Inhaltsverzeichnis

Modul:Effective protection level/Doku

Dies ist die Dokumentationsseite für Modul:Effective protection level

This module provides a way to retrieve the group required to perform a given action on a page. It currently tests the following criteria:

- The page being pending-changes protected: autoconfirmed
- The page being a JavaScript or CSS subpage in userspace, or in the MediaWiki namespace: interfaceadmin
- The page being in the MediaWiki namespace: sysop
- The page being a JSON subpage in userspace: sysop
- The page being protected: sysop, templateeditor, extendedconfirmed, or autoconfirmed
- The page being used in a cascading-protected page: sysop
- The page's title matching the titleblacklist: templateeditor or autoconfirmed
- A file being moved: filemover
- A page being moved or a file being uploaded: autoconfirmed
- A non-Draft non-talk page being created: user
- Anything else: *

Note that if a template-protected file is moved, both filemover and templateeditor are required, but this will return only templateeditor. This is not likely to be changed any time soon, since template protection currently shouldn't be used on files.

Inhaltsverzeichnis

Usage

Warning: This module will use up to 4 expensive parser function calls each time it is ran. It should only be used if the exact effective protection level is necessary. Otherwise, consider using title.protectionLevels instead.

From other modules

To load this module:

```
local effectiveProtectionLevel = require('Module:Effective protection level')._ma
```

The function accepts two parameters. The first is a string containing the action to check, which must be one of "edit", "create", "move", "upload", "undelete", or "autoreview". The second is optional, and can either be the name of the page to check, or a title returned from the mw.title functions. If the second parameter is omitted, the page being displayed is the one checked against. The return value is a string containing the name of the group required to perform the given action.

From wikitext

The parameters are the same as when it is called directly.

```
Vorlage:Tlinv
```

See also

- Module:Effective protection expiry

Modul:Effective protection expiry

This module provides a way to retrieve the expiry of a restriction over a given action on a page.

Inhaltsverzeichnis

Usage

This module will use up to 1 expensive parser function call each time it is ran. It will not use any if Module:Effective protection level was already called.

From other modules

To load this module:

```
local effectiveProtectionExpiry = require('Module:Effective protection expiry').
```

The function accepts two parameters. The first is a string containing the action to check, which must be one of "edit", "create", "move", "upload", or "autoreview". The second is optional, and can either be the name of the page to check, or a title returned from the mw.title functions. If the second parameter is omitted, the page being displayed is the one checked against.

The return value is either a date string in `YY-MM-DDThh:mm:ss` format, or one of the following strings:

- `infinity` - for pages protected indefinitely, or pages which exist and are not protected
- `unknown` - for pages where the expiry is unknown, or pages which do not exist and are not protected

Note that if an existing page is not protected for the requested action, this will return 'infinity'. You need to check separately with Module:Effective protection level.

From wikitext

The parameters are the same as when it is called directly.

```
Vorlage:Tlinv
```

See also

- Module:Effective protection level

```
local p = {}

-- Returns the expiry of a restriction of an action on a given title, or unknown
-- If no title is specified, the title of the page being displayed is used.
function p._main(action, pagename)
	local title
	if type(pagename) == 'table' and pagename.prefixedText then
		title = pagename
	elseif pagename then
		title = mw.title.new(pagename)
	else
		title = mw.title.getCurrentTitle()
	end
	pagename = title.prefixedText
	if action == 'autoreview' then
		local stabilitySettings = mw.ext.FlaggedRevs.getStabilitySettings
		return stabilitySettings and stabilitySettings.expiry or 'unknown
	elseif action ~= 'edit' and action ~= 'move' and action ~= 'create' and a
		error( 'First parameter must be one of edit, move, create, upload
	end
	local rawExpiry = mw.getCurrentFrame():callParserFunction('PROTECTIONEXPI
	if rawExpiry == 'infinity' then
		return 'infinity'
	elseif rawExpiry == '' then
		return 'unknown'
	else
		local year, month, day, hour, minute, second = rawExpiry:match(
			'^(%d%d%d%d)(%d%d)(%d%d)(%d%d)(%d%d)(%d%d)$'
		)
		if year then
			return string.format(
				'%s-%s-%sT%s:%s:%s',
				year, month, day, hour, minute, second
			)
		else
			error('internal error in Module:Effective protection expi
		end
	end
end

setmetatable(p, { __index = function(t, k)
	return function(frame)
		return t._main(k, frame.args[1])
	end
end })

return p
```

Modul:Effective protection level

This module provides a way to retrieve the group required to perform a given action on a page. It currently tests the following criteria:

- The page being pending-changes protected: autoconfirmed
- The page being a JavaScript or CSS subpage in userspace, or in the MediaWiki namespace: interfaceadmin
- The page being in the MediaWiki namespace: sysop
- The page being a JSON subpage in userspace: sysop
- The page being protected: sysop, templateeditor, extendedconfirmed, or autoconfirmed
- The page being used in a cascading-protected page: sysop
- The page's title matching the titleblacklist: templateeditor or autoconfirmed
- A file being moved: filemover
- A page being moved or a file being uploaded: autoconfirmed
- A non-Draft non-talk page being created: user
- Anything else: *

Note that if a template-protected file is moved, both filemover and templateeditor are required, but this will return only templateeditor. This is not likely to be changed any time soon, since template protection currently shouldn't be used on files.

Inhaltsverzeichnis

Usage

Warning: This module will use up to 4 expensive parser function calls each time it is ran. It should only be used if the exact effective protection level is necessary. Otherwise, consider using title.protectionLevels instead.

From other modules

To load this module:

```
local effectiveProtectionLevel = require('Module:Effective protection level')._ma
```

The function accepts two parameters. The first is a string containing the action to check, which must be one of "edit", "create", "move", "upload", "undelete", or "autoreview". The second is optional, and can either be the name of the page to check, or a title returned from the mw.title functions. If the second parameter is omitted, the page being displayed is the one checked against. The return value is a string containing the name of the group required to perform the given action.

From wikitext

The parameters are the same as when it is called directly.

```
Vorlage:Tlinv
```

See also

- Module:Effective protection expiry

```
local p = {}

-- Returns the permission required to perform a given action on a given title.
-- If no title is specified, the title of the page being displayed is used.
function p._main(action, pagename)
	local title
	if type(pagename) == 'table' and pagename.prefixedText then
		title = pagename
	elseif pagename then
		title = mw.title.new(pagename)
	else
		title = mw.title.getCurrentTitle()
	end
	pagename = title.prefixedText
	if action == 'autoreview' then
		local level = mw.ext.FlaggedRevs.getStabilitySettings(title)
		level = level and level.autoreview
		if level == 'review' then
			return 'reviewer'
		elseif level ~= '' then
			return level
		else
			return nil -- not '*'. a page not being PC-protected is c
		end
	elseif action ~= 'edit' and action ~= 'move' and action ~= 'create' and a
		error( 'First parameter must be one of edit, move, create, uploa
	end
	if title.namespace == 8 then -- MediaWiki namespace
		if title.text:sub(-3) == '.js' or title.text:sub(-4) == '.css' or
			return 'interfaceadmin'
		else -- any non-JS/CSS MediaWiki page
			return 'sysop'
		end
	elseif title.namespace == 2 and title.isSubpage then
		if title.contentModel == 'javascript' or title.contentModel == 'c
			return 'interfaceadmin'
		elseif title.contentModel == 'json' then -- user JSON page
			return 'sysop'
```

```
                end
        end
        if action == 'undelete' then
                return 'sysop'
        end
        local level = title.protectionLevels[action] and title.protectionLevels[a
        if level == 'sysop' or level == 'editprotected' then
                return 'sysop'
        elseif title.cascadingProtection.restrictions[action] and title.cascading
                return 'sysop'
        elseif level == 'templateeditor' then
                return 'templateeditor'
        elseif action == 'move' then
                local blacklistentry = mw.ext.TitleBlacklist.test('edit', pagenam
                if blacklistentry and not blacklistentry.params.autoconfirmed the
                        return 'templateeditor'
                elseif title.namespace == 6 then
                        return 'filemover'
                elseif level == 'extendedconfirmed' then
                        return 'extendedconfirmed'
                else
                        return 'autoconfirmed'
                end
        end
        local blacklistentry = mw.ext.TitleBlacklist.test(action, pagename)
        if blacklistentry then
                if not blacklistentry.params.autoconfirmed then
                        return 'templateeditor'
                elseif level == 'extendedconfirmed' then
                        return 'extendedconfirmed'
                else
                        return 'autoconfirmed'
                end
        elseif level == 'editsemiprotected' then -- create-semiprotected pages re
                return 'autoconfirmed'
        elseif level then
                return level
        elseif action == 'upload' then
                return 'autoconfirmed'
        elseif action == 'create' and title.namespace % 2 == 0 and title.namespac
                return 'user'
        else
                return '*'
        end
end

setmetatable(p, { __index = function(t, k)
        return function(frame)
                return t._main(k, frame.args[1])
        end
end })

return p
```