



Inhaltsverzeichnis

1. Modul:Protection banner/config	2
2. Modul:Protection banner	19

Modul:Protection banner/config

This module contains configuration data for [Module:Protection banner](#). For documentation please see the module comments, and if you're not sure how something works you can ask on the [module talk page](#).

```
-- This module provides configuration data for [[Module:Protection banner]].
return {
-----
--
--                                     BANNER DATA
-----

--[[
-- Banner data consists of six fields:
-- * text - the main protection text that appears at the top of protection
--   banners.
-- * explanation - the text that appears below the main protection text, used
--   to explain the details of the protection.
-- * tooltip - the tooltip text you see when you move the mouse over a small
--   padlock icon.
-- * link - the page that the small padlock icon links to.
-- * alt - the alt text for the small padlock icon. This is also used as tooltip
--   text for the large protection banners.
-- * image - the padlock image used in both protection banners and small padlock
--   icons.
--
-- The module checks in three separate tables to find a value for each field.
-- First it checks the banners table, which has values specific to the reason
-- for the page being protected. Then the module checks the defaultBanners
-- table, which has values specific to each protection level. Finally, the
-- module checks the masterBanner table, which holds data for protection
-- templates to use if no data has been found in the previous two tables.
--
-- The values in the banner data can take parameters. These are specified
-- using #{TEXTLIKETHIS} (a dollar sign preceding a parameter name
-- enclosed in curly braces).
--
--                                     Available parameters:
--
-- #{CURRENTVERSION} - a link to the page history or the move log, with the
-- display message "current-version-edit-display" or
-- "current-version-move-display".
--
-- #{EDITREQUEST} - a link to create an edit request for the current page.
--
-- #{EXPLANATIONBLURB} - an explanation blurb, e.g. "Please discuss any changes
-- on the talk page; you may submit a request to ask an administrator to make
-- an edit if it is minor or supported by consensus."
--
-- #{IMAGELINK} - a link to set the image to, depending on the protection
-- action and protection level.
--
--
```



```
-- ${INTROBLURB} - the PROTECTIONBLURB parameter, plus the expiry if an expiry
-- is set. E.g. "Editing of this page by new or unregistered users is currently
-- disabled until dd Month YYYY."
--
-- ${INTROFRAGMENT} - the same as ${INTROBLURB}, but without final punctuation
-- so that it can be used in run-on sentences.
--
-- ${PAGETYPE} - the type of the page, e.g. "article" or "template".
-- Defined in the cfg.pagetypes table.
--
-- ${PROTECTIONBLURB} - a blurb explaining the protection level of the page, e.g
-- "Editing of this page by new or unregistered users is currently disabled"
--
-- ${PROTECTIONDATE} - the protection date, if it has been supplied to the
-- template.
--
-- ${PROTECTIONLEVEL} - the protection level, e.g. "fully protected" or
-- "semi-protected".
--
-- ${PROTECTIONLOG} - a link to the protection log or the pending changes log,
-- depending on the protection action.
--
-- ${TALKPAGE} - a link to the talk page. If a section is specified, links
-- straight to that talk page section.
--
-- ${TOOLTIPBLURB} - uses the PAGETYPE, PROTECTIONTYPE and EXPIRY parameters to
-- create a blurb like "This template is semi-protected", or "This article is
-- move-protected until DD Month YYYY".
--
-- ${VANDAL} - links for the specified username (or the root page name)
-- using Module:Vandal-m.
--
--                                     Functions
--
-- For advanced users, it is possible to use Lua functions instead of strings
-- in the banner config tables. Using functions gives flexibility that is not
-- possible just by using parameters. Functions take two arguments, the
-- protection object and the template arguments, and they must output a string.
--
-- For example:
--
-- text = function (protectionObj, args)
--     if protectionObj.level == 'autoconfirmed' then
--         return 'foo'
--     else
--         return 'bar'
--     end
-- end
--
-- Some protection object properties and methods that may be useful:
-- protectionObj.action - the protection action
-- protectionObj.level - the protection level
-- protectionObj.reason - the protection reason
-- protectionObj.expiry - the expiry. Nil if unset, the string "indef" if set
-- to indefinite, and the protection time in unix time if temporary.
-- protectionObj.protectionDate - the protection date in unix time, or nil if
-- unspecified.
-- protectionObj.bannerConfig - the banner config found by the module. Beware
-- of editing the config field used by the function, as it could create an
-- infinite loop.
-- protectionObj:isProtected - returns a boolean showing whether the page is
-- protected.
-- protectionObj:isTemporary - returns a boolean showing whether the expiry is
-- temporary.
```

```
-- protectionObj:isIncorrect - returns a boolean showing whether the protection
-- template is incorrect.
--]]

-- The master banner data, used if no values have been found in banners or
-- defaultBanners.
masterBanner = {
    text = '${INTROBLURB}',
    explanation = '${EXPLANATIONBLURB}',
    tooltip = '${TOOLTIPBLURB}',
    link = '${IMAGELINK}',
    alt = 'Page ${PROTECTIONLEVEL}'
},

-- The default banner data. This holds banner data for different protection
-- levels.
-- *required* - this table needs edit, move, autoreview and upload subtables.
defaultBanners = {
    edit = {},
    move = {},
    autoreview = {
        default = {
            alt = 'Page protected with pending changes',
            tooltip = 'All edits by unregistered and new users are su
            image = 'Pending-protection-shackle.svg'
        }
    },
    upload = {}
},

-- The banner data. This holds banner data for different protection reasons.
-- In fact, the reasons specified in this table control which reasons are
-- valid inputs to the first positional parameter.
--
-- There is also a non-standard "description" field that can be used for items
-- in this table. This is a description of the protection reason for use in the
-- module documentation.
--
-- *required* - this table needs edit, move, autoreview and upload subtables.
banners = {
    edit = {
        blp = {
            description = 'For pages protected to promote compliance
            .. ' [[Wikipedia:Biographies of living persons'
            .. '|biographies of living persons]] policy',
            text = '${INTROFRAGMENT} to promote compliance with'
            .. ' [[Wikipedia:Biographies of living persons'
            .. "|Wikipedia's&nbsp;policy on&nbsp;the&nbsp;bi
            .. ' of&nbsp;living&nbsp;people]].',
            tooltip = '${TOOLTIPFRAGMENT} to promote compliance with
            .. ' biographies of living persons',
        },
        dmca = {
            description = 'For pages protected by the Wikimedia Found
            .. ' due to [[Digital Millennium Copyright Act]]
            explanation = function (protectionObj, args)
                local ret = 'Pursuant to a rights owner notice ur
                .. ' Millennium Copyright Act (DMCA) rega
                .. ' in this article, the Wikimedia Found
                .. ' applicable law and took down and res
                .. ' in question.'
                if args.notice then
                    ret = ret .. ' A copy of the received not
                    .. args.notice .. '.'
```

```
end
ret = ret .. ' For more information, including we
.. ' how to file a counter-notice, please
.. " [[Wikipedia:Office actions]] and the
.. "'Do not remove this template from t
.. " restrictions are withdrawn''.'"
return ret
end,
image = 'Office-protection-shackle.svg',
},
dispute = {
description = 'For pages protected due to editing dispute
text = function (protectionObj, args)
-- Find the value of "disputes".
local display = 'disputes'
local disputes
if args.section then
disputes = string.format(
'[[%S:%S#%S|%S]]',
mw.site.namespaces[protectionObj
protectionObj.title.text,
args.section,
display
)
else
disputes = display
end

-- Make the blurb, depending on the expiry.
local msg
if type(protectionObj.expiry) == 'number' then
msg = '${INTROFRAGMENT} or until editing
else
msg = '${INTROFRAGMENT} until editing %s
end
return string.format(msg, disputes)
end,
explanation = "This protection is '''not''' an endorsement
.. ' ${CURRENTVERSION}. ${EXPLANATIONBLURB}',
tooltip = '${TOOLTIPFRAGMENT} due to editing disputes',
},
ecp = {
description = 'For articles in topic areas authorized by
.. ' [[Wikipedia:Arbitration Committee|ArbCom]] c
.. ' meets the criteria for community use',
tooltip = 'This ${PAGETYPE} is extended-confirmed protect
alt = 'Extended-protected ${PAGETYPE}',
},
mainpage = {
description = 'For pages protected for being displayed on
text = 'This file is currently'
.. ' [[Wikipedia:This page is protected|protected
.. ' editing because it is currently or will soon
.. ' on the [[Main Page]].',
explanation = 'Images on the Main Page are protected due
.. ' visibility. Please discuss any necessary cha
.. '<br /><span style="font-size:90%;">'
.. "'Administrators:'' Once this image is defi
.. ' please unprotect this file, or reduce to sen
.. ' as appropriate.</span>',
},
office = {
description = 'For pages protected by the Wikimedia Found
text = function (protectionObj, args)
```

```
        local ret = 'This ${PAGETYPE} is currently under
            .. ' scrutiny of the'
            .. ' [[Wikipedia:Office actions|Wikimedia
            .. ' and is protected.'
        if protectionObj.protectionDate then
            ret = ret .. ' It has been protected since
        end
        return ret
    end,
    explanation = "If you can edit this page, please discuss
        .. " additions on the ${TALKPAGE} first. ''Do not
        .. " page unless you are authorized by the Wikimedia
        .. " so.'''",
    image = 'Office-protection-shackle.svg',
},
reset = {
    description = 'For pages protected by the Wikimedia Found
        .. ' "reset" to a bare-bones version',
    text = 'This ${PAGETYPE} is currently under the'
        .. ' scrutiny of the'
        .. ' [[Wikipedia:Office actions|Wikimedia
        .. ' and is protected.',
    explanation = function (protectionObj, args)
        local ret = ''
        if protectionObj.protectionDate then
            ret = ret .. 'On ${PROTECTIONDATE} this $
        else
            ret = ret .. 'This ${PAGETYPE} has been'
        end
        ret = ret .. ' reduced to a'
        .. ' simplified, "bare bones" version so that it
        .. ' rewritten to ensure it meets the policies of
        .. ' [[WP:NPOV|Neutral Point of View]] and [[WP:V
        .. ' Standard Wikipedia policies will apply to it
        .. ' will eventually be open to all editors—and v
        .. ' enforced. The ${PAGETYPE} has been ${PROTECT
        .. ' it is being rebuilt.\n\n'
        .. 'Any insertion of material directly from'
        .. ' pre-protection revisions of the ${PAGETYPE}
        .. ' will any material added to the ${PAGETYPE} t
        .. ' sourced. The associated talk page(s) were al
        .. " same date.\n\n"
        .. "If you can edit this page, please discuss all
        .. " additions on the ${TALKPAGE} first. ''Do not
        .. " this action, and do not remove protection fr
        .. " unless you are authorized by the Wikimedia F
        .. " to do so. No editor may remove this notice.

        return ret
    end,
    image = 'Office-protection-shackle.svg',
},
sock = {
    description = 'For pages protected due to'
        .. ' [[Wikipedia:Sock puppetry|sock puppetry]]',
    text = '${INTROFRAGMENT} to prevent [[Wikipedia:Sock pupp
        .. ' [[Wikipedia:Blocking policy|blocked]] or'
        .. ' [[Wikipedia:Banning policy|banned users]]'
        .. ' from editing it.',
    tooltip = '${TOOLTIPFRAGMENT} to prevent sock puppets of
        .. ' editing it',
},
template = {
    description = 'For [[Wikipedia:High-risk templates|high-t
```

```
.. ' templates and Lua modules',
text = 'This is a permanently [[Help:Protection|protected
.. ' as it is [[Wikipedia:High-risk templates|high
explanation = 'Please discuss any changes on the ${TALKPAGE}
.. ' ${EDITREQUEST} to ask an
.. ' [[Wikipedia:Administrators|administrator]] or
.. ' [[Wikipedia:Template editor|template editor]]
.. ' it is [[Help:Minor edit#When to mark an edit
.. ' |uncontroversial]] or supported by'
.. ' [[Wikipedia:Consensus|consensus]]. You can a
.. ' [[Wikipedia:Requests for page protection|reque
.. ' unprotected.',
tooltip = 'This high-risk ${PAGETYPE} is permanently ${PAGE
.. ' to prevent vandalism',
alt = 'Permanently protected ${PAGETYPE}',
},
usertalk = {
description = 'For pages protected against disruptive edi
.. ' particular user',
text = '${INTROFRAGMENT} to prevent ${VANDAL} from using
.. ' such as abusing the'
.. ' &#123;&#123;[[Template:unblock|unblock]]&#12
explanation = 'If you cannot edit this user talk page and
.. ' make a change or leave a message, you can'
.. ' [[Wikipedia:Requests for page protection'
.. ' #Current requests for edits to a protected pa
.. ' |request an edit]],'
.. ' [[Wikipedia:Requests for page protection'
.. ' #Current requests for reduction in protection
.. ' |request unprotection]],'
.. ' [[Special:Userlogin|log in]],'
.. ' or [[Special:UserLogin/signup|create an acco
},
vandalism = {
description = 'For pages protected against'
.. ' [[Wikipedia:Vandalism|vandalism]]',
text = '${INTROFRAGMENT} due to [[Wikipedia:Vandalism|vandalism]]',
explanation = function (protectionObj, args)
local ret = ''
if protectionObj.level == 'sysop' then
ret = ret .. "This protection is ''not''
.. ' ${CURRENTVERSION}.'
end
return ret .. '${EXPLANATIONBLURB}'
end,
tooltip = '${TOOLTIPFRAGMENT} due to vandalism',
}
},
move = {
dispute = {
description = 'For pages protected against page moves due
.. ' disputes over the page title',
explanation = "This protection is ''not'' an endorsement
.. ' ${CURRENTVERSION}. ${EXPLANATIONBLURB}',
image = 'Move-protection-shackle.svg'
},
vandalism = {
description = 'For pages protected against'
.. ' [[Wikipedia:Vandalism#Page-move vandalism'
.. ' |page-move vandalism]]'
}
},
autoreview = {},
upload = {}
```

```
},
-----
--
--                                GENERAL DATA TABLES
--
-----
-- Protection blurbs
-----
-- This table produces the protection blurbs available with the
-- ${PROTECTIONBLURB} parameter. It is sorted by protection action and
-- protection level, and is checked by the module in the following order:
-- 1. page's protection action, page's protection level
-- 2. page's protection action, default protection level
-- 3. "edit" protection action, default protection level
--
-- It is possible to use banner parameters inside this table.
-- *required* - this table needs edit, move, autoreview and upload subtables.
protectionBlurbs = {
    edit = {
        default = 'This ${PAGETYPE} is currently [[Help:Protection|
            .. 'protected]] from editing',
        autoconfirmed = 'Editing of this ${PAGETYPE} by [[Wikipedia:User
            .. ' levels#New users|new]] or [[Wikipedia:User access le
            .. ' users|unregistered]] users is currently [[Help:Prote
        extendedconfirmed = 'This ${PAGETYPE} is currently under extended
    },
    move = {
        default = 'This ${PAGETYPE} is currently [[Help:Protection|protec
            .. ' from [[Help:Moving a page|page moves]]'
    },
    autoreview = {
        default = 'All edits made to this ${PAGETYPE} by'
            .. ' [[Wikipedia:User access levels#New users|new]] or'
            .. ' [[Wikipedia:User access levels#Unregistered users|un
            .. ' users are currently'
            .. ' [[Wikipedia:Pending changes|subject to review]]'
    },
    upload = {
        default = 'Uploading new versions of this ${PAGETYPE} is currentl
    }
},
-----
-- Explanation blurbs
-----
-- This table produces the explanation blurbs available with the
-- ${EXPLANATIONBLURB} parameter. It is sorted by protection action,
-- protection level, and whether the page is a talk page or not. If the page is
-- a talk page it will have a talk key of "talk"; otherwise it will have a talk
-- key of "subject". The table is checked in the following order:
-- 1. page's protection action, page's protection level, page's talk key
-- 2. page's protection action, page's protection level, default talk key
-- 3. page's protection action, default protection level, page's talk key
-- 4. page's protection action, default protection level, default talk key
--
-- It is possible to use banner parameters inside this table.
-- *required* - this table needs edit, move, autoreview and upload subtables.
explanationBlurbs = {
```

```
edit = {
  autoconfirmed = {
    subject = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' cannot edit this ${PAGETYPE} and you wish to
    .. ' ${EDITREQUEST}, discuss changes on the ${TALKPAGE}
    .. ' [[Wikipedia:Requests for page protection|
    .. ' #Current requests for reduction in protection
    .. ' |request unprotection]], [[Special:Userlogin|
    .. ' [[Special:UserLogin/signup|create an account
    default = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' cannot edit this ${PAGETYPE} and you wish to
    .. ' [[Wikipedia:Requests for page protection|
    .. ' #Current requests for reduction in protection
    .. ' |request unprotection]], [[Special:Userlogin|
    .. ' [[Special:UserLogin/signup|create an account
  },
  extendedconfirmed = {
    default = 'Extended confirmed protection prevents edits by
    .. ' and registered users with fewer than 30 days
    .. ' The [[Wikipedia:Protection policy#extended|
    .. ' specifies that extended confirmed protection
    .. ' disruption, if semi-protection has proven to
    .. ' Extended confirmed protection may also be applied
    .. ' [[Wikipedia:Arbitration Committee|arbitration
    .. ' Please discuss any changes on the ${TALKPAGE}
    .. ' ${EDITREQUEST} to ask for uncontroversial changes
    .. ' [[Wikipedia:Consensus|consensus]].'
  },
  default = {
    subject = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' Please discuss any changes on the ${TALKPAGE}
    .. ' may ${EDITREQUEST} to ask an
    .. ' [[Wikipedia:Administrators|administrator]] if
    .. ' is [[Help:Minor edit#When to mark an edit as
    .. ' |uncontroversial]] or supported by [[Wikipedia:
    .. ' |consensus]]. You may also [[Wikipedia:Requests
    .. ' page protection#Current requests for reduction
    .. ' |request]] that this page be unprotected.',
    default = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' You may [[Wikipedia:Requests for page
    .. ' protection#Current requests for edits to a page
    .. ' edit]] to this page, or [[Wikipedia:Requests
    .. ' page protection#Current requests for reduction
    .. ' |ask]] for it to be unprotected.'
  }
},
move = {
  default = {
    subject = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' The page may still be edited but cannot be moved
    .. ' until unprotected. Please discuss any suggestions
    .. ' ${TALKPAGE} or at [[Wikipedia:Requested moves|
    .. ' [[Wikipedia:Requests for page protection|request
    .. ' unprotected.',
    default = 'See the [[Wikipedia:Protection policy|
    .. 'protection policy]] and ${PROTECTIONLOG} for
    .. ' The page may still be edited but cannot be moved
    .. ' until unprotected. Please discuss any suggestions
    .. ' [[Wikipedia:Requested moves]]. You can also
```



```
.. ' [[Wikipedia:Requests for page protection|request unprotection]], [[Special:Userlogin|request unprotection]], [[Special:Userlogin/signup|create an account]]',
.. ' unprotected.'
}
},
autoreview = {
  default = {
    default = 'See the [[Wikipedia:Protection policy|protection policy]] and ${PROTECTIONLOG} for
    .. ' protection policy]] and ${PROTECTIONLOG} for
    .. ' Edits to this ${PAGETYPE} by new and unregistered users
    .. ' will not be visible to readers until they are approved by
    .. ' a reviewer. To avoid the need for your edits to be
    .. ' reviewed, you may
    .. ' [[Wikipedia:Requests for page protection|request unprotection]]
    .. ' #Current requests for reduction in protection level
    .. ' |request unprotection]], [[Special:Userlogin/signup|create an account]]',
    .. ' [[Special:UserLogin/signup|create an account]]',
  },
},
upload = {
  default = {
    default = 'See the [[Wikipedia:Protection policy|protection policy]] and ${PROTECTIONLOG} for
    .. ' protection policy]] and ${PROTECTIONLOG} for
    .. ' The page may still be edited but new versions cannot be
    .. ' uploaded until it is unprotected. request that a new version be
    .. ' uploaded by using [[Wikipedia:Edit requests|protected edit requests]]
    .. ' can [[Wikipedia:Requests for page protection|request unprotection]]
    .. ' that the file be unprotected.'
  },
},
},
-----
-- Protection levels
-----
-- This table provides the data for the ${PROTECTIONLEVEL} parameter, which
-- produces a short label for different protection levels. It is sorted by
-- protection action and protection level, and is checked in the following
-- order:
-- 1. page's protection action, page's protection level
-- 2. page's protection action, default protection level
-- 3. "edit" protection action, default protection level
--
-- It is possible to use banner parameters inside this table.
-- *required* - this table needs edit, move, autoreview and upload subtables.
protectionLevels = {
  edit = {
    default = 'protected',
    templateeditor = 'template-protected',
    extendedconfirmed = 'extended-protected',
    autoconfirmed = 'semi-protected',
  },
  move = {
    default = 'move-protected'
  },
  autoreview = {
  },
  upload = {
    default = 'upload-protected'
  }
},
-----
```

```
-- Images
-----

-- This table lists different padlock images for each protection action and
-- protection level. It is used if an image is not specified in any of the
-- banner data tables, and if the page does not satisfy the conditions for using
-- the ['image-filename-indef'] image. It is checked in the following order:
-- 1. page's protection action, page's protection level
-- 2. page's protection action, default protection level
images = {
    edit = {
        default = 'Full-protection-shackle.svg',
        templateeditor = 'Template-protection-shackle.svg',
        extendedconfirmed = 'Extended-protection-shackle.svg',
        autoconfirmed = 'Semi-protection-shackle.svg'
    },
    move = {
        default = 'Move-protection-shackle.svg',
    },
    autoreview = {
        default = 'Pending-protection-shackle.svg'
    },
    upload = {
        default = 'Upload-protection-shackle.svg'
    }
},

-- Pages with a reason specified in this table will show the special "indef"
-- padlock, defined in the 'image-filename-indef' message, if no expiry is set.
indefImageReasons = {
    template = true
},

-----

-- Image links
-----

-- This table provides the data for the ${IMAGELINK} parameter, which gets
-- the image link for small padlock icons based on the page's protection action
-- and protection level. It is checked in the following order:
-- 1. page's protection action, page's protection level
-- 2. page's protection action, default protection level
-- 3. "edit" protection action, default protection level
--
-- It is possible to use banner parameters inside this table.
-- *required* - this table needs edit, move, autoreview and upload subtables.
imageLinks = {
    edit = {
        default = 'Wikipedia:Protection policy#full',
        templateeditor = 'Wikipedia:Protection policy#template',
        extendedconfirmed = 'Wikipedia:Protection policy#extended',
        autoconfirmed = 'Wikipedia:Protection policy#semi'
    },
    move = {
        default = 'Wikipedia:Protection policy#move'
    },
    autoreview = {
        default = 'Wikipedia:Protection policy#pending'
    },
    upload = {
        default = 'Wikipedia:Protection policy#upload'
    }
},
```



-- Padlock indicator names

-- This table provides the "name" attribute for the <indicator> extension tag with which small padlock icons are generated. All indicator tags on a page are displayed in alphabetical order based on this attribute, and with indicator tags with duplicate names, the last tag on the page wins. The attribute is chosen based on the protection action; table keys must be a protection action name or the string "default".

```
padlockIndicatorNames = {
    autoreview = 'pp-autoreview',
    default = 'pp-default'
},
```

-- Protection categories

--[[
-- The protection categories are stored in the protectionCategories table.
-- Keys to this table are made up of the following strings:
--
-- 1. the expiry date
-- 2. the namespace
-- 3. the protection reason (e.g. "dispute" or "vandalism")
-- 4. the protection level (e.g. "sysop" or "autoconfirmed")
-- 5. the action (e.g. "edit" or "move")
--
-- When the module looks up a category in the table, first it will will check to see a key exists that corresponds to all five parameters. For example, a user page semi-protected from vandalism for two weeks would have the key "temp-user-vandalism-autoconfirmed-edit". If no match is found, the module changes the first part of the key to "all" and checks the table again. It keeps checking increasingly generic key combinations until it finds the field, or until it reaches the key "all-all-all-all-all".
--
-- The module uses a binary matrix to determine the order in which to search. This is best demonstrated by a table. In this table, the "0" values represent "all", and the "1" values represent the original data (e.g. "indef" or "file" or "vandalism").

order	expiry	namespace	reason	level	action
1	1	1	1	1	1
2	0	1	1	1	1
3	1	0	1	1	1
4	0	0	1	1	1
5	1	1	0	1	1
6	0	1	0	1	1
7	1	0	0	1	1
8	0	0	0	1	1
9	1	1	1	0	1
10	0	1	1	0	1
11	1	0	1	0	1
12	0	0	1	0	1
13	1	1	0	0	1
14	0	1	0	0	1
15	1	0	0	0	1
16	0	0	0	0	1
17	1	1	1	1	0
18	0	1	1	1	0
19	1	0	1	1	0
20	0	0	1	1	0



```

-- 21      1      1      0      1      0
-- 22      0      1      0      1      0
-- 23      1      0      0      1      0
-- 24      0      0      0      1      0
-- 25      1      1      1      0      0
-- 26      0      1      1      0      0
-- 27      1      0      1      0      0
-- 28      0      0      1      0      0
-- 29      1      1      0      0      0
-- 30      0      1      0      0      0
-- 31      1      0      0      0      0
-- 32      0      0      0      0      0
--
-- In this scheme the action has the highest priority, as it is the last
-- to change, and the expiry has the least priority, as it changes the most.
-- The priorities of the expiry, the protection level and the action are
-- fixed, but the priorities of the reason and the namespace can be swapped
-- through the use of the cfg.bannerDataNamespaceHasPriority table.
--]]

-- If the reason specified to the template is listed in this table,
-- namespace data will take priority over reason data in the protectionCategories
-- table.
reasonsWithNamespacePriority = {
    vandalism = true,
},

-- The string to use as a namespace key for the protectionCategories table for each
-- namespace number.
categoryNamespaceKeys = {
    [ 2] = 'user',
    [ 3] = 'user',
    [ 4] = 'project',
    [ 6] = 'file',
    [ 8] = 'mediawiki',
    [10] = 'template',
    [12] = 'project',
    [14] = 'category',
    [100] = 'portal',
    [828] = 'module',
},

protectionCategories = {
    ['all|all|all|all|all'] = 'Wikipedia fully protected',
    ['all|all|office|all|all'] = 'Wikipedia Office-protected',
    ['all|all|reset|all|all'] = 'Wikipedia Office-protected',
    ['all|all|dmca|all|all'] = 'Wikipedia Office-protected',
    ['all|all|mainpage|all|all'] = 'Wikipedia fully-protected',
    ['all|all|all|extendedconfirmed|all'] = 'Wikipedia extended-confirmed',
    ['all|all|ecp|extendedconfirmed|all'] = 'Wikipedia extended-confirmed',
    ['all|template|all|all|edit'] = 'Wikipedia fully protected',
    ['all|all|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['indef|all|all|autoconfirmed|edit'] = 'Wikipedia indefinitely semi-protected',
    ['all|all|blp|autoconfirmed|edit'] = 'Wikipedia indefinitely semi-protected',
    ['temp|all|blp|autoconfirmed|edit'] = 'Wikipedia temporarily semi-protected',
    ['all|all|dispute|autoconfirmed|edit'] = 'Wikipedia pages semi-protected',
    ['all|all|sock|autoconfirmed|edit'] = 'Wikipedia pages semi-protected',
    ['all|all|vandalism|autoconfirmed|edit'] = 'Wikipedia pages semi-protected',
    ['all|category|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['all|file|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['all|portal|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['all|project|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['all|talk|all|autoconfirmed|edit'] = 'Wikipedia semi-protected',
    ['all|template|all|autoconfirmed|edit'] = 'Wikipedia semi-protected'
}

```



```

    ['all|user|all|autoconfirmed|edit'] = 'Wikipedia semi-protected t
    ['all|all|all|templateeditor|edit'] = 'Wikipedia template-protect
    ['all|template|all|templateeditor|edit'] = 'Wikipedia template-protect
    ['all|template|all|templateeditor|move'] = 'Wikipedia template-protect
    ['all|all|blp|sysop|edit'] = 'Wikipedia indefinitely prote
    ['temp|all|blp|sysop|edit'] = 'Wikipedia temporarily prof
    ['all|all|dispute|sysop|edit'] = 'Wikipedia pages protected
    ['all|all|sock|sysop|edit'] = 'Wikipedia pages protected
    ['all|all|vandalism|sysop|edit'] = 'Wikipedia pages protected
    ['all|category|all|sysop|edit'] = 'Wikipedia fully protected
    ['all|file|all|sysop|edit'] = 'Wikipedia fully-protected
    ['all|project|all|sysop|edit'] = 'Wikipedia fully-protected
    ['all|talk|all|sysop|edit'] = 'Wikipedia fully-protected
    ['all|template|all|extendedconfirmed|edit'] = 'Wikipedia extended-confir
    ['all|template|all|sysop|edit'] = 'Wikipedia fully protected
    ['all|user|all|sysop|edit'] = 'Wikipedia fully protected
    ['all|module|all|all|edit'] = 'Wikipedia fully-protected
    ['all|module|all|templateeditor|edit'] = 'Wikipedia template-protect
    ['all|module|all|extendedconfirmed|edit'] = 'Wikipedia extended-confir
    ['all|module|all|autoconfirmed|edit'] = 'Wikipedia semi-protected n
    ['all|all|all|sysop|move'] = 'Wikipedia move-protected p
    ['indef|all|all|sysop|move'] = 'Wikipedia indefinitely mov
    ['all|all|dispute|sysop|move'] = 'Wikipedia pages move-prot
    ['all|all|vandalism|sysop|move'] = 'Wikipedia pages move-prot
    ['all|portal|all|sysop|move'] = 'Wikipedia move-protected p
    ['all|project|all|sysop|move'] = 'Wikipedia move-protected p
    ['all|talk|all|sysop|move'] = 'Wikipedia move-protected t
    ['all|template|all|sysop|move'] = 'Wikipedia move-protected t
    ['all|user|all|sysop|move'] = 'Wikipedia move-protected u
    ['all|all|all|autoconfirmed|autoreview'] = 'Wikipedia pending changes
    ['all|file|all|all|upload'] = 'Wikipedia upload-protected
},

```

-- Expiry category config

```

-- This table configures the expiry category behaviour for each protection
-- action.
-- * If set to true, setting that action will always categorise the page if
--   an expiry parameter is not set.
-- * If set to false, setting that action will never categorise the page.
-- * If set to nil, the module will categorise the page if:
--   1) an expiry parameter is not set, and
--   2) a reason is provided, and
--   3) the specified reason is not blacklisted in the reasonsWithoutExpiryCheck
--   table.

```

```

expiryCheckActions = {
    edit = nil,
    move = false,
    autoreview = true,
    upload = false
},

reasonsWithoutExpiryCheck = {
    blp = true,
    template = true,
},

```

-- Pagetypes



```
-- This table produces the page types available with the ${PAGETYPE} parameter.
-- Keys are namespace numbers, or the string "default" for the default value.
pagetypes = {
    [0] = 'article',
    [6] = 'file',
    [10] = 'template',
    [14] = 'category',
    [828] = 'module',
    default = 'page'
},

-----

-- Strings marking indefinite protection
-----

-- This table contains values passed to the expiry parameter that mean the page
-- is protected indefinitely.
indefStrings = {
    ['indef'] = true,
    ['indefinite'] = true,
    ['indefinitely'] = true,
    ['infinite'] = true,
},

-----

-- Group hierarchy
-----

-- This table maps each group to all groups that have a superset of the original
-- group's page editing permissions.
hierarchy = {
    sysop = {},
    reviewer = {'sysop'},
    filemover = {'sysop'},
    templateeditor = {'sysop'},
    extendedconfirmed = {'sysop'},
    autoconfirmed = {'reviewer', 'filemover', 'templateeditor', 'extendedconf'},
    user = {'autoconfirmed'},
    ['*'] = {'user'}
},

-----

-- Wrapper templates and their default arguments
-----

-- This table contains wrapper templates used with the module, and their
-- default arguments. Templates specified in this table should contain the
-- following invocation, and no other template content:
--
-- {{#invoke:Protection banner|main}}
--
-- If other content is desired, it can be added between
-- <noinclude>...</noinclude> tags.
--
-- When a user calls one of these wrapper templates, they will use the
-- default arguments automatically. However, users can override any of the
-- arguments.
wrappers = {
    ['Template:Pp'] = {},
    ['Template:Pp-extended'] = {'ecp'},
    ['Template:Pp-blp'] = {'blp'},
    -- we don't need Template:Pp-create
    ['Template:Pp-dispute'] = {'dispute'},
    ['Template:Pp-main-page'] = {'mainpage'},
}
```




```
.. ' is [[Help:Minor edit#When to mark an edit as a minor edit'
.. '|uncontroversial]] or supported by [[Wikipedia:Consensus'
.. '|consensus]].',

-----

-- Protection log display values

-----

-- These messages determine the display values for the protection log link
-- or the pending changes log link produced by the ${PROTECTIONLOG} parameter.
-- It is possible to use banner parameters in these messages.
['protection-log-display'] = 'protection log',
['pc-log-display'] = 'pending changes log',

-----

-- Current version display values

-----

-- These messages determine the display values for the page history link
-- or the move log link produced by the ${CURRENTVERSION} parameter.
-- It is possible to use banner parameters in these messages.
['current-version-move-display'] = 'current title',
['current-version-edit-display'] = 'current version',

-----

-- Talk page

-----

-- This message determines the display value of the talk page link produced
-- with the ${TALKPAGE} parameter.
-- It is possible to use banner parameters in this message.
['talk-page-link-display'] = 'talk page',

-----

-- Edit requests

-----

-- This message determines the display value of the edit request link produced
-- with the ${EDITREQUEST} parameter.
-- It is possible to use banner parameters in this message.
['edit-request-display'] = 'submit an edit request',

-----

-- Expiry date format

-----

-- This is the format for the blurb expiry date. It should be valid input for
-- the first parameter of the #time parser function.
['expiry-date-format'] = 'F j, Y "at" H:i e',

-----

-- Tracking categories

-----

-- These messages determine which tracking categories the module outputs.
['tracking-category-incorrect'] = 'Wikipedia pages with incorrect protection temp
['tracking-category-template'] = 'Wikipedia template-protected pages other than t

-----

-- Images

-----

-- These are images that are not defined by their protection action and protectio
['image-filename-indef'] = 'Full-protection-shackle.svg',
```



```
['image-filename-default'] = 'Transparent.gif',  
-----  
-- End messages  
-----  
}  
-----  
-- End configuration  
-----  
}
```

Modul:Protection banner

Vorlage:Lua This module creates protection banners and padlock icons that are placed at the top of **protected pages**.

Inhaltsverzeichnis	
1 Usage	19
1.1 From wikitext	19
1.2 From Lua	20
2 Parameters	20
3 Reasons	20
4 Errors	21
4.1 Invalid protection date	21
4.2 Invalid action	21
4.3 Reasons cannot contain the pipe character	21
4.4 Other errors	21
5 Technical details	21

Usage

Most users will not need to use this module directly. For adding protection templates to pages you can use the **Vorlage:TI** template, or you may find it more convenient to use one of the more specific protection templates in the table below.

Vorlage:Protection templates

From wikitext

```

{{#invoke:Protection banner|main
| 1      = reason
| small  = yes/no
| action = action
| date   = protection date
| user   = username
| section = talk page section name
| category = no
}}
```

The `#invoke` syntax can be used for creating protection templates more specific than **Vorlage:TI**. For example, it is possible to create a protection template which always shows a padlock icon by using the code `{{#invoke:Protection banner|main|small=yes}}`. Pages which call this template will still be able to use other arguments, like *action*. However, this only works one level deep; a page calling a template which calls another template containing the above code will not automatically be able to use parameters like *action*.

Note: You should no longer specify the expiry, as it is automatically retrieved in all cases.

From Lua

First, load the module.

```
local mProtectionBanner = require('Module:Protection banner')
```

Then you can make protection banners by using the `_main` function.

```
mProtectionBanner._main(args, cfg, titleObj)
```

args is a table of arguments to pass to the module. For possible keys and values for this table, see the [parameters section](#). The *cfg* and *titleObj* variables are intended only for testing; *cfg* specifies a customised config table to use instead of [Module:Protection banner/config](#), and *titleObj* specifies a mw.title object to use instead of the current title. *args*, *cfg* and *titleObj* are all optional.

Parameters

All parameters are optional.

- **1** - the reason that the page was protected. If set, this must be one of the values listed in the [reasons table](#).
- **small** - if set to "yes", "y", "1", or "true", a padlock icon is generated instead of a full protection banner.
- **action** - the protection action. Must be one of "edit" (for normal protection), "move" (for move-protection) or "autoreview" (for pending changes). The default value is "edit".
- **date** - the protection date. This must be valid input to the second parameter of the [#time parser function](#). This argument has an effect for reasons that use the PROTECTIONDATE parameter in their configuration. As of July 2014, those were the "office" and "reset" reasons.
- **user** - the username of the user to generate links for. As of July 2014, this only has an effect when the "usertalk" reason is specified.
- **section** - the section name of the protected page's talk page where discussion is taking place. This works for most, but not all, values of *reason*.
- **category** - categories are suppressed if this is set to "no", "n", "0", or "false".
- **catonly** - if set to "yes", "y", "1", or "true", will only return the protection categories, and not return the banner or padlock. This has no visible output.

Reasons

The following table contains the available reasons, plus the actions for which they are available.

Skriptfehler: Ein solches Modul „Protection banner/documentation“ ist nicht vorhanden.



Errors

Below is a list of some of the common errors that this module can produce, and how to fix them.

Invalid protection date

Vorlage:Error

This error is produced if you supply a **Vorlage:Para** parameter value that is not recognised as a valid date by the `#time` parser function. If in doubt, you can just use a date in the format "dd Month YYYY", e.g. "24 April 2026". To see a full range of valid inputs, see the [#time documentation](#) (only the first parameter, the *format string*, may be specified).

Invalid action

Vorlage:Error

This error is produced if you specify an invalid protection action. There are only three valid actions: `edit` (the default, for normal protection), `move` (for move-protection), and `autoreview` (for **pending changes**). This should only be possible if you are using a template that supports manually specifying the protection action, such as **Vorlage:TI**, or if you are using `#invoke` directly. If this is not the case, please leave a message on [Module talk:Protection banner](#).

Reasons cannot contain the pipe character

Vorlage:Error

This error is produced if you specify a reason using the **Vorlage:Para** parameter that includes a pipe character (`|`). Please check that you are not entering the **Vorlage:TI** template into this parameter by mistake. The pipe character is disallowed as the module uses it internally. A list of valid reasons can be seen in the [reasons section](#).

Other errors

If you see an error other than the ones above, it is likely to either be a bug in the module or mistake in the configuration. Please post a message about it at [Module talk:Protection banner](#).

Technical details

This module uses configuration data from [Module:Protection banner/config](#). Most of the module's behaviour can be configured there, making it easily portable across different wikis and different languages.

General test cases for the module can be found at [Module:Protection banner/testcases](#), and test cases specific to enwiki's config can be found at [Module:Protection banner/config/testcases](#).

Bug reports and feature requests should be made on [the module's talk page](#).



```
-- This module implements {{pp-meta}} and its daughter templates such as
-- {{pp-dispute}}, {{pp-vandalism}} and {{pp-sock}}.

-- Initialise necessary modules.
require('Module:No globals')
local makeFileLink = require('Module:File link')._main
local effectiveProtectionLevel = require('Module:Effective protection level')._main
local effectiveProtectionExpiry = require('Module:Effective protection expiry')._main
local yesno = require('Module:Yesno')

-- Lazily initialise modules and objects we don't always need.
local getArgs, makeMessageBox, lang

-- Set constants.
local CONFIG_MODULE = 'Module:Protection banner/config'

-----
-- Helper functions
-----

local function makeCategoryLink(cat, sort)
    if cat then
        return string.format(
            '[[%s:%s|%s]]',
            mw.site.namespaces[14].name,
            cat,
            sort
        )
    end
end

-- Validation function for the expiry and the protection date
local function validateDate(dateString, dateType)
    if not lang then
        lang = mw.language.getContentLanguage()
    end
    local success, result = pcall(lang.formatDate, lang, 'U', dateString)
    if success then
        result = tonumber(result)
        if result then
            return result
        end
    end
    error(string.format(
        'invalid %s: %s',
        dateType,
        tostring(dateString)
    ), 4)
end

local function makeFullUrl(page, query, display)
    return string.format(
        '[%s %s]',
        tostring(mw.uri.fullUrl(page, query)),
        display
    )
end

-- Given a directed graph formatted as node -> table of direct successors,
-- get a table of all nodes reachable from a given node (though always
-- including the given node).
local function getReachableNodes(graph, start)
    local toWalk, retval = {[start] = true}, {}
    while true do
```

```
        -- Can't use pairs() since we're adding and removing things as we
        local k = next(toWalk) -- This always gets the "first" key
        if k == nil then
            return retval
        end
        toWalk[k] = nil
        retval[k] = true
        for _,v in ipairs(graph[k]) do
            if not retval[v] then
                toWalk[v] = true
            end
        end
    end
end
end

-----
-- Protection class
-----

local Protection = {}
Protection.__index = Protection

Protection.supportedActions = {
    edit = true,
    move = true,
    autoreview = true,
    upload = true
}

Protection.bannerConfigFields = {
    'text',
    'explanation',
    'tooltip',
    'alt',
    'link',
    'image'
}

function Protection.new(args, cfg, title)
    local obj = {}
    obj._cfg = cfg
    obj.title = title or mw.title.getCurrentTitle()

    -- Set action
    if not args.action then
        obj.action = 'edit'
    elseif Protection.supportedActions[args.action] then
        obj.action = args.action
    else
        error(string.format(
            'invalid action: %s',
            tostring(args.action)
        ), 3)
    end

    -- Set level
    obj.level = args.demolevel or effectiveProtectionLevel(obj.action, obj.ti
    if not obj.level or (obj.action == 'move' and obj.level == 'autoconfirmed
        -- Users need to be autoconfirmed to move pages anyway, so treat
        -- semi-move-protected pages as unprotected.
        obj.level = '*'
    end

    -- Set expiry
```

```
local effectiveExpiry = effectiveProtectionExpiry(obj.action, obj.title)
if effectiveExpiry == 'infinity' then
    obj.expiry = 'indef'
elseif effectiveExpiry ~= 'unknown' then
    obj.expiry = validateDate(effectiveExpiry, 'expiry date')
end

-- Set reason
if args[1] then
    obj.reason = mw.ustring.lower(args[1])
    if obj.reason:find('|') then
        error('reasons cannot contain the pipe character ("|")',
    end
end

-- Set protection date
if args.date then
    obj.protectionDate = validateDate(args.date, 'protection date')
end

-- Set banner config
do
    obj.bannerConfig = {}
    local configTables = {}
    if cfg.banners[obj.action] then
        configTables[#configTables + 1] = cfg.banners[obj.action]
    end
    if cfg.defaultBanners[obj.action] then
        configTables[#configTables + 1] = cfg.defaultBanners[obj
        configTables[#configTables + 1] = cfg.defaultBanners[obj
    end
    configTables[#configTables + 1] = cfg.masterBanner
    for i, field in ipairs(Protection.bannerConfigFields) do
        for j, t in ipairs(configTables) do
            if t[field] then
                obj.bannerConfig[field] = t[field]
                break
            end
        end
    end
end
end
return setmetatable(obj, Protection)
end

function Protection:isUserScript()
    -- Whether the page is a user JavaScript or CSS page.
    local title = self.title
    return title.namespace == 2 and (
        title.contentModel == 'javascript' or title.contentModel == 'css
    )
end

function Protection:isProtected()
    return self.level ~= '*'
end

function Protection:shouldShowLock()
    -- Whether we should output a banner/padlock
    return self:isProtected() and not self:isUserScript()
end

-- Whether this page needs a protection category.
Protection.shouldHaveProtectionCategory = Protection.shouldShowLock
```



```
function Protection:isTemporary()
    return type(self.expiry) == 'number'
end

function Protection:makeProtectionCategory()
    if not self:shouldHaveProtectionCategory() then
        return ''
    end

    local cfg = self._cfg
    local title = self.title

    -- Get the expiry key fragment.
    local expiryFragment
    if self.expiry == 'indef' then
        expiryFragment = self.expiry
    elseif type(self.expiry) == 'number' then
        expiryFragment = 'temp'
    end

    -- Get the namespace key fragment.
    local namespaceFragment = cfg.categoryNamespaceKeys[title.namespace]
    if not namespaceFragment and title.namespace % 2 == 1 then
        namespaceFragment = 'talk'
    end

    -- Define the order that key fragments are tested in. This is done with a
    -- array of tables containing the value to be tested, along with its
    -- position in the cfg.protectionCategories table.
    local order = {
        {val = expiryFragment,    keypos = 1},
        {val = namespaceFragment, keypos = 2},
        {val = self.reason,       keypos = 3},
        {val = self.level,        keypos = 4},
        {val = self.action,       keypos = 5}
    }

    --[[
    -- The old protection templates used an ad-hoc protection category system
    -- with some templates prioritising namespaces in their categories, and
    -- others prioritising the protection reason. To emulate this in this mod
    -- we use the config table cfg.reasonsWithNamespacePriority to set the
    -- reasons for which namespaces have priority over protection reason.
    -- If we are dealing with one of those reasons, move the namespace table
    -- the end of the order table, i.e. give it highest priority. If not, the
    -- reason should have highest priority, so move that to the end of the table
    -- instead.
    --]]
    table.insert(order, table.remove(order, self.reason and cfg.reasonsWithNamespacePriority[self.reason]) or 1)

    --[[
    -- Define the attempt order. Inactive subtables (subtables with nil "value"
    -- fields) are moved to the end, where they will later be given the key
    -- "all". This is to cut down on the number of table lookups in
    -- cfg.protectionCategories, which grows exponentially with the number of
    -- non-nil keys. We keep track of the number of active subtables with the
    -- noActive parameter.
    --]]
    local noActive, attemptOrder
    do
        local active, inactive = {}, {}
        for i, t in ipairs(order) do
            if t.val then
                active[#active + 1] = t
            else
                inactive[#inactive + 1] = t
            end
        end
        attemptOrder = active
        noActive = #inactive
    end
end
```

```
                else
                    inactive[#inactive + 1] = t
                end
            end
            noActive = #active
            attemptOrder = active
            for i, t in ipairs(inactive) do
                attemptOrder[#attemptOrder + 1] = t
            end
        end
    end

--[[
-- Check increasingly generic key combinations until we find a match. If
-- specific category exists for the combination of key fragments we are
-- given, that match will be found first. If not, we keep trying different
-- key fragment combinations until we match using the key
-- "all-all-all-all-all".
--
-- To generate the keys, we index the key subtables using a binary matrix
-- with indexes i and j. j is only calculated up to the number of active
-- subtables. For example, if there were three active subtables, the matrix
-- would look like this, with 0 corresponding to the key fragment "all",
-- 1 corresponding to other key fragments.
--
--   j 1  2  3
--   i
--   1  1  1  1
--   2  0  1  1
--   3  1  0  1
--   4  0  0  1
--   5  1  1  0
--   6  0  1  0
--   7  1  0  0
--   8  0  0  0
--
-- Values of j higher than the number of active subtables are set
-- to the string "all".
--
-- A key for cfg.protectionCategories is constructed for each value of i.
-- The position of the value in the key is determined by the keypos field
-- each subtable.
--]]
local cats = cfg.protectionCategories
for i = 1, 2^noActive do
    local key = {}
    for j, t in ipairs(attemptOrder) do
        if j > noActive then
            key[t.keypos] = 'all'
        else
            local quotient = i / 2 ^ (j - 1)
            quotient = math.ceil(quotient)
            if quotient % 2 == 1 then
                key[t.keypos] = t.val
            else
                key[t.keypos] = 'all'
            end
        end
    end
    end
    key = table.concat(key, '|')
    local attempt = cats[key]
    if attempt then
        return makeCategoryLink(attempt, title.text)
    end
end
```

```
        return ''
    end

    function Protection:isIncorrect()
        local expiry = self.expiry
        return not self:shouldHaveProtectionCategory()
            or type(expiry) == 'number' and expiry < os.time()
    end

    function Protection:isTemplateProtectedNonTemplate()
        local action, namespace = self.action, self.title.namespace
        return self.level == 'templateeditor'
            and (
                (action ~= 'edit' and action ~= 'move')
                or (namespace ~= 10 and namespace ~= 828)
            )
    end

    function Protection:makeCategoryLinks()
        local msg = self._cfg.msg
        local ret = {self:makeProtectionCategory()}
        if self:isIncorrect() then
            ret[#ret + 1] = makeCategoryLink(
                msg['tracking-category-incorrect'],
                self.title.text
            )
        end
        if self:isTemplateProtectedNonTemplate() then
            ret[#ret + 1] = makeCategoryLink(
                msg['tracking-category-template'],
                self.title.text
            )
        end
        return table.concat(ret)
    end

end

-----
-- Blurb class
-----

local Blurb = {}
Blurb.__index = Blurb

Blurb.bannerTextFields = {
    text = true,
    explanation = true,
    tooltip = true,
    alt = true,
    link = true
}

function Blurb.new(protectionObj, args, cfg)
    return setmetatable({
        _cfg = cfg,
        _protectionObj = protectionObj,
        _args = args
    }, Blurb)
end

-- Private methods --

function Blurb:_formatDate(num)
    -- Formats a Unix timestamp into dd Month, YYYY format.
    lang = lang or mw.language.getContentLanguage()
```



```
        local success, date = pcall(
            lang.formatDate,
            lang,
            self._cfg.msg['expiry-date-format'] or 'j F Y',
            '@' .. tostring(num)
        )
        if success then
            return date
        end
    end

end

function Blurb:_getExpandedMessage(msgKey)
    return self:_substituteParameters(self._cfg.msg[msgKey])
end

function Blurb:_substituteParameters(msg)
    if not self._params then
        local parameterFuncs = {}

        parameterFuncs.CURRENTVERSION = self._makeCurrentVersionParameter
        parameterFuncs.EDITREQUEST = self._makeEditRequestParameter
        parameterFuncs.EXPIRY = self._makeExpiryParameter
        parameterFuncs.EXPLANATIONBLURB = self._makeExplanationBlurbParameter
        parameterFuncs.IMAGELINK = self._makeImageLinkParameter
        parameterFuncs.INTROBLURB = self._makeIntroBlurbParameter
        parameterFuncs.INTROFRAGMENT = self._makeIntroFragmentParameter
        parameterFuncs.PAGETYPE = self._makePagetypeParameter
        parameterFuncs.PROTECTIONBLURB = self._makeProtectionBlurbParameter
        parameterFuncs.PROTECTIONDATE = self._makeProtectionDateParameter
        parameterFuncs.PROTECTIONLEVEL = self._makeProtectionLevelParameter
        parameterFuncs.PROTECTIONLOG = self._makeProtectionLogParameter
        parameterFuncs.TALKPAGE = self._makeTalkPageParameter
        parameterFuncs.TOOLTIPBLURB = self._makeTooltipBlurbParameter
        parameterFuncs.TOOLTIPFRAGMENT = self._makeTooltipFragmentParameter
        parameterFuncs.VANDAL = self._makeVandalTemplateParameter

        self._params = setmetatable({}, {
            __index = function (t, k)
                local param
                if parameterFuncs[k] then
                    param = parameterFuncs[k](self)
                end
                param = param or ''
                t[k] = param
                return param
            end
        })
    end

    msg = msg:gsub('${(%u+)}', self._params)
    return msg
end

function Blurb:_makeCurrentVersionParameter()
    -- A link to the page history or the move log, depending on the kind of
    -- protection.
    local pagename = self._protectionObj.title.prefixedText
    if self._protectionObj.action == 'move' then
        -- We need the move log link.
        return makeFullUrl(
            'Special:Log',
            {type = 'move', page = pagename},
            self:_getExpandedMessage('current-version-move-display')
        )
    end
end
```

```
        else
            -- We need the history link.
            return makeFullUrl(
                pagename,
                {action = 'history'},
                self:_getExpandedMessage('current-version-edit-display')
            )
        end
    end
end

function Blurb:_makeEditRequestParameter()
    local mEditRequest = require('Module:Submit an edit request')
    local action = self._protectionObj.action
    local level = self._protectionObj.level

    -- Get the edit request type.
    local requestType
    if action == 'edit' then
        if level == 'autoconfirmed' then
            requestType = 'semi'
        elseif level == 'extendedconfirmed' then
            requestType = 'extended'
        elseif level == 'templateeditor' then
            requestType = 'template'
        end
    end
    requestType = requestType or 'full'

    -- Get the display value.
    local display = self:_getExpandedMessage('edit-request-display')

    return mEditRequest._link{type = requestType, display = display}
end

function Blurb:_makeExpiryParameter()
    local expiry = self._protectionObj.expiry
    if type(expiry) == 'number' then
        return self:_formatDate(expiry)
    else
        return expiry
    end
end

function Blurb:_makeExplanationBlurbParameter()
    -- Cover special cases first.
    if self._protectionObj.title.namespace == 8 then
        -- MediaWiki namespace
        return self:_getExpandedMessage('explanation-blurb-nounprotect')
    end

    -- Get explanation blurb table keys
    local action = self._protectionObj.action
    local level = self._protectionObj.level
    local talkKey = self._protectionObj.title.isTalkPage and 'talk' or 'subject'

    -- Find the message in the explanation blurb table and substitute any
    -- parameters.
    local explanations = self._cfg.explanationBlurbs
    local msg
    if explanations[action][level] and explanations[action][level][talkKey] then
        msg = explanations[action][level][talkKey]
    elseif explanations[action][level] and explanations[action][level].default then
        msg = explanations[action][level].default
    elseif explanations[action].default and explanations[action].default[talkKey] then
        msg = explanations[action].default[talkKey]
    end
end
```

```
        msg = explanations[action].default[talkKey]
    elseif explanations[action].default and explanations[action].default.defa
        msg = explanations[action].default.default
    else
        error(string.format(
            'could not find explanation blurb for action "%s", level
            action,
            level,
            talkKey
        ), 8)
    end
    return self:_substituteParameters(msg)
end

function Blurb:_makeImageLinkParameter()
    local imageLinks = self._cfg.imageLinks
    local action = self._protectionObj.action
    local level = self._protectionObj.level
    local msg
    if imageLinks[action][level] then
        msg = imageLinks[action][level]
    elseif imageLinks[action].default then
        msg = imageLinks[action].default
    else
        msg = imageLinks.edit.default
    end
    return self:_substituteParameters(msg)
end

function Blurb:_makeIntroBlurbParameter()
    if self._protectionObj.isTemporary() then
        return self:_getExpandedMessage('intro-blurb-expiry')
    else
        return self:_getExpandedMessage('intro-blurb-noexpiry')
    end
end

function Blurb:_makeIntroFragmentParameter()
    if self._protectionObj.isTemporary() then
        return self:_getExpandedMessage('intro-fragment-expiry')
    else
        return self:_getExpandedMessage('intro-fragment-noexpiry')
    end
end

function Blurb:_makePagetypeParameter()
    local pagetypes = self._cfg.pagetypes
    return pagetypes[self._protectionObj.title.namespace]
        or pagetypes.default
        or error('no default pagetype defined', 8)
end

function Blurb:_makeProtectionBlurbParameter()
    local protectionBlurbs = self._cfg.protectionBlurbs
    local action = self._protectionObj.action
    local level = self._protectionObj.level
    local msg
    if protectionBlurbs[action][level] then
        msg = protectionBlurbs[action][level]
    elseif protectionBlurbs[action].default then
        msg = protectionBlurbs[action].default
    elseif protectionBlurbs.edit.default then
        msg = protectionBlurbs.edit.default
    else
```

```
        error('no protection blurb defined for protectionBlurbs.edit.default')
    end
    return self:_substituteParameters(msg)
end

function Blurb:_makeProtectionDateParameter()
    local protectionDate = self._protectionObj.protectionDate
    if type(protectionDate) == 'number' then
        return self:_formatDate(protectionDate)
    else
        return protectionDate
    end
end

function Blurb:_makeProtectionLevelParameter()
    local protectionLevels = self._cfg.protectionLevels
    local action = self._protectionObj.action
    local level = self._protectionObj.level
    local msg
    if protectionLevels[action][level] then
        msg = protectionLevels[action][level]
    elseif protectionLevels[action].default then
        msg = protectionLevels[action].default
    elseif protectionLevels.edit.default then
        msg = protectionLevels.edit.default
    else
        error('no protection level defined for protectionLevels.edit.default')
    end
    return self:_substituteParameters(msg)
end

function Blurb:_makeProtectionLogParameter()
    local pagename = self._protectionObj.title.prefixedText
    if self._protectionObj.action == 'autoreview' then
        -- We need the pending changes log.
        return makeFullUrl(
            'Special:Log',
            {type = 'stable', page = pagename},
            self:_getExpandedMessage('pc-log-display')
        )
    else
        -- We need the protection log.
        return makeFullUrl(
            'Special:Log',
            {type = 'protect', page = pagename},
            self:_getExpandedMessage('protection-log-display')
        )
    end
end

function Blurb:_makeTalkPageParameter()
    return string.format(
        '[[[:s:#s|s]]',
        mw.site.namespaces[self._protectionObj.title.namespace].talk.name,
        self._protectionObj.title.text,
        self._args.section or 'top',
        self:_getExpandedMessage('talk-page-link-display')
    )
end

function Blurb:_makeTooltipBlurbParameter()
    if self._protectionObj.isTemporary() then
        return self:_getExpandedMessage('tooltip-blurb-expiry')
    else

```

```
        return self:_getExpandedMessage('tooltip-blurb-noexpiry')
    end
end

function Blurb:_makeTooltipFragmentParameter()
    if self._protectionObj.isTemporary() then
        return self:_getExpandedMessage('tooltip-fragment-expiry')
    else
        return self:_getExpandedMessage('tooltip-fragment-noexpiry')
    end
end

function Blurb:_makeVandalTemplateParameter()
    return mw.getCurrentFrame():expandTemplate{
        title="vandal-m",
        args={self._args.user or self._protectionObj.title.baseText}
    }
end

-- Public methods --

function Blurb:makeBannerText(key)
    -- Validate input.
    if not key or not Blurb.bannerTextFields[key] then
        error(string.format(
            '"%s" is not a valid banner config field',
            tostring(key)
        ), 2)
    end

    -- Generate the text.
    local msg = self._protectionObj.bannerConfig[key]
    if type(msg) == 'string' then
        return self:_substituteParameters(msg)
    elseif type(msg) == 'function' then
        msg = msg(self._protectionObj, self._args)
        if type(msg) ~= 'string' then
            error(string.format(
                'bad output from banner config function with key
                .. ' (expected string, got %s)',
                tostring(key),
                type(msg)
            ), 4)
        end
        return self:_substituteParameters(msg)
    end
end

-----
-- BannerTemplate class
-----

local BannerTemplate = {}
BannerTemplate.__index = BannerTemplate

function BannerTemplate.new(protectionObj, cfg)
    local obj = {}
    obj._cfg = cfg

    -- Set the image filename.
    local imageFilename = protectionObj.bannerConfig.image
    if imageFilename then
        obj._imageFilename = imageFilename
    else

```

```
-- If an image filename isn't specified explicitly in the banner
-- generate it from the protection status and the namespace.
local action = protectionObj.action
local level = protectionObj.level
local namespace = protectionObj.title.namespace
local reason = protectionObj.reason

-- Deal with special cases first.
if (
    namespace == 10
    or namespace == 828
    or reason and obj._cfg.indefImageReasons[reason]
)
    and action == 'edit'
    and level == 'sysop'
    and not protectionObj:isTemporary()
then
    -- Fully protected modules and templates get the special
    -- padlock.
    obj._imageFilename = obj._cfg.msg['image-filename-indef']
else
    -- Deal with regular protection types.
    local images = obj._cfg.images
    if images[action] then
        if images[action][level] then
            obj._imageFilename = images[action][level]
        elseif images[action].default then
            obj._imageFilename = images[action].default
        end
    end
end
end
return setmetatable(obj, BannerTemplate)
end

function BannerTemplate:renderImage()
    local filename = self._imageFilename
        or self._cfg.msg['image-filename-default']
        or 'Transparent.gif'
    return makeFileLink{
        file = filename,
        size = (self.imageWidth or 20) .. 'px',
        alt = self._imageAlt,
        link = self._imageLink,
        caption = self.imageCaption
    }
end

-----
-- Banner class
-----

local Banner = setmetatable({}, BannerTemplate)
Banner.__index = Banner

function Banner.new(protectionObj, blurbObj, cfg)
    local obj = BannerTemplate.new(protectionObj, cfg) -- This doesn't need t
    obj.imageWidth = 40
    obj.imageCaption = blurbObj:makeBannerText('alt') -- Large banners use th
    obj._reasonText = blurbObj:makeBannerText('text')
    obj._explanationText = blurbObj:makeBannerText('explanation')
    obj._page = protectionObj.title.prefixedText -- Only makes a difference i
    return setmetatable(obj, Banner)
end
```

```
function Banner:__tostring()
    -- Renders the banner.
    makeMessageBox = makeMessageBox or require('Module:Message box').main
    local reasonText = self._reasonText or error('no reason text set', 2)
    local explanationText = self._explanationText
    local mbargs = {
        page = self._page,
        type = 'protection',
        image = self:renderImage(),
        text = string.format(
            "''''%s''''%s",
            reasonText,
            explanationText and '<br />' .. explanationText or ''
        )
    }
    return makeMessageBox('mbox', mbargs)
end

-----
-- Padlock class
-----

local Padlock = setmetatable({}, BannerTemplate)
Padlock.__index = Padlock

function Padlock.new(protectionObj, blurbObj, cfg)
    local obj = BannerTemplate.new(protectionObj, cfg) -- This doesn't need
    obj.imageWidth = 20
    obj.imageCaption = blurbObj:makeBannerText('tooltip')
    obj._imageAlt = blurbObj:makeBannerText('alt')
    obj._imageLink = blurbObj:makeBannerText('link')
    obj._indicatorName = cfg.padlockIndicatorNames[protectionObj.action]
        or cfg.padlockIndicatorNames.default
        or 'pp-default'
    return setmetatable(obj, Padlock)
end

function Padlock:__tostring()
    local frame = mw.getCurrentFrame()
    -- The nowiki tag helps prevent whitespace at the top of articles.
    return frame:extensionTag{name = 'nowiki'} .. frame:extensionTag{
        name = 'indicator',
        args = {name = self._indicatorName},
        content = self:renderImage()
    }
end

-----
-- Exports
-----

local p = {}

function p._exportClasses()
    -- This is used for testing purposes.
    return {
        Protection = Protection,
        Blurb = Blurb,
        BannerTemplate = BannerTemplate,
        Banner = Banner,
        Padlock = Padlock,
    }
end
```

```
function p._main(args, cfg, title)
  args = args or {}
  cfg = cfg or require(CONFIG_MODULE)

  local protectionObj = Protection.new(args, cfg, title)

  local ret = {}

  -- If a page's edit protection is equally or more restrictive than its
  -- protection from some other action, then don't bother displaying anything
  -- for the other action (except categories).
  if not yesno(args.catonly) and (protectionObj.action == 'edit' or
    args.demolevel or
    not getReachableNodes(
      cfg.hierarchy,
      protectionObj.level
    )[effectiveProtectionLevel('edit', protectionObj.title)])
  then
    -- Initialise the blurb object
    local blurbObj = Blurb.new(protectionObj, args, cfg)

    -- Render the banner
    if protectionObj:shouldShowLock() then
      ret[#ret + 1] = tostring(
        (yesno(args.small) and Padlock or Banner)
        .new(protectionObj, blurbObj, cfg)
      )
    end
  end

  -- Render the categories
  if yesno(args.category) ~= false then
    ret[#ret + 1] = protectionObj:makeCategoryLinks()
  end

  return table.concat(ret)
end

function p.main(frame, cfg)
  cfg = cfg or require(CONFIG_MODULE)

  -- Find default args, if any.
  local parent = frame.getParent and frame.getParent()
  local defaultArgs = parent and cfg.wrappers[parent:getTitle():gsub('/', sand

  -- Find user args, and use the parent frame if we are being called from a
  -- wrapper template.
  getArgs = getArgs or require('Module:Arguments').getArgs
  local userArgs = getArgs(frame, {
    parentOnly = defaultArgs,
    frameOnly = not defaultArgs
  })

  -- Build the args table. User-specified args overwrite default args.
  local args = {}
  for k, v in pairs(defaultArgs or {}) do
    args[k] = v
  end
  for k, v in pairs(userArgs) do
```



```
        args[k] = v
    end
    return p._main(args, cfg)
end
return p
```