**[MS-CUSTOMUI]:**

**Custom UI XML Markup Specification**

Intellectual Property Rights Notice for Open Specifications Documentation

* **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
* **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights**. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

**Preliminary Documentation.** This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final version of this technology, there are risks in relying on this preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 1/15/2009 | 1.0 | Major | Initial Availability |
| 7/13/2009 | 1.01 | Major | Revised and edited the technical content |
| 8/28/2009 | 1.02 | Editorial | Revised and edited the technical content |
| 11/6/2009 | 1.03 | Editorial | Revised and edited the technical content |
| 2/19/2010 | 2.0 | Editorial | Revised and edited the technical content |
| 3/31/2010 | 2.01 | Editorial | Revised and edited the technical content |
| 4/30/2010 | 2.02 | Editorial | Revised and edited the technical content |
| 6/7/2010 | 2.03 | Editorial | Revised and edited the technical content |
| 6/29/2010 | 2.04 | Editorial | Changed language and formatting in the technical content. |
| 7/23/2010 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 9/27/2010 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/15/2010 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/17/2010 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/18/2011 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/10/2011 | 2.04 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/20/2012 | 2.5 | Minor | Clarified the meaning of the technical content. |
| 4/11/2012 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/16/2012 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2012 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/11/2013 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/30/2013 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/18/2013 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/10/2014 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 4/30/2014 | 2.5 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/31/2014 | 2.6 | Minor | Clarified the meaning of the technical content. |
| 10/30/2014 | 3.0 | Major | Significantly changed the technical content. |
| 3/16/2015 | 4.0 | Major | Significantly changed the technical content. |
| 9/4/2015 | 4.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/15/2016 | 4.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 9/14/2016 | 4.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/17/2016 | 4.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/10/2017 | 5.0 | Major | Significantly changed the technical content. |
| 9/19/2017 | 6.0 | Major | Significantly changed the technical content. |
| 4/27/2018 | 7.0 | Major | Significantly changed the technical content. |

Table of Contents

[1 Introduction 7](#_Toc512415400)

[1.1 Glossary 7](#_Toc512415401)

[1.2 References 7](#_Toc512415402)

[1.2.1 Normative References 8](#_Toc512415403)

[1.2.2 Informative References 8](#_Toc512415404)

[2 Custom UI 9](#_Toc512415405)

[2.1 Parts 9](#_Toc512415406)

[2.1.1 Quick Access Toolbar Customizations Part 9](#_Toc512415407)

[2.1.2 Ribbon Extensibility Part 10](#_Toc512415408)

[2.2 Elements 11](#_Toc512415409)

[2.2.1 box (Box Grouping Container) 11](#_Toc512415410)

[2.2.2 button (Button) 15](#_Toc512415411)

[2.2.3 button (Unsized Button) 25](#_Toc512415412)

[2.2.4 button (Button Inside of a Split Button) 34](#_Toc512415413)

[2.2.5 buttonGroup (Button Grouping Container) 43](#_Toc512415414)

[2.2.6 checkBox (Check Box) 46](#_Toc512415415)

[2.2.7 comboBox (Combo Box) 56](#_Toc512415416)

[2.2.8 command (Repurposed Command) 67](#_Toc512415417)

[2.2.9 commands (List of Repurposed Commands) 69](#_Toc512415418)

[2.2.10 contextualTabs (List of Contextual Tab Sets) 69](#_Toc512415419)

[2.2.11 control (Unsized Control Clone) 70](#_Toc512415420)

[2.2.12 control (Control Clone) 78](#_Toc512415421)

[2.2.13 control (Quick Access Toolbar Control Clone) 87](#_Toc512415422)

[2.2.14 customUI (Custom UI Document Root) 96](#_Toc512415423)

[2.2.15 dialogBoxLauncher (Dialog Box Launcher) 97](#_Toc512415424)

[2.2.16 documentControls (List of Document-Specific Quick Access Toolbar Controls) 98](#_Toc512415425)

[2.2.17 dropDown (Drop-down Control) 99](#_Toc512415426)

[2.2.18 dynamicMenu (Unsized Dynamic Menu) 111](#_Toc512415427)

[2.2.19 dynamicMenu (Dynamic Menu) 120](#_Toc512415428)

[2.2.20 editBox (Edit Box) 131](#_Toc512415429)

[2.2.21 gallery (Gallery) 140](#_Toc512415430)

[2.2.22 gallery (Unsized Gallery) 156](#_Toc512415431)

[2.2.23 group (Group) 170](#_Toc512415432)

[2.2.24 item (Selection Item) 178](#_Toc512415433)

[2.2.25 labelControl (Text Label) 181](#_Toc512415434)

[2.2.26 menu (Unsized Menu) 189](#_Toc512415435)

[2.2.27 menu (Menu with Title) 198](#_Toc512415436)

[2.2.28 menu (Menu) 208](#_Toc512415437)

[2.2.29 menu (Dynamic Menu Root XML Element) 219](#_Toc512415438)

[2.2.30 menuSeparator (Menu Separator) 221](#_Toc512415439)

[2.2.31 officeMenu (Office Menu) 224](#_Toc512415440)

[2.2.32 qat (Quick Access Toolbar) 225](#_Toc512415441)

[2.2.33 ribbon (Ribbon) 226](#_Toc512415442)

[2.2.34 separator (Separator) 227](#_Toc512415443)

[2.2.35 sharedControls (List of Shared Quick Access Toolbar Controls) 230](#_Toc512415444)

[2.2.36 splitButton (Unsized Split Button) 231](#_Toc512415445)

[2.2.37 splitButton (Split Button with Title) 239](#_Toc512415446)

[2.2.38 splitButton (Split Button) 248](#_Toc512415447)

[2.2.39 tab (Tab) 257](#_Toc512415448)

[2.2.40 tabs (List of Tabs) 261](#_Toc512415449)

[2.2.41 tabSet (Contextual Tab Set) 262](#_Toc512415450)

[2.2.42 toggleButton (Unsized Toggle Button) 263](#_Toc512415451)

[2.2.43 toggleButton (Toggle Button) 273](#_Toc512415452)

[2.2.44 toggleButton (Toggle Button Inside of a Split Button) 283](#_Toc512415453)

[2.3 Simple Types 292](#_Toc512415454)

[2.3.1 ST\_BoxStyle (Box Style) 292](#_Toc512415455)

[2.3.2 ST\_Delegate (Callback Function Name) 292](#_Toc512415456)

[2.3.3 ST\_GalleryItemWidthHeight (Gallery Item Width or Height) 295](#_Toc512415457)

[2.3.4 ST\_GalleryRowColumnCount (Gallery Row or Column Count) 295](#_Toc512415458)

[2.3.5 ST\_ID (Control ID) 296](#_Toc512415459)

[2.3.6 ST\_ItemSize (Menu Item Size) 297](#_Toc512415460)

[2.3.7 ST\_Keytip (Keytip) 297](#_Toc512415461)

[2.3.8 ST\_LongString (Long String) 298](#_Toc512415462)

[2.3.9 ST\_QID (Qualified Control ID) 298](#_Toc512415463)

[2.3.10 ST\_Size (Control Size) 300](#_Toc512415464)

[2.3.11 ST\_String (Short String) 301](#_Toc512415465)

[2.3.12 ST\_StringLength (String Length) 302](#_Toc512415466)

[2.3.13 ST\_UniqueID (Custom Control ID) 302](#_Toc512415467)

[2.3.14 ST\_Uri (Image Relationship ID) 303](#_Toc512415468)

[3 Appendix A: Custom UI Control ID Tables 304](#_Toc512415469)

[3.1 idMso Tables 304](#_Toc512415470)

[3.1.1 Word 2007 304](#_Toc512415471)

[3.1.2 Excel 2007 358](#_Toc512415472)

[3.1.3 PowerPoint 2007 393](#_Toc512415473)

[3.2 imageMso Table 419](#_Toc512415474)

[4 Appendix B: Product Behavior 522](#_Toc512415475)

[5 Change Tracking 523](#_Toc512415476)

[6 Index 524](#_Toc512415477)

# Introduction

In creating an interoperable implementation, it is helpful to understand specific implementation choices made by other products implementing the same standard. For example, portions of the standard may provide only general guidance, leaving specific implementation choices up to the application implementer; in some circumstances it may be helpful for other implementers to understand those choices.

The information contained in this document provides information about how to implement UI customization in the context of ECMA-376 Office Open XML File Formats, as described in [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054).

## Glossary

This document uses the following terms:

**add-in**: Supplemental functionality that is provided by an external application or macro to extend the capabilities of an application.

**KeyTip**: A small, pop-up window that appears over commands on the ribbon when users press the ALT key. By pressing the key that is displayed in a KeyTip, users can execute the command that is associated with the KeyTip.

**macro**: A set of instructions that are recorded or written, and then typically saved to a file. When a macro is run, all of the instructions are performed automatically.

**XML fragment**: Lines of text that adhere to XML tag rules, as described in [[XML]](https://go.microsoft.com/fwlink/?LinkId=90598), but do not have a Document Type Definition (DTD) or schema, processing instructions, or any other header information.

**XML namespace**: A collection of names that is used to identify elements, types, and attributes in XML documents identified in a URI reference [[RFC3986]](https://go.microsoft.com/fwlink/?LinkId=90453). A combination of XML namespace and local name allows XML documents to use elements, types, and attributes that have the same names but come from different sources. For more information, see [[XMLNS-2ED]](https://go.microsoft.com/fwlink/?LinkId=90602).

**XML namespace prefix**: An abbreviated form of an [**XML namespace**](#gt_485f05b3-df3b-45ac-b8bf-d05f5d185a24), as described in [XML].

**XML schema**: A description of a type of XML document that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by XML itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

**XML schema definition (XSD)**: The World Wide Web Consortium (W3C) standard language that is used in defining XML schemas. Schemas are useful for enforcing structure and constraining the types of data that can be used validly within other XML documents. XML schema definition refers to the fully specified and currently recommended standard for use in authoring [**XML schemas**](#gt_bd0ce6f9-c350-4900-827e-951265294067).

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](https://go.microsoft.com/fwlink/?linkid=850906).

### Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[ECMA-376] ECMA International, "Office Open XML File Formats", 1st Edition, ECMA-376, December 2006, [http://www.ecma-international.org/publications/standards/Ecma-376.htm](https://go.microsoft.com/fwlink/?LinkId=200054)

[MS-CUSTOMUI2] Microsoft Corporation, "[Custom UI XML Markup Version 2 Specification](%5bMS-CUSTOMUI2%5d.pdf#Section_bad56c217b1541bcaf328b5afe6e922e)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, [http://www.rfc-editor.org/rfc/rfc2119.txt](https://go.microsoft.com/fwlink/?LinkId=90317)

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, [http://www.w3.org/TR/2009/REC-xml-names-20091208/](https://go.microsoft.com/fwlink/?LinkId=191840)

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, [http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/](https://go.microsoft.com/fwlink/?LinkId=90608)

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, [http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/](https://go.microsoft.com/fwlink/?LinkId=90610)

### Informative References

None.

# Custom UI

The subordinate clauses specify the semantics for the Custom UI XML markup contained within the ECMA-376 Office Open XML File Formats, as specified in [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054). These semantics describe customization of the UI interface. Examples in the following clauses highlight customizations in the context of the Microsoft Office Fluent interface (UI) but the concepts extend naturally to any user interface.

Customization of the UI is accomplished via the addition of parts containing Custom UI XML markup to the containing document package.

## Parts

The parts described in the subordinate sections detail the additional part types utilized by CustomUI in an ECMA-376 Office Open XML File Formats [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054) file.

### Quick Access Toolbar Customizations Part

| Content Type: | application/xml |
| --- | --- |
| Root Namespace: | http://schemas.microsoft.com/office/2006/01/customui |
| Source Relationship: | http://schemas.microsoft.com/office/2006/relationships/ui/userCustomization |

The syntax of the structures contained in this part uses [**XML schema definition (XSD)**](#gt_c7e91c99-e45a-44c2-a08a-c34f137a2cae), as specified in [[XMLSCHEMA1]](https://go.microsoft.com/fwlink/?LinkId=90608) and [[XMLSCHEMA2]](https://go.microsoft.com/fwlink/?LinkId=90610).

This specification defines and references various [**XML namespaces**](#gt_485f05b3-df3b-45ac-b8bf-d05f5d185a24) by using the mechanisms specified in [[XMLNS]](https://go.microsoft.com/fwlink/?LinkId=191840).

An instance of this part type contains information about the quick access toolbar customizations specific to the containing package.

For example, a user can customize the quick access toolbar for his WordProcessingML document to contain the UI controls that they commonly use.

A package is permitted to contain at most one Quick Access Toolbar Customizations part, and that part is the target of a relationship in the package-relationship item for the document.

For example, the following package part-relationship item contains a relationship to a Quick Access Toolbar Customizations part, which is stored in the ZIP item /userCustomization/customUI.xml:

1. <Relationships xmlns="…">
2. <Relationship Id="rId2"
3. Type="http://…/2006/relationships/ui/userCustomization"
4. Target="/userCustomization/customUI.xml" />
5. </Relationships>

The root element for a part of this content type is **customUI**.

For example, the following Quick Access Toolbar Customizations content markup specifies that the control with identifier "SpellingAndGrammar" is to be added to the quick access toolbar for the package:

1. <mso:customUI xmlns:mso="http://schemas.microsoft.com/office/2006/01/customui">
2. <mso:ribbon>
3. <mso:qat>
4. <mso:documentControls>
5. <mso:control idQ="mso:SpellingAndGrammar" visible="true" />
6. </mso:documentControls>
7. </mso:qat>
8. </mso:ribbon>
9. </mso:customUI>

A Quick Access Toolbar Customizations part is located within the package containing the source relationship. Expressed syntactically, the **TargetMode** attribute of the **Relationship** element is "Internal".

A Quick Access Toolbar Customizations part does not have implicit or explicit relationships to any other part defined by ECMA-376 Office Open XML File Formats, as specified in [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054).

### Ribbon Extensibility Part

| Content Type: | application/xml |
| --- | --- |
| Root Namespace: | http://schemas.microsoft.com/office/2006/01/customui |
| Source Relationship: | http://schemas.microsoft.com/office/2006/relationships/ui/extensibility |

The syntax of the structures contained in this part uses [**XML schema definition (XSD)**](#gt_c7e91c99-e45a-44c2-a08a-c34f137a2cae), as specified in [[XMLSCHEMA1]](https://go.microsoft.com/fwlink/?LinkId=90608) and [[XMLSCHEMA2]](https://go.microsoft.com/fwlink/?LinkId=90610).

This specification defines and references various [**XML namespaces**](#gt_485f05b3-df3b-45ac-b8bf-d05f5d185a24) by using the mechanisms specified in [[XMLNS]](https://go.microsoft.com/fwlink/?LinkId=191840).

An instance of this part type contains information about the ribbon customizations specific to the containing package.

For example, a SpreadsheetML document that represents a timecard could contain custom UI controls to guide the user in filling out the timecard.

A package is permitted to contain at most one Ribbon Extensibility part, and that part is the target of a relationship in the package-relationship item for the document.

For example, the following package part-relationship item contains a relationship to a Ribbon Extensibility part, which is stored in the ZIP item /customUI/customUI.xml:

1. <Relationships xmlns="…">
2. <Relationship Id="rId5"
3. Type="http://…/2006/relationships/ui/extensibility"
4. Target="/customUI/customUI.xml" />
5. </Relationships>

The root element for a part of this content type is **customUI**.

For example, the following Ribbon Extensibility content markup specifies that the ribbon tab with identifier "TabHome" is to be hidden for the containing package:

1. <customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
2. <ribbon>
3. <tabs>
4. <tab idMso="TabHome" visible="false" />
5. </tabs>
6. </ribbon>
7. </customUI>

A Ribbon Extensibility part is located within the package containing the source relationship. Expressed syntactically, the **TargetMode** attribute of the **Relationship** element is "Internal".

A Ribbon Extensibility part is permitted to have explicit relationships to the following parts defined by ECMA-376 Office Open XML File Formats, as specified in [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054):

* Image Part, as specified in [ECMA-376] Part 1 section15.2.13.

## Elements

A Custom UI document contains customizations of an application's UI. Customizations are mainly of two types:

* Modifications of the application's built-in UI, such as hiding or disabling built-in UI controls or repurposing command actions.
* Creation of custom UI controls, such as a custom ribbon tab, menu item, or quick access toolbar button.

For example, consider the following Custom UI document:

1. <customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
2. <commands>
3. <command idMso="Bold" enabled="false" />
4. </commands>
5. <ribbon>
6. <tabs>
7. <tab idMso="TabHome" visible="false" />
8. <tab id="CustomTab" label="Custom Tab">
9. <group id="CustomGroup" label="Custom Group">
10. <button id="CustomButton" label="Custom Button"
11. size="large" imageMso="HappyFace" onAction="OnButtonClick" />
12. </group>
13. </tab>
14. </tabs >
15. </ribbon>
16. </customUI>

This example disables the command with an identifier of "Bold", hides the ribbon tab with an identifier of "TabHome", and creates a new custom ribbon tab with a custom button in it.

The full XML Schema Definition of the XML Schema fragments listed in this section is defined in Appendix A of [[MS-CUSTOMUI2]](%5bMS-CUSTOMUI2%5d.pdf#Section_bad56c217b1541bcaf328b5afe6e922e).

### box (Box Grouping Container)

This element specifies a grouping container control that can be used to align controls vertically or horizontally. **Box** elements can be nested to create complex UI layouts.

For example, consider a group of controls that are laid out horizontally, as follows:

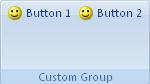


Figure 1: Controls grouped horizontally

This layout is specified using the following [**XML fragment**](#gt_775c2c6d-1ce7-49c7-838b-5fa0dcb7ee4d):

1. <box id="box" boxStyle="horizontal">
2. <button id="button1" label="Button 1" imageMso="HappyFace" />
3. <button id="button2" label="Button 2" imageMso="HappyFace" />
4. </box>

This is contrasted to the default vertical layout that is used if the **box** element is not specified.

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **box** | 2.2.1 |
| **group** | [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **box** (Box Grouping Container) | 2.2.1 |
| **button** (Button) | [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b) |
| **buttonGroup** (Button Grouping Container) | [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **comboBox** (Combo Box) | [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e) |
| **control** (Control Clone) | [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755) |
| **dropDown** (Drop-down Control) | [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69) |
| **dynamicMenu** (Dynamic Menu) | [2.2.19](#Section_26d7db807ea442d09a063f63365043c7) |
| **editBox** (Edit Box) | [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34) |
| **gallery** (Gallery) | [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f) |
| **labelControl** (Text Label) | [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb) |
| **menu** (Menu) | [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf) |
| **splitButton** (Split Button) | [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8) |
| **toggleButton** (Toggle Button) | [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **boxStyle** (box style) | Specifies the layout direction for the child controls inside of the **box** element.  If this attribute is omitted, the child controls SHOULD be laid out horizontally.  For example, consider a group of controls to be laid out vertically. This is specified using the following XML:   1. <box id="box" boxStyle="vertical"> 2. … 3. </box>   The possible values for this attribute are defined by the **ST\_BoxStyle** simple type, as specified in section [2.3.1](#Section_3a299df5af064f4f813db0043d772546). |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function that is called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes is to be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or **containers** created by other Custom UI documents.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes is to be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an [**XML namespace prefix**](#gt_b9a20be7-31d9-4dcd-9cb9-ba72302857a2) for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an **id** of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is to be hidden.  The possible values for this attribute are defined by the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Box">
2. <xsd:group ref="EG\_Controls" minOccurs="0" maxOccurs="1000"/>
3. <xsd:attributeGroup ref="AG\_IDCustom"/>
4. <xsd:attributeGroup ref="AG\_Visible"/>
5. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
6. <xsd:attribute name="boxStyle" type="ST\_BoxStyle" use="optional"/>
7. </xsd:complexType>

### button (Button)

This element specifies a standard push-button control that performs an action when clicked.

For example, consider a button control, as follows:



Figure 2: A button control

This is specified using the following XML fragment:

1. <button id="button" label="Button" imageMso="HappyFace" />

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **box** | [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e) |
| **group** | [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with detailed description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is always disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function that is called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function that is called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is to be called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is to be called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function that is called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is to be called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is to be called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is to be called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application is to display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is to be called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application is to display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is to be called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is to be called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is to be called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is to be called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control is passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a tab in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image that is to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is to be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image that is to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** that is to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom button with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a button that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip text  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** that is to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Control with a supertip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a button with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an **id** of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Button">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ButtonRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### button (Unsized Button)

This element specifies a push-button that, because of its location, cannot have its size changed. The **size** attribute is not present. This element otherwise behaves like the regular **button** element, as specified in section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b).

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **buttonGroup** | [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21) |
| **dialogBoxLauncher** | [2.2.15](#Section_76d9f8786b2749ff8c4cee008f345cac) |
| **documentControls** | [2.2.16](#Section_de33fcbe3b5b4d30b875d2ec922cf32f) |
| **dropDown** | [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69) |
| **gallery** | [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f) |
| **gallery** | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** | [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf) |
| **menu** | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **menu** | [2.2.29](#Section_e662c00135584511a56507a3e32021e3) |
| **menu** | [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace) |
| **officeMenu** | [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844) |
| **sharedControls** | [2.2.35](#Section_0ca32ee5aae74ff18baa8a8c9be2ebb3) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is always disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is to be called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is to be called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function that is called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is to be called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is to be called when the application needs to determine the label of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is to be called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application is to display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is to be called when the application needs to determine whether to display the icon of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application is to display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is to be called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is to be called when the application needs to determine the supertip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is to be called when the application needs to determine the visibility of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, "ex" is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that **tab** cannot be found, it is created. A new group belonging to this file is added to the **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is to be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom **button** that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (Keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (Label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a **button** that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the **label** of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip text  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a **button** that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (Show Label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a label, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **boolean** datatype. |
| **supertip** (Supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with supe rtip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (Tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ButtonRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_Control">
4. <xsd:attributeGroup ref="AG\_Action"/>
5. <xsd:attributeGroup ref="AG\_Enabled"/>
6. <xsd:attributeGroup ref="AG\_Description"/>
7. <xsd:attributeGroup ref="AG\_Image"/>
8. </xsd:extension>
9. </xsd:complexContent>
10. </xsd:complexType>

### button (Button Inside of a Split Button)

This element specifies a push-button that is a child of a split button control. The **visible** and **getVisible** attributes are not present because the visibility is controlled by the split button. This element otherwise behaves in the same way as the unsized **button** element, as specified in section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd).

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **splitButton** | [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8) |
| **splitButton** | [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d) |
| **splitButton** | [2.2.37](#Section_1c6721206402408b96dde98650c12da5) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a **button** with a detailed **description**, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new **button** that is always disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no **label** SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the **label** of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function that is called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function that is called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function that is called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function that is called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function that is called to determine the visibility state of this control. This attribute is prohibited.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image that is used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image that is used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an **id** of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of x:OtherTab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a string to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a **button** that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip text  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control. This attribute is prohibited.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_VisibleButton">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_ButtonRegular">
4. <xsd:attribute name="visible" use="prohibited"/>
5. <xsd:attribute name="getVisible" use="prohibited"/>
6. </xsd:restriction>
7. </xsd:complexContent>
8. </xsd:complexType>

### buttonGroup (Button Grouping Container)

This element specifies a grouping container that groups controls together visually. The child controls are laid out horizontally.

For example, consider a group of buttons, as follows:



Figure 3: A group of buttons

This is specified using the following XML fragment:

1. <buttonGroup id="buttonGroup">
2. <button id="button1" imageMso="Bold" />
3. <button id="button2" imageMso="Italic" />
4. <button id="button3" imageMso="Underline" />
5. </buttonGroup>

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **box** | [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e) |
| **group** | [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Unsized Menu) | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **splitButton** (Unsized Split Button) | [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d) |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new **group** belonging to this file is added to the **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an **id** of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ButtonGroup">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="1000">
4. <xsd:element name="control" type="CT\_ControlCloneRegular"/>
5. <xsd:element name="button" type="CT\_ButtonRegular"/>
6. <xsd:element name="toggleButton" type="CT\_ToggleButtonRegular"/>
7. <xsd:element name="gallery" type="CT\_GalleryRegular"/>
8. <xsd:element name="menu" type="CT\_MenuRegular"/>
9. <xsd:element name="dynamicMenu" type="CT\_DynamicMenuRegular"/>
10. <xsd:element name="splitButton" type="CT\_SplitButtonRegular"/>
11. </xsd:choice>
12. </xsd:sequence>
13. <xsd:attributeGroup ref="AG\_IDCustom"/>
14. <xsd:attributeGroup ref="AG\_Visible"/>
15. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
16. </xsd:complexType>

### checkBox (Check Box)

This element specifies a standard checkbox control.

For example, consider a checkbox control, as follows:



Figure 4: A checkbox control

This is specified using the following XML fragment:

1. <checkBox id="checkBox" label="CheckBox" />

The following table summarizes the elements that are parents of this element.

| Parent Elements | Section |
| --- | --- |
| **box** | [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e) |
| **group** | [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4) |
| **menu** | [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf) |
| **menu** | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **menu** | [2.2.29](#Section_e662c00135584511a56507a3e32021e3) |
| **menu** | [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace) |
| **officeMenu** | [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is always disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getPressed** (**getPressed** callback) | Specifies the name of a callback function to be called to determine the toggled state of this control.  If this attribute is omitted, the control SHOULD default to the off state.  For example, consider the following XML fragment:   1. <toggleButton id="toggle" getPressed="IsButtonToggled" />   In this example, the **IsButtonToggled** callback function is called when the application needs to determine the toggle state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application displays the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new **group** belonging to this file is added to the **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is to be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom **button** to use the built-in image with an id of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an id of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Tool Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a button that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an **id** of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_CheckBox">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_ToggleButtonRegular">
4. <xsd:attribute name="image" use="prohibited"/>
5. <xsd:attribute name="imageMso" use="prohibited"/>
6. <xsd:attribute name="getImage" use="prohibited"/>
7. <xsd:attribute name="showImage" use="prohibited"/>
8. <xsd:attribute name="getShowImage" use="prohibited"/>
9. <xsd:attribute name="showLabel" use="prohibited"/>
10. <xsd:attribute name="getShowLabel" use="prohibited"/>
11. </xsd:restriction>
12. </xsd:complexContent>
13. </xsd:complexType>

### comboBox (Combo Box)

This element specifies a standard combo box control that allows a user to input a text **string** or select one from a list.

For example, consider a combo box control, as follows:

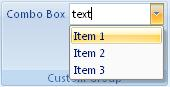


Figure 5: A combo box control

This is specified using the following XML fragment:

1. <comboBox id="comboBox" label="Combo Box">
2. <item id="item1" label="Item 1" />
3. <item id="item2" label="Item 2" />
4. <item id="item3" label="Item 3" />
5. </comboBox>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **item** (Selection Item) | [2.2.24](#Section_6417edcc85b847f781fd56918e8df537) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new **button** that is disabled. A permanently disabled **button** is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemCount** (**getItemCount** callback) | Specifies the name of a callback function to be called to determine the number of selection items in this control.  If this attribute is omitted, the control SHOULD display any selection items that are specified as child elements. If no such items are specified, the control SHOULD be empty.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" />   In this example, the **GetGalleryItemCount** callback function is called when the application needs to determine the number of items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemID** (**getItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD have empty identifiers.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetGalleryItemID" />   In this example, the **GetGalleryItemID** callback function is called when the application needs to determine the identifier of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemImage** (**getItemImage** callback) | Specifies the name of a callback function to be called to determine the icon of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display icons.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemImage="GetGalleryItemImage" />   In this example, the **GetGalleryItemImage** callback function is called when the application needs to determine the icon of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemLabel** (**getItemLabel** callback) | Specifies the name of a callback function to be called to determine the label of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display labels.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemLabel="GetGalleryItemLabel" />   In this example, the **GetGalleryItemLabel** callback function is called when the application needs to determine the label of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemScreentip** (**getItemScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD use their labels as their screentips, or display no screentips at all.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemScreentip="GetGalleryItemScreentip" />   In this example, the **GetGalleryItemScreentip** callback function is called when the application needs to determine the screentip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemSupertip** (**getItemSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display supertips.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemSupertip="GetGalleryItemSupertip" />   In this example, the **GetGalleryItemSupertip** callback function is called when the application needs to determine the supertip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getText** (**getText** callback) | Specifies the name of a callback function to be called to determine the text that is displayed in the control.  For example, consider the following XML fragment:   1. <editBox id="editBox" getText="GetEditBoxText" />   In this example, the **GetEditBoxText** callback function is called when the application needs to determine the text to display in the control.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an id of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **invalidateContentOnDrop** (invalidate content on drop) | Specifies whether this control invalidates its contents and re-queries for them when the user opens its drop-down menu.  If this attribute is omitted, its value SHOULD default to "false".  For example, consider the following XML fragment:   1. <comboBox id="comboBox" getItemCount="GetComboBoxItemCount" 2. getItemLabel="GetComboBoxItemLabel" 3. invalidateContentOnDrop="true" />   In this example, this combo box clears out its items and re-calls the **GetComboBoxItemCount** and **GetComboBoxItemLabel** callback functions to populate its contents each time the user opens it.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with a Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **maxLength** (maximum input **string** length) | Specifies an **integer** to be used as the maximum length of a **string** that can be entered into the control.  If the **maxLength** attribute is omitted, the length of the input **string** SHOULD NOT be limited, except by application-specific constraints.  For example, consider the following XML fragment:   1. <editBox id="editBox" maxLength="10" />   This specifies an edit box control that can only accept **strings** up to 10 characters in length.  The possible values for this attribute are defined by the **ST\_StringLength** simple type, as specified in section [2.3.12](#Section_4a327c6229ec496e935fa1aa77d26d5e). |
| **onChange** (**onChange** callback) | Specifies the name of a callback function to be called when the text in the control has been changed by the user.  For example, consider the following XML fragment:   1. <editBox id="editBox" onChange="EditBoxTextChanged" />   This specifies an edit box control that calls the **EditBoxTextChanged** callback function when the user inputs a text **string**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with a screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemImage** (show item image) | Specifies whether this control displays icons on its selection items.  If this attribute is omitted, the items' icons SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemImage="false" > 2. <item id="item1" label="Item 1" /> 3. <item id="item2" label="Item 1" /> 4. <item id="item3" label="Item 2" /> 5. <item id="item4" label="Item 3" /> 6. </gallery>   This specifies a gallery control that does not show any icons on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **sizeString** (size string) | Specifies a **string** whose size is used to determine the width of the text input area of this control.  If this attribute is omitted, the application SHOULD determine the width of the text input area of the control automatically.  For example, consider the following XML fragment:   1. <editBox id="editBox" sizeString="WWWWWWWWWWWWW" />   This specifies an edit box control that SHOULD be wide enough to display the **string** "WWWWWWWWWWWWW".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with a super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ComboBox">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_EditBox">
4. <xsd:sequence>
5. <xsd:element name="item" type="CT\_Item" minOccurs="0" maxOccurs="1000"/>
6. </xsd:sequence>
7. <xsd:attributeGroup ref="AG\_DropDownAttributes"/>
8. <xsd:attributeGroup ref="AG\_DynamicContentAttributes"/>
9. </xsd:extension>
10. </xsd:complexContent>
11. </xsd:complexType>

### command (Repurposed Command)

This element specifies that a particular built-in command in the application is to be repurposed.

The **enabled** and **getEnabled** attributes can be specified to disable a command.

The **onAction** attribute allows the functionality of a command to be repurposed to run a callback function. Only commands that execute simple actions (for example, commands represented as button controls) can be repurposed using **onAction**.

For example, consider the following XML fragment:

1. <commands>
2. <command idMso="Bold" enabled="false" />
3. <command idMso="Paste" onAction="MyPasteFunction" />
4. </commands>

In this example, the **Bold** command is permanently disabled and that the callback function **MyPasteFunction** is called when the **Paste** command is invoked.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **commands** (section [2.2.9](#Section_f914543a2b8e48d4bf5e98b1326e7fc2)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a button that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Command" mixed="false">
2. <xsd:attributeGroup ref="AG\_Action"/>
3. <xsd:attributeGroup ref="AG\_Enabled"/>
4. <xsd:attributeGroup ref="AG\_IDMso"/>
5. </xsd:complexType>

### commands (List of Repurposed Commands)

This element specifies a list of repurposed commands. This element SHOULD NOT be specified if the containing Custom UI XML document is a Quick Access Toolbar Customizations part.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **customUI** (section [2.2.14](#Section_8a27e8523f8b424aac6732c58181e9d3)) |

The following table summarizes the child elements of this element.

| Child Elements | Subclause |
| --- | --- |
| **command** (Repurposed Command) | section [2.2.8](#Section_21316865fbce4a3fa9ffa8277cce5f2d) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Commands">
2. <xsd:sequence>
3. <xsd:element name="command" type="CT\_Command" minOccurs="1" maxOccurs="5000"/>
4. </xsd:sequence>
5. </xsd:complexType>

### contextualTabs (List of Contextual Tab Sets)

This element specifies a list of contextual tab sets. This element SHOULD NOT be specified if the containing Custom UI XML document is a Quick Access Toolbar Customizations part.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **ribbon** (section [2.2.33](#Section_574eeee87a03406ab95ff9e51e53dd9d)) |

The following table summarizes the child elements of this element.

| Child Elements | Subclause |
| --- | --- |
| **tabSet** (Contextual Tab Set) | section [2.2.41](#Section_90a00968474745128601b61440747033) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ContextualTabs">
2. <xsd:sequence>
3. <xsd:element name="tabSet" type="CT\_TabSet" minOccurs="1" maxOccurs="100"/>
4. </xsd:sequence>
5. </xsd:complexType>

### control (Unsized Control Clone)

This element specifies a clone of a control that, because of its location, cannot have its size changed. The **size** attribute is not present. The element otherwise behaves like the regular **control** element, as specified in section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ControlCloneRegular">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_Control">
4. <xsd:attribute name="id" use="prohibited"/>
5. </xsd:restriction>
6. </xsd:complexContent>
7. </xsd:complexType>

### control (Control Clone)

This element specifies a clone of an existing control. Built-in controls can be cloned using the **idMso** attribute. Custom controls cannot be cloned. Custom controls cannot be created using the **control** element.

When an existing control is cloned, its non-location-specific properties, such as the icon and label, are copied to the clone. Location-specific properties, such as the size and visibility of the control, are not copied. These properties can be set by specifying additional attributes on the **control** element.

For example, consider the following XML fragment:

1. <control idMso="Paste" size="large" />

This results in a large copy of the **Paste** control, as follows:



Figure 6: A Paste control

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose text  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application displays the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a button that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  A large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ControlClone">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_Button">
4. <xsd:attribute name="id" use="prohibited"/>
5. <xsd:attribute name="onAction" use="prohibited"/>
6. </xsd:restriction>
7. </xsd:complexContent>
8. </xsd:complexType>

### control (Quick Access Toolbar Control Clone)

This element specifies a clone of an existing control. It is specific to control clones on the quick access toolbar, but otherwise behaves the same way as the regular **control** element, as specified in section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **documentControls** (section [2.2.16](#Section_de33fcbe3b5b4d30b875d2ec922cf32f)); **sharedControls** (section [2.2.35](#Section_0ca32ee5aae74ff18baa8a8c9be2ebb3)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (custom control identifier) | Specifies the identifier for a custom control. All new custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control. The identifier is qualified with an XML namespace prefix that specifies the owner of the control. If the namespace is equal to the Custom UI namespace, the **idQ** attribute behaves in the same manner as the **idMso** attribute. If the namespace is equal to the name of the current file, the **idQ** attribute behaves like the **id** attribute. If the namespace is equal to the name of a different file, the attribute references a control from that file.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive.  For example, consider the following XML fragment:   1. <tab idQ="x:OtherTab"> 2. <group id="MyGroup" label="My Group"> 3. … 4. </group> 5. </tab>   In this case **x** is an XML namespace equal to the name of another file that has a Custom UI document with a **tab** with an identifier of "OtherTab". This example adds a custom **group** to that **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood. it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a string to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ControlCloneQat">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ControlBase">
4. <xsd:attribute name="id" type="ST\_ID" use="optional"/>
5. <xsd:attribute name="idQ" type="ST\_QID" use="optional"/>
6. <xsd:attributeGroup ref="AG\_IDMso"/>
7. <xsd:attributeGroup ref="AG\_Description"/>
8. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
9. </xsd:extension>
10. </xsd:complexContent>
11. </xsd:complexType>

### customUI (Custom UI Document Root)

This element specifies the root tag in a Custom UI XML document.

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **commands** (List of Repurposed Commands) | [2.2.9](#Section_f914543a2b8e48d4bf5e98b1326e7fc2) |
| **ribbon** (Ribbon) | [2.2.33](#Section_574eeee87a03406ab95ff9e51e53dd9d) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **loadImage** (**loadImage** callback) | Specifies the name of a callback function to be called when the application needs to load an image for a control's icon.  For example, consider the following XML fragment:   1. <customUI xmlns="…" loadImage="LoadImageFunction" />   In this example, the **LoadImageFunction** callback is called to load icon images.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **onLoad** (**onLoad** callback) | Specifies the name of a callback function to be called when the Custom UI file is loaded by the application.  For example, consider the following XML fragment:   1. <customUI xmlns="…" onLoad="OnCustomUILoaded" />   In this example, the **OnCustomUILoaded** callback function is called when the containing Custom UI file is loaded.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_CustomUI">
2. <xsd:sequence>
3. <xsd:element name="commands" type="CT\_Commands" minOccurs="0" maxOccurs="1"/>
4. <xsd:element name="ribbon" type="CT\_Ribbon" minOccurs="0" maxOccurs="1"/>
5. </xsd:sequence>
6. <xsd:attribute name="onLoad" type="ST\_Delegate" use="optional"/>
7. <xsd:attribute name="loadImage" type="ST\_Delegate" use="optional"/>
8. </xsd:complexType>

### dialogBoxLauncher (Dialog Box Launcher)

This element specifies a button that is the dialog box launcher control for a ribbon group.

For example, consider a dialog box launcher control, as follows:

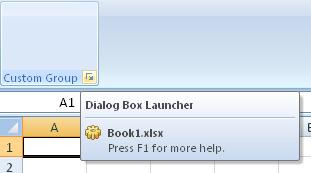


Figure 7: A dialog box launcher control

This is specified using the following XML fragment:

1. <group id="customGroup" label="Custom Group">
2. <dialogBoxLauncher>
3. <button id="button" screentip="Dialog Box Launcher" />
4. </dialogBoxLauncher>
5. </group>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_DialogLauncher">
2. <xsd:sequence>
3. <xsd:element name="button" type="CT\_ButtonRegular" minOccurs="1" maxOccurs="1"/>
4. </xsd:sequence>
5. </xsd:complexType>

### documentControls (List of Document-Specific Quick Access Toolbar Controls)

This element specifies the list of controls on the quick access toolbar which are specific to the containing file.

For example, consider a set of controls on the document-specific quick access toolbar, as follows:

A set of controls on the document-specific quick access toolbar

Figure 8: A set of controls on the document-specific quick access toolbar

This is specified using the following XML fragment:

1. <documentControls>
2. <control idMso="CalculateNow" />
3. <control idMso="HyperlinkInsert" />
4. </documentControls>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **qat** (section [2.2.32](#Section_93e2f741a06c40968dbf5a97fefca545)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **control** (Quick Access Toolbar Control Clone) | [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2) |
| **separator** (Separator) | [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_QatItems">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="1000">
4. <xsd:element name="control" type="CT\_ControlCloneQat"/>
5. <xsd:element name="button" type="CT\_ButtonRegular"/>
6. <xsd:element name="separator" type="CT\_Separator"/>
7. </xsd:choice>
8. </xsd:sequence>
9. </xsd:complexType>

### dropDown (Drop-down Control)

This element specifies a drop-down control that allows users to make a selection from a list of options. A drop-down control can optionally have buttons after its selection items.

For example, consider a drop-down control, as follows:

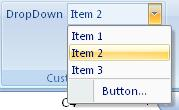


Figure 9: A drop-down control

This is specified using the following XML fragment:

1. <dropDown id="dropDown" label="DropDown">
2. <item id="item1" label="Item 1" />
3. <item id="item2" label="Item 2" />
4. <item id="item3" label="Item 3" />
5. <button id="button" label="Button..." />
6. </dropDown>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **item** (Selection Item) | [2.2.24](#Section_6417edcc85b847f781fd56918e8df537) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new **button** that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemCount** (**getItemCount** callback) | Specifies the name of a callback function to be called to determine the number of selection items in this control.  If this attribute is omitted, the control SHOULD display any selection items that are specified as child elements. If no such items are specified, the control SHOULD be empty.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" />   In this example, the **GetGalleryItemCount** callback function is called when the application needs to determine the number of items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemID** (**getItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD have empty identifiers.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetGalleryItemID" />   In this example, the **GetGalleryItemID** callback function is called when the application needs to determine the identifier of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemImage** (**getItemImage** callback) | Specifies the name of a callback function to be called to determine the icon of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display icons.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemImage="GetGalleryItemImage" />   In this example, the **GetGalleryItemImage** callback function is called when the application needs to determine the icon of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemLabel** (**getItemLabel** callback) | Specifies the name of a callback function to be called to determine the label of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display labels.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemLabel="GetGalleryItemLabel" />   In this example, the **GetGalleryItemLabel** callback function is called when the application needs to determine the label of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemScreentip** (**getItemScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD use their labels as their screentips, or display no screentips at all.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemScreentip="GetGalleryItemScreentip" />   In this example, the **GetGalleryItemScreentip** callback function is called when the application needs to determine the screentip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemSupertip** (**getItemSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display supertips.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemSupertip="GetGalleryItemSupertip" />   In this example, the **GetGalleryItemSupertip** callback function is called when the application needs to determine the supertip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemID** (**getSelectedItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of the item to be selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetItemID" 3. getSelectedItemID="GetGallerySelectedItemID" />   In this example, the **GetGallerySelectedItemID** callback function is called when the application needs to determine the selected item in the gallery. In this example the callback function returns one of the identifiers returned by the **GetItemID** callback function.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemIndex** (**getSelectedItemIndex** callback) | Specifies the name of a callback function to be called to determine the index of the item to be selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getSelectedItemIndex="GetGallerySelectedItemIndex" />   In this example, the **GetGallerySelectedItemIndex** callback function is called when the application needs to determine the selected item in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application displays the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application displays the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ,** and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that **tab** cannot be found, it is created. A new **group** belonging to this file is added to the **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom **button** whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom **tab** with an identifier of "MyTab" is to be inserted after the custom **tab** with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom **tab** with an identifier of "MyTab" is to be inserted before the built-in **tab** with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom **tab** with an identifier of "MyTab" is to be inserted before the custom **tab** with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a string to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a button that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemImage** (show item image) | Specifies whether this control displays icons on its selection items.  If this attribute is omitted, the items' icons SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemImage="false" > 2. <item id="item1" label="Item 1" /> 3. <item id="item2" label="Item 1" /> 4. <item id="item3" label="Item 2" /> 5. <item id="item4" label="Item 3" /> 6. </gallery>   This specifies a gallery control that does not show any icons on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemLabel** (show item label) | Specifies whether this control displays labels on its selection items.  If this attribute is omitted, the item's labels SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemLabel="false" > 2. <item id="item1" image="Forest" /> 3. <item id="item2" image="Desert" /> 4. <item id="item3" image="Mountain" /> 5. <item id="item4" image="Ocean" /> 6. </gallery>   This specifies a gallery control that does not show any labels on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **sizeString** (size string) | Specifies a **string** whose size is used to determine the width of the text input area of this control.  If this attribute is omitted, the application SHOULD determine the width of the text input area of the control automatically.  For example, consider the following XML fragment:   1. <editBox id="editBox" sizeString="WWWWWWWWWWWWW" />   This specifies an edit box control that is wide enough to display the **string** "WWWWWWWWWWWWW".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in **tab** with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_DropDownRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_Control">
4. <xsd:sequence>
5. <xsd:element name="item" type="CT\_Item" minOccurs="0" maxOccurs="1000"/>
6. <xsd:element name="button" type="CT\_ButtonRegular" minOccurs="0" maxOccurs="16"/>
7. </xsd:sequence>
8. <xsd:attributeGroup ref="AG\_Action"/>
9. <xsd:attributeGroup ref="AG\_Enabled"/>
10. <xsd:attributeGroup ref="AG\_Image"/>
11. <xsd:attributeGroup ref="AG\_DropDownAttributes"/>
12. <xsd:attribute name="getSelectedItemID" type="ST\_Delegate" use="optional"/>
13. <xsd:attribute name="getSelectedItemIndex" type="ST\_Delegate" use="optional"/>
14. <xsd:attribute name="showItemLabel" type="xsd:boolean" use="optional"/>
15. </xsd:extension>
16. </xsd:complexContent>
17. </xsd:complexType>

### dynamicMenu (Unsized Dynamic Menu)

This element specifies a dynamic menu control that, because of its location, cannot have its anchor size changed. The **size** attribute is not present. It otherwise behaves identically to the regular **dynamicMenu** element, as specified in section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getContent** (**getContent** callback) | Specifies the name of a callback function to be called when the application needs to determine the contents of the control.  For example, consider a dynamic menu control, as follows:  Dynamic menu control with 3 buttons  This is specified using the following XML fragment:   1. <dynamicMenu id="dynamic" label="Dynamic Menu" 2. getContent="GetMenuContent" />   The **GetMenuContent** callback function is called when the menu is dropped, and in this case would return a string with the following XML:   1. <menu 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui"> 3. <button id="button1" label="Button 1" /> 4. <button id="button2" label="Button 2" /> 5. <button id="button3" label="Button 3" /> 6. </menu>   The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image that is used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **invalidateContentOnDrop** (invalidate content on drop) | Specifies whether this control SHOULD invalidate its contents and re-query for them when the user opens its drop-down menu.  If this attribute is omitted, its value SHOULD default to false.  For example, consider the following XML fragment:   1. <comboBox id="comboBox" getItemCount="GetComboBoxItemCount" 2. getItemLabel="GetComboBoxItemLabel" 3. invalidateContentOnDrop="true" />   In this example, this combo box clears out its items and re-calls the **GetComboBoxItemCount** and **GetComboBoxItemLabel** callback functions to populate its contents each time the user opens it.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_DynamicMenuRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ControlBase">
4. <xsd:attributeGroup ref="AG\_Description"/>
5. <xsd:attributeGroup ref="AG\_IDAttributes"/>
6. <xsd:attributeGroup ref="AG\_GetContentAttributes"/>
7. <xsd:attributeGroup ref="AG\_DynamicContentAttributes"/>
8. </xsd:extension>
9. </xsd:complexContent>
10. </xsd:complexType>

### dynamicMenu (Dynamic Menu)

This element specifies a dynamic menu control that populates its contents dynamically.

For example, consider a dynamic menu control, as follows:



Figure 10: A dynamic menu control

This is specified using the following XML fragment:

1. <dynamicMenu id="dynamic" label="Dynamic Menu" getContent="GetMenuContent" />

The **GetMenuContent** callback function is called when the menu is dropped, and in this case would return a string with the following XML:

1. <menu xmlns="http://schemas.microsoft.com/office/2006/01/customui">
2. <button id="button1" label="Button 1" />
3. <button id="button2" label="Button 2" />
4. <button id="button3" label="Button 3" />
5. </menu>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getContent** (**getContent** callback) | Specifies the name of a callback function to be called when the application needs to determine the contents of the control.  For example, consider a dynamic menu control, as follows:  Dynamic menu control with 3 buttons  This is specified using the following XML fragment:   1. <dynamicMenu id="dynamic" label="Dynamic Menu" 2. getContent="GetMenuContent" />   The **GetMenuContent** callback function is called when the menu is dropped, and in this case would return a string with the following XML:   1. <menu 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui"> 3. <button id="button1" label="Button 1" /> 4. <button id="button2" label="Button 2" /> 5. <button id="button3" label="Button 3" /> 6. </menu>   The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the ST\_ID simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **invalidateContentOnDrop** (invalidate content on drop) | Specifies whether this control SHOULD invalidate its contents and re-query for them when the user opens its drop-down menu.  If this attribute is omitted, its value SHOULD default to false.  For example, consider the following XML fragment:   1. <comboBox id="comboBox" getItemCount="GetComboBoxItemCount" 2. getItemLabel="GetComboBoxItemLabel" 3. invalidateContentOnDrop="true" />   In this example, this combo box clears out its items and re-calls the **GetComboBoxItemCount** and **GetComboBoxItemLabel** callback functions to populate its contents each time the user opens it.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_DynamicMenu">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_DynamicMenuRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### editBox (Edit Box)

This element specifies an edit box control that allows a user to enter a **string** of text.

For example, consider an edit box control, as follows:



Figure 11: An edit box control

This is specified using the following XML fragment:

1. <editBox id="editBox" label="Edit Box" />

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (Enabled State) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getText** (**getText** callback) | Specifies the name of a callback function to be called to determine the text that SHOULD be displayed in the control.  For example, consider the following XML fragment:   1. <editBox id="editBox" getText="GetEditBoxText" />   In this example, the **GetEditBoxText** callback function is called when the application needs to determine the text to display in the control.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom button with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **maxLength** (maximum input **string** length) | Specifies an **integer** to be used as the maximum length of a **string** that can be entered into the control.  If the **maxLength** attribute is omitted, the length of the input **string** SHOULD NOT be limited except by application-specific constraints.  For example, consider the following XML fragment:   1. <editBox id="editBox" maxLength="10" />   This specifies an edit box control that can only accept **strings** up to 10 characters in length.  The possible values for this attribute are defined by the **ST\_StringLength** simple type, as specified in section [2.3.12](#Section_4a327c6229ec496e935fa1aa77d26d5e). |
| **onChange** (**onChange** callback) | Specifies the name of a callback function to be called when the text in the control has been changed by the user.  For example, consider the following XML fragment:   1. <editBox id="editBox" onChange="EditBoxTextChanged" />   This specifies an edit box control that calls the **EditBoxTextChanged** callback function when the user inputs a text **string**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (Show Label) | Specifies whether this control displays its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **sizeString** (size string) | Specifies a string whose size is used to determine the width of the text input area of this control.  If this attribute is omitted, the application SHOULD determine the width of the text input area of the control automatically.  For example, consider the following XML fragment:   1. <editBox id="editBox" sizeString="WWWWWWWWWWWWW" />   This specifies an edit box control that is wide enough to display the **string** "WWWWWWWWWWWWW".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_EditBox">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_Control">
4. <xsd:attributeGroup ref="AG\_Enabled"/>
5. <xsd:attributeGroup ref="AG\_Image"/>
6. <xsd:attribute name="maxLength" type="ST\_StringLength" use="optional"/>
7. <xsd:attribute name="getText" type="ST\_Delegate" use="optional"/>
8. <xsd:attribute name="onChange" type="ST\_Delegate" use="optional"/>
9. <xsd:attribute name="sizeString" type="ST\_String" use="optional"/>
10. </xsd:extension>
11. </xsd:complexContent>
12. </xsd:complexType>

### gallery (Gallery)

This element specifies a gallery control, which displays a drop-down grid of items that the user can select from. A gallery can optionally have buttons following its selection items.

For example, consider a gallery control that shows a selection of pictures, as follows:



Figure 12: A gallery control

This is specified using the following XML fragment:

1. <gallery id="gallery" label="Gallery" itemWidth="88" itemHeight="68"
2. size="large" imageMso="HappyFace" >
3. <item id="item1" image="Desert" />
4. <item id="item2" image="Forest" />
5. <item id="item3" image="Toucan" />
6. <item id="item4" image="Tree" />
7. </gallery>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **item** (Selection Item) | [2.2.24](#Section_6417edcc85b847f781fd56918e8df537) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **columns** (column count) | Specifies the number of columns that the gallery's items SHOULD be arranged into.  If the **columns** attribute is omitted, the application SHOULD choose the number of columns automatically based on the number of items.  For example, consider a gallery control with six items arranged into two columns, as follows:  Gallery control with six items in two columns, 3 items per column  This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" columns="2" 2. size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. <item id="item5" image="Flowers" /> 8. <item id="item6" image="Whale" /> 9. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryRowColumnCount** simple type, as specified in section [2.3.4](#Section_b09148682e5949dd96cfdf995849a1bf). |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemCount** (**getItemCount** callback) | Specifies the name of a callback function to be called to determine the number of selection items in this control.  If this attribute is omitted, the control SHOULD display any selection items that are specified as child elements. If no such items are specified, the control SHOULD be empty.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" />   In this example, the **GetGalleryItemCount** callback function is called when the application needs to determine the number of items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemHeight** (**getItemHeight** callback) | Specifies the name of a callback function to be called to determine the height of the selection items in this control.  The **itemHeight** and **getItemHeight** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **getItemHeight** and **getItemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemHeight="GetGalleryItemHeight" 3. getItemWidth="GetGalleryItemWidth" />   In this example, the **GetGalleryItemHeight** callback function is called when the application needs to determine the height of the items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemID** (**getItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD have empty identifiers.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetGalleryItemID" />   In this example, the **GetGalleryItemID** callback function is called when the application needs to determine the identifier of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemImage** (**getItemImage** callback) | Specifies the name of a callback function to be called to determine the icon of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display icons.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemImage="GetGalleryItemImage" />   In this example, the **GetGalleryItemImage** callback function is called when the application needs to determine the icon of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemLabel** (**getItemLabel** callback) | Specifies the name of a callback function to be called to determine the label of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display labels.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemLabel="GetGalleryItemLabel" />   In this example, the **GetGalleryItemLabel** callback function is called when the application needs to determine the label of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemScreentip** (**getItemScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD use their labels as their screentips, or display no screentips at all.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemScreentip="GetGalleryItemScreentip" />   In this example, the **GetGalleryItemScreentip** callback function is called when the application needs to determine the screentip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemSupertip** (**getItemSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display supertips.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemSupertip="GetGalleryItemSupertip" />   In this example, the **GetGalleryItemSupertip** callback function is called when the application needs to determine the supertip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemWidth** (**getItemWidth** callback) | Specifies the name of a callback function to be called to determine the width of the selection items in this control.  The **itemWidth** and **getItemWidth** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **getItemHeight** and **getItemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemHeight="GetGalleryItemHeight" 3. getItemWidth="GetGalleryItemWidth" />   In this example, the **GetGalleryItemWidth** callback function is called when the application needs to determine the width of the items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemID** (**getSelectedItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of the item that is selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetItemID" 3. getSelectedItemID="GetGallerySelectedItemID" />   In this example, the **GetGallerySelectedItemID** callback function is called when the application needs to determine the selected item in the gallery. In this example the callback function returns one of the identifiers returned by the **GetItemID** callback function.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemIndex** (**getSelectedItemIndex** callback) | Specifies the name of a callback function to be called to determine the index of the item to be selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getSelectedItemIndex="GetGallerySelectedItemIndex" />   In this example, the **GetGallerySelectedItemIndex** callback function is called when the application needs to determine the selected item in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **invalidateContentOnDrop** (invalidate content on drop) | Specifies whether this control SHOULD invalidate its contents and re-query for them when the user opens its drop-down menu.  If this attribute is omitted, its value SHOULD default to false.  For example, consider the following XML fragment:   1. <comboBox id="comboBox" getItemCount="GetComboBoxItemCount" 2. getItemLabel="GetComboBoxItemLabel" 3. invalidateContentOnDrop="true" />   In this example, this combo box SHOULD clear out its items and re-call the **GetComboBoxItemCount** and **GetComboBoxItemLabel** callback functions to populate its contents each time the user opens it.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **itemHeight** (selection item height) | Specifies the height of the selection items in this control.  The **itemHeight** and **getItemHeight** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **itemHeight** and **itemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider a gallery control with 68 pixel tall items. This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" itemWidth="88" 2. itemHeight="68" size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryItemWidthHeight** simple type, as specified in section [2.3.3](#Section_76f0a96c409d49a5927028e9b6b7f0a4). |
| **itemWidth** (selection item width) | Specifies the width of the selection items in this control.  The **itemWidth** and **getItemWidth** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **itemHeight** and **itemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider a gallery control with 88 pixel wide items. This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" itemWidth="88" 2. itemHeight="68" size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryItemWidthHeight** simple type, as specified in section 2.3.3. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   In this example, the button calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **rows** (row count) | Specifies the number of rows that the gallery's items are arranged into.  If the **rows** attribute is omitted, the application SHOULD choose the number of rows automatically based on the number of items.  For example, consider a gallery control with six items arranged into two rows, as follows:  Gallery control with six items in two rows  This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" rows="2" 2. size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. <item id="item5" image="Flowers" /> 8. <item id="item6" image="Whale" /> 9. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryRowColumnCount** simple type, as specified in section 2.3.4. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemImage** (show item image) | Specifies whether this control displays icons on its selection items.  If this attribute is omitted, the items' icons SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemImage="false" > 2. <item id="item1" label="Item 1" /> 3. <item id="item2" label="Item 1" /> 4. <item id="item3" label="Item 2" /> 5. <item id="item4" label="Item 3" /> 6. </gallery>   This specifies a gallery control that does not show any icons on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemLabel** (show item label) | Specifies whether this control displays labels on its selection items.  If this attribute is omitted, the item's labels SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemLabel="false" > 2. <item id="item1" image="Forest" /> 3. <item id="item2" image="Desert" /> 4. <item id="item3" image="Mountain " /> 5. <item id="item4" image="Ocean" /> 6. </gallery>   In this example, the gallery control does not show any labels on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showInRibbon** (show in ribbon) | This attribute has no meaning and MUST NOT be used. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **sizeString** (size string) | Specifies a string whose size is used to determine the width of the text input area of this control.  If this attribute is omitted, the application SHOULD determine the width of the text input area of the control automatically.  For example, consider the following XML fragment:   1. <editBox id="editBox" sizeString="WWWWWWWWWWWWW" />   This specifies an edit box control that is wide enough to display the **string** "WWWWWWWWWWWWW".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Gallery">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_GalleryRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### gallery (Unsized Gallery)

This element specifies a gallery which, because of its location, cannot have its size changed. The **size** attribute is not present. It otherwise behaves identically to the regular **gallery** element, as specified in section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **item** (Selection Item) | [2.2.24](#Section_6417edcc85b847f781fd56918e8df537) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **columns** (column count) | Specifies the number of columns that the gallery's items are arranged into.  If the **columns** attribute is omitted, the application SHOULD choose the number of columns automatically based on the number of items.  For example, consider a gallery control with six items arranged into two columns, as follows:  A gallery control with six items arranged in two columns. 3 items per column  This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" columns="2" 2. size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. <item id="item5" image="Flowers" /> 8. <item id="item6" image="Whale" /> 9. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryRowColumnCount** simple type, as specified in section [2.3.4](#Section_b09148682e5949dd96cfdf995849a1bf). |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the ST\_LongString simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemCount** (**getItemCount** callback) | Specifies the name of a callback function to be called to determine the number of selection items in this control.  If this attribute is omitted, the control SHOULD display any selection items that are specified as child elements. If no such items are specified, the control SHOULD be empty.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" />   In this example, the **GetGalleryItemCount** callback function is called when the application needs to determine the number of items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemHeight** (**getItemHeight** callback) | Specifies the name of a callback function to be called to determine the height of the selection items in this control.  The **itemHeight** and **getItemHeight** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **getItemHeight** and **getItemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemHeight="GetGalleryItemHeight" 3. getItemWidth="GetGalleryItemWidth" />   In this example, the **GetGalleryItemHeight** callback function is called when the application needs to determine the height of the items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemID** (**getItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD have empty identifiers.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetGalleryItemID" />   In this example, the **GetGalleryItemID** callback function is called when the application needs to determine the identifier of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemImage** (**getItemImage** callback) | Specifies the name of a callback function to be called to determine the icon of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display icons.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemImage="GetGalleryItemImage" />   In this example, the **GetGalleryItemImage** callback function is called when the application needs to determine the icon of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemLabel** (**getItemLabel** callback) | Specifies the name of a callback function to be called to determine the label of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display labels.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemLabel="GetGalleryItemLabel" />   In this example, the **GetGalleryItemLabel** callback function is called when the application needs to determine the label of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemScreentip** (**getItemScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD use their labels as their screentips, or display no screentips at all.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemScreentip="GetGalleryItemScreentip" />   In this example, the **GetGalleryItemScreentip** callback function is called when the application needs to determine the screentip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemSupertip** (**getItemSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of a specific dynamically-created selection item, identified by index.  If this attribute is omitted, dynamically-created selection items SHOULD NOT display supertips.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemSupertip="GetGalleryItemSupertip" />   In this example, the **GetGalleryItemSupertip** callback function is called when the application needs to determine the supertip of a selection item.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getItemWidth** (**getItemWidth** callback) | Specifies the name of a callback function to be called to determine the width of the selection items in this control.  The **itemWidth** and **getItemWidth** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **getItemHeight** and **getItemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemHeight="GetGalleryItemHeight" 3. getItemWidth="GetGalleryItemWidth" />   In this example, the **GetGalleryItemWidth** callback function is called when the application needs to determine the width of the items in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemID** (**getSelectedItemID** callback) | Specifies the name of a callback function to be called to determine the identifier of the item to be selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getItemID="GetItemID" 3. getSelectedItemID="GetGallerySelectedItemID" />   In this example, the **GetGallerySelectedItemID** callback function is called when the application needs to determine the selected item in the gallery. The callback function returns one of the identifiers returned by the **GetItemID** callback function.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSelectedItemIndex** (**getSelectedItemIndex** callback) | Specifies the name of a callback function to be called to determine the index of the item to be selected in this control.  The **getSelectedItemID** and **getSelectedItemIndex** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display a selected item.  For example, consider the following XML fragment:   1. <gallery id="gallery" getItemCount="GetGalleryItemCount" 2. getSelectedItemIndex="GetGallerySelectedItemIndex" />   In this example, the **GetGallerySelectedItemIndex** callback function is called when the application needs to determine the selected item in the gallery.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **invalidateContentOnDrop** (invalidate content on drop) | Specifies whether this control SHOULD invalidate its contents and re-query for them when the user opens its drop-down menu.  If this attribute is omitted, its value SHOULD default to false.  For example, consider the following XML fragment:   1. <comboBox id="comboBox" getItemCount="GetComboBoxItemCount" 2. getItemLabel="GetComboBoxItemLabel" 3. invalidateContentOnDrop="true" />   In this example, this combo box SHOULD clear out its items and re-call the **GetComboBoxItemCount** and **GetComboBoxItemLabel** callback functions to populate its contents each time the user opens it.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **itemHeight** (selection item height) | Specifies the height of the selection items in this control.  The **itemHeight** and **getItemHeight** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **itemHeight** and **itemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider a gallery control with 68 pixel tall items. This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" itemWidth="88" 2. itemHeight="68" size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryItemWidthHeight** simple type, as specified in section [2.3.3](#Section_76f0a96c409d49a5927028e9b6b7f0a4). |
| **itemWidth** (selection item width) | Specifies the width of the selection items in this control.  The **itemWidth** and **getItemWidth** attributes are mutually exclusive. If neither attribute is specified, the items SHOULD all take the size of the first item, based on its contents.  The **itemHeight** and **itemWidth** attributes are mutually required. If only one of the attributes is specified, its value is ignored.  For example, consider a gallery control with 88 pixel wide items. This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" itemWidth="88" 2. itemHeight="68" size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryItemWidthHeight** simple type, as specified in section 2.3.3. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   In this example, the button calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **rows** (row count) | Specifies the number of rows that the gallery's items are arranged into.  If the **rows** attribute is omitted, the application SHOULD choose the number of rows automatically based on the number of items.  For example, consider a gallery control with six items arranged into two rows, as follows:  Gallery control with six items arranged in two rows. Three items per row  This is specified using the following XML fragment:   1. <gallery id="gallery" label="Gallery" rows="2" 2. size="large" imageMso="HappyFace" > 3. <item id="item1" image="Desert" /> 4. <item id="item2" image="Forest" /> 5. <item id="item3" image="Toucan" /> 6. <item id="item4" image="Tree" /> 7. <item id="item5" image="Flowers" /> 8. <item id="item6" image="Whale" /> 9. </gallery>   The possible values for this attribute are defined by the **ST\_GalleryRowColumnCount** simple type, as specified in section 2.3.4. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemImage** (show item image) | Specifies whether this control displays icons on its selection items.  If this attribute is omitted, the items' icons SHOULD be shown by default.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemImage="false" > 2. <item id="item1" label="Item 1" /> 3. <item id="item2" label="Item 1" /> 4. <item id="item3" label="Item 2" /> 5. <item id="item4" label="Item 3" /> 6. </gallery>   This specifies a gallery control that does not show any icons on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showItemLabel** (show item label) | Specifies whether this control displays labels on its selection items.  For example, consider the following XML fragment:   1. <gallery id="gallery" label="Gallery" showItemLabel="false" > 2. <item id="item1" image="Forest" /> 3. <item id="item2" image="Desert" /> 4. <item id="item3" image="Mountain" /> 5. <item id="item4" image="Ocean" /> 6. </gallery>   In this example, the gallery control does not show any labels on its selection items.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **sizeString** (size string) | Specifies a string whose size is used to determine the width of the text input area of this control.  If this attribute is omitted, the application SHOULD determine the width of the text input area of the control automatically.  For example, consider the following XML fragment:   1. <editBox id="editBox" sizeString="WWWWWWWWWWWWW" />   This specifies an edit box control that is wide enough to display the **string** "WWWWWWWWWWWWW".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_GalleryRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_DropDownRegular">
4. <xsd:attributeGroup ref="AG\_Description"/>
5. <xsd:attributeGroup ref="AG\_DynamicContentAttributes"/>
6. <xsd:attribute name="columns" type="ST\_GalleryRowColumnCount" use="optional"/>
7. <xsd:attribute name="rows" type="ST\_GalleryRowColumnCount" use="optional"/>
8. <xsd:attribute name="itemWidth" type="ST\_GalleryItemWidthHeight" use="optional"/>
9. <xsd:attribute name="itemHeight" type="ST\_GalleryItemWidthHeight" use="optional"/>
10. <xsd:attribute name="getItemWidth" type="ST\_Delegate" use="optional"/>
11. <xsd:attribute name="getItemHeight" type="ST\_Delegate" use="optional"/>
12. <xsd:attribute name="showItemLabel" type="xsd:boolean" use="optional"/>
13. </xsd:extension>
14. </xsd:complexContent>
15. </xsd:complexType>

### group (Group)

This element specifies a grouping of controls on a ribbon tab. All controls displayed in a ribbon tab MUST be contained within a **group**.

For example, consider a group with a single button, as follows:



Figure 13: A group with a single button

This is specified using the following XML fragment:

1. <group id="group" label="Custom Group">
2. <button id="button" label="Button" imageMso="HappyFace" />
3. </group>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **tab** (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **box** (Box Grouping Container) | [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e) |
| **button** (Button) | [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b) |
| **buttonGroup** (Button Grouping Container) | [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **comboBox** (Combo Box) | [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e) |
| **control** (Control Clone) | [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755) |
| **dialogBoxLauncher** (Dialog Box Launcher) | [2.2.15](#Section_76d9f8786b2749ff8c4cee008f345cac) |
| **dropDown** (Drop-down Control) | [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69) |
| **dynamicMenu** (Dynamic Menu) | [2.2.19](#Section_26d7db807ea442d09a063f63365043c7) |
| **editBox** (Edit Box) | [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34) |
| **gallery** (Gallery) | [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f) |
| **labelControl** (Text Label) | [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb) |
| **menu** (Menu) | [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf) |
| **separator** (Separator) | [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b) |
| **splitButton** (Split Button) | [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8) |
| **toggleButton** (Toggle Button) | [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Group">
2. <xsd:sequence>
3. <xsd:sequence>
4. <xsd:choice minOccurs="0" maxOccurs="1000">
5. <xsd:group ref="EG\_Controls"/>
6. <xsd:element name="separator" type="CT\_Separator"/>
7. </xsd:choice>
8. </xsd:sequence>
9. <xsd:element name="dialogBoxLauncher" type="CT\_DialogLauncher" minOccurs="0"
10. maxOccurs="1"/>
11. </xsd:sequence>
12. <xsd:attributeGroup ref="AG\_IDAttributes"/>
13. <xsd:attributeGroup ref="AG\_Label"/>
14. <xsd:attributeGroup ref="AG\_Image"/>
15. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
16. <xsd:attributeGroup ref="AG\_Screentip"/>
17. <xsd:attributeGroup ref="AG\_Visible"/>
18. <xsd:attributeGroup ref="AG\_Keytip"/>
19. </xsd:complexType>

### item (Selection Item)

This element specifies an item in a selection-type control.

For example, consider a drop-down control with three selection items, as follows:

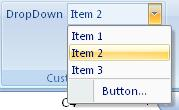


Figure 14: A drop-down control with selection items

This is specified using the following XML fragment:

1. <dropDown id="dropDown" label="DropDown">
2. <item id="item1" label="Item 1" />
3. <item id="item2" label="Item 2" />
4. <item id="item3" label="Item 3" />
5. <button id="button" label="Button..." />
6. </dropDown>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **comboBox** (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); **dropDown** (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); **gallery** (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); **gallery** (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **id** (custom control identifier) | Specifies the identifier for a custom control. All new custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **image** (custom image identifier) | Specifies the identification information for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **image**, and **imageMso** attributes are mutually exclusive.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   In this example, the custom button has an icon that is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **image**, and **imageMso** attributes are mutually exclusive.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   In this example, the custom button uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Item">
2. <xsd:attribute name="id" type="ST\_UniqueID" use="optional"/>
3. <xsd:attribute name="label" type="ST\_String" use="optional"/>
4. <xsd:attribute name="image" type="ST\_Uri" use="optional"/>
5. <xsd:attribute name="imageMso" type="ST\_ID" use="optional"/>
6. <xsd:attribute name="screentip" type="ST\_String" use="optional"/>
7. <xsd:attribute name="supertip" type="ST\_String" use="optional"/>
8. </xsd:complexType>

### labelControl (Text Label)

This element specifies a control that displays a simple string of text.

For example, consider a label control, as follows:



Figure 15: A label control

This is specified using the following XML fragment:

1. <labelControl id="label" label="Label Control" />

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_LabelControl">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_Control">
4. <xsd:attribute name="image" use="prohibited"/>
5. <xsd:attribute name="imageMso" use="prohibited"/>
6. <xsd:attribute name="getImage" use="prohibited"/>
7. <xsd:attribute name="keytip" use="prohibited"/>
8. <xsd:attribute name="getKeytip" use="prohibited"/>
9. <xsd:attribute name="showImage" use="prohibited"/>
10. <xsd:attribute name="getShowImage" use="prohibited"/>
11. </xsd:restriction>
12. </xsd:complexContent>
13. </xsd:complexType>

### menu (Unsized Menu)

This element specifies a menu control that, because of its location, cannot have its size changed. The **size** attribute is not present. It otherwise behaves identically to the regular **menu** element, as specified in section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section 2.2.28); **menu** (section 2.2.26); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **splitButton** (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); **splitButton** (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Unsized Menu) | 2.2.26 |
| **menuSeparator** (Menu Separator) | [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686) |
| **splitButton** (Unsized Split Button) | 2.2.36 |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **itemSize** (item size) | Specifies the size of the child controls in this menu.  If this attribute is omitted, the menu's child controls SHOULD default to the normal size.  For example, consider a menu control with large menu items, as follows:  Menu control with 3 large buttons  This is specified using the following XML fragment:   1. <menu id="menu" label="Menu with large items" itemSize="large"> 2. <button id="button1" label="Button 1" imageMso="HappyFace" /> 3. <button id="button2" label="Button 2" imageMso="Paste" /> 4. <button id="button3" label="Button 3" imageMso="Copy" /> 5. </menu>   The possible values for this attribute are defined by the **ST\_ItemSize** simple type, as specified in section [2.3.6](#Section_c65c8b6052e5404dad5a8c09627c60eb). |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_MenuRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ControlBase">
4. <xsd:sequence>
5. <xsd:choice minOccurs="0" maxOccurs="1000">
6. <xsd:group ref="EG\_MenuControlsBase"/>
7. <xsd:group ref="EG\_MenuOrSplitButtonRegular"/>
8. </xsd:choice>
9. </xsd:sequence>
10. <xsd:attribute name="itemSize" type="ST\_ItemSize" use="optional"/>
11. <xsd:attributeGroup ref="AG\_Description"/>
12. <xsd:attributeGroup ref="AG\_IDAttributes"/>
13. </xsd:extension>
14. </xsd:complexContent>
15. </xsd:complexType>

### menu (Menu with Title)

This element specifies a menu control that, because of its location, can optionally include a title string via the **title** or **getTitle** attributes. It otherwise behaves identically to the regular **menu** element, as specified in section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf).

For example, consider a menu control with a title, as follows:

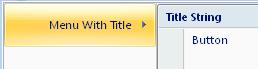


Figure 16: A menu control with title

This is specified with the following XML fragment:

1. <menu id="menu" label="Menu With Title" title="Title String">
2. <button id="button" label="Button" />
3. </menu>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **menu** (section 2.2.27); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)); **splitButton** (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Menu with Title) | 2.2.27 |
| **menuSeparator** (Menu Separator) | [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686) |
| **splitButton** (Split Button with Title) | 2.2.37 |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getTitle** (**getTitle** callback) | Specifies the name of a callback function to be called to determine the title of this control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified no title SHOULD be shown.  For example, consider the following XML fragment:   1. <menu id="menu" label="Menu" getTitle="GetMenuTitle"> 2. … 3. </menu>   In this example, the **GetMenuTitle** callback function is called when the application needs to determine the title of the menu.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **itemSize** (item size) | Specifies the size of the child controls in this menu.  If this attribute is omitted, the menu's child controls SHOULD default to the normal size.  For example, consider a menu control with large menu items, as follows:  Menu control with 3 large buttons  This is specified using the following XML fragment:   1. <menu id="menu" label="Menu with large items" itemSize="large"> 2. <button id="button1" label="Button 1" imageMso="HappyFace" /> 3. <button id="button2" label="Button 2" imageMso="Paste" /> 4. <button id="button3" label="Button 3" imageMso="Copy" /> 5. </menu>   The possible values for this attribute are defined by the **ST\_ItemSize** simple type, as specified in section [2.3.6](#Section_c65c8b6052e5404dad5a8c09627c60eb). |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **title** (title) | Specifies a **string** to be displayed as the title of the control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified, no title SHOULD be shown.  For example, consider a menu control with a title, as follows:  Menu control with a title  This is specified with the following XML fragment:   1. <menu id="menu" label="Menu With Title" title="Title String"> 2. <button id="button" label="Button" /> 3. </menu>   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_MenuWithTitle">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ControlBase">
4. <xsd:sequence>
5. <xsd:choice minOccurs="0" maxOccurs="1000">
6. <xsd:group ref="EG\_MenuControlsBase"/>
7. <xsd:group ref="EG\_MenuOrSplitButtonWithTitle"/>
8. </xsd:choice>
9. </xsd:sequence>
10. <xsd:attributeGroup ref="AG\_IDAttributes"/>
11. <xsd:attribute name="itemSize" type="ST\_ItemSize" use="optional"/>
12. <xsd:attributeGroup ref="AG\_Title"/>
13. </xsd:extension>
14. </xsd:complexContent>
15. </xsd:complexType>

### menu (Menu)

This element specifies a drop-menu control.

For example, consider a menu control, as follows:



Figure 17: A menu control

This is specified using the following XML fragment:

1. <menu id="menu" label="Menu" imageMso="HappyFace" >
2. <button id="button1" label="Button 1" imageMso="FileSave" />
3. <button id="button2" label="Button 2" imageMso="Bold" />
4. <button id="button3" label="Button 3" imageMso="Undo" />
5. </menu>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Unsized Menu) | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **menuSeparator** (Menu Separator) | [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686) |
| **splitButton** (Unsized Split Button) | [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d) |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **itemSize** (item size) | Specifies the size of the child controls in this menu.  If this attribute is omitted, the menu's child controls SHOULD default to the normal size.  For example, consider a menu control with large menu items, as follows:  Menu with 3 large buttons  This is specified using the following XML fragment:   1. <menu id="menu" label="Menu with large items" itemSize="large"> 2. <button id="button1" label="Button 1" imageMso="HappyFace" /> 3. <button id="button2" label="Button 2" imageMso="Paste" /> 4. <button id="button3" label="Button 3" imageMso="Copy" /> 5. </menu>   The possible values for this attribute are defined by the **ST\_ItemSize** simple type, as specified in section [2.3.6](#Section_c65c8b6052e5404dad5a8c09627c60eb). |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Menu">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_MenuRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. <xsd:attribute name="itemSize" type="ST\_ItemSize" use="optional"/>
6. </xsd:extension>
7. </xsd:complexContent>
8. </xsd:complexType>

### menu (Dynamic Menu Root XML Element)

This element specifies the root tag of the XML **string** returned by a dynamic menu control.

For example, consider a dynamic menu control, as follows:



Figure 18: A dynamic menu control

This is specified using the following XML fragment:

1. <dynamicMenu id="dynamic" label="Dynamic Menu" getContent="GetMenuContent" />

The **GetMenuContent** callback function is called when the menu is dropped, and in this case returns a string with the following XML:

1. <menu xmlns="http://schemas.microsoft.com/office/2006/01/customui">
2. <button id="button1" label="Button 1" />
3. <button id="button2" label="Button 2" />
4. <button id="button3" label="Button 3" />
5. </menu>

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Unsized Menu) | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **menuSeparator** (Menu Separator) | [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686) |
| **splitButton** (Unsized Split Button) | [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d) |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getTitle** (**getTitle** callback) | Specifies the name of a callback function to be called to determine the title of this control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified no title SHOULD be shown.  For example, consider the following XML fragment:   1. <menu id="menu" label="Menu" getTitle="GetMenuTitle"> 2. … 3. </menu>   In this example, the **GetMenuTitle** callback function is to be called when the application needs to determine the title of the menu.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **itemSize** (item size) | Specifies the size of the child controls in this menu.  If this attribute is not specified, the menu's child controls SHOULD default to the normal size.  For example, consider a menu control with large menu items, as follows:  Menu control with 3 large buttons  This is specified using the following XML fragment:   1. <menu id="menu" label="Menu with large items" itemSize="large"> 2. <button id="button1" label="Button 1" imageMso="HappyFace" /> 3. <button id="button2" label="Button 2" imageMso="Paste" /> 4. <button id="button3" label="Button 3" imageMso="Copy" /> 5. </menu>   The possible values for this attribute are defined by the **ST\_ItemSize** simple type, as specified in section [2.3.6](#Section_c65c8b6052e5404dad5a8c09627c60eb). |
| **title** (title) | Specifies a **string** to be displayed as the title of the control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified, no title SHOULD be shown.  For example, consider a menu control with a title, as follows:  Menu control with a title  This is specified with the following XML fragment:   1. <menu id="menu" label="Menu With Title" title="Title String"> 2. <button id="button" label="Button" /> 3. </menu>   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_MenuRoot">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="1000">
4. <xsd:group ref="EG\_MenuControlsBase"/>
5. <xsd:group ref="EG\_MenuOrSplitButtonRegular"/>
6. </xsd:choice>
7. </xsd:sequence>
8. <xsd:attributeGroup ref="AG\_Title"/>
9. <xsd:attribute name="itemSize" type="ST\_ItemSize" use="optional"/>
10. </xsd:complexType>

### menuSeparator (Menu Separator)

This element specifies a horizontal separator line in a menu control. Menu separators can optionally have title strings, which SHOULD display as headers in the menu.

For example, consider a menu with a separator in between two of its items, as follows:



Figure 19: Menu control with separator

This is specified using the following XML fragment:

1. <menu id="menu" label="Menu" imageMso="HappyFace" >
2. <button id="button1" label="Button 1" imageMso="FileSave" />
3. <menuSeparator id="separator" />
4. <button id="button2" label="Button 2" imageMso="Bold" />
5. </menu>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getTitle** (**getTitle** callback) | Specifies the name of a callback function to be called to determine the title of this control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified no title SHOULD be shown.  For example, consider the following XML fragment:   1. <menu id="menu" label="Menu" getTitle="GetMenuTitle"> 2. … 3. </menu>   In this example, the **GetMenuTitle** callback function is called when the application needs to determine the title of the menu.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **title** (title) | Specifies a **string** to be displayed as the title of the control.  The **title** and **getTitle** attributes are mutually exclusive. If neither attribute is specified, no title SHOULD be shown.  For example, consider a menu control with a title, as follows:  Menu control with a title  This is specified with the following XML fragment:   1. <menu id="menu" label="Menu With Title" title="Title String"> 2. <button id="button" label="Button" /> 3. </menu>   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_MenuSeparator">
2. <xsd:attributeGroup ref="AG\_IDCustom"/>
3. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
4. <xsd:attributeGroup ref="AG\_Title"/>
5. </xsd:complexType>

### officeMenu (Office Menu)

This element specifies the Office Menu of the application. It is used to reference the built-in **Office** Menu. This element SHOULD NOT be specified if the containing Custom UI XML document is a Quick Access Toolbar Customizations part.

For example, consider the following XML fragment:

1. <officeMenu>
2. <control idMso="FileSave" visible="false" />
3. </officeMenu>

This XML fragment specifies that the command with an identifier of "FileSave" on the Office Menu is hidden.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **ribbon** (section [2.2.33](#Section_574eeee87a03406ab95ff9e51e53dd9d)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **checkBox** (Check Box) | [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde) |
| **control** (Unsized Control Clone) | [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf) |
| **dynamicMenu** (Unsized Dynamic Menu) | [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d) |
| **gallery** (Unsized Gallery) | [2.2.22](#Section_7bfba338481f453e97670e0855c8f029) |
| **menu** (Menu with Title) | [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace) |
| **menuSeparator** (Menu Separator) | [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686) |
| **splitButton** (Split Button with Title) | [2.2.37](#Section_1c6721206402408b96dde98650c12da5) |
| **toggleButton** (Unsized Toggle Button) | [2.2.42](#Section_69b127857fce431591c145aa114726ea) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_OfficeMenu">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="1000">
4. <xsd:group ref="EG\_MenuControlsBase"/>
5. <xsd:group ref="EG\_MenuOrSplitButtonWithTitle"/>
6. </xsd:choice>
7. </xsd:sequence>
8. </xsd:complexType>

### qat (Quick Access Toolbar)

This element specifies the quick access toolbar. If the containing Custom UI file is a Ribbon Extensibility part the **qat** element cannot be used unless the **startFromScratch** attribute on the ribbon element is set to "true". In this case only the **sharedControls** child element SHOULD be used. If the containing Custom UI file is a Quick Access Toolbar Customizations part, the **documentControls** child element SHOULD be used.

For example, consider the following controls on the document-specific quick access toolbar:

Controls on the quick access toolbar

Figure 20: Controls on the quick access toolbar

This is specified using the following XML fragment:

1. <qat>
2. <documentControls>
3. <control idMso="CalculateNow" />
4. <control idMso="HyperlinkInsert" />
5. </documentControls>
6. </qat>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **ribbon** (section [2.2.33](#Section_574eeee87a03406ab95ff9e51e53dd9d)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **documentControls** (List of Document-Specific Quick Access Toolbar Controls) | [2.2.16](#Section_de33fcbe3b5b4d30b875d2ec922cf32f) |
| **sharedControls** (List of Shared Quick Access Toolbar Controls) | [2.2.35](#Section_0ca32ee5aae74ff18baa8a8c9be2ebb3) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Qat">
2. <xsd:sequence>
3. <xsd:element name="sharedControls" type="CT\_QatItems" minOccurs="0"/>
4. <xsd:element name="documentControls" type="CT\_QatItems" minOccurs="0"/>
5. </xsd:sequence>
6. </xsd:complexType>

### ribbon (Ribbon)

This element is used to reference the Ribbon of the application and its contents.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **customUI** (section [2.2.14](#Section_8a27e8523f8b424aac6732c58181e9d3)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **contextualTabs** (List of Contextual Tab Sets) | [2.2.10](#Section_75ae6d6f6a8549a2a5fb84b970d54ffb) |
| **officeMenu** (Office Menu) | [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844) |
| **qat** (Quick Access Toolbar) | [2.2.32](#Section_93e2f741a06c40968dbf5a97fefca545) |
| **tabs** (List of Tabs) | [2.2.40](#Section_6cbd1ae9780f421886aec536f6c0e97a) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **startFromScratch** (start from scratch) | Specifies that the application's built-in ribbon UI is reduced to a minimal set of features, providing a clean slate on which to build custom UI.  If this attribute is omitted, its value SHOULD default to "false".  For example, consider the following XML fragment:   1. <ribbon startFromScratch="true"> 2. … 3. </ribbon>   In this example, the application's ribbon is put into start from scratch mode.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Ribbon">
2. <xsd:all>
3. <xsd:element name="officeMenu" type="CT\_OfficeMenu" minOccurs="0" maxOccurs="1"/>
4. <xsd:element name="qat" type="CT\_Qat" minOccurs="0" maxOccurs="1">
5. <xsd:unique name="qatControls">
6. <xsd:selector xpath="\*/\*"/>
7. <xsd:field xpath="@id"/>
8. </xsd:unique>
9. </xsd:element>
10. <xsd:element name="tabs" type="CT\_Tabs" minOccurs="0" maxOccurs="1"/>
11. <xsd:element name="contextualTabs" type="CT\_ContextualTabs" minOccurs="0" maxOccurs="1"/>
12. </xsd:all>
13. <xsd:attribute name="startFromScratch" type="xsd:boolean" use="optional"/>
14. </xsd:complexType>

### separator (Separator)

This element specifies a vertical separator line between two sets of controls, either in the Quick Access Toolbar or within **group** elements.

For example, consider a vertical separator control between two buttons, as follows:



Figure 21: A vertical separator control

This is specified using the following XML fragment:

1. <button id="button1" label="Button 1" imageMso="HappyFace" size="large" />
2. <separator id="separator" />
3. <button id="button2" label="Button 2" imageMso="HappyFace" size="large" />

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **documentControls** (section [2.2.16](#Section_de33fcbe3b5b4d30b875d2ec922cf32f)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); **sharedControls** (section [2.2.35](#Section_0ca32ee5aae74ff18baa8a8c9be2ebb3)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id** and **idQ** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Separator">
2. <xsd:attributeGroup ref="AG\_IDCustom"/>
3. <xsd:attributeGroup ref="AG\_Visible"/>
4. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
5. </xsd:complexType>

### sharedControls (List of Shared Quick Access Toolbar Controls)

This element specifies the section of the quick access toolbar that is shared among all documents. This element SHOULD NOT be specified if the containing Custom UI XML document is a Quick Access Toolbar Customizations part. If the containing Custom UI XML document is a Ribbon Extensibility part, this element can be used if the **startFromScratch** attribute is set to "true" on the ribbon element.

For example, consider a Ribbon Extensibility XML document that adds the two buttons to the shared section of the quick access toolbar:

Shared controls on the quick access toolbar

Figure 22: Shared controls on the quick access toolbar

This is specified using the following XML fragment:

1. <qat>
2. <sharedControls>
3. <button id="button1" imageMso="HappyFace" />
4. <button idMso="Cut" />
5. </sharedControls>
6. </qat>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **qat** (section [2.2.32](#Section_93e2f741a06c40968dbf5a97fefca545)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Unsized Button) | [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd) |
| **control** (Quick Access Toolbar Control Clone) | [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2) |
| **separator** (Separator) | [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_QatItems">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="1000">
4. <xsd:element name="control" type="CT\_ControlCloneQat"/>
5. <xsd:element name="button" type="CT\_ButtonRegular"/>
6. <xsd:element name="separator" type="CT\_Separator"/>
7. </xsd:choice>
8. </xsd:sequence>
9. </xsd:complexType>

### splitButton (Unsized Split Button)

This element specifies a split button control that, because of its location, cannot have its size changed. The **size** attribute is not present. It otherwise behaves identically to the regular **splitButton** element, as specified in section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Button Inside of a Split Button) | [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e) |
| **menu** (Unsized Menu) | 2.2.26 |
| **toggleButton** (Toggle Button Inside of a Split Button) | [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip String  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super stip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_SplitButtonRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_SplitButtonRestricted">
4. <xsd:sequence minOccurs="0">
5. <xsd:choice minOccurs="0">
6. <xsd:element name="button" type="CT\_VisibleButton"/>
7. <xsd:element name="toggleButton" type="CT\_VisibleToggleButton"/>
8. </xsd:choice>
9. <xsd:element name="menu" type="CT\_MenuRegular"/>
10. </xsd:sequence>
11. </xsd:extension>
12. </xsd:complexContent>
13. </xsd:complexType>

### splitButton (Split Button with Title)

This element specifies a split button control that, because of its location, can optionally include a title string via the **title** or **getTitle** attributes. It otherwise behaves identically to the regular **splitButton** element, as specified in section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Button Inside of a Split Button) | [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e) |
| **menu** (Menu with Title) | 2.2.27 |
| **toggleButton** (Toggle Button Inside of a Split Button) | [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no **label** SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_SplitButtonWithTitle">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_SplitButtonRestricted">
4. <xsd:sequence minOccurs="0">
5. <xsd:choice minOccurs="0">
6. <xsd:element name="button" type="CT\_VisibleButton"/>
7. <xsd:element name="toggleButton" type="CT\_VisibleToggleButton"/>
8. </xsd:choice>
9. <xsd:element name="menu" type="CT\_MenuWithTitle"/>
10. </xsd:sequence>
11. </xsd:extension>
12. </xsd:complexContent>
13. </xsd:complexType>

### splitButton (Split Button)

This element specifies a split button control. A split button control is composed of either a button or a toggle button, and a drop-down menu. The icon and label shown on the split button come from the **button** or **toggleButton** child element.

For example, consider a split button control, as follows:



Figure 23: A split button control

This is specified using the following XML fragment:

1. <splitButton id="splitButton" size="large" >
2. <button id="button" imageMso="HappyFace" label="Split Button" />
3. <menu id="menu">
4. <button id="button1" label="Button 1" />
5. <button id="button2" label="Button 2" />
6. </menu>
7. </splitButton>

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **button** (Button Inside of a Split Button) | [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e) |
| **menu** (Unsized Menu) | [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98) |
| **toggleButton** (Toggle Button Inside of a Split Button) | [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip text  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This example is a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_SplitButton">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_SplitButtonRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### tab (Tab)

This element specifies a ribbon tab control.

For example, consider the following XML fragment:

1. <tab id="MyTab" label="My Custom Tab">
2. …
3. </tab>

This XML fragment specifies a custom tab with the **label** "My Custom Tab".

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **tabs** (section [2.2.40](#Section_6cbd1ae9780f421886aec536f6c0e97a)); **tabSet** (section [2.2.41](#Section_90a00968474745128601b61440747033)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **group** (Group) | [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that **tab** cannot be found, it is created. A new **group** belonging to this file is added to the **tab**.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Tab">
2. <xsd:sequence>
3. <xsd:choice minOccurs="0" maxOccurs="100">
4. <xsd:element name="group" type="CT\_Group"/>
5. </xsd:choice>
6. </xsd:sequence>
7. <xsd:attributeGroup ref="AG\_IDAttributes"/>
8. <xsd:attributeGroup ref="AG\_Label"/>
9. <xsd:attributeGroup ref="AG\_PositionAttributes"/>
10. <xsd:attributeGroup ref="AG\_Visible"/>
11. <xsd:attributeGroup ref="AG\_Keytip"/>
12. </xsd:complexType>

### tabs (List of Tabs)

This element specifies a list of ribbon tab controls. This element SHOULD NOT be specified if the containing Custom UI XML document is a Quick Access Toolbar Customizations part.

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **ribbon** (section [2.2.33](#Section_574eeee87a03406ab95ff9e51e53dd9d)) |

The following table summarizes the child elements of this element.

| Child Elements | Section |
| --- | --- |
| **tab** (Tab) | [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed) |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_Tabs">
2. <xsd:sequence>
3. <xsd:element name="tab" type="CT\_Tab" minOccurs="1" maxOccurs="100"/>
4. </xsd:sequence>
5. </xsd:complexType>

### tabSet (Contextual Tab Set)

This element specifies a contextual tab set control. As the **id** and **idQ** attributes are not present, this element can only be used to refer to existing built-in tab sets. This element cannot be used to create new contextual tab sets.

For example, consider the following XML fragment:

1. <tabSet idMso="TabSetPictureTools">
2. <tab id="tab" label="Custom Tab">
3. …
4. </tab>
5. </tabSet>

This XML fragment is used to add a new custom tab to the tab set with an identifier of "TabSetPictureTools".

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **contextualTabs** (section [2.2.10](#Section_75ae6d6f6a8549a2a5fb84b970d54ffb)) |

The following table summarizes the child elements of this element.

| Child Elements | Subclause |
| --- | --- |
| **tab** (Tab) | section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This is used to create a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_TabSet">
2. <xsd:sequence>
3. <xsd:element name="tab" type="CT\_Tab" minOccurs="0" maxOccurs="50"/>
4. </xsd:sequence>
5. <xsd:attribute name="idMso" type="ST\_ID" use="required"/>
6. <xsd:attributeGroup ref="AG\_Visible"/>
7. </xsd:complexType>

### toggleButton (Unsized Toggle Button)

This element specifies a toggle button control that, because of its location, cannot have its size changed. The **size** attribute is not present. It otherwise behaves identically to the regular **toggleButton** element, as specified in section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **buttonGroup** (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); **menu** (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); **menu** (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); **menu** (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); **menu** (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); **officeMenu** (section [2.2.31](#Section_d9422554dbaf42e09e4b6416a80b0844)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which SHOULD be displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getPressed** (**getPressed** callback) | Specifies the name of a callback function to be called to determine the toggled state of this control.  If this attribute is omitted, the control SHOULD default to the off state.  For example, consider the following XML fragment:   1. <toggleButton id="toggle" getPressed="IsButtonToggled" />   In this example, the **IsButtonToggled** callback function is called when the application needs to determine the toggle state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Butto with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a string that SHOULD be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   In this example, the button calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icons  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ToggleButtonRegular">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ButtonRegular">
4. <xsd:attribute name="getPressed" type="ST\_Delegate" use="optional"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### toggleButton (Toggle Button)

This element specifies a toggle button control that can be toggled between the pressed and un-pressed states by the end-user.

For example, consider a toggle button control, as follows:



Figure 24: A toggle button control

This is specified with the following XML fragment:

1. <toggleButton id="toggleButton" label="Toggle Button" />

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **box** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); **group** (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new button that is disabled. A permanently disabled button is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getPressed** (**getPressed** callback) | Specifies the name of a callback function to be called to determine the toggled state of this control.  If this attribute is omitted, the control SHOULD default to the "off" state.  For example, consider the following XML fragment:   1. <toggleButton id="toggle" getPressed="IsButtonToggled" />   In this example, the **IsButtonToggled** callback function is called when the application needs to determine the toggle state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application displays the icon of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application displays the label of this control.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSize** (**getSize** callback) | Specifies the name of a callback function to be called to determine the size of this control.  The **getSize** and **size** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider the following XML fragment:   1. <button id="button" getSize="GetButtonSize" />   In this example, the **GetButtonSize** callback function is called when the application needs to determine the size of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control.  The **getVisible** and **visible** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an **id** of ""MyButton"".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image to be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom **button** whose icon is the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image to be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that uses the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control is to be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control is to be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an **id** of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   This specifies a **button** that calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip text  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control displays its label.  This attribute SHOULD have no effect if the **size** or **getSize** attributes specify that the control is "large".  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the label is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **size** (control size) | Specifies the size of the control.  The **size** and **getSize** attributes are mutually exclusive. If neither attribute is specified, the control's size SHOULD default to the normal size.  For example, consider a large button, as follows:  Large button  This is specified using the following XML fragment:   1. <toggleButton idMso="Bold" size="large" />   The possible values for this attribute are defined by the **ST\_Size** simple type, as specified in section [2.3.10](#Section_acf53d2ea68046de946752a2f4a51631). |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control.  The **getVisible** and **visible** attributes are mutually exclusive. If these attributes are omitted, the control SHOULD default to being visible.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_ToggleButton">
2. <xsd:complexContent>
3. <xsd:extension base="CT\_ToggleButtonRegular">
4. <xsd:attributeGroup ref="AG\_SizeAttributes"/>
5. </xsd:extension>
6. </xsd:complexContent>
7. </xsd:complexType>

### toggleButton (Toggle Button Inside of a Split Button)

This element specifies a toggle button control that is part of a split button control. The **visible** and **getVisible** attributes are not present because the visibility is controlled by the split button. This element otherwise behaves in the same way as the regular **toggleButton** element, as specified in section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149).

The following table summarizes the elements that are parents of this element.

| Parent Elements |
| --- |
| **splitButton** (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); **splitButton** (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); **splitButton** (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)) |

The following table summarizes the attributes of this element.

| Attributes | Description |
| --- | --- |
| **description** (description) | Specifies a detailed description of the control, which is displayed in detailed views.  The **description** and **getDescription** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider a button with a detailed description, as follows:  Button with verbose description  This is specified using the following XML fragment:   1. <button id="button" label="Button" imageMso="HappyFace" 2. description="This is a verbose description that describes 3. the function of this control in detail." />   The possible values for this attribute are defined by the **ST\_LongString** simple type, as specified in section [2.3.8](#Section_94ee0d879533413ebf87bbee57270f7e). |
| **enabled** (enabled state) | Specifies the enabled state of the control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  This attribute cannot be used to enable a built-in control that would otherwise be disabled by the application.  For example, consider the following XML fragment:   1. <button id="button" label="Disabled Button" enabled="false" />   This specifies a new **button** that is disabled. A permanently disabled **button** is not very useful, thus the **enabled** attribute is not commonly used.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **getDescription** (**getDescription** callback) | Specifies the name of a callback function to be called to determine the detailed description of this control.  The **getDescription** and **description** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD NOT display any detailed text.  For example, consider the following XML fragment:   1. <button id="button" getDescription="GetButtonDescription" />   In this example, the **GetButtonDescription** callback function is called when the application needs to determine the detailed description of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section [2.3.2](#Section_188cc098eef7453e895cd96e8e99576c). |
| **getEnabled** (**getEnabled** callback) | Specifies the name of a callback function to be called to determine the enabled state of this control.  The **getEnabled** and **enabled** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to being enabled.  For example, consider the following XML fragment:   1. <button id="button" getEnabled="IsButtonEnabled" />   In this example, the **IsButtonEnabled** callback function is called when the application needs to determine the enabled state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getImage** (**getImage** callback) | Specifies the name of a callback function to be called to determine the icon of this control.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getImage="GetButtonImage" />   In this example, the **GetButtonImage** callback function is called when the application needs to determine the **icon** of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getKeytip** (**getKeytip** callback) | Specifies the name of a callback function to be called to determine the suggested [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) of this control.  The **getKeytip** and **keytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider the following XML fragment:   1. <button id="button" getKeytip="GetButtonKeytip" />   In this example, the **GetButtonKeytip** callback function is called when the application needs to determine the KeyTip of the **button**.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getLabel** (**getLabel** callback) | Specifies the name of a callback function to be called to determine the label of this control.  The **getLabel** and **label** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" getLabel="GetButtonLabel" />   In this example, the **GetButtonLabel** callback function is called when the application needs to determine the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getPressed** (**getPressed** callback) | Specifies the name of a callback function to be called to determine the toggled state of this control.  If this attribute is omitted, the control SHOULD default to the off state.  For example, consider the following XML fragment:   1. <toggleButton id="toggle" getPressed="IsButtonToggled" />   In this example, the **IsButtonToggled** callback function is called when the application needs to determine the toggle state of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getScreentip** (**getScreentip** callback) | Specifies the name of a callback function to be called to determine the screentip of this control.  The **getScreentip** and **screentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider the following XML fragment:   1. <button id="button" getScreentip="GetButtonScreentip" />   In this example, the **GetButtonScreentip** callback function is called when the application needs to determine the screentip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowImage** (**getShowImage** callback) | Specifies the name of a callback function to be called to determine whether the application displays the icon of this control.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider the following XML fragment:   1. <button id="button" getShowImage="IsButtonImageVisible" />   In this example, the **IsButtonImageVisible** callback function is called when the application needs to determine whether to display the icon of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getShowLabel** (**getShowLabel** callback) | Specifies the name of a callback function to be called to determine whether the application SHOULD display the label of this control.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" getShowLabel="IsButtonLabelVisible" />   In this example, the **IsButtonLabelVisible** callback function is called when the application needs to determine whether to display the label of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getSupertip** (**getSupertip** callback) | Specifies the name of a callback function to be called to determine the supertip of this control.  The **getSupertip** and **supertip** attributes are mutually exclusive. If neither attribute is specified, no supertip for this control SHOULD be shown.  For example, consider the following XML fragment:   1. <button id="button" getSupertip="GetButtonSupertip" />   In this example, the **GetButtonSupertip** callback function is called when the application needs to determine the supertip of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **getVisible** (**getVisible** callback) | Specifies the name of a callback function to be called to determine the visibility state of this control. This attribute is prohibited and the visibility is controlled by the split button.  For example, consider the following XML fragment:   1. <button id="button" getVisible="IsButtonVisible" />   In this example, the **IsButtonVisible** callback function is called when the application needs to determine the visibility of the button.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **id** (control identifier) | Specifies the identifier for a custom control. All custom controls MUST have unique identifiers. The identifier of a control SHOULD be passed to callback functions to identify which control corresponds to the function call.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <button id="MyButton" label="Button" />   This specifies a custom button control with an identifier of "MyButton".  The possible values for this attribute are defined by the **ST\_UniqueID** simple type, as specified in section [2.3.13](#Section_ed3ee661bfc848d3b1598e8917a2bd36). |
| **idMso** (built-in control identifier) | Specifies the identifier of a built-in control.  The contents of this attribute are application-defined.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <control idMso="Bold" />   This creates a clone of the control with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section [2.3.5](#Section_18fbbd72e04f48148f9a45b2c4cc8393). |
| **idQ** (qualified control identifier) | Specifies a qualified identifier for a control.  The **idQ** attribute can be used to reference controls or containers created by other Custom UI documents.  The **id**, **idQ**, and **idMso** attributes are mutually exclusive. At least one of these attributes MUST be specified.  For example, consider the following XML fragment:   1. <customUI 2. xmlns="http://schemas.microsoft.com/office/2006/01/customui" 3. xmlns:ex="http://www.example.com"> 4. <ribbon> 5. <tabs> 6. <tab idQ="ex:OtherTab" label="Shared Tab"> 7. <group id="MyGroup" label="My Group"> 8. … 9. </group> 10. </tab> 11. </tabs> 12. </ribbon> 13. </customUI>   In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the tab.  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section [2.3.9](#Section_7c80b304b2f8432a83ee7f8ec792af0c). |
| **image** (custom image identifier) | Specifies the relationship identifier for an image which SHOULD be used as the icon for this control. This attribute is used to specify an embedded picture that resides locally within the containing file.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" image="ForestPic" />   This specifies a custom button whose icon SHOULD be the embedded image file referenced by the relationship identifier of "ForestPic".  The possible values for this attribute are defined by the **ST\_Uri** simple type, as specified in section [2.3.14](#Section_5512155ce46d445ea7273b99b4009cc6). |
| **imageMso** (built-in image identifier) | Specifies the identifier of a built-in image which SHOULD be used as the icon of this control.  The contents of this attribute are application-defined and SHOULD be ignored if not understood.  The **getImage**, **image**, and **imageMso** attributes are mutually exclusive. If none of these attributes are specified, no icon SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" imageMso="Bold" />   This specifies a custom button that SHOULD use the built-in image with an identifier of "Bold".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterMso** (identifier of built-in control to insert after) | Specifies the identifier of a built-in control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertAfterQ** (qualified identifier of control to insert after) | Specifies the qualified identifier of a control that this control SHOULD be inserted after. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertAfterQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted after the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **insertBeforeMso** (identifier of built-in control to insert before) | Specifies the identifier of a built-in control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeMso="TabHome" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the built-in tab with an identifier of "TabHome".  The possible values for this attribute are defined by the **ST\_ID** simple type, as specified in section 2.3.5. |
| **insertBeforeQ** (qualified identifier of control to insert before) | Specifies the qualified identifier of a control that this control SHOULD be inserted before. If the value of this attribute is not understood, it SHOULD be ignored.  The **insertAfterMso**, **insertAfterQ**, **insertBeforeMso**, and **insertBeforeQ** attributes are mutually exclusive. If none of these attributes are specified, the controls SHOULD be appended to the existing set of controls, in the order they are defined in the XML.  For example, consider the following XML fragment:   1. <tab id="MyTab" insertBeforeQ="x:OtherTab" label="Custom Tab"> 2. … 3. </tab>   In this example, a new custom tab with an identifier of "MyTab" is to be inserted before the custom tab with a qualified identifier of "x:OtherTab".  The possible values for this attribute are defined by the **ST\_QID** simple type, as specified in section 2.3.9. |
| **keytip** (keytip) | Specifies a **string** to be used as the suggested KeyTip for this control.  The **keytip** and **getKeytip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD generate a KeyTip for the control automatically.  For example, consider a button with KeyTip 'K', as follows:  Button with Key Tip K  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" keytip="K" />   The possible values for this attribute are defined by the **ST\_Keytip** simple type, as specified in section [2.3.7](#Section_78170415e2fa4fc0899a76096fdc6aa6). |
| **label** (label) | Specifies a **string** to be used as the label for this control.  The **label** and **getLabel** attributes are mutually exclusive. If neither attribute is specified, no label SHOULD be displayed.  For example, consider the following XML fragment:   1. <button id="button" label="Custom Button" />   This specifies a custom **button** with a **label** of "Custom Button".  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section [2.3.11](#Section_d104fcb261774eb9a4000a5f8ddcd539). |
| **onAction** (**onAction** callback) | Specifies the name of a callback function to be called when this control is invoked by the user.  For example, consider the following XML fragment:   1. <button id="button" label="Button" onAction="ButtonClicked" />   In this example, the button calls the **ButtonClicked** callback function when it is invoked.  The possible values for this attribute are defined by the **ST\_Delegate** simple type, as specified in section 2.3.2. |
| **screentip** (screentip) | Specifies a **string** to be shown as the screentip for this control.  The **screentip** and **getScreentip** attributes are mutually exclusive. If neither attribute is specified, the application SHOULD display the label of the control as the screentip or display no screentip at all.  For example, consider a button with a screentip, as follows:  Button with screen tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="This is the screentip" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **showImage** (show image) | Specifies whether this control displays an icon.  The **showImage** and **getShowImage** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD display its icon.  For example, consider a button that does not display an icon, as follows:  Button with no icon  This is specified using the following XML fragment:   1. <button id="button" showImage="false" 2. label="Button with no icon" />   The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **showLabel** (show label) | Specifies whether this control SHOULD display its label.  The **showLabel** and **getShowLabel** attributes are mutually exclusive. If neither attribute is specified, the control SHOULD default to showing its label.  For example, consider the following XML fragment:   1. <button id="button" label="Label" showLabel="false" 2. imageMso="HappyFace" />   This specifies a **button** that has a **label**, but does not show it. Even though the **label** is hidden, it is provided to accessibility tools.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |
| **supertip** (supertip) | Specifies a **string** to be shown as the supertip of the control.  The **supertip** and **getSupertip** attributes are mutually exclusive. If neither attribute is specified no supertip for this control SHOULD be shown.  For example, consider a control with a supertip, as follows:  Button with super tip string  This is specified using the following XML fragment:   1. <button id="button" imageMso="HappyFace" label="Button" 2. size="large" screentip="Screentip" 3. supertip="This is the supertip string" />   The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **tag** (tag) | Specifies an arbitrary **string** that can be used to hold data or identify the control. The contents of this attribute SHOULD be passed to any callback functions specified on this control.  If this attribute is omitted, the control's **tag** value SHOULD default to an empty **string**.  For example, consider the following XML fragment:   1. <button id="button" label="Button" tag="123456" 2. onAction="ButtonClicked" />   This specifies a **button** with a **tag** value of "123456", which is passed to the **ButtonClicked** callback function.  The possible values for this attribute are defined by the **ST\_String** simple type, as specified in section 2.3.11. |
| **visible** (control visibility) | Specifies the visibility state of the control. This attribute is prohibited and the visibility is controlled by the split button.  For example, consider the following XML fragment:   1. <tab idMso="TabHome" visible="false" />   In this example, the built-in tab with an identifier of "TabHome" is hidden.  The possible values for this attribute are defined by the XML schema **boolean** datatype. |

The following XML schema fragment defines the contents of this element:

1. <xsd:complexType name="CT\_VisibleToggleButton">
2. <xsd:complexContent>
3. <xsd:restriction base="CT\_ToggleButtonRegular">
4. <xsd:attribute name="visible" use="prohibited"/>
5. <xsd:attribute name="getVisible" use="prohibited"/>
6. </xsd:restriction>
7. </xsd:complexContent>
8. </xsd:complexType>

## Simple Types

This is the complete list of simple types in the http://schemas.microsoft.com/office/2006/01/customui namespace.

### ST\_BoxStyle (Box Style)

Specifies the layout style of a **box** control.

This simple type's contents are a restriction of the XML schema **string** datatype.

The following are possible **enumeration** values for this type:

| Enumeration Value | Description |
| --- | --- |
| **horizontal** (Horizontal) | Specifies that the child controls are laid out horizontally. |
| **vertical** (Vertical) | Specifies that the child controls are laid out vertically. |

| Referenced By |
| --- |
| **box@boxStyle** (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_BoxStyle">
2. <xsd:restriction base="xsd:string">
3. <xsd:enumeration value="horizontal"/>
4. <xsd:enumeration value="vertical"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_Delegate (Callback Function Name)

Specifies the name of a callback function. The format of this string is application-defined and SHOULD be ignored if not understood.

Examples of this simple type are [**macro**](#gt_cd2933d3-08d1-4931-bd5c-7ae0a668fe7c) scripts and [**add-in**](#gt_a3be101e-9d37-484a-a5e6-b70d559146c6) callback functions.

This simple type's contents are a restriction of the XML schema string datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 characters.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| box@getVisible (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); button@getDescription (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@getDescription (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@getDescription (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); button@getEnabled (section 2.2.2); button@getEnabled (section 2.2.3); button@getEnabled (section 2.2.4); button@getImage (section 2.2.2); button@getImage (section 2.2.3); button@getImage (section 2.2.4); button@getKeytip (section 2.2.2); button@getKeytip (section 2.2.3); button@getKeytip (section 2.2.4); button@getLabel (section 2.2.2); button@getLabel (section 2.2.3); button@getLabel (section 2.2.4); button@getScreentip (section 2.2.2); button@getScreentip (section 2.2.3); button@getScreentip (section 2.2.4); button@getShowImage (section 2.2.2); button@getShowImage (section 2.2.3); button@getShowImage (section 2.2.4); button@getShowLabel (section 2.2.2); button@getShowLabel (section 2.2.3); button@getShowLabel (section 2.2.4); button@getSize (section 2.2.2); button@getSupertip (section 2.2.2); button@getSupertip (section 2.2.3); button@getSupertip (section 2.2.4); button@getVisible (section 2.2.2); button@getVisible (section 2.2.3); button@getVisible (section 2.2.4); button@onAction (section 2.2.2); button@onAction (section 2.2.3); button@onAction (section 2.2.4); buttonGroup@getVisible (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); checkBox@getDescription (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); checkBox@getEnabled (section 2.2.6); checkBox@getImage (section 2.2.6); checkBox@getKeytip (section 2.2.6); checkBox@getLabel (section 2.2.6); checkBox@getPressed (section 2.2.6); checkBox@getScreentip (section 2.2.6); checkBox@getShowImage (section 2.2.6); checkBox@getShowLabel (section 2.2.6); checkBox@getSupertip (section 2.2.6); checkBox@getVisible (section 2.2.6); checkBox@onAction (section 2.2.6); comboBox@getEnabled (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); comboBox@getImage (section 2.2.7); comboBox@getItemCount (section 2.2.7); comboBox@getItemID (section 2.2.7); comboBox@getItemImage (section 2.2.7); comboBox@ getItemLabel (section 2.2.7); comboBox@getItemScreentip (section 2.2.7); comboBox@getItemSupertip (section 2.2.7); comboBox@getKeytip (section 2.2.7); comboBox@getLabel (section 2.2.7); comboBox@getScreentip (section 2.2.7); comboBox@getShowImage (section 2.2.7); comboBox@getShowLabel (section 2.2.7); comboBox@getSupertip (section 2.2.7); comboBox@getText (section 2.2.7); comboBox@getVisible (section 2.2.7); comboBox@onChange (section 2.2.7); command@getEnabled (section [2.2.8](#Section_21316865fbce4a3fa9ffa8277cce5f2d)); command@onAction (section 2.2.8); control@getDescription (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@getDescription (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@getEnabled (section 2.2.12); control@getEnabled (section 2.2.13); control@getEnabled (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); control@getImage (section 2.2.12); control@getImage (section 2.2.13); control@getImage (section 2.2.11); control@getKeytip (section 2.2.12); control@getKeytip (section 2.2.13); control@getKeytip (section 2.2.11); control@getLabel (section 2.2.12); control@getLabel (section 2.2.13); control@getLabel (section 2.2.11); control@getScreentip (section 2.2.12); control@getScreentip (section 2.2.13); control@getScreentip (section 2.2.11); control@getShowImage (section 2.2.12); control@getShowImage (section 2.2.13); control@getShowImage (section 2.2.11); control@getShowLabel (section 2.2.12); control@getShowLabel (section 2.2.13); control@getShowLabel (section 2.2.11); control@getSize (section 2.2.12); control@getSize (section 2.2.13); control@getSupertip (section 2.2.12); control@getSupertip (section 2.2.13); control@getSupertip (section 2.2.11); control@getVisible (section 2.2.12); control@getVisible (section 2.2.13); control@getVisible (section 2.2.11); control@onAction (section 2.2.12); customUI@loadImage (section [2.2.14](#Section_8a27e8523f8b424aac6732c58181e9d3)); customUI@onLoad (section 2.2.14); dropDown@getEnabled (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dropDown@getImage (section 2.2.17); dropDown@getItemCount (section 2.2.17); dropDown@getItemID (section 2.2.17); dropDown@getItemImage (section 2.2.17); dropDown@getItemLabel (section 2.2.17); dropDown@getItemScreentip (section 2.2.17); dropDown@getItemSupertip (section 2.2.17); dropDown@getKeytip (section 2.2.17); dropDown@getLabel (section 2.2.17); dropDown@getScreentip (section 2.2.17); dropDown@getSelectedItemID (section 2.2.17); dropDown@getSelectedItemIndex (section 2.2.17); dropDown@getShowImage (section 2.2.17); dropDown@getShowLabel (section 2.2.17); dropDown@getSupertip (section 2.2.17); dropDown@getVisible (section 2.2.17); dropDown@onAction (section 2.2.17); dynamicMenu@getContent (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@getContent (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); dynamicMenu@getDescription (section 2.2.19); dynamicMenu@getDescription (section 2.2.18); dynamicMenu@getEnabled (section 2.2.19); dynamicMenu@getEnabled (section 2.2.18); dynamicMenu@getImage (section 2.2.19); dynamicMenu@getImage (section 2.2.18); dynamicMenu@getKeytip (section 2.2.19); dynamicMenu@getKeytip (section 2.2.18); dynamicMenu@getLabel (section 2.2.19); dynamicMenu@getLabel (section 2.2.18); dynamicMenu@getScreentip (section 2.2.19); dynamicMenu@getScreentip (section 2.2.18); dynamicMenu@getShowImage (section 2.2.19); dynamicMenu@getShowImage (section 2.2.18); dynamicMenu@getShowLabel (section 2.2.19); dynamicMenu@getShowLabel (section 2.2.18); dynamicMenu@getSize (section 2.2.19); dynamicMenu@getSupertip (section 2.2.19); dynamicMenu@getSupertip (section 2.2.18); dynamicMenu@getVisible (section 2.2.19); dynamicMenu@getVisible (section 2.2.18); editBox@getEnabled (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); editBox@getImage (section 2.2.20); editBox@getKeytip (section 2.2.20); editBox@getLabel (section 2.2.20); editBox@getScreentip (section 2.2.20); editBox@getShowImage (section 2.2.20); editBox@getShowLabel (section 2.2.20); editBox@getSupertip (section 2.2.20); editBox@getText (section 2.2.20); editBox@getVisible (section 2.2.20); editBox@onChange (section 2.2.20); gallery@getDescription (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@getDescription (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@getEnabled (section 2.2.21); gallery@getEnabled (section 2.2.22); gallery@getImage (section 2.2.21); gallery@getImage (section 2.2.22); gallery@getItemCount (section 2.2.21); gallery@getItemCount (section 2.2.22); gallery@getItemHeight (section 2.2.21); gallery@getItemHeight (section 2.2.22); gallery@getItemID (section 2.2.21); gallery@getItemID (section 2.2.22); gallery@getItemImage (section 2.2.21); gallery@getItemImage (section 2.2.22); gallery@getItemLabel (section 2.2.21); gallery@getItemLabel (section 2.2.22); gallery@getItemScreentip (section 2.2.21); gallery@getItemScreentip (section 2.2.22); gallery@getItemSupertip (section 2.2.21); gallery@getItemSupertip (section 2.2.22); gallery@getItemWidth (section 2.2.21); gallery@getItemWidth (section 2.2.22); gallery@getKeytip (section 2.2.21); gallery@getKeytip (section 2.2.22); gallery@getLabel (section 2.2.21); gallery@getLabel (section 2.2.22); gallery@getScreentip (section 2.2.21); gallery@getScreentip (section 2.2.22); gallery@getSelectedItemID (section 2.2.21); gallery@getSelectedItemID (section 2.2.22); gallery@getSelectedItemIndex (section 2.2.21); gallery@getSelectedItemIndex (section 2.2.22); gallery@getShowImage (section 2.2.21); gallery@getShowImage (section 2.2.22); gallery@getShowLabel (section 2.2.21); gallery@getShowLabel (section 2.2.22); gallery@getSize (section 2.2.21); gallery@getSupertip (section 2.2.21); gallery@getSupertip (section 2.2.22); gallery@getVisible (section 2.2.21); gallery@getVisible (section 2.2.22); gallery@onAction (section 2.2.21); gallery@onAction (section 2.2.22); group@getImage (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); group@getKeytip (section 2.2.23); group@getLabel (section 2.2.23); group@getScreentip (section 2.2.23); group@getSupertip (section 2.2.23); group@getVisible (section 2.2.23); labelControl@getEnabled (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); labelControl@getImage (section 2.2.25); labelControl@getKeytip (section 2.2.25); labelControl@getLabel (section 2.2.25); labelControl@getScreentip (section 2.2.25); labelControl@getShowImage (section 2.2.25); labelControl@getShowLabel (section 2.2.25); labelControl@getSupertip (section 2.2.25); labelControl@getVisible (section 2.2.25); menu@getDescription (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@getDescription (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@getEnabled (section 2.2.28); menu@getEnabled (section 2.2.26); menu@getEnabled (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); menu@getImage (section 2.2.28); menu@getImage (section 2.2.26); menu@getImage (section 2.2.27); menu@getKeytip (section 2.2.28); menu@getKeytip (section 2.2.26); menu@getKeytip (section 2.2.27); menu@getLabel (section 2.2.28); menu@getLabel (section 2.2.26); menu@getLabel (section 2.2.27); menu@getScreentip (section 2.2.28); menu@getScreentip (section 2.2.26); menu@getScreentip (section 2.2.27); menu@getShowImage (section 2.2.28); menu@getShowImage (section 2.2.26); menu@getShowImage (section 2.2.27); menu@getShowLabel (section 2.2.28); menu@getShowLabel (section 2.2.26); menu@getShowLabel (section 2.2.27); menu@getSize (section 2.2.28); menu@getSupertip (section 2.2.28); menu@getSupertip (section 2.2.26); menu@getSupertip (section 2.2.27); menu@getTitle (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); menu@getTitle (section 2.2.27); menu@getVisible (section 2.2.28); menu@getVisible (section 2.2.26); menu@getVisible (section 2.2.27); menuSeparator@getTitle (section [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686)); separator@getVisible (section [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b)); splitButton@getEnabled (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@getEnabled (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@getEnabled (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); splitButton@getImage (section 2.2.38); splitButton@getImage (section 2.2.36); splitButton@getImage (section 2.2.37); splitButton@getKeytip (section 2.2.38); splitButton@getKeytip (section 2.2.36); splitButton@getKeytip (section 2.2.37); splitButton@getLabel (section 2.2.38); splitButton@getLabel (section 2.2.36); splitButton@getLabel (section 2.2.37); splitButton@getScreentip (section 2.2.38); splitButton@getScreentip (section 2.2.36); splitButton@getScreentip (section 2.2.37); splitButton@getShowImage (section 2.2.38); splitButton@getShowImage (section 2.2.36); splitButton@getShowImage (section 2.2.37); splitButton@getShowLabel (section 2.2.38); splitButton@getShowLabel (section 2.2.36); splitButton@getShowLabel (section 2.2.37); splitButton@getSize (section 2.2.38); splitButton@getSupertip (section 2.2.38); splitButton@getSupertip (section 2.2.36); splitButton@getSupertip (section 2.2.37); splitButton@getVisible (section 2.2.38); splitButton@getVisible (section 2.2.36); splitButton@getVisible (section 2.2.37); tab@getKeytip (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); tab@getLabel (section 2.2.39); tab@getVisible (section 2.2.39); tabSet@getVisible (section [2.2.41](#Section_90a00968474745128601b61440747033)); toggleButton@getDescription (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@getDescription (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@getDescription (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)); toggleButton@getEnabled (section 2.2.43); toggleButton@getEnabled (section 2.2.42); toggleButton@getEnabled (section 2.2.44); toggleButton@getImage (section 2.2.43); toggleButton@getImage (section 2.2.42); toggleButton@getImage (section 2.2.44); toggleButton@getKeytip (section 2.2.43); toggleButton@getKeytip (section 2.2.42); toggleButton@getKeytip (section 2.2.44); toggleButton@getLabel (section 2.2.43); toggleButton@getLabel (section 2.2.42); toggleButton@getLabel (section 2.2.44); toggleButton@getPressed (section 2.2.43); toggleButton@getPressed (section 2.2.42); toggleButton@getPressed (section 2.2.44); toggleButton@getScreentip (section 2.2.43); toggleButton@getScreentip (section 2.2.42); toggleButton@getScreentip (section 2.2.44); toggleButton@getShowImage (section 2.2.43); toggleButton@getShowImage (section 2.2.42); toggleButton@getShowImage (section 2.2.44); toggleButton@getShowLabel (section 2.2.43); toggleButton@getShowLabel (section 2.2.42); toggleButton@getShowLabel (section 2.2.44); toggleButton@getSize (section 2.2.43); toggleButton@getSupertip (section 2.2.43); toggleButton@getSupertip (section 2.2.42); toggleButton@getSupertip (section 2.2.44); toggleButton@getVisible (section 2.2.43); toggleButton@getVisible (section 2.2.42); toggleButton@getVisible (section 2.2.44); toggleButton@onAction (section 2.2.43); toggleButton@onAction (section 2.2.42); toggleButton@onAction (section 2.2.44) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_Delegate">
2. <xsd:restriction base="xsd:string">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_GalleryItemWidthHeight (Gallery Item Width or Height)

Specifies the width or height of gallery items, in pixels.

This simple type's contents are a restriction of the XML schema positiveInteger datatype.

This simple type also specifies the following restrictions:

* This simple type has a minimum value of greater than or equal to 1.
* This simple type has a maximum value of less than or equal to 4096.

| Referenced By |
| --- |
| gallery@itemHeight (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@itemHeight (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@itemWidth (section 2.2.21); gallery@itemWidth (section 2.2.22) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_GalleryItemWidthHeight">
2. <xsd:restriction base="xsd:positiveInteger">
3. <xsd:minInclusive value="1"/>
4. <xsd:maxInclusive value="4096"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_GalleryRowColumnCount (Gallery Row or Column Count)

Specifies the count of rows or columns in a **gallery** control.

This simple type's contents are a restriction of the XML schema positiveInteger datatype.

This simple type also specifies the following restrictions:

* This simple type has a minimum value of greater than or equal to 1.
* This simple type has a maximum value of less than or equal to 1024.

| Referenced By |
| --- |
| gallery@columns (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@columns (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@rows (section 2.2.21); gallery@rows (section 2.2.22) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_GalleryRowColumnCount">
2. <xsd:restriction base="xsd:positiveInteger">
3. <xsd:minInclusive value="1"/>
4. <xsd:maxInclusive value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_ID (Control ID)

Specifies the identifier of a built-in control. The format of this **string** is defined by per application by the Custom UI Control identifier Tables, as specified in section [3](#Section_c8d2792a413349f592a6f146f1cc5010).

This simple type's contents are a restriction of the XML schema **NCName** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| box@insertAfterMso (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); box@insertBeforeMso (section 2.2.1); button@idMso (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@idMso (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@idMso (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); button@imageMso (section 2.2.2); button@imageMso (section 2.2.3); button@imageMso (section 2.2.4); button@insertAfterMso (section 2.2.2); button@insertAfterMso (section 2.2.3); button@insertAfterMso (section 2.2.4); button@insertBeforeMso (section 2.2.2); button@insertBeforeMso (section 2.2.3); button@insertBeforeMso (section 2.2.4); buttonGroup@insertAfterMso (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); buttonGroup@insertBeforeMso (section 2.2.5); checkBox@idMso (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); checkBox@imageMso (section 2.2.6); checkBox@insertAfterMso (section 2.2.6); checkBox@insertBeforeMso (section 2.2.6); comboBox@idMso (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); comboBox@imageMso (section 2.2.7); comboBox@insertAfterMso (section 2.2.7); comboBox@insertBeforeMso (section 2.2.7); command@idMso (section [2.2.8](#Section_21316865fbce4a3fa9ffa8277cce5f2d)); control@id (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@idMso (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@idMso (section 2.2.13); control@idMso (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); control@imageMso (section 2.2.12); control@imageMso (section 2.2.13); control@imageMso (section 2.2.11); control@insertAfterMso (section 2.2.12); control@insertAfterMso (section 2.2.13); control@insertAfterMso (section 2.2.11); control@insertBeforeMso (section 2.2.12); control@insertBeforeMso (section 2.2.13); control@insertBeforeMso (section 2.2.11); dropDown@idMso (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dropDown@imageMso (section 2.2.17); dropDown@insertAfterMso (section 2.2.17); dropDown@insertBeforeMso (section 2.2.17); dynamicMenu@idMso (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@idMso (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); dynamicMenu@imageMso (section 2.2.19); dynamicMenu@imageMso (section 2.2.18); dynamicMenu@insertAfterMso (section 2.2.19); dynamicMenu@insertAfterMso (section 2.2.18); dynamicMenu@insertBeforeMso (section 2.2.19); dynamicMenu@insertBeforeMso (section 2.2.18); editBox@idMso (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); editBox@imageMso (section 2.2.20); editBox@insertAfterMso (section 2.2.20); editBox@insertBeforeMso (section 2.2.20); gallery@idMso (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@idMso (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@imageMso (section 2.2.21); gallery@imageMso (section 2.2.22); gallery@insertAfterMso (section 2.2.21); gallery@insertAfterMso (section 2.2.22); gallery@insertBeforeMso (section 2.2.21); gallery@insertBeforeMso (section 2.2.22); group@idMso (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); group@imageMso (section 2.2.23); group@insertAfterMso (section 2.2.23); group@insertBeforeMso (section 2.2.23); item@imageMso (section [2.2.24](#Section_6417edcc85b847f781fd56918e8df537)); labelControl@idMso (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); labelControl@imageMso (section 2.2.25); labelControl@insertAfterMso (section 2.2.25); labelControl@insertBeforeMso (section 2.2.25); menu@idMso (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@idMso (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@idMso (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); menu@imageMso (section 2.2.28); menu@imageMso (section 2.2.26); menu@imageMso (section 2.2.27); menu@insertAfterMso (section 2.2.28); menu@insertAfterMso (section 2.2.26); menu@insertAfterMso (section 2.2.27); menu@insertBeforeMso (section 2.2.28); menu@insertBeforeMso (section 2.2.26); menu@insertBeforeMso (section 2.2.27); menuSeparator@insertAfterMso (section [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686)); menuSeparator@insertBeforeMso (section 2.2.30); separator@insertAfterMso (section [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b)); separator@insertBeforeMso (section 2.2.34); splitButton@idMso (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@idMso (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@idMso (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); splitButton@imageMso (section 2.2.38); splitButton@imageMso (section 2.2.36); splitButton@imageMso (section 2.2.37); splitButton@insertAfterMso (section 2.2.38); splitButton@insertAfterMso (section 2.2.36); splitButton@insertAfterMso (section 2.2.37); splitButton@insertBeforeMso (section 2.2.38); splitButton@insertBeforeMso (section 2.2.36); splitButton@insertBeforeMso (section 2.2.37); tab@idMso (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); tab@insertAfterMso (section 2.2.39); tab@insertBeforeMso (section 2.2.39); tabSet@idMso (section [2.2.41](#Section_90a00968474745128601b61440747033)); toggleButton@idMso (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@idMso (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@idMso (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)); toggleButton@imageMso (section 2.2.43); toggleButton@imageMso (section 2.2.42); toggleButton@imageMso (section 2.2.44); toggleButton@insertAfterMso (section 2.2.43); toggleButton@insertAfterMso (section 2.2.42); toggleButton@insertAfterMso (section 2.2.44); toggleButton@insertBeforeMso (section 2.2.43); toggleButton@insertBeforeMso (section 2.2.42); toggleButton@insertBeforeMso (section 2.2.44) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_ID">
2. <xsd:restriction base="xsd:NCName">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_ItemSize (Menu Item Size)

Specifies the size of the child controls in a **menu** control.

This simple type's contents are a restriction of the XML schema **string** datatype.

The following are possible **enumeration** values for this type:

| Enumeration Value | Description |
| --- | --- |
| large (Large) | Specifies that the child controls have large sizes. |
| normal (Normal) | Specifies that the child controls have normal sizes. |

| Referenced By |
| --- |
| menu@itemSize (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@itemSize (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@itemSize (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); menu@itemSize (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_ItemSize">
2. <xsd:restriction base="xsd:string">
3. <xsd:enumeration value="normal"/>
4. <xsd:enumeration value="large"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_Keytip (Keytip)

Specifies a [**KeyTip**](#gt_f9905654-b489-43a0-acc5-d67ecd4ba9f4) string.

This simple type's contents are a restriction of the XML schema **token** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 3 characters.

| Referenced By |
| --- |
| button@keytip (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@keytip (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@keytip (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); checkBox@keytip (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); comboBox@keytip (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); control@keytip (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@keytip (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@keytip (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); dropDown@keytip (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dynamicMenu@keytip (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@keytip (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); editBox@keytip (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); gallery@keytip (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@keytip (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); group@keytip (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); labelControl@keytip (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); menu@keytip (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@keytip (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@keytip (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); splitButton@keytip (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@keytip (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@keytip (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); tab@keytip (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); toggleButton@keytip (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@keytip (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@keytip (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_Keytip">
2. <xsd:restriction base="xsd:token">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="3"/>
5. <xsd:whiteSpace value="collapse"/>
6. </xsd:restriction>
7. </xsd:simpleType>

### ST\_LongString (Long String)

Specifies a string that can have an extended length.

This simple type's contents are a restriction of the XML schema **string** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 4096 characters.

| Referenced By |
| --- |
| button@description (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@description (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@description (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); checkBox@description (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); control@description (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@description (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); dynamicMenu@description (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@description (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); gallery@description (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@description (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); menu@description (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@description (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); toggleButton@description (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@description (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@description (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_LongString">
2. <xsd:restriction base="xsd:string">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="4096"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_QID (Qualified Control ID)

Specifies a control identifier that is qualified by an XML namespace prefix. The prefix determines which namespace to which the control belongs.

If the namespace is equal to the Custom UI namespace, the qualified identifier references the application's built-in control set.

For example, consider the following XML fragment:

1. <customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
2. xmlns:mso="http://schemas.microsoft.com/office/2006/01/customui">
3. <ribbon>
4. <tabs>
5. <tab idQ="mso:TabHome" visible="false" />
6. </tabs>
7. </ribbon>
8. </customUI>

In this example the **mso** namespace prefix is set to the Custom UI namespace, so names qualified with **mso** refer to built-in controls. Thus, the use of the **idQ** attribute on the **tab** element is equivalent to using the **idMso** attribute, as follows:

1. <tab idMso="TabHome" visible="false" />

If the prefix is set to any other value, qualified identifiers reference controls in a unique custom namespace. If multiple Custom UI documents refer to controls in the same namespace, they can share common containers.

For example, consider the following XML fragment:

1. <customUI
2. xmlns="http://schemas.microsoft.com/office/2006/01/customui"
3. xmlns:ex="http://www.example.com">
4. <ribbon>
5. <tabs>
6. <tab idQ="ex:OtherTab" label="Shared Tab">
7. <group id="MyGroup" label="My Group">
8. …
9. </group>
10. </tab>
11. </tabs>
12. </ribbon>
13. </customUI>

In this case, **ex** is an XML namespace prefix for the namespace http://www.example.com. This XML fragment refers to a **tab** in that namespace with an identifier of "OtherTab". If that tab cannot be found, it is created. A new group belonging to this file is added to the **tab**.

This simple type's contents are a restriction of the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **QName** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| box@idQ (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); box@insertAfterQ (section 2.2.1); box@insertBeforeQ (section 2.2.1); button@idQ (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@idQ (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@idQ (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); button@insertAfterQ (section 2.2.2); button@insertAfterQ (section 2.2.3); button@insertAfterQ (section 2.2.4); button@insertBeforeQ (section 2.2.2); button@insertBeforeQ (section 2.2.3); button@insertBeforeQ (section 2.2.4); buttonGroup@idQ (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); buttonGroup@insertAfterQ (section 2.2.5); buttonGroup@insertBeforeQ (section 2.2.5); checkBox@idQ (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); checkBox@insertAfterQ (section 2.2.6); checkBox@insertBeforeQ (section 2.2.6); comboBox@idQ (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); comboBox@insertAfterQ (section 2.2.7); comboBox@insertBeforeQ (section 2.2.7); control@idQ (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@idQ (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@idQ (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); control@insertAfterQ (section 2.2.12); control@insertAfterQ (section 2.2.13); control@insertAfterQ (section 2.2.11); control@insertBeforeQ (section 2.2.12); control@insertBeforeQ (section 2.2.13); control@insertBeforeQ (section 2.2.11); dropDown@idQ (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dropDown@insertAfterQ (section 2.2.17); dropDown@insertBeforeQ (section 2.2.17); dynamicMenu@idQ (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@idQ (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); dynamicMenu@insertAfterQ (section 2.2.19); dynamicMenu@insertAfterQ (section 2.2.18); dynamicMenu@insertBeforeQ (section 2.2.19); dynamicMenu@insertBeforeQ (section 2.2.18); editBox@idQ (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); editBox@insertAfterQ (section 2.2.20); editBox@insertBeforeQ (section 2.2.20); gallery@idQ (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@idQ (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@insertAfterQ (section 2.2.21); gallery@insertAfterQ (section 2.2.22); gallery@insertBeforeQ (section 2.2.21); gallery@insertBeforeQ (section 2.2.22); group@idQ (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); group@insertAfterQ (section 2.2.23); group@insertBeforeQ (section 2.2.23); labelControl@idQ (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); labelControl@insertAfterQ (section 2.2.25); labelControl@insertBeforeQ (section 2.2.25); menu@idQ (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@idQ (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@idQ (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); menu@insertAfterQ (section 2.2.28); menu@insertAfterQ (section 2.2.26); menu@insertAfterQ (section 2.2.27); menu@insertBeforeQ (section 2.2.28); menu@insertBeforeQ (section 2.2.26); menu@insertBeforeQ (section 2.2.27); menuSeparator@idQ (section [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686)); menuSeparator@insertAfterQ (section 2.2.30); menuSeparator@insertBeforeQ (section 2.2.30); separator@idQ (section [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b)); separator@insertAfterQ (section 2.2.34); separator@insertBeforeQ (section 2.2.34); splitButton@idQ (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@idQ (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@idQ (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); splitButton@insertAfterQ (section 2.2.38); splitButton@insertAfterQ (section 2.2.36); splitButton@insertAfterQ (section 2.2.37); splitButton@insertBeforeQ (section 2.2.38); splitButton@insertBeforeQ (section 2.2.36); splitButton@insertBeforeQ (section 2.2.37); tab@idQ (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); tab@insertAfterQ (section 2.2.39); tab@insertBeforeQ (section 2.2.39); toggleButton@idQ (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@idQ (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@idQ (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)); toggleButton@insertAfterQ (section 2.2.43); toggleButton@insertAfterQ (section 2.2.42); toggleButton@insertAfterQ (section 2.2.44); toggleButton@insertBeforeQ (section 2.2.43); toggleButton@insertBeforeQ (section 2.2.42); toggleButton@insertBeforeQ (section 2.2.44) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_QID">
2. <xsd:restriction base="xsd:QName">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_Size (Control Size)

Specifies the size of a control.

This simple type's contents are a restriction of the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **string** datatype.

The following are possible **enumeration** values for this type:

| Enumeration Value | Description |
| --- | --- |
| large (Large Control Size) | Specifies the large control size. |
| normal (Normal Control Size) | Specifies the normal control size. |

| Referenced By |
| --- |
| button@size (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); control@size (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@size (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); dynamicMenu@size (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); gallery@size (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); menu@size (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); splitButton@size (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); toggleButton@size (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_Size">
2. <xsd:restriction base="xsd:string">
3. <xsd:enumeration value="normal"/>
4. <xsd:enumeration value="large"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_String (Short String)

Specifies a string with a limited length.

This simple type's contents are a restriction of the XML schema **string** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| button@label (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@label (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@label (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); button@screentip (section 2.2.2); button@screentip (section 2.2.3); button@screentip (section 2.2.4); button@supertip (section 2.2.2); button@supertip (section 2.2.3); button@supertip (section 2.2.4); button@tag (section 2.2.2); button@tag (section 2.2.3); button@tag (section 2.2.4); checkBox@label (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); checkBox@screentip (section 2.2.6); checkBox@supertip (section 2.2.6); checkBox@tag (section 2.2.6); comboBox@label (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); comboBox@screentip (section 2.2.7); comboBox@sizeString (section 2.2.7); comboBox@supertip (section 2.2.7); comboBox@tag (section 2.2.7); control@label (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@label (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@label (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); control@screentip (section 2.2.12); control@screentip (section 2.2.13); control@screentip (section 2.2.11); control@supertip (section 2.2.12); control@supertip (section 2.2.13); control@supertip (section 2.2.11); control@tag (section 2.2.12); control@tag (section 2.2.11); dropDown@label (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dropDown@screentip (section 2.2.17); dropDown@sizeString (section 2.2.17); dropDown@supertip (section 2.2.17); dropDown@tag (section 2.2.17); dynamicMenu@label (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@label (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); dynamicMenu@screentip (section 2.2.19); dynamicMenu@screentip (section 2.2.18); dynamicMenu@supertip (section 2.2.19); dynamicMenu@supertip (section 2.2.18); dynamicMenu@tag (section 2.2.19); dynamicMenu@tag (section 2.2.18); editBox@label (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); editBox@screentip (section 2.2.20); editBox@sizeString (section 2.2.20); editBox@supertip (section 2.2.20); editBox@tag (section 2.2.20); gallery@label (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@label (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); gallery@screentip (section 2.2.21); gallery@screentip (section 2.2.22); gallery@sizeString (section 2.2.21); gallery@sizeString (section 2.2.22); gallery@supertip (section 2.2.21); gallery@supertip (section 2.2.22); gallery@tag (section 2.2.21); gallery@tag (section 2.2.22); group@label (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); group@screentip (section 2.2.23); group@supertip (section 2.2.23); group@tag (section 2.2.23); item@label (section [2.2.24](#Section_6417edcc85b847f781fd56918e8df537)); item@screentip (section 2.2.24); item@supertip (section 2.2.24); labelControl@label (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); labelControl@screentip (section 2.2.25); labelControl@supertip (section 2.2.25); labelControl@tag (section 2.2.25); menu@label (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@label (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@label (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); menu@screentip (section 2.2.28); menu@screentip (section 2.2.26); menu@screentip (section 2.2.27); menu@supertip (section 2.2.28); menu@supertip (section 2.2.26); menu@supertip (section 2.2.27); menu@tag (section 2.2.28); menu@tag (section 2.2.26); menu@tag (section 2.2.27); menu@title (section [2.2.29](#Section_e662c00135584511a56507a3e32021e3)); menu@title (section 2.2.27); menuSeparator@title (section [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686)); splitButton@label (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@label (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@label (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); splitButton@screentip (section 2.2.38); splitButton@screentip (section 2.2.36); splitButton@screentip (section 2.2.37); splitButton@supertip (section 2.2.38); splitButton@supertip (section 2.2.36); splitButton@supertip (section 2.2.37); splitButton@tag (section 2.2.38); splitButton@tag (section 2.2.36); splitButton@tag (section 2.2.37); tab@label (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); tab@tag (section 2.2.39); toggleButton@label (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@label (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@label (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)); toggleButton@screentip (section 2.2.43); toggleButton@screentip (section 2.2.42); toggleButton@screentip (section 2.2.44); toggleButton@supertip (section 2.2.43); toggleButton@supertip (section 2.2.42); toggleButton@supertip (section 2.2.44); toggleButton@tag (section 2.2.43); toggleButton@tag (section 2.2.42); toggleButton@tag (section 2.2.44) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_String">
2. <xsd:restriction base="xsd:string">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_StringLength (String Length)

Specifies the length of a string, in characters.

This simple type's contents are a restriction of the XML schema **positiveInteger** datatype.

This simple type also specifies the following restrictions:

* This simple type has a minimum value of greater than or equal to 1.
* This simple type has a maximum value of less than or equal to 1024.

| Referenced By |
| --- |
| comboBox@maxLength (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); editBox@maxLength (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_StringLength">
2. <xsd:restriction base="xsd:positiveInteger">
3. <xsd:minInclusive value="1"/>
4. <xsd:maxInclusive value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_UniqueID (Custom Control ID)

Specifies a custom control identifier.

This simple type's contents are a restriction of the XML schema **identifier** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 character.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| box@id (section [2.2.1](#Section_fb660d68051e4a86ab97e0d93b32817e)); button@id (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@id (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@id (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); buttonGroup@id (section [2.2.5](#Section_d9f6097943e740e6bd211f9693600c21)); checkBox@id (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); comboBox@id (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); control@id (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@id (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); dropDown@id (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dynamicMenu@id (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@id (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); editBox@id (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); gallery@id (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@id (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); group@id (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); item@id (section [2.2.24](#Section_6417edcc85b847f781fd56918e8df537)); labelControl@id (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); menu@id (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@id (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@id (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); menuSeparator@id (section [2.2.30](#Section_00c26e34b4a84c97a4f6d547a555a686)); separator@id (section [2.2.34](#Section_21312cb8be0f412c8184acd533a1410b)); splitButton@id (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@id (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@id (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); tab@id (section [2.2.39](#Section_141f881ca5a4473f944955d3d36579ed)); toggleButton@id (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@id (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@id (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_UniqueID">
2. <xsd:restriction base="xsd:identifier">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

### ST\_Uri (Image Relationship ID)

Specifies the relationship identifier of a part that is the target of a relationship from the containing Custom UI document.

The target part is an image part type, as specified in [[ECMA-376]](https://go.microsoft.com/fwlink/?LinkId=200054) Part 1 section 15.2.13.

This simple type's contents are a restriction of the [**XML schema**](#gt_bd0ce6f9-c350-4900-827e-951265294067) **string** datatype.

This simple type also specifies the following restrictions:

* This simple type's contents have a minimum length of 1 characters.
* This simple type's contents have a maximum length of 1024 characters.

| Referenced By |
| --- |
| button@image (section [2.2.2](#Section_846e8fb607d3460b816bbcfae841c95b)); button@image (section [2.2.3](#Section_2e7b9955ff774a2f9d93401f0541e8fd)); button@image (section [2.2.4](#Section_2257e779b94944cb8ec0719a46421f3e)); checkBox@image (section [2.2.6](#Section_d189aedcd7634295bc2b07746447dfde)); comboBox@image (section [2.2.7](#Section_3caf6b1834cb4352bd3177a26980c00e)); control@image (section [2.2.12](#Section_f2bf1b5d049246a488f7dc1a9528f755)); control@image (section [2.2.13](#Section_14909d674b9f4796badae79edaedcdf2)); control@image (section [2.2.11](#Section_ada8edc4b469410e8bd05d667ffe7dbf)); dropDown@image (section [2.2.17](#Section_700e4451870640c58d7b896e4ae21b69)); dynamicMenu@image (section [2.2.19](#Section_26d7db807ea442d09a063f63365043c7)); dynamicMenu@image (section [2.2.18](#Section_fd0825c70f294038861754af0dec8c7d)); editBox@image (section [2.2.20](#Section_7b552756a970466db7c2ee5bbe171e34)); gallery@image (section [2.2.21](#Section_017f9f94700f4846ba19c695ad32929f)); gallery@image (section [2.2.22](#Section_7bfba338481f453e97670e0855c8f029)); group@image (section [2.2.23](#Section_190f51e63a3c4cf8affbcec9b906aae4)); item@image (section [2.2.24](#Section_6417edcc85b847f781fd56918e8df537)); labelControl@image (section [2.2.25](#Section_965555a293db48a6a6bf5dd08a13f6bb)); menu@image (section [2.2.28](#Section_0243b261903e43d98e8b43f66ea0bcaf)); menu@image (section [2.2.26](#Section_1590adf87f2a4ec8944f9c4329e7ff98)); menu@image (section [2.2.27](#Section_f785dc2b045b4b86b5424a10d264dace)); splitButton@image (section [2.2.38](#Section_aa41c698c7e84486b15fb73bedbf2be8)); splitButton@image (section [2.2.36](#Section_6477dffcc8f24fc9815b637dabf3911d)); splitButton@image (section [2.2.37](#Section_1c6721206402408b96dde98650c12da5)); toggleButton@image (section [2.2.43](#Section_ec42bfd0149c495b895c3bc708b8a149)); toggleButton@image (section [2.2.42](#Section_69b127857fce431591c145aa114726ea)); toggleButton@image (section [2.2.44](#Section_539b9c44822d45cb8abe66b65a8bc14c)) |

The following XML schema fragment defines the contents of this simple type:

1. <xsd:simpleType name="ST\_Uri">
2. <xsd:restriction base="xsd:string">
3. <xsd:minLength value="1"/>
4. <xsd:maxLength value="1024"/>
5. </xsd:restriction>
6. </xsd:simpleType>

# Appendix A: Custom UI Control ID Tables

## idMso Tables

### Word 2007

| idMso | Control Type | Label |
| --- | --- | --- |
| Spelling | button | Spelling... |
| FileSave | button | Save |
| FilePrint | button | Print |
| ZoomOnePage | button | One Page |
| ZoomPageWidth | button | Page Width |
| Zoom100 | button | Zoom 100% |
| ColumnsDialog | button | More Columns... |
| Numbering | toggleButton | Numbering |
| Bullets | toggleButton | Bullets |
| PageOrientationPortraitLandscape | button | Portrait/Landscape |
| OutdentClassic | button | Decrease Indent |
| IndentClassic | button | Increase Indent |
| DrawingInsert | button | Insert Drawing |
| ChartInsert | button | Chart... |
| FileNew | button | New |
| Copy | button | Copy |
| Cut | button | Cut |
| Paste | button | Paste |
| FileOpen | button | Open |
| EnvelopesAndLabelsDialog | button | Envelopes... |
| Superscript | toggleButton | Superscript |
| Subscript | toggleButton | Subscript |
| UnderlineDouble | toggleButton | Double Underline |
| UnderlineWords | button | Word Underline |
| FontSizeIncreaseWord | button | Grow Font |
| FontSizeDecreaseWord | button | Shrink Font |
| FileClose | button | Close |
| TableAutoFormat | button | Table AutoFormat... |
| FormatPainter | toggleButton | Format Painter |
| FilePrintPreview | toggleButton | Print Preview |
| PasteApplyStyle | button | Apply Style |
| Bold | toggleButton | Bold |
| Italic | toggleButton | Italic |
| Underline | toggleButton | Underline |
| ParagraphMarks | toggleButton | Show All |
| AlignLeft | toggleButton | Align Left |
| AlignRight | toggleButton | Align Right |
| AlignCenter | toggleButton | Center |
| AlignJustify | toggleButton | Justify |
| HeaderFooterPageNumberInsert | menu | Page Number |
| Undo | gallery | Undo |
| Redo | gallery | Redo |
| OutlinePromote | button | Promote |
| OutlineDemote | button | Demote |
| OutlineMoveUp | button | Move Up |
| OutlineMoveDown | button | Move Down |
| OutlineDemoteToBodyText | button | Demote to Body Text |
| OutlineExpand | button | Expand |
| OutlineCollapse | button | Collapse |
| TextBoxInsert | button | Text Box |
| FileFind | button | Find File... |
| FindDialog | button | Find... |
| TableExcelSpreadsheetInsert | button | Excel Spreadsheet |
| AutoFormat | button | AutoFormat... |
| BorderInside | toggleButton | Inside Borders |
| BorderOutside | toggleButton | Outside Borders |
| BorderNone | toggleButton | No Border |
| MailMergeGoToFirstRecord | button | First |
| MailMergeGoToPreviousRecord | button | Previous |
| MailMergeGoToNextRecord | button | Next |
| MailMergeGotToLastRecord | button | Last |
| MailMergeMergeToDocument | button | Edit Individual Documents... |
| MailMergeMergeToPrinter | button | Print Documents... |
| MailMergeAutoCheckForErrors | button | Auto Check for Errors... |
| DataFormSource | button | Data Form |
| MailMergeResultsPreview | toggleButton | Preview Results |
| ObjectsGroup | button | Group |
| ObjectsUngroup | button | Ungroup |
| ObjectBringToFront | button | Bring to Front |
| ObjectSendToBack | button | Send to Back |
| ObjectBringForward | button | Bring Forward |
| ObjectSendBackward | button | Send Backward |
| Magnifier | checkBox | Magnifier |
| PrintPreviewShrinkOnePage | button | Shrink One Page |
| ViewFullScreenView | button | Full Screen |
| VoiceInsert | button | Voice Comment |
| ObjectsSelect | toggleButton | Select Objects |
| TableFind | button | Find |
| MacroRecord | button | Record Macro... |
| MacroRecorderPause | button | Pause Recording |
| MacroPlay | button | Macros |
| ShapeFreeform | toggleButton | Freeform |
| ObjectEditPoints | toggleButton | Edit Points |
| CalloutOptions | button | Callout Options |
| DataFormAddRecord | button | Add |
| DataFormDeleteRecord | button | Delete |
| FieldsUpdate | button | Update |
| DatabaseInsert | button | Insert Database |
| GridSettings | button | Grid Settings... |
| WordPicture | button | Word Picture |
| FormControlEditBox | button | Edit Box |
| FormControlCheckBox | button | Check Box |
| FormControlComboBox | button | Combo Box |
| PropertySheet | button | Property Sheet |
| FieldShading | toggleButton | Show Field Shading |
| ViewDraftView | toggleButton | Draft |
| Lock | toggleButton | Lock |
| AutoSum | button | Sum |
| MasterDocumentShow | toggleButton | Show Document |
| MasterDocumentCreateSubdocument | button | Create |
| MasterDocumentUnlinkSubdocument | button | Unlink |
| MasterDocumentInsertSubdocument | button | Insert... |
| MasterDocumentSplitSubdocuments | button | Split |
| MasterDocumentMergeSubdocuments | button | Merge |
| MasterDocumentLockSubdocument | toggleButton | Lock Document |
| HeaderOrFooterShow | button | Show Header/Footer |
| HeaderFooterPreviousSection | button | Previous Section |
| HeaderFooterNextSection | button | Next Section |
| AlignDialog | button | Align |
| MailMergeDocument | button | Mail Merge Document |
| MergeOptions | button | Merge... |
| MailMergeHelper | button | Mail Merge... |
| PageSetupDialog | button | Page Setup... |
| BodyTextHide | button | Hide Body Text |
| HeaderFooterLinkToPrevious | toggleButton | Link to Previous |
| OutlineShowFirstLineOnly | checkBox | Show First Line Only |
| OutlineShowTextFormatting | checkBox | Show Text Formatting |
| FontDialog | button | Font... |
| StylesDialogClassic | button | Edit Cell Styles |
| FootnoteInsert | button | Insert Footnote |
| MicrosoftExcel | button | Microsoft Excel |
| MicrosoftAccess | button | Microsoft Access |
| MicrosoftPowerPoint | button | Microsoft Office PowerPoint |
| MicrosoftPublisher | button | Microsoft Publisher |
| MicrosoftProject | button | Microsoft Project |
| ViewPrintLayoutView | toggleButton | Print Layout |
| FieldCodes | toggleButton | View Field Codes |
| DropCapOptionsDialog | button | Drop Cap Options... |
| Strikethrough | toggleButton | Strikethrough |
| TextSmallCaps | toggleButton | Small Caps |
| CellsDelete | button | Delete Cells... |
| TableRowsDelete | button | Delete Rows |
| TableColumnsDelete | button | Delete Columns |
| CellsInsertDialog | button | Insert Cells... |
| TableRowsInsertWord | button | Insert Rows |
| WindowsArrangeAll | button | Arrange All |
| MarginsAdjust | button | Adjust Margins |
| ViewGridlinesWord | checkBox | View Gridlines |
| SubdocumentOpen | button | Open Subdocument |
| WindowSplit | button | Split |
| WindowNew | button | New Window |
| ReviewAcceptOrRejectChangeDialog | button | Accept/Reject Changes |
| TextAllCaps | toggleButton | All Caps |
| PictureDisassemble | button | Disassemble Picture |
| ChangeCaseDialogClassic | button | Change Case... |
| FontSizeDecrease1Point | button | Shrink Font 1 Pt |
| FontSizeIncrease1Point | button | Grow Font 1 Pt |
| Repaginate | button | Repaginate |
| ReplaceDialog | button | Replace... |
| StartOfLine | button | Start of Line |
| EndOfLine | button | End of Line |
| PagePrevious | button | Previous Page |
| PageNext | button | Next Page |
| StartOfDocument | button | Start of Document |
| EndOfDocument | button | End of Document |
| Grammar | button | Grammar... |
| FileCloseOrCloseAll | button | Close |
| TextToOrFromTable | button | Text to/from Table |
| TableRowsOrColumnsOrCellsInsert | button | Insert Table |
| TableRowsOrColumnsOrCellsDelete | button | Delete Rows/Columns/Cells |
| RedoOrRepeat | button | Redo |
| ProtectOrUnprotectDocument | button | Protect Document |
| FrameInsertOrFormat | button | Insert Frame |
| ObjectsRegroup | button | Regroup |
| AutoFormatChange | button | Tip Wizard 6 |
| AddressBook | button | Address Book... |
| Reply | button | Reply |
| ReplyAll | button | Reply to All |
| Forward | button | Forward |
| MailMove | button | Move Mail |
| MailDelete | button | Delete Mail |
| MessagePrevious | button | Previous Item |
| MessageNext | button | Next Item |
| MailSelectNames | button | Select Names... |
| AsianLayoutCharacterScaling | menu | Character Scaling |
| ShapeScribble | toggleButton | Scribble |
| PrintSetupDialog | button | Print Setup... |
| RowHeight | button | Row Height... |
| ColumnWidth | button | Column Width... |
| OleObjectctInsert | button | Object... |
| Cancel | button | Cancel |
| FindNext | button | Find Next |
| PasteDuplicate | button | Duplicate |
| ClipArtInsert | toggleButton | Clip Art... |
| ParagraphSpacingIncrease | button | Increase Paragraph Spacing |
| ParagraphSpacingDecrease | button | Decrease Paragraph Spacing |
| OrganizationChartInsert | button | Organization Chart |
| CombineCharacters | toggleButton | Yoko-Gumi |
| DoubleStrikethrough | toggleButton | Double Strikethrough |
| PictureCrop | toggleButton | Crop |
| ViewOutlineView | toggleButton | Outline |
| FileCloseAll | button | Close All |
| FileSaveAs | button | Save As |
| SaveAll | button | Save All |
| AdvancedFileProperties | button | View Document Properties... |
| DocumentTemplate | button | Document Template |
| CopyAsPicture | button | Copy as Picture... |
| PasteSpecialDialog | button | Paste Special... |
| SelectAll | button | Select All |
| GoTo | button | Go To... |
| BookmarkInsert | button | Bookmark... |
| FileLinksToFiles | button | Edit Links to Files |
| ViewOnlineLayoutViewClassic | button | Online Layout |
| FootnotesEndnotesShow | button | Show Notes |
| BreakInsertDialog | button | Break |
| DateAndTimeInsert | button | Date & Time... |
| NumberInsert | button | Number... |
| FieldInsert | button | Field... |
| FormField | button | Form Field... |
| CaptionInsert | button | Insert Caption... |
| CrossReferenceInsert | button | Cross-reference... |
| IndexAndTables | button | Index and Tables |
| TextFromFileInsert | button | Text from File... |
| ParagraphDialog | button | Paragraph... |
| TabsDialog | button | Tabs... |
| BordersShadingDialog | button | Borders and Shading... |
| TextDirectionOptionsDialog | button | Text Direction Options... |
| BulletsAndNumberingBulletsDialog | button | Bullets and Numbering... |
| StyleGalleryDialog | button | Style Gallery... |
| FrameDialog | button | Frame... |
| SetLanguage | button | Set Language... |
| WordCount | button | Word Count... |
| AutoCorrect | button | AutoCorrect Options... |
| EnvelopesAndLabels | button | Envelope & Label Wizard |
| LabelsDialog | button | Labels... |
| MergeCells | button | Merge Cells |
| SplitCells | button | Split Cells... |
| TableRowSelect | button | Select Row |
| TableColumnSelect | button | Select Column |
| TableSelect | button | Select Table |
| TableCellHeightWidth | button | Cell Height and Width... |
| TableRepeatHeaderRows | toggleButton | Repeat Header Rows |
| ConvertTextToTable | button | Convert Text to Table... |
| TableFormulaDialog | button | Formula... |
| TableSplitTable | button | Split Table |
| ShowClipboard | button | Office Clipboard... |
| NumberingSkip | button | Skip Numbering |
| KeyboardCustomization | button | Customize Keyboard... |
| ShowAllHeadings | button | All |
| ImeDictionaryUpdate | button | Update IME Dictionary... |
| OutlookTaskCreate | button | Create Microsoft Office Outlook Task |
| WindowMinimize | button | Minimize |
| WindowRestore | button | Restore |
| WindowClose | button | Close |
| WindowMove | button | Move |
| WindowSize | button | Size |
| WindowNext | button | Next Window |
| ClearFormats | button | Clear Formats |
| OK | button | OK |
| ClosePane | button | Close |
| PrintPreviewClose | button | Close Print Preview |
| HeaderFooterClose | button | Close Header and Footer |
| ZoomDialog | button | Zoom... |
| About | button | About |
| SortDialogClassic | button | Sort... |
| ConvertTableToText | button | Convert to Text... |
| ExchangeFolder | button | Exchange Folder... |
| ChartEditDataSource | button | Select Data... |
| WindowMoreWindowsDialog | toggleButton | More Windows... |
| ObjectEditDialog | button | Object... |
| ObjectFormatDialog | button | Object... |
| AutoTextCreate | button | Create AutoText... |
| ContentsAndIndex | button | Contents and Index |
| Help | button | Help |
| FontColorMoreColorsDialog | button | More Colors... |
| WebGoBack | button | Back |
| WebGoForward | button | Forward |
| AddToFavorites | button | Add to Favorites... |
| BrowsePrevious | button | Previous |
| BrowseNext | button | Next |
| SmartArtInsert | button | SmartArt... |
| ShapeRerouteConnectors | toggleButton | Reroute Connectors |
| ObjectNudgeUp | button | Up |
| ObjectNudgeDown | button | Down |
| ObjectNudgeLeft | button | Left |
| ObjectNudgeRight | button | Right |
| ShapeCurve | toggleButton | Curve |
| ShapeStraightConnector | toggleButton | Straight Connector |
| ShapeElbowConnector | toggleButton | Elbow Connector |
| ObjectFillMoreColorsDialog | button | More Fill Colors... |
| ObjectBorderOutlineColorMoreColorsDialog | button | More Outline Colors... |
| OutlineLinePatternFill | button | Pattern... |
| LineStylesDialog | button | More Lines... |
| ArrowsMore | button | More Arrows... |
| WordArtVerticalText | toggleButton | Vertical Text |
| WordArtEvenTextHeightClassic | toggleButton | Even Height |
| ContrastMore | button | More Contrast |
| ContrastLess | button | Less Contrast |
| BrightnessMore | button | More Brightness |
| BrightnessLess | button | Less Brightness |
| ShadowNudgeUpClassic | button | Nudge Shadow Up |
| ShadowNudgeDownClassic | button | Nudge Shadow Down |
| ShadowNudgeLeftClassic | button | Nudge Shadow Left |
| ShadowNudgeRightClassic | button | Nudge Shadow Right |
| ObjectShadowColorMoreColorsDialog | button | More Shadow Colors... |
| \_3DEffectColorPickerMoreClassic | button | More 3-D Colors... |
| TextAlignLeft | toggleButton | Left Align |
| TextAlignCenter | toggleButton | Center |
| ShapeRectangle | toggleButton | Rectangle |
| ShapeRoundedRectangle | toggleButton | Rounded Rectangle |
| ShapeIsoscelesTriangle | toggleButton | Isosceles Triangle |
| ShapeOval | toggleButton | Oval |
| ShapeLeftBrace | toggleButton | Left Brace |
| ShapeRightBrace | toggleButton | Right Brace |
| ShapeArc | toggleButton | Arc |
| ShapeRightArrow | toggleButton | Right Arrow |
| ShapeDownArrow | toggleButton | Down Arrow |
| ShapeRoundedRectangularCallout | toggleButton | Rounded Rectangular Callout |
| ShapeStar | toggleButton | 5-Point Star |
| TextAlignRight | toggleButton | Right Align |
| TextAlignLetterJustify | toggleButton | Letter Justify |
| TextAlignWordJustify | toggleButton | Word Justify |
| TextAlignStretchJustify | toggleButton | Stretch Justify |
| WordArtSpacingVeryTight | toggleButton | Very Tight |
| WordArtSpacingTight | toggleButton | Tight |
| WordArtSpacingNormal | toggleButton | Normal |
| WordArtSpacingLoose | toggleButton | Loose |
| WordArtSpacingVeryLoose | toggleButton | Very Loose |
| WordArtSpacingKernCharacterPairs | toggleButton | Kern Character Pairs |
| PictureReset | button | Reset Picture |
| TextWrappingSquare | toggleButton | Square |
| TextWrappingTight | toggleButton | Tight |
| TextWrappingNoneClassic | toggleButton | None |
| TextWrappingEditWrapPoints | toggleButton | Edit Wrap Points |
| \_3DEffectsOnOffClassic | toggleButton | 3-D On/Off |
| \_3DTiltDownClassic | button | Tilt Down |
| \_3DTiltUpClassic | button | Tilt Up |
| \_3DTiltLeftClassic | button | Tilt Left |
| \_3DTiltRightClassic | button | Tilt Right |
| \_3DExtrusionPerspectiveClassic | toggleButton | Perspective |
| \_3DExtrusionParallelClassic | toggleButton | Parallel |
| \_3DLightingFlatClassic | toggleButton | Bright |
| \_3DLightingNormalClassic | toggleButton | Normal |
| \_3DLightingDimClassic | toggleButton | Dim |
| ObjectEditText | button | Edit Text |
| PictureFormatDialog | button | Picture... |
| ViewVisualBasicCode | button | View Code |
| DrawingNewClassic | button | New Drawing |
| WebOpenInNewWindow | button | Open in New Window |
| HyperlinkCopy | button | Copy Hyperlink |
| HyperlinkInsert | button | Hyperlink... |
| HyperlinkEdit | button | Edit Hyperlink... |
| HyperlinkSelect | button | Select Hyperlink |
| ReviewNewComment | button | New Comment |
| ReviewPreviousComment | button | Previous |
| ReviewNextComment | button | Next |
| ReviewDeleteComment | button | Delete |
| ReviewShowAllComments | button | Show All Comments |
| DesignMode | toggleButton | Design Mode |
| WordArtInsertDialogClassic | button | WordArt Gallery |
| FormFieldProperties | button | Properties |
| FullScreenViewClassic | button | Full Screen |
| AutoScroll | button | Auto Scroll |
| MasterDocumentExpandOrCollapseSubdocuments | toggleButton | Expand/Collapse Subdocuments |
| VisualBasic | button | Visual Basic |
| BordersAll | toggleButton | All Borders |
| AutoSummarize | button | Auto Summarize |
| ViewDocumentMap | checkBox | Document Map |
| ReviewAcceptChange | button | Accept Change |
| ReviewRejectChange | button | Reject Change |
| TableDrawBorderPenStyle | dropDown | Pen Style |
| AutoSummaryExitView | button | Close |
| Font | comboBox | Font: |
| WhoIs | button | Who Is... |
| FontSize | comboBox | Font Size: |
| StyleGalleryClassic | comboBox | Style: |
| ZoomClassic | button | Zoom: |
| DocumentLocation | comboBox | Address: |
| MessageHeaderToggle | button | Message Header |
| BorderInsideHorizontal | toggleButton | Inside Horizontal Border |
| BorderInsideVertical | toggleButton | Inside Vertical Border |
| BorderDiagonalDown | toggleButton | Diagonal Down Border |
| BorderDiagonalUp | toggleButton | Diagonal Up Border |
| TextDirectionLeftToRight | toggleButton | Left-to-Right |
| TextDirectionRightToLeft | toggleButton | Right-to-Left |
| ActiveXCheckBox | button | Check Box |
| ActiveXTextBox | button | Text Box |
| ActiveXButton | button | Command Button |
| ActiveXRadioButton | button | Option Button |
| ActiveXListBox | button | List Box |
| ActiveXComboBox | button | Combo Box |
| ActiveXToggleButton | button | Toggle Button |
| ActiveXSpinButton | button | Spin Button |
| ActiveXScrollBar | button | Scroll Bar |
| ActiveXLabel | button | Label |
| ShadowSemitransparentClassic | toggleButton | Semitransparent Shadow |
| OleConvert | button | Convert... |
| ReviewTrackChanges | toggleButton | Track Changes |
| ReviewHighlightChanges | button | Highlight Changes... |
| ReviewEditComment | button | Edit Comment |
| TableDrawTable | toggleButton | Draw Table |
| TableEraser | toggleButton | Eraser |
| TableCellAlignTop | toggleButton | Align Top |
| TableCellAlignCenterVertically | toggleButton | Center Vertically |
| TableCellAlignBottom | toggleButton | Align Bottom |
| TableColumnsDistribute | button | Distribute Columns |
| TableRowsDistribute | button | Distribute Rows |
| ActiveXFrame | button | Frame |
| ActiveXImage | button | Image |
| WordArtEditTextClassic | button | Edit Text... |
| TableInsertCellsDialog | button | Insert Cells... |
| Organizer | button | Organizer |
| ShadowOnOrOffClassic | toggleButton | Shadow On/Off |
| ObjectSetShapeDefaults | button | Set AutoShape Defaults |
| ThesaurusClassic | button | Thesaurus... |
| MacroRecorderStop | button | Stop Recording |
| FileSendAsAttachment | button | E-mail |
| AutoSummaryViewByHighlight | toggleButton | Highlight/Show Only Summary |
| MasterDocument | button | Master Document |
| SystemInformation | button | Microsoft System Info |
| Overtype | button | Overtype |
| ExtendSelection | button | Extend Selection |
| Spike | button | Spike |
| SpikeInsert | button | Insert Spike |
| ChangeCase | button | Change Case |
| MoveText | button | Move Text |
| CopyText | button | Copy Text |
| AutoTextInsert | button | Insert AutoText |
| WindowOtherPane | button | Other Pane |
| WindowPrevious | button | Previous Window |
| FieldNext | button | Next Field |
| FieldPrevious | button | Previous Field |
| TableColumnSelectWord | button | Column Select |
| FieldCharactersInsert | button | Insert Field Chars |
| ListNumFieldInsert | button | Insert ListNum Field |
| FieldsUnlink | button | Unlink Fields |
| FieldsLock | button | Lock Fields |
| FieldsUnlock | button | Unlock Fields |
| UpdateSource | button | Update Source |
| HangingIndent | button | Hanging Indent |
| UnHang | button | Un Hang |
| HideText | button | Hidden |
| FontSpacingNormal | button | Normal Font Spacing |
| FontPositionNormal | button | Normal Font Position |
| ParagraphWidowOrphanControl | button | Para Widow Orphan Control |
| ParagraphKeepLinesTogether | button | Para Keep Lines Together |
| ParagraphKeepWithNext | toggleButton | Para Keep With Next |
| BreakParagraphPageBreakBefore | button | Para Page Break Before |
| ParagraphSpaceBeforeNone | button | No Space Before |
| ParagraphSpaceBefore | button | Space Before |
| ParagraphSpaceAddOrRemoveBefore | button | Add/Remove Space Before |
| ParagraphReset | button | Reset Para |
| PreviousEdit | button | Previous Edit |
| NextEdit | button | Next Edit |
| SaveTemplate | button | Save Template |
| PagePreviousWord | button | Previous Page |
| PageNextWord | button | Next Page |
| ObjectNext | button | Next Object |
| ObjectPrevious | button | Previous Object |
| FileConfirmConversions | button | File Confirm Conversions |
| MailMergeRecepientsUseExistingList | button | Use Existing List... |
| MailMergeOpenHeaderSource | button | Mail Merge Open Header Source |
| MailMergeQueryOptions | button | Query Options |
| MailMergeRuleIfThenElse | button | Mail Merge Insert If |
| MailMergeRuleMergeRecordNumber | button | Mail Merge Insert Merge Rec |
| MailMergeRuleMergeSequenceNumber | button | Mail Merge Insert Merge Seq |
| MailMergeRuleNextRecord | button | Mail Merge Insert Next |
| MailMergeRuleNextRecordIf | button | Mail Merge Insert Next If |
| MailMergeRuleSkipRecordIf | button | Mail Merge Insert Skip If |
| MailMergeRuleFillIn | button | Mail Merge Insert Fill In |
| MailMergeRuleAsk | button | Mail Merge Insert Ask |
| MailMergeRuleSetBookmark | button | Mail Merge Insert Set |
| MailMergeReset | button | Mail Merge Reset |
| MailMergeCreateDataSource | button | Mail Merge Create Data Source |
| MailMergeCreateHeaderSource | button | Mail Merge Create Header Source |
| GoToPreviousSection | button | Go To Previous Section |
| GoToNextSection | button | Go To Next Section |
| GoToPreviousPage | button | Go To Previous Page |
| GoToNextPage | button | Go To Next Page |
| FootnotePreviousWord | button | Previous Footnote |
| FootnoteNextWord | button | Next Footnote |
| EndnotePreviousWord | button | Previous Endnote |
| EndnoteNextWord | button | Next Endnote |
| ObjectActivate | button | Activate Object |
| TableAutoFormatUpdate | button | Table Update AutoFormat |
| DraftViewClassic | button | View Draft |
| NormalViewHeaderArea | button | Normal View Header Area |
| SectionBreakInsert | button | Insert Section Break |
| EndnoteInsertWord | button | Insert Endnote |
| FootnotesConvertAll | button | Edit Convert All Footnotes |
| EndnotesConvertAll | button | Edit Convert All Endnotes |
| SwapAllNotes | button | Edit Swap All Notes |
| InsertEnSpace | button | Insert En Space |
| InsertEmSpace | button | Insert Em Space |
| IndexMarkEntry | button | Mark Entry... |
| AutoMarkIndexEntries | button | Auto Mark Index Entries |
| CitationMark | button | Mark Citation... |
| TableOfAuthoritiesEditCategory | button | Edit TOA Category |
| IndexInsert | button | Insert Index... |
| TableOfContentsDialog | button | Insert Table of Contents... |
| TableOfContentsMarkEntry | button | Mark Entry |
| TableOfFiguresInsert | button | Insert Table of Figures... |
| TableOfAuthoritiesInsert | button | Insert Table of Authorities... |
| DrawingUnselect | button | Draw Unselect |
| DrawingSelectNext | button | Draw Select Next |
| DrawingSelectPrevious | button | Draw Select Previous |
| TextBoxLinkCreate | button | Create Link |
| TextBoxLinkBreak | button | Break Link |
| TextBoxNextLinked | button | Next Text Box |
| TextBoxPreviousLinked | button | Previous Text Box |
| FormatSectionLayout | button | Format Section Layout |
| StylesRedefineStyle | button | Redefine Style |
| Heading1Apply | button | Apply Heading 1 |
| Heading2Apply | button | Apply Heading 2 |
| Heading3Apply | button | Apply Heading 3 |
| ListBulletApply | button | Apply List Bullet |
| TextBoxConvertToFrame | button | Convert Text Box To Frame |
| ListPromote | button | Promote List |
| ListDemote | button | Demote List |
| NextMisspeling | button | Next Misspelling |
| HyphenationManual | button | Manual |
| BulletsAndNumberingClassic | button | Tools Bullets Numbers |
| CompareAndCombine | button | Compare & Combine |
| Calculate | button | Tools Calculate |
| KeyboardCustomizationWord | button | Tools Customize Keyboard Shortcut |
| ListCommands | button | List Commands |
| PrintOptionsMenuWord | button | Options |
| SpellingRecheckDocument | button | Tools Spelling Recheck Document |
| ReviewChangeUserName | button | Change User Name... |
| AutoFormatOptions | button | Tools Options AutoFormat |
| AutoFormatAsYouType | button | Tools Options AutoFormat As You Type |
| MailMergeConvertChevrons | button | Mail Merge Convert Chevrons |
| MailMergeAskToConvertChevrons | button | Mail Merge Ask To Convert Chevrons |
| ControlRun | button | Control Run |
| ShrinkSelection | button | Shrink Selection |
| StyleNormal | button | Normal Style |
| TableCellNext | button | Next Cell |
| TableCellPrevious | button | Previous Cell |
| StartOfRow | button | Start Of Row |
| EndOfRow | button | End Of Row |
| StartOfColumn | button | Start Of Column |
| EndOfColumn | button | End Of Column |
| WindowMinimizeAll | button | Minimize All |
| WindowMaximizeAll | button | Maximize All |
| WindowRestoreAll | button | Restore All |
| FieldClick | button | Do Field Click |
| SelectCurrentFont | button | Select Cur Font |
| SelectCurrentAlignment | button | Select Cur Alignment |
| SelectCurrentSpacing | button | Select Cur Spacing |
| SelectCurrentIndent | button | Select Cur Indent |
| SelectCurrentTabs | button | Select Cur Tabs |
| SelectCurrentColor | button | Select Cur Color |
| FramesRemove | button | Remove Frames |
| MenuMode | button | Menu Mode |
| PageNumberFormat | button | Format Page Numbers... |
| Zoom200 | button | View Zoom200 |
| Zoom75 | button | View Zoom75 |
| AddressFontsFormat | button | Format Addr Fonts |
| ReturnAddressFormatFontDialog | button | Format Ret Addr Fonts |
| FileLocations | button | Tools Options File Locations |
| CreateDirectoryClassic | button | Tools Create Directory |
| TableOfContentsUpdateClassic | button | Update Table of Contents... |
| FootnoteSeparatorWord | button | View Footnote Separator |
| FootnoteContinuationSeparator | button | View Footnote Cont Separator |
| FootnoteContinuationNotice | button | View Footnote Cont Notice |
| EndnoteSeparator | button | View Endnote Separator |
| EndnoteContinuationSeparator | button | View Endnote Cont Separator |
| EndnoteContinuationNotice | button | View Endnote Cont Notice |
| AutoCaptionInsert | button | Insert Auto Caption |
| CaptionInsertWord | button | Insert Add Caption |
| InsertCaptionNumbering | button | Insert Caption Numbering |
| AutoCorrectReplaceText | button | Tools AutoCorrect Replace Text |
| AutoCorrectInitialCaps | button | Tools AutoCorrect Initial Caps |
| AutoCorrectSentenceCaps | button | Tools AutoCorrect Sentence Caps |
| AutoCorrectDays | button | Tools AutoCorrect Days |
| AutoCorrectSmartQuotes | button | Tools AutoCorrect Smart Quotes |
| AutoCorrectCapsLockOff | button | Tools AutoCorrect Caps Lock Off |
| AutoCorrectExceptions | button | Tools AutoCorrect Exceptions |
| WindowSizeAll | button | Size All |
| WindowMoveAll | button | Move All |
| ConnectToNetworkDrive | button | Connect |
| GoToAnnotationScope | button | Goto Annotation Scope |
| FontSubstitution | button | Font Substitution |
| ScreenRefresh | button | Screen Refresh |
| CharacterLeft | button | Char Left |
| CharacterRight | button | Char Right |
| WordLeft | button | Word Left |
| WordRight | button | Word Right |
| ExtendSelectionLeft | button | Sent Left |
| ExtendSelectionRight | button | Sent Right |
| ParagraphUp | button | Para Up |
| ParagraphDown | button | Para Down |
| LineUp | button | Line Up |
| LineDown | button | Line Down |
| CharacterLeftExtend | button | Char Left Extend |
| CharacterRightExtend | button | Char Right Extend |
| WordLeftExtend | button | Word Left Extend |
| WordRightExtend | button | Word Right Extend |
| ExtendSelectionLeftSentence | button | Sent Left Extend |
| ExtendSelectionRightSentence | button | Sent Right Extend |
| ParagraphUpExtend | button | Para Up Extend |
| ParagraphDownExtend | button | Para Down Extend |
| LineUpExtend | button | Line Up Extend |
| LineDownExtend | button | Line Down Extend |
| PageUpExtend | button | Page Up Extend |
| PageDownExtend | button | Page Down Extend |
| StartOfLineExtend | button | Start Of Line Extend |
| EndOfLineExtend | button | End Of Line Extend |
| StartOfWindowExtend | button | Start Of Window Extend |
| EndOfWindowExtend | button | End Of Window Extend |
| StartOfDocumentExtend | button | Start Of Doc Extend |
| EndOfDocumentExtend | button | End Of Doc Extend |
| SymbolFont | button | Symbol Font |
| GrammarSettingsDialog | button | Tools Gram Settings |
| FileNewDefault | button | New |
| FilePrintQuick | button | Quick Print |
| SpellingAndGrammar | button | Spelling & Grammar |
| ReviewPreviousChangeClassic | button | Previous Change |
| ReviewNextChangeClassic | button | Next Change |
| MessageProperties | button | Properties |
| PictureInsertFromFile | button | Picture... |
| TableDrawBorderPenWeight | dropDown | Pen Weight |
| TableShowGridlines | toggleButton | View Gridlines |
| ShapeStraightConnectorArrow | toggleButton | Straight Arrow Connector |
| ShapeElbowConnectorArrow | toggleButton | Elbow Arrow Connector |
| HyperlinkOpen | button | Open Hyperlink |
| TextWrappingTopAndBottom | toggleButton | Top and Bottom |
| TextWrappingThrough | toggleButton | Through |
| MacroRecordOrStop | button | Record Macro / Stop Recorder |
| AutoManager | button | AutoManager... |
| EndnoteOrFootnoteConvert | button | Convert Endnote/Footnote |
| FootnoteSeparatorReset | button | Reset |
| PasteAsHyperlink | button | Paste as Hyperlink |
| ProofingOptions | button | Options... |
| ParagraphDistributed | toggleButton | Distributed |
| HyphenationOptions | button | Hyphenation Options... |
| TableRowsOrColumnsDistribute | button | Distribute Rows/Columns |
| MergeOrSplitCells | button | Merge/Split Cells |
| ReviewJapaneseConsistencyChecker | button | Japanese Consistency Checker... |
| AutoSummaryResummarize | button | Resummarize |
| AutoSummaryUpdateProperties | button | Update Properties |
| DeleteWord | button | Delete Word |
| DeleteWordBack | button | Delete Back Word |
| CharacterFormattingReset | button | Reset Character Formatting |
| HeadingNumbers | button | Heading Numbers |
| PictureSetTransparentColor | toggleButton | Set Transparent Color |
| PageColorMoreColorsDialog | button | More Colors... |
| PageColorFillEffects | button | Fill Effects... |
| BorderTopWord | toggleButton | Top Border |
| BorderBottomWord | toggleButton | Bottom Border |
| BorderLeftWord | toggleButton | Left Border |
| BorderRightWord | toggleButton | Right Border |
| TextDirection | button | Text Direction |
| FieldsManage | button | Manage |
| FileSaveAsHtml | button | Save as HTML... |
| SortAscendingWord | button | Sort Ascending |
| SortDescendingWord | button | Sort Descending |
| FrameInsertHorizontal | button | Horizontal Frame |
| FieldCodesToggle | button | Toggle Field Codes |
| GoToFootnote | button | Go to Footnote |
| GoToEndnote | button | Go to Endnote |
| SpellingHideErrors | toggleButton | Hide Spelling Errors |
| GrammarHideErrors | toggleButton | Hide Grammar Errors |
| Dictionary | button | Dictionary |
| SummaryInformation | button | Summary Information... |
| FootnoteEndnoteOptions | button | Footnote/Endnote Options |
| UnderlineDotted | toggleButton | Dotted Underline |
| NumberingRemove | button | Remove Numbering |
| PictureEditClassic | button | Picture |
| GoToStartOfWindow | button | Start of Window |
| GoToEndOfWindow | button | End of Window |
| AutoCorrectHECorrect | button | HECorrect |
| FileSendToPowerPoint | button | Send to Microsoft Office PowerPoint |
| FormatObjectDialogClassic | button | AutoShape... |
| AutoFormatNow | button | AutoFormat... |
| DataFormWord | button | Data Form |
| BulletListDefault | button | Tools Bullet List Default |
| NumberListDefault | button | Tools Number List Default |
| OutlineNumberDefault | button | Format Outline Number Default |
| FormatNumberDefault | button | Format Number Default |
| TableOfContentsRebuild | button | Rebuild Table of Contents |
| FootnoteEndnoteDialog | button | Footnote and Endnote Dialog... |
| TableInsertDialogWord | button | Insert Table... |
| FormFieldClear | button | Clear Form Field |
| ObjectBringInFrontOfText | button | Bring in Front of Text |
| ObjectSendBehindText | button | Send Behind Text |
| PageBreakInsertWord | button | Page Break |
| BordersShadingDialogWord | button | Borders and Shading... |
| TextBoxWordClassic | button | Text Box |
| IndentIncreaseWord | button | Increase Indent |
| IndentDecreaseWord | button | Decrease Indent |
| SelectObjects | button | Select Objects |
| Callout | button | Callout |
| ReplaceWithAutoText | button | Replace with AutoText |
| LinkToPreviousClassic | button | Link to Previous |
| HangulHanjaConversion | button | Hangul Hanja Conversion... |
| HeaderSourceEdit | button | Edit Header Source |
| IndentIncrease | button | Increase Indent |
| IndentDecrease | button | Decrease Indent |
| AsianLayoutFitText | button | Fit Text... |
| AsianLayoutPhoneticGuide | button | Phonetic Guide... |
| AsianLayoutCombineCharacters | button | Combine Characters... |
| JapanesePostcardDialog | button | Japanese Postcard... |
| CharacterBorder | toggleButton | Character Border |
| CharacterShading | toggleButton | Character Shading |
| ViewWebLayoutView | toggleButton | Web Layout |
| PasteAlternative | button | Paste Table |
| PasteAsNestedTable | button | Paste as Nested Table |
| HyperlinkRemove | button | Remove Hyperlink |
| MacroSecurity | button | Macro Security |
| HorizontalLineInsert | button | Horizontal Line |
| WebPagePreview | button | Web Page Preview |
| RightToLeftRun | button | Rtl Run |
| LeftToRightRun | button | Ltr Run |
| BoldRun | button | Bold Run |
| ItalicRun | button | Italic Run |
| TableSelectCell | button | Select Cell |
| TableDelete | button | Delete Table |
| TableRowsInsertAboveWord | button | Insert Above |
| TableRowsInsertBelowWord | button | Insert Below |
| TableColumnsInsertLeft | button | Insert Left |
| TableColumnsInsertRight | button | Insert Right |
| TablePropertiesDialog | button | Properties... |
| TableOptionsDialog | button | Cell Margins... |
| TableCellOptions | button | Cell Options... |
| SendCopySendNow | button | Send Now |
| SendCopySelectNames | button | Select Names |
| SendCopyCheckNames | button | Check Names |
| SendCopyAddressBookTo | button | To: Focus |
| SendCopyAddressBookCc | button | CC: Focus |
| SendCopyAddressBookBcc | button | Bcc: Focus |
| SendCopyFocusSubject | button | Subject Focus |
| SendCopyOptions | button | Mail Options |
| SendCopyFlag | button | Mail Flag |
| SendCopySaveAttachment | button | Save Mail Attachments |
| FileCloseOrExit | button | Close or Exit |
| ImeReconvert | button | Reconvert |
| SendCopySendToMailRecipient | toggleButton | Mail Recipient |
| TableOfContentsInFrame | button | Table of Contents in Frame |
| SetLanguageMenu | comboBox | Language |
| TableWrapping | button | Table Wrapping |
| EmailOptions | button | E-mail Options... |
| ComAddInsDialog | button | COM Add-Ins... |
| SignaturesStationeryDialog | button | Signatures... |
| FramePropertiesDialog | button | Frame Properties... |
| OfficeOnTheWeb | button | Microsoft Office Online |
| PictureBulletsInsert | button | Picture Bullets... |
| FileNewWebPage | button | New Web Page |
| FileNewBlankDocument | button | New Blank Document |
| FileNewDialogClassic | button | New Document or Template... |
| FileSaveAsWebPage | button | Save as Web Page... |
| HorizontalLineInsertClassic | button | Horizontal Line... |
| WebOptionsDialog | button | Web Options... |
| FramesNewFramesPageWizard | button | New Frames Page |
| FrameCreateAbove | button | New Frame Above |
| FrameCreateBelow | button | New Frame Below |
| FrameCreateLeft | button | New Frame Left |
| FrameCreateRight | button | New Frame Right |
| FrameDelete | button | Delete Frame |
| EastAsianEditingMarks | toggleButton | Show/Hide Editing Marks |
| TableAutoFitContents | button | AutoFit Contents |
| TableAutoFitWindow | button | AutoFit Window |
| TableAutoFitFixedColumnWidth | button | Fixed Column Width |
| TableCellAlignTopLeft | toggleButton | Align Top Left |
| TableCellAlignTopCenter | toggleButton | Align Top Center |
| TableCellAlignTopRight | toggleButton | Align Top Right |
| TableCellAlignMiddleLeft | toggleButton | Align Center Left |
| TableCellAlignMiddleCenter | toggleButton | Align Center |
| TableCellAlignMiddleRight | toggleButton | Align Center Right |
| TableCellAlignBottomLeft | toggleButton | Align Bottom Left |
| TableCellAlignBottomCenter | toggleButton | Align Bottom Center |
| TableCellAlignBottomRight | toggleButton | Align Bottom Right |
| WebControlCheckBox | button | Checkbox |
| WebControlOptionButton | button | Option Button |
| WebControlDropDownBox | button | Drop-Down Box |
| WebControlListBox | button | List Box |
| WebControlTextBox | button | Textbox |
| WebControlTextArea | button | Text Area |
| WebControlSubmit | button | Submit |
| WebControlSubmitWithImage | button | Submit with Image |
| WebControlReset | button | Reset |
| WebControlHidden | button | Hidden |
| WebControlPassword | button | Password |
| UnderlineColorMoreColorsDialog | button | More Colors... |
| ChineseTranslationDialog | button | Translate with Options... |
| TableInsertMultidiagonalCell | button | Insert Multidiagonal Cell... |
| AsianLayoutHorizontalInVertical | button | Horizontal in Vertical... |
| AsianLayoutTwoLinesInOne | button | Two Lines in One... |
| AsianLayoutCharactersEnclose | button | Enclose Characters... |
| EnvelopeChineseDialog | button | Chinese Envelope... |
| ObjectsMultiSelect | button | Select Multiple Objects |
| TranslateToTraditionalChinese | button | Traditional |
| TextWrappingBehindText | toggleButton | Behind Text |
| TextWrappingInFrontOfText | toggleButton | In Front of Text |
| WatermarkCustomDialog | button | Custom Watermark... |
| FrameSaveCurrentAs | button | Save Current Frame As... |
| TranslateToSimplifiedChinese | button | Simplified |
| WebDesignMode | toggleButton | Web Design Mode |
| ViewMasterDocumentViewClassic | button | Master Document Tools |
| WhiteSpaceBetweenPagesShowHide | button | White Space Between Pages |
| EditField | button | Edit Field... |
| WordCountRecount | button | Recount |
| StylesModifyStyle | button | Modify Style |
| StyleByExample | button | Style by Example |
| CssLinksEdit | button | Edit CSS Links |
| StylesPane | button | Styles... |
| DeleteStyle | button | Delete Style |
| StylesRenameStyle | button | Rename Style |
| SelectNumber | button | Select Number |
| NumberingRestart | button | Restart Numbering |
| DrawingCanvasInsert | button | New Drawing Canvas |
| DiagramRadialInsertClassic | button | Radial Diagram |
| DiagramCycleInsertClassic | button | Cycle Diagram |
| DiagramPyramidInsertClassic | button | Pyramid Diagram |
| DiagramTargetInsertClassic | button | Target Diagram |
| DiagramVennDiagramInsertClassic | button | Venn Diagram |
| DiagramChangeToRadialClassic | button | Radial |
| DiagramChangeToCycleClassic | button | Cycle |
| DiagramChangeToTargetClassic | button | Target |
| DiagramChangeToVennDiagramClassic | button | Venn |
| CopyPasteSettings | button | Copy & paste settings... |
| PasteByAppendingTable | button | Paste by Appending Table |
| OrganizationChartInsertAssistant | button | Assistant |
| OrganizationChartInsertCoworker | button | Coworker |
| OrganizationChartInsertSubordinate | button | Subordinate |
| OrganizationChartDeleteNode | button | Delete |
| DrawingCanvasFit | button | Fit |
| DrawingCanvasResize | button | Resize |
| LabelOptions | button | Label Options... |
| SendCopySetup | button | Envelope Setup... |
| MailMergeMergeToEMail | button | Send E-mail Messages... |
| MailMergeMergeToFax | button | Merge to Fax |
| MailMergeCreateList | button | Type New List... |
| MailMergeEditList | button | Edit Mail Merge List |
| DrawingCanvasExpand | button | Expand |
| ActivateProduct | button | Activate Product... |
| TextWrappingInLineWithText | toggleButton | In Line with Text |
| ConsistencyCheck | button | Consistency Check... |
| SelectTextWithSimilarFormatting | button | Select Text with Similar Formatting |
| ReviewSendForReview | button | Send for Review... |
| WebComponent | button | Web Component... |
| DiagramChangeToPyramidClassic | button | Pyramid |
| DiagramShapeMoveBackwardClassic | button | Move Shape Backward |
| DiagramShapeMoveForwardClassic | button | Move Shape Forward |
| CharacterCodeToggle | button | Toggle Character Code |
| SmartTagOptions | button | Tools Options Smart Tag |
| SendCopyFocusIntroduction | button | Introduction Focus |
| StylesStyleVisibility | button | Style Visibility |
| DiagramStylesClassic | button | Diagram Styles... |
| MailMergeHighlightMergeFields | toggleButton | Highlight Merge Fields |
| MailMergeWizard | toggleButton | Step by Step Mail Merge Wizard... |
| OrganizationChartAutoLayout | toggleButton | AutoLayout |
| OrganizationChartSelectLevel | button | Level |
| OrganizationChartSelectBranch | button | Branch |
| OrganizationChartSelectAllAssistants | button | All Assistants |
| OrganizationChartSelectAllConnectors | button | All Connectors |
| MailMergeJapaneseGreetingInsert | button | Greeting... |
| MailMergeJapaneseGreetingJapaneseOpeningSentenceInsert | button | Opening... |
| MailMergeJapaneseGreetingClosingSentenceInsert | button | Closing... |
| RevealFormatting | button | Reveal Formatting... |
| DiagramReverseClassic | button | Reverse |
| DiagramAutoLayoutClassic | toggleButton | AutoLayout |
| TextBoxAutosize | button | Autosize textbox |
| TranslationPane | button | Translate... |
| GoToTableOfContents | button | Go to TOC |
| TableOfContentsUpdate | button | Update Table... |
| OutlineLevelGallery | dropDown | Outline Level |
| OutlineShowLevel | dropDown | Show Level: |
| NumberingContinue | button | Continue Numbering |
| FileCheckOut | button | Check Out |
| FileCheckIn | button | Check In |
| OrganizationChartLayoutStandard | toggleButton | Standard |
| OrganizationChartLayoutBothHanging | toggleButton | Both Hanging |
| OrganizationChartLayoutLeftHanging | toggleButton | Left Hanging |
| OrganizationChartLayoutRightHanging | toggleButton | Right Hanging |
| ReviewShowReviewersMenu | menu | Reviewers |
| ReviewReplyWithChanges | button | Reply with Changes... |
| ReviewEndReview | button | End Review... |
| NormalizeText | button | Normalize text |
| StylesStyleSeparator | button | Style Separator |
| SpeakLearnFromDocument | button | Learn from Document... |
| PictureEditWord | button | Edit Picture |
| OutlinePromoteToHeading | button | Promote to Heading 1 |
| MicrosoftOutlook | button | Microsoft Outlook |
| ReviewShowComments | toggleButton | Comments |
| ReviewShowInsertionsAndDeletions | toggleButton | Insertions and Deletions |
| ReviewShowFormatting | toggleButton | Formatting |
| ReviewPreviousChange | button | Previous |
| ReviewNextChange | button | Next |
| ReviewReviewingPane | toggleButton | Reviewing Pane |
| ReviewAcceptAllChangesShown | button | Accept All Changes Shown |
| ReviewAcceptAllChangesInDocument | button | Accept All Changes in Document |
| ReviewRejectAllChangesShown | button | Reject All Changes Shown |
| ReviewRejectAllChangesInDocument | button | Reject All Changes in Document |
| ReviewDeleteAllCommentsShown | button | Delete All Comments Shown |
| ReviewDeleteAllCommentsInDocument | button | Delete All Comments in Document |
| ShowRepairs | button | Show Repairs |
| MailMergeMatchFields | button | Match Fields... |
| MailMergeAddressBlockInsert | button | Address Block... |
| MailMergeGreetingLineInsert | button | Greeting Line... |
| MailMergeMergeFieldInsert | button | Insert Merge Field |
| MailMergeRecipientsEditList | button | Edit Recipient List... |
| MailMergeEmailOptions | button | Mail Merge E-Mail Options |
| MailMergePrintOptions | button | Mail Merge Print Options |
| MailMergeFaxOptions | button | Mail Merge Fax Options |
| MailMergeMergeToNewDocumentOptions | button | Mail Merge to New Document Options |
| PicturesCompress | button | Compress Pictures... |
| Security | button | Security |
| TableAutoFormatStyle | button | Table AutoFormat... |
| MailMergeEditAddressBlock | button | Edit Address Block... |
| MailMergeEditGreetingLine | button | Edit Greeting Line... |
| MailMergeFindRecipient | button | Find Recipient... |
| FormFieldReset | button | Reset Form Fields |
| MailMergeUpdateLabels | button | Update Labels |
| DiagramFitToContentsClassic | button | Fit to Contents |
| DiagramResizeClassic | toggleButton | Resize |
| DiagramExpandClassic | button | Expand |
| OrganizationChartResize | button | Resize Organization Chart |
| AccountSettings | button | Account Settings... |
| MailMergeSetDocumentType | button | Main document setup |
| DiagramShapeInsertClassic | button | Insert Shape |
| OrganizationChartStyle | button | Style... |
| ReviewDisplayForReview | dropDown | Display for Review |
| DiagramAutoFormatClassic | button | Use AutoFormat |
| Translate | button | Translate... |
| ClearContentsWord | button | Contents |
| DrawingCanvasScale | button | Scale Drawing |
| ProtectDocument | toggleButton | Protect Document... |
| XmlViewStructure | button | View XML Structure |
| ReadingViewClose | button | Edit |
| ResearchPane | toggleButton | Research... |
| DocumentMapReadingView | button | Document Map |
| ReadingViewResearchPane | button | Research... |
| ReadingViewFontSizeIncrease | button | Increase Text Size |
| ReadingViewFontSizeDecrease | button | Decrease Text Size |
| ReadingViewShowPrintedPage | button | Show Printed Page |
| ViewRulerWord | checkBox | Ruler |
| FileInternetFax | button | Internet Fax |
| DocumentUpdatePane | button | Show Document Update Pane |
| ViewDocumentActionsPane | toggleButton | Document Actions |
| XmlToggleTagView | button | Toggle XML Tag View |
| InkDeleteAllInk | button | Delete All Ink |
| LookUp | button | Look Up |
| WindowSideBySide | toggleButton | View Side by Side |
| PrivacyOptionsDialog | button | Privacy Options... |
| FileVersionHistoryWord | button | View Version History |
| WindowSideBySideSynchronousScrolling | toggleButton | Synchronous Scrolling |
| WindowResetPosition | button | Reset Window Position |
| InkColorMoreColorsDialog | button | More Ink Colors... |
| XmlOptionsDialog | button | XML Options... |
| XmlTransformation | toggleButton | Transformation |
| StyleEnforcementSettings | button | Style Enforcement Settings |
| ContactUs | button | Contact Us... |
| FilePermissionUnrestricted | toggleButton | Unrestricted Access |
| FilePermissionDoNotDistribute | toggleButton | Restricted Access |
| FilePermissionView | button | View Permission |
| FilePermission | button | Permission |
| ReadingViewAllowMultiplePages | button | Allow Multiple Pages |
| ReadingViewStartInking | button | Ink |
| ReadingViewUnlockDocumentLayout | button | Unlock Document Layout |
| VoiceInsertInComment | button | Insert Voice |
| ViewThumbnails | checkBox | Thumbnails |
| Thesaurus | button | Thesaurus... |
| InkingStart | button | Start Inking |
| ReviewShowRevisionsInBalloons | toggleButton | Show Revisions in Balloons |
| ReviewShowRevisionsInline | toggleButton | Show All Revisions Inline |
| ReviewShowOnlyCommentsAndFormattingInBaloons | toggleButton | Show Only Comments and Formatting in Balloons |
| ReviewShowInkMarkup | toggleButton | Ink |
| CheckForUpdates | button | Check for Updates |
| ViewFullScreenReadingView | toggleButton | Full Screen Reading |
| InkCopyAsText | button | Copy Ink As Text |
| InkDrawingAndWriting | button | Ink Drawing and Writing |
| ReviewInkCommentPen | toggleButton | Pen |
| ReviewInkCommentEraser | toggleButton | Eraser |
| InkEraser | toggleButton | Eraser |
| SendCopyAttachmentOptions | button | Attachment Options |
| FilePermissionRestrictAs | button | Manage Credentials |
| ListSetNumberingValue | button | Set Numbering Value... |
| FileViewDigitalSignatures | toggleButton | View Signatures |
| FileWorkflowTasks | button | View Workflow Tasks |
| FileStartWorkflow | button | Workflows |
| SignatureLineInsert | button | Signature Line |
| BibliographyInsert | button | Insert Bibliography |
| BibliographyStyle | comboBox | Style: |
| CitationInsert | gallery | Insert Citation |
| BibliographyManageSources | button | Manage Sources... |
| BibliographyAddNewSource | button | Add New Source... |
| LabelInsert | button | Label |
| BarcodeInsert | button | Barcode |
| ReviewShowMarkupAreaHighlight | toggleButton | Markup Area Highlight |
| ChartStylesGallery | gallery | Quick Styles |
| ChartLayoutGallery | gallery | Quick Layout |
| ChartSaveTemplates | button | Save As Template |
| ChartAxisTitles | menu | Axis Titles |
| ChartAxes | menu | Axes |
| ChartGridlines | menu | Gridlines |
| ChartFormatSelection | button | Format Selection |
| ChartElementSelector | comboBox | Chart Elements |
| PageMarginsGallery | gallery | Margins |
| DropCapInsertGallery | gallery | Drop Cap |
| TabPictureToolsFormat | tab | Format |
| ShapesInsertGallery | gallery | Shapes |
| ShapeChangeShapeGallery | gallery | Change Shape |
| ShapeFillTextureGallery | gallery | Texture |
| ShapeStylesGallery | gallery | Quick Styles |
| PageOrientationGallery | gallery | Orientation |
| FileServerTasksMenu | menu | Server |
| FileSendMenu | menu | Send |
| TabInsert | tab | Insert |
| TabReferences | tab | References |
| TabMailings | tab | Mailings |
| TabReviewWord | tab | Review |
| TabView | tab | View |
| GroupFont | group | Font |
| GroupParagraph | group | Paragraph |
| GroupStyles | group | Styles |
| GroupProofing | group | Proofing |
| GroupInsertPages | group | Pages |
| GroupInsertIllustrations | group | Illustrations |
| GroupWordArtText | group | Text |
| GroupParagraphLayout | group | Paragraph |
| GroupCitationsAndBibliography | group | Citations & Bibliography |
| GroupFootnotes | group | Footnotes |
| GroupTableOfContents | group | Table of Contents |
| GroupMailMergeWriteInsertFields | group | Write & Insert Fields |
| GroupMailMergePreviewResults | group | Preview Results |
| GroupMailMergeFinish | group | Finish |
| GroupChangesTracking | group | Tracking |
| GroupComments | group | Comments |
| GroupChanges | group | Changes |
| GroupCompare | group | Compare |
| GroupPictureSize | group | Size |
| GroupTableAlignment | group | Alignment |
| GroupTextBoxText | group | Text |
| GroupArrange | group | Arrange |
| GroupShapeStyles | group | Shape Styles |
| GroupChartLayouts | group | Chart Layouts |
| GroupChartStyles | group | Chart Styles |
| GroupChartAxes | group | Axes |
| GroupChartShapes | group | Insert |
| GroupOrganizationChartShapeInsert | group | Insert |
| StylesManageStyles | button | Manage Styles |
| StylesStyleInspector | toggleButton | Style Inspector |
| ObjectEffectPresetGallery | gallery | Preset |
| PictureEffectsPresetGallery | gallery | Preset |
| \_3DRotationGallery | gallery | 3-D Rotation |
| TabSmartArtToolsDesign | tab | Design |
| TabSmartArtToolsFormat | tab | Format |
| TabChartToolsDesign | tab | Design |
| TabChartToolsLayout | tab | Layout |
| TabChartToolsFormat | tab | Format |
| ShapeFillColorPicker | gallery | Shape Fill |
| OutlineColorPicker | gallery | Picture Border |
| FileDocumentInspect | button | Inspect Document |
| QuickStylesGallery | gallery | Quick Styles |
| QuickStylesSets | gallery | Style Set |
| ClearFormatting | button | Clear Formatting |
| PanningHand | toggleButton | Panning Hand |
| BulletsGalleryWord | gallery | Bullets |
| NumberingGalleryWord | gallery | Numbering |
| GroupControls | group | Controls |
| GroupZoom | group | Zoom |
| ArrowStyleGallery | gallery | Arrows |
| OutlineDashesGallery | gallery | Dashes |
| OutlineWeightGallery | gallery | Weight |
| TabTableToolsLayout | tab | Layout |
| GroupPictureTools | group | Adjust |
| GroupSize | group | Size |
| ReviewTrackChangesMenu | splitButton | Track Changes |
| ReviewAcceptChangeMenu | splitButton | Accept |
| ReviewRejectChangeMenu | splitButton | Reject |
| ReviewBalloonsMenu | menu | Balloons |
| GroupTableRowsAndColumns | group | Rows & Columns |
| GroupTableData | group | Data |
| ObjectAlignMenu | menu | Align |
| ObjectRotateGallery | gallery | Rotate |
| SelectMenu | menu | Select |
| FontColorPicker | gallery | Font Color |
| TableColumnsGallery | gallery | Columns |
| TabHome | tab | Home |
| ChartTitle | gallery | Chart Title |
| ChartPrimaryHorizontalAxisTitle | gallery | Primary Horizontal Axis Title |
| ChartPrimaryVerticalAxisTitle | gallery | Primary Vertical Axis Title |
| ChartDepthAxisTitle | gallery | Depth Axis Title |
| ChartLegend | gallery | Legend |
| ChartDataLabel | gallery | Data Labels |
| ChartPrimaryHorizontalGridlines | gallery | Primary Horizontal Gridlines |
| ChartPrimaryVerticalGridlines | gallery | Primary Vertical Gridlines |
| ChartDepthGridlines | gallery | Depth Gridlines |
| ChartPrimaryHorizontalAxis | gallery | Primary Horizontal Axis |
| ChartPrimaryVerticalAxis | gallery | Primary Vertical Axis |
| ChartDepthAxis | gallery | Depth Axis |
| ChartDataTable | gallery | Data Table |
| ChartTrendline | gallery | Trendline |
| ChartErrorBars | gallery | Error Bars |
| ChartLines | gallery | Lines |
| ChartUpDownBars | gallery | Up/Down Bars |
| ChartPlotArea | gallery | Plot Area |
| ChartWall | gallery | Chart Wall |
| ChartFloor | gallery | Chart Floor |
| TabPageLayoutWord | tab | Page Layout |
| SmartArtAddShape | button | Add Shape |
| SmartArtLargerShape | button | Larger |
| SmartArtSmallerShape | button | Smaller |
| SmartArtResetGraphic | button | Reset Graphic |
| SmartArtTextPane | toggleButton | Text Pane |
| SmartArtEditIn2D | toggleButton | Edit in 2-D |
| SmartArtLayoutGallery | gallery | Change Layout |
| SmartArtMoreLayoutsDialog | button | More Layouts... |
| SmartArtStylesGallery | gallery | Quick Styles |
| SmartArtChangeColorsGallery | gallery | Change Colors |
| ObjectEffectSoftEdgesGallery | gallery | Soft Edges |
| ObjectEffectGlowGallery | gallery | Glow |
| GradientGallery | gallery | Gradient |
| ObjectEffectShadowGallery | gallery | Shadow |
| TextEffectTransformGallery | gallery | Transform |
| TabHeaderAndFooterToolsDesign | tab | Design |
| GroupHeaderFooterOptions | group | Options |
| ReviewPreviousCommentWord | button | Previous |
| ReviewNextCommentWord | button | Next |
| BulletDefineNew | button | Define New Bullet... |
| DefineNewNumberFormat | button | Define New Number Format... |
| ThemeColorsGallery | gallery | Colors |
| HeaderInsertGallery | gallery | Header |
| FooterInsertGallery | gallery | Footer |
| CoverPageInsertGallery | gallery | Cover Page |
| PageNumberFieldInsertGallery | gallery | Current Position |
| WatermarkGallery | gallery | Watermark |
| EquationInsertGallery | gallery | Equation |
| QuickTablesInsertGallery | gallery | Quick Tables |
| QuickPartsInsertGallery | gallery | Quick Parts |
| GroupSmartArtLayouts | group | Layouts |
| GroupSmartArtQuickStyles | group | SmartArt Styles |
| GroupSmartArtCreateGraphic | group | Create Graphic |
| GroupSmartArtReset | group | Reset |
| GroupSmartArtSize | group | Size |
| SaveSelectionToQuickPartGallery | button | Save Selection to Quick Part Gallery... |
| SaveSelectionToCoverPageGallery | button | Save Selection to Cover Page Gallery... |
| SaveSelectionToEquationGallery | button | Save Selection to Equation Gallery... |
| SaveSelectionToFooterGallery | button | Save Selection to Footer Gallery... |
| SaveSelectionToHeaderGallery | button | Save Selection to Header Gallery... |
| SaveSelectionToPageNumberGallery | button | Save Selection to Page Number Gallery... |
| SaveSelectionToQuickTablesGallery | button | Save Selection to Quick Tables Gallery... |
| SaveSelectionToWaterMarkGallery | button | Save Selection to Watermark Gallery... |
| ThemeSearchOfficeOnline | button | More Themes on Microsoft Office Online... |
| TabAddIns | tab | Add-Ins |
| ReviewShowMarkupMenu | menu | Show Markup |
| ObjectEditShapeMenu | menu | Edit Shape |
| SymbolInsertGallery | gallery | Symbol |
| TableStyleClear | button | Clear |
| FileSaveAsPdfOrXps | button | Publish as PDF or XPS |
| FileSaveAsWordOpenDocumentText | button | OpenDocument Text |
| SearchLibraries | button | Search Libraries... |
| MoreControlsDialog | button | More Controls... |
| GroupCode | group | Code |
| TabDeveloper | tab | Developer |
| GroupXml | group | XML |
| XmlStructure | toggleButton | Structure |
| XmlSchema | button | Schema |
| XmlExpansionPacksWord | button | Expansion Packs |
| GroupCaptions | group | Captions |
| GroupIndex | group | Index |
| GroupTableOfAuthorities | group | Table of Authorities |
| GroupEditing | group | Editing |
| SelectMenuExcel | splitButton | Find & Select |
| GroupClipboard | group | Clipboard |
| GroupInsertTables | group | Tables |
| GroupInsertLinks | group | Links |
| GroupInsertSymbols | group | Symbols |
| GroupInsertBarcode | group | Barcode |
| PageSizeGallery | gallery | Size |
| ObjectPictureFill | button | Picture... |
| TextWrappingMenu | menu | Text Wrapping |
| WindowSwitchWindowsMenuWord | menu | Switch Windows |
| ThemeColorsCreateNew | button | Create New Theme Colors... |
| ThemeFontsCreateNew | button | Create New Theme Fonts... |
| ShapeFillMoreGradientsDialog | button | More Gradients... |
| ShadowOptionsDialog | button | Shadow Options... |
| BuildingBlocksOrganizer | button | Building Blocks Organizer... |
| ReviewCompareMenu | menu | Compare |
| ReviewCompareTwoVersions | button | Compare... |
| ReviewCombineRevisions | button | Combine... |
| ReviewCompareMajorVersion | button | Major Version |
| ReviewCompareLastVersion | button | Last Version |
| ReviewCompareSpecificVersion | button | Specific Version... |
| PropertyInsert | gallery | Document Property |
| ObjectsAlignSelectedSmart | toggleButton | Align Selected Objects |
| ObjectsAlignRelativeToContainerSmart | toggleButton | Align to Slide |
| ObjectsAlignLeftSmart | button | Align Left |
| ObjectsAlignRightSmart | button | Align Right |
| ObjectsAlignTopSmart | button | Align Top |
| ObjectsAlignBottomSmart | button | Align Bottom |
| ObjectsAlignCenterHorizontalSmart | button | Align Center |
| ObjectsAlignMiddleVerticalSmart | button | Align Middle |
| AlignDistributeHorizontally | button | Distribute Horizontally |
| AlignDistributeVertically | button | Distribute Vertically |
| GroupProtect | group | Protect |
| IndexUpdate | button | Update Index |
| MarginsCustomMargins | button | Custom Margins... |
| MailMergeFinishAndMergeMenu | menu | Finish & Merge |
| HyphenationMenu | menu | Hyphenation |
| HyphenationAutomatic | toggleButton | Automatic |
| HyphenationNone | toggleButton | None |
| MailMergeRules | menu | Rules |
| TabWordArtToolsFormat | tab | Format |
| MailMergeStartMailMergeMenu | menu | Start Mail Merge |
| MailMergeStartLetters | toggleButton | Letters |
| MailMergeStartEmail | toggleButton | E-Mail Messages |
| MailMergeStartEnvelopes | toggleButton | Envelopes... |
| MailMergeStartLabels | toggleButton | Labels... |
| MailMergeStartDirectory | toggleButton | Directory |
| MailMergeClearMergeType | toggleButton | Normal Word Document |
| MailMergeSelectRecipients | menu | Select Recipients |
| TableOfContentsAddTextGallery | gallery | Add Text |
| BorderColorPicker | gallery | Pen Color |
| TranslationToolTip | gallery | Translation ScreenTip |
| ThemeFontsGallery | gallery | Fonts |
| ThemeEffectsGallery | gallery | Effects |
| FileProperties | toggleButton | Properties |
| TabPrintPreview | tab | Print Preview |
| GroupPrintPreviewPrint | group | Print |
| GroupPrintPreviewPreview | group | Preview |
| BreaksGallery | gallery | Breaks |
| LineNumbersMenu | menu | Line Numbers |
| LineNumbersOff | toggleButton | None |
| LineNumbersContinuous | toggleButton | Continuous |
| LineNumbersResetPage | toggleButton | Restart Each Page |
| LineNumbersResetSection | toggleButton | Restart Each Section |
| LineNumbersSuppress | toggleButton | Suppress for Current Paragraph |
| TabOutlining | tab | Outlining |
| GroupOutliningClose | group | Close |
| GroupOutliningTools | group | Outline Tools |
| GroupMasterDocument | group | Master Document |
| TableSelectMenu | menu | Select |
| TableDeleteRowsAndColumnsMenuWord | menu | Delete |
| GroupTableMerge | group | Merge |
| TableAutoFitMenu | menu | AutoFit |
| GroupTableDrawBorders | group | Draw Borders |
| TableBordersMenu | splitButton | Borders |
| FileCreateDocumentWorkspace | toggleButton | Create Document Workspace |
| FileSaveToDocumentManagementServer | button | Document Management Server |
| FileDocumentManagementInformation | toggleButton | Document Management Information |
| QuickAccessToolbarCustomization | button | Customize Quick Access Toolbar... |
| FilePrepareMenu | menu | Prepare |
| FileMarkAsFinal | toggleButton | Mark as Final |
| FileAddDigitalSignature | button | Add a Digital Signature |
| SignatureServicesAdd | button | Add Signature Services... |
| QuickStylesSaveSelectionAsNew | button | Save Selection as a New Quick Style... |
| StylesPaneNewStyle | button | New Style... |
| QuickStylesSaveQuickStyleSet | button | Save as Quick Style Set... |
| ChangeCaseGallery | gallery | Change Case |
| AlignJustifyMenu | menu | Justify |
| ControlProperties | button | Properties |
| GroupHeaderFooterInsert | group | Insert |
| GroupHeaderFooterNavigation | group | Navigation |
| GroupHeaderFooterPosition | group | Position |
| HeaderFooterDifferentFirstPageWord | checkBox | Different First Page |
| HeaderFooterDifferentOddEvenPageWord | checkBox | Different Odd & Even Pages |
| HeaderFooterShowDocumentText | checkBox | Show Document Text |
| BlankPageInsert | button | Blank Page |
| ShadowStyleGalleryClassic | gallery | Shadow Effects |
| TabOrganizationChartToolsFormat | tab | Format |
| TabDiagramToolsFormatClassic | tab | Format |
| GroupInkSelect | group | Select |
| WordArtSpacingMenu | menu | Spacing |
| TextAlignMenu | menu | Alignment |
| DiagramChangeToMenuClassic | menu | Change To |
| PictureBrightnessGallery | gallery | Brightness |
| PictureContrastGallery | gallery | Contrast |
| PicturePositionGallery | gallery | Position |
| TabPictureToolsFormatClassic | tab | Format |
| PictureBrightnessAndContrastDialog | button | Picture Correction Options... |
| GroupMailMergeStart | group | Start Mail Merge |
| OleObjectInsertMenu | splitButton | Object |
| ShadingColorPicker | gallery | Shading |
| ShadingColorsMoreColorsDialog | button | More Colors... |
| SmartArtAddShapeAfter | button | Add Shape After |
| SmartArtAddShapeBefore | button | Add Shape Before |
| SmartArtAddShapeAbove | button | Add Shape Above |
| SmartArtAddShapeBelow | button | Add Shape Below |
| SmartArtAddAssistant | button | Add Assistant |
| ChartSwitchRowColumn | button | Switch Row/Column |
| ChartShowData | button | Edit Data... |
| ChartRefresh | button | Refresh Data |
| ChartChangeType | button | Change Chart Type... |
| GroupChartData | group | Data |
| GroupChartType | group | Type |
| \_3DRotationOptionsDialog | button | 3-D Rotation Options... |
| \_3DBevelOptionsDialog | button | 3-D Options... |
| SmartArtOrganizationChartLeftHanging | button | Left Hanging |
| SmartArtOrganizationChartRightHanging | button | Right Hanging |
| SmartArtOrganizationChartBoth | button | Both |
| SmartArtOrganizationChartStandard | button | Standard |
| SmartArtRightToLeft | toggleButton | Right to Left |
| ListLevelGallery | gallery | Change List Level |
| MultilevelListGallery | gallery | Multilevel List |
| ListDefineNew | button | Define New Multilevel List... |
| ListDefineNewStyle | button | Define New List Style... |
| ViewMessageBar | checkBox | Message Bar |
| ApplyStylesPane | toggleButton | Apply Styles... |
| InsertAlignmentTab | button | Insert Alignment Tab |
| ShapeStylesOtherThemeFillsGallery | gallery | Other Theme Fills |
| SmartArtOrganizationChartMenu | menu | Layout |
| ReviewChangeTrackingOptions | button | Change Tracking Options... |
| SymbolsDialog | button | More Symbols... |
| ReviewReviewingPaneHorizontal | toggleButton | Reviewing Pane Horizontal... |
| ReviewReviewingPaneVertical | toggleButton | Reviewing Pane Vertical... |
| \_3DEffectsGalleryClassic | gallery | 3-D Effects |
| \_3DDirectionGalleryClassic | gallery | Direction |
| \_3DLightingGalleryClassic | gallery | Lighting |
| TabDrawingToolsFormatClassic | tab | Format |
| GroupShadowEffects | group | Shadow Effects |
| Group3DEffects | group | 3-D Effects |
| WordArtStylesGalleryClassic | gallery | WordArt |
| WordArtInsertGalleryClassic | gallery | WordArt |
| TableInsertGallery | gallery | Table |
| ShapeStylesGalleryClassic | gallery | Shape Styles |
| WordArtChangeShapeGallery | gallery | Change Shape |
| ShadowColorPickerClassic | gallery | Shadow Color |
| \_3DEffectColorPickerClassic | gallery | 3-D Color |
| ShapeFillGradientGalleryClassic | gallery | Gradient |
| AsianLayoutMenu | menu | Asian Layout |
| JapaneseGreetingsInsertMenu | menu | Japanese Greetings |
| AlignJustifyWithMixedLanguages | toggleButton | Justify |
| AlignJustifyLow | toggleButton | Justify Low |
| AlignJustifyMedium | toggleButton | Justify Medium |
| AlignJustifyHigh | toggleButton | Justify High |
| AlignJustifyThai | toggleButton | Distribute |
| TextHighlightColorPicker | gallery | Text Highlight Color |
| PageColorPicker | gallery | Page Color |
| GoToHeader | button | Go to Header |
| GoToFooter | button | Go to Footer |
| HighlightingStop | button | Stop Highlighting |
| UnderlineGallery | gallery | Underline |
| UnderlineColorPicker | gallery | Underline Color |
| UnderlineMoreUnderlinesDialog | button | More Underlines... |
| TextDirectionGalleryWord | gallery | Text Direction |
| GroupAddInsMenuCommands | group | Menu Commands |
| GroupAddInsToolbarCommands | group | Toolbar Commands |
| GroupInk | group | Ink |
| TabInkToolsPens | tab | Pens |
| GroupInkPens | group | Pens |
| GroupInkClose | group | Close |
| InkBallpointPen | toggleButton | Ballpoint Pen |
| InkFeltTipPen | toggleButton | Felt Tip Pen |
| InkHighlighter | toggleButton | Highlighter |
| GroupBorder | group | Border |
| PictureRecolorGalleryWord | gallery | Recolor |
| \_3DSurfaceMaterialGalleryClassic | gallery | Surface |
| \_3DExtrusionDepthGalleryClassic | gallery | Depth |
| GroupHeaderFooter | group | Header & Footer |
| GroupPageLayoutSetup | group | Page Setup |
| GroupPageBackground | group | Page Background |
| ThemeSaveCurrent | button | Save Current Theme... |
| ThemesGallery | gallery | Themes |
| ChartResetToMatchStyle | button | Reset to Match Style |
| Chart3DView | button | 3-D Rotation... |
| ObjectSizeDialog | button | Size... |
| AutoTextGallery | gallery | AutoText |
| TextBoxInsertGallery | gallery | Text Box |
| PageNumbersInHeaderInsertGallery | gallery | Top of Page |
| PageNambersInFooterInsertGallery | gallery | Bottom of Page |
| PageNambersInMarginsInsertGallery | gallery | Page Margins |
| SaveSelectionToAutoTextGallery | button | Save Selection to AutoText Gallery |
| SaveSelectionToTextBoxGallery | button | Save Selection to Text Box Gallery |
| SaveSelectionToPageNumberTop | button | Save Selection as Page Number (Top) |
| SaveSelectionToPageNumberBottom | button | Save Selection as Page Number (Bottom) |
| SaveSelectionToPageNumberMargin | button | Save Selection as Page Number (Margin) |
| HeaderFooterEditHeader | button | Edit Header |
| TableStyleHeaderRowWord | checkBox | Header Row |
| TableStyleTotalRowWord | checkBox | Total Row |
| TableStylesFirstColumnWord | checkBox | First Column |
| TableStyleLastColumnWord | checkBox | Last Column |
| TableStyleBandedRowsWord | checkBox | Banded Rows |
| TableStyleBandedColumnsWord | checkBox | Banded Columns |
| TableStyleModify | button | Modify Table Style... |
| TabTableToolsDesign | tab | Design |
| TableStylesGalleryWord | gallery | Table Styles |
| ReviewViewChangesInTheSourceDocument | button | View changes in the source document |
| GroupThemesWord | group | Themes |
| LayoutOptionsDialog | button | More Layout Options... |
| DrawingObjectFormatDialog | button | Advanced Tools... |
| ReflectionGallery | gallery | Reflection |
| PictureRecolorGallery | gallery | Recolor |
| SmartArtPromote | button | Promote |
| SmartArtDemote | button | Demote |
| ChartTitleOptionsDialog | button | More Title Options... |
| ChartLegendOptionsDialogDialog | button | More Legend Options... |
| ChartDataLabelDialog | button | More Data Label Options... |
| ChartPrimaryHorizontalAxisTitleOptionsDialog | button | More Primary Horizontal Axis Title Options... |
| ChartPrimaryVerticalAxisTitleOptionsDialog | button | More Primary Vertical Axis Title Options... |
| ChartSecondaryHorizontalAxisTitleOptionsDialog | button | More Secondary Horizontal Axis Title Options... |
| ChartSecondaryVerticalAxisTitleOptionsDialog | button | More Secondary Vertical Axis Title Options... |
| ChartDepthAxisTitleOptionsDialog | button | More Depth Axis Title Options... |
| ChartPrimaryHorizontalGridlinesOptionsDialog | button | More Primary Horizontal Gridlines Options... |
| ChartPrimaryVerticalGridlinesOptionsDialog | button | More Primary Vertical Gridlines Options... |
| ChartSecondaryHorizontalGridlinesOptionsDialog | button | More Secondary Horizontal Gridlines Options... |
| ChartSecondaryVerticalGridlinesOptionsDialog | button | More Secondary Vertical Gridlines Options... |
| ChartDepthGridlinesOptionsDialog | button | More Depth Gridlines Options... |
| ChartPrimaryHorizontalAxisOptionsDialog | button | More Primary Horizontal Axis Options... |
| ChartPrimaryVerticalAxisOptionsDialog | button | More Primary Vertical Axis Options... |
| ChartSecondaryHorizontalAxisOption | button | More Secondary Horizontal Axis Options... |
| ChartSecondaryVerticalAxisOptionsDialog | button | More Secondary Vertical Axis Options... |
| ChartDepthAxisOptionsDialog | button | More Depth Axis Options... |
| ChartDataTableOptionsDialog | button | More Data Table Options... |
| ChartTrendlineOptionsDialog | button | More Trendline Options... |
| ChartErrorBarsOptionsDialog | button | More Error Bars Options... |
| ChartUpDownBarsOptionsDialog | button | More Up/Down Bars Options... |
| ChartPlotAreaOptionsDialog | button | More Plot Area Options... |
| ChartWallOptionsDialog | button | More Walls Options... |
| ChartFloorOptionsDialog | button | More Floor Options... |
| ChartSecondaryHorizontalAxisTitle | gallery | Secondary Horizontal Axis Title |
| ChartSecondaryVerticalAxisTitle | gallery | Secondary Vertical Axis Title |
| ChartSecondaryHorizontalGridlines | gallery | Secondary Horizontal Gridlines |
| ChartSecondaryVerticalGridlines | gallery | Secondary Vertical Gridlines |
| ChartSecondaryHorizontalAxis | gallery | Secondary Horizontal Axis |
| ChartSecondaryVerticalAxis | gallery | Secondary Vertical Axis |
| GroupAddInsCustomToolbars | group | Custom Toolbars |
| ReviewDeleteCommentsMenu | splitButton | Delete |
| ObjectBringToFrontMenu | splitButton | Bring to Front |
| ObjectSendToBackMenu | splitButton | Send to Back |
| ObjectsGroupMenu | menu | Group |
| SignatureLineInsertMenu | splitButton | Signature Line |
| FileSaveAsWord97\_2003 | button | Word 97-2003 Document |
| EnglishWritingAssistant | button | English Assistant |
| TableOfContentsGallery | gallery | Table of Contents |
| SaveSelectionToTableOfContentsGallery | button | Save Selection to Table of Contents Gallery... |
| ObjectsAlignRelativeToMargin | toggleButton | Align to Margin |
| TabTextBoxToolsFormat | tab | Format |
| TextBoxStyleGallery | gallery | Text Box Style |
| TextBoxPositionGallery | gallery | Position |
| FileSaveAsMenu | splitButton | Save As Other Format |
| FilePrintMenu | splitButton | Preview and Print |
| FilePermissionRestrictMenu | menu | Restrict Permission |
| GroupEnvelopeLabelCreate | group | Create |
| ReviewInkCommentNew | button | Ink Comment |
| DocumentPanelTemplate | button | Document Panel |
| ObjectSizeDialogClassic | button | Size... |
| TextBoxInsertVerticalWord | button | Draw Vertical Text Box |
| BevelShapeGallery | gallery | Bevel |
| \_3DBevelPictureTopGallery | gallery | Bevel |
| BibliographyAddNewPlaceholder | button | Add New Placeholder... |
| GroupTable | group | Table |
| GroupTableCellSize | group | Cell Size |
| GlowColorPicker | gallery | More Glow Colors |
| RecolorColorPicker | gallery | More Variations |
| GlowColorMoreColorsDialog | button | More Colors... |
| PictureRecolorMoreColorsDialog | button | More Colors... |
| QuickStylesResetDocumentStyles | button | Reset Document Quick Styles |
| QuickStylesResetFromTemplate | button | Reset to Quick Styles from Template |
| WatermarkRemove | button | Remove Watermark |
| CoverPageRemove | button | Remove Current Cover Page |
| HeaderFooterRemoveHeaderWord | button | Remove Header |
| HeaderFooterRemoveFooterWord | button | Remove Footer |
| PageNumbersRemove | button | Remove Page Numbers |
| TableOfContentsRemove | button | Remove Table of Contents |
| SmartArtAddBullet | button | Add Bullet |
| ThemeResetFromTemplate | button | Reset to Theme from Template |
| PictureChange | button | Change Picture... |
| GroupWordArtStyles | group | WordArt Styles |
| TextFillColorPicker | gallery | Text Fill |
| TextOutlineColorPicker | gallery | Text Outline |
| TextOutlineColorMoreColorsDialog | button | More Outline Colors... |
| TextEffectsMenu | menu | Text Effects |
| TextStylesGallery | gallery | Quick Styles |
| WordArtClear | button | Clear WordArt |
| TextPictureFill | button | Picture... |
| TextFillGradientGallery | gallery | Gradient |
| TextFillMoreGradientsDialog | button | More Gradients... |
| TextFillTextureGallery | gallery | Texture |
| TextOutlineDashesGallery | gallery | Dashes |
| TextOutlineMoreLinesDialog | button | More Lines... |
| TextOutlineWeightGallery | gallery | Weight |
| TextEffectShadowGallery | gallery | Shadow |
| TextEffectsMoreShadowsDialog | button | Shadow Options... |
| TextEffectsBevelMore3DOptionsDialog | button | 3-D Options... |
| TextEffects3DRotationGallery | gallery | 3-D Rotation |
| TextEffects3DRotationOptionsDialog | button | 3-D Rotation Options... |
| TextEffectGlowGallery | gallery | Glow |
| TextGlowColorPicker | gallery | More Glow Colors |
| TextGlowColorMoreColorsDialog | button | More Colors... |
| TextReflectionGallery | gallery | Reflection |
| ShapeEffectsMenu | menu | Shape Effects |
| UpgradeDocument | button | Convert |
| GroupHeaderFooterClose | group | Close |
| GroupChartCurrentSelection | group | Current Selection |
| GroupChartLabels | group | Labels |
| PageSizeMorePaperSizesDialog | button | More Paper Sizes... |
| LineNumbersOptionsDialog | button | Line Numbering Options... |
| TableOfAuthoritiesUpdate | button | Update Table |
| TableOfFiguresUpdate | button | Update Table |
| ContentControlsGroup | button | Group |
| BevelTextGallery | gallery | Bevel |
| PictureCorrectionsDialog | button | Picture Corrections Options... |
| GroupTextBoxStyles | group | Text Box Styles |
| GroupTableStylesWord | group | Table Styles |
| GroupWordArtStylesClassic | group | WordArt Styles |
| PageBorderAndShadingDialog | button | Page Borders... |
| OutlineViewClose | button | Close Outline View |
| SmartArtAddShapeSplitMenu | splitButton | Add Shape Options |
| ContentControlRichText | button | Rich Text |
| ContentControlText | button | Text |
| ContentControlPicture | button | Picture |
| ContentControlComboBox | button | Combo Box |
| ContentControlDropDownList | button | Drop-Down List |
| ContentControlBuildingBlockGallery | button | Building Block Gallery |
| ContentControlDate | button | Date |
| AutoSummaryToolsMenu | menu | AutoSummary Tools |
| GroupInkFormat | group | Format |
| InkColorPicker | gallery | Color |
| GroupDiagramLayoutClassic | group | Layout |
| GroupDiagramStylesClassic | group | Styles |
| GroupOrganizationChartLayoutClassic | group | Layout |
| GroupOrganizationChartStyleClassic | group | Styles |
| TextBoxDrawMenu | menu | Draw Text Box |
| TextBoxInsertWord | button | Text Box |
| GroupOrganizationChartSelect | group | Select |
| ShapeFillEffectMoreGradientsDialogClassic | button | More Gradients... |
| ShapeFillEffectMoreTexturesDialogClassic | button | More Textures... |
| ShapeFillEffectPatternClassic | button | Pattern... |
| BibliographyGallery | gallery | Bibliography |
| CustomHeaderGallery | gallery | Custom Header |
| CustomFooterGallery | gallery | Custom Footer |
| CustomCoverPageGallery | gallery | Custom Cover Page |
| CustomPageNumberGallery | gallery | Custom Page Number |
| CustomPageNumberTopGallery | gallery | Custom Top of Page |
| CustomPageNumberBottomGallery | gallery | Custom Bottom of Page |
| CustomPageMargins | gallery | Custom Page Margins |
| CustomWatermarkGallery | gallery | Custom Watermark |
| CustomEquationsGallery | gallery | Custom Equation |
| CustomTablesGallery | gallery | Custom Tables |
| CustomQuickPartsGallery | gallery | Custom Quick Parts |
| CustomAutoTextGallery | gallery | Custom AutoText |
| CustomTextBoxGallery | gallery | Custom Text Box |
| CustomTableOfContentsGallery | gallery | Custom Table of Contents |
| CustomBibliographyGallery | gallery | Custom Bibliography |
| CustomGallery1 | gallery | Custom Gallery 1 |
| CustomGallery2 | gallery | Custom Gallery 2 |
| CustomGallery3 | gallery | Custom Gallery 3 |
| CustomGallery4 | gallery | Custom Gallery 4 |
| CustomGallery5 | gallery | Custom Gallery 5 |
| SaveSelectionToBibliographyGallery | button | Save Selection to Bibliography Gallery... |
| MailMergeRecepientsUseOutlookContacts | button | Select from Outlook Contacts... |
| FootnoteNext | splitButton | Next Footnote |
| ReviewReviewingPaneMenu | splitButton | Reviewing Pane |
| GroupSizeClassic | group | Size |
| GroupPictureSizeClassic | group | Size |
| GroupPictureToolsClassic | group | Adjust |
| GalleryAllShapesAndCanvas | gallery | Shapes |
| GroupShapesClassic | group | Insert Shapes |
| GroupSmartArtShapes | group | Shapes |
| GroupShapeStylesClassic | group | Shape Styles |
| GroupInsertText | group | Text |
| Drawing1ColorPickerFill | gallery | Shape Fill |
| ShapeOutlineColorPicker | gallery | Picture Border |
| Drawing1ColorPickerLineStyles | gallery | Picture Border |
| Drawing1GalleryTextures | gallery | Texture |
| InsertBuildingBlocksEquationsGallery | gallery | Equation |
| Drawing1GalleryBrightness | gallery | Brightness |
| Drawing1GalleryContrast | gallery | Contrast |
| GroupDiagramArrangeClassic | group | Arrange |
| GroupTextBoxArrange | group | Arrange |
| ContentControlsGroupMenu | menu | Group |
| ContentControlsUngroup | button | Ungroup |
| ControlsGalleryClassic | gallery | Legacy Tools |
| ReviewShowSourceDocumentsMenu | gallery | Show Source Documents |
| HeaderFooterEditFooter | button | Edit Footer |
| QuickStylesSetAsDefault | button | Set as Default |
| EquationInsertNew | button | Insert New Equation |
| EquationProfessional | button | Professional |
| EquationLinearFormat | button | Linear |
| EquationNormalText | toggleButton | Normal Text |
| EquationSymbolsInsertGallery | gallery | Equation Symbols |
| EquationIntegralGallery | gallery | Integral |
| EquationFractionGallery | gallery | Fraction |
| EquationRadicalGallery | gallery | Radical |
| EquationLargeOperatorGallery | gallery | Large Operator |
| EquationDelimiterGallery | gallery | Bracket |
| EquationScriptGallery | gallery | Script |
| EquationFunctionGallery | gallery | Function |
| EquationAccentGallery | gallery | Accent |
| EquationLimitGallery | gallery | Limit and Log |
| EquationOperatorGallery | gallery | Operator |
| EquationMatrixGallery | gallery | Matrix |
| EquationOptions | button | Equation Options... |
| TabEquationToolsDesign | tab | Design |
| GroupEquationTools | group | Tools |
| GroupEquationSymbols | group | Symbols |
| GroupEquationStructures | group | Structures |
| MailMergeMergeFieldInsertMenu | splitButton | Insert Merge Field |
| PasteMenu | splitButton | Paste |
| GroupPictureStyles | group | Picture Styles |
| PictureStylesGallery | gallery | Quick Styles |
| ReviewAcceptChangeAndMoveToNext | button | Accept and Move to Next |
| ReviewRejectChangeAndMoveToNext | button | Reject and Move to Next |
| PictureEffectsShadowGallery | gallery | Shadow |
| PictureEffectsGlowGallery | gallery | Glow |
| PictureEffectsSoftEdgesGallery | gallery | Soft Edges |
| PictureReflectionGallery | gallery | Reflection |
| PictureRotationGallery | gallery | 3-D Rotation |
| InkToolsClose | button | Close Ink Tools |
| GroupChineseTranslation | group | Chinese Translation |
| LineSpacingMenu | menu | Line spacing |
| GroupDocumentViews | group | Document Views |
| GroupViewShowHide | group | Show/Hide |
| GroupWindow | group | Window |
| ViewGridlines | checkBox | View Gridlines |
| FileDocumentEncrypt | toggleButton | Encrypt Document |
| WordArtFormatDialog | button | Format Text Effects... |
| ObjectRotationOptionsDialog | button | More Rotation Options... |
| MoreTextureOptions | button | More Textures... |
| TextFillColorMoreColorsDialog | button | More Fill Colors... |
| ZoomTwoPages | button | Two Pages |
| QuickPartsInsertFromOnline | button | Get More on Office Online... |
| GroupPrintPreviewPageSetup | group | Page Setup |
| ShowRuler | checkBox | Ruler |
| FileEmailAsPdfEmailAttachment | button | E-mail as PDF Attachment |
| FileEmailAsXpsEmailAttachment | button | E-mail as XPS Attachment |
| GroupTemplates | group | Templates |
| SpellingMenu | splitButton | Spelling |
| PictureEffectsMenu | menu | Picture Effects |
| PictureShapeGallery | gallery | Change Shape |
| PictureBorderColorPickerClassic | gallery | Picture Border |
| GroupChartBackground | group | Background |
| GroupChartAnalysis | group | Analysis |
| FileNewBlogPost | button | Blog |
| TabBlogPost | tab | Blog Post |
| GroupBlogPublish | group | Blog |
| BlogPublishMenu | splitButton | Publish |
| BlogPublish | button | Publish |
| BlogPublishDraft | button | Publish as Draft |
| BlogManageAccounts | button | Manage Accounts |
| BlogCategoryInsert | button | Insert Category |
| BlogOpenExisting | button | Open Existing |
| GroupBlogBasicText | group | Basic Text |
| TabBlogInsert | tab | Insert |
| GroupBlogInsertText | group | Text |
| NewTableStyleWord | button | New Table Style... |
| MenuPublish | menu | Publish |
| ChangeStylesMenu | menu | Change Styles |
| GroupBlogInsertLinks | group | Links |
| FileCompatibilityCheckerWord | button | Run Compatibility Checker |
| FileSaveAsOtherFormats | button | Other Formats |
| FileSaveAsWordDocx | button | Word Document |
| FileSaveAsWordDotx | button | Word Template |
| ZoomCurrent100 | button | 100% |
| Drawing1ColorPickerLineStylesWordArt | gallery | Picture Border |
| Drawing1ColorPickerFillWordArt | gallery | Shape Fill |
| BlogInsertCategories | button | Insert Category |
| GroupTableLayout | group | Table Style Options |
| TextFillMoreTextures | button | More Textures... |
| GroupMacros | group | Macros |
| PlayMacro | button | Macros |
| MenuMacros | splitButton | Macros |
| AdvertisePublishAs | button | Find add-ins for other file formats |
| ReviewProtectDocumentMenu | menu | Protect Document |
| ReviewRestrictFormatting | toggleButton | Restrict Formatting and Editing |
| BlogHomePage | button | Home Page |
| GroupBlogProofing | group | Proofing |
| GroupBlogStyles | group | Styles |
| AlternativeText | button | Size... |
| ThemeBrowseForThemes | button | Browse for Themes... |
| FileCheckOutDiscard | button | Discard Check Out |
| GroupBlogSymbols | group | Symbols |
| ClearMenuWord | menu | Clear |
| MdiChildSystemMenu | menu | Document |
| NudgeMenu | menu | Nudge |
| RevisionsMenu | menu | Track Changes |
| TableCellVerticalAlignmentMenu | menu | Alignment |
| FramesetMenu | menu | Frames |

### Excel 2007

| idMso | Control Type | Label |
| --- | --- | --- |
| Spelling | button | Spelling... |
| FileSave | button | Save |
| FilePrint | button | Print |
| ChartInsert | button | Chart... |
| FileNew | button | New |
| Copy | button | Copy |
| Cut | button | Cut |
| Paste | button | Paste |
| FileOpen | button | Open |
| ZoomPrintPreviewExcel | toggleButton | Zoom |
| Repeat | button | Repeat |
| UnderlineDouble | toggleButton | Double Underline |
| FileClose | button | Close |
| FormatPainter | toggleButton | Format Painter |
| FilePrintPreview | button | Print Preview |
| Bold | toggleButton | Bold |
| Italic | toggleButton | Italic |
| Underline | toggleButton | Underline |
| DarkShading | button | Dark Shading |
| AlignLeft | toggleButton | Align Left |
| AlignRight | toggleButton | Align Right |
| AlignCenter | toggleButton | Center |
| AlignJustify | toggleButton | Justify |
| Undo | gallery | Undo |
| Redo | gallery | Redo |
| BorderTop | toggleButton | Top Border |
| BorderBottom | toggleButton | Bottom Border |
| BorderLeft | toggleButton | Left Border |
| BorderRight | toggleButton | Right Border |
| BorderInside | button | Inside Borders |
| BorderOutside | button | Outside Borders |
| BorderNone | button | No Border |
| ObjectsGroup | button | Group |
| ObjectsUngroup | button | Ungroup |
| ObjectBringToFront | button | Bring to Front |
| ObjectSendToBack | button | Send to Back |
| ObjectBringForward | button | Bring Forward |
| ObjectSendBackward | button | Send Backward |
| ViewFullScreenView | toggleButton | Full Screen |
| ObjectsSelect | toggleButton | Select Objects |
| MacroRecord | button | Record Macro... |
| MacroPlay | button | Macros |
| ObjectFlipHorizontal | button | Flip Horizontal |
| ObjectFlipVertical | button | Flip Vertical |
| ObjectRotateRight90 | button | Rotate Right 90° |
| ObjectRotateLeft90 | button | Rotate Left 90° |
| ShapeFreeform | toggleButton | Freeform |
| ObjectEditPoints | toggleButton | Edit Points |
| FormControlEditBox | button | Edit Box |
| FormControlCheckBox | button | Check Box |
| FormControlComboBox | button | Combo Box |
| PropertySheet | button | Property Sheet |
| Lock | toggleButton | Lock |
| AutoSum | button | Sum |
| StylesDialogClassic | button | Edit Cell Styles |
| Camera | button | Camera |
| FormControlButton | button | Button |
| Calculator | button | Calculator |
| Strikethrough | toggleButton | Strikethrough |
| CellsDelete | button | Delete Cells... |
| CellsInsertDialog | button | Insert Cells... |
| WindowsArrangeAll | button | Arrange All |
| WindowNew | button | New Window |
| ReviewAcceptOrRejectChangeDialog | button | Accept/Reject Changes |
| SymbolInsert | button | Symbol... |
| ReplaceDialog | button | Replace... |
| PagePrevious | button | Previous Page |
| PageNext | button | Next Page |
| TextBoxInsertVertical | toggleButton | Vertical Text Box |
| ObjectsRegroup | button | Regroup |
| PrintAreaSetPrintArea | button | Set Print Area |
| PasteFormatting | button | Paste Formatting |
| PasteValues | button | Paste Values |
| FillRight | button | Right |
| FillDown | button | Down |
| EqualSign | button | Equal Sign |
| PlusSign | button | Plus Sign |
| MinusSign | button | Minus Sign |
| MultiplicationSign | button | Multiplication Sign |
| DivisionSign | button | Division Sign |
| ExponentiationSign | button | Exponentiation Sign |
| ParenthesisLeft | button | Left Parenthesis |
| ParenthesisRight | button | Right Parenthesis |
| ColonSign | button | Colon |
| CommaSign | button | Comma |
| PercentSign | button | Percent Sign |
| DollarSign | button | Dollar Sign |
| FunctionWizard | button | Insert Function... |
| ConstrainNumeric | button | Constrain Numeric |
| LightShading | button | Light Shading |
| AccountingFormat | button | Accounting Number Format |
| PercentStyle | button | Percent Style |
| CommaStyle | button | Comma Style |
| DecimalsIncrease | button | Increase Decimal |
| DecimalsDecrease | button | Decrease Decimal |
| MergeCenter | toggleButton | Merge & Center |
| FontSizeIncrease | button | Increase Font Size |
| FontSizeDecrease | button | Decrease Font Size |
| TextOrientationVertical | toggleButton | Vertical Text |
| TextOrientationRotateUp | toggleButton | Rotate Text Up |
| TextOrientationRotateDown | toggleButton | Rotate Text Down |
| AlignDistributeHorizontallyClassic | button | Distribute Horizontally |
| ShapeScribble | toggleButton | Scribble |
| OutlineSymbolsShowHide | button | Show Outline Symbols |
| TableSelectVisibleCells | button | Select Visible Cells |
| SelectCurrentRegion | button | Select Current Region |
| FreezePanes | button | Freeze Panes |
| ZoomIn | button | Zoom In |
| ZoomOut | button | Zoom Out |
| FormControlRadioButton | button | Option Button |
| FormControlScrollBar | button | Scroll Bar |
| FormControlListBox | button | List Box |
| TraceDependentRemoveArrows | button | Remove Dependent Arrows |
| TraceDependents | button | Trace Dependents |
| TracePrecedentsRemoveArrows | button | Remove Precedent Arrows |
| TraceRemoveAllArrows | button | Remove Arrows |
| FileUpdate | button | Update File |
| ReadOnly | button | Toggle Read Only |
| AutoFilterClassic | button | AutoFilter |
| Refresh | button | Refresh |
| PivotTableFieldSettings | button | Field Settings |
| PivotTableShowPages | button | Show Report Filter Pages... |
| OutlineShowDetail | button | Show Detail |
| TraceError | button | Trace Error |
| OutlineHideDetail | button | Hide Detail |
| AlignDistributeVerticallyClassic | button | Distribute Vertically |
| FormControlGroupBox | button | Group Box |
| FormControlSpinner | button | Spinner |
| TabOrder | button | Tab Order... |
| RunDialog | button | Run Dialog |
| FormControlCombinationListEdit | button | Combination List-Edit |
| FormControlCombinationDropDownEdit | button | Combination Drop-Down Edit |
| FormControlLabel | button | Label |
| TracePrecedents | button | Trace Precedents |
| CodeEdit | button | Code |
| PageBreakInsertOrRemove | button | Insert Page Break |
| QueryParameters | button | Parameters |
| RowHeight | button | Row Height... |
| ColumnWidth | button | Column Width... |
| OleObjectctInsert | button | Object... |
| SnapToGrid | toggleButton | Snap to Grid |
| ObjectsAlignLeft | button | Align Left |
| ObjectsAlignRight | button | Align Right |
| ObjectsAlignTop | button | Align Top |
| ObjectsAlignBottom | button | Align Bottom |
| ObjectsAlignCenterHorizontal | button | Align Center |
| ObjectsAlignMiddleVertical | button | Align Middle |
| ClipArtInsert | toggleButton | Clip Art... |
| ObjectRotateFree | button | Free Rotate |
| CombineCharacters | toggleButton | Yoko-Gumi |
| ViewNormalViewExcel | toggleButton | Normal |
| ViewPageBreakPreviewView | toggleButton | Page Break Preview |
| PictureCrop | toggleButton | Crop |
| FileCloseAll | button | Close All |
| FileSaveAs | button | Save As |
| AdvancedFileProperties | button | View Document Properties... |
| CopyAsPicture | button | Copy as Picture... |
| PasteSpecialDialog | button | Paste Special... |
| SelectAll | button | Select All |
| GoTo | button | Go To... |
| FileLinksToFiles | button | Edit Links to Files |
| HeaderFooterInsert | button | Header & Footer... |
| BulletsAndNumberingBulletsDialog | button | Bullets and Numbering... |
| AutoFormatDialog | button | AutoFormat... |
| MergeCells | button | Merge Cells |
| SplitCells | button | Split Cells... |
| ConvertTextToTable | button | Convert Text to Table... |
| ShowClipboard | button | Office Clipboard... |
| OutlookTaskCreate | button | Create Microsoft Office Outlook Task |
| WindowMinimize | button | Minimize |
| WindowRestore | button | Restore |
| WindowClose | button | Close |
| WindowSaveWorkspace | button | Save Workspace... |
| SheetDelete | button | Delete Sheet |
| SheetMoveOrCopy | button | Move or Copy Sheet... |
| ViewFormulaBar | checkBox | Formula Bar |
| SheetInsert | button | Insert Sheet |
| FormatCellsDialog | button | Format Cells... |
| GoalSeek | button | Goal Seek... |
| ScenarioManager | button | Scenario Manager... |
| DataFormExcel | button | Form... |
| OutlineSubtotals | button | Subtotal |
| DataTable | button | Data Table... |
| Consolidate | button | Consolidate... |
| WindowHide | button | Hide |
| WindowUnhide | button | Unhide... |
| FillUp | button | Up |
| FillLeft | button | Left |
| FillAcrossWorksheets | button | Across Worksheets... |
| FillSeries | button | Series... |
| FillJustify | button | Justify |
| ClearFormats | button | Clear Formats |
| ClearContents | button | Clear Contents |
| ClearComments | button | Clear Comments |
| NamePasteName | button | Paste Names... |
| NameCreateFromSelection | button | Create from Selection... |
| NamesApply | button | Apply Names... |
| RowHeightAutoFit | button | AutoFit Row Height |
| RowsHide | button | Hide Rows |
| RowsUnhide | button | Unhide Rows |
| ColumnWidthAutoFit | button | AutoFit Column Width |
| ColumnsHide | button | Hide Columns |
| ColumnsUnhide | button | Unhide Columns |
| ColumnWidthDefault | button | Default Width... |
| SheetRename | button | Rename Sheet |
| SheetHide | button | Hide Sheet |
| SheetUnhide | button | Unhide Sheet... |
| SheetProtect | button | Protect Sheet... |
| ReviewProtectWorkbook | toggleButton | Protect Workbook... |
| MacroRelativeReferences | toggleButton | Use Relative References |
| Filter | toggleButton | Filter |
| SortClear | button | Clear |
| AdvancedFilterDialog | button | Advanced... |
| OutlineAuto | button | Auto Outline |
| OutlineClear | button | Clear Outline |
| OutlineSettings | button | Group and Outline Settings |
| PrintPreviewClose | button | Close Print Preview |
| ZoomDialog | button | Zoom... |
| SortDialogClassic | button | Sort... |
| ExchangeFolder | button | Exchange Folder... |
| AddInManager | button | Add-Ins... |
| ViewCustomViews | button | Custom Views... |
| SheetBackground | button | Background... |
| ChartEditDataSource | button | Select Data... |
| ChartPlacement | button | Move Chart... |
| CalculateNow | button | Calculate Now |
| ObjectFormatDialog | button | Object... |
| Help | button | Help |
| PivotTableEnableSelection | toggleButton | Enable Selection |
| PivotTableListFormulas | button | List Formulas |
| PivotTableSelectData | button | Values |
| PivotTableSelectLabelAndData | button | Labels and Values |
| PivotTableSelectLabel | button | Labels |
| PasteAsPicture | button | Paste as Picture |
| PastePictureLink | button | Paste Picture Link |
| CalculateSheet | button | Calculate Sheet |
| TextOrientationAngleCounterclockwise | toggleButton | Angle Counterclockwise |
| TextOrientationAngleClockwise | toggleButton | Angle Clockwise |
| WebGoBack | button | Back |
| WebGoForward | button | Forward |
| SmartArtInsert | button | SmartArt... |
| ShapeRerouteConnectors | toggleButton | Reroute Connectors |
| ObjectNudgeUp | button | Up |
| ObjectNudgeDown | button | Down |
| ObjectNudgeLeft | button | Left |
| ObjectNudgeRight | button | Right |
| ShapeCurve | toggleButton | Curve |
| ShapeStraightConnector | toggleButton | Straight Connector |
| ShapeElbowConnector | toggleButton | Elbow Connector |
| ObjectFillMoreColorsDialog | button | More Fill Colors... |
| ObjectBorderOutlineColorMoreColorsDialog | button | More Outline Colors... |
| LineStylesDialog | button | More Lines... |
| ArrowsMore | button | More Arrows... |
| WordArtVerticalText | button | Vertical Text |
| ContrastMore | button | More Contrast |
| ContrastLess | button | Less Contrast |
| BrightnessMore | button | More Brightness |
| BrightnessLess | button | Less Brightness |
| ShadowNudgeUpClassic | button | Nudge Shadow Up |
| ShadowNudgeDownClassic | button | Nudge Shadow Down |
| ShadowNudgeLeftClassic | button | Nudge Shadow Left |
| ShadowNudgeRightClassic | button | Nudge Shadow Right |
| ObjectShadowColorMoreColorsDialog | button | More Shadow Colors... |
| \_3DEffectColorPickerMoreClassic | button | More 3-D Colors... |
| ShapeRectangle | toggleButton | Rectangle |
| ShapeRoundedRectangle | toggleButton | Rounded Rectangle |
| ShapeIsoscelesTriangle | toggleButton | Isosceles Triangle |
| ShapeOval | toggleButton | Oval |
| ShapeLeftBrace | toggleButton | Left Brace |
| ShapeRightBrace | toggleButton | Right Brace |
| ShapeArc | toggleButton | Arc |
| ShapeRightArrow | toggleButton | Right Arrow |
| ShapeDownArrow | toggleButton | Down Arrow |
| ShapeRoundedRectangularCallout | toggleButton | Rounded Rectangular Callout |
| ShapeStar | toggleButton | 5-Point Star |
| PictureReset | button | Reset Picture |
| \_3DEffectsOnOffClassic | toggleButton | 3-D On/Off |
| \_3DTiltDownClassic | button | Tilt Down |
| \_3DTiltUpClassic | button | Tilt Up |
| \_3DTiltLeftClassic | button | Tilt Left |
| \_3DTiltRightClassic | button | Tilt Right |
| \_3DSurfaceMaterialClassic | menu | Surface |
| \_3DExtrusionPerspectiveClassic | toggleButton | Perspective |
| \_3DExtrusionParallelClassic | toggleButton | Parallel |
| \_3DLightingFlatClassic | toggleButton | Bright |
| \_3DLightingNormalClassic | toggleButton | Normal |
| \_3DLightingDimClassic | toggleButton | Dim |
| \_3DSurfaceMatteClassic | toggleButton | Matte |
| \_3DSurfacePlasticClassic | toggleButton | Plastic |
| \_3DSurfaceMetalClassic | toggleButton | Metal |
| \_3DSurfaceWireFrameClassic | toggleButton | Wire Frame |
| SnapToShapes | toggleButton | Snap to Shape |
| HyperlinkInsert | button | Hyperlink... |
| PrintAreaAddToPrintArea | button | Add to Print Area |
| PrintAreaClearPrintArea | button | Clear Print Area |
| PageBreaksResetAll | button | Reset All Page Breaks |
| ReviewNewComment | button | New Comment |
| ReviewPreviousComment | button | Previous |
| ReviewNextComment | button | Next |
| ReviewDeleteComment | button | Delete |
| ReviewShowOrHideComment | button | Show/Hide Comment |
| ReviewShowAllComments | toggleButton | Show All Comments |
| PivotTableFieldInsert | button | Calculated Field... |
| PivotTableCalculatedItem | button | Calculated Item... |
| PivotTableSelectEntireTable | button | Entire PivotTable |
| PivotTableOptions | button | Options |
| DesignMode | toggleButton | Design Mode |
| PhoneticGuideEdit | button | Edit Phonetic |
| PhoneticGuideSettings | button | Phonetic Settings... |
| PhoneticGuideFieldShow | toggleButton | Show Phonetic Field |
| CircularReferences | gallery | Circular References |
| VisualBasic | button | Visual Basic |
| BorderThickBottom | button | Thick Bottom Border |
| BorderTopAndBottom | button | Top and Bottom Border |
| BorderTopAndDoubleBottom | button | Top and Double Bottom Border |
| BorderTopAndThickBottom | button | Top and Thick Bottom Border |
| BordersAll | button | All Borders |
| BorderThickOutside | button | Thick Box Border |
| Font | comboBox | Font: |
| FontSize | comboBox | Font Size: |
| StyleGalleryClassic | gallery | Style: |
| ZoomClassic | gallery | Zoom: |
| ScenarioGallery | gallery | Scenario: |
| DocumentLocation | comboBox | Address: |
| MergeCellsAcross | button | Merge Across |
| BorderInsideHorizontal | button | Inside Horizontal Border |
| BorderInsideVertical | button | Inside Vertical Border |
| BorderDiagonalDown | button | Diagonal Down Border |
| BorderDiagonalUp | button | Diagonal Up Border |
| TextDirectionLeftToRight | toggleButton | Left-to-Right |
| TextDirectionRightToLeft | toggleButton | Right-to-Left |
| ActiveXCheckBox | button | Check Box |
| FindDialogExcel | button | Find... |
| ActiveXTextBox | button | Text Box |
| ActiveXButton | button | Command Button |
| ActiveXRadioButton | button | Option Button |
| ActiveXListBox | button | List Box |
| ActiveXComboBox | button | Combo Box |
| ActiveXToggleButton | button | Toggle Button |
| ActiveXSpinButton | button | Spin Button |
| ActiveXScrollBar | button | Scroll Bar |
| ActiveXLabel | button | Label |
| ShadowSemitransparentClassic | toggleButton | Semitransparent Shadow |
| RightToLeftDocument | toggleButton | Right-to-Left Document |
| EditQuery | button | Edit Query... |
| DataRangeProperties | button | Properties |
| RefreshAll | button | Refresh All |
| RefreshCancel | button | Cancel Refresh |
| RefreshStatus | button | Refresh Status |
| PasteLink | button | Paste Link... |
| ClearAll | button | Clear All |
| DataValidation | button | Data Validation... |
| DataValidationCircleInvalid | button | Circle Invalid Data |
| ReviewShareWorkbook | button | Share Workbook... |
| ReviewHighlightChanges | button | Highlight Changes... |
| CompareAndMergeWorkbooks | button | Compare and Merge Workbooks... |
| DatabaseQueryNew | button | New Database Query... |
| DataValidationClearValidationCircles | button | Clear Validation Circles |
| ActiveXImage | button | Image |
| ShadowOnOrOffClassic | toggleButton | Shadow On/Off |
| ObjectSetShapeDefaults | button | Set AutoShape Defaults |
| FileSendAsAttachment | button | E-mail |
| FileNewDefault | button | New |
| FilePrintQuick | button | Quick Print |
| PictureInsertFromFile | button | Picture... |
| ShapeStraightConnectorArrow | toggleButton | Straight Arrow Connector |
| ShapeElbowConnectorArrow | toggleButton | Elbow Arrow Connector |
| PasteAsHyperlink | button | Paste as Hyperlink |
| ParagraphDistributed | toggleButton | Distributed |
| PictureSetTransparentColor | toggleButton | Set Transparent Color |
| PivotTableSolveOrder | button | Solve Order... |
| PivotTableReport | button | PivotTable and PivotChart Wizard |
| ReviewProtectAndShareWorkbook | button | Protect Sharing |
| OutlineGroup | button | Group... |
| OutlineUngroup | button | Ungroup... |
| IndentIncreaseExcel | button | Increase Indent |
| IndentDecreaseExcel | button | Decrease Indent |
| HangulHanjaConversion | button | Hangul Hanja Conversion... |
| MacroSecurity | button | Macro Security |
| WebPagePreview | button | Web Page Preview |
| SendCopySendNow | button | Send Now |
| SendCopySelectNames | button | Select Names |
| SendCopySendToMailRecipient | toggleButton | Mail Recipient |
| ComAddInsDialog | button | COM Add-Ins... |
| PivotChartInsertClassic | button | PivotChart |
| PivotFieldListShowHide | toggleButton | Field List |
| FileSaveAsWebPage | button | Save as Web Page... |
| GetExternalDataFromWeb | button | From Web |
| WebOptionsDialog | button | Web Options... |
| GetExternalDataFromText | button | From Text |
| FilePublishAsWebPage | button | Publish as Web Page... |
| PivotTableOlapOffline | button | Offline OLAP... |
| ObjectsMultiSelect | button | Select Multiple Objects |
| TextDirectionContext | toggleButton | Context |
| CalculateFull | button | TBA |
| WatchWindow | toggleButton | Watch Window |
| FormulaEvaluate | button | Evaluate Formula |
| AutoSumAverage | button | Average |
| AutoSumCount | button | Count Numbers |
| AutoSumMax | button | Max |
| AutoSumMin | button | Min |
| AutoSumMoreFunctions | button | More Functions... |
| PasteFormulas | button | Formulas |
| PasteNoBorders | button | No Borders |
| PasteTranspose | button | Transpose |
| DrawingCanvasFit | button | Fit |
| DrawingCanvasResize | button | Resize |
| DrawingCanvasExpand | button | Expand |
| ReviewSendForReview | button | Send for Review... |
| BorderDrawMenu | splitButton | Draw Border |
| BorderErase | toggleButton | Erase Border |
| BorderStyle | dropDown | Line Style |
| ShowFormulas | toggleButton | Show Formulas |
| PivotTableOlapPropertyFields | button | Property Fields... |
| TranslationPane | button | Translate... |
| ErrorChecking | button | Error Checking... |
| FileCheckOut | button | Check Out |
| FileCheckIn | button | Check In |
| PivotTableGenerateGetPivotData | checkBox | Generate GetPivotData |
| ReviewReplyWithChanges | button | Reply with Changes... |
| ReviewEndReview | button | End Review... |
| BorderDrawLine | toggleButton | Draw Border |
| BorderDrawGrid | toggleButton | Draw Border Grid |
| GetExternalDataImportClassic | button | Import External Data |
| PicturesCompress | button | Compress Pictures... |
| VerticallyDistributed | button | Vertically Distributed |
| ReviewAllowUsersToEditRanges | button | Allow Users to Edit Ranges... |
| SpeakCells | button | Speak Cells |
| SpeakStop | button | Stop Speaking |
| SpeakByRows | toggleButton | By Rows |
| SpeakByColumns | toggleButton | By Columns |
| SpeakOnEnter | toggleButton | On Enter |
| TableInsertExcel | button | Table |
| ResearchPane | toggleButton | Research... |
| TableStyleTotalsRow | checkBox | Total Row |
| TableRowsInsertAboveExcel | button | Insert Table Rows Above |
| TableRowsDeleteExcel | button | Delete Table Rows |
| TableConvertToRange | button | Convert to Range |
| PrintListRange | button | Print List |
| FileInternetFax | button | Internet Fax |
| XmlExport | button | Export |
| XmlImport | button | Import |
| ViewDocumentActionsPane | toggleButton | Document Actions |
| ReviewShowInk | toggleButton | Show Ink |
| TableColumnsInsertLeftExcel | button | Insert Table Columns to the Left |
| TableColumnsDeleteExcel | button | Delete Table Columns |
| InkDeleteAllInk | button | Delete All Ink |
| TableUnlinkExternalData | button | Unlink |
| TableExportTableToSharePointList | button | Export Table to SharePoint List... |
| PrivacyOptionsDialog | button | Privacy Options... |
| ListSynchronize | button | Synchronize List |
| ChangesDiscardAndRefresh | button | Discard Changes and Refresh |
| TableOpenInBrowser | button | Open in Browser |
| TableResize | button | Resize Table |
| XmlExpansionPacksExcel | button | Expansion Packs |
| FileVersionHistory | button | View Version History |
| XmlDataRefresh | button | Refresh Data |
| XmlMapProperties | button | Map Properties |
| WindowSideBySideSynchronousScrolling | toggleButton | Synchronous Scrolling |
| WindowResetPosition | button | Reset Window Position |
| InkColorMoreColorsDialog | button | More Ink Colors... |
| ContactUs | button | Contact Us... |
| FilePermissionUnrestricted | toggleButton | Unrestricted Access |
| FilePermissionDoNotDistribute | toggleButton | Restricted Access |
| FilePermissionView | button | View Permission |
| FilePermission | button | Permission |
| Thesaurus | button | Thesaurus... |
| InkingStart | button | Start Inking |
| CheckForUpdates | button | Check for Updates |
| InkCopyAsText | button | Copy Ink As Text |
| InkEraser | toggleButton | Eraser |
| FilePermissionRestrictAs | button | Manage Credentials |
| Connections | button | Connections |
| FileViewDigitalSignatures | toggleButton | View Signatures |
| FileWorkflowTasks | button | View Workflow Tasks |
| FileStartWorkflow | button | Workflows |
| SignatureLineInsert | button | Signature Line |
| LabelInsert | button | Label |
| BarcodeInsert | button | Barcode |
| ViewPageLayoutView | toggleButton | Page Layout |
| PivotClearAll | button | Clear All |
| ChartStylesGallery | gallery | Quick Styles |
| ChartLayoutGallery | gallery | Quick Layout |
| ChartSaveTemplates | button | Save As Template |
| ChartAxisTitles | menu | Axis Titles |
| ChartAxes | menu | Axes |
| ChartGridlines | menu | Gridlines |
| ChartFormatSelection | button | Format Selection |
| ChartElementSelector | comboBox | Chart Elements |
| PageMarginsGallery | gallery | Margins |
| TabPictureToolsFormat | tab | Format |
| TabDrawingToolsFormat | tab | Format |
| ShapesInsertGallery | gallery | Shapes |
| ShapeChangeShapeGallery | gallery | Change Shape |
| ShapeFillTextureGallery | gallery | Texture |
| ShapeStylesGallery | gallery | Quick Styles |
| PageOrientationGallery | gallery | Orientation |
| FileServerTasksMenu | menu | Server |
| FileSendMenu | menu | Send |
| TabInsert | tab | Insert |
| TabPageLayoutExcel | tab | Page Layout |
| TabView | tab | View |
| GroupFont | group | Font |
| GroupStyles | group | Styles |
| GroupProofing | group | Proofing |
| GroupInsertIllustrations | group | Illustrations |
| GroupShapes | group | Insert Shapes |
| GroupPageSetup | group | Page Setup |
| GroupComments | group | Comments |
| GroupPictureSize | group | Size |
| GroupDrawBorders | group | Draw Borders |
| GroupTableProperties | group | Properties |
| GroupTableTools | group | Tools |
| GroupArrange | group | Arrange |
| GroupShapeStyles | group | Shape Styles |
| TabFormulas | tab | Formulas |
| TabData | tab | Data |
| TabReview | tab | Review |
| GroupChartLayouts | group | Chart Layouts |
| GroupChartStyles | group | Chart Styles |
| GroupChartAxes | group | Axes |
| GroupChartShapes | group | Insert |
| GroupNumber | group | Number |
| GroupAlignmentExcel | group | Alignment |
| GroupCells | group | Cells |
| GroupSortFilter | group | Sort & Filter |
| GroupInsertTablesExcel | group | Tables |
| GroupPageLayoutScaleToFit | group | Scale to Fit |
| GroupPageLayoutSheetOptions | group | Sheet Options |
| GroupFunctionLibrary | group | Function Library |
| GroupNamedCells | group | Defined Names |
| GroupFormulaAuditing | group | Formula Auditing |
| GroupGetExternalData | group | Get External Data |
| GroupConnections | group | Connections |
| GroupOutline | group | Outline |
| GroupDataTools | group | Data Tools |
| GroupChangesExcel | group | Changes |
| ObjectEffectPresetGallery | gallery | Preset |
| PictureEffectsPresetGallery | gallery | Preset |
| \_3DRotationGallery | gallery | 3-D Rotation |
| TabSmartArtToolsDesign | tab | Design |
| TabSmartArtToolsFormat | tab | Format |
| TabChartToolsDesign | tab | Design |
| TabChartToolsLayout | tab | Layout |
| TabChartToolsFormat | tab | Format |
| ShapeFillColorPicker | gallery | Shape Fill |
| OutlineColorPicker | gallery | Picture Border |
| FileDocumentInspect | button | Inspect Document |
| AlignLeftToRightMenu | splitButton | Left-to-Right |
| GroupControls | group | Controls |
| GroupZoom | group | Zoom |
| ArrowStyleGallery | gallery | Arrows |
| OutlineDashesGallery | gallery | Dashes |
| OutlineWeightGallery | gallery | Weight |
| GroupPictureTools | group | Adjust |
| GroupSize | group | Size |
| FormatCellsNumberDialog | button | Format Cell Number |
| FormatCellsFontDialog | button | Format Cell Font |
| CellAlignmentOptions | button | Format Cell Alignment |
| PageSetupPageDialog | button | Page Setup |
| PageSetupSheetDialog | button | Sheet Options |
| TabPivotTableToolsOptions | tab | Options |
| PivotTableLayoutGrandTotals | menu | Grand Totals |
| TabPivotTableToolsDesign | tab | Design |
| GroupPivotTableActiveField | group | Active Field |
| GroupPivotTableLayout | group | Layout |
| GroupPivotTableSort | group | Sort |
| GroupPivotTableShowHide | group | Show/Hide |
| PivotTableLayoutSubtotals | menu | Subtotals |
| GroupPivotTableGroup | group | Group |
| GroupPivotTableTools | group | Tools |
| GroupPivotTableData | group | Data |
| GroupPivotTableOptions | group | PivotTable |
| GroupPivotTableStyles | group | PivotTable Styles |
| GroupPivotTableStyleOptions | group | PivotTable Style Options |
| WrapText | toggleButton | Wrap Text |
| ClearMenu | menu | Clear |
| ReviewTrackChangesMenu | menu | Track Changes |
| ObjectAlignMenu | menu | Align |
| ObjectRotateGallery | gallery | Rotate |
| FillMenu | menu | Fill |
| OrientationMenu | menu | Orientation |
| MergeCenterMenu | splitButton | Merge |
| AutoSumMenu | splitButton | AutoSum |
| PrintAreaMenu | menu | Print Area |
| PageBreakMenu | menu | Breaks |
| NameDefineMenu | splitButton | Define Name |
| RefreshMenu | splitButton | Refresh |
| WhatIfAnalysisMenu | menu | What-If Analysis |
| PivotTableFormulasMenu | menu | Formulas |
| PivotTableOlapTools | menu | OLAP tools |
| PivotTableOptionsMenu | splitButton | Table Options |
| ErrorCheckingMenu | splitButton | Error Checking |
| TraceRemoveArrowsMenu | splitButton | Remove Arrows |
| SortFilterMenu | menu | Sort & Filter |
| FontColorPicker | gallery | Font Color |
| CellFillColorPicker | gallery | Shading |
| BorderDoubleBottom | button | Bottom Double Border |
| TabHome | tab | Home |
| ChartTitle | gallery | Chart Title |
| ChartPrimaryHorizontalAxisTitle | gallery | Primary Horizontal Axis Title |
| ChartPrimaryVerticalAxisTitle | gallery | Primary Vertical Axis Title |
| ChartDepthAxisTitle | gallery | Depth Axis Title |
| ChartLegend | gallery | Legend |
| ChartDataLabel | gallery | Data Labels |
| ChartPrimaryHorizontalGridlines | gallery | Primary Horizontal Gridlines |
| ChartPrimaryVerticalGridlines | gallery | Primary Vertical Gridlines |
| ChartDepthGridlines | gallery | Depth Gridlines |
| ChartPrimaryHorizontalAxis | gallery | Primary Horizontal Axis |
| ChartPrimaryVerticalAxis | gallery | Primary Vertical Axis |
| ChartDepthAxis | gallery | Depth Axis |
| ChartDataTable | gallery | Data Table |
| ChartTrendline | gallery | Trendline |
| ChartErrorBars | gallery | Error Bars |
| ChartLines | gallery | Lines |
| ChartUpDownBars | gallery | Up/Down Bars |
| ChartPlotArea | gallery | Plot Area |
| ChartWall | gallery | Chart Wall |
| ChartFloor | gallery | Chart Floor |
| SmartArtAddShape | button | Add Shape |
| SmartArtLargerShape | button | Larger |
| SmartArtSmallerShape | button | Smaller |
| SmartArtResetGraphic | button | Reset Graphic |
| SmartArtTextPane | toggleButton | Text Pane |
| SmartArtEditIn2D | toggleButton | Edit in 2-D |
| SmartArtLayoutGallery | gallery | Change Layout |
| SmartArtMoreLayoutsDialog | button | More Layouts... |
| SmartArtStylesGallery | gallery | Quick Styles |
| SmartArtChangeColorsGallery | gallery | Change Colors |
| ObjectEffectSoftEdgesGallery | gallery | Soft Edges |
| ObjectEffectGlowGallery | gallery | Glow |
| GradientGallery | gallery | Gradient |
| ObjectEffectShadowGallery | gallery | Shadow |
| WordArtInsertGallery | gallery | WordArt |
| TextEffectTransformGallery | gallery | Transform |
| TabHeaderAndFooterToolsDesign | tab | Design |
| HeaderFooterHeaderGallery | gallery | Header |
| HeaderFooterFooterGallery | gallery | Footer |
| GroupHeaderFooterElements | group | Header & Footer Elements |
| HeaderFooterPageNumberInsertExcel | button | Page Number |
| HeaderFooterNumberOfPagesInsert | button | Number of Pages |
| HeaderFooterCurrentDate | button | Current Date |
| HeaderFooterCurrentTimeInsert | button | Current Time |
| HeaderFooterFilePathInsert | button | File Path |
| HeaderFooterFileNameInsert | button | File Name |
| HeaderFooterSheetNameInsert | button | Sheet Name |
| HeaderFooterAlignMargins | checkBox | Align with Page Margins |
| HeaderFooterPictureInsert | button | Picture |
| HeaderFooterFormatPicture | button | Format Picture |
| GroupHeaderFooterOptions | group | Options |
| HeaderFooterDifferentOddEvenPageExcel | checkBox | Different Odd & Even Pages |
| HeaderFooterDifferentFirstPageExcel | checkBox | Different First Page |
| HeaderFooterScaleWithDocument | checkBox | Scale with Document |
| SheetTabColorGallery | gallery | Tab Color |
| FontShadingColorMoreColorsDialog | button | More Colors... |
| FontColorMoreColorsDialogExcel | button | More Colors... |
| BorderMoreColorsDialog | button | More Colors... |
| SheetTabColorMoreColorsDialog | button | More Colors... |
| PivotTableNewStyle | button | New PivotTable Style... |
| PivotPlusMinusFieldHeadersShowHide | toggleButton | Field Headers |
| PivotTableExpandField | button | Expand Entire Field |
| PivotCollapseField | button | Collapse Entire Field |
| ConditionalFormattingDataBarsGallery | gallery | Data Bars |
| ConditionalFormattingColorScalesGallery | gallery | Color Scales |
| ConditionalFormattingIconSetsGallery | gallery | Icon Sets |
| ConditionalFormattingDataBarsMoreOptions | button | More Rules... |
| ConditionalFormattingColorScalesMore | button | More Rules... |
| ConditionalFormattingIconSetsMore | button | More Rules... |
| TableColumnsInsertRightExcel | button | Insert Table Column to the Right |
| TableRowsInsertBelowExcel | button | Insert Table Row Below |
| ConditionalFormattingHighlightBetween | button | Between... |
| ConditionalFormattingClearSelectedCells | button | Clear Rules from Selected Cells |
| ConditionalFormattingClearSheet | button | Clear Rules from Entire Sheet |
| ConditionalFormattingClearTable | button | Clear Rules from This Table |
| ConditionalFormattingClearPivotTable | button | Clear Rules from This PivotTable |
| PivotTableStylesGallery | gallery | Quick Styles |
| FormatAsTableGallery | gallery | Format as Table |
| TableStylesGalleryExcel | gallery | Quick Styles |
| ConditionalFormattingsManage | button | Manage Rules... |
| ConditionalFormattingHighlightGreaterThan | button | Greater Than... |
| ConditionalFormattingHighlightLessThan | button | Less Than... |
| ConditionalFormattingHighlightEqualTo | button | Equal To... |
| ConditionalFormattingHighlightTextContaining | button | Text that Contains... |
| ConditionalFormattingHighlightDateOccuring | button | A Date Occurring... |
| ConditionalFormattingHighlightDuplicateValues | button | Duplicate Values... |
| ConditionalFormattingTopNItems | button | Top 10 Items... |
| ConditionalFormattingTopNPercent | button | Top 10 %... |
| ConditionalFormattingBottomNItems | button | Bottom 10 Items... |
| ConditionalFormattingBottomNPercent | button | Bottom 10 %... |
| ConditionalFormattingAboveAverage | button | Above Average... |
| ConditionalFormattingBelowAverage | button | Below Average... |
| RemoveDuplicates | button | Remove Duplicates |
| FilterReapply | button | Reapply |
| ThemeColorsGallery | gallery | Colors |
| PivotTableInsert | button | PivotTable |
| PivotChartInsert | button | PivotChart |
| PivotTableMove | button | Move PivotTable |
| PivotTableChangeDataSource | button | Change Data Source... |
| GroupSmartArtLayouts | group | Layouts |
| GroupSmartArtQuickStyles | group | SmartArt Styles |
| GroupSmartArtCreateGraphic | group | Create Graphic |
| GroupSmartArtReset | group | Reset |
| GroupSmartArtSize | group | Size |
| ConditionalFormattingHighlightCellsMenu | menu | Highlight Cells Rules |
| ConditionalFormattingTopBottomMenu | menu | Top/Bottom Rules |
| FormatCellsMenu | menu | Format |
| ConditionalFormattingClearMenu | menu | Clear Rules |
| ThemeSearchOfficeOnline | button | More Themes on Microsoft Office Online... |
| FontColorCycle | button | Color |
| TabAddIns | tab | Add-Ins |
| CellsInsertSmart | button | Insert Cells |
| CellsDeleteSmart | button | Delete Cells... |
| ObjectEditShapeMenu | menu | Edit Shape |
| PivotTableLayoutReportLayout | menu | Report Layout |
| PivotTableLayoutShowInCompactForm | button | Show in Compact Form |
| PivotTableLayoutShowInOutlineForm | button | Show in Outline Form |
| PivotTableLayoutShowInTabularForm | button | Show in Tabular Form |
| PivotTableClearMenu | menu | Clear |
| ConditionalFormattingTopBottomMore | button | More Rules... |
| ConditionalFormattingHighlightRulesMore | button | More Rules... |
| CellStylesGallery | gallery | Cell Styles |
| CellStyleNew | button | New Cell Style... |
| CellStylesMerge | button | Merge Styles... |
| TableStyleNew | button | New Table Style... |
| TableStyleClear | button | Clear |
| TableStyleHeaderRow | checkBox | Header Row |
| FilePublishExcelServices | button | Excel Services |
| PivotTableOlapConvertToFormulas | button | Convert to Formulas |
| PivotTableLayoutBlankRows | menu | Blank Rows |
| TableStyleFirstColumn | checkBox | First Column |
| TableStyleLastColumn | checkBox | Last Column |
| TableStyleBandedRows | checkBox | Banded Rows |
| TableStyleBandedColumns | checkBox | Banded Columns |
| TableStyleRowHeaders | checkBox | Row Headers |
| TableStyleColumnHeaders | checkBox | Column Headers |
| TableSummarizeWithPivot | button | Summarize with PivotTable |
| ConnectionProperties | button | Connection Properties... |
| PivotClearFilters | button | Clear Filters |
| GetExternalDataFromAccess | button | From Access |
| GetExternalDataFromOtherSources | gallery | From Other Sources |
| GetExternalDataExistingConnections | button | Existing Connections |
| GroupThemesExcel | group | Themes |
| PivotPlusMinusButtonsShowHide | toggleButton | +/- Buttons |
| FileSaveAsPdfOrXps | button | Publish as PDF or XPS |
| FileSaveAsExcelOpenDocumentSpreadsheet | button | OpenDocument Spreadsheet |
| MoreControlsDialog | button | More Controls... |
| GroupCode | group | Code |
| TabDeveloper | tab | Developer |
| GroupXml | group | XML |
| PageScaleToFitWidth | comboBox | Width: |
| PageScaleToFitHeight | comboBox | Height: |
| SelectMenuExcel | menu | Find & Select |
| GoToSpecial | button | Go To Special... |
| GoToFormulas | button | Formulas |
| GoToComments | button | Comments |
| GoToConditionalFormatting | button | Conditional Formatting |
| GoToConstants | button | Constants |
| GoToDataValidation | button | Data Validation |
| PrintTitles | button | Print Titles |
| NameUseInFormula | gallery | Use in Formula |
| CalculationOptionsMenu | menu | Calculation Options |
| CalculationOptionsManuallly | toggleButton | Manual |
| CalculationOptionsAutomatically | toggleButton | Automatic |
| CalculationOptionsAutomaticallyExceptDataTables | toggleButton | Automatic Except for Data Tables |
| XmlSource | toggleButton | Source... |
| GroupClipboard | group | Clipboard |
| GroupInsertLinks | group | Links |
| GroupInsertBarcode | group | Barcode |
| GroupCalculation | group | Calculation |
| BordersGallery | splitButton | Borders |
| BordersMoreDialog | button | More Borders... |
| PageScaleToFitOptionsDialog | button | More Pages... |
| PageBreakInsertExcel | button | Insert Page Break |
| PageBreakRemove | button | Remove Page Break |
| PageSizeGallery | gallery | Size |
| ObjectPictureFill | button | Picture... |
| PivotTableGroupSelection | button | Group Selection |
| PivotTableGroupField | button | Group Field |
| WindowSwitchWindowsMenuExcel | menu | Switch Windows |
| ThemeColorsCreateNew | button | Create New Theme Colors... |
| ThemeFontsCreateNew | button | Create New Theme Fonts... |
| ShapeFillMoreGradientsDialog | button | More Gradients... |
| ShadowOptionsDialog | button | Shadow Options... |
| MarginsCustomMargins | button | Custom Margins... |
| FunctionsRecentlyUsedtInsertGallery | gallery | Recently Used |
| FunctionsFinancialInsertGallery | gallery | Financial |
| FunctionsDateTimeInsertGallery | gallery | Date & Time |
| FunctionsMathTrigInsertGallery | gallery | Math & Trig |
| FunctionsTextInsertGallery | gallery | Text |
| FunctionsLogicalInsertGallery | gallery | Logical |
| FunctionsStatisticalInsertGallery | gallery | Statistical |
| FunctionsLookupReferenceInsertGallery | gallery | Lookup & Reference |
| FunctionsInformationInsertGallery | gallery | Information |
| TabTableToolsDesignExcel | tab | Design |
| GroupTableStyleOptions | group | Table Style Options |
| GroupTableExternalData | group | External Table Data |
| GroupEditingExcel | group | Editing |
| FileCompatibilityChecker | button | Run Compatibility Checker |
| ThemeFontsGallery | gallery | Fonts |
| ThemeEffectsGallery | gallery | Effects |
| FileProperties | toggleButton | Properties |
| TabPrintPreview | tab | Print Preview |
| GroupPrintPreviewPrint | group | Print |
| GroupPrintPreviewPreview | group | Preview |
| SortDialog | button | Sort... |
| SortAscendingExcel | button | Sort Ascending |
| SortDescendingExcel | button | Sort Descending |
| SortCustomExcel | button | Custom Sort... |
| FileCreateDocumentWorkspace | toggleButton | Create Document Workspace |
| FileSaveToDocumentManagementServer | button | Document Management Server |
| FileDocumentManagementInformation | toggleButton | Document Management Information |
| QuickAccessToolbarCustomization | button | Customize Quick Access Toolbar... |
| FilePrepareMenu | menu | Prepare |
| FileMarkAsFinal | toggleButton | Mark as Final |
| FileAddDigitalSignature | button | Add a Digital Signature |
| SignatureServicesAdd | button | Add Signature Services... |
| TextBoxInsertMenu | splitButton | Text Box |
| TextBoxInsertHorizontal | toggleButton | Horizontal Text Box |
| ControlProperties | button | Properties |
| ViewCode | button | View Code |
| GroupHeaderFooterNavigation | group | Navigation |
| ShadowStyleGalleryClassic | gallery | Shadow Effects |
| GroupInkSelect | group | Select |
| NumberFormatGallery | comboBox | Number Format |
| NumberFormatsDialog | button | More Number Formats... |
| PageSizeMorePaperSizesDialogExcel | button | More Paper Sizes... |
| PictureBrightnessGallery | gallery | Brightness |
| PictureContrastGallery | gallery | Contrast |
| SmartArtAddShapeAfter | button | Add Shape After |
| SmartArtAddShapeBefore | button | Add Shape Before |
| SmartArtAddShapeAbove | button | Add Shape Above |
| SmartArtAddShapeBelow | button | Add Shape Below |
| SmartArtAddAssistant | button | Add Assistant |
| ChartSwitchRowColumn | button | Switch Row/Column |
| ChartChangeType | button | Change Chart Type... |
| GroupChartData | group | Data |
| GroupChartLocation | group | Location |
| GroupChartType | group | Type |
| \_3DRotationOptionsDialog | button | 3-D Rotation Options... |
| \_3DBevelOptionsDialog | button | 3-D Options... |
| SelectionPane | toggleButton | Selection Pane... |
| SmartArtOrganizationChartLeftHanging | button | Left Hanging |
| SmartArtOrganizationChartRightHanging | button | Right Hanging |
| SmartArtOrganizationChartBoth | button | Both |
| SmartArtOrganizationChartStandard | button | Standard |
| SmartArtRightToLeft | toggleButton | Right to Left |
| ViewMessageBar | checkBox | Message Bar |
| NameManager | button | Name Manager |
| NameDefine | button | Define Name... |
| AccountingFormatExcel | button |  |
| AccountingFormatMoreExcel | button | More Accounting Formats... |
| ShapeStylesOtherThemeFillsGallery | gallery | Other Theme Fills |
| SmartArtOrganizationChartMenu | menu | Layout |
| \_3DEffectsGalleryClassic | gallery | 3-D Effects |
| \_3DDirectionGalleryClassic | gallery | Direction |
| \_3DLightingGalleryClassic | gallery | Lighting |
| GroupShadowEffects | group | Shadow Effects |
| Group3DEffects | group | 3-D Effects |
| ShadowColorPickerClassic | gallery | Shadow Color |
| \_3DEffectColorPickerClassic | gallery | 3-D Color |
| ControlsGallery | gallery | Insert |
| GoToHeader | button | Go to Header |
| GoToFooter | button | Go to Footer |
| UnderlineGallery | splitButton | Underline |
| GroupAddInsMenuCommands | group | Menu Commands |
| GroupAddInsToolbarCommands | group | Toolbar Commands |
| ViewFreezePanesGallery | gallery | Freeze Panes |
| GroupInk | group | Ink |
| TabInkToolsPens | tab | Pens |
| GroupInkPens | group | Pens |
| GroupInkClose | group | Close |
| InkBallpointPen | toggleButton | Ballpoint Pen |
| InkFeltTipPen | toggleButton | Felt Tip Pen |
| InkHighlighter | toggleButton | Highlighter |
| PhoneticGuideMenu | splitButton | Phonetic Guide |
| MarginsShowHide | checkBox | Show Margins |
| \_3DSurfaceMaterialGalleryClassic | gallery | Surface |
| \_3DExtrusionDepthGalleryClassic | gallery | Depth |
| GroupHeaderFooter | group | Header & Footer |
| FunctionsCubeInsertGallery | gallery | Cube |
| FunctionsEngineeringInsertGallery | gallery | Engineering |
| ThemeSaveCurrent | button | Save Current Theme... |
| ThemesGallery | gallery | Themes |
| ChartResetToMatchStyle | button | Reset to Match Style |
| Chart3DView | button | 3-D Rotation... |
| ObjectSizeAndPropertiesDialog | button | Size and Properties... |
| ShapeConvertToFreeform | button | Convert to Freeform |
| ReflectionGallery | gallery | Reflection |
| PictureRecolorGallery | gallery | Recolor |
| SmartArtPromote | button | Promote |
| SmartArtDemote | button | Demote |
| TabPivotChartToolsAnalyze | tab | Analyze |
| GroupPivotChartShowOrHide | group | Show/Hide |
| GroupPivotChartData | group | Data |
| GroupPivotChartActiveField | group | Active Field |
| PivotChartFilterShow | toggleButton | PivotChart Filter |
| ChartTitleOptionsDialog | button | More Title Options... |
| ChartLegendOptionsDialogDialog | button | More Legend Options... |
| ChartDataLabelDialog | button | More Data Label Options... |
| ChartPrimaryHorizontalAxisTitleOptionsDialog | button | More Primary Horizontal Axis Title Options... |
| ChartPrimaryVerticalAxisTitleOptionsDialog | button | More Primary Vertical Axis Title Options... |
| ChartSecondaryHorizontalAxisTitleOptionsDialog | button | More Secondary Horizontal Axis Title Options... |
| ChartSecondaryVerticalAxisTitleOptionsDialog | button | More Secondary Vertical Axis Title Options... |
| ChartDepthAxisTitleOptionsDialog | button | More Depth Axis Title Options... |
| ChartPrimaryHorizontalGridlinesOptionsDialog | button | More Primary Horizontal Gridlines Options... |
| ChartPrimaryVerticalGridlinesOptionsDialog | button | More Primary Vertical Gridlines Options... |
| ChartSecondaryHorizontalGridlinesOptionsDialog | button | More Secondary Horizontal Gridlines Options... |
| ChartSecondaryVerticalGridlinesOptionsDialog | button | More Secondary Vertical Gridlines Options... |
| ChartDepthGridlinesOptionsDialog | button | More Depth Gridlines Options... |
| ChartPrimaryHorizontalAxisOptionsDialog | button | More Primary Horizontal Axis Options... |
| ChartPrimaryVerticalAxisOptionsDialog | button | More Primary Vertical Axis Options... |
| ChartSecondaryHorizontalAxisOption | button | More Secondary Horizontal Axis Options... |
| ChartSecondaryVerticalAxisOptionsDialog | button | More Secondary Vertical Axis Options... |
| ChartDepthAxisOptionsDialog | button | More Depth Axis Options... |
| ChartDataTableOptionsDialog | button | More Data Table Options... |
| ChartTrendlineOptionsDialog | button | More Trendline Options... |
| ChartErrorBarsOptionsDialog | button | More Error Bars Options... |
| ChartUpDownBarsOptionsDialog | button | More Up/Down Bars Options... |
| ChartPlotAreaOptionsDialog | button | More Plot Area Options... |
| ChartWallOptionsDialog | button | More Walls Options... |
| ChartFloorOptionsDialog | button | More Floor Options... |
| ChartSecondaryHorizontalAxisTitle | gallery | Secondary Horizontal Axis Title |
| ChartSecondaryVerticalAxisTitle | gallery | Secondary Vertical Axis Title |
| ChartSecondaryHorizontalGridlines | gallery | Secondary Horizontal Gridlines |
| ChartSecondaryVerticalGridlines | gallery | Secondary Vertical Gridlines |
| ChartSecondaryHorizontalAxis | gallery | Secondary Horizontal Axis |
| ChartSecondaryVerticalAxis | gallery | Secondary Vertical Axis |
| GroupAddInsCustomToolbars | group | Custom Toolbars |
| ConditionalFormattingMenu | menu | Conditional Formatting |
| TabPivotChartToolsDesign | tab | Design |
| TabPivotChartToolsLayout | tab | Layout |
| TabPivotChartToolsFormat | tab | Format |
| ObjectBringToFrontMenu | splitButton | Bring to Front |
| ObjectSendToBackMenu | splitButton | Send to Back |
| ObjectsGroupMenu | menu | Group |
| SignatureLineInsertMenu | splitButton | Signature Line |
| FileSaveAsExcel97\_2003 | button | Excel 97-2003 Workbook |
| TextBoxInsertExcel | toggleButton | Text Box |
| FileSaveAsMenu | splitButton | Save As Other Format |
| FilePrintMenu | splitButton | Preview and Print |
| FilePermissionRestrictMenu | menu | Restrict Permission |
| InsertCellstMenu | splitButton | Insert |
| PivotTableInsertMenu | splitButton | PivotTable |
| OutlineGroupMenu | splitButton | Group |
| OutlineUngroupMenu | splitButton | Ungroup |
| FormulaMoreFunctionsMenu | menu | More Functions |
| DocumentPanelTemplate | button | Document Panel |
| GroupModify | group | Modify |
| ViewGridlinesToggleExcel | toggleButton | View Gridlines |
| BevelShapeGallery | gallery | Bevel |
| \_3DBevelPictureTopGallery | gallery | Bevel |
| EditLinks | button | Edit Links |
| GlowColorPicker | gallery | More Glow Colors |
| RecolorColorPicker | gallery | More Variations |
| GlowColorMoreColorsDialog | button | More Colors... |
| PictureRecolorMoreColorsDialog | button | More Colors... |
| SmartArtAddBullet | button | Add Bullet |
| PictureChange | button | Change Picture... |
| GroupWordArtStyles | group | WordArt Styles |
| TextFillColorPicker | gallery | Text Fill |
| TextOutlineColorPicker | gallery | Text Outline |
| TextOutlineColorMoreColorsDialog | button | More Outline Colors... |
| TextEffectsMenu | menu | Text Effects |
| TextStylesGallery | gallery | Quick Styles |
| WordArtClear | button | Clear WordArt |
| TextPictureFill | button | Picture... |
| TextFillGradientGallery | gallery | Gradient |
| TextFillMoreGradientsDialog | button | More Gradients... |
| TextFillTextureGallery | gallery | Texture |
| TextOutlineDashesGallery | gallery | Dashes |
| TextOutlineMoreLinesDialog | button | More Lines... |
| TextOutlineWeightGallery | gallery | Weight |
| TextEffectShadowGallery | gallery | Shadow |
| TextEffectsMoreShadowsDialog | button | Shadow Options... |
| TextEffectsBevelMore3DOptionsDialog | button | 3-D Options... |
| TextEffects3DRotationGallery | gallery | 3-D Rotation |
| TextEffects3DRotationOptionsDialog | button | 3-D Rotation Options... |
| TextEffectGlowGallery | gallery | Glow |
| TextGlowColorPicker | gallery | More Glow Colors |
| TextGlowColorMoreColorsDialog | button | More Colors... |
| TextReflectionGallery | gallery | Reflection |
| ShapeEffectsMenu | menu | Shape Effects |
| PivotTableSubtotalsDoNotShow | button | Do Not Show Subtotals |
| PivotTableSubtotalsOnBottom | button | Show all Subtotals at Bottom of Group |
| PivotTableSubtotalsOnTop | button | Show all Subtotals at Top of Group |
| PivotTableGrandTotalsOffForRowsAndColumns | button | Off for Rows and Columns |
| PivotTableGrandTotalsOnForRowsAndColumns | button | On for Rows and Columns |
| PivotTableGrandTotalsOnForRowsOnly | button | On for Rows Only |
| PivotTableGrandTotalsOnForColumnsOnly | button | On for Columns Only |
| PivotTableBlankRowsInsert | button | Insert Blank Line after Each Item |
| PivotTableBlankRowsRemove | button | Remove Blank Line after Each Item |
| GroupChartCurrentSelection | group | Current Selection |
| GroupChartLabels | group | Labels |
| AlignTopExcel | toggleButton | Top Align |
| AlignMiddleExcel | toggleButton | Middle Align |
| AlignBottomExcel | toggleButton | Bottom Align |
| BevelTextGallery | gallery | Bevel |
| PictureCorrectionsDialog | button | Picture Corrections Options... |
| GroupTableStylesExcel | group | Table Styles |
| ConditionalFormattingNewRule | button | New Rule... |
| SmartArtAddShapeSplitMenu | splitButton | Add Shape Options |
| ViewRulerExcel | checkBox | Ruler |
| GroupInkFormat | group | Format |
| InkColorPicker | gallery | Color |
| BorderColorPickerExcel | gallery | Line Color |
| GroupSmartArtShapes | group | Shapes |
| GroupInsertText | group | Text |
| ShapeOutlineColorPicker | gallery | Picture Border |
| TableDeleteRowsAndColumnsMenu | splitButton | Delete |
| AccountingFormatMenu | splitButton |  |
| GroupChartProperties | group | Properties |
| PivotTableEditDataSource | splitButton | Edit Data Source |
| FileExcelServicesOptions | button | Excel Services Options |
| TableExportMenu | menu | Export |
| TableExportTableToVisioPivotDiagram | button | Export Table to Visio PivotDiagram... |
| PasteMenu | splitButton | Paste |
| GroupPictureStyles | group | Picture Styles |
| PictureStylesGallery | gallery | Quick Styles |
| GroupInsertChartsExcel | group | Charts |
| ChartTypeColumnInsertGallery | gallery | Column |
| ChartTypeLineInsertGallery | gallery | Line |
| ChartTypePieInsertGallery | gallery | Pie |
| ChartTypeBarInsertGallery | gallery | Bar |
| ChartTypeAreaInsertGallery | gallery | Area |
| ChartTypeXYScatterInsertGallery | gallery | Scatter |
| ChartTypeOtherInsertGallery | gallery | Other Charts |
| ChartTypeAllInsertDialog | button | All Chart Types... |
| PivotChartClearMenu | menu | Clear |
| PictureEffectsShadowGallery | gallery | Shadow |
| PictureEffectsGlowGallery | gallery | Glow |
| PictureEffectsSoftEdgesGallery | gallery | Soft Edges |
| PictureReflectionGallery | gallery | Reflection |
| PictureRotationGallery | gallery | 3-D Rotation |
| InkToolsClose | button | Close Ink Tools |
| SheetRowsInsert | button | Insert Sheet Rows |
| SheetColumnsInsert | button | Insert Sheet Columns |
| SheetRowsDelete | button | Delete Sheet Rows |
| SheetColumnsDelete | button | Delete Sheet Columns |
| GroupViewShowHide | group | Show/Hide |
| GroupWindow | group | Window |
| GroupWorkbookViews | group | Workbook Views |
| ViewHeadings | checkBox | View |
| RefreshAllMenu | splitButton | Refresh |
| HideAndUnhideMenu | menu | Hide & Unhide |
| DataValidationMenu | splitButton | Data Validation |
| FileDocumentEncrypt | toggleButton | Encrypt Document |
| WordArtFormatDialog | button | Format Text Effects... |
| ObjectRotationOptionsDialog | button | More Rotation Options... |
| MoreTextureOptions | button | More Textures... |
| TextFillColorMoreColorsDialog | button | More Fill Colors... |
| BorderTopNoToggle | button | Top Border |
| BorderBottomNoToggle | button | Bottom Border |
| BorderLeftNoToggle | button | Left Border |
| BorderRightNoToggle | button | Right Border |
| WindowSplitToggle | toggleButton | Split |
| FileEmailAsPdfEmailAttachment | button | E-mail as PDF Attachment |
| FileEmailAsXpsEmailAttachment | button | E-mail as XPS Attachment |
| GroupPrintPreviewZoom | group | Zoom |
| PictureEffectsMenu | menu | Picture Effects |
| PictureShapeGallery | gallery | Change Shape |
| GroupChartBackground | group | Background |
| GroupChartAnalysis | group | Analysis |
| ZoomToSelection | button | Zoom to Selection |
| GridlinesExcel | checkBox | View |
| UnmergeCells | button | Split Cells... |
| MenuPublish | menu | Publish |
| ViewSideBySide | toggleButton | View Side by Side |
| FileSaveAsOtherFormats | button | Save As |
| FileSaveAsExcelXlsx | button | Excel Workbook |
| FileSaveAsExcelXlsxMacro | button | Excel Macro-Enabled Workbook |
| FileSaveAsExcelXlsb | button | Excel Binary Workbook |
| PasteAsPictureMenu | menu | As Picture |
| GroupPivotActions | group | Actions |
| PivotTableSelectFlyout | menu | Select |
| ZoomCurrent100 | button | 100% |
| TextFillMoreTextures | button | More Textures... |
| GroupMacros | group | Macros |
| PlayMacro | button | Macros |
| MenuMacros | splitButton | Macros |
| AdvertisePublishAs | button | Find add-ins for other file formats |
| UpgradeWorkbook | button | Convert |
| ReviewProtectWorkbookMenu | menu | Protect Workbook |
| ReviewRestrictEditing | toggleButton | Protect Structure and Windows |
| AlternativeText | button | Size and Properties... |
| ThemeBrowseForThemes | button | Browse for Themes... |
| FileCheckOutDiscard | button | Discard Check Out |
| MdiChildSystemMenu | menu | Document |

### PowerPoint 2007

| idMso | Control Type | Label |
| --- | --- | --- |
| Spelling | button | Spelling... |
| FileSave | button | Save |
| FilePrint | button | Print |
| TableInsert | button | Insert Table... |
| ChartInsert | button | Chart... |
| FileNew | button | New |
| Copy | button | Copy |
| Cut | button | Cut |
| Paste | button | Paste |
| FileOpen | button | Open |
| Clear | button | Clear |
| Superscript | toggleButton | Superscript |
| Subscript | toggleButton | Subscript |
| FileClose | button | Close |
| FormatPainter | toggleButton | Format Painter |
| FilePrintPreview | toggleButton | Print Preview |
| PickUpStyle | button | Pick Up Style |
| PasteApplyStyle | button | Apply Style |
| Bold | toggleButton | Bold |
| Italic | toggleButton | Italic |
| Underline | toggleButton | Underline |
| AlignLeft | toggleButton | Align Left |
| AlignRight | toggleButton | Align Right |
| AlignCenter | toggleButton | Center |
| AlignJustify | toggleButton | Justify |
| Undo | gallery | Undo |
| Redo | gallery | Redo |
| OutlinePromote | button | Promote |
| OutlineDemote | button | Demote |
| OutlineMoveUp | button | Move Up |
| OutlineMoveDown | button | Move Down |
| OutlineExpand | button | Expand |
| OutlineCollapse | button | Collapse |
| TextBoxInsert | toggleButton | Text Box |
| FindDialog | button | Find... |
| BorderTop | toggleButton | Top Border |
| BorderBottom | toggleButton | Bottom Border |
| BorderLeft | toggleButton | Left Border |
| BorderRight | toggleButton | Right Border |
| BorderInside | toggleButton | Inside Borders |
| BorderOutside | toggleButton | Outside Borders |
| BorderNone | toggleButton | No Border |
| ObjectsGroup | button | Group |
| ObjectsUngroup | button | Ungroup |
| ObjectBringToFront | button | Bring to Front |
| ObjectSendToBack | button | Send to Back |
| ObjectBringForward | button | Bring Forward |
| ObjectSendBackward | button | Send Backward |
| ViewFullScreenView | button | Full Screen |
| ViewRulerPowerPoint | checkBox | Ruler |
| ObjectsSelect | toggleButton | Select Objects |
| MacroPlay | button | Macros |
| ObjectFlipHorizontal | button | Flip Horizontal |
| ObjectFlipVertical | button | Flip Vertical |
| ObjectRotateRight90 | button | Rotate Right 90° |
| ObjectRotateLeft90 | button | Rotate Left 90° |
| GroupDrawing | group | Drawing |
| ObjectEditPoints | toggleButton | Edit Points |
| GridSettings | button | Grid Settings... |
| PropertySheet | button | Property Sheet |
| OutlineShowTextFormatting | toggleButton | Show Text Formatting |
| Strikethrough | toggleButton | Strikethrough |
| WindowsArrangeAll | button | Arrange All |
| WindowNew | button | New Window |
| SymbolInsert | button | Symbol... |
| ReplaceDialog | button | Replace... |
| PagePrevious | button | Previous Page |
| PageNext | button | Next Page |
| TextBoxInsertVertical | toggleButton | Vertical Text Box |
| RedoOrRepeat | button | Redo |
| ObjectsRegroup | button | Regroup |
| FontSizeIncrease | button | Increase Font Size |
| FontSizeDecrease | button | Decrease Font Size |
| OleObjectctInsert | button | Object... |
| SnapToGrid | toggleButton | Snap to Grid |
| FindNext | button | Find Next |
| PasteDuplicate | button | Duplicate |
| SlideNew | button | New Slide |
| ClipArtInsert | toggleButton | Clip Art... |
| CreateHandoutsInWord | button | Create Handouts in Microsoft Office Word |
| Shadow | toggleButton | Shadow |
| ObjectRotateFree | button | Free Rotate |
| ShapesMoreShapes | button | More AutoShapes |
| SlideMasterMasterLayout | button | Slide Layout... |
| CollapseAll | button | Collapse All |
| OutlineExpandAll | button | Expand All |
| CombineCharacters | toggleButton | Yoko-Gumi |
| SlideHide | toggleButton | Hide Slide |
| AnimationCustom | toggleButton | Custom Animation... |
| PictureCrop | toggleButton | Crop |
| SlideShowRehearseTimings | button | Rehearse Timings |
| ViewSlideView | toggleButton | Slide |
| ViewOutlineView | toggleButton | Outline |
| ViewSlideSorterView | toggleButton | Slide Sorter |
| ViewNotesPageView | toggleButton | Notes Page |
| ViewSlideShowView | button | Slide Show |
| ViewSlideMasterView | toggleButton | Slide Master |
| FileSaveAs | button | Save As |
| AdvancedFileProperties | button | View Document Properties... |
| PasteSpecialDialog | button | Paste Special... |
| SelectAll | button | Select All |
| FileLinksToFiles | button | Edit Links to Files |
| HeaderFooterInsert | button | Header & Footer... |
| DateAndTimeInsert | button | Date & Time... |
| NumberInsert | button | Number... |
| BordersShadingDialog | button | Borders and Shading... |
| BulletsAndNumberingBulletsDialog | button | Bullets and Numbering... |
| SetLanguage | button | Set Language... |
| AutoCorrect | button | AutoCorrect Options... |
| MergeCells | button | Merge Cells |
| SplitCells | button | Split Cells... |
| TableRowSelect | button | Select Row |
| TableColumnSelect | button | Select Column |
| TableSelect | button | Select Table |
| ShowClipboard | button | Office Clipboard... |
| OutlookTaskCreate | button | Create Microsoft Office Outlook Task |
| WindowMinimize | button | Minimize |
| WindowRestore | button | Restore |
| WindowClose | button | Close |
| PrintPreviewClose | button | Close Print Preview |
| ZoomDialog | button | Zoom... |
| About | button | About |
| PictureInsertFromFilePowerPoint | button | Picture... |
| ExchangeFolder | button | Exchange Folder... |
| AddInManager | button | Add-Ins... |
| ChartEditDataSource | button | Select Data... |
| WindowMoreWindowsDialog | toggleButton | More Windows... |
| ObjectEditDialog | button | Object... |
| ObjectFormatDialog | button | Object... |
| Help | button | Help |
| WebGoBack | button | Back |
| WebGoForward | button | Forward |
| SmartArtInsert | button | SmartArt... |
| ShapeRerouteConnectors | toggleButton | Reroute Connectors |
| ObjectNudgeUp | button | Up |
| ObjectNudgeDown | button | Down |
| ObjectNudgeLeft | button | Left |
| ObjectNudgeRight | button | Right |
| ShapeStraightConnector | toggleButton | Straight Connector |
| ShapeElbowConnector | toggleButton | Elbow Connector |
| ObjectFillMoreColorsDialog | button | More Fill Colors... |
| ObjectBorderOutlineColorMoreColorsDialog | button | More Outline Colors... |
| LineStylesDialog | button | More Lines... |
| ArrowsMore | button | More Arrows... |
| WordArtVerticalText | button | Vertical Text |
| ContrastMore | button | More Contrast |
| ContrastLess | button | Less Contrast |
| BrightnessMore | button | More Brightness |
| BrightnessLess | button | Less Brightness |
| ShadowNudgeUpClassic | button | Nudge Shadow Up |
| ShadowNudgeDownClassic | button | Nudge Shadow Down |
| ShadowNudgeLeftClassic | button | Nudge Shadow Left |
| ShadowNudgeRightClassic | button | Nudge Shadow Right |
| ShapeRectangle | toggleButton | Rectangle |
| ShapeRoundedRectangle | toggleButton | Rounded Rectangle |
| ShapeIsoscelesTriangle | toggleButton | Isosceles Triangle |
| ShapeOval | toggleButton | Oval |
| ShapeLeftBrace | toggleButton | Left Brace |
| ShapeRightBrace | toggleButton | Right Brace |
| ShapeArc | toggleButton | Arc |
| ShapeRightArrow | toggleButton | Right Arrow |
| ShapeDownArrow | toggleButton | Down Arrow |
| ShapeRoundedRectangularCallout | toggleButton | Rounded Rectangular Callout |
| ShapeStar | toggleButton | 5-Point Star |
| PictureReset | button | Reset Picture |
| SnapToShapes | toggleButton | Snap to Shape |
| ViewVisualBasicCode | button | View Code |
| MasterViewClose | button | Close |
| HyperlinkInsert | button | Hyperlink... |
| ReviewNewComment | button | New Comment |
| VisualBasic | button | Visual Basic |
| BordersAll | toggleButton | All Borders |
| SlideBackgroundFormatDialog | button | Format Background... |
| TableDrawBorderPenStyle | dropDown | Pen Style |
| Font | comboBox | Font: |
| FontSize | comboBox | Font Size: |
| ZoomClassic | gallery | Zoom: |
| DocumentLocation | comboBox | Address: |
| InsertTab | button | Tab |
| WindowsCascade | button | Cascade |
| BorderInsideHorizontal | toggleButton | Inside Horizontal Border |
| BorderInsideVertical | toggleButton | Inside Vertical Border |
| BorderDiagonalDown | toggleButton | Diagonal Down Border |
| BorderDiagonalUp | toggleButton | Diagonal Up Border |
| TextDirectionLeftToRight | toggleButton | Left-to-Right |
| TextDirectionRightToLeft | toggleButton | Right-to-Left |
| ActiveXCheckBox | button | Check Box |
| ActiveXTextBox | button | Text Box |
| ActiveXButton | button | Command Button |
| ActiveXRadioButton | button | Option Button |
| ActiveXListBox | button | List Box |
| ActiveXComboBox | button | Combo Box |
| ActiveXToggleButton | button | Toggle Button |
| ActiveXSpinButton | button | Spin Button |
| ActiveXScrollBar | button | Scroll Bar |
| ActiveXLabel | button | Label |
| OleConvert | button | Convert... |
| ReviewEditComment | button | Edit Comment |
| TableDrawTable | toggleButton | Draw Table |
| TableEraser | toggleButton | Eraser |
| TableCellAlignTop | toggleButton | Align Top |
| TableCellAlignCenterVertically | toggleButton | Center Vertically |
| TableCellAlignBottom | toggleButton | Align Bottom |
| TableColumnsDistribute | button | Distribute Columns |
| TableRowsDistribute | button | Distribute Rows |
| ActiveXImage | button | Image |
| TableDeleteRows | button | Delete Rows |
| TableDeleteColumns | button | Delete Columns |
| ShadowOnOrOffClassic | button | Shadow On/Off |
| ObjectSetShapeDefaults | button | Set AutoShape Defaults |
| FileSendAsAttachment | button | E-mail |
| FileNewDefault | button | New |
| FilePrintQuick | button | Quick Print |
| PictureInsertFromFile | button | Picture... |
| TableDrawBorderPenWeight | dropDown | Pen Weight |
| TableShowGridlines | toggleButton | View Gridlines |
| TableBorderPenColorPicker | gallery | Pen Color |
| ShapeStraightConnectorArrow | toggleButton | Straight Arrow Connector |
| ShapeElbowConnectorArrow | toggleButton | Elbow Arrow Connector |
| SlideDelete | button | Delete |
| ViewHandoutMasterView | toggleButton | Handout Master |
| ViewNotesMasterView | toggleButton | Notes Master |
| SlidesReuseSlides | toggleButton | Reuse Slides... |
| SlidesFromOutline | button | Slides from Outline... |
| MovieFromClipOrganizerInsert | button | Movie from Organizer... |
| MovieFromFileInsert | button | Movie from File... |
| SoundInsertFromClipOrganizer | button | Sound from Clip Organizer... |
| SoundInsertFromFile | button | Sound from File... |
| CDAudioPlayTrack | button | Play CD Audio Track... |
| FontsReplaceFonts | button | Replace Fonts... |
| ChangeCaseToggle | button | Toggle Case |
| BlackAndWhiteAutomatic | toggleButton | B & W Automatic |
| BlackAndWhiteBlack | toggleButton | B & W |
| BlackAndWhiteBlackWithGrayscaleFill | toggleButton | B & W Black with Grayscale Fill |
| BlackAndWhiteBlackWithWhiteFill | toggleButton | B & W Black with White Fill |
| BlackAndWhiteDontShow | toggleButton | B & W Don't Show |
| BlackAndWhiteGrayWithWhiteFill | toggleButton | B & W Gray with White Fill |
| BlackAndWhiteGrayscale | toggleButton | B & W Grayscale |
| BlackAndWhiteInverseGrayscale | toggleButton | B & W Inverse Grayscale |
| BlackAndWhiteWhite | toggleButton | B & W White |
| BlackAndWhiteLightGrayscale | toggleButton | B & W Light Grayscale |
| RecordNarration | button | Record Narration |
| SlideShowSetUpDialog | button | Set Up Slide Show... |
| PasteAsHyperlink | button | Paste as Hyperlink |
| ParagraphDistributed | toggleButton | Distributed |
| CharacterFormattingReset | button | Reset Character Formatting |
| PictureSetTransparentColor | toggleButton | Set Transparent Color |
| SlideShowCustom | button | Custom Slide Show |
| WindowFitToPage | button | Fit to Page |
| SoundRecord | button | Record Sound... |
| DuplicateSelectedSlides | button | Duplicate Selected Slides |
| ActionInsert | button | Action |
| SlideShowPreviousSlide | button | Previous |
| SlideShowNextSlide | button | Next |
| LinkBreak | button | Break Link |
| LinkChange | button | Change Link |
| LinksUpdate | button | Update Links |
| BaselineLower | button | Lower Baseline |
| BaselineRaise | button | Raise Baseline |
| SlideShowInAWindow | button | Slide Show in a Window |
| HangulHanjaConversionPowerPoint | button | Hangul Hanja Conversion |
| SlidesPerPage2Slides | toggleButton | 2 Slides |
| SlidesPerPage3Slides | toggleButton | 3 Slides |
| SlidesPerPage6Slides | toggleButton | 6 Slides |
| SlidesPerPageSlideOutline | toggleButton | Slide Outline |
| GoToProperty | button | Go To... |
| IndentIncrease | button | Increase Indent |
| IndentDecrease | button | Decrease Indent |
| MoviePlay | button | Preview |
| SlidesPerPage4Slides | toggleButton | 4 Slides |
| SlidesPerPage9Slides | toggleButton | 9 Slides |
| MacroSecurity | button | Macro Security |
| WebPagePreview | button | Web Page Preview |
| TableInsertDialog | button | Table... |
| TableInsertRowsAbove | button | Insert Above |
| TableInsertRowsBelow | button | Insert Below |
| TableInsertColumnsLeft | button | Insert Left |
| TableInsertColumnsRight | button | Insert Right |
| ComAddInsDialog | button | COM Add-Ins... |
| ViewNormalViewPowerPoint | toggleButton | Normal |
| WebOptionsDialog | button | Web Options... |
| ViewDirectionLeftToRight | toggleButton | Left-to-Right |
| ViewDirectionRightToLeft | toggleButton | Right-to-Left |
| NextPane | button | Next Pane |
| WindowMoveSplit | button | Move Split |
| TableSetLeftToRight | toggleButton | Set Left-to-Right Table |
| TableSetRightToLeft | toggleButton | Set Right-to-Left Table |
| SlideShowResumeShow | button | Resume Slide Show |
| ObjectsMultiSelect | button | Select Multiple Objects |
| FontColorMoreColorsDialogPowerPoint | button | More Colors... |
| DrawingCanvasFit | button | Fit |
| DrawingCanvasResize | button | Resize |
| ViewBackToColorView | toggleButton | Back To Color View |
| ViewDisplayInGrayscale | toggleButton | Grayscale |
| ViewDisplayInHighContrast | toggleButton | High Contrast |
| ViewDisplayInPureBlackAndWhite | toggleButton | Pure Black and White |
| PrintWhat | dropDown | Print What |
| DrawingCanvasExpand | button | Expand |
| SlidesPerPage1Slide | toggleButton | 1 Slide |
| SlideMasterPreserveMaster | toggleButton | Preserve |
| SlideMasterRenameMaster | button | Rename |
| OutlineThumbnailsShowHide | button | Show Outline |
| BulletsAndNumberingNumberingDialog | button | Bullets and Numbering... |
| FileCheckOut | button | Check Out |
| FileCheckIn | button | Check In |
| PageOrientationLandscape | toggleButton | Landscape |
| PageOrientationPortrait | toggleButton | Portrait |
| SlideReset | button | Reset |
| PicturesCompress | button | Compress Pictures... |
| ReviewDeleteCommentPowerPoint | button | Delete |
| ReviewNextCommentPowerPoint | button | Next |
| ReviewPreviousCommentPowerPoint | button | Previous |
| ReviewShowOrHideMarkup | toggleButton | Show Markup |
| ReviewDeleteAllMarkupOnSlide | button | Delete All Markup on the Current Slide |
| Translate | button | Translate... |
| ViewGridlinesPowerPoint | checkBox | View Gridlines |
| PhotoAlbumInsert | button | New Photo Album... |
| PhotoAlbumEdit | button | Edit Photo Album... |
| DrawingCanvasScale | button | Scale Drawing |
| ResearchPane | toggleButton | Research... |
| FileInternetFax | button | Internet Fax |
| FilePackageForCD | button | Package for CD |
| FileVersionHistory | button | View Version History |
| InkColorMoreColorsDialog | button | More Ink Colors... |
| ContactUs | button | Contact Us... |
| FilePermissionUnrestricted | toggleButton | Unrestricted Access |
| FilePermissionDoNotDistribute | toggleButton | Restricted Access |
| FilePermissionView | button | View Permission |
| Thesaurus | button | Thesaurus... |
| InkingStart | button | Start Inking |
| CheckForUpdates | button | Check for Updates |
| InkEraser | toggleButton | Eraser |
| FilePermissionRestrictAs | button | Manage Credentials |
| ThemeBrowseForThemesPowerPoint | button | Browse for Themes... |
| SlideMasterInsertLayout | button | Insert Layout |
| FileViewDigitalSignatures | toggleButton | View Signatures |
| FileWorkflowTasks | button | View Workflow Tasks |
| FileStartWorkflow | button | Workflows |
| SlideThemesGallery | gallery | Themes |
| LabelInsert | button | Label |
| BarcodeInsert | button | Barcode |
| ChartStylesGallery | gallery | Quick Styles |
| ChartLayoutGallery | gallery | Quick Layout |
| ChartSaveTemplates | button | Save As Template |
| ChartAxisTitles | menu | Axis Titles |
| ChartAxes | menu | Axes |
| ChartGridlines | menu | Gridlines |
| ChartFormatSelection | button | Format Selection |
| ChartElementSelector | comboBox | Chart Elements |
| SlideShowFromCurrent | button | From Current Slide |
| BulletsGallery | gallery | Bullets |
| NumberingGallery | gallery | Numbering |
| LineSpacingGalleryPowerPoint | gallery | Line Spacing |
| SlideNewGallery | gallery | New Slide |
| SlideLayoutGallery | gallery | Layout |
| TabPictureToolsFormat | tab | Format |
| TabDrawingToolsFormat | tab | Format |
| ShapesInsertGallery | gallery | Shapes |
| ShapeChangeShapeGallery | gallery | Change Shape |
| ShapeFillTextureGallery | gallery | Texture |
| ShapeStylesGallery | gallery | Quick Styles |
| PageOrientationGallery | menu | Orientation |
| FileServerTasksMenu | menu | Server |
| FileSendMenu | menu | Send |
| TabInsert | tab | Insert |
| TabView | tab | View |
| GroupFont | group | Font |
| GroupParagraph | group | Paragraph |
| GroupProofing | group | Proofing |
| GroupInsertIllustrations | group | Illustrations |
| GroupShapes | group | Insert Shapes |
| GroupPageSetup | group | Page Setup |
| GroupComments | group | Comments |
| GroupPictureSize | group | Size |
| GroupDrawBorders | group | Draw Borders |
| TabDesign | tab | Design |
| TabAnimations | tab | Animations |
| TabSlideShow | tab | Slide Show |
| GroupSlides | group | Slides |
| GroupArrange | group | Arrange |
| GroupInsertMediaClips | group | Media Clips |
| GroupSlideThemes | group | Themes |
| GroupBackground | group | Background |
| GroupPreview | group | Preview |
| GroupAnimations | group | Animations |
| GroupTransitionToThisSlide | group | Transition to This Slide |
| GroupSlideShowStart | group | Start Slide Show |
| GroupSlideShowSetup | group | Set Up |
| GroupShapeStyles | group | Shape Styles |
| TabReview | tab | Review |
| GroupChartLayouts | group | Chart Layouts |
| GroupChartStyles | group | Chart Styles |
| GroupChartAxes | group | Axes |
| GroupChartShapes | group | Insert |
| ObjectEffectPresetGallery | gallery | Preset |
| PictureEffectsPresetGallery | gallery | Preset |
| \_3DRotationGallery | gallery | 3-D Rotation |
| TabSmartArtToolsDesign | tab | Design |
| TabSmartArtToolsFormat | tab | Format |
| TabChartToolsDesign | tab | Design |
| TabChartToolsLayout | tab | Layout |
| TabChartToolsFormat | tab | Format |
| ShapeFillColorPicker | gallery | Shape Fill |
| OutlineColorPicker | gallery | Picture Border |
| FileDocumentInspect | button | Inspect Document |
| ClearFormatting | button | Clear Formatting |
| GroupControls | group | Controls |
| GroupZoom | group | Zoom |
| FilePublishSlides | button | Publish Slides |
| SlideShowUsePresenterView | checkBox | Use Presenter View |
| ArrowStyleGallery | gallery | Arrows |
| OutlineDashesGallery | gallery | Dashes |
| OutlineWeightGallery | gallery | Weight |
| TabTableToolsLayout | tab | Layout |
| GroupAlignment | group | Alignment |
| GroupPictureTools | group | Adjust |
| GroupSize | group | Size |
| TabSlideMaster | tab | Slide Master |
| GroupTableStylesPowerPoint | group | Table Styles |
| GroupTableRowsAndColumns | group | Rows & Columns |
| ObjectAlignMenu | menu | Align |
| MovieInsert | splitButton | Movie |
| SoundInsertMenu | splitButton | Sound |
| ObjectRotateGallery | gallery | Rotate |
| SelectMenu | menu | Select |
| AnimationGallery | dropDown | Animate: |
| AnimationTransitionGallery | gallery | Transition Scheme |
| AnimationTransitionSpeedGallery | dropDown | Transition Speed: |
| FontColorPicker | gallery | Font Color |
| TableColumnsGallery | gallery | Columns |
| TabHome | tab | Home |
| ChartTitle | gallery | Chart Title |
| ChartPrimaryHorizontalAxisTitle | gallery | Primary Horizontal Axis Title |
| ChartPrimaryVerticalAxisTitle | gallery | Primary Vertical Axis Title |
| ChartDepthAxisTitle | gallery | Depth Axis Title |
| ChartLegend | gallery | Legend |
| ChartDataLabel | gallery | Data Labels |
| ChartPrimaryHorizontalGridlines | gallery | Primary Horizontal Gridlines |
| ChartPrimaryVerticalGridlines | gallery | Primary Vertical Gridlines |
| ChartDepthGridlines | gallery | Depth Gridlines |
| ChartPrimaryHorizontalAxis | gallery | Primary Horizontal Axis |
| ChartPrimaryVerticalAxis | gallery | Primary Vertical Axis |
| ChartDepthAxis | gallery | Depth Axis |
| ChartDataTable | gallery | Data Table |
| ChartTrendline | gallery | Trendline |
| ChartErrorBars | gallery | Error Bars |
| ChartLines | gallery | Lines |
| ChartUpDownBars | gallery | Up/Down Bars |
| ChartPlotArea | gallery | Plot Area |
| ChartWall | gallery | Chart Wall |
| ChartFloor | gallery | Chart Floor |
| SmartArtAddShape | button | Add Shape |
| SmartArtLargerShape | button | Larger |
| SmartArtSmallerShape | button | Smaller |
| SmartArtResetGraphic | button | Reset Graphic |
| SmartArtTextPane | toggleButton | Text Pane |
| SmartArtEditIn2D | toggleButton | Edit in 2-D |
| SmartArtLayoutGallery | gallery | Change Layout |
| SmartArtMoreLayoutsDialog | button | More Layouts... |
| SmartArtStylesGallery | gallery | Quick Styles |
| SmartArtChangeColorsGallery | gallery | Change Colors |
| ObjectEffectSoftEdgesGallery | gallery | Soft Edges |
| ObjectEffectGlowGallery | gallery | Glow |
| GradientGallery | gallery | Gradient |
| ObjectEffectShadowGallery | gallery | Shadow |
| WordArtInsertGallery | gallery | WordArt |
| TextEffectTransformGallery | gallery | Transform |
| ThemeColorsGallery | gallery | Colors |
| GroupSmartArtLayouts | group | Layouts |
| GroupSmartArtQuickStyles | group | SmartArt Styles |
| GroupSmartArtCreateGraphic | group | Create Graphic |
| GroupSmartArtReset | group | Reset |
| GroupSmartArtSize | group | Size |
| ThemeSearchOfficeOnlinePowerPoint | button | More Themes on Microsoft Office Online... |
| TabAddIns | tab | Add-Ins |
| ObjectEditShapeMenu | menu | Edit Shape |
| SlideMasterContentPlaceholderInsert | button | Content |
| SlideMasterTextPlaceholderInsert | button | Text |
| SlideMasterChartPlaceholderInsert | button | Chart |
| SlideMasterTablePlaceholderInsert | button | Table |
| SlideMasterDiagramPlaceholderInsert | button | SmartArt |
| SlideMasterMediaPlaceholderInsert | button | Media |
| SlideMasterClipArtPlaceholderInsert | button | Clip Art |
| SlideMasterVerticalTextPlaceholderInsert | button | Text |
| SlideMasterShowTitle | checkBox | Title |
| SlideMasterShowFooters | checkBox | Footers |
| TableStylesGallery | gallery | Table Styles |
| FileSaveAsPdfOrXps | button | Publish as PDF or XPS |
| FileSaveAsPowerPointOpenDocumentPresentation | button | OpenDocument Presentation |
| MoreControlsDialog | button | More Controls... |
| GroupCode | group | Code |
| TabDeveloper | tab | Developer |
| SlideTransitionApplyToAll | button | Apply To All |
| SlideTransitionOnMouseClick | checkBox | On Mouse Click |
| AnimationPreview | button | Preview |
| GroupEditing | group | Editing |
| GroupClipboard | group | Clipboard |
| GroupInsertTables | group | Tables |
| GroupInsertLinks | group | Links |
| GroupInsertBarcode | group | Barcode |
| AnimationTransitionSoundGallery | dropDown | Transition Sound: |
| ObjectPictureFill | button | Picture... |
| SlideShowFromBeginning | button | From Beginning |
| WindowSwitchWindowsMenuPowerPoint | menu | Switch Windows |
| ThemeColorsCreateNew | button | Create New Theme Colors... |
| ThemeFontsCreateNew | button | Create New Theme Fonts... |
| ShapeFillMoreGradientsDialog | button | More Gradients... |
| ShadowOptionsDialog | button | Shadow Options... |
| ObjectsAlignSelectedSmart | toggleButton | Align Selected Objects |
| ObjectsAlignRelativeToContainerSmart | toggleButton | Align to Slide |
| ObjectsAlignLeftSmart | button | Align Left |
| ObjectsAlignRightSmart | button | Align Right |
| ObjectsAlignTopSmart | button | Align Top |
| ObjectsAlignBottomSmart | button | Align Bottom |
| ObjectsAlignCenterHorizontalSmart | button | Align Center |
| ObjectsAlignMiddleVerticalSmart | button | Align Middle |
| AlignDistributeHorizontally | button | Distribute Horizontally |
| AlignDistributeVertically | button | Distribute Vertically |
| SlideShowCustomMenu | menu | Custom Slide Show |
| SlideShowResolutionGallery | dropDown | Resolution: |
| SlideShowUseRehearsedTimings | checkBox | Use Rehearsed Timings |
| SlideShowShowPresentationOnGallery | dropDown | Show Presentation On: |
| GroupMonitors | group | Monitors |
| ThemeFontsGallery | gallery | Fonts |
| ThemeEffectsGallery | gallery | Effects |
| FileProperties | toggleButton | Properties |
| TabPrintPreview | tab | Print Preview |
| GroupPrintPreviewPrint | group | Print |
| GroupPrintPreviewPreview | group | Preview |
| TabSoundToolsOptions | tab | Options |
| TabMovieToolsOptions | tab | Options |
| GroupPlay | group | Play |
| GroupMovieOptions | group | Movie Options |
| GroupSoundOptions | group | Sound Options |
| MediaClipToolsHideDuringShow | checkBox | Hide During Show |
| MediaClipLoopUntilStopped | checkBox | Loop Until Stopped |
| SoundPlaySoundGallery | dropDown | Play Sound: |
| SlideShowVolume | gallery | Slide Show Volume |
| MoviePlayFullScreen | checkBox | Play Full Screen |
| MovieRewindAfterPlaying | checkBox | Rewind Movie After Playing |
| TableBordersMenu | splitButton | Borders |
| FileCreateDocumentWorkspace | toggleButton | Create Document Workspace |
| FileSaveToDocumentManagementServer | button | Document Management Server |
| FileDocumentManagementInformation | toggleButton | Document Management Information |
| QuickAccessToolbarCustomization | button | Customize Quick Access Toolbar... |
| FilePrepareMenu | menu | Prepare |
| FileMarkAsFinal | toggleButton | Mark as Final |
| FileAddDigitalSignature | button | Add a Digital Signature |
| SlideBackgroundHideGraphics | checkBox | Hide Background Graphics |
| ChangeCaseGallery | gallery | Change Case |
| ReviewDeleteCommentsMenuPowerPoint | splitButton | Delete |
| GroupMerge | group | Merge |
| AlignJustifyMenu | menu | Justify |
| TextBoxInsertMenu | splitButton | Text Box |
| TextBoxInsertHorizontal | toggleButton | Horizontal Text Box |
| ControlProperties | button | Properties |
| HandoutOrientation | gallery | Handout Orientation |
| SlidesPerPageGallery | menu | Slides Per Page |
| MasterShowSlideImage | checkBox | Slide Image |
| MasterShowBody | checkBox | Body |
| MasterShowDate | checkBox | Date |
| MasterShowPageNumber | checkBox | Page Number |
| MasterShowHeader | checkBox | Header |
| MasterShowFooter | checkBox | Footer |
| TabHandoutMaster | tab | Handout Master |
| TabNotesMaster | tab | Notes Master |
| GroupPlaceholdersHandoutMaster | group | Placeholders |
| GroupMasterEditTheme | group | Edit Theme |
| GroupMasterClose | group | Close |
| GroupMasterEdit | group | Edit Master |
| GroupMasterLayout | group | Master Layout |
| TabGrayscale | tab | Grayscale |
| TabBlackAndWhite | tab | Black And White |
| GroupColorModeSetting | group | Change Selected Object |
| GroupInkSelect | group | Select |
| PictureBrightnessGallery | gallery | Brightness |
| PictureContrastGallery | gallery | Contrast |
| ShadingColorPicker | gallery | Shading |
| SmartArtAddShapeAfter | button | Add Shape After |
| SmartArtAddShapeBefore | button | Add Shape Before |
| SmartArtAddShapeAbove | button | Add Shape Above |
| SmartArtAddShapeBelow | button | Add Shape Below |
| SmartArtAddAssistant | button | Add Assistant |
| ThemeColorsReset | button | Reset Slide Theme Colors |
| ChartSwitchRowColumn | button | Switch Row/Column |
| ChartShowData | button | Edit Data... |
| ChartRefresh | button | Refresh Data |
| ChartChangeType | button | Change Chart Type... |
| GroupChartData | group | Data |
| GroupChartType | group | Type |
| GroupTableSize | group | Table Size |
| TableCellMarginsGallery | gallery | Cell Margins |
| SlideMasterVerticalContentPlaceholderInsert | button | Content (Vertical) |
| \_3DRotationOptionsDialog | button | 3-D Rotation Options... |
| \_3DBevelOptionsDialog | button | 3-D Options... |
| SlideBackgroundStylesGallery | gallery | Background Styles |
| TextDirectionGallery | gallery | Text Direction |
| GroupTableStyleOptionsPowerPoint | group | Table Style Options |
| TableStyleFirstRowPowerPoint | checkBox | Header Row |
| TableStyleFirstColumnPowerPoint | checkBox | First Column |
| TableStyleTotalRowPowerPoint | checkBox | Total Row |
| TableStyleLastColumnPowerPoint | checkBox | Last Column |
| TableStyleBandedRowsPowerPoint | checkBox | Banded Rows |
| TableStyleBandedColumnsPowerPoint | checkBox | Banded Columns |
| SelectionPane | toggleButton | Selection Pane... |
| SmartArtOrganizationChartLeftHanging | button | Left Hanging |
| SmartArtOrganizationChartRightHanging | button | Right Hanging |
| SmartArtOrganizationChartBoth | button | Both |
| SmartArtOrganizationChartStandard | button | Standard |
| SmartArtRightToLeft | toggleButton | Right to Left |
| TableCellCustomMarginsDialog | button | Custom Margins... |
| ViewMessageBar | checkBox | Message Bar |
| ShapeStylesOtherThemeFillsGallery | gallery | Other Theme Fills |
| SmartArtOrganizationChartMenu | menu | Layout |
| TabSlideMasterHome | tab | Home |
| TableInsertGallery | gallery | Table |
| AlignJustifyWithMixedLanguages | toggleButton | Justify |
| AlignJustifyLow | toggleButton | Justify Low |
| AlignJustifyThai | toggleButton | Distribute |
| ExcelSpreadsheetInsert | button | Excel Spreadsheet |
| TabCDAudioToolsOptions | tab | Options |
| SlideTransitionAutomaticallyAfter | checkBox | Automatically After: |
| SlideOrientationGallery | gallery | Slide Orientation |
| GroupAddInsMenuCommands | group | Menu Commands |
| GroupAddInsToolbarCommands | group | Toolbar Commands |
| GroupInk | group | Ink |
| TabInkToolsPens | tab | Pens |
| GroupInkPens | group | Pens |
| GroupInkClose | group | Close |
| InkBallpointPen | toggleButton | Ballpoint Pen |
| InkFeltTipPen | toggleButton | Felt Tip Pen |
| InkHighlighter | toggleButton | Highlighter |
| ThemeSaveCurrentPowerPoint | button | Save Current Theme... |
| TextAlignGallery | gallery | Align Text |
| CharacterSpacingGallery | gallery | Character Spacing |
| ChartResetToMatchStyle | button | Reset to Match Style |
| Chart3DView | button | 3-D Rotation... |
| ObjectSizeAndPositionDialog | button | Size and Position... |
| ShapeConvertToFreeform | button | Convert to Freeform |
| TabTableToolsDesign | tab | Design |
| ConvertToSmartArt | gallery | Convert to SmartArt |
| ConvertToSmartArtMoreSmartArtGraphicsDialog | button | More SmartArt Graphics... |
| ReflectionGallery | gallery | Reflection |
| PictureRecolorGallery | gallery | Recolor |
| SmartArtPromote | button | Promote |
| SmartArtDemote | button | Demote |
| TableEffectsCellBevelGallery | gallery | Cell Bevel |
| TableBackgroundGallery | gallery | Table Background |
| ChartTitleOptionsDialog | button | More Title Options... |
| ChartLegendOptionsDialogDialog | button | More Legend Options... |
| ChartDataLabelDialog | button | More Data Label Options... |
| ChartPrimaryHorizontalAxisTitleOptionsDialog | button | More Primary Horizontal Axis Title Options... |
| ChartPrimaryVerticalAxisTitleOptionsDialog | button | More Primary Vertical Axis Title Options... |
| ChartSecondaryHorizontalAxisTitleOptionsDialog | button | More Secondary Horizontal Axis Title Options... |
| ChartSecondaryVerticalAxisTitleOptionsDialog | button | More Secondary Vertical Axis Title Options... |
| ChartDepthAxisTitleOptionsDialog | button | More Depth Axis Title Options... |
| ChartPrimaryHorizontalGridlinesOptionsDialog | button | More Primary Horizontal Gridlines Options... |
| ChartPrimaryVerticalGridlinesOptionsDialog | button | More Primary Vertical Gridlines Options... |
| ChartSecondaryHorizontalGridlinesOptionsDialog | button | More Secondary Horizontal Gridlines Options... |
| ChartSecondaryVerticalGridlinesOptionsDialog | button | More Secondary Vertical Gridlines Options... |
| ChartDepthGridlinesOptionsDialog | button | More Depth Gridlines Options... |
| ChartPrimaryHorizontalAxisOptionsDialog | button | More Primary Horizontal Axis Options... |
| ChartPrimaryVerticalAxisOptionsDialog | button | More Primary Vertical Axis Options... |
| ChartSecondaryHorizontalAxisOption | button | More Secondary Horizontal Axis Options... |
| ChartSecondaryVerticalAxisOptionsDialog | button | More Secondary Vertical Axis Options... |
| ChartDepthAxisOptionsDialog | button | More Depth Axis Options... |
| ChartDataTableOptionsDialog | button | More Data Table Options... |
| ChartTrendlineOptionsDialog | button | More Trendline Options... |
| ChartErrorBarsOptionsDialog | button | More Error Bars Options... |
| ChartUpDownBarsOptionsDialog | button | More Up/Down Bars Options... |
| ChartPlotAreaOptionsDialog | button | More Plot Area Options... |
| ChartWallOptionsDialog | button | More Walls Options... |
| ChartFloorOptionsDialog | button | More Floor Options... |
| ChartSecondaryHorizontalAxisTitle | gallery | Secondary Horizontal Axis Title |
| ChartSecondaryVerticalAxisTitle | gallery | Secondary Vertical Axis Title |
| ChartSecondaryHorizontalGridlines | gallery | Secondary Horizontal Gridlines |
| ChartSecondaryVerticalGridlines | gallery | Secondary Vertical Gridlines |
| ChartSecondaryHorizontalAxis | gallery | Secondary Horizontal Axis |
| ChartSecondaryVerticalAxis | gallery | Secondary Vertical Axis |
| GroupAddInsCustomToolbars | group | Custom Toolbars |
| MoviePlayAutomatically | dropDown | Play Movie: |
| ObjectBringToFrontMenu | splitButton | Bring to Front |
| ObjectSendToBackMenu | splitButton | Send to Back |
| ObjectsGroupMenu | menu | Group |
| ViewDisplayInColor | toggleButton | Color |
| FileSaveAsPowerPoint97\_2003 | button | PowerPoint 97-2003 Presentation |
| FileSaveAsMenu | splitButton | Save As Other Format |
| FilePrintMenu | splitButton | Preview and Print |
| FilePermissionRestrictMenu | menu | Restrict Permission |
| DocumentPanelTemplate | button | Document Panel |
| GroupModify | group | Modify |
| BevelShapeGallery | gallery | Bevel |
| \_3DBevelPictureTopGallery | gallery | Bevel |
| TableStyleClearTable | button | Clear Table |
| GroupTable | group | Table |
| GroupTableCellSize | group | Cell Size |
| TableEffectsMenu | menu | Effects |
| GlowColorPicker | gallery | More Glow Colors |
| RecolorColorPicker | gallery | More Variations |
| GlowColorMoreColorsDialog | button | More Colors... |
| PictureRecolorMoreColorsDialog | button | More Colors... |
| SmartArtAddBullet | button | Add Bullet |
| TableLockAspectRatio | checkBox | Lock Aspect Ratio |
| SlideMasterPicturePlaceholderInsert | button | Picture |
| SlideMasterInsertPlaceholderMenu | splitButton | Insert Placeholder |
| TableBorderColorMoreColorsDialog | button | More Border Colors... |
| TableFillColorMoreColorsDialog | button | More Fill Colors... |
| PictureChange | button | Change Picture... |
| GroupWordArtStyles | group | WordArt Styles |
| TextFillColorPicker | gallery | Text Fill |
| TextOutlineColorPicker | gallery | Text Outline |
| TextOutlineColorMoreColorsDialog | button | More Outline Colors... |
| TextEffectsMenu | menu | Text Effects |
| TextStylesGallery | gallery | Quick Styles |
| WordArtClear | button | Clear WordArt |
| TextPictureFill | button | Picture... |
| TextFillGradientGallery | gallery | Gradient |
| TextFillMoreGradientsDialog | button | More Gradients... |
| TextFillTextureGallery | gallery | Texture |
| TextOutlineDashesGallery | gallery | Dashes |
| TextOutlineMoreLinesDialog | button | More Lines... |
| TextOutlineWeightGallery | gallery | Weight |
| TextEffectShadowGallery | gallery | Shadow |
| TextEffectsMoreShadowsDialog | button | Shadow Options... |
| TextEffectsBevelMore3DOptionsDialog | button | 3-D Options... |
| TextEffects3DRotationGallery | gallery | 3-D Rotation |
| TextEffects3DRotationOptionsDialog | button | 3-D Rotation Options... |
| TextEffectGlowGallery | gallery | Glow |
| TextGlowColorPicker | gallery | More Glow Colors |
| TextGlowColorMoreColorsDialog | button | More Colors... |
| TextReflectionGallery | gallery | Reflection |
| ShapeEffectsMenu | menu | Shape Effects |
| TransitionSoundLoopUntilNextSound | toggleButton | Loop Until Next Sound |
| GroupChartCurrentSelection | group | Current Selection |
| GroupChartLabels | group | Labels |
| BevelTextGallery | gallery | Bevel |
| PictureCorrectionsDialog | button | Picture Corrections Options... |
| SmartArtAddShapeSplitMenu | splitButton | Add Shape Options |
| MasterNotesPageOrientation | gallery | Notes Page Orientation |
| GroupInkFormat | group | Format |
| InkColorPicker | gallery | Color |
| TextDirectionMoreOptionsDialog | button | More Options... |
| TextAlignMoreOptionsDialog | button | More Options... |
| ParagraphMoreColumnsDialog | button | More Columns... |
| SlideBackgroundReset | button | Reset Slide Background |
| CDAudioPlayTrackAutomatically | dropDown | Play Track: |
| GroupSmartArtShapes | group | Shapes |
| GroupInsertText | group | Text |
| ShapeOutlineColorPicker | gallery | Picture Border |
| GroupColorModeClose | group | Close |
| GroupPageSetupNotesMaster | group | Page Setup |
| GroupPageSetupHandoutMaster | group | Page Setup |
| GroupPlaceholdersNotesMaster | group | Placeholders |
| ReplaceMenu | splitButton | Replace... |
| TableSelectMenuPowerPoint | menu | Select |
| TableDeleteRowsAndColumnsMenu | menu | Delete |
| GroupCDAudioSetup | group | Set Up |
| PrintOptionsMenu | menu | Options |
| ReviewDeleteAllMarkupInPresentation | button | Delete All Markup in the Presentation |
| PasteMenu | splitButton | Paste |
| GroupPictureStyles | group | Picture Styles |
| PictureStylesGallery | gallery | Quick Styles |
| PictureEffectsShadowGallery | gallery | Shadow |
| PictureEffectsGlowGallery | gallery | Glow |
| PictureEffectsSoftEdgesGallery | gallery | Soft Edges |
| PictureReflectionGallery | gallery | Reflection |
| PictureRotationGallery | gallery | 3-D Rotation |
| InkToolsClose | button | Close Ink Tools |
| GroupViewShowHide | group | Show/Hide |
| GroupWindow | group | Window |
| ViewGridlines | checkBox | View Gridlines |
| GroupPresentationViews | group | Presentation Views |
| GroupColorGrayscale | group | Color/Grayscale |
| FileDocumentEncrypt | toggleButton | Encrypt Document |
| WordArtFormatDialog | button | Format Text Effects... |
| ObjectRotationOptionsDialog | button | More Rotation Options... |
| MoreTextureOptions | button | More Textures... |
| TextFillColorMoreColorsDialog | button | More Fill Colors... |
| ZoomFitToWindow | button | Fit to Window |
| GroupPrintPreviewPageSetup | group | Page Setup |
| ViewDirectionMenu | menu | View Direction |
| FileEmailAsPdfEmailAttachment | button | E-mail as PDF Attachment |
| FileEmailAsXpsEmailAttachment | button | E-mail as XPS Attachment |
| PictureEffectsMenu | menu | Picture Effects |
| PictureShapeGallery | gallery | Change Shape |
| PhotoAlbumInsertMenu | splitButton | Photo Album... |
| GroupChartBackground | group | Background |
| GroupChartAnalysis | group | Analysis |
| MenuPublish | menu | Publish |
| FileCompatibilityCheckerPowerPoint | button | Run Compatibility Checker |
| TableBackgroundPictureFill | button | Picture... |
| UpgradePresentation | button | Convert |
| FileSaveAsOtherFormats | button | Save As |
| FileSaveAsPowerPointPptx | button | PowerPoint Presentation |
| FileSaveAsPowerPointPpsx | button | PowerPoint Show |
| FontDialogPowerPoint | button | Font... |
| PowerPointParagraphDialog | button | Paragraph... |
| PowerPointPageSetup | button | Page Setup... |
| ShapeQuickStylesHome | gallery | Quick Styles |
| GalleryAllShapesAndTextboxes | gallery | Shapes |
| TableTextStylesGallery | gallery | Quick Styles |
| GroupTextStylesTable | group | WordArt Styles |
| ObjectsArrangeMenu | menu | Arrange |
| TextFillMoreTextures | button | More Textures... |
| GroupMacros | group | Macros |
| AdvertisePublishAs | button | Find add-ins for other file formats |
| GroupPermission | group | Protect |
| ReviewProtectPresentationMenu | menu | Protect Presentation |
| AlternativeText | button | Size and Position... |
| FileCheckOutDiscard | button | Discard Check Out |
| MdiChildSystemMenu | menu | Document |

## imageMso Table

| imageMso |
| --- |
| Spelling |
| FileSave |
| FilePrint |
| ZoomOnePage |
| ZoomPageWidth |
| Zoom100 |
| TableInsert |
| ColumnsDialog |
| Numbering |
| Bullets |
| PageOrientationPortraitLandscape |
| OutdentClassic |
| IndentClassic |
| DrawingInsert |
| ChartInsert |
| FileNew |
| Copy |
| Cut |
| Paste |
| FileOpen |
| EnvelopesAndLabelsDialog |
| ZoomPrintPreviewExcel |
| PenComment |
| Folder |
| Repeat |
| UpArrow2 |
| RightArrow2 |
| DownArrow2 |
| LeftArrow2 |
| Clear |
| Breakpoint |
| Piggy |
| Superscript |
| Subscript |
| HappyFace |
| UnderlineDouble |
| UnderlineWords |
| FontSizeIncreaseWord |
| FontSizeDecreaseWord |
| \_0 |
| \_1 |
| \_2 |
| \_3 |
| \_4 |
| \_5 |
| \_6 |
| \_7 |
| \_8 |
| \_9 |
| A |
| B |
| C |
| D |
| E |
| F |
| G |
| H |
| I |
| J |
| K |
| L |
| M |
| N |
| O |
| P |
| Q |
| R |
| S |
| T |
| U |
| V |
| W |
| X |
| Y |
| Z |
| FileClose |
| TableAutoFormat |
| FormatPainter |
| FilePrintPreview |
| PickUpStyle |
| PasteApplyStyle |
| Bold |
| Italic |
| Underline |
| DarkShading |
| ParagraphMarks |
| AlignLeft |
| AlignRight |
| AlignCenter |
| AlignJustify |
| ContextHelp |
| HeaderFooterPageNumberInsert |
| Undo |
| Redo |
| ControlLine |
| ControlRectangle |
| OutlinePromote |
| OutlineDemote |
| OutlineMoveUp |
| OutlineMoveDown |
| OutlineDemoteToBodyText |
| OutlineExpand |
| OutlineCollapse |
| TextBoxInsert |
| FileFind |
| FindDialog |
| TableExcelSpreadsheetInsert |
| AutoFormat |
| BorderTop |
| BorderBottom |
| BorderLeft |
| BorderRight |
| BorderInside |
| BorderOutside |
| BorderNone |
| MailMergeGoToFirstRecord |
| MailMergeGoToPreviousRecord |
| MailMergeGoToNextRecord |
| MailMergeGotToLastRecord |
| MailMergeMergeToDocument |
| MailMergeMergeToPrinter |
| MailMergeAutoCheckForErrors |
| DataFormSource |
| MailMergeResultsPreview |
| ObjectsGroup |
| ObjectsUngroup |
| ObjectBringToFront |
| ObjectSendToBack |
| AboveText |
| BehindText |
| ObjectBringForward |
| ObjectSendBackward |
| Magnifier |
| PrintPreviewShrinkOnePage |
| MultiplePages |
| ViewFullScreenView |
| ViewRulerPowerPoint |
| VoiceInsert |
| ObjectsSelect |
| TableFind |
| MacroRecord |
| MacroRecorderPause |
| MacroPlay |
| ObjectFlipHorizontal |
| ObjectFlipVertical |
| ObjectRotateRight90 |
| ObjectRotateLeft90 |
| ShapeFreeform |
| GroupDrawing |
| ObjectEditPoints |
| CalloutOptions |
| SortUp |
| SortDown |
| TableDesign |
| DataFormAddRecord |
| DataFormDeleteRecord |
| FieldsUpdate |
| DatabaseInsert |
| GridSettings |
| WordPicture |
| FormControlEditBox |
| FormControlCheckBox |
| FormControlComboBox |
| PropertySheet |
| FieldShading |
| ViewDraftView |
| Lock |
| AutoSum |
| MasterDocumentShow |
| MasterDocumentCreateSubdocument |
| MasterDocumentUnlinkSubdocument |
| MasterDocumentInsertSubdocument |
| MasterDocumentSplitSubdocuments |
| MasterDocumentMergeSubdocuments |
| MasterDocumentLockSubdocument |
| HeaderOrFooterShow |
| HeaderFooterPreviousSection |
| HeaderFooterNextSection |
| AlignDialog |
| MailMergeDocument |
| MergeOptions |
| MailMergeHelper |
| PageSetupDialog |
| BodyTextHide |
| HeaderFooterLinkToPrevious |
| OutlineShowFirstLineOnly |
| OutlineShowTextFormatting |
| FontDialog |
| StylesDialogClassic |
| RoutingSlip |
| FootnoteInsert |
| MicrosoftExcel |
| MicrosoftAccess |
| Schedule |
| MicrosoftVisualFoxPro |
| MicrosoftPowerPoint |
| MicrosoftPublisher |
| MicrosoftProject |
| SadFace |
| Pushpin |
| Camera |
| FormControlButton |
| Calculator |
| ViewPrintLayoutView |
| FieldCodes |
| DropCapOptionsDialog |
| Strikethrough |
| TextSmallCaps |
| CellsDelete |
| TableRowsDelete |
| TableColumnsDelete |
| CellsInsertDialog |
| TableRowsInsertWord |
| QueryInsertColumns |
| WindowsArrangeAll |
| MarginsAdjust |
| ViewGridlinesWord |
| SubdocumentOpen |
| WindowSplit |
| WindowNew |
| LegalBlackline |
| ReviewAcceptOrRejectChangeDialog |
| TextAllCaps |
| PictureDisassemble |
| SymbolInsert |
| ChangeCaseDialogClassic |
| FontSizeDecrease1Point |
| FontSizeIncrease1Point |
| Repaginate |
| ReplaceDialog |
| StartOfLine |
| EndOfLine |
| PagePrevious |
| PageNext |
| TextBoxInsertVertical |
| StartOfDocument |
| EndOfDocument |
| Grammar |
| FileCloseOrCloseAll |
| TextToOrFromTable |
| TableRowsOrColumnsOrCellsInsert |
| TableRowsOrColumnsOrCellsDelete |
| RedoOrRepeat |
| ObjectsRegroup |
| \_3DStyle |
| TipWizardHelp |
| AutoFormatChange |
| AddressBook |
| Reply |
| ReplyAll |
| Forward |
| MailMove |
| MailDelete |
| MessagePrevious |
| MessageNext |
| CheckNames |
| MailSelectNames |
| PrintAreaSetPrintArea |
| PasteFormatting |
| FillRight |
| FillDown |
| EqualSign |
| PlusSign |
| MinusSign |
| MultiplicationSign |
| DivisionSign |
| ExponentiationSign |
| ParenthesisLeft |
| ParenthesisRight |
| ColonSign |
| CommaSign |
| PercentSign |
| DollarSign |
| FunctionWizard |
| AsianLayoutCharacterScaling |
| ConstrainNumeric |
| LightShading |
| AccountingFormat |
| PercentStyle |
| CommaStyle |
| DecimalsIncrease |
| DecimalsDecrease |
| MergeCenter |
| FontSizeIncrease |
| FontSizeDecrease |
| TextOrientationVertical |
| TextOrientationRotateUp |
| TextOrientationRotateDown |
| AlignDistributeHorizontallyClassic |
| ShapeScribble |
| ChartAreaChart |
| Chart3DBarChart |
| Chart3DColumnChart |
| Chart3DPieChart |
| ChartRadarChart |
| OutlineSymbolsShowHide |
| TableSelectVisibleCells |
| SelectCurrentRegion |
| FreezePanes |
| ZoomIn |
| ZoomOut |
| FormControlRadioButton |
| FormControlScrollBar |
| FormControlListBox |
| TraceDependentRemoveArrows |
| TraceDependents |
| TracePrecedentsRemoveArrows |
| TraceRemoveAllArrows |
| FileUpdate |
| ReadOnly |
| AutoFilterClassic |
| Refresh |
| PivotTableFieldSettings |
| PivotTableShowPages |
| OutlineShowDetail |
| TraceError |
| OutlineHideDetail |
| AlignDistributeVerticallyClassic |
| FormControlGroupBox |
| FormControlSpinner |
| TabOrder |
| RunDialog |
| FormControlCombinationListEdit |
| FormControlCombinationDropDownEdit |
| FormControlLabel |
| Delete |
| Fish |
| Coffee |
| Heart |
| Diamond |
| Spade |
| Club |
| ViewSheetGridlines |
| TracePrecedents |
| Info |
| CodeEdit |
| InsertDialog |
| ApplyFilter |
| DatasheetView |
| SortAndFilterAdvanced |
| ControlSubFormReport |
| FieldList |
| ViewsFormView |
| Grouping |
| AdpPrimaryKey |
| ControlBoundObjectFrame |
| ControlUnboundObjectFrame |
| PageBreakInsertOrRemove |
| PrintSetupDialog |
| CreateFormInDesignView |
| CreateQueryFromWizard |
| CreateReportInDesignView |
| MacroConditions |
| MacroNames |
| ControlToggleButton |
| DatabaseRelationships |
| TableIndexes |
| ViewsAdpDiagramSqlView |
| QueryTableNamesShowHide |
| QueryShowTable |
| QuerySelectQueryType |
| QueryCrosstab |
| QueryMakeTable |
| QueryUpdate |
| QueryAppend |
| QueryDelete |
| QueryParameters |
| RecordsSaveRecord |
| GoToNewRecord |
| RowHeight |
| ColumnWidth |
| RecordsFreezeColumns |
| GridlinesGallery |
| OleObjectctInsert |
| ControlToolboxOutlook |
| SnapToGrid |
| SizeToFit |
| PageHeaderOrFooterShowHide |
| FormHeaderOrFooterShowHide |
| First10RecordsPreview |
| ControlSpecialEffectRaised |
| ControlSpecialEffectSunken |
| AutoDial |
| FindNext |
| PasteDuplicate |
| DatabasePermissions |
| ControlAlignToGrid |
| ControlSpecialEffectFlat |
| CreateTableInDesignView |
| MacroDefault |
| ModuleInsert |
| FilterToggleFilter |
| FilterClearAllFilters |
| Head |
| ReminderSound |
| CreateModule |
| RelationshipsDirectRelationships |
| RelationshipDesignAllRelationships |
| ControlWizards |
| MergeToWord |
| FilterAdvancedByForm |
| CreateMacro |
| AutoFormatWizard |
| PrintPreviewZoomTwoPages |
| FilterBySelection |
| RecordsDeleteRecord |
| QueryBuilder |
| DatabaseDocumenter |
| DatabaseAnalyzePerformance |
| DatabaseAnalyzeTable |
| ObjectsAlignLeft |
| ObjectsAlignRight |
| ObjectsAlignTop |
| ObjectsAlignBottom |
| ObjectsAlignCenterHorizontal |
| ObjectsAlignMiddleVertical |
| SlideNew |
| ClipArtInsert |
| CreateHandoutsInWord |
| Shadow |
| ObjectRotateFree |
| ShapesMoreShapes |
| CopyToPersonalContacts |
| ParagraphSpacingIncrease |
| ParagraphSpacingDecrease |
| SlideMasterMasterLayout |
| OrganizationChartInsert |
| CollapseAll |
| OutlineExpandAll |
| CombineCharacters |
| DoubleStrikethrough |
| QueryInsertColumn |
| EncryptMessage |
| DigitallySignMessage |
| CreateMailRule |
| ViewNormalViewExcel |
| ViewPageBreakPreviewView |
| SlideHide |
| AnimationCustom |
| PictureCrop |
| SlideShowRehearseTimings |
| ViewSlideView |
| ViewOutlineView |
| ViewSlideSorterView |
| ViewNotesPageView |
| ViewSlideShowView |
| ViewSlideMasterView |
| FileCloseAll |
| FileSaveAs |
| SaveAll |
| AdvancedFileProperties |
| DocumentTemplate |
| FileExit |
| PasteSpecialDialog |
| SelectAll |
| GoTo |
| BookmarkInsert |
| FileLinksToFiles |
| ViewOnlineLayoutViewClassic |
| HeaderFooterInsert |
| FootnotesEndnotesShow |
| BreakInsertDialog |
| DateAndTimeInsert |
| NumberInsert |
| FieldInsert |
| CaptionInsert |
| CrossReferenceInsert |
| TextFromFileInsert |
| ParagraphDialog |
| BordersShadingDialog |
| TextDirectionOptionsDialog |
| BulletsAndNumberingBulletsDialog |
| AutoFormatDialog |
| SetLanguage |
| WordCount |
| AutoCorrect |
| EnvelopesAndLabels |
| LabelsDialog |
| MergeCells |
| SplitCells |
| TableRowSelect |
| TableColumnSelect |
| TableSelect |
| TableRepeatHeaderRows |
| ConvertTextToTable |
| TableFormulaDialog |
| TableSplitTable |
| ShowClipboard |
| TechnicalSupport |
| ImeDictionaryUpdate |
| OutlookTaskCreate |
| WindowMinimize |
| WindowRestore |
| WindowClose |
| WindowSaveWorkspace |
| SheetDelete |
| ViewFormulaBar |
| SheetInsert |
| FormatCellsDialog |
| DataFormExcel |
| OutlineSubtotals |
| Consolidate |
| WindowHide |
| WindowUnhide |
| FillUp |
| FillLeft |
| ClearFormats |
| NameCreateFromSelection |
| SheetProtect |
| ReviewProtectWorkbook |
| MacroRelativeReferences |
| Filter |
| SortClear |
| AdvancedFilterDialog |
| OutlineSettings |
| PrintPreviewClose |
| HeaderFooterClose |
| ZoomDialog |
| SortDialogClassic |
| ConvertTableToText |
| PictureInsertFromFilePowerPoint |
| ExchangeFolder |
| VisualBasicReferences |
| ViewCustomViews |
| SheetBackground |
| ChartEditDataSource |
| ChartPlacement |
| CalculateNow |
| ObjectEditDialog |
| ObjectFormatDialog |
| QueryRunQuery |
| ControlImage |
| RulerShowHide |
| GridShowHide |
| ContentsAndIndex |
| Help |
| PivotTableEnableSelection |
| PivotTableListFormulas |
| PivotTableSelectData |
| PivotTableSelectLabelAndData |
| PivotTableSelectLabel |
| CalculateSheet |
| FontColorMoreColorsDialog |
| FillEffects |
| TextOrientationAngleCounterclockwise |
| TextOrientationAngleClockwise |
| HyperlinkOpenExcel |
| OpenStartPage |
| WebGoBack |
| WebGoForward |
| AddToFavorites |
| BrowsePrevious |
| BrowseNext |
| BrowseSelector |
| SmartArtInsert |
| ShapeRerouteConnectors |
| ObjectNudgeUp |
| ObjectNudgeDown |
| ObjectNudgeLeft |
| ObjectNudgeRight |
| ShapeCurve |
| ShapeStraightConnector |
| ShapeElbowConnector |
| ObjectFillMoreColorsDialog |
| ObjectBorderOutlineColorMoreColorsDialog |
| LineStylesDialog |
| ArrowsMore |
| TextEffectAlignment |
| TextEffectTracking |
| WordArtVerticalText |
| WordArtEvenTextHeightClassic |
| ContrastMore |
| ContrastLess |
| BrightnessMore |
| BrightnessLess |
| ShadowNudgeUpClassic |
| ShadowNudgeDownClassic |
| ShadowNudgeLeftClassic |
| ShadowNudgeRightClassic |
| ObjectShadowColorMoreColorsDialog |
| HighImportance |
| LowImportance |
| AttachMenu |
| InviteAttendees |
| AcceptInvitation |
| DeclineInvitation |
| TentativeAcceptInvitation |
| NewContact |
| NewTask |
| NewAppointment |
| TextAlignLeft |
| TextAlignCenter |
| ShapeRectangle |
| ShapeRoundedRectangle |
| ShapeIsoscelesTriangle |
| ShapeOval |
| ShapeSmileyFace |
| ShapeDonut |
| ShapeLeftBrace |
| ShapeRightBrace |
| ShapeArc |
| ShapeLightningBolt |
| ShapeHeart |
| ShapeRightArrow |
| ShapeLeftArrow |
| ShapeUpArrow |
| ShapeDownArrow |
| ShapeRoundedRectangularCallout |
| ShapeStar |
| ShapeSeal8 |
| ShapeSeal16 |
| ShapeSeal24 |
| TextAlignRight |
| TextAlignLetterJustify |
| TextAlignWordJustify |
| TextAlignStretchJustify |
| PictureReset |
| PictureRecolorAutomatic |
| PictureRecolorGrayscale |
| PictureRecolorBlackAndWhite |
| PictureRecolorWashout |
| TextWrappingSquare |
| TextWrappingTight |
| TextWrappingNoneClassic |
| TextWrappingEditWrapPoints |
| \_3DEffectsOnOffClassic |
| \_3DTiltDownClassic |
| \_3DTiltUpClassic |
| \_3DTiltLeftClassic |
| \_3DTiltRightClassic |
| \_3DExtrusionDirectionClassic |
| \_3DLightingClassic |
| \_3DSurfaceMaterialClassic |
| \_3DExtrusionDepthNoneClassic |
| \_3DExtrusionDepth36Classic |
| \_3DExtrusionDepth72Classic |
| \_3DExtrusionDepth144Classic |
| \_3DExtrusionDepth288Classic |
| \_3DExtrusionDepthInfinityClassic |
| \_3DExtrusionPerspectiveClassic |
| \_3DExtrusionParallelClassic |
| \_3DLightingFlatClassic |
| \_3DLightingNormalClassic |
| \_3DLightingDimClassic |
| \_3DSurfaceMatteClassic |
| \_3DSurfacePlasticClassic |
| \_3DSurfaceMetalClassic |
| \_3DSurfaceWireFrameClassic |
| ObjectEditText |
| SnapToShapes |
| TextWrappingMenuClassic |
| WindowsArrangeIcons |
| PictureFormatDialog |
| ViewVisualBasicCode |
| RemoveFromCalendar |
| MasterViewClose |
| CreateShortcutMenuFromMacro |
| DrawingNewClassic |
| HyperlinkInsert |
| HyperlinkEdit |
| ReviewNewComment |
| ReviewPreviousComment |
| ReviewNextComment |
| ReviewDeleteComment |
| ReviewShowOrHideComment |
| ReviewShowAllComments |
| PivotTableOptions |
| DesignMode |
| WordArtInsertDialogClassic |
| FormFieldProperties |
| PhoneticGuideEdit |
| FullScreenViewClassic |
| PhoneticGuideSettings |
| PhoneticGuideFieldShow |
| CircularReferences |
| MasterDocumentExpandOrCollapseSubdocuments |
| Chart3DConeChart |
| InternationalCurrency |
| ObjectsAlignCenterInFormHorixontally |
| ObjectsAlignCenterInFormVertically |
| SizeToControlWidth |
| SizeToControlHeight |
| SizeToControlHeightAndWidth |
| HorizontalSpacingDecrease |
| HorizontalSpacingIncrease |
| ObjectsAlignDistributeHorizontallyRemove |
| VerticalSpacingDecrease |
| VerticalSpacingIncrease |
| ObjectsAlignDistributeVerticallyRemove |
| ObjectsArrangeBottom |
| ObjectsArrangeRight |
| CancelMeeting |
| Private |
| AcceptTask |
| SaveAndNew |
| CopyFolder |
| EmptyTrash |
| RecordInJournal |
| MarkAsUnread |
| CopyToFolder |
| MoveToFolder |
| ShapeFillColorPickerClassic |
| ControlLineColorPicker |
| VisualBasic |
| DoubleBottomBorder |
| BorderThickBottom |
| BorderTopAndBottom |
| BorderTopAndDoubleBottom |
| BorderTopAndThickBottom |
| BordersAll |
| BorderThickOutside |
| SlideBackgroundFormatDialog |
| AutoSummarize |
| ViewDocumentMap |
| ReviewAcceptChange |
| ReviewRejectChange |
| TableDrawBorderPenStyle |
| Font |
| FontSize |
| ZoomClassic |
| CreateMap |
| MergeCellsAcross |
| FieldChooser |
| MessageHeaderToggle |
| MeetingRequest |
| NewNote |
| RecurrenceEdit |
| SendStatusReport |
| ImagerScan |
| QueryReturnGallery |
| SelectRecord |
| SelectAllRecords |
| TableTestValidationRules |
| RelationshipsClearLayout |
| SaveAsQuery |
| LoadFromQuery |
| DatasheetColumnRename |
| FileServerLinkTables |
| DatasheetColumnLookup |
| RecordsRefreshRecords |
| RelationshipsEditRelationships |
| RelationshipsHideTable |
| ReplicationRecoverDesignMaster |
| ReplicationResolveConflicts |
| ReplicationCreateReplica |
| ReplicationSynchronizeNow |
| SetDatabasePassword |
| DatabaseUserLevelSecurityWizard |
| DatabaseUserAndGroupAccounts |
| ControlSpecialEffectEtched |
| ControlSpecialEffectShadowed |
| ControlSpecialEffectChiseled |
| WindowsCascade |
| PositionFitToWindow |
| BorderInsideHorizontal |
| BorderInsideVertical |
| BorderDiagonalDown |
| BorderDiagonalUp |
| MagicEightBall |
| TextDirectionLeftToRight |
| TextDirectionRightToLeft |
| ActiveXCheckBox |
| FindDialogExcel |
| ActiveXTextBox |
| ActiveXButton |
| ActiveXRadioButton |
| ActiveXListBox |
| ActiveXComboBox |
| ActiveXToggleButton |
| ActiveXSpinButton |
| ActiveXScrollBar |
| ActiveXLabel |
| ShowBcc |
| ShowFrom |
| DefinePrintStyles |
| TagMarkComplete |
| NewContactFromSameCompany |
| ChooseForm |
| RecoverInviteToMeeting |
| FormPublish |
| SkipOccurrence |
| RightToLeftDocument |
| EditQuery |
| DataRangeProperties |
| RefreshAll |
| RefreshCancel |
| RefreshStatus |
| ClearAll |
| NewPostInThisFolder |
| ReplyToAttendeesWithMessage |
| SaveAndClose |
| AssignTask |
| Recurrence |
| NewMessageToContact |
| NewTaskForContact |
| NewMeetingWithContact |
| NewMessageToAttendees |
| SendUpdate |
| ReplyToAllAttendeesWithMessage |
| PostReplyToFolder |
| ViewAppointmentInCalendar |
| NewJournalEntry |
| NewMailMessage |
| CancelAcceptTask |
| CancelTaskAssignment |
| CancelDeclineTask |
| NewTaskRequest2 |
| RecurrenceEditSeries |
| DataValidation |
| DataValidationCircleInvalid |
| ReviewShareWorkbook |
| ReviewTrackChanges |
| ReviewHighlightChanges |
| DatabaseQueryNew |
| DataValidationClearValidationCircles |
| ReviewEditComment |
| TableDrawTable |
| TableEraser |
| TableCellAlignTop |
| TableCellAlignCenterVertically |
| TableCellAlignBottom |
| TableColumnsDistribute |
| TableRowsDistribute |
| FileCompactAndRepairDatabase |
| DatabaseMakeMdeFile |
| DatabaseEncodeDecode |
| SizeToTallest |
| SizeToShortest |
| SizeToWidest |
| SizeToNarrowest |
| QueryUnionQuery |
| QueryDataDefinition |
| QuerySqlPassThroughQuery |
| ClearGrid |
| ActiveXFrame |
| ActiveXImage |
| ShapeConnectorStyleStraight |
| ShapeConnectorStyleElbow |
| ShapeConnectorStyleCurved |
| WordArtEditTextClassic |
| FilterByResource |
| TableInsertCellsDialog |
| DeleteCells2 |
| TableDeleteRows |
| TableDeleteColumns |
| Organizer |
| ShadowOnOrOffClassic |
| MacroRecorderStop |
| FileSendAsAttachment |
| AutoSummaryViewByHighlight |
| MasterDocument |
| ChangeCase |
| ListNumFieldInsert |
| ParagraphSpaceBeforeNone |
| ParagraphSpaceBefore |
| ParagraphSpaceAddOrRemoveBefore |
| PagePreviousWord |
| PageNextWord |
| MailMergeRecepientsUseExistingList |
| FootnoteNextWord |
| EndnoteInsertWord |
| IndexMarkEntry |
| CitationMark |
| IndexInsert |
| TableOfContentsDialog |
| TableOfFiguresInsert |
| TableOfAuthoritiesInsert |
| TextBoxLinkCreate |
| TextBoxLinkBreak |
| TextBoxNextLinked |
| TextBoxPreviousLinked |
| CompareAndCombine |
| PrintOptionsMenuWord |
| PageNumberFormat |
| CancelInvitation |
| ContactWebPage |
| AttachItem |
| SendAgain |
| EditComposePage |
| EditReadPage |
| DesignThisForm |
| FileNewDefault |
| FilePrintQuick |
| WindowsTileVertically |
| WindowsTileHorizontally |
| SpellingAndGrammar |
| CopyToPersonalCalendar |
| CopyToPersonalTaskList |
| Post |
| NewOfficeDocument |
| CreateClassModule |
| ControlTabControl |
| ControlPage |
| ReviewPreviousChangeClassic |
| ReviewNextChangeClassic |
| DialMenu |
| SendDefault |
| MessageProperties |
| PictureInsertFromFile |
| TableDrawBorderPenWeight |
| FormatObject2 |
| TableShowGridlines |
| TableBorderPenColorPicker |
| ShowAutoShapesAndDrawingBars |
| ShapeStraightConnectorArrow |
| ShapeElbowConnectorArrow |
| SourceControlAddObjects |
| SourceControlGetLatestVersion |
| SourceControlCheckOut |
| SourceControlCheckIn |
| SourceControlUndoCheckOut |
| SourceControlShareObjects |
| SourceControlShowDifferences |
| SourceControlShowHistory |
| SourceControlRun |
| SourceControlProperties |
| SourceControlCreateDatabaseFromProject |
| SourceControlAddDatabase |
| SourceControlOptions |
| SourceControlRefreshStatus |
| DatabaseMoveToSharePoint |
| NewPostInThisFolder3 |
| SlideDelete |
| ViewHandoutMasterView |
| ViewNotesMasterView |
| SlidesReuseSlides |
| SlidesFromOutline |
| MovieFromFileInsert |
| SoundInsertFromFile |
| FontsReplaceFonts |
| SpeakerNotes |
| BlackAndWhiteAutomatic |
| BlackAndWhiteBlack |
| BlackAndWhiteBlackWithGrayscaleFill |
| BlackAndWhiteBlackWithWhiteFill |
| BlackAndWhiteDontShow |
| BlackAndWhiteGrayWithWhiteFill |
| BlackAndWhiteGrayscale |
| BlackAndWhiteInverseGrayscale |
| BlackAndWhiteWhite |
| BlackAndWhiteLightGrayscale |
| RecordNarration |
| SlideShowSetUpDialog |
| SummarizeSlide |
| TextWrappingTopAndBottom |
| TextWrappingThrough |
| MoveToFolderMenu |
| MacroRecordOrStop |
| PasteAsHyperlink |
| ParagraphDistributed |
| HyphenationOptions |
| TableRowsOrColumnsDistribute |
| MergeOrSplitCells |
| ReviewJapaneseConsistencyChecker |
| ShowGridlines\_HideGridlines |
| AutoSummaryResummarize |
| ViewFooter |
| PictureSetTransparentColor |
| DataRefreshAll |
| BorderTopWord |
| BorderBottomWord |
| BorderLeftWord |
| BorderRightWord |
| TextDirection |
| SlideShowCustom |
| DuplicateSelectedSlides |
| ActionInsert |
| SlideMiniature |
| PivotTableReport |
| ControlSetControlDefaults |
| ControlActiveX |
| FileNewDatabase |
| FileOpenDatabase |
| FileDatabaseProperties |
| DatasheetColumnDelete |
| SelectAllAccess |
| QueryTotalsShowHide |
| MacroConvertMacrosToVisualBasic |
| ViewsDesignView |
| SlideShowInAWindow |
| HangulHanjaConversionPowerPoint |
| ReturnToTaskList |
| MarkTaskComplete |
| NewMailMessage2 |
| PostReply |
| ShowGridOutlook |
| SizeToGridOutlook |
| ControlSnapToGrid |
| MacroSingleStep |
| MicrosoftOnTheWeb01 |
| MicrosoftOnTheWeb02 |
| MicrosoftOnTheWeb03 |
| MicrosoftOnTheWeb04 |
| MicrosoftOnTheWeb05 |
| MicrosoftOnTheWeb06 |
| MicrosoftOnTheWeb07 |
| MicrosoftOnTheWeb08 |
| MicrosoftOnTheWeb09 |
| MicrosoftOnTheWeb10 |
| MicrosoftOnTheWeb11 |
| MicrosoftOnTheWeb12 |
| MicrosoftOnTheWeb13 |
| MicrosoftOnTheWeb14 |
| MicrosoftOnTheWeb15 |
| MicrosoftOnTheWeb16 |
| SendItem |
| SlidesPerPage2Slides |
| SlidesPerPage3Slides |
| SlidesPerPage6Slides |
| SlidesPerPageSlideOutline |
| FontConditionalFormatting |
| ReviewProtectAndShareWorkbook |
| ObjectsAlignLeftOutlook |
| ObjectsAlignRightOutlook |
| ObjectsAlignTopOutlook |
| ObjectsAlignBottomOutlook |
| ObjectsAlignCenterHorizontalOutlook |
| ObjectsAlignMiddleVerticalOutlook |
| MicrosoftOnTheWeb17 |
| SizeToGridAccess |
| SizeToFitAccess |
| FieldsManage |
| HorizontalSpacingMakeEqual |
| VerticalSpacingMakeEqual |
| ObjectsAlignToGridOutlook |
| SortAscendingWord |
| SortDescendingWord |
| OutlineGroup |
| OutlineUngroup |
| IndentIncreaseExcel |
| IndentDecreaseExcel |
| ControlAdvancedProperties |
| SaveAttachments |
| ViewVisualBasicCodeAccess |
| FrameInsertHorizontal |
| FormatPictureOrShapeDialogClassic |
| FileSendToPowerPoint |
| DeclineTask |
| AutoFormatNow |
| DataFormWord |
| ReviseContents |
| BulletListDefault |
| NumberListDefault |
| OutlineNumberDefault |
| FormatNumberDefault |
| FootnoteEndnoteDialog |
| TableInsertDialogWord |
| FormFieldClear |
| ObjectBringInFrontOfText |
| ObjectSendBehindText |
| RotateRight2 |
| RotateLeft2 |
| PageBreakInsertWord |
| BordersShadingDialogWord |
| TextBoxWordClassic |
| WordIndent |
| WordOutdent |
| IndentIncreaseWord |
| IndentDecreaseWord |
| SelectObjects |
| Callout |
| ListMacros |
| ReplaceWithAutoText |
| FormatBackground |
| HangulHanjaConversion |
| HeaderSourceEdit |
| IndentIncrease |
| IndentDecrease |
| AsianLayoutFitText |
| AsianLayoutPhoneticGuide |
| AsianLayoutCombineCharacters |
| JapanesePostcardDialog |
| CharacterBorder |
| CharacterShading |
| MoviePlay |
| SlidesPerPage4Slides |
| SlidesPerPage9Slides |
| ViewWebLayoutView |
| PasteAlternative |
| PasteAsNestedTable |
| HyperlinkRemove |
| MacroSecurity |
| HorizontalLineInsert |
| WebPagePreview |
| TableSelectCell |
| TableDelete |
| TableInsertDialog |
| TableRowsInsertAboveWord |
| TableInsertRowsAbove |
| TableRowsInsertBelowWord |
| TableInsertRowsBelow |
| TableColumnsInsertLeft |
| TableInsertColumnsLeft |
| TableColumnsInsertRight |
| TableInsertColumnsRight |
| TablePropertiesDialog |
| TableOptionsDialog |
| SendCopySendNow |
| SendCopySelectNames |
| SendCopyCheckNames |
| SendCopyOptions |
| SendCopyFlag |
| ConferenceMeetNow |
| SendCopySendToMailRecipient |
| TableOfContentsInFrame |
| ComAddInsDialog |
| FramePropertiesDialog |
| PivotChartInsertClassic |
| PivotFieldListShowHide |
| PictureBulletsInsert |
| FileNewWebPage |
| FileNewEmail |
| ToolboxAudio |
| ToolboxVideo |
| ToolboxMarquee |
| CreateStoredProcedure |
| FileNewBlankDocument |
| FileNewDialogClassic |
| ViewNormalViewPowerPoint |
| SynchronizeHtml |
| IgnoreHtmlChanges |
| FileSaveAsWebPage |
| HorizontalLineInsertClassic |
| GetExternalDataFromWeb |
| FrameCreateAbove |
| FrameCreateBelow |
| FrameCreateLeft |
| FrameCreateRight |
| FrameDelete |
| RelationshipsReport |
| NewView |
| GetExternalDataFromText |
| ServerConnection |
| RecordsInsertSubdatasheet |
| AdpDiagramNewLabel |
| AdpDiagramAddRelatedTables |
| AdpDiagramShowRelationshipLabels |
| AdpDiagramViewPageBreaks |
| AdpDiagramRecalculatePageBreaks |
| AdpDiagramArrangeSelection |
| AdpDiagramArrangeTables |
| AdpDiagramNewTable |
| AdpDiagramColumnProperties |
| AdpDiagramColumnNames |
| AdpDiagramKeys |
| AdpDiagramNameOnly |
| AdpDiagramCustomView |
| AdpDiagramDeleteTable |
| AdpDiagramHideTable |
| AdpDiagramAutosizeSelectedTables |
| DiagramDeleteRelationship |
| WindowMoveSplit |
| EastAsianEditingMarks |
| FilePublishAsWebPage |
| TableAutoFitContents |
| TableAutoFitWindow |
| TableAutoFitFixedColumnWidth |
| TableCellAlignTopLeft |
| TableCellAlignTopCenter |
| TableCellAlignTopRight |
| TableCellAlignMiddleLeft |
| TableCellAlignMiddleCenter |
| TableCellAlignMiddleRight |
| TableCellAlignBottomLeft |
| TableCellAlignBottomCenter |
| TableCellAlignBottomRight |
| ServerFilterApply |
| WebControlCheckBox |
| WebControlOptionButton |
| WebControlDropDownBox |
| WebControlListBox |
| WebControlTextBox |
| WebControlTextArea |
| WebControlSubmit |
| WebControlSubmitWithImage |
| WebControlReset |
| WebControlHidden |
| WebControlPassword |
| TableSetLeftToRight |
| TableSetRightToLeft |
| ChineseTranslationDialog |
| TableInsertMultidiagonalCell |
| AsianLayoutCharactersEnclose |
| ServerFilterByForm |
| EnvelopeChineseDialog |
| AdpVerifySqlSyntax |
| AdpDiagramTableModesMenu |
| CreateDiagram |
| RecordsExpandAllSubdatasheets |
| RecordsCollapseAllSubdatasheets |
| ObjectsMultiSelect |
| DatabaseLinedTableManager |
| TranslateToTraditionalChinese |
| TextWrappingBehindText |
| TextWrappingInFrontOfText |
| WatermarkCustomDialog |
| FrameSaveCurrentAs |
| AdpOutputOperationsTableRemove |
| AdpOutputOperationsGroupBy |
| AdpViewDiagramPane |
| AdpViewGridPane |
| AdpViewSqlPane |
| AdpOutputOperationsAddToOutput |
| AdpOutputOperationsSortAscending |
| AdpOutputOperationsSortDescending |
| DatabaseAccessBackEnd |
| DatabasePartialReplica |
| TextDirectionContext |
| TranslateToSimplifiedChinese |
| ChineseTranslationMenu |
| FileBackUpSqlDatabase |
| ServerRestoreSqlDatabase |
| FileDropSqlDatabase |
| DatabaseSetLogonSecurity |
| DatabaseSqlServer |
| DatabaseSwitchboardManager |
| WebServerDiscussions |
| FontColorMoreColorsDialogPowerPoint |
| HyperlinksVerify |
| BlackAndWhite |
| CancelRequest |
| ViewMasterDocumentViewClassic |
| HyperlinkInsertPowerPoint |
| AddOrRemoveAttendees |
| NewDistributionList |
| CalculateFull |
| SizeMenu |
| ScriptDebugger |
| RunThisForm |
| MessageFormatPlainText |
| MessageFormatHtml |
| MessageFormatRichText |
| ExportToVCardFile |
| MessageOptions |
| MapContactAddress |
| SignatureInsertMenu |
| DesignAFormOutlook |
| WatchWindow |
| FormulaEvaluate |
| LineSpacing |
| WordCountList |
| StylesModifyStyle |
| MacroRun |
| ServerProperties |
| SignatureShow |
| StylesPane |
| SpeechMicrophone |
| SubformInNewWindow |
| NumberingRestart |
| DrawingCanvasInsert |
| ViewsPivotTableView |
| ViewsPivotChartView |
| DiagramRadialInsertClassic |
| DiagramCycleInsertClassic |
| DiagramPyramidInsertClassic |
| DiagramTargetInsertClassic |
| DiagramVennDiagramInsertClassic |
| DiagramChangeToRadialClassic |
| DiagramChangeToCycleClassic |
| DiagramChangeToTargetClassic |
| DiagramChangeToVennDiagramClassic |
| PasteByAppendingTable |
| OrganizationChartInsertAssistant |
| OrganizationChartInsertCoworker |
| OrganizationChartInsertSubordinate |
| AdpStoredProcedureEditSql |
| DrawingCanvasFit |
| DrawingCanvasResize |
| ColorGrayscaleMenu |
| ViewBackToColorView |
| ViewDisplayInGrayscale |
| ViewDisplayInHighContrast |
| ViewDisplayInPureBlackAndWhite |
| PivotAutoFilter |
| PivotSubtotal |
| PivotRefresh |
| PivotMoveToFieldArea |
| PivotMoveToColumnArea |
| PivotMoveToFilterArea |
| PivotMoveToDetailArea |
| PivotExpandField |
| PivotExportToExcel |
| PivotExpandIndicators |
| PivotDropAreas |
| SearchUI |
| MailMergeMergeToEMail |
| MailMergeMergeToFax |
| MailMergeCreateList |
| DrawingCanvasExpand |
| TextWrappingInLineWithText |
| SlidesPerPage1Slide |
| ReviewSendForReview |
| WorkgroupAdmin |
| AdpDiagramAddTable |
| WebComponent |
| PasteSpecial |
| AttachFile |
| DiagramChangeToPyramidClassic |
| DiagramShapeMoveBackwardClassic |
| DiagramShapeMoveForwardClassic |
| SlideMasterPreserveMaster |
| SlideMasterRenameMaster |
| CharacterCodeToggle |
| LinkBarCustom |
| BorderDrawMenu |
| BorderErase |
| BorderStyle |
| StylesStyleVisibility |
| ShowFormulas |
| DatabaseCopyDatabaseFile |
| FileServerTransferDatabase |
| DiagramStylesClassic |
| MailMergeHighlightMergeFields |
| MailMergeWizard |
| OrganizationChartAutoLayout |
| OrganizationChartSelectLevel |
| OrganizationChartSelectBranch |
| OrganizationChartSelectAllAssistants |
| OrganizationChartSelectAllConnectors |
| PivotTableOlapPropertyFields |
| RevealFormatting |
| DiagramReverseClassic |
| DiagramAutoLayoutClassic |
| PasteTableFromExcel |
| TranslationPane |
| GoToTableOfContents |
| TableOfContentsUpdate |
| OutlineLevelGallery |
| OutlineShowLevel |
| ErrorChecking |
| BulletsAndNumberingNumberingDialog |
| NumberingContinue |
| FileCheckOut |
| FileCheckIn |
| OrganizationChartLayoutStandard |
| OrganizationChartLayoutBothHanging |
| OrganizationChartLayoutLeftHanging |
| OrganizationChartLayoutRightHanging |
| PivotTableGenerateGetPivotData |
| ReviewReplyWithChanges |
| StylesStyleSeparator |
| BorderDrawLine |
| BorderDrawGrid |
| OutlinePromoteToHeading |
| MicrosoftOutlook |
| ReviewPreviousChange |
| ReviewNextChange |
| ReviewReviewingPane |
| ReviewAcceptChangeClassic |
| ReviewRejectChangeOrDeleteComment |
| PageOrientationLandscape |
| PageOrientationPortrait |
| GetExternalDataImportClassic |
| SlideReset |
| AnimationOnClick |
| StartAfterPrevious |
| MailMergeMatchFields |
| MailMergeAddressBlockInsert |
| MailMergeGreetingLineInsert |
| MailMergeMergeFieldInsert |
| MailMergeRecipientsEditList |
| PicturesCompress |
| TableAutoFormatStyle |
| PivotAutoCalcMenu |
| PivotChartType |
| PivotChartMultiplePlots |
| PivotChartMultipleUnified |
| PivotSwitchRowColumn |
| DrillInto |
| MailMergeFindRecipient |
| VerticallyDistributed |
| ReviewDeleteCommentPowerPoint |
| ReviewNextCommentPowerPoint |
| ReviewPreviousCommentPowerPoint |
| ReviewShowOrHideMarkup |
| FontSchemes |
| FormFieldReset |
| MailMergeUpdateLabels |
| PivotCollapseFieldAccess |
| PivotShowDetails |
| PivotHideDetails |
| PivotChartLegendShowHide |
| AnimationAudio |
| DiagramFitToContentsClassic |
| DiagramResizeClassic |
| DiagramExpandClassic |
| PivotShowAsMenu |
| OrganizationChartResize |
| AcceptProposal |
| AppointmentColor0 |
| AppointmentColor1 |
| AppointmentColor2 |
| AppointmentColor3 |
| AppointmentColor4 |
| AppointmentColor5 |
| AppointmentColor6 |
| AppointmentColor7 |
| AppointmentColor8 |
| AppointmentColor9 |
| AppointmentColor10 |
| AppointmentColorDialog |
| ProposeNewTime |
| ViewAllProposals |
| ReviewDeleteAllMarkupOnSlide |
| PasteOption |
| MailMergeSetDocumentType |
| DiagramShapeInsertClassic |
| OrganizationChartStyle |
| AdpStoredProcedureQuerySelect |
| AdpStoredProcedureQueryMakeTable |
| AdpStoredProcedureQueryUpdate |
| AdpStoredProcedureQueryAppend |
| AdpStoredProcedureQueryAppendValues |
| AdpStoredProcedureQueryDelete |
| ReviewDisplayForReview |
| ReviewNewCommentMenu |
| ReviewAllowUsersToEditRanges |
| SpeakCells |
| SpeakStop |
| SpeakByRows |
| SpeakByColumns |
| SpeakOnEnter |
| Translate |
| PivotClearCustomOrdering |
| PivotFilterBySelection |
| PivotRemoveField |
| PivotGroupItems |
| PivotUngroupItems |
| DrillOut |
| AdpManageIndexes |
| ViewGridlinesPowerPoint |
| AdpDiagramIndexesKeys |
| AdpDiagramRelationships |
| AdpConstraints |
| PhotoAlbumInsert |
| PhotoAlbumEdit |
| DrawingCanvasScale |
| OrgChartScale |
| DiagramScale |
| ProtectDocument |
| BrightnessAndContrastEdit |
| TableInsertExcel |
| ReadingMode |
| ReadingViewClose |
| PersonaStatusOnline |
| PersonaStatusOffline |
| PersonaStatusAway |
| PersonaStatusBusy |
| GoToMail |
| ResearchPane |
| TableStyleTotalsRow |
| TableRowsInsertAboveExcel |
| TableRowsDeleteExcel |
| TableConvertToRange |
| PrintListRange |
| ReadingModeMini |
| DocumentMapReadingView |
| ReadingViewResearchPane |
| ReadingViewFontSizeIncrease |
| ReadingViewFontSizeDecrease |
| ReadingViewShowPrintedPage |
| ViewRulerWord |
| FileInternetFax |
| XmlExport |
| XmlImport |
| ViewDocumentActionsPane |
| DatabaseObjectDependencies |
| ReviewShowInk |
| InkEraseMode |
| TableColumnsInsertLeftExcel |
| TableColumnsDeleteExcel |
| NewAlert |
| InkDeleteAllInk |
| TableUnlinkExternalData |
| TableExportTableToSharePointList |
| LookUp |
| FilePackageForCD |
| DesignXml |
| WindowSideBySide |
| ViewGridlinesFrontPage |
| ChangeBinding |
| ListSynchronize |
| ChangesDiscardAndRefresh |
| TableOpenInBrowser |
| TableResize |
| XmlExpansionPacksExcel |
| FileVersionHistory |
| FileVersionHistoryWord |
| FrontPageToggleBookmark |
| XmlDataRefresh |
| XmlMapProperties |
| WindowSideBySideSynchronousScrolling |
| WindowResetPosition |
| InkColorMoreColorsDialog |
| XmlTransformation |
| XmlEditQuery |
| JotInkStyle1 |
| InsertDrawingCanvas |
| FilePermissionView |
| FilePermission |
| ReadingViewAllowMultiplePages |
| VoiceInsertInComment |
| RmsInvokeBrowser |
| RmsImportanceCheck |
| RmsImportanceUncheck |
| ViewThumbnails |
| Thesaurus |
| InkingStart |
| VoteMenu |
| RmsNavigationBar |
| RmsSendBizcard |
| RmsSendBizcardDesign |
| FindText |
| AudioNoteDelete |
| ViewFullScreenReadingView |
| VisibilityVisible |
| VisibilityHidden |
| VisibilityInherit |
| InkDrawingAndWriting |
| ReviewInkCommentPen |
| ReviewInkCommentEraser |
| NewInternetFax |
| MeetingsWorkspace |
| InkEraser |
| InkStopInkingReadingView |
| StarUnratedEmpty |
| StarUnratedFull |
| StarRatedEmpty |
| StarRatedFull |
| StarRatedHalf |
| OutlookGlobe |
| OutlookGears |
| MsnLogo |
| GoLeftToRight |
| GoLtrHover |
| GoLtrFocus |
| GoLtrDown |
| GoRtl |
| GoRtlHover |
| GoRtlFocus |
| GoRtlDown |
| StopLeftToRight |
| StopLtrHover |
| StopLtrFocus |
| StopLtrDown |
| StopRtl |
| StopRtlHover |
| StopRtlFocus |
| StopRtlDown |
| ForwardToMobile |
| JunkEmailAddToBlockedSendersList |
| JunkEmailMarkAsNotJunk |
| JunkEmailOptions |
| CategoryCollapse |
| DataSourceCatalogServerScript |
| PermissionRestrict |
| Risks |
| SetPertWeights |
| PanAndZoomWindow |
| FileBackupDatabase |
| OrganizationOutlook |
| ThemeBrowseForThemesPowerPoint |
| AccessNavigationOptions |
| WorkflowComplete |
| WorkflowPending |
| Connections |
| ViewGoBack |
| ViewGoForward |
| ListSetNumberingValue |
| SlideMasterInsertLayout |
| FileViewDigitalSignatures |
| SignaturesLoading |
| AppointmentAttachment |
| ViewsReportView |
| FileCloseDatabase |
| OpenInfopathForm |
| FileWorkflowTasks |
| FileStartWorkflow |
| SharingRequestAllow |
| SharingRequestDeny |
| SharingOpenCalendarFolder |
| SignatureLineInsert |
| SlideThemesGallery |
| ApplicationOptionsDialog |
| BibliographyInsert |
| BibliographyStyle |
| CitationInsert |
| BibliographyManageSources |
| BibliographyAddNewSource |
| LabelInsert |
| BarcodeInsert |
| CongratulatoryEvent |
| CondolatoryEvent |
| ViewPageLayoutView |
| PivotClearAll |
| ChartStylesGallery |
| ChartLayoutGallery |
| ChartSaveTemplates |
| ChartAxisTitles |
| ChartAxes |
| ChartGridlines |
| ChartMoreElement |
| ChartFormatSelection |
| ChartElementSelector |
| PageMarginsGallery |
| LimeCatsAndDogsGallery |
| DropCapInsertGallery |
| TrustCenter |
| ReadingViewAllowTyping |
| ReadingViewMarginSettingsMenu |
| CalendarInsert |
| SlideShowFromCurrent |
| BulletsGallery |
| NumberingGallery |
| LineSpacingGalleryPowerPoint |
| SlideNewGallery |
| SlideLayoutGallery |
| ShapesInsertGallery |
| ShapeChangeShapeGallery |
| ShapeFillTextureGallery |
| ShapeStylesGallery |
| PageOrientationGallery |
| FileServerTasksMenu |
| FileSendMenu |
| GroupFont |
| GroupParagraph |
| GroupStyles |
| GroupProofing |
| GroupInsertPages |
| GroupInsertIllustrations |
| GroupShapes |
| GroupWordArtText |
| GroupPageSetup |
| GroupParagraphLayout |
| GroupCitationsAndBibliography |
| GroupFootnotes |
| GroupTableOfContents |
| GroupMailMergeWriteInsertFields |
| GroupMailMergePreviewResults |
| GroupMailMergeFinish |
| GroupChangesTracking |
| GroupComments |
| GroupChanges |
| GroupCompare |
| GroupPictureSize |
| GroupDrawBorders |
| GroupTableAlignment |
| GroupTableProperties |
| GroupTableTools |
| GroupSlides |
| GroupTextBoxText |
| GroupArrange |
| GroupInsertMediaClips |
| GroupSlideThemes |
| GroupThemes |
| GroupBackground |
| GroupPreview |
| GroupAnimations |
| GroupTransitionToThisSlide |
| GroupSlideShowStart |
| GroupSlideShowSetup |
| GroupShapeStyles |
| GroupFill |
| GroupLines |
| GroupEffects |
| GroupChartLayouts |
| GroupChartStyles |
| GroupChartAxes |
| GroupChartShapes |
| GroupNumber |
| GroupAlignmentExcel |
| GroupCells |
| GroupSortFilter |
| GroupInsertTablesExcel |
| GroupOrganizationChartShapeInsert |
| GroupPageLayoutScaleToFit |
| GroupPageLayoutSheetOptions |
| GroupFunctionLibrary |
| GroupNamedCells |
| GroupFormulaAuditing |
| GroupGetExternalData |
| GroupConnections |
| GroupOutline |
| GroupDataTools |
| GroupChangesExcel |
| FollowUpReadMenu |
| StylesManageStyles |
| StylesStyleInspector |
| ObjectEffectPresetGallery |
| PictureEffectsPresetGallery |
| \_3DRotationGallery |
| ControlLayoutTabular |
| ControlLayoutStacked |
| ControlLayoutRemove |
| ShapeFillColorPicker |
| OutlineColorPicker |
| ResultsPaneStartFindAndReplace |
| FileDocumentInspect |
| ImportSavedImports |
| ImportAccess |
| ImportExcel |
| ImportTextFile |
| ImportSharePointList |
| ImportXmlFile |
| ImportOdbcDatabase |
| ImportHtmlDocument |
| ImportOutlook |
| ImportDBase |
| ImportParadox |
| ImportLotus |
| ExportSavedExports |
| ExportExcel |
| ExportSharePointList |
| ExportWord |
| ExportAccess |
| ExportTextFile |
| ExportXmlFile |
| ExportOdbcDatabase |
| ExportSnapshot |
| ExportHtmlDocument |
| ExportDBase |
| ExportParadox |
| ExportLotus |
| QuickStylesGallery |
| QuickStylesSets |
| ClearFormatting |
| FormRegionOpen |
| FormRegionSave |
| PanningHand |
| BulletsGalleryWord |
| NumberingGalleryWord |
| SharePointListsWorkOffline |
| SynchronizeData |
| SharePointListsDiscardAllChangesAndRefresh |
| FileManageMenu |
| ViewsModeMenu |
| GroupSortAndFilter |
| SortSelectionMenu |
| FiltersMenu |
| RecordsMoreRecordsMenu |
| FieldsMenu |
| RecordsSubdatasheetMenu |
| GroupAutoFormatAccess |
| GroupTextFormatting |
| GroupDataTypeAndFormatting |
| AlignLeftToRightMenu |
| GroupCreateTables |
| GroupCreateForms |
| GroupCreateReports |
| GroupCreateOther |
| GroupSharepointLists |
| GroupCollectData |
| GroupImport |
| ImportMoreMenu |
| GroupExport |
| ExportMoreMenu |
| GroupDatabaseTools |
| DatabasePermissionsMenu |
| GroupMacro |
| GroupDatabaseSourceControl |
| GroupSourceControlShow |
| GroupSourceControlManage |
| GroupRelationships |
| GroupMacroRows |
| GroupQuerySetup |
| GroupQueryType |
| SqlSpecificMenu2 |
| GroupQueryTools |
| GroupControls |
| GroupPageLayoutAccess |
| GroupZoom |
| GroupToolsAccess |
| GroupRichText |
| GroupSizeAndPosition |
| PivotShowTopAndBottomItemsMenu |
| PivotFormulasMenu |
| FilePublishSlides |
| SlideShowUsePresenterView |
| ArrowStyleGallery |
| OutlineDashesGallery |
| OutlineWeightGallery |
| GroupAlignment |
| GroupPictureTools |
| GroupSize |
| FormatCellsNumberDialog |
| FormatCellsFontDialog |
| CellAlignmentOptions |
| PageSetupPageDialog |
| PageSetupSheetDialog |
| PivotTableLayoutGrandTotals |
| GroupPivotTableActiveField |
| GroupPivotTableLayout |
| GroupPivotTableSort |
| GroupPivotTableShowHide |
| PivotTableLayoutSubtotals |
| GroupPivotTableGroup |
| GroupPivotTableTools |
| GroupPivotTableData |
| GroupPivotTableOptions |
| GroupPivotTableStyles |
| GroupPivotTableStyleOptions |
| WrapText |
| ClearMenu |
| ProofingToolsFlyoutAnchor |
| PictureInsertMenu |
| ReviewTrackChangesMenu |
| ReviewAcceptChangeMenu |
| ReviewRejectChangeMenu |
| ReviewBalloonsMenu |
| GroupTableStylesPowerPoint |
| GroupTableRowsAndColumns |
| GroupTableData |
| ObjectAlignMenu |
| MovieInsert |
| ObjectRotateGallery |
| FillMenu |
| SelectMenu |
| OrientationMenu |
| MergeCenterMenu |
| AutoSumMenu |
| PrintAreaMenu |
| PageBreakMenu |
| NameDefineMenu |
| RefreshMenu |
| WhatIfAnalysisMenu |
| PivotTableFormulasMenu |
| PivotTableOlapTools |
| PivotTableOptionsMenu |
| AnimationGallery |
| AnimationTransitionGallery |
| AnimationTransitionSpeedGallery |
| ErrorCheckingMenu |
| TraceRemoveArrowsMenu |
| SortFilterMenu |
| FontColorPicker |
| TableColumnsGallery |
| CellFillColorPicker |
| BorderDoubleBottom |
| ChartTitle |
| ChartPrimaryHorizontalAxisTitle |
| ChartPrimaryVerticalAxisTitle |
| ChartDepthAxisTitle |
| ChartLegend |
| ChartDataLabel |
| ChartPrimaryHorizontalGridlines |
| ChartPrimaryVerticalGridlines |
| ChartDepthGridlines |
| ChartPrimaryHorizontalAxis |
| ChartPrimaryVerticalAxis |
| ChartDepthAxis |
| ChartDataTable |
| ChartTrendline |
| ChartErrorBars |
| ChartLines |
| ChartUpDownBars |
| ChartPlotArea |
| ChartWall |
| ChartFloor |
| GroupMarginsAndPadding |
| ControlLineTypeGallery |
| ControlLineThicknessGallery |
| GroupMacroTools |
| ReplicationOptionsMenu |
| GroupRecords |
| LassoSelect |
| SmartArtAddShape |
| SmartArtIncreaseFontSize |
| SmartArtDecreaseFontSize |
| SmartArtLargerShape |
| SmartArtSmallerShape |
| SmartArtResetGraphic |
| SmartArtResetShape |
| SmartArtTextPane |
| SmartArtEditIn2D |
| SmartArtLayoutGallery |
| SmartArtMoreLayoutsDialog |
| SmartArtStylesGallery |
| SmartArtChangeColorsGallery |
| ObjectEffectSoftEdgesGallery |
| ObjectEffectGlowGallery |
| GradientGallery |
| ObjectEffectShadowGallery |
| WordArtInsertGallery |
| TextEffectTransformGallery |
| TextNoTransform |
| TextPathArchUp |
| TextPathArchDown |
| TextPathCircle |
| TextPathButton |
| TextPlain |
| TextStop |
| TextTriangle |
| TextTriangleInverted |
| TextChevron |
| TextChevronInverted |
| TextRingInside |
| TextRingOutside |
| TextArchUpPour |
| TextArchDownPour |
| TextCirclePour |
| TextButtonPour |
| TextCurveUp |
| TextCurveDown |
| TextCascadeUp |
| TextCascadeDown |
| TextWave1 |
| TextWave2 |
| TextWave3 |
| TextWave4 |
| TextInflate |
| TextDeflate |
| TextInflateBottom |
| TextDeflateBottom |
| TextInflateTop |
| TextDeflateTop |
| TextDeflateInflate |
| TextDeflateInflateDeflate |
| TextFadeRight |
| TextFadeLeft |
| TextFadeUp |
| TextFadeDown |
| TextSlantUp |
| TextSlantDown |
| TextCanUp |
| TextCanDown |
| GroupHeaderFooterExcel |
| HeaderFooterHeaderGallery |
| HeaderFooterFooterGallery |
| GroupHeaderFooterElements |
| HeaderFooterPageNumberInsertExcel |
| HeaderFooterNumberOfPagesInsert |
| HeaderFooterCurrentDate |
| HeaderFooterCurrentTimeInsert |
| HeaderFooterFilePathInsert |
| HeaderFooterFileNameInsert |
| HeaderFooterSheetNameInsert |
| HeaderFooterPictureInsert |
| HeaderFooterFormatPicture |
| GroupHeaderFooterOptions |
| FontShadingColorMoreColorsDialog |
| FontColorMoreColorsDialogExcel |
| BorderMoreColorsDialog |
| SheetTabColorMoreColorsDialog |
| PivotTableNewStyle |
| PivotPlusMinusFieldHeadersShowHide |
| PivotTableExpandField |
| PivotCollapseField |
| ConditionalFormattingDataBarsGallery |
| ConditionalFormattingColorScalesGallery |
| ConditionalFormattingIconSetsGallery |
| TableColumnsInsertRightExcel |
| TableRowsInsertBelowExcel |
| ConditionalFormattingHighlightBetween |
| ConditionalFormattingHighlightCompareColumns |
| PivotTableStylesGallery |
| FormatAsTableGallery |
| TableStylesGalleryExcel |
| ConditionalFormattingsManage |
| ConditionalFormattingHighlightGreaterThan |
| ConditionalFormattingHighlightLessThan |
| ConditionalFormattingHighlightEqualTo |
| ConditionalFormattingHighlightTextContaining |
| ConditionalFormattingHighlightDateOccuring |
| ConditionalFormattingHighlightDuplicateValues |
| ConditionalFormattingTopNItems |
| ConditionalFormattingTopNPercent |
| ConditionalFormattingBottomNItems |
| ConditionalFormattingBottomNPercent |
| ConditionalFormattingAboveAverage |
| ConditionalFormattingBelowAverage |
| RemoveDuplicates |
| FilterReapply |
| CreateEmail |
| ManageReplies |
| ReviewPreviousCommentWord |
| ReviewNextCommentWord |
| ThemeColorsGallery |
| PivotTableInsert |
| PivotChartInsert |
| PivotTableMove |
| PivotTableChangeDataSource |
| BuildingBlocksCreateNewFromSel |
| HeaderInsertGallery |
| FooterInsertGallery |
| CoverPageInsertGallery |
| PageNumberFieldInsertGallery |
| WatermarkGallery |
| EquationInsertGallery |
| QuickTablesInsertGallery |
| QuickPartsInsertGallery |
| SharePointListsDiscardAllChanges |
| SortRemoveAllSorts |
| GroupSmartArtLayouts |
| GroupSmartArtQuickStyles |
| GroupSmartArtChooseColor |
| GroupSmartArtCreateGraphic |
| GroupSmartArtReset |
| GroupSmartArtSize |
| ConditionalFormattingHighlightCellsMenu |
| ConditionalFormattingTopBottomMenu |
| SaveSelectionToQuickPartGallery |
| SaveSelectionToCoverPageGallery |
| SaveSelectionToEquationGallery |
| SaveSelectionToFooterGallery |
| SaveSelectionToHeaderGallery |
| SaveSelectionToPageNumberGallery |
| SaveSelectionToQuickTablesGallery |
| SaveSelectionToWaterMarkGallery |
| MenuView2 |
| FormatCellsMenu |
| ConditionalFormattingClearMenu |
| FontColorCycle |
| CellsInsertSmart |
| CellsDeleteSmart |
| ReviewShowMarkupMenu |
| ObjectEditShapeMenu |
| PivotTableLayoutReportLayout |
| PivotTableLayoutShowInCompactForm |
| PivotTableLayoutShowInOutlineForm |
| PivotTableLayoutShowInTabularForm |
| SymbolInsertGallery |
| ControlChart |
| SpellingAccess |
| ViewsDatasheetView |
| PivotTableClearMenu |
| MacroArguments |
| MacroShowAllActions |
| GroupControlsAccess |
| CreateTable |
| CreateTableTemplatesGallery |
| AccessTableContacts |
| AccessTableTasks |
| AccessTableIssues |
| AccessTableEvents |
| AccessTableAssets |
| CreateTableUsingSharePointListsGallery |
| AccessListContacts |
| AccessListTasks |
| AccessListIssues |
| AccessListEvents |
| AccessListAssets |
| AccessListCustom |
| AccessListCustomDatasheet |
| CreateForm |
| CreateFormSplitForm |
| CreateFormWithMultipleItems |
| CreateFormMoreFormsGallery |
| AccessFormWizard |
| CreateFormPivotChart |
| AccessFormPivotTable |
| AccessFormDatasheet |
| AccessFormModalDialog |
| CreateFormBlankForm |
| CreateReport |
| AccessReportMore |
| CreateReportFromWizard |
| CreateLabels |
| CreateReportBlankReport |
| CreateOtherObjectsMenu |
| ControlTitle |
| ControlLogo |
| ShowMargins |
| SlideMasterContentPlaceholderInsert |
| SlideMasterTextPlaceholderInsert |
| SlideMasterChartPlaceholderInsert |
| SlideMasterTablePlaceholderInsert |
| SlideMasterDiagramPlaceholderInsert |
| SlideMasterMediaPlaceholderInsert |
| SlideMasterClipArtPlaceholderInsert |
| SlideMasterVerticalTextPlaceholderInsert |
| SlideMasterShowFooters |
| CellStylesGallery |
| CellStyleNew |
| CellStylesMerge |
| TableStylesGallery |
| TableStyleNew |
| TableStyleClear |
| FilePublishExcelServices |
| PivotTableOlapConvertToFormulas |
| PivotTableLayoutBlankRows |
| TableStyleFirstColumn |
| TableStyleLastColumn |
| TableStyleBandedRows |
| TableStyleBandedColumns |
| TableStyleRowHeaders |
| TableStyleColumnHeaders |
| TableSummarizeWithPivot |
| DisplayRuler |
| GetExternalDataFromAccess |
| GetExternalDataFromOtherSources |
| GetExternalDataExistingConnections |
| GroupThemesExcel |
| PivotPlusMinusButtonsShowHide |
| FileSaveAsPdfOrXps |
| FormattingDataType |
| FormattingRequiredField |
| FormattingFormat |
| ApplyCurrencyFormat |
| ApplyPercentageFormat |
| ApplyCommaFormat |
| FormattingIncreaseDecimals |
| FormattingDecreaseDecimals |
| BusinessCardInsertMenu |
| RecentFileList |
| DataGraphicIconSet |
| SearchLibraries |
| MoreControlsDialog |
| GroupCode |
| GroupRestrictions |
| GroupXml |
| PageScaleToFitWidth |
| PageScaleToFitHeight |
| SlideTransitionApplyToAll |
| AnimationPreview |
| XmlStructure |
| XmlSchema |
| XmlExpansionPacksWord |
| GroupCaptions |
| GroupIndex |
| GroupTableOfAuthorities |
| GroupEditing |
| SelectMenuExcel |
| PrintTitles |
| NameUseInFormula |
| CalculationOptionsMenu |
| XmlSource |
| GroupClipboard |
| GroupInsertTables |
| GroupInsertLinks |
| GroupInsertSymbols |
| GroupInsertBarcode |
| GroupCalculation |
| BordersGallery |
| BordersMoreDialog |
| PageScaleToFitOptionsDialog |
| PageSizeGallery |
| AnimationTransitionSoundGallery |
| ObjectPictureFill |
| TextWrappingMenu |
| RecordsAddFromOutlook |
| RecordsSaveAsOutlookContact |
| DatasheetNewField |
| PivotTableGroupSelection |
| PivotTableGroupField |
| SlideShowFromBeginning |
| WindowSwitchWindowsMenuWord |
| WindowSwitchWindowsMenuPowerPoint |
| WindowSwitchWindowsMenuExcel |
| ShapeFillMoreGradientsDialog |
| ShadowOptionsDialog |
| BuildingBlocksOrganizer |
| TableSharePointListsModifyColumnsAndSettings |
| TableListAlertMe |
| TableSharePointListsModifyWorkflow |
| TableListPermissions |
| TableSharePointListsRefreshList |
| CreateQueryInDesignView |
| ControlAttachment |
| GroupFieldsAndColumns |
| DataOptionsMenu |
| FormattingUnique |
| RecordsRefreshMenu |
| GroupLayoutShowHide |
| GroupQueryShowHide |
| GroupMacroShowHide |
| GroupPosition |
| GroupAnalyze |
| GroupControlLayout |
| GroupRelationshipsTools |
| GroupTableDesignTools |
| GroupTableRows |
| GroupSharePointList |
| GroupPivotTableFilterAndSort |
| GroupPivotTableToolsAccess |
| GroupPivotTableShowHideAccess |
| GroupPivotTableSelections |
| GroupPivotTableDataAccess |
| GroupPivotTableActiveFieldAccess |
| GroupPivotChartFilterAndSort |
| GroupPivotChartTools |
| GroupPivotChartShowHide |
| GroupPivotChartDataAccess |
| GroupPivotChartActiveFieldAccess |
| GroupPivotChartType |
| ControlMarginsGallery |
| ControlPaddingGallery |
| AutoFormatGallery |
| PivotMoveField |
| PivotChartSortByTotalMenu |
| SharePointListsDiscardChangesMenu |
| ReviewCompareMenu |
| ReviewCompareTwoVersions |
| ReviewCombineRevisions |
| ReviewCompareMajorVersion |
| ReviewCompareLastVersion |
| ReviewCompareSpecificVersion |
| PropertyInsert |
| ObjectsAlignLeftSmart |
| ObjectsAlignRightSmart |
| ObjectsAlignTopSmart |
| ObjectsAlignBottomSmart |
| ObjectsAlignCenterHorizontalSmart |
| ObjectsAlignMiddleVerticalSmart |
| AlignDistributeHorizontally |
| AlignDistributeVertically |
| GroupProtect |
| IndexUpdate |
| MailMergeFinishAndMergeMenu |
| HyphenationMenu |
| MailMergeRules |
| SlideShowCustomMenu |
| SlideShowResolutionGallery |
| SlideShowUseRehearsedTimings |
| SlideShowShowPresentationOnGallery |
| GroupMonitors |
| MailMergeStartMailMergeMenu |
| MailMergeStartLetters |
| MailMergeStartEmail |
| MailMergeStartEnvelopes |
| MailMergeStartLabels |
| MailMergeStartDirectory |
| MailMergeClearMergeType |
| MailMergeSelectRecipients |
| TableOfContentsAddTextGallery |
| FunctionsRecentlyUsedtInsertGallery |
| FunctionsFinancialInsertGallery |
| FunctionsDateTimeInsertGallery |
| FunctionsMathTrigInsertGallery |
| FunctionsTextInsertGallery |
| FunctionsLogicalInsertGallery |
| FunctionsStatisticalInsertGallery |
| FunctionsLookupReferenceInsertGallery |
| FunctionsInformationInsertGallery |
| GroupTableStyleOptions |
| GroupTableExternalData |
| GroupEditingExcel |
| BorderColorPicker |
| TranslationToolTip |
| FileCompatibilityChecker |
| FollowUpComposeMenu |
| ThemeFontsGallery |
| ThemeEffectsGallery |
| PrintColumns |
| PrintDataOnly |
| GroupControlAlignment |
| RmsNavigationBarHome |
| FileProperties |
| ForwardAttach |
| BackAttach |
| ManageAttachments |
| GroupPrintPreviewPrint |
| GroupPrintPreviewPreview |
| GroupPlay |
| GroupMovieOptions |
| GroupSoundOptions |
| SoundPlaySoundGallery |
| SlideShowVolume |
| BreaksGallery |
| LineNumbersMenu |
| GroupOutliningClose |
| GroupOutliningTools |
| GroupMasterDocument |
| TableSelectMenu |
| TableDeleteRowsAndColumnsMenuWord |
| GroupTableMerge |
| TableAutoFitMenu |
| GroupTableDrawBorders |
| TableBordersMenu |
| SortDialog |
| SortAscendingExcel |
| SortDescendingExcel |
| SortCustomExcel |
| EditListItems |
| PublishToPdfOrEdoc |
| FileSaveAsCurrentFileFormat |
| FileSaveAsAccess2007 |
| FileSaveAsAccess2002\_2003 |
| FileSaveAsAccess2000 |
| AcceptAndAdvance |
| RejectAndAdvance |
| FileCreateDocumentWorkspace |
| FileSaveToDocumentManagementServer |
| FileDocumentManagementInformation |
| ScreenNavigatorBack |
| ScreenNavigatorForward |
| FilePrepareMenu |
| FileMarkAsFinal |
| FileAddDigitalSignature |
| ForwardAsBusinessCard |
| ChooseInfoPathForm |
| RecordsTotals |
| StylesPaneNewStyle |
| ChangeCaseGallery |
| ReviewDeleteCommentsMenuPowerPoint |
| GroupMerge |
| AlignJustifyMenu |
| TextBoxInsertMenu |
| TextBoxInsertHorizontal |
| ControlProperties |
| ViewCode |
| GroupHeaderFooterLayout |
| GroupHeaderFooterInsert |
| GroupHeaderFooterNavigation |
| GroupHeaderFooterPosition |
| BlankPageInsert |
| HandoutOrientation |
| SlidesPerPageGallery |
| GroupPlaceholdersHandoutMaster |
| GroupMasterEditTheme |
| GroupMasterClose |
| GroupMasterEdit |
| GroupMasterLayout |
| GroupColorModeSetting |
| ShadowStyleGalleryClassic |
| GroupInkSelect |
| WordArtSpacingMenu |
| TextAlignMenu |
| DiagramChangeToMenuClassic |
| NumberFormatGallery |
| PictureBrightnessGallery |
| PictureContrastGallery |
| PicturePositionGallery |
| GroupPictureReset |
| GroupPictureCompress |
| GroupMailMergeStart |
| OleObjectInsertMenu |
| ViewsLayoutView |
| ForwardAsAttachment |
| InkInsertSpace |
| ShadingColorPicker |
| ShadingColorsMoreColorsDialog |
| SmartArtAddShapeAfter |
| SmartArtAddShapeBefore |
| SmartArtAddShapeAbove |
| SmartArtAddShapeBelow |
| SmartArtAddAssistant |
| ChartSwitchRowColumn |
| ChartShowData |
| ChartRefresh |
| ChartChangeType |
| GroupChartData |
| GroupChartLocation |
| GroupChartType |
| GroupTableSize |
| TableCellMarginsGallery |
| SlideMasterVerticalContentPlaceholderInsert |
| \_3DRotationOptionsDialog |
| \_3DBevelOptionsDialog |
| SlideBackgroundStylesGallery |
| TextDirectionGallery |
| GroupTableStyleOptionsPowerPoint |
| TableStyleBandedRowsPowerPoint |
| TableStyleBandedColumnsPowerPoint |
| SelectionPane |
| SmartArtOrganizationChartLeftHanging |
| SmartArtOrganizationChartRightHanging |
| SmartArtOrganizationChartBoth |
| SmartArtOrganizationChartStandard |
| SmartArtRightToLeft |
| TableCellCustomMarginsDialog |
| ForwardContact |
| PositionAnchoringGallery |
| GroupMoveData |
| FilterAdvancedMenu |
| OmsSend |
| OmsFileSend |
| OmsSave |
| GroupOmsSend |
| GroupOmsInsert |
| GroupOmsView |
| GroupOmsMessageOptions |
| OmsViewAccountSetting |
| OmsInsertSymbolGallery |
| OmsEmoticonStringInsertGallery |
| OmsEmoticonInsertGallery |
| OmsPreviewPane |
| OmsChangeZoom |
| OmsAccountSetup |
| OmsOptions |
| OmsSlideInsert |
| OmsChangeSlideLayoutGallery |
| OmsDelete |
| OmsInsertPicture |
| OmsInsertAudio |
| OmsCustomizeLayout |
| OmsAudioFromFile |
| OmsImageFromFile |
| OmsImageFromClip |
| OmsScanImage |
| OmsNewTextMessage |
| OmsNewMultimediaMessage |
| OmsNewTextMessageToContact |
| OmsNewMultimediaMessageToContact |
| OmsForwardAsTextMessage |
| OmsForwardAsMultimediaMessage |
| OmsMaximumMessages |
| OmsGroupCreateSlides |
| ListLevelGallery |
| MultilevelListGallery |
| BuildingBlockProperties |
| ApplyStylesPane |
| NameManager |
| NameDefine |
| OpenAttachedCalendar |
| RssShareThisFeed |
| OpenInBrowserOutlook |
| InsertAlignmentTab |
| SharingOpenMailFolder |
| SharingOpenContactFolder |
| SharingOpenTaskFolder |
| SharingOpenNotesFolder |
| SharingOpenJournalFolder |
| SharingOpenDocumentFolder |
| ReplyWithInstantMessage |
| DefaultView |
| HideDetails |
| GridlinesWidthGallery |
| GridlinesColorPicker |
| GridlinesStyleGallery |
| SmartArtOrganizationChartMenu |
| SmartArtChangeLayout |
| ReadingViewToolsMenu |
| ReadingViewNextPage |
| ReadingViewPreviousPage |
| ReadingViewShowOnePage |
| ReadingViewShowTwoPages |
| ReadingViewViewOptionsMenu |
| EditWorkflowTask |
| SymbolsDialog |
| ReviewReviewingPaneHorizontal |
| ReviewReviewingPaneVertical |
| \_3DEffectsGalleryClassic |
| \_3DDirectionGalleryClassic |
| \_3DLightingGalleryClassic |
| GroupShadowEffects |
| Group3DEffects |
| WordArtStylesGalleryClassic |
| WordArtInsertGalleryClassic |
| TableInsertGallery |
| ShapeStylesGalleryClassic |
| WordArtChangeShapeGallery |
| ShadowColorPickerClassic |
| \_3DEffectColorPickerClassic |
| ShapeFillGradientGalleryClassic |
| AsianLayoutMenu |
| JapaneseGreetingsInsertMenu |
| AlignJustifyWithMixedLanguages |
| AlignJustifyLow |
| AlignJustifyMedium |
| AlignJustifyHigh |
| AlignJustifyThai |
| ExcelSpreadsheetInsert |
| ParagraphIndentLeft |
| ParagraphIndentRight |
| ParagraphSpacingBefore |
| ParagraphSpacingAfter |
| HeaderFooterPositionHeaderFromTop |
| HeaderFooterPositionFooterFromBottom |
| ControlsGallery |
| ShapeHeight |
| ShapeWidth |
| TransitionTimeAutomaticallyAfter |
| SoundMaximumFileSize |
| CDAudioStartTrack |
| CDAudioStopTrack |
| CDAudioStartTime |
| CDAudioStopTime |
| SlideOrientationGallery |
| TextHighlightColorPicker |
| PageColorPicker |
| GoToHeader |
| GoToFooter |
| UnderlineGallery |
| UnderlineColorPicker |
| TextDirectionGalleryWord |
| GroupAddInsMenuCommands |
| GroupAddInsToolbarCommands |
| ViewFreezePanesGallery |
| GroupInk |
| GroupInkPens |
| GroupInkClose |
| InkBallpointPen |
| InkFeltTipPen |
| InkHighlighter |
| PhoneticGuideMenu |
| PageScaleToFitScale |
| GroupBorder |
| PictureRecolorGalleryWord |
| \_3DSurfaceMaterialGalleryClassic |
| \_3DExtrusionDepthGalleryClassic |
| GroupHeaderFooter |
| GroupPageLayoutSetup |
| GroupPageBackground |
| FunctionsCubeInsertGallery |
| FunctionsEngineeringInsertGallery |
| ThemeSaveCurrentPowerPoint |
| ThemeSaveCurrent |
| TextAlignGallery |
| ThemesGallery |
| CharacterSpacingGallery |
| ChartFormatDataSeries |
| ChartFormatDataLabels |
| ChartResetToMatchStyle |
| ChartSeriesTypeChange |
| Chart3DView |
| ChartFormatDataPoint |
| ChartFormatDataLabel |
| ChartFormatAxis |
| ChartFormatGridlines |
| ChartFormatDisplayUnit |
| ChartFormatLegendEntry |
| ChartFormatChartArea |
| ChartFormatLegend |
| ChartFormatErrorBars |
| ChartFormatUpBars |
| ChartFormatDownBars |
| ChartFormatHighLowLine |
| ChartFormatDropLines |
| ChartFormatTrendline |
| ChartFormatTrendlineLabel |
| ChartFormatSeriesLine |
| ChartFormatDataTable |
| ChartFormatAxisTitle |
| ChartFormatChartTitle |
| ChartFormatFloor |
| ChartFormatSideWall |
| ChartFormatBackWall |
| ChartFormatWalls |
| ChartFormatPlotArea |
| ObjectSizeAndPositionDialog |
| ObjectSizeAndPropertiesDialog |
| ObjectSizeDialog |
| ShapeConvertToFreeform |
| AutoTextGallery |
| TextBoxInsertGallery |
| PageNumbersInHeaderInsertGallery |
| PageNambersInFooterInsertGallery |
| PageNambersInMarginsInsertGallery |
| SaveSelectionToAutoTextGallery |
| SaveSelectionToTextBoxGallery |
| BuildingBlocksCreateLayout |
| SaveSelectionToPageNumberTop |
| SaveSelectionToPageNumberBottom |
| SaveSelectionToPageNumberMargin |
| BuildingBlocksSaveCoverPage |
| BuildingBlocksSaveHeader |
| BuildingBlocksSaveFooter |
| BuildingBlocksSavePageNumTop |
| BuildingBlocksSavePageNumBottom |
| BuildingBlocksSavePageNum |
| HeaderFooterEditHeader |
| TableStyleBandedRowsWord |
| TableStyleBandedColumnsWord |
| MSWordApplyTableStyle |
| TableStyleModify |
| TableStylesGalleryWord |
| ReviewViewChangesInTheSourceDocument |
| GroupThemesWord |
| OfficeDiagnostics |
| WindowsSwitch |
| LayoutOptionsDialog |
| DrawingObjectFormatDialog |
| FontAlternateFillBackColorPicker |
| GroupGroupingAndTotals |

# Index

C

[Change tracking](#section_7e45371b0f1d4ff6b01f2fcbef31233f) 523

Custom UI

[parts](#section_edc80b0591694ff795ee03af067f35b1) 9

Custom UI control id table

[imageMso table](#section_fe2124a15aaa4adfb2855d58da9d5e2a) 419

E

[Elements](#section_d842006e31874f66a17d0819a3cc94b5) 11

[box (box grouping container)](#section_fb660d68051e4a86ab97e0d93b32817e) 11

[button (button inside of a split button)](#section_2257e779b94944cb8ec0719a46421f3e) 34

[button (button)](#section_846e8fb607d3460b816bbcfae841c95b) 15

[button (unsized button)](#section_2e7b9955ff774a2f9d93401f0541e8fd) 25

[buttonGroup (button grouping container)](#section_d9f6097943e740e6bd211f9693600c21) 43

[checkBox (check box)](#section_d189aedcd7634295bc2b07746447dfde) 46

[comboBox (combo box)](#section_3caf6b1834cb4352bd3177a26980c00e) 56

[command (repurposed command)](#section_21316865fbce4a3fa9ffa8277cce5f2d) 67

[commands (list of repurposed commands)](#section_f914543a2b8e48d4bf5e98b1326e7fc2) 69

[contextualTabs (list of contextual tab sets)](#section_75ae6d6f6a8549a2a5fb84b970d54ffb) 69

[control (control clone)](#section_f2bf1b5d049246a488f7dc1a9528f755) 78

[control (Quick Access Toolbar control clone)](#section_14909d674b9f4796badae79edaedcdf2) 87

[control (unsized control clone)](#section_ada8edc4b469410e8bd05d667ffe7dbf) 70

[customUI (custom UI document root)](#section_8a27e8523f8b424aac6732c58181e9d3) 96

[dialogBoxLauncher (dialog box launcher)](#section_76d9f8786b2749ff8c4cee008f345cac) 97

[documentControls (list of document-specific Quick Access Toolbar controls)](#section_de33fcbe3b5b4d30b875d2ec922cf32f) 98

[dropDown (drop-down control)](#section_700e4451870640c58d7b896e4ae21b69) 99

[dynamicMenu (dynamic menu)](#section_26d7db807ea442d09a063f63365043c7) 120

[dynamicMenu (unsized dynamic menu)](#section_fd0825c70f294038861754af0dec8c7d) 111

[editBox (edit box)](#section_7b552756a970466db7c2ee5bbe171e34) 131

[gallery (gallery)](#section_017f9f94700f4846ba19c695ad32929f) 140

[gallery (unsized gallery)](#section_7bfba338481f453e97670e0855c8f029) 156

[group (group)](#section_190f51e63a3c4cf8affbcec9b906aae4) 170

[item (selection item)](#section_6417edcc85b847f781fd56918e8df537) 178

[labelControl (text label)](#section_965555a293db48a6a6bf5dd08a13f6bb) 181

[menu (dynamic menu root XML element)](#section_e662c00135584511a56507a3e32021e3) 219

[menu (menu with title)](#section_f785dc2b045b4b86b5424a10d264dace) 198

[menu (menu)](#section_0243b261903e43d98e8b43f66ea0bcaf) 208

[menu (unsized menu)](#section_1590adf87f2a4ec8944f9c4329e7ff98) 189

[menuSeparator (menu separator)](#section_00c26e34b4a84c97a4f6d547a555a686) 221

[officeMenu (Office menu)](#section_d9422554dbaf42e09e4b6416a80b0844) 224

[qat (Quick Access Toolbar)](#section_93e2f741a06c40968dbf5a97fefca545) 225

[ribbon (ribbon)](#section_574eeee87a03406ab95ff9e51e53dd9d) 226

[separator (separator)](#section_21312cb8be0f412c8184acd533a1410b) 227

[sharedControls (list of shared Quick Access Toolbar controls)](#section_0ca32ee5aae74ff18baa8a8c9be2ebb3) 230

[splitButton (split button with title)](#section_1c6721206402408b96dde98650c12da5) 239

[splitButton (split button)](#section_aa41c698c7e84486b15fb73bedbf2be8) 248

[splitButton (unsized split button)](#section_6477dffcc8f24fc9815b637dabf3911d) 231

[tab (tab)](#section_141f881ca5a4473f944955d3d36579ed) 257

[tabs (list of tabs)](#section_6cbd1ae9780f421886aec536f6c0e97a) 261

[tabSet (contextual tab set)](#section_90a00968474745128601b61440747033) 262

[toggleButton (toggle button inside of a split button)](#section_539b9c44822d45cb8abe66b65a8bc14c) 283

[toggleButton (toggle button)](#section_ec42bfd0149c495b895c3bc708b8a149) 273

[toggleButton (unsized toggle button)](#section_69b127857fce431591c145aa114726ea) 263

G

[Glossary](#section_315de154485a48fca96e8c1a2b320964) 7

I

idMso tables

[Excel 2007](#section_bb5bb594ec8c45f68424f98e8308b25a) 358

[PowerPoint 2007](#section_f2a8e3c014cb4ad388cda8b5b1b9a8a0) 393

[Word 2007](#section_a9f90d736e694bb78e384f268cd42038) 304

[Informative references](#section_dbe3a851c6b94ae0ba7984f1e88432d1) 8

[Introduction](#section_5b5f9bab46874f838f2cb743c813e4c3) 7

N

[Normative references](#section_ea98067859b3466892052fedb6615013) 8

P

Parts

[additional part types](#section_728fd283b4524a14ba8c33100cf6f088) 9

[quick access toolbar customizations part](#section_087c607c5565418c8d31772b91d93ff3) 9

[ribbon extensibility part](#section_52faf7b6fecc48d996dbee80a631a5ac) 10

[Product behavior](#section_30685057c9f24d9c82acf3274539540f) 522

R

References

[informative](#section_dbe3a851c6b94ae0ba7984f1e88432d1) 8

[normative](#section_ea98067859b3466892052fedb6615013) 8

S

[Simple types](#section_869c8c9a45f84119b068f61e76d04322) 292

[ST\_BoxStyle (box style)](#section_3a299df5af064f4f813db0043d772546) 292

[ST\_Delegate (callback function name)](#section_188cc098eef7453e895cd96e8e99576c) 292

[ST\_GalleryItemWidthHeight (gallery item width or height)](#section_76f0a96c409d49a5927028e9b6b7f0a4) 295

[ST\_GalleryRowColumnCount (gallery row or column count)](#section_b09148682e5949dd96cfdf995849a1bf) 295

[ST\_ID (control identifier)](#section_18fbbd72e04f48148f9a45b2c4cc8393) 296

[ST\_ItemSize (menu item size)](#section_c65c8b6052e5404dad5a8c09627c60eb) 297

[ST\_Keytip (key tip)](#section_78170415e2fa4fc0899a76096fdc6aa6) 297

[ST\_LongString (long string)](#section_94ee0d879533413ebf87bbee57270f7e) 298

[ST\_QID (qualified control identifier)](#section_7c80b304b2f8432a83ee7f8ec792af0c) 298

[ST\_Size (control size)](#section_acf53d2ea68046de946752a2f4a51631) 300

[ST\_String (short string)](#section_d104fcb261774eb9a4000a5f8ddcd539) 301

[ST\_StringLength (string length)](#section_4a327c6229ec496e935fa1aa77d26d5e) 302

[ST\_UniqueID (custom control identifier)](#section_ed3ee661bfc848d3b1598e8917a2bd36) 302

[ST\_Uri (image relationship identifier)](#section_5512155ce46d445ea7273b99b4009cc6) 303

T

Tables

[idMso table – Excel 2007](#section_bb5bb594ec8c45f68424f98e8308b25a) 358

[idMso table – PowerPoint 2007](#section_f2a8e3c014cb4ad388cda8b5b1b9a8a0) 393

[idMso table – Word 2007](#section_a9f90d736e694bb78e384f268cd42038) 304

[imageMso table](#section_fe2124a15aaa4adfb2855d58da9d5e2a) 419

[Tracking changes](#section_7e45371b0f1d4ff6b01f2fcbef31233f) 523