|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| solutiont | Microsoft Active Accessibility Verification Tool (MsaaVerify) Design Specification | | | |
|  |  | | Last updated: 1/8/2007 7:25 PM |
|  | | | | |
| **Program Manager** | | | SaraF@Microsoft.com | |
| **Project Homepage** | | | <http://www.codeplex.com/MsaaVerify> | |

This is the design specification for the Microsoft Active Accessibility Verification Tool codenamed MsaaVerify located on CodePlex.

1. Overview 1

1.1 Definitions 1

1.1.1 MSAA 1

1.2 Goals 1

1.2.1 Release 1.0 1

1.2.2 Future Releases 1

1.3 Non-goals 1

1.4 System Requirements 1

2. High Level Design 1

2.1 Overview 1

3. Design 2

3.1 Class Design 2

3.1.1 UI Components 2

3.1.1.1 MainForm 2

3.1.1.2 HelpVerifications 2

3.1.2 AccessibleObject 2

3.1.3 Verifications 2

3.1.4 Exceptions 3

3.2 Execution Workflow 3

3.2.1 Workflow Diagram 3

4. Unit Tests 4

5. Potential Design Enhancements 4

5.1 Additional Control Verification 4

5.2 Minimum Accessibility Requirement Verification 4

5.3 How to really do screen reader testing 4

5.4 Implementing the Role-based strategy for MSAA 4

6. Change History 4

# Overview

MsaaVerify is a testing tool that verifies the MSAA properties and methods off of the IAccessible interface for commonly-used controls.

## Definitions

### MSAA

[Microsoft Active Accessibility](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msaa/msaastart_9w2t.asp) - Microsoft® Active Accessibility® 2.0 is a COM-based technology that improves the way accessibility aids, i.e. screen readers, Braille displays, screen magnifiers, work with applications running on Microsoft Windows® by providing a mechanism for exposing information about user interface elements.

## Goals

### Release 1.0

MsaaVerify currently provides the following features

* view a control’s window handle (hwnd) under the mouse pointer
* view properties of a captured accessible object
* test properties of a captured accessible object

### Future Releases

Future plans for MsaaVerify consist of

* implementing my [MSAA Role-Based Testing Strategy](http://blogs.msdn.com/saraford/archive/2005/06/06/425729.aspx)
* providing a series of automated tests for [how to really do screen reader testing](http://blogs.msdn.com/saraford/archive/2004/08/20/218078.aspx)

The ideal accessibility tool will tell the developer what must be done at the minimum cost to support baseline accessibility given the current layout.

For additional information, see [Section 5](#_Potential_Design_Enhancements) in this doc.

## Non-goals

I do not want MsaaVerify to morph into any other UI testing tool or UI design guideline. Since I believe that [UI Consistency and Accessibility are two sides of the same coin](http://blogs.msdn.com/saraford/archive/2004/02/10/70481.aspx), some best practices and overlapping UI testing functionality may appear in the tool. However, it is an explicit non-goal for such functionality to be designed into the tool.

## System Requirements

* .NET Framework 2.0
* Full support for MSAA v2.0 is provided in Windows XP and Windows Server 2003. For all other Windows versions, view the [supported platforms guide on MSDN](http://msdn.microsoft.com/library/en-us/msaa/gettingstarted_2do3.asp).

# High Level Design

## Overview

By default, MsaaVerify uses crosshairs, similar to applications like Spy++ and AccExplorer, to instantiate an accessible object found under the mouse pointer’s current position. An additional option is provided to capture the accessible object based on the user-entered control window handle.

Only one accessible object can be captured at a time. If a child element is captured, e.g. a listbox item, the parent object will be the main object verified. Very lightweight verification will be performed on the child elements, as most accessibility bugs are found at the parent level for common controls. For custom controls, there is a high risk of accessibility bugs at both parent and child levels; however, implementing verification techniques for custom controls is out of scope for the 1.1 release.

All accessibility verifications are based on the accessible object role type. For example, an accessible object of role “PushButton” will run through different test suite than an accessible object of role “Editable Text.”

Verification results are displayed in the UI along with additional information describing why the error occurred. My hope is to update the results UI for the 1.1 release to make it more presentable.

# Design

## Class Design

*MsaaVerify* is the only namespace used throughout the project.

### UI Components

#### MainForm

The MainForm class represents the main winform for MsaaVerify. The following methods are represented in this class

* Handling the Spy-like crosshairs
* Drawing the box around the captured object
* Filtering out the desired accessible object from the captured object
* Updating the UI with the desired accessible object
* Handling all controls on the main winform

#### HelpVerifications

The HelpVerifications class is a simple modal dialog to show the corresponding MSDN topic for the selected control or MSAA property. This dialog is in a desperate need for a redesign.

### AccessibleObject

The AccessibleObject class serves two primary functions:

* Wrapper class for the captured control’s IAccessible implementation, providing all the necessary method and properties to obtain MSAA information
* Any Window’s user32.dll calls needed to obtain additional information on the control, for example, retrieving the control’s caption

### Verifications

This class performs the actual verifications against the AccessibleObject object. This class is broken down into two types of classes:

* BaseVerifications – contains the baseline verifications that are performed among all the various MSAA properties
* <Control>Verifications – derives from BasVerifications, this class contains the verifications specific to that control. For example, a text box control requires a MSAA Value, but a push button does not.

### Exceptions

This class contains exceptions unique to MsaaVerify.

## Execution Workflow

There are two ways to capture an accessible object using MsaaVerify. Given that there are no issues with MSAA (note MsaaVerify does not and will not test for this), both functions will return the same AccessibleObject.

AccessibleObjectFromPoint

The default setting is to use the mouse pointer to capture the accessible object that resides beneath it. When the mouse click is released, the function AccessibleObjectFromPoint is called passing in the mouse pointer’s current (x,y) coordinates.

AccessibleObjectFromWindow

There is an optional setting to enter a control’s window handle to call the function AccessibleObjectFromWindow.

### Workflow Diagram



# Unit Tests

Yeah, these would be a good thing one day.

# Potential Design Enhancements

## Additional Control Verification

The first work item for MsaaVerify is to implement support for commonly-used controls, like static text, list boxes, tree views, and so forth.

## Minimum Accessibility Requirement Verification

There’s a lot more to supporting accessibility than just implementing MSAA properly. For example, there’s the use of High Contrast, Tab Order, Font Size, and so forth. There needs to be a single “capture the window and verify all priority 1 accessibility requirements” verification method.

## How to really do screen reader testing

The ideal MsaaVerify tool will provide a series of automated test for [how to really do screen reader testing](http://blogs.msdn.com/saraford/archive/2004/08/20/218078.aspx).

## Implementing the Role-based strategy for MSAA

Eventually, I’d like to implement my [MSAA Role-Based Testing Strategy](http://blogs.msdn.com/saraford/archive/2005/06/06/425729.aspx), where any object on any control will have an array of test suites, from a baseline series of tests to a full and complete verification based on the [MSAA docs](http://msdn.microsoft.com/library/en-us/msaa/msaapndx_2a05.asp).

# Change History

|  |  |  |
| --- | --- | --- |
| Change | Changed By | Date |
| Created | SaraF | 8 January 2007 |