#### protectedRange (Protected Range)

A specified range to be protected. Ranges listed here are protected only when the [sheet](sheet.docx) [protection](protection.docx) is ON and the [cell](cell.docx) is flagged as being locked. If no password is specified here, then read/write permissions are automatically given to all [users](users.docx), regardless of additional security descriptor information. In other words, the security descriptor information (specific types of access) at the user level is only applied if a password for this range is specified.

When a password is specified, then [users](users.docx) not listed specifically as having access should be prompted with a password. If that user supplies the correct password, then they may edit the range or [cell](cell.docx) in question. This [protection](protection.docx) is optional and may be ignored by applications who choose not to support this functionality.

|  |
| --- |
| Parent Elements |
| [protectedRanges](protectedRanges.docx) (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| [name](name.docx) (Name) | Range title. This is used as a descriptor, not as a named range definition.The possible values for this attribute are defined by the [ST\_Xstring](ST_Xstring.docx) simple type (§). |
| password (Password) | Specifies the hash of the password required for editing this range. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the logic defined in the revisionsPassword attribute of §.The resulting value is hashed using the algorithm defined below.[Note: An example algorithm to hash the user input into the value stored is as follows:// Function Input:// szPassword: NULL terminated C-Style string// cchPassword: The number of characters in szPassword (not including the NULL terminator)WORD GetPasswordHash(const [CHAR](CHAR.docx) \*szPassword, int cchPassword) { WORD wPasswordHash; const [CHAR](CHAR.docx) \*pch;  wPasswordHash = 0;  if (cchPassword > 0) { pch = &szPassword[cchPassword]; while (pch-- != szPassword) { wPasswordHash = ((wPasswordHash >> 14) & 0x01) | ((wPasswordHash << 1) & 0x7fff); wPasswordHash ^= \*pch; } wPasswordHash ^= (0x8000 | ('N' << 8) | 'K'); }  return(wPasswordHash);}end note]The possible values for this attribute are defined by the [ST\_UnsignedShortHex](ST_UnsignedShortHex.docx) simple type (§). |
| securityDescriptor (Security Descriptor) | Optional setting to specify the relative security descriptor. The security descriptor defines user accounts who may edit this range without providing a password to access the range. Removing this attribute shall remove all permissions granted or denied to [users](users.docx) for this range.The possible values for this attribute are defined by the XML [Schema](Schema.docx) string datatype. |
| sqref (Sequence of References) | The range to be protected.The possible values for this attribute are defined by the [ST\_Sqref](ST_Sqref.docx) simple type (§). |

The following XML [Schema](Schema.docx) fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_ProtectedRange">

 <attribute [name](name.docx)="password" type="[ST\_UnsignedShortHex](ST_UnsignedShortHex.docx)" use="optional"/>

 <attribute [name](name.docx)="sqref" type="[ST\_Sqref](ST_Sqref.docx)" use="required"/>

 <attribute [name](name.docx)="[name](name.docx)" type="[ST\_Xstring](ST_Xstring.docx)" use="required"/>

 <attribute [name](name.docx)="securityDescriptor" type="xsd:string" use="optional"/>

</complexType>