーション内のオブジェクトを強調表示するには，QuickTest がフォーカスをアプリケーション・ウィンドウに切り替え，ウィンドウ内の指定したオブジェクトを検索できる必要があります。この QuickTest 機能をサポートするには，Testing Agent は，アプリケーション・ウィンドウに対するハンドル，QuickTest が指定する任意の可視なオブジェクトの画面座標を提供する必要があります。これを行うには，**IHWNDSupplier** インタフェースで **GetHWND** メソッドを，**IRectangleSupplier** インタフェースで **GetRectangle** メソッドを実装します。

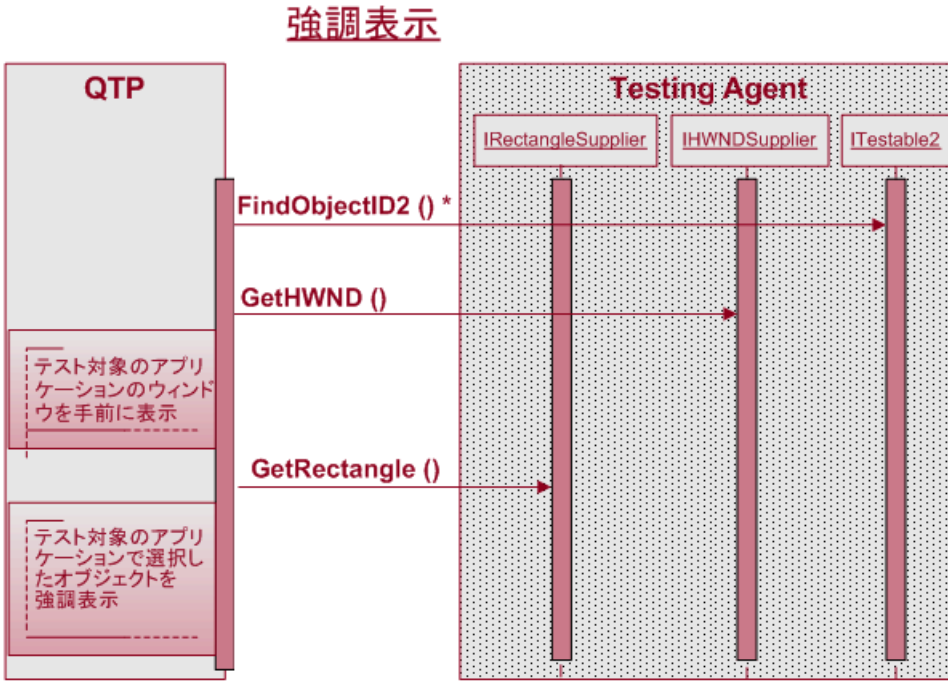
GetHWND の実装

指定したオブジェクトを含むウィンドウのハンドルを返すには，**GetHWND** メソッドを実装します。QuickTest はこのハンドルを使用して，オブジェクトを含むウィンドウ上でウィンドウ関連の操作（ウィンドウの前面への移動など）を実行します。このメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

GetRectangle の実装

指定したオブジェクトを囲む矩形の座標を返すには，**GetRectangle** メソッドを実装します。QuickTest は，フォーカスをアプリケーション・ウィンドウに切り替えた後，この矩形をアプリケーション・ウィンドウで強調表示します。QuickTest がフォーカスを切り替えられるようにするには，**IHWNDSupplier** インタフェースで **GetHWND** メソッドも実装する必要があります。これらのメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

次の図は，ユーザがアプリケーションで強調表示するテスト・オブジェクトを指定したときに，QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。たとえば，QuickTest は，最初に **ITestable2** インタフェースから **FindObjectId2** を，次に **IHWNDSupplier** インタフェースから **GetHWND** を，最後に **IRectangleSupplier** インタフェースから **GetRectangle** を呼び出します。



オブジェクトを強調表示するためのサポートのテスト

オブジェクトの強調表示をサポートするのに必要なメソッドを実装したら，Testing Agent のこの機能をテストできます。

オブジェクトを強調表示するためのサポートをテストするには，次の手順で行います。

1 61 ページ「オブジェクト・スパイのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- Testing Agent をアップグレードします。
- QuickTest を開いて，Testing Agent を実行します。
- 環境のオブジェクトを含むアプリケーションを実行します。

2 アプリケーションが強調表示するオブジェクトを含むウィンドウに対して開いていることを確認します。

3 オブジェクト・リポジトリ内で強調表示するオブジェクトを選択します。

4 ［**アプリケーションを強調表示**］ボタンをクリックするか，［**表示**］＞［**アプリケーションを強調表示**］を選択します。選択したオブジェクトがアプリケーション内で強調表示されていることを確認します。

画面キャプチャのサポートの開発

QuickTest は, テストの各ステップでアプリケーションのスナップショット（**ActiveScreen**）を格納できます。これにより，ユーザは，アプリケーションおよびステップの実行時に表示されるアプリケーションの中のすべてのオブジェクトの全状態を確認できます。また，QuickTest ユーザは，アプリケーションが使用できない場合やテスト・ステップに当初含まれていなかったオブジェクトに対しても，ActiveScreen を使用してステップをテストに追加できます。

ActiveScreen をキャプチャするには，QuickTest がフォーカスをアプリケーション・ウィンドウに切り替え，ウィンドウ内で可視のオブジェクトに関する情報を収集できる必要があります。この QuickTest 機能をサポートするには，Testing Agent はアプリケーションの可視なすべてのオブジェクトに関する情報（その場所や記述を含む）を提供する必要があります。この情報は，「**ActiveScreen Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。

ActiveScreen キャプチャをサポートするには，**IHWNDSupplier** インタフェースで **GetHWND** メソッドを，**IActiveScreenSupplier** インタフェースで **GetActiveScreen** メソッドを実装します。詳細については，77 ページ「GetHWND の実装」を参照してください。

テスト実行セッション中，QuickTest はアプリケーションの画像をキャプチャおよび保存して，実行結果に表示することもできます。これらの**ステップ画面キャプチャ**を使用すると，テスト・ステップを実行したときのアプリケーションの状態を確認できます。QuickTest Testing Extensibility は，ActiveScreen をサポートしているメソッドと同じメソッドを使用して，ステップ画面キャプチャをサポートします。

GetActiveScreen の実装

ActiveScreen で提供される詳細レベルは, QuickTest ユーザが ActiveScreen で作業時に使用できるテスト編集オプションに影響を与えます。たとえば，QuickTest ユーザが ActiveScreen からオブジェクトをテストまたはオブジェクト・リポジトリに追加する場合，QuickTest は ActiveScreen に格納されているオブジェクト情報のみ使用します。QuickTest ユーザが当初テスト・ステップに含まれていなかったオブジェクトを追加できるようにするには，ActiveScreen 情報に，テスト・オブジェクトの記述を構成している認識プロパティとこれらが記述プロパティであることの指示を含める必要があります。その他の認識プロパティを含めることもできます。

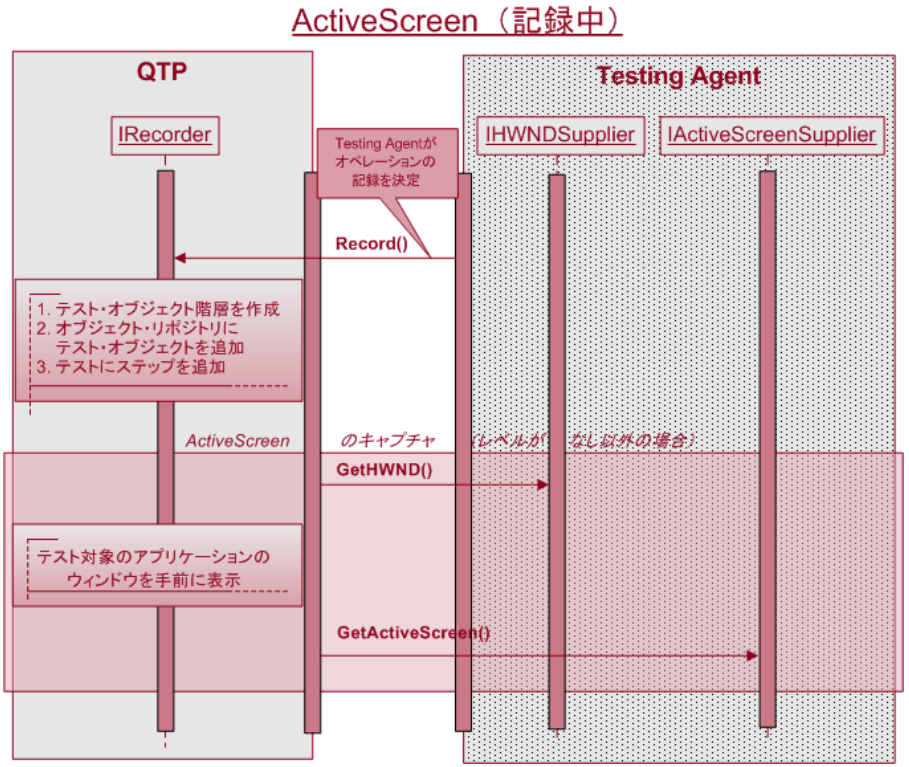
QuickTest ユーザが ActiveScreen からチェックポイントを追加できるようにするには，ActiveScreen 情報に，テスト環境の XML で **ForVerification** 属性を指定したテスト・オブジェクトの認識プロパティを含める必要があります。

各詳細レベルを選択した場合の結果の詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

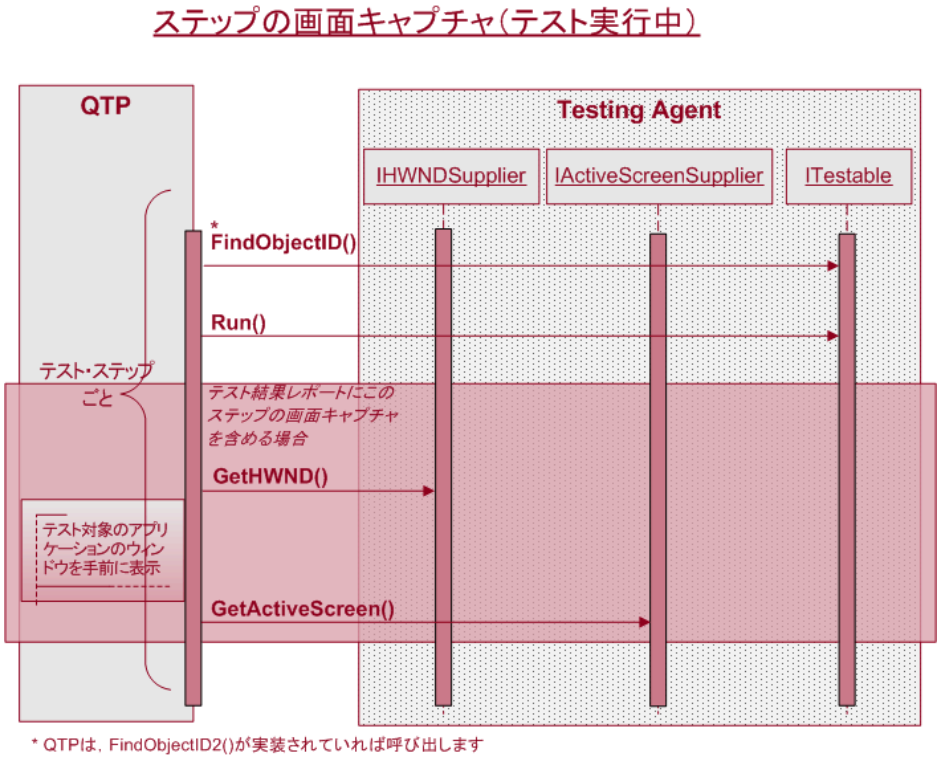
QuickTest は，フォーカスをアプリケーション・ウィンドウに切り替えた後に ActiveScreen をキャプチャします。QuickTest がフォーカスを切り替えられるようにするには，**IHWNDSupplier** インタフェースで **GetHWND** メソッドを実装する必要があります。

これらのメソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

次の図は，記録セッション中に ActiveScreen をキャプチャするときに，QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。記録する必要があるすべてのユーザ操作について，Testing Agent は **Record** メソッドを呼び出します。この時点で，QuickTest の ActiveScreen キャプチャ・レベルが **None** 以外の値に設定されている場合，QuickTest は，**IHWNDSupplier** インタフェースで **GetHWND** を呼び出し，次に **IActiveScreenSupplier** インタフェースで **GetActiveScreen** を呼び出します。



次の図は，実行結果レポート用にステップ画面キャプチャを保存するときに，QuickTest が呼び出すメソッドの一般的なシーケンスを示します。これは，Testing Agent が実装する必要があるメソッドの目的を示しています。



画面キャプチャのサポートのテスト

ActiveScreen キャプチャとステップ画面キャプチャをサポートするのに必要なメソッドを実装したら，Testing Agent のこの機能をテストできます。

ActiveScreen キャプチャのサポートをテストするには，次の手順で行います。

1 61 ページ「オブジェクト・スパイのサポートのテスト」の手順 1 から 4 を実行して，次のことを行います。

- ➤ Testing Agent をアップグレードします。
- ➤ QuickTest を開いて，Testing Agent を実行します。
- ➤ 環境のオブジェクトを含むアプリケーションを実行します。

2 QuickTest の［オプション］ダイアログ・ボックスの［ActiveScreen］表示枠（［**ツール**］＞［**オプション**］＞［**GUI テスト**］タブ＞［**ActiveScreen**］ノード）で，［**キャプチャのレベル**］を［**なし**］以外の値に設定します。また，［実行］＞［画面キャプチャ］表示枠の［**静止画像キャプチャをテスト結果へ保存**］を［**常に**］に設定します。

3 アプリケーションでステップを記録し，ActiveScreen が正しくキャプチャされることを確認します。アプリケーション上でテストを実行し，ステップ画面キャプチャが実行結果レポートに正しく追加されることを確認します。

4 ［オプション］ダイアログ・ボックスで異なる設定を使用して手順 2 と 3 を繰り返し，ActiveScreen が適切にキャプチャされることを確認します。

テスト・イベント通知の有効化

ITestEvents インタフェースはオプションです。QuickTest はこのインタフェースを使用して，さまざまなイベントが発生したときに Testing Agent に通知します。イベントが発生したときに Testing Agent に特定のアクションを実行させる場合に，このインタフェースを実装します。このインタフェースを実装すると，テスト実行の開始または終了時，テスト実行の一時停止または再開時，テストを閉じるとき，QuickTest を閉じる前に，QuickTest が Testing Agent に通知します。

QuickTest は，**ITestEvents** インタフェースの関連するメソッドを呼び出すことで，Testing Agent にこれらのイベントを通知します。Testing Agent が対応アクションを実行するイベントを表すメソッドを実装してください。

たとえば，テスト実行が終了したら，テスト実行中に QuickTest が開いたアプリケーションを Testing Agent によって閉じることができます。

第 5 章

QuickTest Testing Agent のデプロイ

Testing Agent を開発したら，それを QuickTest がインストールされているコンピュータ上にインストールおよび登録します。これにより，QuickTest では，環境内のオブジェクトを識別し，このようなオブジェクトを含むアプリケーション上でテストを実行できるようになります。

本章の内容

QuickTest Testing Agent のデプロイについて

QuickTest が Testing Agent と通信するには，Testing Agent を QuickTest と同じコンピュータ上にインストールし，QuickTest に登録する必要があります。ユーザの観点から見ると，Testing Agent をインストールし登録すると，Testing Agent は QuickTest アドインとして機能します。たとえば，QuickTest は，アドインまたはサポートされている環境のリストを表示するすべてのダイアログ・ボックスで環境の名前を表示します。また，QuickTest は，各アドインで使用できるテスト・オブジェクト・クラスのリストを表示するダイアログ・ボックスで，Testing Agent によって定義されたテスト・オブジェクト・クラスのリストを表示します。

使用しているテクノロジに合った手順を使用して，Testing Agent をインストールできます。このインストールは，ソフトウェア・インストールの一環として行うことも，別個の手順で行うこともできます。

Testing Agent は，QuickTest Professional インストール・フォルダにインストールしないでください。

デプロイでは, 環境の XML ファイルのコピーを次の場所にインストールする必要があります。

- QuickTest で使用するためにインストールする場合，テスト環境のファイルを **<QuickTest Professional インストール・フォルダ>\dat\Extensibility\TEA** フォルダにインストールする必要があります。
- Business Process Testing で QuickTest Testing Extensibility を使用する場合，テスト環境のファイルを **<QuickTest Add-in for ALM/QC インストール・フォルダ>\dat\Extensibility\TEA** フォルダにもインストールする必要があります。Business Process Testing の詳細については，『HP QuickTest Professional for Business Process Testing ユーザーズ・ガイド』および『HP Business Process Testing ユーザーズ・ガイド』を参照してください。

Testing Agent を登録するには，QuickTest がこの目的で提供する C 関数を呼び出します。詳細については，89 ページ「Testing Agent の登録」を参照してください。登録は，Testing Agent インストールの一環として行うことも，別個の手順で行うこともできます。

この登録で, QuickTest が Testing Agent と通信できるようになります。QuickTest に Testing Agent を登録しても，オペレーティング・システムで COM オブジェクトの登録が実行されるわけではありません。Testing Agent COM オブジェクト自身のコンピュータへの登録は，別個に処理する必要があります。

注：Business Process Testing で QuickTest Testing Extensibility を使用する場合，テスト環境を定義している XML ファイルもインストールする必要があります。テスト環境の XML ファイルは，**<QuickTest Add-in for ALM/QC インストール・フォルダ>\dat\Extensibility\TEA** フォルダに配置する必要があります。

テスト環境の XML ファイルは，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』(**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に準拠する必要があります。
テスト環境の XML ファイルに含める必要がある定義については，20 ページ「テスト・オブジェクト・クラスとテスト・オブジェクト・メソッドの定義」を参照してください。

Business Process Testing については，『HP QuickTest Professional for Business Process Testing ユーザーズ・ガイド』および『HP Business Process Testing ユーザーズ・ガイド』を参照してください。

Testing Agent の登録

QuickTest のインストール後に, Testing Agent を登録する必要があります。アプリケーションのインストールで QuickTest のインストール前にエージェントのインストールと登録が行われる場合は，QuickTest のインストール後に登録を繰り返してください。

QuickTest Testing Extensibility SDK には，Testing Agent の登録と登録解除，QuickTest がインストールされているかどうかの確認に使用できる C 関数が用意されています。これらの関数を使用してエージェントを登録および登録解除するアプリケーションを作成します。コードは次の条件を満たす必要があります。

- **<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\lib\<VS バージョン>\TEASDK.lib** ライブラリにリンクしていること
- **<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\include\RegisterAUT.h** をインクルードしていること

RegisterTeaAut 関数の使用

Testing Agent をインストールする場合，**RegisterTeaAut** を呼び出します。

long RegisterTeaAut (const wchar_t* environmentID**, const wchar_t*** szEnvDisplayName**, const wchar_t*** progid**);**

この関数は，Testing Agent の登録に成功すると，ERROR_SUCCESS (0) を返します。

また，次の null で終了する Unicode 文字列引数を受け取ります。

- **environmentID :** QuickTest がテスト環境の識別に使用する名前。この引数の値は，テスト環境の XML ファイル内の **PackageName** 属性（**TypeInformation** 要素）と同じである必要があります。
- **szEnvDisplayName :** QuickTest がサポートされている環境のリストを表示するダイアログ・ボックスでユーザに表示する環境の名前。この引数の値は，テスト環境の XML ファイル内の **AddinName** 属性（**TypeInformation** 要素）と同じである必要があります。
- **progid :** Testing Agent COM オブジェクトをコンピュータに登録したときに使用したプログラム ID。QuickTest はこのプログラム ID を使用して，Testing Agent を起動します。

UnRegisterTeaAut 関数の使用

Testing Agent をアンインストールする場合，**UnRegisterTeaAut** を呼び出します。

long UnRegisterTeaAut (const wchar_t* environmentId**);**

この関数は，Testing Agent の登録解除に成功すると，ERROR_SUCCESS (0) を返します。

また，次の null で終了する Unicode 文字列引数を受け取ります。

environmentID：Testing Agent の登録に使用した環境 ID。

QuickTest がインストールされているかどうかの確認

QuickTest がコンピュータにインストールされていないときに Testing Agent のインストールまたは登録を回避する場合，**IsQTPInstalled** 関数を使用して，QuickTest がコンピュータにインストールされているかどうかを確認できます。

long IsQTPInstalled();

この関数は，QuickTest バージョン 9.2 以降がインストールされている場合，ERROR_SUCCESS (0) を返します。

第 6 章

QuickTest Testing Agent のテストとデバッグ

Testing Agent の開発段階では，QuickTest が Testing Agent の支援を受けて環境のアプリケーションに対してどのように動作するかをテストします。QuickTest の動作がニーズを満たしていない場合，QuickTest Professional ログ記録メカニズムを使用して Testing Agent をデバッグできます。

本章の内容

Testing Agent のテスト

Testing Agent の開発は段階的に実行します。最初に，最も基本的な QuickTest 機能のサポートを作成し，順次その他の機能のサポートを作成していきます。

開発の各段階が完了したら，Testing Agent を QuickTest がインストールされているコンピュータ上にインストールおよび登録し，QuickTest を実行して，サポートを作成した機能をテストします。

さまざまな段階のテストに使用する手順については，各開発段階について説明している項に記載されています。詳細については，次を参照してください。

➤ 32 ページ「QuickTest がテスト環境を認識することのテスト」

➤ 44 ページ「テストを実行するためのサポートのテスト」

➤ 45 ページ「スマート認識のサポートのテスト」

➤ 49 ページ「プロパティ値を取得するためのサポートのテスト」

➤ 50 ページ「チェックポイントを作成するためのサポートのテスト」

➤ 51 ページ「出力値ステップのサポートのテスト」

➤ 55 ページ「オブジェクトを学習するためのサポートのテスト」

➤ 61 ページ「オブジェクト・スパイのサポートのテスト」

➤ 63 ページ「ネイティブのプロパティと操作にアクセスするためのサポートのテスト」

➤ 69　ページ「記録セッション中にチェックポイントを作成するためのサポートのテスト」

➤ 75 ページ「テストを記録するためのサポートのテスト」

➤ 79 ページ「オブジェクトを強調表示するためのサポートのテスト」

➤ 84 ページ「画面キャプチャのサポートのテスト」

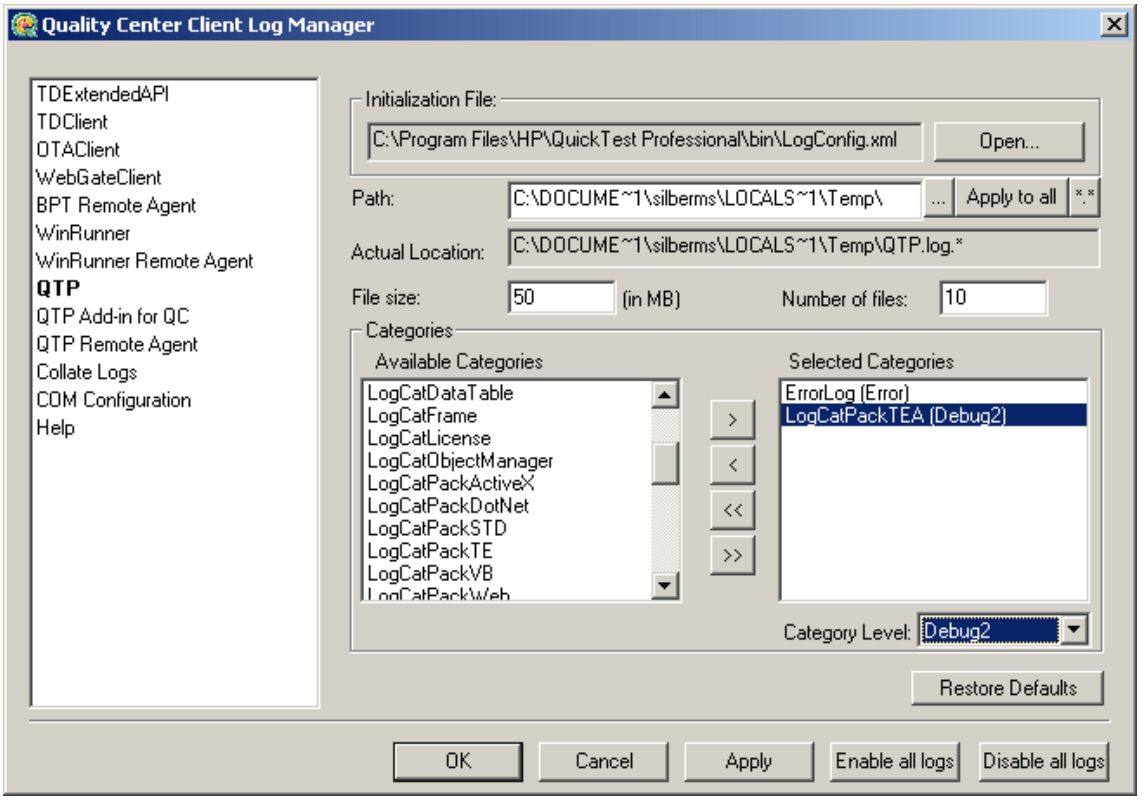
Testing Agent のデバッグ

Testing Agent の機能をテストする場合, QuickTest が期待どおりに動作しなければ, ロガーを使用して問題点を調べることができます。ロガーの Testing Extensibility カテゴリには, 重要なエントリ・ポイントとエラー時に生成されたログが含まれています。このため，ロガーをアクティブにし，Testing Extensibility カテゴリからログ・メッセージを生成するように設定すると，問題の特定に役立ちます。

ロガーをアクティブにするには，次の手順で行います。

1 QuickTest Professional を閉じます。

2 **<QuickTest Professional インストール・フォルダ>\bin** フォルダを参照して，**ClientLogs.exe** を実行します。Client Log Manager で，左側の表示枠の［**QTP**］を選択します。



3 ［**レベル**］リストで，［**Debug2**］を選択します。このデバッグ・レベルは，ログ・メッセージの最も広範なセットを提供します。

4 ［**パス**］ボックスで，ログ・ファイルを保存するフォルダを指定します。参照ボタン（...）をクリックすると，特定のフォルダに移動できます。QuickTest を実行する前に，フォルダが存在することを確認してください。

5 ［**ファイル サイズ**］ボックスで，ログ・ファイルに許可する最大サイズ（MB）を指定します。

ログ・ファイルがこの上限に到達すると，同じフォルダに新しいログ・ファイルが作成されます。その後のログ・メッセージは新しいファイルに書き込まれます。

6 ［Categories］領域の［**Available categories**］リストには，ログ・ファイルに含まれないカテゴリが表示されます。［**Selected categories**］リストには，ログ・ファイルに含まれるログ・メッセージのカテゴリが表示されます。

カテゴリ名を選択し，矢印ボタンをクリックして，カテゴリを［**Available categories**］リストと［**Selected categories**］リストの間で移動します。

ErrorLog と **LogCatPackTEA** が［**Selected categories**］リスト内にあることを確認します。これで，Testing Extensibility に関連するログ・メッセージを含むログ・ファイルを作成するように QuickTest が設定されます。

7 QuickTest を実行して，問題のシナリオを再現します。

8 指定したフォルダに作成されたログ・ファイルを確認します。

付録 A

QuickTest Testing Extensibility を使用する際のその他の考慮事項

本章では，QuickTest Professional Testing Extensibility Testing Agent の設計時に留意する必要がある追加情報を提供します。

本章の内容

XML での大文字と小文字の区別

XML 構文には，大文字と小文字の区別があります。

XML データの検証

QuickTest に提供する XML データを，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\SDK\schemas** フォルダ内の該当するスキーマ・ファイルに照らして検証します。

micclass 認識プロパティのサポート

内部で使用するために，QuickTest は **micclass** 認識プロパティをすべてのテスト・オブジェクトに追加します。Testing Agent では，このプロパティに値を提供する必要はありません。このプロパティは，QuickTest のいずれのダイアログ・ボックスにも表示されません。ただし，QuickTest ユーザは，**GetTOProperty** および **GetTOProperties** メソッドを使用してこのプロパティを取得できます。

テスト・オブジェクト・メソッドからのテスト・オブジェクトの戻し

テスト・オブジェクト・メソッドがテスト・オブジェクトを返す必要がある場合があります。たとえば，テーブルにチェック・ボックスなどのテスト・オブジェクトが含まれ

ている場合，セルの内容を返すテスト・オブジェクト・メソッドは，テスト・オブジェクトを返す必要があります。

QuickTest Extensibility Virtual Object タイプ・ライブラリに定義されている **TEAVirtualObjectImp** コクラスを使用すると，メソッドはオブジェクトを返すことができます。コクラスを使用するには，**TEAVirtualObject.idl** をプロジェクトにインポートします。このファイルは，**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\lfs** にあります。詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

テスト・オブジェクト・メソッドからテスト・オブジェクトを返すには，**TEAVirtualObjectImp** オブジェクトを作成し，返すテスト・オブジェクトの実行時 ID をそれに設定し，テスト・オブジェクト・メソッドから仮想オブジェクトを返します。

QuickTest が仮想オブジェクトを返すテスト・オブジェクト・メソッドを実行する場合，QuickTest は仮想オブジェクトを実際のテスト・オブジェクトに変換します。このメカニズムは QuickTest ユーザにとって透過的であり，ユーザは返されたテスト・オブジェクトを変数に割り当て，後続のテスト・ステップで使用できます。

たとえば，Hours Report サンプル・アプリケーション用に作成されたサポートには，アプリケーション全体を表すテスト・オブジェクトがあります。このオブジェクトには，Working Hours Report リストを表すテスト・オブジェクトが含まれています。これらの各テスト・オブジェクトは，別個に学習できますが，このデモでは，アプリケーションを表す **DemoApp** テスト・オブジェクトにも，Working Hours Report リスト用のテスト・オブジェクトを返すテスト・オブジェクト・メソッド（**GetReport**）があるものとします。

GetReport テスト・オブジェクト・メソッドをサポートするように設計された次のコードの抜粋は，仮想オブジェクトを使用してテスト・オブジェクトを返す方法を示します。

```
HRESULT CAppDlg::Run(/*in*/  BSTR       sMethod,
                     /*in*/  SAFEARRAY*  arguments,
                     /*out*/ VARIANT*    vtRetValue,
                     /*out*/ CComBSTR&   sErrorMsg)
{
...
// Create Virtual test object and return it in the out parameter
CComPtr<ITEAVirtualObject> spVirtualObject;
spVirtualObject.CoCreateInstance(L"TEAVirtualObject.TEAVirtualObjectImp");
    if (spVirtualObject)
    {
        // Get run-time ID for the Report test object
        CComVariant vtReoprtRtId = m_pWorkingHourReport->GetRuntimeID();

        // Set run-time Id into the virtual test object
        spVirtualObject->SetRuntimeID(vtReoprtRtId);

        // Return virtual test object in output parameter
        CComVariant vtVirtualObject(spVirtualObject);
        vtVirtualObject.Detach(vtRetValue);
    }
...
```

この例は，**CAppDlg::Run** 関数内の **<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\samples\HoursReport\vs2003\src\TeaAutDemo\GUIElement.cpp** からのものです。

テストを編集する場合，QuickTest ユーザは次のステップを作成して，Working Hours Report リストのテスト・オブジェクトを取得し，レポート内の行数をメッセージ・ボックスに印刷することもできます。

```
Set var_Report = DemoApp("TEA Demo").GetReport()
MsgBox var_Report.GetReportLength()
```

テスト環境の XML に関する追加情報

テスト環境の XML でテスト・オブジェクト・モデルを定義する場合，設定できる多数の役に立つ属性があります。詳細については，35 ページ「テスト・オブジェクト・モデルのサポートの開発」を参照してください。考えられるすべての要素と属性の詳細については，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

Withable 属性の使用

QuickTest Professional ユーザは，テストで VBScript の **With** キーワードを使用して，同じ親階層を持つ連続するステートメントをグループ化できます。**With** ステートメントを使用すると，QuickTest は，**With** ブロック内のすべてのステップに同じ実行時 ID を使用し，テスト・オブジェクト記述に従って各ステップの実行時 ID を取得しないようになります。

各ステップに対してその実行時 ID を取得する必要があるテスト・オブジェクトの場合，テスト環境の XML 内のテスト・オブジェクト・クラスの **Withable** 属性を false に設定します。

列挙のサポート

QuickTest は，テスト・オブジェクト・メソッド引数の入力時に，定義済みのリストの値を使用するようにユーザに要求できます。テスト環境の XML では，このような値の定義済みのリストを定義できます。次に，テスト環境の XML で関連するテスト・オブジェクト・メソッド引数を定義するときに，引数のタイプで値のリストの名前を使用します。

たとえば，次のように列挙型 CtrlShift を定義できます。

```
<ListOfValues Name="CtrlShift">
        <EnumValue Name="None" RealValue="0"/>
        <EnumValue Name="Ctrl" RealValue="1"/>
        <EnumValue Name="Shift" RealValue="2"/>
        <EnumValue Name="CtrlShift" RealValue="3"/>
</ListOfValues>
```

これらの値を使用する必要があるテスト・オブジェクト・メソッドを定義する場合，引数のタイプを次のように定義します。

```
<Type VariantType="Enumeration" ListOfValuesName="CtrlShift"/>
```

テスト・オブジェクト・クラスの QuickTest 汎用タイプへの割り当て

テスト環境の XML で定義するテスト・オブジェクト・クラスの **GenericTypeID** 属性を既存の汎用タイプに設定する場合，テスト・オブジェクトをそのタイプのオブジェクトとして処理するように QuickTest を設定します。つまり，次のようになります。

1. QuickTest は，サポート対象でこの汎用タイプに共通のテスト・オブジェクト・メソッドに対して，キーワード・ビューの［注釈］カラムの汎用定義を提供します。たとえば，テスト・オブジェクト・クラスが汎用タイプ **Button** に属し，**Click** メソッドをサポートすると定義した場合，ボタン・オブジェクトの注釈の文字列は，ほかのすべてのボタンに使用される文字列と同様になります。
2. QuickTest によってユーザがテスト・オブジェクトをタイプでフィルタすることが許可されている場合，テスト・オブジェクト・クラスのオブジェクトは，定義した汎用タイプのオブジェクトとみなされます。次に例を示します。
   - ステップ・ジェネレータの［ステップでオブジェクトを選択］ダイアログ・ボックスでは，特定の汎用タイプのオブジェクトのみ表示するように選択できます。
   - コンテナ・オブジェクトをオブジェクト・リポジトリに追加したときにアクセスできる［オブジェクト タイプの選択］ダイアログ・ボックスでは，ユーザは特定の汎用タイプの子孫のみ学習するように QuickTest を設定できます。
   - オブジェクト・リポジトリ内のオブジェクトの検索に使用される［検索と置換］ダイアログ・ボックスでは，ユーザは汎用オブジェクト・タイプに従って検索を絞り込むことができます。

テスト・オブジェクト・クラスを QuickTest の汎用タイプに割り当てても，このクラスのテスト・オブジェクトのサポートは提供されません。Testing Agent で，テスト・オブジェクト・メソッドのすべて，およびこのテスト・オブジェクト・クラスの認識プロパティのすべてのサポートを実装する必要があります。Testing Agent は，テスト・オブジェクト・メソッドが呼び出されたときに関連する操作を実行し，必要に応じて認識プロパティの値を返す必要があります。

ステートメントの自動補完のサポート

QuickTest 内でステートメントの自動補完をサポートするには，テスト環境の XML を **<QuickTest Professional インストール・フォルダ>\dat\Extensibility\tea** フォルダにインストールします。ファイル名には，XML 拡張子が必要になります。

付録 B

QuickTest および QuickTest Testing Extensibility Testing Agent を使った作業

本章では，QuickTest Testing Extensibility Testing Agent を使用した場合の QuickTest の動作方法に関する情報を提供します。これらの情報のいくつかを Testing Agent 用に提供するユーザ・ドキュメントに含めることもできます。

本章の内容

Testing Agent のロード

QuickTest が開くと，登録されている Testing Agent が確認され，アドイン・マネージャにそれらの環境が表示されます。QuickTest ユーザはアドインを選択し，Testing Agent の COM オブジェクトを作成することでその環境のサポートをロードするように QuickTest を設定できます。［アドイン マネージャ ］ダイアログ・ボックスの詳細については，『HP Unified Functional Testing アドイン・ガイド』を参照してください。

Testing Agent の環境表示名が一意な場合，QuickTest オートメーション・スクリプトを使用して Testing Agent をロードすることもできます。これを行うには，**SetActiveAddins** メソッドを呼び出し，**AddinNames** 引数に環境表示名を指定します。

この引数は，次の場所で指定する名前と同じである必要があります。

➤ **RegisterTeaAut** メソッド（Testing Agent の登録に使用）の **szEnvDisplayName** 引数。

➤ テスト環境の XML の **TypeInformation** 要素の **AddinName** 属性。

この名前は，QuickTest が開いたときに［アドイン マネージャ］ダイアログ・ボックスに表示されます。**SetActiveAddins** メソッドの詳細については，『QuickTest Professional Automation Object Model Reference』を参照してください。

環境のサポートがロードされると，次のようになります。

➤ QuickTest は，環境内のコントロールを認識し，そのコントロールに関するテストを実行できるようになります。

➤ QuickTest は，アドインまたはサポートされる環境のリストを表示するすべてのダイアログ・ボックスに，環境の表示名を表示します（［新規テスト オブジェクトの定義］ダイアログ・ボックス，［オブジェクトの認識］ダイアログ・ボックスなど）。

➤ QuickTest は，各アドインまたは環境で使用できるテスト・オブジェクト・クラスのリストを表示するダイアログ・ボックスで，テスト環境の XML で定義されたテスト・オブジェクト・クラスのリストを表示します。

［オブジェクトの認識］ダイアログ・ボックスの使用

QuickTest ユーザは，テスト環境の XML で定義されるテスト・オブジェクト記述を構成する認識プロパティのセットを変更することはできません。このため，QuickTest ユーザが［オブジェクトの認識］ダイアログ・ボックスで Testing Extensibility Testing Agent によってサポートされている環境を選択すると，プロパティの追加と削除用のボタン，および［**順序識別子**］オプションが無効になります。

スマート認識の設定と使用

［オブジェクトの認識］ダイアログ・ボックスでは，QuickTest ユーザは，Testing Extensibility によってサポートされている環境に属するテスト・オブジェクト・クラスを選択し，そのプロパティをスマート認識用に設定できます。スマート認識の設定の詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

アプリケーション内にテストに格納されている記述と一致するオブジェクトがない場合，QuickTest はスマート認識（設定されている場合）を使用して，アプリケーション内のオブジェクトを認識しようとします。

複数のオブジェクトが記述に一致する場合に，QuickTest がスマート認識メカニズムを使用して正しいオブジェクトを特定できるようにするには，Testing Agent で **FindObjectId2** を実装する必要があります。詳細については，40 ページ「FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て」を参照してください。

10.00 より前のバージョンを使用して開発された Testing Agent のアップグレード

「Testing Environment Schema」は，バージョン 9.5 から 10.00 への移行時に変更されました。このため，10.00 より前のバージョンの SDK を使用して開発された Testing Agent を，QuickTest のバージョン 10.00 以降で使用するには，次の手順を実行する必要があります。

1 次のように環境の XML を変更します。

a **TypeInformation** 要素の **id** 属性名を **PackageName** に変更します。**PackageName** 属性の値が，**RegisterTeaAut** environmentID 引数に使用されているものと同じであることを確認します。

b 環境の表示名を使用して，**AddinName** 属性を **TypeInformation** 要素に追加します。この値が，**RegisterTeaAut** szEnvDisplayName 引数に使用されているものと同じであることを確認します。

2 環境の XML を **<QuickTest Professional インストール・フォルダ>\dat\Extensibility\TEA** フォルダにファイルとして保存します。

トラブルシューティングと制限事項

➤ QuickTest Professional Add-in for Quality Center を使用してビジネス・コンポーネントを作成または変更するときに，継承されたメソッドが［オートメーション］タブに表示されません。テスト・オブジェクト固有のメソッドのみ表示されます。

回避策：使用可能なすべてのメソッドを表示するには，Quality Center でなく QuickTest Professional でコンポーネントを編集します。

➤ QuickTest では，［**テスト オブジェクトの記述を更新**］オプションまたは［**ActiveScreen の画像と値を更新**］オプションを選択した状態で，Testing Extensibility テスト・オブジェクト・クラス上で更新実行モードでテストを実行することはできません。

➤ QuickTest の［**ナビゲートして学習**］機能は，Testing Extensibility テスト・オブジェクトに対してはサポートされていません。

付録 C

QuickTest Testing Extensibility の開始方法

QuickTest Professional Testing Extensibility SDK には，サンプル・アプリケーション，そのうえでの基本的な QuickTest テストのサポートを提供するために開発された Testing Agent，いくつかのバージョンのエージェントのソース・コードが含まれています。各バージョンのコードは，さまざまなバージョンの Visual Studio で使用するためのものです。これらのファイルは，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples** フォルダにインストールされます。

本章では，QuickID Testing Agent を開発するために従ったプロセスを順を追ってたどりながら，QuickTest Professional Testing Extensibility の基本事項を学習します。本ガイドを通じて，本章で説明するトピックの詳細，Testing Agent がサポートできるその他の QuickTest 機能を見つけることができます。

本章の内容

QuickTest Testing Extensibility の開始方法について

QuickTest Professional Testing Extensibility を使用すると，QuickTest 自動テスト機能を拡張して，それぞれの環境内のオブジェクトを認識および操作できるように QuickTest を設定できます。

QuickTest Professional Testing Extensibility を実装するには，QuickTest とテスト対象アプリケーション間のインタフェースをとる **Testing Agent** を開発します。Testing Agent は，QuickTest Professional Testing Extensibility SDK で定義されている一連の API を実装するシングルトン COM コクラスである必要があります。

Testing Agent を QuickTest がインストールされているコンピュータにインストールおよび登録したら，QuickTest を使用し，Testing Agent によって定義されたテスト・オブジェクト，プロパティ，メソッドのカスタム・セットを使用して環境のアプリケーションのテストを作成できます。

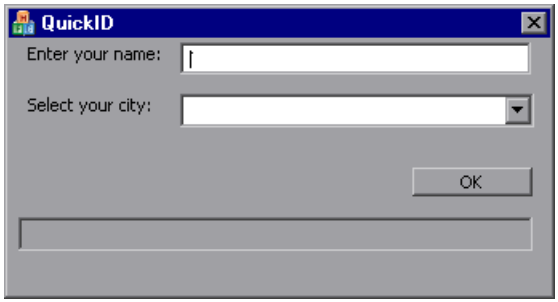
本章では，QuickID サンプル・アプリケーションの Testing Agent を開発した方法を説明することで，Testing Agent の開発方法を示します。このデモでは，サンプル・アプリケーションは QuickTest がサポートする必要がある環境を表すものとします。

QuickID サンプル・アプリケーションの概要

このサンプル・アプリケーションとその Testing Agent のソース・ファイルは，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\src** フォルダにインストールされます。**QuickID** フォルダの下には，Visual Studio のいくつかのバージョン用のサブフォルダがあります。サンプルに合った Visual Studio のバージョンを使用してプロジェクトを開くか，ソース・ファイルを個別に開くことができます。

QuickID アプリケーションは，少数のコントロールを持つ非常に単純な MFC ベースのアプリケーションです。このため，複雑なアプリケーションを学習せずに，Testing Agent を作成する方法を学習できます。

<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\bin\QuickID.exe をダブルクリックして，アプリケーションを実行します。



しばらくアプリケーションに親しんでから，閉じます。

Testing Agent 登録前の QuickTest によるアプリケーションのテスト

本項では，QuickID アプリケーション上で QuickTest テストを記録します。Testing Agent を使用しない場合，QuickTest では，QuickID アプリケーション上で非常に汎用的なテストしか提供できません。

QuickID アプリケーション上で QuickTest テストを記録するには, 次の手順で行います。

1 QuickTest テストを開きます。

2 次のようにしてサンプル・アプリケーションを実行します。

<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\bin フォルダを参照して，**QuickID.exe** ファイルをダブルクリックします。

3 QuickTest で，［記録と実行環境設定］ダイアログ・ボックスの［**開かれている Windows ベースのアプリケーションすべてでテストを記録して実行する**］オプションが選択されていることを確認します。次に記録セッションを開始します。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

4 QuickID アプリケーション上で次の操作を実行します。

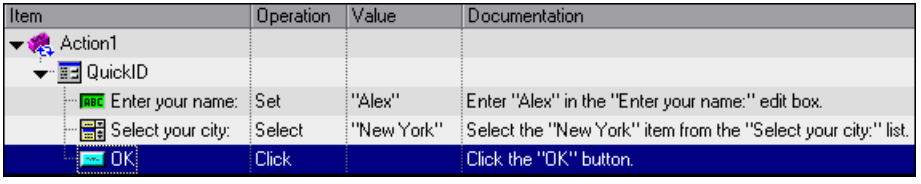
a テキスト・ボックスに名前を入力します。

b 下向き矢印ボタンをクリックし，リストから都市を選択します。

c ［**OK**］ボタンをクリックします。

QuickTest が実行した操作を記録します。

5 QuickTest で記録セッションを停止します。記録されたテスト・ステップは，次のようになります。



Item	Operation	Value	Documentation
Action1			
QuickID			
Enter your name:	Set	"Alex"	Enter "Alex" in the "Enter your name:" edit box.
Select your city:	Select	"New York"	Select the "New York" item from the "Select your city:" list.
OK	Click		Click the "OK" button.

テストとオブジェクト・リポジトリの両方で，QuickTest が物理 Windows コントロールに一致するテスト・オブジェクト・クラスを使用して，アプリケーションを標準 Windows アプリケーションとして識別していることを確認できます。

次の項では，Testing Agent が登録されている場合に QuickTest が同じ操作のセットを記録する方法を確認できます。

Testing Agent 登録後の QuickTest によるアプリケーションのテスト

本項では，QuickID アプリケーション用に Testing Agent を登録し，前に実行したステップを繰り返して，Testing Agent の支援のもとで QuickTest テストを記録します。新しいテストでは，Testing Agent によって定義されたより意味のあるテスト・オブジェクトとテスト・オブジェクト・メソッドが使用されます。

次の手順で Testing Agent を登録し，QuickID アプリケーション上で QuickTest テストを記録します。

1 コマンド・ウィンドウを開き，**<QuickTest Professional Testing Extensibility インストール・フォルダ>**\samples\QuickID**<Visual Studio バージョン>**\bin に移動します。install.bat を実行します。

QuickID アプリケーション用の Testing Agent が登録されます。

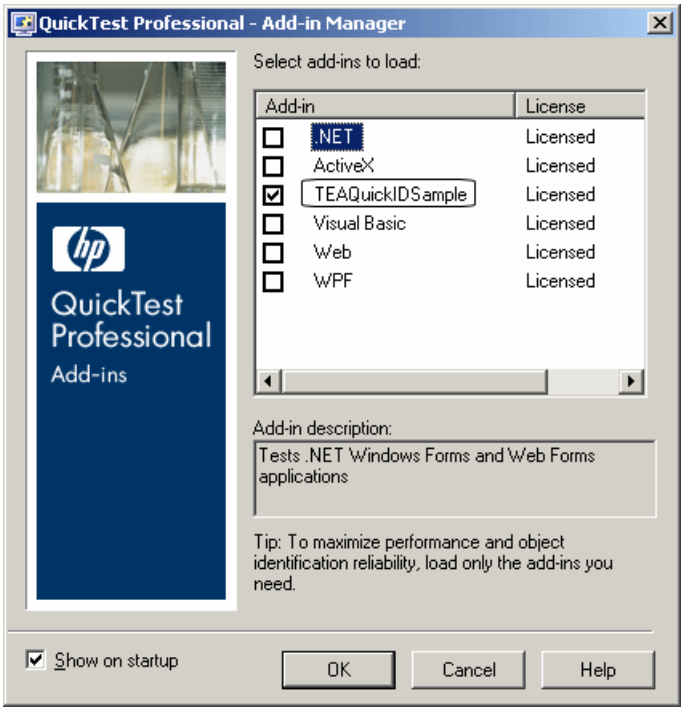
2 次のようにしてサンプル・アプリケーションを実行します。

<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\bin フォルダを参照して，**QuickID.exe** ファイルをダブルクリックします。

注意：QuickID サンプルの場合，QuickTest を開く前にアプリケーションを実行する必要があります。QuickID アプリケーションを実行すると，シングルトンの Testing Agent が作成されます。次に QuickTest を開くと，このインスタンスが使用されます。ただし，アプリケーションを開く前に QuickTest を開くと，QuickTest とアプリケーションのそれぞれによってお互いに通信しない別個の Testing Agent が作成されます。

---

3 QuickTest を開きます。QuickTest が登録済みの Testing Agent を認識し，その環境名（TEAQuickIDSample）を［アドイン マネージャ］ダイアログ・ボックスに表示します（［アドイン マネージャ］ダイアログ・ボックスが開かない場合は，『HP Unified Functional Testing アドイン・ガイド』の手順を参照してください）。



4 ［**TEAQuickIDSample**］のチェック・ボックスを選択し，［**OK**］をクリックします。QuickTest が開き，Testing Agent を実行します。

5 QuickTest で，［記録と実行環境設定］ダイアログ・ボックスの［**開かれている Windows ベースのアプリケーションすべてでテストを記録して実行する**］オプションが選択されていることを確認し，記録セッションを開始します。詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

6 QuickID アプリケーション上で次の操作を実行します。

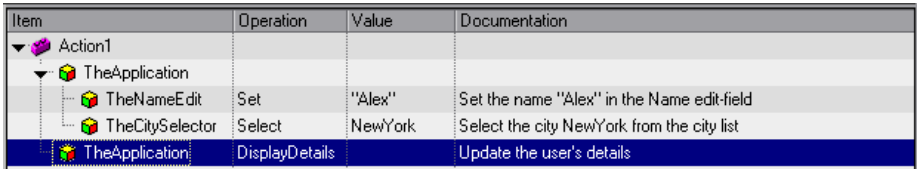
a テキスト・ボックスに名前を入力します。

b 下向き矢印ボタンをクリックし，リストから都市を選択します。

c ［OK］ボタンをクリックします。

QuickTest が実行した操作を記録します。

7 QuickTest で記録セッションを停止します。記録されたテスト・ステップは，次のようになります。



Item	Operation	Value	Documentation
Action1			
TheApplication			
TheNameEdit	Set	"Alex"	Set the name "Alex" in the Name edit-field
TheCitySelector	Select	NewYork	Select the city NewYork from the city list
TheApplication	DisplayDetails		Update the user's details

QuickTest では，オブジェクトを表す環境固有のテスト・オブジェクトを使用して，このアプリケーション内のオブジェクトをより明確に認識できるようになります。各テスト・オブジェクトも，特定のテスト・オブジェクト・メソッドを使用するようになります。この Testing Agent はそのテスト・オブジェクトに使用する特定のアイコンを定義していないため，QuickTest ではそれらのテスト・オブジェクト用に汎用アイコンを表示します。

Testing Agent の登録後，記録メカニズムを使用せずに QuickTest でテストを作成することもできます。［新規テスト オブジェクトの定義］ダイアログ・ボックスで，**TEAQuickIDSample** 環境を選択し，QuickTest が表示する環境固有のテスト・オブジェクトから選択します。各テスト・オブジェクトには，標準設定のテスト・オブジェクト・メソッドが定義されています。

QuickID アプリケーション用の Testing Agent の開発

本項では, QuickID アプリケーション用の Testing Agent の開発に使用したプロセスについて説明します。本項には，Testing Agent を開発した方法に関する一般的な情報と開発の段階の説明が記載されています。

Testing Agent の開発について

QuickID アプリケーションとその Testing Agent は, 単独の Microsoft Visual Studio プロジェクトで開発されました。サポートされている Visual Studio バージョンごとにプロジェクトがあります。このプロジェクトの出力は，アプリケーションの実行と Testing Agent のホストの両方に使用される **QuickID.exe** 実行可能ファイルです。

QuickID アプリケーションは，主に次のクラスで実装されます。

➤ **CTEASampleDlg**：アプリケーションのダイアログ・ボックスを実装します。

➤ **CTEASampleApp**：アプリケーション自身を実装します。

Testing Agent アプリケーションは，主に **CTEASampleTestableImp** クラスで実装されます。このクラスの開発について，本章で詳細に説明します。

プロジェクトに含まれている一部のコードは, Microsoft Visual Studio によって自動的に生成されたものです（MFC および ATL コード）。

第 1 段階：インフラストラクチャの準備 - 基本的な Testing Agent の作成

Testing Agent 開発の第 1 段階は，インフラストラクチャの作成です。本項では，QuickID アプリケーションの Testing Agent の COM オブジェクトを作成した方法，Testing Extensibility の主要インタフェース（**ITestable**）を実装するようにそのオブジェクトを設定した方法について説明します。

COM オブジェクトの作成

COM オブジェクトは，Microsoft Visual Studio 2005 ウィザードを使用して作成しました。このウィザードには，［**プロジェクト**］＞［**クラスの追加**］を選択してアクセスできます。左側の表示枠で，［ATL］を選択します。右側の表示枠で，［ATL **シンプル オブジェクト**］を選択して［追加］をクリックします。新しい［**ATL シンプル オブジェクト**］クラスを作成するために，次の詳細情報を入力しました。

- 短い名前：CTEASampleTestableImp
- クラス：CTEASampleTestableImp
- コクラス：TEASampleTestableImp
- インタフェース：ITEASampleTestable
- ProgID：TEASample.TEASampleTestableImp

このウィザードは，**CTEASampleModule** クラス（プロジェクトの ATL サポート用）と次のファイルを生成しました。

- TEASampleTestableImp.h
- TEASampleTestableImp.cpp
- TEASampleTestableImp.rgs
- TEASample.idl

これらのファイルは，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\src** フォルダにあります。

COM オブジェクトを Testing Agent にする

COM オブジェクトを Testing Agent にして，QuickTest が環境内のオブジェクトを認識し操作できるようにするには，COM オブジェクトで **AutInterface.idl**（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\Ifs**）に定義されているインタフェースを実装する必要があります。

QuickTest Extensibility Agent type library内の最も重要なインタフェースは，**ITestable** です。このインタフェースは，基本的な QuickTest 機能をサポートするために Testing Agent が実装する必要があるメソッドを定義します。このため，**ITestable** インタフェースは，Testing Agent の開発に必須のインタフェースとみなされます。

初期の開発段階では，QuickID Testing Agent は **ITestable** インタフェースのみ実装するように設計されていました（Testing Agent 開発の後の段階の 128 ページ「第 7 段階 : 記録のサポートの実装」と 134 ページ「第 8 段階 : オブジェクト・スパイのサポートの実装」で，追加のインタフェースが実装されました）。

この目的で，**QuickTest Extensibility Agent** type libraryがインポートされ，**ITestable** インタフェースから継承するように Testing Agent クラスが次のように定義されました。

1 QuickTest Extensibility Agent type libraryのインポート

TEASample.idl ファイルで，importlib("AutInterface.tlb"); という行が TEASampleLib タイプ・ライブラリ定義に追加されました。

タイプ・ライブラリによってインポートされる **AutInterface.tlb** ファイルは，**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\Ifs** フォルダにあります。

2 ITestable インタフェースからの継承

Testing Agent が **ITestable** インタフェースを実装することを定義するために, 次の手順が実行されました。

a **TEASample.idl** ファイルで，**ITestable** インタフェースを実装するように，コクラス **TEASampleTestableImp** の定義が変更されました。

b **TEASampleTestableImp.h** ファイルで次の処理が行われました。

➤ **CTEASampleTestableImp** クラスの継承リストに，次の行が追加されました。

```
public IDispatchImpl<ITestable, &__uuidof(ITestable), &LIBID_AutInterface>
```

➤ **CTEASampleTestableImp** クラスの COM マップに，次の行が追加されました。

```
COM_INTERFACE_ENTRY(ITestable)
```

➤ **ITestable** インタフェース内のすべてのメソッドが，**CTEASampleTestableImp** クラスのメンバとして宣言されました。

c **TEASampleTestableImp.cpp** ファイルで，すべての **ITestable** メソッドが E_NOTIMPL を返す標準設定の実装を使用して追加されました。

第 2 段階：環境の QuickTest への導入

Testing Agent 用のインフラストラクチャを作成したら，次の段階は，環境を QuickTest に導入するために **GetTestingEnvironment** メソッドを実装することです。QuickTest は，Testing Agent のロード時に **GetTestingEnvironment** メソッドを呼び出します。

QuickID Testing Agent 開発のこの段階では, QuickID アプリケーションが表す環境の XML 定義が作成され，この XML 定義を返すために **GetTestingEnvironment** メソッドが実装されました。

環境の定義

Testing Agent を実装する前に, Testing Agent がサポートする必要がある環境を明確に定義する必要があります。この中には，環境に関連するテスト・オブジェクト，QuickTest 内のテスト・オブジェクトを表すアイコン，テスト・オブジェクトの認識プロパティ，各テスト・オブジェクトで実行できる操作などの定義が含まれています。テスト環境を定義する方法の詳細については，第 2 章「QuickTest Testing Agent の計画」を参照してください。

QuickTest では，環境定義を，「**Testing Environment Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。正しい形式の環境定義を含む XML ファイルを作成し，このファイルをロードして XML 文字列として QuickTest に返すように **GetTestingEnvironment** を設計すると便利です。

QuickID 用に作成した XML 環境定義は，**<QuickTest Professional Testing Extensibility インストール・フォルダ>\samples\QuickID\<Visual Studio バージョン>\dat** フォルダにある，**TEASampleEnvironment.xml** ファイルに格納されています。このファイルを開いて，その内容を検証します。

各 **classInfo** 要素は，テスト・オブジェクト・クラスを定義しています。QuickID 用には，次のテスト・オブジェクト・クラスが定義されています。

- **TEASampleApplication**：アプリケーション自身を表す QuickTest テスト・オブジェクト。
- **TEASampleNameEdit**：ユーザ名を入力するためのテキスト・ボックスを表す QuickTest テスト・オブジェクト。
- **TEASampleCitySelector**：都市を選択するためのドロップダウン・メニューを表す QuickTest テスト・オブジェクト。

各テスト・オブジェクト・クラス定義には，テスト・オブジェクト・メソッドと認識プロパティが含まれています。次に例を示します。

- TEASampleCitySelector テスト・オブジェクト・クラス定義には，Select テスト・オブジェクト・メソッドが含まれています。このメソッドは，選択する都市の名前（定義済みのリスト以外）を受け取り，メソッドの実行時に，アプリケーションでその都市を選択します。
- TEASampleApplication テスト・オブジェクト・クラス定義には，DisplayDetails テスト・オブジェクト・メソッドが含まれています。このテスト・オブジェクト・メソッドを実行すると，アプリケーションで［**OK**］ボタンを押す操作をシミュレートして，次の文が表示されます。

  ```
  Your name is <入力した名前>You live in <選択した都市>
  ```

  このテスト・オブジェクト・メソッドは，QuickTest Testing Extensibility の機能と目的のデモです。アプリケーションで［**OK**］ボタンを押すというプリミティブな操作は，テストではより意味のある論理操作で表されます。

➤ TEASampleCitySelector テスト・オブジェクト・クラス定義には，selectedcity というテスト・オブジェクトの認識プロパティが含まれています。このプロパティの値は，利用可能な都市のリスト内の選択した都市のインデックスとなります。Testing Agent がロードされると，QuickTest では，オブジェクト・スパイのポインタで TEASampleCitySelector テスト・オブジェクトをポイントしたときに，このテスト・オブジェクトの認識プロパティを表示します。

GetTesingEnvironment() メソッドの実装

QuickTest が Testing Agent をロードすると，XML 環境定義を返す **GetTestingEnvironment** メソッドを呼び出します。QuickTest は，このメソッドから返される情報を使用して，この環境に使用するテスト・オブジェクトのコレクションを構築します。

QuickID Testing Agent の場合，前述の **TEASampleEnvironment.xml** ファイルをロードしその内容を返すために，**CTEASampleTestableImp::GetTestingEnvironment** メソッド（**TEASampleTestableImp.cpp** ファイル内）が実装されました。

第 3 段階：Testing Agent の QuickTest への登録

Testing Agent は，QuickTest とテスト対象アプリケーション間のインタフェースをとることで，サポートしている環境に対する QuickTest テスト機能を拡張します。ただし，QuickTest が Testing Agent と通信するには，Testing Agent を登録する必要があります。

QuickTest が開くと，登録されている Testing Agent が確認され，アドイン・マネージャにそれらの環境が表示されます。QuickTest ユーザは環境を選択し，Testing Agent の COM オブジェクトを起動することでその環境のサポートをロードするように QuickTest を設定できます。QuickTest は Testing Agent を起動すると，**GetTestingEnvironment** メソッド（115 ページ「第 2 段階：環境の QuickTest への導入」を参照）を呼び出します。

QuickTest Professional Testing Extensibility SDK には，**TeaSDK.lib** ライブラリ（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\lib\<VS バージョン>** フォルダ内）が含まれています。このライブラリは，Testing Agent の登録と登録解除に使用できる **RegisterTeaAut** および **UnRegisterTeaAut** 関数を提供します。

Testing Agent を開発する場合，**TeaSDK.lib** ライブラリをリンクし，**RegisterAut.h** ファイル（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\SDK\include** フォルダ内）をインクルードする必要があります。

QuickID Testing Agent の場合，登録は **CTEASampleApp::InitInstance()** メソッド（**TEASample.cpp** ファイル内）で処理されました。このメソッドは，QuickID アプリケーションを実行するのに使用されます。Register/Unregister TEA Application で始まるメソッドのセクションの関連するコードを見てください。QuickID アプリケーションを /register または /unregister スイッチを使用して実行すると，アプリケーションが呼び出されないことがわかります。代わりに，スイッチに応じて（**RegisterTeaAut** または **UnRegisterTeaAut** を呼び出すことで）Testing Agent が登録または登録解除されます。登録メソッドの使用方法の詳細については，89 ページ「Testing Agent の登録」を参照してください。

注：TEASample の実装では，register スイッチを使用してアプリケーションを実行すると，QuickTest でアプリケーションが登録され，Testing Agent COM オブジェクトがオペレーティング・システムにも登録されます。これは Testing Agent の実装で必要な機能ではありません。ほかの実装では，COM オブジェクトのオペレーティング・システムへの登録は，別に処理する必要がある場合があります。

第 4 段階：基本的な Testing Agent のテスト

基本的な Testing Agent COM オブジェクトを作成し，**GetTestingEnvironment** メソッドを実装して，登録メカニズムを開発したら，Testing Agent を QuickTest で使用する影響をテストできます。

Testing Agent を登録し，QuickTest を実行すると，環境の登録時に入力した表示名が［アドイン マネージャ］ダイアログ・ボックスに表示されます。環境のチェック・ボックスを選択すると，QuickTest が Testing Agent を実行します。

QuickTest が開くと，QuickTest が環境を認識していることがわかります。次に例を示します。

- ［新規テスト オブジェクトの定義］ダイアログ・ボックスで，［**環境**］リストに環境の名前が表示されます。環境をリストから選択すると，テスト環境の XML で定義したテスト・オブジェクト・クラスが［クラス］リストに表示されます。
- ［新規テスト オブジェクトの定義］ダイアログ・ボックスを使用すると，新しいテスト・オブジェクトをオブジェクト・リポジトリに追加できます。オブジェクト・リポジトリに環境内のオブジェクトを表すテスト・オブジェクトを追加したら，定義したテスト・オブジェクトを使用してキーワード・ビューまたはエディタでテストを作成できます。

- エディタでは，ステートメントの自動補完で，テスト・オブジェクトに対して使用できるすべての操作と，これらの操作に対して使用できる入力値が表示されます。この表示は，テスト環境の XML 内の定義に基づいています。
- キーワード・ビューでは，［**項目**］カラムでオブジェクト・リポジトリのオブジェクトを選択すると，［**操作**］カラム内の使用可能な操作のリストに，テスト環境の XML 内の定義が反映されます。操作を選択すると，選択した操作の引数の数に基づいて［**値**］セルが分割されます。操作に可能な値を（ListOfValues 要素で）定義してある場合は，値のリストが表示されます。たとえば，QuickID Testing Agent での作業時に，**TEASampleCitySelector** クラスのテスト・オブジェクトを作成し，そのテスト・オブジェクト上に **Select** 操作を持つテスト・ステップを作成した場合，［**値**］セルをクリックすると，**TEASampleEnvironment.xml** で定義した都市のリストが表示されます。
- テスト環境の XML で定義したテスト・オブジェクト・メソッドに対して記述と注釈の文字列を定義した場合，それらはツールヒントと［注釈］カラムにそれぞれ表示されます。

QuickTest でこれらのオプションを使用する方法の詳細については，『HP Unified Functional Testing ユーザーズ・ガイド』を参照してください。

第 5 段階：テストを実行するためのサポートの追加

Testing Agent 開発のこの時点では，環境に合わせてカスタマイズされたテスト・オブジェクトとテスト・オブジェクト・メソッドを持つテストを作成できますが，これらのテストをまだ実行できません。テストの実行をサポートするには，**ITestable** インタフェースで **FindObjectId** および **Run** メソッドを実装する必要があります。

FindObjectId メソッドは，QuickTest からオブジェクトの記述を受け取り，記述に一致したオブジェクトに即座にアクセスできるオブジェクト ID を返します。詳細については，31 ページ「テスト環境を QuickTest に認識させる方法」および 40 ページ「FindObjectId または FindObjectID2 の実装：記述のオブジェクト ID への割り当て」を参照してください。

QuickTest は，**FindObjectId** が返すオブジェクト ID を Testing Agent メソッドに対する後続の呼び出しで使用します。たとえば，テストの実行時に，QuickTest はテストに格納されているテスト・オブジェクトの記述を使用して **FindObjectId** を呼び出します。次に，返されたオブジェクト ID を使用して **Run** メソッドを呼び出し，テスト・ステップを実行します。詳細については，42 ページ「テストを実行するためのサポートの開発」を参照してください。

オブジェクト ID をオブジェクトに割り当てるには，お好みのメソッドを使用できます。唯一の要件は，QuickTest がオブジェクト ID を使用して Testing Agent メソッドを呼び出す場合に, Testing Agent がテスト対象アプリケーション内の該当するオブジェクトに即座にアクセスできる必要があるということです。

本項では，QuickID Testing Agent が QuickID アプリケーションと通信する方法，および **FindObjectId** と **Run** メソッドを実装した方法について説明します。

Testing Agent とアプリケーション間の通信の設計

QuickID プロジェクトでは, それぞれが QuickID アプリケーション内のオブジェクトを表す，いくつかのクラスが開発されました。QuickID Testing Agent は，アプリケーションと直接通信する代わりに，これらのクラスのオブジェクトと通信します。

この実装のスタイルは，QuickTest Testing Extensibility 実装の一つの方法で，必須ではありません。

QuickID プロジェクトでは，QuickID アプリケーション内のオブジェクトを表す，次のクラスが開発されました。

- **CTeaSampleObject** クラス（**TeaSampleObject.h** および **TeaSampleObject.cpp** ファイル内）は，オブジェクトを表すクラスのすべての基本クラスです。
- **CTeaSampleApplication** クラス（**TeaSampleApplication.h** および **TeaSampleApplication.cpp** 内）は，アプリケーション自身を表します。
- **CTeaSampleNameEdit** クラス（**TeaSampleNameEdit.h** および **TeaSampleNameEdit.cpp** ファイル内）は，ユーザ名を入力するためのテキスト・ボックスを表します。
- **CTeaSampleCitySelector** クラス（**TeaSampleCitySelector.h** および **TeaSampleCitySelector.cpp** ファイル内）は，都市を選択するためのドロップダウン・メニューを表します。

前述のクラスのそれぞれに対して 1 つのオブジェクト（本ガイドではグローバル・オブジェクトと呼ぶ）が作成されました（**Common.h** ファイルの **g_pTEASampleApplication**, **g_pTEASampleNameEdit**, **g_pTEASampleCitySelector** を参照してください)。アプリケーションと Testing Agent のどちらもこれらのオブジェクトにアクセスできます。各グローバル・オブジェクトは，それが表すオブジェクトに関して QuickTest が必要とするすべての情報（タイプ，名前，状態など）を格納しています。

Testing Agent メソッドのほとんどは，要求をこれらのグローバル・オブジェクトの 1 つに委任するためだけに実装されました。

FindObjectId Method メソッドの実装

QuickID アプリケーション上でのテストの実行のサポートを開発するために，**TEASampleTestableImp** クラス内の **FindObjectId** および **Run** メソッドの標準設定の実装がより意味のある実装で置き換えられました。

QuickID では，アプリケーションが単純なため，整数の定数がオブジェクト ID として使用されました。運用環境の Testing Agent では，オブジェクト ID は通常動的で，実行時にしかわかりません（hWnd など）。

QuickID アプリケーション・コントロール用のオブジェクト ID は，**Common.h** ファイルで次のように定義されました。

- TEASampleApplication（0）- アプリケーション・オブジェクト用
- TEASampleNameEdit（1）- ユーザ名を入力するためのテキスト・ボックス用
- TEASampleCitySelector（2）- 都市を選択するためのドロップダウン・メニュー用

QuickID Testing Agent では，**FindObjectId** の実装は，**TEASampleTestableImp.cpp** ファイルの **TEASampleTestableImp** クラス内にあります。メソッドは次のように実装されます。

1. **FindObjectId** は, QuickTest 提供の記述を使用して **GetElementByDescription** を呼び出します。
2. **GetElementByDescription** は, 記述に一致するアプリケーション内のオブジェクトを表すグローバル・オブジェクトを返します。
   QuickTest では，記述を，「**Description Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供します。この記述は，選択したオブジェクトとその先祖を再帰的に記述します。**GetElementByDescription** が記述を解析して，階層内の最下位オブジェクトを見つけ, その最下位オブジェクトに指定されているタイプに対応するグローバル・オブジェクトを返します。
3. **FindObjectId** がグローバル・オブジェクトの **GetObjectID** 関数を呼び出します。この関数は，それが表すアプリケーション内のオブジェクトのオブジェクト ID（整数）を返します（このオブジェクト ID は，作成時にグローバル・オブジェクトで初期化されます）。

独自の Testing Agent を設計する場合，実装はより複雑になる可能性がありますが，原則は同じです。

Run メソッドの実装

FindObjectId メソッドの実装後，**TEASampleTestableImp** クラス内の **Run** メソッドの標準設定の実装が，QuickID アプリケーション上でのテストの実行をサポートするより意味のある実装で置き換えられました。

QuickTest が **Run** メソッドを呼び出して，テスト・ステップを実行します。**Run** メソッドは次のものを受け取ります。

- ステップを実行する必要があるアプリケーション・オブジェクトのオブジェクト ID
- 実行するテスト・オブジェクト・メソッドの名前
- テスト・オブジェクト・メソッドの引数の SAFEARRAY リスト

たとえば，QuickID アプリケーションでテストを実行する場合，**Run** メソッドは，TEASampleCitySelector テスト・オブジェクト，Select テスト・オブジェクト・メソッド，引数 2（都市リストの NewYork のインデックス・エントリ）を使用して呼び出すことができます。

QuickID Testing Agent では，**Run** メソッドは，**TEASampleTestableImp.cpp** ファイルの **TEASampleTestableImp** クラス内に実装されます。次のように実装されます。

1. Testing Agent の **Run** メソッドが，受け取ったオブジェクト ID を使用して **GetElementByRuntimeID** を呼び出します。
2. **GetElementByRuntimeID** が，オブジェクト ID の参照先のアプリケーション・オブジェクトを決定し，そのアプリケーション・オブジェクトを表すグローバル・オブジェクトを返します。
3. Testing Agent の **Run** メソッドが，QuickTest が渡したすべてのパラメータを使用してグローバル・オブジェクトの **Run** メソッドを呼び出します。
4. グローバル・オブジェクトの **Run** メソッドが，**TEASampleObject.cpp** ファイル内の **CTEASampleObject** 基本グローバル・クラスに実装されます。このメソッドは，要求を QuickID アプリケーション・ダイアログ・オブジェクトの **RunMethod** 関数に委任します。これは，**TEASampleDlg.cpp** ファイルに実装されている **CTEASampleDlg** クラスのオブジェクトです。この **CTEASampleObject** オブジェクトが，実際にはテスト・ステップの操作を実行します。
5. **RunMethod** が，ステップを実行する必要があるアプリケーション・オブジェクトおよび実行する必要があるテスト・オブジェクト・メソッドを決定し，適切なアプリケーション・オブジェクト上でテスト・オブジェクト・メソッドを実行します。

たとえば，**RunMethod** を **TEASampleCitySelector** オブジェクト，Select テスト・オブジェクト・メソッド，リスト内の 1 つの都市のインデックス・エントリを使用して呼び出すと，**RunMethod** は指定されたインデックス・エントリを使用して **CTEASampleDlg::SelectCity** を呼び出します。**SelectCity** は，ドロップダウン・メニュー・コントロールに対するポインタを取得し，選択する都市のインデックスを使用してその **SetCurSel** を呼び出します。**SetCurSel** は，都市の選択を実行します。

また，**SelectCity** は，ドロップダウン・メニュー・コントロールを表す **g_pTEASampleCitySelector** グローバル・オブジェクトの **selectedCity** プロパティも更新します。これにより，グローバル・オブジェクトは，QuickTest からの要求時に **selectedCity** プロパティの正しい値を提供できるようになります。

RunMethod が **TEASampleApplication** オブジェクトと DisplayDetails 操作を使用して呼び出されると，**CTEASampleDlg::OnBnClickedButtonok** を呼び出すことで，［**OK**］ボタンをクリックする操作をシミュレートします。アプリケーションの **OnBnClickedButtonok** メソッドは, 該当する情報をテキスト･ボックスに表示します。

第 6 段階：追加の ITestable インタフェース・メソッドの実装

本項では，Testing Agent に実装する必要がある **ITestable** インタフェースの追加のメソッドについて説明します。QuickTest で呼び出す場合の各メソッドの使用方法，および各メソッドを QuickID Testing Agent に実装した方法について説明します。

GetParent

GetParent メソッドは，オブジェクト ID を受け取り，オブジェクトの親のオブジェクト ID を返します。詳細については，38 ページ「外部の親の概念について」および 40 ページ「GetParent の実装」を参照してください。

QuickTest は **GetParent** メソッドを呼び出して, テスト･オブジェクト階層を構築します。たとえば，QuickTest ユーザがポイントするアプリケーション内の場所からオブジェクトを学習する場合などです。QuickTest では，関連するテスト・オブジェクトの階層を［オブジェクトの選択］ダイアログ・ボックスに表示し，QuickTest ユーザが正しいオブジェクトを選択できるようにします。この階層を構築するために，各テスト・オブジェクトに対して **GetParent** が呼び出されます。

QuickID Testing Agent では，**GetParent** メソッドは，**TEASampleTestableImp.cpp** ファイルの **TEASampleTestableImp** クラス内に実装されています。次のように実装されました。

1 Testing Agent の **GetParent** メソッドが，受け取ったオブジェクト ID を使用して **GetElementByRuntimeID** を呼び出します。

2 **GetElementByRuntimeID** が，オブジェクト ID の参照先のアプリケーション・オブジェクトを決定し，そのアプリケーション・オブジェクトを表すグローバル・オブジェクトを返します。

3 Testing Agent の **GetParent** メソッドが，受け取ったオブジェクト ID を使用してグローバル・オブジェクトの **GetParent** メソッドを呼び出します。グローバル・オブジェクトの **GetParent** メソッドが，**CTEASampleObject** 基本グローバル・クラスに実装されます。**TEASampleObject.h** ファイルを参照してください。

4 グローバル・オブジェクトの **GetParent** メソッドが，構築時に親として定義されたグローバル・オブジェクトを返します。QuickID では，次の 2 つの戻り値しかありません。

- NULL：入力オブジェクトがアプリケーション・オブジェクトで，親がない場合。
- **g_pTEASampleApplication**：入力オブジェクトがアプリケーション・コントロールの 1 つであった場合。**g_pTEASampleApplication** は，アプリケーション自身を表すグローバル・オブジェクトです。

5 グローバル・オブジェクトの **GetParent** メソッドが NULL を返す場合，Testing Agent の **GetParent** メソッドは VT_EMPTY を返し，指定したオブジェクトがトップレベル・オブジェクトで，親がないことを示します。

それ以外の場合，Testing Agent の **GetParent** メソッドが親のグローバル・オブジェクトの **GetObjectID** を呼び出し，それが表すアプリケーション内のオブジェクトのオブジェクト ID を返します。この場合，メソッドはアプリケーション・コントロールのオブジェクト ID を返します。

GetChildren

GetChildren メソッドは，オブジェクト ID を受け取り，その子のオブジェクト ID のリストを返します。詳細については，41 ページ「GetChildren の実装」を参照してください。

QuickTest が **GetChildren** を呼び出すのは，ChildObjects 操作を使用してテスト・ステップを実行する場合などです。

QuickID Testing Agent での **GetChildren** メソッドの実装は，**TEASampleTestableImp** クラスで確認できるように，**GetParent** の実装に似ています。

GetDisplayName

GetDisplayName メソッドは，オブジェクト ID を受け取り，その指定されたオブジェクトに対して定義されている表示名を返します。詳細については，39 ページ「GetDisplayName の実装：オブジェクト名の戻し」を参照してください。

QuickTest は，**GetDisplayName** から返された名前をオブジェクト・リポジトリ，キーワード・ビュー，エディタなどに表示します。QuickTest はまた，［オブジェクトの選択］ダイアログ・ボックスに表示される階層の構築時に **GetDisplayName** を繰り返し呼び出します。

QuickID Testing Agent 内の **GetDisplayName** メソッドの実装は，**TEASampleTestableImp** クラスで確認できるように，**GetParent** の実装に似ています。

GetElementType

GetElementType メソッドは，オブジェクト ID を受け取り，それが表すテスト・オブジェクト・クラスを返します。詳細については，48 ページ「GetElementType と GetProperties の実装」を参照してください。

QuickTest は，**GetElementType** から返された名前をオブジェクト・リポジトリ，キーワード・ビュー，エディタなどに表示します。QuickTest はまた，［オブジェクトの選択］ダイアログ・ボックスに表示される階層の構築時に **GetElementType** を呼び出し，階層内の各オブジェクトのテスト・オブジェクト・クラスを取得します。

QuickID Testing Agent 内の **GetElementType** メソッドの実装は，**TEASampleTestableImp** クラスで確認できるように，**GetParent** の実装に似ています。

GetProperties

GetProperties メソッドは，オブジェクト ID とプロパティ名の SAFEARRAY を受け取り，リストされたプロパティの値を返します。詳細については，48 ページ「GetElementType と GetProperties の実装」を参照してください。

QuickTest は，認識プロパティをオブジェクト・スパイに表示する場合や，**GetROProperty**（または **GetROProperties**）操作を使用してテスト・ステップを実行する場合などに，GetProperties を呼び出します。

QuickID Testing Agent では，**GetProperties** メソッドは，**TEASampleTestableImp.cpp** ファイルの **TEASampleTestableImp** クラス内に実装されます。次のように実装されました。

1 **GetProperties** メソッドが **GetElementByRuntimeID** を呼び出し，**GetProperties** が受け取ったオブジェクト ID を渡します。

2 **GetElementByRuntimeID** が，オブジェクト ID の参照先のアプリケーション・オブジェクトを決定し，そのアプリケーション・オブジェクトを表すグローバル・オブジェクトを返します。

3 受け取ったリスト内の各プロパティ名について，Testing Agent の **GetProperties** がそのプロパティ名を使用して，グローバル・オブジェクトの **GetProperty** メソッドを呼び出します。

4 グローバル・オブジェクトの **GetProperty** メソッドが，**TEASampleObject.cpp** ファイル内の **CTEASampleObject** 基本グローバル・クラスに実装されます。グローバル・オブジェクトのプロパティは，オブジェクトの作成時に初期化され，アプリケーションが変更されるたびに更新されるため，**GetProperty** はグローバル・オブジェクトから直接プロパティの値を返します。

BuildDescription

BuildDescription メソッドは，オブジェクト ID を受け取り，アプリケーション内でこのオブジェクトを一意に識別する記述を返します。この記述は，指定したテスト・オブジェクトのテスト・オブジェクト階層全体を再帰的に記述する必要があります。オブジェクトの記述は，「**AppDescription Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。

この記述に加えて，**BuildDescription** では，**externalParent** 引数を設定して，Testing Agent のテスト環境のテスト・オブジェクト階層をテスト対象アプリケーションのより大きなテスト・オブジェクト階層に統合するように，QuickTest を設定できます。QuickID アプリケーションはスタンドアロン・アプリケーションであるため，エージェントは **externalParent** 引数を設定しません。

QuickTest は，オブジェクトをオブジェクト・リポジトリに追加する前に **BuildDescription** メソッドを呼び出します。詳細については，39 ページ「BuildDescription の実装：オブジェクト ID の記述への割り当て」を参照してください。

QuickID Testing Agent では，**BuildDescription** メソッドは，**TEASampleTestableImp.cpp** ファイルの **TEASampleTestableImp** クラス内に実装されます。次のように実装されました。

1 **BuildDescription** メソッドが，受け取ったオブジェクト ID を使用して **GetElementByRuntimeID** を呼び出します。

2 **GetElementByRuntimeID** が，オブジェクト ID の参照先のアプリケーション・オブジェクトを決定し，そのアプリケーション・オブジェクトを表すグローバル・オブジェクトを返します。

3 Testing Agent の **BuildDescription** メソッドが，QuickTest が渡したすべてのパラメータを使用してグローバル・オブジェクトの **BuildDescription** メソッドを呼び出します。

4 グローバル・オブジェクトの **BuildDescription** メソッドが，**TEASampleObject.cpp** ファイル内の **CTEASampleObject** クラスに実装されます。グローバル・オブジェクトには，オブジェクトの記述の構築に必要なすべての情報（認識プロパティの名前と値，親のオブジェクト ID）が含まれています。**BuildDescription** メソッドは，「**AppDescription Schema**」に従って XML オブジェクトの記述を再帰的に構築します。記述には，テスト環境の XML で for description 属性を使用して定義された認識プロパティのみが含まれます。

GetLastError

QuickTest は，Testing Agent がエラー・コードを返す場合に **GetLastError** メソッドを呼び出します。**GetLastError** メソッドは，QuickTest ユーザに対して表示可能で，前のアクションの結果発生したエラーを説明するメッセージを返す必要があります。詳細については，44 ページ「GetLastError メソッドの実装：エラーの報告」を参照してください。

QuickID Testing Agent は，そのメソッドの 1 つがエラー・コードを返すときに，エラー・メッセージをプライベート・データ・メンバ（**m_bsLastError**）に保存するように実装されました。QuickTest が **GetLastError** を呼び出すと，Testing Agent は単にこのデータ・メンバの値を返します。

ITestable インタフェースの次のメソッドは，QuickID Testing Agent に完全には実装されませんでした。

- **CompareObjectIds** および **LearnChildObjects**：QuickTest で QuickID アプリケーションが表す環境のオブジェクトを学習し，それらのオブジェクトをオブジェクト・リポジトリに追加できるようにします。
- **GetTabularData** および **GetTabularAttributes**：テーブル・チェックポイントおよびテーブルからのデータの取得をサポートします。

QuickID Testing Agent の作成プロセスから学習したことに基づき, 必要に応じて QuickTest のこれらの基本機能を完全にサポートするために，これらのメソッドを Testing Agent に実装できます。

第 7 段階：記録のサポートの実装

前の段階では，Testing Agent の作成と **ITestable** インタフェースの実装，QuickTest が新しい環境を認識し，アプリケーションでテストを実行できるようにすることについて説明しました。本段階までは，オブジェクト・リポジトリで定義したテスト・オブジェクトまたはエディタでのテスト・オブジェクトのプログラム的記述に基づいて，テストを手動で作成する必要がありました。

QuickTest ではまた, アプリケーション上で実行される一連の操作を記録することで, ユーザがテストを自動的に作成することもできます。QuickTest は，実行する操作を表すテスト・ステップを作成します。

QuickTest の記録オプションをサポートするために，Testing Agent では，**IRecordable** インタフェースを実装し，**Record** API メソッドを呼び出して QuickTest にユーザの操作に基づいて作成するテスト・ステップを指示し，**Suppress**/**UnSuppress** API メソッドを使用して QuickTest が同じ操作の汎用テスト・ステップを記録しないようにする必要があります。詳細については，71 ページ「記録のサポートの開発」を参照してください。

本項では, 記録をサポートするために QuickID Testing Agent を拡張した方法について説明します。

IRecordable インタフェースの実装

IRecordable インタフェースを実装するために，次のステップが実行されました。

Testing Agent による IRecordable インタフェースの実装の宣言

TEASampleTestableImp.h ファイルで，QuickID Testing Agent が（**ITestable** インタフェースに加えて）**IRecordable** インタフェースを実装することを定義するために, 次の手順が実行されました。

➤ **CTEASampleTestableImp** クラスの継承リストに，次の行が追加されました。

```
public IDispatchImpl<IRecordable, &__uuidof(IRecordable),
&LIBID_AutInterface>
```

➤ **CTEASampleTestableImp** クラスの COM マップに，次の行が追加されました。

```
COM_INTERFACE_ENTRY(IRecordable)
```

➤ **IRecordable** インタフェース内のメソッド（**BeginRecording**，**EndRecording**）が **CTEASampleTestableImp** クラスのメンバとして宣言されました。

QuickTest Extensibility type library ヘッダ・ファイルのインクルード

Testing Agent が **QuickTest Extensibility** type libraryからメソッドを呼び出せるように，次の行が **TEASampleTestableImp.h** ファイルに追加されました。

```
#include "QtpInterface.h"
```

IRecordable インタフェース・メソッドの実装

BeginRecording および **EndRecording** メソッドが **TEASampleTestableImp.cpp** ファイルに実装されました。

QuickTest は，ユーザが記録セッションを開始または終了すると，**BeginRecording** または **EndRecording** を呼び出します。このため，**BeginRecording** および **EndRecording** メソッドを実装すると，記録セッションに関連する変数の初期化またはリセット，動的記録抑制のアクティブ化または非アクティブ化を行うことができます。

記録セッション時に，Testing Agent は，ユーザがアプリケーション上で実行する操作に基づいてステップをテストに追加するように QuickTest に指示できます。これを行うために，Testing Agent は，QuickTest が **IRecorder** インタフェースで提供する **Record** API コールバック・メソッドを呼び出します。このインタフェースは，**QuickTest Extensibility** type libraryで定義されています。

また，Testing Agent は，**IRecordSuppressor** インタフェースから **Suppress**/**UnSuppress** API メソッドを呼び出すことで，アプリケーション上でのネイティブ・ステップの記録を抑制するように QuickTest に指示できます。詳細については，73 ページ「ステップの記録」および 74 ページ「QuickTest のネイティブの記録の抑制」を参照してください。

BeginRecording メソッドは，**IRecorder** および **IRecordSuppressor** インタフェースを公開するオブジェクトへの参照を受け取ります。Testing Agent は，この参照を使用して，**IRecorder** および **IRecordSuppressor** インタフェース・メソッドにアクセスします。

QuickID Testing Agent では，**BeginRecording** および **EndRecording** メソッドは，**CTEASampleTestableImp** クラス内に実装されます。次のように実装されます。

QuickID Testing Agent（このクラス内）で，次のように実装されました。

- **BeginRecording** は次の目的で実装されました。
  - **IRecorder** および **IRecordSuppressor** インタフェース・メソッドの呼び出し時に使用できるように，受け取ったオブジェクト参照をクラスの **m_spRecorder** データ・メンバに保存します。
  - **CTEASampleTestableImp::SuppressWindow** メソッドを呼び出し，このメソッドがネイティブの記録抑制を処理します（詳細については，132 ページ「ネイティブの記録の抑制と抑制解除」を参照してください）。
- **EndRecording** は，**CTEASampleTestableImp::UnSuppressWindow** メソッドを呼び出してネイティブの記録抑制をオフにするように実装されました。詳細については，132 ページ「ネイティブの記録の抑制と抑制解除」を参照してください。

QuickTest に対するステップの記録の指示

記録セッション時に，Testing Agent は，ユーザがアプリケーション上で実行する操作に基づいてステップをテストに追加するように QuickTest に指示できます。これを行うために，Testing Agent は，QuickTest が **IRecorder** インタフェース（**QuickTest Extensibility** type libraryで定義されます）で提供する **Record** API コールバック・メソッドを呼び出します。詳細については，73 ページ「ステップの記録」を参照してください。

QuickID Testing Agent では，記録のサポートは次のように設計されました。

TEASample.idl ファイルでは，Testing Agent に対してローカルの新しいインタフェースが定義されます。これが **ITEASampleTestable** インタフェースですこのインタフェースは，アプリケーションと Testing Agent 間の通信を定義します。QuickID アプリケーションを実行すると，Testing Agent オブジェクトが作成され，Testing Agent の **ITEASampleTestable** インタフェースに対する参照がローカル・メンバの **m_spTea** に格納されます。**ITEASampleTestable** インタフェースは，**Record** 関数を公開します。

QuickID アプリケーションは，QuickTest ユーザがそのステップを記録する必要がある操作を実行するときに，常に **ITEASampleTestable** インタフェースの **Record** メソッドを呼び出すように設計されました。Testing Agent は，**IRecorder** インタフェースの **Record** メソッドを呼び出すことで，この要求を QuickTest に委任します。**IRecorder** インタフェースは，エージェントが **BeginRecording** で受け取ったオブジェクト参照によって公開されます。

次に例を示します。

QuickTest ユーザが記録セッションを開始し，ドロップダウン・メニュー・コントロールに表示される都市のリストから都市を選択すると，アプリケーションの **OnCbnSelchangeComboCity** ハンドラ関数が呼び出されます。

OnCbnSelchangeComboCity は，ドロップダウン・メニュー・コントロールのオブジェクト ID，操作名（Select），操作の引数を使用して，Testing Agent の **ITEASampleTestable::Record** メソッドを呼び出します。この場合の引数は，選択した都市のインデックスとなります。**OnCbnSelchangeComboCity** の詳細については，**TEASampleDlg.cpp** ファイルを参照してください。

Testing Agent の **Record** メソッドは，受け取ったパラメータとこのメソッドに必要な追加のパラメータを使用して，**IRecorder** インタフェースから **Record** メソッドを呼び出します。**Record** メソッドの構文の詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。

提供する必要がある追加のパラメータは，操作を実行したオブジェクトの記述，このオブジェクトとルート・オブジェクトまでのそのすべての先祖のオブジェクト ID を含む SAFEARRAY です。オブジェクトの記述は，「**AppDescription Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。ドロップダウン・メニュー・コントロールとその親のアプリケーション・ダイアログ・ボックスのオブジェクト ID のリストを生成するために，**CTEASampleObject** クラスにアプリケーション固有のメソッド（**GetObjectIDHierarchy** 関数）が追加されました。

Testing Agent は，QuickTest に対してユーザの各操作で記録するステップを指示します。これにより，Testing Agent はアプリケーションの論理テスト・モデルをサポートできるようになります。たとえば，ユーザがアプリケーション上で［**OK**］ボタンをクリックしたときに記録される操作は，**DisplayDetails** であって，物理的な **Click** 操作ではありません。これは **TEASampleDlg.cpp** ファイル内の **OnBnClickedButtonok** メソッドに実装されます。このメソッドは，**IRecorder** インタフェースから **Record** メソッドを呼び出して，QuickTest テスト・ステップを記録します。

テキスト・ボックスで実行された操作を記録するための実装は，**TEASampleDlg.cpp** ファイル内の **OnEnKillfocusEditName** ハンドラ・メソッドに実装されました。

ITEASampleTestable インタフェースは，**IsRecording** メソッドも公開します。このメソッドは，記録セッションが現在アクティブであれば S_OK を，それ以外の場合に S_FALSE を返すように，QuickID Testing Agent によって実装されました。Testing Agent の **Record** メソッドを呼び出してステップを記録する前に，アプリケーションはこのメソッドを呼び出して記録セッションがアクティブかどうかを確認します。アクティブな記録セッションがない場合, **Record** メソッドは呼び出されません。これにより, QuickTest がロードされていて記録セッションがアクティブでない限り, Testing Agent はユーザがアプリケーション上で実行するすべての操作のテスト・ステップを記録しないようになります。

ネイティブの記録の抑制と抑制解除

QuickTest は，アプリケーション上で実行される標準 Windows 環境のすべての操作のステップをネイティブに記録します。Testing Agent もアプリケーション固有のステップを記録します。同じ操作に対して 2 つのテスト・ステップを記録しないように，Testing Agent では，ネイティブの QuickTest 記録を抑制する必要があります。

たとえば，次のテストは，QuickID Testing Agent に抑制を実装する前に記録されました。QuickTest ユーザが記録セッションを開始し, QuickID アプリケーション名テキスト・ボックスに「123」と入力して，都市リストから Tokyo を選択しました。

```
Dialog("Who are you?").WinEdit("Enter your name").Set "123"
TEASampleApplication("TheApplication").
    TEASampleNameEdit("TheNameEdit").Set "123"
Dialog("Who are you?").WinComboBox("Select your city").Select "Tokyo"
TEASampleApplication("TheApplication").TEASampleCitySelector("TheCitySele
ctor").Select Tokyo
```

各操作に対して 2 つのステップが記録されました。1 つは Testing Agent の指示に基づくもので，1 つはアプリケーションを標準 Windows アプリケーションとして認識している QuickTest に基づくものです。

静的と動的の 2 つのタイプの記録抑制が利用可能です。詳細については，74 ページ「QuickTest のネイティブの記録の抑制」を参照してください。

QuickID Testing Agent は動的抑制を使用します。これを行うために，Testing Agent は，**QuickTest Extensibility** type libraryで定義され，**IRecordSuppressor** インタフェースで提供される **Suppress**/**UnSuppress** メソッドを呼び出します。**Suppress** メソッドは，標準 Windows イベントを無視する環境の ID，標準イベントを無視するプロセス ID や hWnd のリストを受け取ります。プロセス ID のリストは，「**Suppression Schema**」（『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照）に従って XML 形式で提供する必要があります。

UnSuppress メソッドも同様に動作しますが，指定したプロセスや hWnd で発生する抑制をキャンセルします。

QuickID Testing Agent では，抑制は **TEASampleTestableImp.cpp** ファイルで次のように処理されます。

a **BeginRecording** メソッドが，**SuppressWindow** プライベート・メソッドを呼び出します。

b **SuppressWindow** が，プロセス ID の XML リストを作成し文字列として返す **GetWindowSuppression** を呼び出します。抑制がある hWnd 上でアクティブ化されている場合，そのウィンドウおよびその子オブジェクト上の標準 Windows イベントは記録されません。したがって，このサンプルでは，アプリケーションのメイン・ウィンドウをその hWnd でクエリして，この hWnd だけで XML リストを作成することで十分でした。

c **SuppressWindow** が，この XML リストを使用して **IRecordSuppressor** インタフェースから **Suppress** メソッドを呼び出します。**IRecordSuppressor** インタフェースへのアクセスは，エージェントが **BeginRecording** で受け取ったオブジェクト参照を通じて行われます。

d 同様に，**EndRecording** メソッドが **UnSuppressWindow** プライベート・メソッドを呼び出し，このプライベート・メソッドが，前述のように該当する XML リストを使用して **IRecordSuppressor** インタフェースの **UnSuppress** メソッドを呼び出します。

第 8 段階：オブジェクト・スパイのサポートの実装

オブジェクト・スパイの指差しメカニズムを使用すると，QuickTest ユーザは，開いているアプリケーション内の任意の可視なオブジェクトのサポートされているプロパティと操作を表示できます。ユーザが指差しアイコンをアプリケーション内のオブジェクト上に移動すると，その詳細がオブジェクト・スパイに表示されます。これらの詳細には，テスト・オブジェクトの階層ツリー，そのプロパティと値，その操作が含まれます。

QuickTest は，テスト・オブジェクトの詳細をオブジェクト・スパイに表示するために，前述の **ITestable** インタフェースのいくつかのメソッド（**GetDisplayName**，**GetElementType**，**GetParent**，**GetProperties** など）を呼び出します。ただし，QuickTest がユーザがポイントしている場所にどのアプリケーション・オブジェクトがあるかを決定するために，Testing Agent では **ISpyable** インタフェースも実装する必要があります。**ISpyable** インタフェースには，**GetElementFromPoint** メソッドだけが含まれています。このメソッドは，ユーザがポイントした場所（アプリケーション上）の座標を受け取り，そのポイントにあるアプリケーション・オブジェクトのオブジェクト ID を返します。詳細については，61 ページ「GetElementFromPoint の実装」を参照してください。

QuickID Testing Agent では，オブジェクト・スパイのサポートは次のように実装されました。

1 **TEASampleTestableImp.h** ファイルで次の処理が行われました。

- **ISpyable** インタフェースを実装することを示すために，**CTEASampleTestableImp** クラスの継承リストに，次の行が追加されました。

```
public IDispatchImpl<ISpyable, &__uuidof(ISpyable), &LIBID_AutInterface>
```

- **CTEASampleTestableImp** クラスの COM マップに，次の行が追加されました。

```
COM_INTERFACE_ENTRY(ISpyable)
```

- **ISpyable** インタフェース内のすべてのメソッドが，**CTEASampleTestableImp** クラスで宣言されました。

2 Testing Agent の **GetElementFromPoint** メソッドが，**TEASampleTestableImp.cpp** ファイルに次のように実装されました。

a Testing Agent の **GetElementFromPoint** メソッドが，アプリケーション自身を表すグローバル・オブジェクトである **g_pTEASampleApplication** オブジェクトの **GetElementFromPoint** を呼び出します。

b グローバル・オブジェクトの **GetElementFromPoint**（**CTEASampleObject** クラスに実装されています。**TEASampleObject.cpp** ファイルを参照）が，指定したポイントを含む最も内側のアプリケーション・オブジェクトのグローバル・オブジェクトを返します。

該当するオブジェクトを見つけるために，**GetElementFromPoint** は，指定したポイントを含む最も内側のオブジェクトを見つけるまで，すべてのグローバル・オブジェクトを再帰的に確認します。該当するオブジェクトが見つかった場合，それが返されます。見つからなかった場合，そのポイントがアプリケーションに含まれていれば，メソッドはアプリケーション自身を表すグローバル・オブジェクトを返します。

ポイントが指定したオブジェクトに含まれているかどうかを調べるために，**CTEASampleObject::GetElementFromPoint** は，そのポイントとオブジェクト ID を使用して **CTEASampleDlg::IsPointInObject** を呼び出します。**IsPointInObject** は，オブジェクト ID を受け取り，オブジェクトを含む矩形を返す **GetRectangle** プライベート・メソッドを呼び出します。次に，**IsPointInObject** が指定したポイントが矩形に含まれているかどうかを確認し，回答を返します。

c Testing Agent の **GetElementFromPoint** メソッドは，**g_pTEASampleApplication** の **GetElementFromPoint** から返されたグローバル・オブジェクトを受け取り，それが表すオブジェクトのオブジェクト ID をクエリして，このオブジェクト ID を返します。

第 9 段階：アプリケーション内のオブジェクトの強調表示のサポートの実装

QuickTest ユーザは，オブジェクト・リポジトリ内のオブジェクトを選択し，テスト対象アプリケーションでこのオブジェクトを強調表示するように QuickTest を設定できます。これにより，ユーザは QuickTest が確実にテスト・オブジェクトをアプリケーション内の正しいオブジェクトに関連付けるようにできます。詳細については，77 ページ「オブジェクトを強調表示するためのサポートの開発」を参照してください。

QuickTest がアプリケーション内のテスト・オブジェクトを強調表示するために，Testing Agent は，**IHWNDSupplier** インタフェースで **GetHWND** メソッド，**IRectangleSupplier**

インタフェースで **GetRectangle** メソッド，そして **ITestable** インタフェースを実装する必要があります。

GetHWND メソッドは，強調表示する必要があるオブジェクトのオブジェクト ID を受け取り，このオブジェクトを含む最も下位のウィンドウの hWnd を返します。QuickTest はこの hWnd を使用して，強調表示する必要があるオブジェクトを可視にします。

GetRectangle メソッドは，オブジェクトのオブジェクト ID を受け取り，オブジェクトを囲む仮想矩形の左上と右下の各ポイントの座標を含む **RECT** 構造を返します。詳細については，『QuickTest Testing Extensibility API Reference』（**<QuickTest Professional Testing Extensibility SDK インストール・フォルダ>\help\QTP_TestExt_Reference.chm**）を参照してください。QuickTest はこの情報を使用して，アプリケーション内のオブジェクトの強調表示に使用する矩形を描画します。

QuickID Testing Agent では，オブジェクトの強調表示は次のようにサポートされました。

1 **TEASampleTestableImp.h** ファイルで次の処理が行われました。

- **IRectangleSupplier** インタフェース（および継承元の **IHWNDSupplier** インタフェース）を実装することを示すために，**CTEASampleTestableImp** クラスの継承リストに，次の行が追加されました。

```
public IDispatchImpl<IRectangleSupplier, &__uuidof(IRectangleSupplier),
&LIBID_AutInterface>
```

- **CTEASampleTestableImp** クラスの COM マップに，次の行が追加されました。

```
COM_INTERFACE_ENTRY(IRectangleSupplier)
COM_INTERFACE_ENTRY2(IHWNDSupplier, IRectangleSupplier)
```

- **IRectangleSupplier** および **IHWNDSupplier** インタフェース内のすべてのメソッドが，**CTEASampleTestableImp** クラスのメンバとして宣言されました。

2 Testing Agent の **GetRectangle** メソッドが，**TEASampleTestableImp.cpp** ファイルに次のように実装されました。

a Testing Agent の **GetRectangle** メソッドが，受け取ったオブジェクト ID を使用して **GetElementByRuntimeID** を呼び出します。

b **GetElementByRuntimeID** が，オブジェクト ID の参照先のアプリケーション・オブジェクトを決定し，そのアプリケーション・オブジェクトを表すグローバル・オブジェクトを返します。

c Testing Agent の **GetRectangle** メソッドが，グローバル・オブジェクトの **CTEASampleObject::GetRectangle** メソッドを呼び出します。このメソッドは，**TEASampleObject.cpp** ファイル内に実装されています。

d グローバル・オブジェクトの **GetRectangle** メソッドが，オブジェクト ID を使用して要求を **CTEASampleDlg::GetRectangle** メソッドに委譲します。**CTEASampleDlg** はアプリケーションのダイアログ・オブジェクトで，**TEASampleDlg.cpp** ファイルに実装されています。

e **CTEASampleDlg** の **GetRectangle** メソッドがオブジェクト ID を受け取り，アプリケーション内の該当するオブジェクトを囲む矩形を返します。

3 Testing Agent の **GetHWND** メソッドは，**GetRectangle** を実装したのと同じようにして **TEASampleTestableImp.cpp** ファイルに実装されました。常にアプリケーション・ダイアログ・ボックスの hWnd を返します。

その他の情報

本章では, QuickID Testing Agent を開発した方法を学習しました。このプロセスを通じて, Testing Agent のインフラストラクチャの作成方法, 新しい環境の QuickTest への導入方法, この環境に属しているアプリケーション上での環境固有の QuickTest テストの実行のサポート方法について学習しました。ほとんどの **ITestable** メソッドの実装方法，その他の QuickTest 機能（オブジェクト・スパイ，テストの記録，アプリケーション内のオブジェクトの強調表示）のサポート方法についても学習しました。

QuickTest Testing Extensibility では，QuickID Testing Agent がサポートしていないその他の QuickTest 機能をサポートできます。たとえば，ActiveScreen，オブジェクトの学習，ネイティブのプロパティと操作へのアクセス，スマート認識などのサポートも作成できます。

このレッスンと QuickID Testing Agent 設計の目的は，実用的な Testing Agent を設計する方法を手順ごとに明瞭に説明し, QuickTest Testing Extensibility を実装するための基本事項を学習できるようにすることです。これまで学習したプロセスと概念を使用して，独自の Testing Agent を設計できます。さまざまな QuickTest 機能に対する Testing Agent のサポートを開発する方法の詳細については，このガイドのほかの章を参照してください。