Module:Citation/CS1

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

<section begin=header />

This Lua module is used on [approximately 4,330,000 pages, or roughly 242985% of all pages](#).

40px To avoid major disruption and server load, any changes should be tested in the module's /sandbox or /testcases subpages, or in your own module sandbox. The tested changes can be added to this page in a single edit. Consider discussing changes on the talk page before implementing them.

32px This module uses TemplateStyles:
• Module:Citation/CS1/styles.css

40x40px This module is subject to page protection. It is a highly visible module in use by a very large number of pages, or is substituted very frequently. Because vandalism or mistakes would affect many pages, and even trivial editing might cause substantial load on the servers, it is protected from editing.

<section end=header />

This module and associated sub-modules support the Citation Style 1 and Citation Style 2 citation templates. In general, it is not intended to be called directly, but is called by one of the core CS1 and CS2 templates. <section begin=module_components_table /> These files comprise the module support for cs1|2 citation templates:

cs1 | cs2 modules

live	sandbox	description

	Module:Citation/CS1	Module:Citation/CS1/sandbox	[edit]	Rendering and support functions
	Module:Citation/CS1/Configuration	Module:Citation/CS1/Configuration/sandbox	[edit]	Translation tables; error and identifier handlers
	Module:Citation/CS1/Whitelist	Module:Citation/CS1/Whitelist/sandbox	[edit]	List of active, deprecated, and obsolete cs1|2 parameters
sysop	Module:Citation/CS1/Date validation	Module:Citation/CS1/Date validation/sandbox	[edit]	Date format validation functions
	Module:Citation/CS1/Identifiers	Module:Citation/CS1/Identifiers/sandbox	[edit]	Functions that support the named identifiers (isbn, doi, pmid, etc)
	Module:Citation/CS1/Utilities	Module:Citation/CS1/Utilities/sandbox	[edit]	Common functions and tables
	Module:Citation/CS1/COinS	Module:Citation/CS1/COinS/sandbox	[edit]	Functions that render a cs1|2 template's metadata
	Module:Citation/CS1/styles.css	Module:Citation/CS1/sandbox/styles.css	[edit]	CSS styles applied to the cs1|2 templates
auto confirmed	Module:Citation/CS1/Suggestions	Module:Citation/CS1/Suggestions/sandbox	[edit]	List that maps common erroneous parameter names to valid parameter names

<section end=module_components_table />

Other documentation:

- Module talk:Citation/CS1/Feature requests
- Module talk:Citation/CS1/COinS

```
--[[-------------------------< F O R W A R D   D E C L A R A T I O N S >----
```

```
---------------------------------
]]

local dates, year_date_check, reformat_dates, date_hyphen_to_dash,
-- functions in Module:Citation/CS1/Date_validation
        date_name_xlate

local is_set, in_array, substitute, error_comment, set_error, select_one,
-- functions in Module:Citation/CS1/Utilities
        add_maint_cat, wrap_style, safe_for_italics, is_wikilink,
make_wikilink,
        strip_apostrophe_markup;

local z ={};
-- tables in Module:Citation/CS1/Utilities

local extract_ids, extract_id_access_levels, build_id_list, is_embargoed;
-- functions in Module:Citation/CS1/Identifiers

local make_coins_title, get_coins_pages, COinS;
-- functions in Module:Citation/CS1/COinS

local cfg = {};
-- table of configuration tables that are defined in
Module:Citation/CS1/Configuration
local whitelist = {};
-- table of tables listing valid template parameter names; defined in
Module:Citation/CS1/Whitelist


--[[--------------------------< P A G E   S C O P E   V A R I A B L E S >----
---------------------------------

declare variables here that have page-wide scope that are not brought in from
other modules; that are created here and used here

]]

local added_deprecated_cat;
-- boolean flag so that the category is added only once
local added_prop_cats = {};
-- list of property categories that have been added to z.properties_cats
local added_vanc_errs;
-- boolean flag so we only emit one Vancouver error / category

local Frame;
-- holds the module's frame table


--[[--------------------------< F I R S T _ S E T >------------------------
---------------------------------
```

```
Locates and returns the first set value in a table of values where the order
established in the table,
left-to-right (or top-to-bottom), is the order in which the values are
evaluated.  Returns nil if none are set.

This version replaces the original 'for _, val in pairs do' and a similar
version that used ipairs.  With the pairs
version the order of evaluation could not be guaranteed.  With the ipairs
version, a nil value would terminate
the for-loop before it reached the actual end of the list.

]]

local function first_set (list, count)
        local i = 1;
        while i <= count do
-- loop through all items in list
                if is_set( list[i] ) then
                        return list[i];
-- return the first set list member
                end
                i = i + 1;
-- point to next
        end
end


--[[--------------------------< A D D _ P R O P _ C A T >--------------------
-----------------------------------

Adds a category to z.properties_cats using names from the configuration file
with additional text if any.

foreign_lang_source and foreign_lang_source_2 keys have a language code
appended to them so that multiple languages
may be categorized but multiples of the same language are not categorized.

added_prop_cats is a table declared in page scope variables above

]]

local function add_prop_cat (key, arguments)
        if not added_prop_cats [key] then
                added_prop_cats [key] = true;
-- note that we've added this category
                key = key:gsub ('(foreign_lang_source_?2?)%a%a%a?[%a%-]*',
'%1');                -- strip lang code from keyname
                table.insert( z.properties_cats, substitute (cfg.prop_cats
[key], arguments));        -- make name then add to table
        end
end
```

```
--[[--------------------------< A D D _ V A N C _ E R R O R >--------------
-----------------------------------

Adds a single Vancouver system error message to the template's output
regardless of how many error actually exist.
To prevent duplication, added_vanc_errs is nil until an error message is
emitted.

added_vanc_errs is a boolean declared in page scope variables above

]]

local function add_vanc_error (source)
	if not added_vanc_errs then
		added_vanc_errs = true;
-- note that we've added this category
		table.insert( z.message_tail, { set_error( 'vancouver',
{source}, true ) } );
	end
end


--[[--------------------------< I S _ S C H E M E >------------------------
-----------------------------------

does this thing that purports to be a uri scheme seem to be a valid scheme?
The scheme is checked to see if it
is in agreement with http://tools.ietf.org/html/std66#section-3.1 which says:
	Scheme names consist of a sequence of characters beginning with a
   letter and followed by any combination of letters, digits, plus
   ("+"), period ("."), or hyphen ("-").

returns true if it does, else false

]]

local function is_scheme (scheme)
	return scheme and scheme:match ('^%a[%a%d%+%.%-]*:');
-- true if scheme is set and matches the pattern
end


--[=[-------------------------< I S _ D O M A I N _ N A M E >--------------
-----------------------------------

Does this thing that purports to be a domain name seem to be a valid domain
name?

Syntax defined here: http://tools.ietf.org/html/rfc1034#section-3.5
BNF defined here: https://tools.ietf.org/html/rfc4234
Single character names are generally reserved; see
```

```
https://tools.ietf.org/html/draft-ietf-dnsind-iana-dns-01#page-15;
        see also [[Single-letter second-level domain]]
list of tlds: https://www.iana.org/domains/root/db

rfc952 (modified by rfc 1123) requires the first and last character of a
hostname to be a letter or a digit.  Between
the first and last characters the name may use letters, digits, and the
hyphen.

Also allowed are IPv4 addresses. IPv6 not supported

domain is expected to be stripped of any path so that the last character in
the last character of the tld.  tld
is two or more alpha characters.  Any preceding '//' (from splitting a url
with a scheme) will be stripped
here.  Perhaps not necessary but retained incase it is necessary for IPv4 dot
decimal.

There are several tests:
        the first character of the whole domain name including subdomains
must be a letter or a digit
        internationalized domain name (ascii characters with .xn-- ASCII
Compatible Encoding (ACE) prefix xn-- in the tld) see
https://tools.ietf.org/html/rfc3490
        single-letter/digit second-level domains in the .org, .cash, and
.today TLDs
        q, x, and z SL domains in the .com TLD
        i and q SL domains in the .net TLD
        single-letter SL domains in the ccTLDs (where the ccTLD is two
letters)
        two-character SL domains in gTLDs (where the gTLD is two or more
letters)
        three-plus-character SL domains in gTLDs (where the gTLD is two or
more letters)
        IPv4 dot-decimal address format; TLD not allowed

returns true if domain appears to be a proper name and tld or IPv4 address,
else false

]=]

local function is_domain_name (domain)
        if not domain then
                return false;
-- if not set, abandon
        end
        domain = domain:gsub ('^//', '');
-- strip '//' from domain name if present; done here so we only have to do it
once
        if not domain:match ('^[%w]') then
-- first character must be letter or digit
```

```
                return false;
        end

        if domain:match ('^%a+:') then
-- hack to detect things that look like s:Page:Title where Page: is namespace
at wikisource
                return false;
        end

        local patterns = {
-- patterns that look like urls
                '%f[%w][%w][%w%-]+[%w]%.%a%a+$',
-- three or more character hostname.hostname or hostname.tld
                '%f[%w][%w][%w%-]+[%w]%.xn%-%-[%w]+$',
-- internationalized domain name with ACE prefix
                '%f[%a][qxz]%.com$',
-- assigned one character .com hostname (x.com times out 2015-12-10)
                '%f[%a][iq]%.net$',
-- assigned one character .net hostname (q.net registered but not active
2015-12-10)
                '%f[%w][%w]%.%a%a$',
-- one character hostname and cctld (2 chars)
                '%f[%w][%w][%w]%.%a%a+$',
-- two character hostname and tld
                '^%d%d?%d?%.%d%d?%d?%.%d%d?%d?%.%d%d?%d?',
-- IPv4 address
                }

        for _, pattern in ipairs (patterns) do
-- loop through the patterns list
                if domain:match (pattern) then
                        return true;
-- if a match then we think that this thing that purports to be a url is a
url
                end
        end

        for _, d in ipairs ({'cash', 'company', 'today', 'org'}) do
-- look for single letter second level domain names for these top level
domains
                if domain:match ('%f[%w][%w]%.' .. d) then
                        return true
                end
        end
        return false;
-- no matches, we don't know what this thing is
end


--[[-------------------------< I S _ U R L >-------------------------------
----------------------------------
```

```
returns true if the scheme and domain parts of a url appear to be a valid
url; else false.

This function is the last step in the validation process.  This function is
separate because there are cases that
are not covered by split_url(), for example is_parameter_ext_wikilink() which
is looking for bracketted external
wikilinks.

]]

local function is_url (scheme, domain)
	if is_set (scheme) then
-- if scheme is set check it and domain
		return is_scheme (scheme) and is_domain_name (domain);
	else
		return is_domain_name (domain);
-- scheme not set when url is protocol relative
	end
end


--[[--------------------------< S P L I T _ U R L >--------------------------
----------------------------------

Split a url into a scheme, authority indicator, and domain.

First remove Fully Qualified Domain Name terminator (a dot following tld) (if
any) and any path(/), query(?) or fragment(#).

If protocol relative url, return nil scheme and domain else return nil for
both scheme and domain.

When not protocol relative, get scheme, authority indicator, and domain.  If
there is an authority indicator (one
or more '/' characters immediately following the scheme's colon), make sure
that there are only 2.

Strip off any port and path;

]]

local function split_url (url_str)
	local scheme, authority, domain;
	url_str = url_str:gsub ('([%a%d])%.?[/%?#].*$', '%1');
-- strip FQDN terminator and path(/), query(?), fragment (#) (the capture
prevents false replacement of '//')

	if url_str:match ('^//%S*') then
-- if there is what appears to be a protocol relative url
		domain = url_str:match ('^//(%S*)')
```

```
        elseif url_str:match ('%S-:/*%S+') then
-- if there is what appears to be a scheme, optional authority indicator, and
domain name
                scheme, authority, domain = url_str:match ('(%S-
:)(/*)(%S+)');                          -- extract the scheme, authority
indicator, and domain portions
                authority = authority:gsub ('//', '', 1);
-- replace place 1 pair of '/' with nothing;
                if is_set(authority) then
-- if anything left (1 or 3+ '/' where authority should be) then
                        return scheme;
-- return scheme only making domain nil which will cause an error message
                end
                domain = domain:gsub ('(%a):%d+', '%1');
-- strip port number if present
        end
        return scheme, domain;
end


--[[--------------------------< L I N K _ P A R A M _ O K >------------------
---------------------------------

checks the content of |title-link=, |series-link=, |author-link= etc for
properly formatted content: no wikilinks, no urls

Link parameters are to hold the title of a wikipedia article so none of the
WP:TITLESPECIALCHARACTERS are allowed:
        # < > [ ] | { } _
except the underscore which is used as a space in wiki urls and # which is
used for section links

returns false when the value contains any of these characters.

When there are no illegal characters, this function returns TRUE if value
DOES NOT appear to be a valid url (the
|<param>-link= parameter is ok); else false when value appears to be a valid
url (the |<param>-link= parameter is NOT ok).

]]

local function link_param_ok (value)
        local scheme, domain;
        if value:find ('[<>%[%]|{}]') then
-- if any prohibited characters
                return false;
        end

        scheme, domain = split_url (value);
-- get scheme or nil and domain or nil from url;
        return not is_url (scheme, domain);
```

```
-- return true if value DOES NOT appear to be a valid url
end


--[[--------------------------< L I N K _ T I T L E _ O K >----------------
--------------------------------

Use link_param_ok() to validate |<param>-link= value and its matching
|<title>= value.

|<title>= may be wikilinked but not when |<param>-link= has a value.  This
function emits an error message when
that condition exists

]]

local function link_title_ok (link, lorig, title, torig)
local orig;

        if is_set (link) then
-- don't bother if <param>-link doesn't have a value
                if not link_param_ok (link) then
-- check |<param>-link= markup
                        orig = lorig;
-- identify the failing link parameter
                elseif title:find ('%[%[') then
-- check |title= for wikilink markup
                        orig = torig;
-- identify the failing |title= parameter
                end
        end

        if is_set (orig) then
                table.insert( z.message_tail, { set_error( 'bad_paramlink',
orig)});        -- url or wikilink in |title= with |title-link=;
        end
end


--[[--------------------------< C H E C K _ U R L >------------------------
--------------------------------

Determines whether a URL string appears to be valid.

First we test for space characters.  If any are found, return false.  Then
split the url into scheme and domain
portions, or for protocol relative (//example.com) urls, just the domain.
Use is_url() to validate the two
portions of the url.  If both are valid, or for protocol relative if domain
is valid, return true, else false.
```

```
Because it is different from a standard url, and because this module used
external_link() to make external links
that work for standard and news: links, we validate newsgroup names here.
The specification for a newsgroup name
is at https://tools.ietf.org/html/rfc5536#section-3.1.4

]]

local function check_url( url_str )
        if nil == url_str:match ("^%S+$") then
-- if there are any spaces in |url=value it can't be a proper url
                return false;
        end
        local scheme, domain;

        scheme, domain = split_url (url_str);
-- get scheme or nil and domain or nil from url;
        if 'news:' == scheme then
-- special case for newsgroups
                return domain:match('^[%a%d%+%-_]+%.[%a%d%+%-_%.]*[%a%d%+%-
_]$');
        end
        return is_url (scheme, domain);
-- return true if value appears to be a valid url
end


--[=[-------------------------< I S _ P A R A M E T E R _ E X T _ W I K I L I
N K >----------------------------

Return true if a parameter value has a string that begins and ends with
square brackets [ and ] and the first
non-space characters following the opening bracket appear to be a url.  The
test will also find external wikilinks
that use protocol relative urls. Also finds bare urls.

The frontier pattern prevents a match on interwiki links which are similar to
scheme:path urls.  The tests that
find bracketed urls are required because the parameters that call this test
(currently |title=, |chapter=, |work=,
and |publisher=) may have wikilinks and there are articles or redirects like
'//Hus' so, while uncommon, |title=[[//Hus]]
is possible as might be [[en://Hus]].

]=]

local function is_parameter_ext_wikilink (value)
local scheme, domain;

        if value:match ('%f[%[]%[%a%S*:%S+.*%]') then
-- if ext wikilink with scheme and domain: [xxxx://yyyyy.zzz]
```

```
                scheme, domain = split_url (value:match
('%f[%[]%[(%a%S*:%S+).*%]'));
        elseif value:match ('%f[%[]%[//%S+.*%]') then
-- if protocol relative ext wikilink: [//yyyyy.zzz]
                scheme, domain = split_url (value:match
('%f[%[]%[(//%S+).*%]'));
        elseif value:match ('%a%S*:%S+') then
-- if bare url with scheme; may have leading or trailing plain text
                scheme, domain = split_url (value:match ('(%a%S*:%S+)'));
        elseif value:match ('//%S+') then
-- if protocol relative bare url: //yyyyy.zzz; may have leading or trailing
plain text
                scheme, domain = split_url (value:match ('(//%S+)'));
-- what is left should be the domain
        else
                return false;
-- didn't find anything that is obviously a url
        end

        return is_url (scheme, domain);
-- return true if value appears to be a valid url
end


--[[-------------------------< C H E C K _ F O R _ U R L >-------------------
----------------------------------

loop through a list of parameters and their values.  Look at the value and if
it has an external link, emit an error message.

]]

local function check_for_url (parameter_list)
local error_message = '';
        for k, v in pairs (parameter_list) do
-- for each parameter in the list
                if is_parameter_ext_wikilink (v) then
-- look at the value; if there is a url add an error message
                        if is_set(error_message) then
-- once we've added the first portion of the error message ...
                                error_message=error_message .. ", ";
-- ... add a comma space separator
                        end
                        error_message=error_message .. "&#124;" .. k .. "=";
-- add the failed parameter
                end
        end
        if is_set (error_message) then
-- done looping, if there is an error message, display it
                table.insert( z.message_tail, { set_error(
'param_has_ext_link', {error_message}, true ) } );
```

```
        end
end


--[[--------------------------< S A F E _ F O R _ U R L >-------------------
---------------------------------

Escape sequences for content that will be used for URL descriptions

]]

local function safe_for_url( str )
        if str:match( "%[%[.-%]%]" ) ~= nil then
                table.insert( z.message_tail, { set_error( 'wikilink_in_url',
{}, true ) } );
        end
        return str:gsub( '[%[%]\n]', {
                ['['] = '&#91;',
                [']'] = '&#93;',
                ['\n'] = ' ' } );
end


--[[--------------------------< E X T E R N A L _ L I N K >-----------------
---------------------------------

Format an external link with error checking

]]

local function external_link( URL, label, source, access)
        local error_str = "";
        local domain;
        local path;
        local base_url;

        if not is_set( label ) then
                label = URL;
                if is_set( source ) then
                        error_str = set_error( 'bare_url_missing_title', {
wrap_style ('parameter', source) }, false, " " );
                else
                        error( cfg.messages["bare_url_no_origin"] );
                end
        end
        if not check_url( URL ) then
                error_str = set_error( 'bad_url', {wrap_style ('parameter',
source)}, false, " " ) .. error_str;
        end
        domain, path = URL:match ('^([/%.%-%+:%a%d]+)([/%?#].*)$');
-- split the url into scheme plus domain and path
```

```
        if path then
-- if there is a path portion
                path = path:gsub ('[%[%]]', {['[']='%5b',[']']='%5d'});
-- replace '[' and ']' with their percent encoded values
                URL = table.concat ({domain, path});
-- and reassemble
        end

        base_url = table.concat({ "[", URL, " ", safe_for_url (label), "]"
});                -- assemble a wikimarkup url

        if is_set (access) then
-- access level (subscription, registration, limited)
                base_url = substitute (cfg.presentation['ext-link-access-
signal'], {cfg.presentation[access].class, cfg.presentation[access].title,
base_url});        -- add the appropriate icon
        end
        return table.concat ({base_url, error_str});
end


--[[--------------------------< D E P R E C A T E D _ P A R A M E T E R >----
-----------------------------------

Categorize and emit an error message when the citation contains one or more
deprecated parameters.  The function includes the
offending parameter name to the error message.  Only one error message is
emitted regardless of the number of deprecated
parameters in the citation.

added_deprecated_cat is a boolean declared in page scope variables above

]]

local function deprecated_parameter(name)
        if not added_deprecated_cat then
                added_deprecated_cat = true;
-- note that we've added this category
                table.insert( z.message_tail, { set_error(
'deprecated_params', {name}, true ) } );        -- add error message
        end
end


--[=[-------------------------< K E R N _ Q U O T E S >----------------------
-----------------------------------

Apply kerning to open the space between the quote mark provided by the Module
and a leading or trailing quote
mark contained in a |title= or |chapter= parameter's value.
```

```
This function will positive kern either single or double quotes:
        "'Unkerned title with leading and trailing single quote marks'"
        " 'Kerned title with leading and trailing single quote marks' " (in
real life the kerning isn't as wide as this example)
Double single quotes (italic or bold wikimarkup) are not kerned.

Replaces unicode quotemarks in plain text or in the label portion of a
[[L|D]] style wikilink with typewriter
quote marks regardless of the need for kerning.  Unicode quote marks are not
replaced in simple [[D]] wikilinks.

Call this function for chapter titles, for website titles, etc; not for book
titles.

]=]

local function kern_quotes (str)
        local cap='';
        local cap2='';
        local wl_type, label, link;

        wl_type, label, link = is_wikilink (str);
-- wl_type is: 0, no wl (text in label variable); 1, [[D]]; 2, [[L|D]]
        if 1 == wl_type then
-- [[D]] simple wikilink with or without quote marks
                if mw.ustring.match (str, '%[%[[\"“”\'‘’].+[\"“”\'‘’]%]%]')
then                -- leading and trailing quote marks
                        str = substitute (cfg.presentation['kern-wl-both'],
str);
                elseif mw.ustring.match (str, '%[%[[\"“”\'‘’].+%]%]')
then                        -- leading quote marks
                        str = substitute (cfg.presentation['kern-wl-left'],
str);
                elseif mw.ustring.match (str, '%[%[.+[\"“”\'‘’]%]%]') then
-- trailing quote marks
                        str = substitute (cfg.presentation['kern-wl-right'],
str);
                end

        else
-- plain text or [[L|D]]; text in label variable
                label= mw.ustring.gsub (label, '[“”]', '\"');
-- replace “” (U+201C & U+201D) with " (typewriter double quote mark)
                label= mw.ustring.gsub (label, '[‘’]', '\'');
-- replace ‘’ (U+2018 & U+2019) with ' (typewriter single quote mark)

                cap, cap2 = mw.ustring.match (label, "^([\"\'])([^\'].+)");
-- match leading double or single quote but not doubled single quotes (italic
markup)
                if is_set (cap) then
                        label = substitute (cfg.presentation['kern-left'],
```

```
{cap, cap2});
                end
                cap, cap2 = mw.ustring.match (label, "^(.+[^\'])([\"\'])$")
-- match trailing double or single quote but not doubled single quotes
(italic markup)
                if is_set (cap) then
                        label = substitute (cfg.presentation['kern-right'],
{cap, cap2});
                end
                if 2 == wl_type then
                        str = make_wikilink (link, label);
-- reassemble the wikilink
                else
                        str = label;
                end
        end
        return str;
end


--[[--------------------------< F O R M A T _ S C R I P T _ V A L U E >------
----------------------------------

|script-title= holds title parameters that are not written in Latin based
scripts: Chinese, Japanese, Arabic, Hebrew, etc. These scripts should
not be italicized and may be written right-to-left.  The value supplied by
|script-title= is concatenated onto Title after Title has been wrapped
in italic markup.

Regardless of language, all values provided by |script-title= are wrapped in
<bdi>...</bdi> tags to isolate rtl languages from the English left to right.

|script-title= provides a unique feature.  The value in |script-title= may be
prefixed with a two-character ISO639-1 language code and a colon:
        |script-title=ja:*** *** (where * represents a Japanese character)
Spaces between the two-character code and the colon and the colon and the
first script character are allowed:
        |script-title=ja : *** ***
        |script-title=ja: *** ***
        |script-title=ja :*** ***
Spaces preceding the prefix are allowed: |script-title = ja:*** ***

The prefix is checked for validity.  If it is a valid ISO639-1 language code,
the lang attribute (lang="ja") is added to the <bdi> tag so that browsers can
know the language the tag contains.  This may help the browser render the
script more correctly.  If the prefix is invalid, the lang attribute
is not added.  At this time there is no error message for this condition.

Supports |script-title=, |script-chapter=, |script-<periodical>=

]]
```

```
local function format_script_value (script_value, script_param)
        local lang='';
-- initialize to empty string
        local name;
        if script_value:match('^%l%l%l?%s*:') then
-- if first 3 or 4 non-space characters are script language prefix
                lang = script_value:match('^(%l%l%l?)%s*:%s*%S.*');
-- get the language prefix or nil if there is no script
                if not is_set (lang) then
                        table.insert( z.message_tail, { set_error(
'script_parameter', {script_param, 'missing title part'}, true ) } );
-- prefix without 'title'; add error message
                        return '';
-- script_value was just the prefix so return empty string
                end
-- if we get this far we have prefix and script
                name = cfg.lang_code_remap[lang] or
mw.language.fetchLanguageName( lang, cfg.this_wiki_code );        -- get
language name so that we can use it to categorize
                if is_set (name) then
-- is prefix a proper ISO 639-1 language code?
                        script_value = script_value:gsub ('^%l+%s*:%s*', '');
-- strip prefix from script
-- is prefix one of these language codes?
                        if in_array (lang, cfg.script_lang_codes) then
                                add_prop_cat ('script_with_name', {name,
lang})
                        else
                                table.insert( z.message_tail, { set_error(
'script_parameter', {script_param, 'unknown language code'}, true ) } );
-- unknown script-language; add error message
                        end
                        lang = ' lang="' .. lang .. '" ';
-- convert prefix into a lang attribute
                else
                        table.insert( z.message_tail, { set_error(
'script_parameter', {script_param, 'invalid language code'}, true ) } );
-- invalid language code; add error message
                        lang = '';
-- invalid so set lang to empty string
                end
        else
                table.insert( z.message_tail, { set_error(
'script_parameter', {script_param, 'missing prefix'}, true ) } );
-- no language code prefix; add error message
        end
        script_value = substitute (cfg.presentation['bdi'], {lang,
script_value});        -- isolate in case script is rtl

        return script_value;
end
```

```
--[[--------------------------< S C R I P T _ C O N C A T E N A T E >--------
---------------------------------

Initially for |title= and |script-title=, this function concatenates those
two parameter values after the script value has been
wrapped in <bdi> tags.
]]

local function script_concatenate (title, script, script_param)
	if is_set (script) then
		script = format_script_value (script, script_param);
-- <bdi> tags, lang atribute, categorization, etc; returns empty string on
error
		if is_set (script) then
			title = title .. ' ' .. script;
-- concatenate title and script title
		end
	end
	return title;
end


--[[--------------------------< W R A P _ M S G >----------------------------
---------------------------------

Applies additional message text to various parameter values. Supplied string
is wrapped using a message_list
configuration taking one argument.  Supports lower case text for {{citation}}
templates.  Additional text taken
from citation_config.messages - the reason this function is similar to but
separate from wrap_style().

]]

local function wrap_msg (key, str, lower)
	if not is_set( str ) then
		return "";
	end
	if true == lower then
		local msg;
		msg = cfg.messages[key]:lower();
-- set the message to lower case before
		return substitute( msg, str );
-- including template text
	else
		return substitute( cfg.messages[key], str );
	end
end


--[[--------------------------< W I K I S O U R C E _ U R L _ M A K E >------
```

```
---------------------------------

makes a wikisource url from wikisource interwiki link.  returns the url and
appropriate label; nil else.

str is the value assigned to |chapter= (or aliases) or |title= or |title-
link=

]]

local function wikisource_url_make (str)
	local wl_type, D, L;
	local ws_url, ws_label;
	local wikisource_prefix = table.concat ({'https://',
cfg.this_wiki_code, '.wikisource.org/wiki/'});

	wl_type, D, L = is_wikilink (str);
-- wl_type is 0 (not a wikilink), 1 (simple wikilink), 2 (complex wikilink)

	if 0 == wl_type then
-- not a wikilink; might be from |title-link=
		str = D:match ('^[Ww]ikisource:(.+)') or D:match
('^[Ss]:(.+)');		-- article title from interwiki link with
long-form or short-form namespace
		if is_set (str) then
			ws_url = table.concat ({
-- build a wikisource url
				wikisource_prefix,
-- prefix
				str,
-- article title
				});
			ws_label = str;
-- label for the url
		end
	elseif 1 == wl_type then
-- simple wikilink: [[Wikisource:ws article]]
		str = D:match ('^[Ww]ikisource:(.+)') or D:match
('^[Ss]:(.+)');		-- article title from interwiki link with
long-form or short-form namespace
		if is_set (str) then
			ws_url = table.concat ({
-- build a wikisource url
				wikisource_prefix,
-- prefix
				str,
-- article title
				});
			ws_label = str;
-- label for the url
		end
```

```
        elseif 2 == wl_type then
-- non-so-simple wikilink: [[Wikisource:ws article|displayed text]] ([[L|D]])
                str = L:match ('^[Ww]ikisource:(.+)') or L:match
('^[Ss]:(.+)');                 -- article title from interwiki link with
long-form or short-form namespace
                if is_set (str) then
                        ws_label = D;
-- get ws article name from display portion of interwiki link
                        ws_url = table.concat ({
-- build a wikisource url
                                wikisource_prefix,
-- prefix
                                str,
-- article title without namespace from link portion of wikilink
                                });
                end
        end
        if ws_url then
                ws_url = mw.uri.encode (ws_url, 'WIKI');
-- make a usable url
                ws_url = ws_url:gsub ('%%23', '#');
-- undo percent encoding of fragment marker
        end

        return ws_url, ws_label, L or D;
-- return proper url or nil and a label or nil
end


--[[--------------------------< F O R M A T _ P E R I O D I C A L >----------
----------------------------------

Format the three periodical parameters: |script-<periodical>=,
|<periodical>=, and |trans-<periodical>= into a single Periodical meta-
parameter.

]]

local function format_periodical (script_periodical,
script_periodical_source, periodical, trans_periodical)
        local periodical_error = '';

        if not is_set (periodical) then
                periodical = '';
-- to be safe for concatenation
        else
                periodical = wrap_style ('italic-title', periodical);
-- style
        end

        periodical = script_concatenate (periodical, script_periodical,
```

```
script_periodical_source);        -- <bdi> tags, lang atribute,
categorization, etc; must be done after title is wrapped

        if is_set (trans_periodical) then
                trans_periodical = wrap_style ('trans-italic-title',
trans_periodical);
                if is_set (periodical) then
                        periodical = periodical ..  ' ' .. trans_periodical;
                else
-- here when trans-periodical without periodical or script-periodical
                        periodical = trans_periodical;
                        periodical_error = ' ' .. set_error
('trans_missing_title', {'periodical'});
                end
        end

        return periodical .. periodical_error;
end


--[[--------------------------< F O R M A T _ C H A P T E R _ T I T L E >----
-----------------------------------

Format the four chapter parameters: |script-chapter=, |chapter=, |trans-
chapter=, and |chapter-url= into a single Chapter meta-
parameter (chapter_url_source used for error messages).

]]

local function format_chapter_title (script_chapter, script_chapter_source,
chapter, chapter_source, trans_chapter, trans_chapter_source, chapter_url,
chapter_url_source, no_quotes, access)
        local chapter_error = '';

        local ws_url, ws_label, L = wikisource_url_make (chapter);
-- make a wikisource url and label from a wikisource interwiki link
        if ws_url then
                ws_label = ws_label:gsub ('_', '');
-- replace underscore separaters with space characters
                chapter = ws_label;
        end

        if not is_set (chapter) then
                chapter = '';
-- to be safe for concatenation
        else
                if false == no_quotes then
                        chapter = kern_quotes (chapter);
-- if necessary, separate chapter title's leading and trailing quote marks
from Module provided quote marks
                        chapter = wrap_style ('quoted-title', chapter);
```

```
                end
        end

        chapter = script_concatenate (chapter, script_chapter,
script_chapter_source);        -- <bdi> tags, lang atribute, categorization,
etc; must be done after title is wrapped

        if is_set (chapter_url) then
                chapter = external_link (chapter_url, chapter,
chapter_url_source, access);        -- adds bare_url_missing_title error if
appropriate
        elseif ws_url then
                chapter = external_link (ws_url, chapter .. ' ', 'ws
link in chapter');        -- adds bare_url_missing_title error if
appropriate; space char to move icon away from chap text; TODO: better way to
do this?
                chapter = substitute (cfg.presentation['interwiki-icon'],
{cfg.presentation['class-wikisource'], L, chapter});
        end

        if is_set (trans_chapter) then
                trans_chapter = wrap_style ('trans-quoted-title',
trans_chapter);
                if is_set (chapter) then
                        chapter = chapter ..  ' ' .. trans_chapter;
                else
-- here when trans_chapter without chapter or script-chapter
                        chapter = trans_chapter;
                        chapter_source = trans_chapter_source:match ('trans%-
?(.+)');                -- when no chapter, get matching name from trans-
<param>
                        chapter_error = ' ' .. set_error
('trans_missing_title', {chapter_source});
                end
        end

        return chapter .. chapter_error;
end


--[[--------------------------< H A S _ I N V I S I B L E _ C H A R S >------
----------------------------------

This function searches a parameter's value for nonprintable or invisible
characters.  The search stops at the
first match.

This function will detect the visible replacement character when it is part
of the wikisource.

Detects but ignores nowiki and math stripmarkers.  Also detects other named
```

```
stripmarkers (gallery, math, pre, ref)
and identifies them with a slightly different error message.  See also
coins_cleanup().

Output of this function is an error message that identifies the character or
the Unicode group, or the stripmarker
that was detected along with its position (or, for multi-byte characters, the
position of its first byte) in the
parameter value.

]]

local function has_invisible_chars (param, v)
        local position = '';
-- position of invisible char or starting position of stripmarker
        local dummy;
-- end of matching string; not used but required to hold end position when a
capture is returned
        local capture;
-- used by stripmarker detection to hold name of the stripmarker
        local i=1;
        local stripmarker, apostrophe;
        capture = string.match (v, '[%w%p ]*');
-- Test for values that are simple ASCII text and bypass other tests if true
        if capture == v then
-- if same there are no unicode characters
                return;
        end

        while cfg.invisible_chars[i] do
                local char=cfg.invisible_chars[i][1]
-- the character or group name
                local pattern=cfg.invisible_chars[i][2]
-- the pattern used to find it
                position, dummy, capture = mw.ustring.find (v, pattern)
-- see if the parameter value contains characters that match the pattern
                if position and (char == 'zero width joiner') then
-- if we found a zero width joiner character
                        if mw.ustring.find (v, cfg.indic_script) then
-- its ok if one of the indic scripts
                                position = nil;
-- unset position
                        end
                end
                if position then
                        if 'nowiki' == capture or 'math' == capture or
-- nowiki and math stripmarkers (not an error condition)
                                ('templatestyles' == capture and in_array
(param, {'id', 'quote'})) then        -- templatestyles stripmarker allowed
in these parameters
                                        stripmarker = true;
```

```
-- set a flag
                        elseif true == stripmarker and 'delete' == char then
-- because stripmakers begin and end with the delete char, assume that we've
found one end of a stripmarker
                                position = nil;
-- unset
                        else
                                local err_msg;
                                if capture then
                                        err_msg = capture .. ' ' .. char;
                                else
                                        err_msg = char .. ' ' .. 'character';
                                end

                                table.insert( z.message_tail, { set_error(
'invisible_char', {err_msg, wrap_style ('parameter', param), position}, true
) } );        -- add error message
                                return;
-- and done with this parameter
                        end
                end
                i=i+1;
-- bump our index
        end
end


--[[--------------------------< A R G U M E N T _ W R A P P E R >------------
----------------------------------

Argument wrapper.  This function provides support for argument mapping
defined in the configuration file so that
multiple names can be transparently aliased to single internal variable.

]]

local function argument_wrapper( args )
        local origin = {};
        return setmetatable({
                ORIGIN = function( self, k )
                        local dummy = self[k]; --force the variable to be
loaded.
                        return origin[k];
                end
        },
        {
                __index = function ( tbl, k )
                        if origin[k] ~= nil then
                                return nil;
                        end
                        local args, list, v = args, cfg.aliases[k];
```

```
                        if type( list ) == 'table' then
                                v, origin[k] = select_one( args, list,
'redundant_parameters' );
                                if origin[k] == nil then
                                        origin[k] = ''; -- Empty string, not
nil
                                end
                        elseif list ~= nil then
                                v, origin[k] = args[list], list;
                        else
                                -- maybe let through instead of raising an
error?
                                -- v, origin[k] = args[k], k;
                                error( cfg.messages['unknown_argument_map']
.. ': ' .. k);
                        end
                        -- Empty strings, not nil;
                        if v == nil then
--                              v = cfg.defaults[k] or '';
                                v = '';
                                origin[k] = '';
                        end
                        tbl = rawset( tbl, k, v );
                        return v;
                end,
        });
end


--[[--------------------------< N O W R A P _ D A T E >----------------------
----------------------------------

When date is YYYY-MM-DD format wrap in nowrap span: <span ...>YYYY-MM-
DD</span>.  When date is DD MMMM YYYY or is
MMMM DD, YYYY then wrap in nowrap span: <span ...>DD MMMM</span> YYYY or
<span ...>MMMM DD,</span> YYYY

DOES NOT yet support MMMM YYYY or any of the date ranges.

]]

local function nowrap_date (date)
        local cap='';
        local cap2='';

        if date:match("^%d%d%d%d%-%d%d%-%d%d$") then
                date = substitute (cfg.presentation['nowrap1'], date);
        elseif date:match("^%a+%s*%d%d?,%s+%d%d%d%d$") or date:match
("^%d%d?%s*%a+%s+%d%d%d%d$") then
                cap, cap2 = string.match (date, "^(.*)%s+(%d%d%d%d)$");
                date = substitute (cfg.presentation['nowrap2'], {cap, cap2});
```

```
        end
        return date;
end


--[[--------------------------< S E T _ T I T L E T Y P E >----------------
-----------------------------------

This function sets default title types (equivalent to the citation including
|type=<default value>) for those templates that have defaults.
Also handles the special case where it is desirable to omit the title type
from the rendered citation (|type=none).

]]

local function set_titletype (cite_class, title_type)
        if is_set (title_type) then
                if 'none' == cfg.keywords_xlate[title_type] then
                        title_type = '';
-- if |type=none then type parameter not displayed
                end
                return title_type;
-- if |type= has been set to any other value use that value
        end

        return cfg.title_types [cite_class] or '';
-- set template's default title type; else empty string for concatenation
end


--[[--------------------------< H Y P H E N _ T O _ D A S H >--------------
-----------------------------------

Converts a hyphen to a dash under certain conditions.  The hyphen must
separate like items; unlike items are
returned unmodified.  These forms are modified:
        letter - letter (A - B)
        digit - digit (4-5)
        digit separator digit - digit separator digit (4.1-4.5 or 4-1-4-5)
        letterdigit - letterdigit (A1-A5) (an optional separator between
letter and digit is supported – a.1-a.5 or a-1-a-5)
        digitletter - digitletter (5a - 5d) (an optional separator between
letter and digit is supported – 5.a-5.d or 5-a-5-d)

any other forms are returned unmodified.

str may be a comma- or semicolon-separated list

]]

local function hyphen_to_dash( str )
```

```
        if not is_set (str) then
                return str;
        end
        str, count = str:gsub ('^%(%((.+)%)%)$', '%1');
-- remove accept-this-as-written markup when it wraps all of str
        if 0 ~= count then
-- non-zero when markup removed; zero else
                return str;
-- nothing to do, we're done
        end
        str = str:gsub ('&[nm]dash;', {['&ndash;'] = '–', ['&mdash;'] =
'—'});                -- replace &mdash; and &ndash; entities  with their
characters; semicolon mucks up the text.split
        str = str:gsub ('&#45;', '-');
-- replace html numeric entity with hyphen character
        str = str:gsub (' ', ' ');
-- replace   entity with generic keyboard space character
        local out = {};
        local list = mw.text.split (str, '%s*[,;]%s*');
-- split str at comma or semicolon separators if there are any

        for _, item in ipairs (list) do
-- for each item in the list
                if mw.ustring.match (item, '^%w*[%.%-]?%w+%s*[%-
——]%s*%w*[%.%-]?%w+$') then        -- if a hyphenated range or has endash or
emdash separators
                        if item:match ('^%a+[%.%-]?%d+%s*%-%s*%a+[%.%-
]?%d+$') or                        -- letterdigit hyphen letterdigit
(optional separator between letter and digit)
                                item:match ('^%d+[%.%-]?%a+%s*%-%s*%d+[%.%-
]?%a+$') or                        -- digitletter hyphen digitletter
(optional separator between digit and letter)
                                item:match ('^%d+[%.%-]%d+%s*%-%s*%d+[%.%-
]%d+$') or                           -- digit separator digit hyphen digit
separator digit
                                item:match ('^%d+%s*%-%s*%d+$') or
-- digit hyphen digit
                                item:match ('^%a+%s*%-%s*%a+$') then
-- letter hyphen letter
                                        item = item:gsub ('(%w*[%.%-
]?%w+)%s*%-%s*(%w*[%.%-]?%w+)', '%1–%2');        -- replace hyphen, remove
extraneous space characters
                        else
                                item = mw.ustring.gsub (item, '%s*[——]%s*',
'–');                                -- for endash or emdash separated
ranges, replace em with en, remove extraneous white space
                        end
                end
                item = item:gsub ('^%(%((.+)%)%)$', '%1');
-- remove the accept-this-as-written markup
                table.insert (out, item);
```

```
-- add the (possibly modified) item to the output table
        end

        return table.concat (out, ', ');
-- concatenate the output table into a comma separated string
end


--[[--------------------------< S A F E _ J O I N >--------------------------
----------------------------------

Joins a sequence of strings together while checking for duplicate separation
characters.

]]

local function safe_join( tbl, duplicate_char )
        local f = {};
-- create a function table appropriate to type of 'dupicate character'
                if 1 == #duplicate_char then
-- for single byte ascii characters use the string library functions
                        f.gsub=string.gsub
                        f.match=string.match
                        f.sub=string.sub
                else
-- for multi-byte characters use the ustring library functions
                        f.gsub=mw.ustring.gsub
                        f.match=mw.ustring.match
                        f.sub=mw.ustring.sub
                end

        local str = '';
-- the output string
        local comp = '';
-- what does 'comp' mean?
        local end_chr = '';
        local trim;
        for _, value in ipairs( tbl ) do
                if value == nil then value = ''; end
                if str == '' then
-- if output string is empty
                        str = value;
-- assign value to it (first time through the loop)
                elseif value ~= '' then
                        if value:sub(1,1) == '<' then
-- Special case of values enclosed in spans and other markup.
                                comp = value:gsub( "%b<>", "" );
-- remove html markup (<span>string</span> -> string)
                        else
                                comp = value;
                        end
```

```
-- typically duplicate_char is sepc
                        if f.sub(comp, 1,1) == duplicate_char then
-- is first character same as duplicate_char? why test first character?
--   Because individual string segments often (always?) begin with terminal
punct for the
--   preceding segment: 'First element' .. 'sepc next element' .. etc?
                                trim = false;
                                end_chr = f.sub(str, -1,-1);
-- get the last character of the output string
                                -- str = str .. "<HERE(enchr=" .. end_chr..
")"                                        -- debug stuff?
                                if end_chr == duplicate_char then
-- if same as separator
                                        str = f.sub(str, 1,-2);
-- remove it
                                elseif end_chr == "'" then
-- if it might be wikimarkup
                                        if f.sub(str, -3,-1) ==
duplicate_char .. "'" then                      -- if last three chars of
str are sepc''
                                                str = f.sub(str, 1, -4) ..
"'";                                                 -- remove them and add
back ''
                                        elseif  f.sub(str, -5,-1) ==
duplicate_char .. "]]'" then        -- if last five chars of str are
sepc]]''
                                                trim = true;
-- why? why do this and next differently from previous?
                                        elseif f.sub(str, -4,-1) ==
duplicate_char .. "]'" then        -- if last four chars of str are sepc]''
                                                trim = true;
-- same question
                                        end
                                elseif end_chr == "]" then
-- if it might be wikimarkup
                                        if f.sub(str, -3,-1) ==
duplicate_char .. "]]" then                      -- if last three chars of
str are sepc]] wikilink
                                                trim = true;
                                        elseif f.sub(str, -3,-1) ==
duplicate_char .. '"]' then              -- if last three chars of str are
sepc"] quoted external link
                                                trim = true;
                                        elseif  f.sub(str, -2,-1) ==
duplicate_char .. "]" then               -- if last two chars of str are
sepc] external link
                                                trim = true;
                                        elseif f.sub(str, -4,-1) ==
duplicate_char .. "']" then        -- normal case when |url=something &
|title=Title.
                                                trim = true;
```

```
                                        end
                                elseif end_chr == " " then
-- if last char of output string is a space
                                        if f.sub(str, -2,-1) ==
duplicate_char .. " " then                      -- if last two chars of str
are <sepc><space>
                                                str = f.sub(str, 1,-3);
-- remove them both
                                        end
                                end

                                if trim then
                                        if value ~= comp then
-- value does not equal comp when value contains html markup
                                                local dup2 = duplicate_char;
                                                if f.match(dup2, "%A" ) then
dup2 = "%" .. dup2; end        -- if duplicate_char not a letter then escape
it
                                                value = f.gsub(value,
"(%b<>)" .. dup2, "%1", 1 )               -- remove duplicate_char if it
follows html markup
                                        else
                                                value = f.sub(value, 2, -1 );
-- remove duplicate_char when it is first character
                                        end
                                end
                        end
                        str = str .. value;
--add it to the output string
                end
        end
        return str;
end


--[[--------------------------< I S _ S U F F I X >-------------------------
-----------------------------------

returns true is suffix is properly formed Jr, Sr, or ordinal in the range
2–9.  Puncutation not allowed.

]]

local function is_suffix (suffix)
        if in_array (suffix, {'Jr', 'Sr', '2nd', '3rd'}) or suffix:match
('^%dth$') then
                return true;
        end
        return false;
end
```

```
--[[--------------------------< I S _ G O O D _ V A N C _ N A M E >----------
---------------------------------

For Vancouver Style, author/editor names are supposed to be rendered in Latin
(read ASCII) characters.  When a name
uses characters that contain diacritical marks, those characters are to
converted to the corresponding Latin character.
When a name is written using a non-Latin alphabet or logogram, that name is
to be transliterated into Latin characters.
These things are not currently possible in this module so are left to the
editor to do.

This test allows |first= and |last= names to contain any of the letters
defined in the four Unicode Latin character sets
        [http://www.unicode.org/charts/PDF/U0000.pdf C0 Controls and Basic
Latin] 0041–005A, 0061–007A
        [http://www.unicode.org/charts/PDF/U0080.pdf C1 Controls and Latin-1
Supplement] 00C0–00D6, 00D8–00F6, 00F8–00FF
        [http://www.unicode.org/charts/PDF/U0100.pdf Latin Extended-A]
0100–017F
        [http://www.unicode.org/charts/PDF/U0180.pdf Latin Extended-B]
0180–01BF, 01C4–024F

|lastn= also allowed to contain hyphens, spaces, and apostrophes.
(http://www.ncbi.nlm.nih.gov/books/NBK7271/box/A35029/)
|firstn= also allowed to contain hyphens, spaces, apostrophes, and periods

This original test:
        if nil == mw.ustring.find (last, "^[A-Za-zÀ-ÖØ-öø-ƿ-%-%s%']*$") or
nil == mw.ustring.find (first, "^[A-Za-zÀ-ÖØ-öø-ƿ-%-%s%'%.]+[2-6%a]*$")
then
was written ouside of the code editor and pasted here because the code editor
gets confused between character insertion point and cursor position.
The test has been rewritten to use decimal character escape sequence for the
individual bytes of the unicode characters so that it is not necessary
to use an external editor to maintain this code.

        \195\128-\195\150 – À-Ö (U+00C0–U+00D6 – C0 controls)
        \195\152-\195\182 – Ø-ö (U+00D8-U+00F6 – C0 controls)
        \195\184-\198\191 – ø-ƿ (U+00F8-U+01BF – C0 controls, Latin extended
A & B)
        \199\132-\201\143 – -  (U+01C4-U+024F – Latin extended B)

]]

local function is_good_vanc_name (last, first, suffix)
        if not suffix then
                if first:find ('[,%s]') then
-- when there is a space or comma, might be first name/initials +
generational suffix
                        first = first:match ('(.-)[,%s]+');
```

```
-- get name/initials
                        suffix = first:match ('[,%s]+(.+)$');
-- get generational suffix
                end
        end
        if is_set (suffix) then
                if not is_suffix (suffix) then
                        add_vanc_error (cfg.err_msg_supl.suffix);
                        return false;
-- not a name with an appropriate suffix
                end
        end
        if nil == mw.ustring.find (last, "^[A-Za-z\195\128-\195\150\195\152-
\195\182\195\184-\198\191\199\132-\201\143%-%s%']*$") or
                nil == mw.ustring.find (first, "^[A-Za-z\195\128-
\195\150\195\152-\195\182\195\184-\198\191\199\132-\201\143%-%s%'%.]*$") then
                        add_vanc_error (cfg.err_msg_supl['non-Latin char']);
                        return false;
-- not a string of latin characters; Vancouver requires Romanization
        end;
        return true;
end


--[[-------------------------< R E D U C E _ T O _ I N I T I A L S >--------
----------------------------------

Attempts to convert names to initials in support of |name-list-format=vanc.

Names in |firstn= may be separated by spaces or hyphens, or for initials, a
period. See http://www.ncbi.nlm.nih.gov/books/NBK7271/box/A35062/.

Vancouver style requires family rank designations (Jr, II, III, etc) to be
rendered as Jr, 2nd, 3rd, etc.  See
http://www.ncbi.nlm.nih.gov/books/NBK7271/box/A35085/.
This code only accepts and understands generational suffix in the Vancouver
format because Roman numerals look like, and can be mistaken for, initials.

This function uses ustring functions because firstname initials may be any of
the unicode Latin characters accepted by is_good_vanc_name ().

]]

local function reduce_to_initials(first)
        local name, suffix = mw.ustring.match(first, "^(%u+)
([%dJS][%drndth]+)$");

        if not name then
-- if not initials and a suffix
                name = mw.ustring.match(first, "^(%u+)$");
-- is it just intials?
```

```
        end

        if name then
-- if first is initials with or without suffix
                if 3 > mw.ustring.len (name) then
-- if one or two initials
                        if suffix then
-- if there is a suffix
                                if is_suffix (suffix) then
-- is it legitimate?
                                        return first;
-- one or two initials and a valid suffix so nothing to do
                                else
                                        add_vanc_error
(cfg.err_msg_supl.suffix);                                              -- one or
two initials with invalid suffix so error message
                                        return first;
-- and return first unmolested
                                end
                        else
                                return first;
-- one or two initials without suffix; nothing to do
                        end
                end
        end
-- if here then name has 3 or more uppercase letters so treat them as a word

        local initials, names = {}, {};
-- tables to hold name parts and initials
        local i = 1;
-- counter for number of initials

        names = mw.text.split (first, '[%s,]+');
-- split into a table of names and possible suffix

        while names[i] do
-- loop through the table
                if 1 < i and names[i]:match ('[%dJS][%drndth]+%.?$') then
-- if not the first name, and looks like a suffix (may have trailing dot)
                        names[i] = names[i]:gsub ('%.', '');
-- remove terminal dot if present
                        if is_suffix (names[i]) then
-- if a legitimate suffix
                                table.insert (initials, ' ' .. names[i]);
-- add a separator space, insert at end of initials table
                                break;
-- and done because suffix must fall at the end of a name
                        end
-- no error message if not a suffix; possibly because of Romanization
                end
                if 3 > i then
```

```
                        table.insert (initials,
mw.ustring.sub(names[i],1,1));                                  -- insert the
intial at end of initials table
                end
                i = i+1;
-- bump the counter
        end
        return table.concat(initials)
-- Vancouver format does not include spaces.
end


--[[--------------------------< L I S T  _ P E O P L E >--------------------
----------------------------------

Formats a list of people (e.g. authors / editors)

]]

local function list_people(control, people, etal)
        local sep;
        local namesep;
        local format = control.format
        local maximum = control.maximum
        local lastauthoramp = control.lastauthoramp;
        local text = {}

        if 'vanc' == format then
-- Vancouver-like author/editor name styling?
                sep = cfg.presentation['sep_nl_vanc'];
-- name-list separator between authors is a comma
                namesep = cfg.presentation['sep_name_vanc'];
-- last/first separator is a space
                lastauthoramp = nil;
-- unset because isn't used by Vancouver style
        else
                sep = cfg.presentation['sep_nl'];
-- name-list separator between authors is a semicolon
                namesep = cfg.presentation['sep_name'];
-- last/first separator is <comma><space>
        end
        if sep:sub(-1,-1) ~= " " then sep = sep .. " " end
        if is_set (maximum) and maximum < 1 then return "", 0; end
-- returned 0 is for EditorCount; not used for authors
        for i,person in ipairs(people) do
                if is_set(person.last) then
                        local mask = person.mask
                        local one
                        local sep_one = sep;
                        if is_set (maximum) and i > maximum then
                                etal = true;
```

```
                                break;
                        elseif (mask ~= nil) then
                                local n = tonumber(mask)
                                if (n ~= nil) then
                                        one = string.rep("&mdash;",n)
                                else
                                        one = mask;
                                        sep_one = " ";
                                end
                        else
                                one = person.last
                                local first = person.first
                                if is_set(first) then
                                        if ( "vanc" == format ) then
-- if vancouver format
                                                one = one:gsub ('%.', '');
-- remove periods from surnames
(http://www.ncbi.nlm.nih.gov/books/NBK7271/box/A35029/)
                                                if not person.corporate and
is_good_vanc_name (one, first) then                -- and name is all Latin
characters; corporate authors not tested
                                                        first =
reduce_to_initials(first)                                       -- attempt
to convert first name(s) to initials
                                                end
                                        end
                                        one = one .. namesep .. first;
                                end
                        end
                        if is_set (person.link) then
                                one = make_wikilink (person.link, one);
-- link author/editor
                        end
                        table.insert (text, one)
                        table.insert (text, sep_one)
                end
        end

        local count = #text / 2;
-- (number of names + number of separators) divided by 2
        if count > 0 then
                if count > 1 and is_set(lastauthoramp) and not etal then
                        text[#text-2] = " & ";
-- replace last separator with ampersand text
                end
                text[#text] = nil;
-- erase the last separator
        end
        local result = table.concat(text)
-- construct list
        if etal and is_set (result) then
```

```
-- etal may be set by |display-authors=etal but we might not have a last-
first list
                result = result .. sep .. ' ' .. cfg.messages['et al'];
-- we've go a last-first list and etal so add et al.
        end
        return result, count
end


--[[--------------------------< A N C H O R _ I D >--------------------------
---------------------------------

Generates a CITEREF anchor ID if we have at least one name or a date.
Otherwise returns an empty string.

namelist is one of the contributor-, author-, or editor-name lists chosen in
that order.  year is Year or anchor_year.

]]

local function anchor_id (namelist, year)
        local names={};
-- a table for the one to four names and year
        for i,v in ipairs (namelist) do
-- loop through the list and take up to the first four last names
                names[i] = v.last
                if i == 4 then break end
-- if four then done
        end
        table.insert (names, year);
-- add the year at the end
        local id = table.concat(names);
-- concatenate names and year for CITEREF id
        if is_set (id) then
-- if concatenation is not an empty string
                return "CITEREF" .. id;
-- add the CITEREF portion
        else
                return '';
-- return an empty string; no reason to include CITEREF id in this citation
        end
end


--[[--------------------------< N A M E _ H A S _ E T A L >------------------
---------------------------------

Evaluates the content of name parameters (author, editor, etc) for variations
on the theme of et al.  If found,
the et al. is removed, a flag is set to true and the function returns the
modified name and the flag.
```

```
This function never sets the flag to false but returns it's previous state
because it may have been set by
previous passes through this function or by the associated |display-
<names>=etal parameter

]]

local function name_has_etal (name, etal, nocat, param)

        if is_set (name) then
-- name can be nil in which case just return
                local patterns = cfg.et_al_patterns;
--get patterns from configuration
                for _, pattern in ipairs (patterns) do
-- loop through all of the patterns
                        if name:match (pattern) then
-- if this 'et al' pattern is found in name
                                name = name:gsub (pattern, '');
-- remove the offending text
                                etal = true;
-- set flag (may have been set previously here or by |display-<names>=etal)
                                if not nocat then
-- no categorization for |vauthors=
                                        table.insert( z.message_tail,
{set_error ('etal', {param})});        -- and set an error if not added
                                end
                        end
                end
        end

        return name, etal;
--
end


--[[--------------------------< N A M E _ I S _ N U M E R I C >--------------
----------------------------------

Add maint cat when name parameter value does not contain letters.  Does not
catch mixed alphanumeric names so
|last=A. Green (1922-1987) does not get caught in the current version of this
test but |first=(1888) is caught.

returns nothing

]]

local function name_is_numeric (name, list_name)
        if is_set (name) then
                if mw.ustring.match (name, '^[%A]+$') then
-- when name does not contain any letters
```

```
                        add_maint_cat ('numeric_names',
cfg.special_case_translation [list_name]);        -- add a maint cat for this
template
                end
        end
end


--[[--------------------------< N A M E _ H A S _ E D _ M A R K U P >--------
----------------------------------

Evaluates the content of author and editor parameters for extranious editor
annotations: ed, ed., eds, (Ed.), etc.
These annotation do not belong in author parameters and are redundant in
editor parameters.  If found, the function
adds the editor markup maintenance category.

returns nothing

]]

local function name_has_ed_markup (name, list_name)
        local patterns = cfg.editor_markup_patterns;
-- get patterns from configuration

        if is_set (name) then
                for _, pattern in ipairs (patterns) do
-- spin through patterns table and
                        if name:match (pattern) then
                                add_maint_cat ('extra_text_names',
cfg.special_case_translation [list_name]);        -- add a maint cat for this
template
                                break;
                        end
                end
        end
end


--[[--------------------------< N A M E _ H A S _ M U L T _ N A M E S >------
----------------------------------

Evaluates the content of author and editor (surnames only) parameters for
multiple names.  Multiple names are
indicated if there is more than one comma and or semicolon.  If found, the
function adds the multiple name
(author or editor) maintenance category.

returns nothing

]]
```

```
local function name_has_mult_names (name, list_name)
	local _, count;
	if is_set (name) then
		_, count = name:gsub ('[;,]', '');
-- count the number of separator-like characters
		if 1 < count then
-- param could be |author= or |editor= so one separator character is
acceptable
			add_maint_cat ('mult_names',
cfg.special_case_translation [list_name]);	-- more than one separator
indicates multiple names so add a maint cat for this template
		end
	end
end


--[[--------------------------< N A M E _ C H E C K S >--------------------
---------------------------------

This function calls various name checking functions used to validate the
content of the various name-holding
parameters.

]]

local function name_checks (last, first, list_name)
	if is_set (last) then
		if last:match ('^%(%(.*%)%)$') then
-- if wrapped in doubled parentheses, accept as written
			last = last:match ('^%(%((.*)%)%)$');
-- strip parens
		else
			name_has_mult_names (last, list_name);
-- check for multiple names in the parameter (last only)
			name_has_ed_markup (last, list_name);
-- check for extraneous 'editor' annotation
			name_is_numeric (last, list_name);
-- check for names that are compsed of digits and punctuation
		end
	end
	if is_set (first) then
		if first:match ('^%(%(.*%)%)$') then
-- if wrapped in doubled parentheses, accept as written
			first = first:match ('^%(%((.*)%)%)$');
-- strip parens
		else
			name_has_ed_markup (first, list_name);
-- check for extraneous 'editor' annotation
			name_is_numeric (first, list_name);
-- check for names that are compsed of digits and punctuation
		end
```

```
        end
        return last, first;
-- done
        end


--[[--------------------------< E X T R A C T _ N A M E S >--------------------
-----------------------------------
Gets name list from the input arguments

Searches through args in sequential order to find |lastn= and |firstn=
parameters (or their aliases), and their matching link and mask parameters.
Stops searching when both |lastn= and |firstn= are not found in args after
two sequential attempts: found |last1=, |last2=, and |last3= but doesn't
find |last4= and |last5= then the search is done.

This function emits an error message when there is a |firstn= without a
matching |lastn=.  When there are 'holes' in the list of last names, |last1=
and |last3=
are present but |last2= is missing, an error message is emitted. |lastn= is
not required to have a matching |firstn=.

When an author or editor parameter contains some form of 'et al.', the 'et
al.' is stripped from the parameter and a flag (etal) returned
that will cause list_people() to add the static 'et al.' text from
Module:Citation/CS1/Configuration.  This keeps 'et al.' out of the
template's metadata.  When this occurs, the page is added to a maintenance
category.

]]

local function extract_names(args, list_name)
        local names = {};                           -- table of names
        local last;                                         -- individual name
components
        local first;
        local link;
        local mask;
        local i = 1;                                   -- loop counter/indexer
        local n = 1;                                   -- output table indexer
        local count = 0;                            -- used to count the number
of times we haven't found a |last= (or alias for authors, |editor-last or
alias for editors)
        local etal=false;                           -- return value set to true
when we find some form of et al. in an author parameter

        local last_alias, first_alias, link_alias;
-- selected parameter aliases used in error messaging
        while true do
                last, last_alias = select_one( args, cfg.aliases[list_name ..
'-Last'], 'redundant_parameters', i );                -- search through args
```

```
for name components beginning at 1
                first, first_alias = select_one( args, cfg.aliases[list_name
.. '-First'], 'redundant_parameters', i );
                link, link_alias = select_one( args, cfg.aliases[list_name ..
'-Link'], 'redundant_parameters', i );
                mask = select_one( args, cfg.aliases[list_name .. '-Mask'],
'redundant_parameters', i );

                last, etal = name_has_etal (last, etal, false, last_alias);
-- find and remove variations on et al.
                first, etal = name_has_etal (first, etal, false,
first_alias);                           -- find and remove variations on et al.
                last, first= name_checks (last, first, list_name);
-- multiple names, extraneous annotation, etc checks
                if first and not last then
-- if there is a firstn without a matching lastn
                        table.insert( z.message_tail, { set_error(
'first_missing_last', {first_alias, first_alias:gsub('first', 'last')}, true
) } );        -- add this error message
                elseif not first and not last then
-- if both firstn and lastn aren't found, are we done?
                        count = count + 1;
-- number of times we haven't found last and first
                        if 2 <= count then
-- two missing names and we give up
                                break;
-- normal exit or there is a two-name hole in the list; can't tell which
                        end
                else
-- we have last with or without a first
                        link_title_ok (link, link_alias, last, last_alias);
-- check for improper wikimarkup
                        if first then
                                link_title_ok (link, link_alias, first,
first_alias);                           -- check for improper wikimarkup
                        end

                        names[n] = {last = last, first = first, link = link,
mask = mask, corporate=false};        -- add this name to our names list
(corporate for |vauthors= only)
                        n = n + 1;
-- point to next location in the names table
                        if 1 == count then
-- if the previous name was missing
                                table.insert( z.message_tail, { set_error(
'missing_name', {list_name:match ("(%w+)List"):lower(), i-1}, true ) } );
-- add this error message
                        end
                        count = 0;
-- reset the counter, we're looking for two consecutive missing names
                end
```

```
                i = i + 1;
-- point to next args location
        end
        return names, etal;
-- all done, return our list of names
end


--[[--------------------------< G E T _ I S O 6 3 9 _ C O D E >--------------
----------------------------------

Validates language names provided in |language= parameter if not an ISO639-1
or 639-2 code.

Returns the language name and associated two- or three-character code.
Because case of the source may be incorrect
or different from the case that WikiMedia uses, the name comparisons are done
in lower case and when a match is
found, the Wikimedia version (assumed to be correct) is returned along with
the code.  When there is no match, we
return the original language name string.

mw.language.fetchLanguageNames(<local wiki language>, 'all') returns a list
of languages that in some cases may include
extensions. For example, code 'cbk-zam' and its associated name 'Chavacano de
Zamboanga' (MediaWiki does not support
code 'cbk' or name 'Chavacano'.  Most (all?) of these languages are not used
a 'language' codes per se, rather they
are used as sub-domain names: cbk-zam.wikipedia.org.  A list of language
names and codes supported by fetchLanguageNames()
can be found at Template:Citation Style documentation/language/doc

Names that are included in the list will be found if that name is provided in
the |language= parameter.  For example,
if |language=Chavacano de Zamboanga, that name will be found with the
associated code 'cbk-zam'.  When names are found
and the associated code is not two or three characters, this function returns
only the WikiMedia language name.

Some language names have multiple entries under different codes:
        Aromanian has code rup and code roa-rup
When this occurs, this function returns the language name and the 2- or 3-
character code

Adapted from code taken from Module:Check ISO 639-1.

]]

local function get_iso639_code (lang, this_wiki_code)
        if cfg.lang_name_remap[lang:lower()] then
-- if there is a remapped name (because MediaWiki uses something that we
```

```
don't think is correct)
                return cfg.lang_name_remap[lang:lower()][1],
cfg.lang_name_remap[lang:lower()][2];        -- for this language 'name',
return a possibly new name and appropriate code
        end

        local ietf_code;
-- because some languages have both ietf-like codes and iso 639-like codes
        local ietf_name;
        local languages = mw.language.fetchLanguageNames(this_wiki_code,
'all')                -- get a list of language names known to Wikimedia
-- ('all' is required for North Ndebele, South Ndebele, and Ojibwa)
        local langlc = mw.ustring.lower(lang);
-- lower case version for comparisons

        for code, name in pairs(languages) do
-- scan the list to see if we can find our language
                if langlc == mw.ustring.lower(name) then
                        if 2 == code:len() or 3 == code:len() then
-- two- or three-character codes only; extensions not supported
                                return name, code;
-- so return the name and the code
                        end
                        ietf_code = code;
-- remember that we found an ietf-like code and save its name
                        ietf_name = name;
-- but keep looking for a 2- or 3-char code
                end
        end
-- didn't find name with 2- or 3-char code; if ietf-like code found return
        return ietf_code and ietf_name or lang;
-- associated name; return original language text else
end


--[[--------------------------< L A N G U A G E _ P A R A M E T E R >--------
----------------------------------
```

Gets language name from a provided two- or three-character ISO 639 code. If a code is recognized by MediaWiki,
use the returned name; if not, then use the value that was provided with the language parameter.

When |language= contains a recognized language (either code or name), the page is assigned to the category for
that code: Category:Norwegian-language sources (no). For valid three-character code languages, the page is assigned
to the single category for '639-2' codes: Category:CS1 ISO 639-2 language sources.

Languages that are the same as the local wiki are not categorized. MediaWiki

```
does not recognize three-character
equivalents of two-character codes: code 'ar' is recognized but code 'ara' is
not.

This function supports multiple languages in the form |language=nb, French,
th where the language names or codes are
separated from each other by commas with optional space characters.

]]

local function language_parameter (lang)
        local code;
-- the two- or three-character language code
        local name;
-- the language name
        local language_list = {};
-- table of language names to be rendered
        local names_table = {};
-- table made from the value assigned to |language=

        local this_wiki_name =
mw.language.fetchLanguageName(cfg.this_wiki_code, cfg.this_wiki_code);
-- get this wiki's language name

        names_table = mw.text.split (lang, '%s*,%s*');
-- names should be a comma separated list

        for _, lang in ipairs (names_table) do
-- reuse lang
                name = cfg.lang_code_remap[lang:lower()];
-- first see if this is a code that is not supported by MediaWiki but is in
remap

                if name then
-- there was a remapped code so
                        if not lang:match ('^%a%a%a?%-x%-%a+$') then
-- if not a private ietf tag
                                lang = lang:gsub ('^(%a%a%a?)%-.*', '%1');
-- strip ietf tags from code
                        end
                else
                        lang = lang:gsub ('^(%a%a%a?)%-.*', '%1');
-- strip any ietf-like tags from code
                        if 2 == lang:len() or 3 == lang:len() then
-- if two-or three-character code
                                name = mw.language.fetchLanguageName
(lang:lower(), cfg.this_wiki_code);        -- get language name if |language=
is a proper code
                        end
                end
```

```
                if is_set (name) then
-- if |language= specified a valid code
                        code = lang:lower();
-- save it
                else
                        name, code = get_iso639_code (lang,
cfg.this_wiki_code);                                    -- attempt to get code
from name (assign name here so that we are sure of proper capitalization)
                end
                if is_set (code) then
-- only 2- or 3-character codes
                        name = cfg.lang_code_remap[code] or name;
-- override wikimedia when they misuse language codes/names

                        if cfg.this_wiki_code ~= code then
-- when the language is not the same as this wiki's language
                                if 2 == code:len() then
-- and is a two-character code
                                        add_prop_cat ('foreign_lang_source'
.. code, {name, code});        -- categorize it; code appended to allow for
multiple language categorization
                                else
-- or is a recognized language (but has a three-character code)
                                        add_prop_cat ('foreign_lang_source_2'
.. code, {code});               -- categorize it differently TODO: support
multiple three-character code categories per cs1|2 template
                                end
                        elseif cfg.local_lang_cat_enable then
-- when the language and this wiki's language are the same and categorization
is enabled
                                add_prop_cat ('local_lang_source', {name,
code});                                 -- categorize it
                        end
                else
                        add_maint_cat ('unknown_lang');
-- add maint category if not already added
                end
                table.insert (language_list, name);
                name = '';
-- so we can reuse it
        end
        code = #language_list
-- reuse code as number of languages in the list
        if 2 >= code then
                name = table.concat (language_list, cfg.messages['parameter-
pair-separator'])               -- insert '<space>and<space>' between two
language names
        elseif 2 < code then
                name = table.concat (language_list, cfg.messages['parameter-
separator'], 1, code-1);                        -- concatenate all but last
                name = table.concat ({name, language_list[code]},
```

```
cfg.messages['parameter-final-separator']);        -- concatenate last with
final separator
        end
        if this_wiki_name == name then
                return '';
-- if one language and that language is this wiki's return an empty string
(no annotation)
        end
        return (" " .. wrap_msg ('language', name));
-- otherwise wrap with '(in ...)'
        --[[ TODO: should only return blank or name rather than full list
        so we can clean up the bunched parenthetical elements Language, Type,
Format
        ]]
end


--[[--------------------------< S E T _ C S 1 _ S T Y L E >-----------------
---------------------------------

Set style settings for CS1 citation templates. Returns separator and
postscript settings
At en.wiki, for cs1:
        ps gets:        '.'
        sep gets:        '.'

]]

local function set_cs1_style (ps)
        if not is_set (ps) then
-- unless explicitly set to something
                ps = cfg.presentation['ps_cs1'];
-- terminate the rendered citation
        end
        return cfg.presentation['sep_cs1'], ps;
-- element separator
end


--[[--------------------------< S E T _ C S 2 _ S T Y L E >-----------------
---------------------------------

Set style settings for CS2 citation templates. Returns separator, postscript,
ref settings
At en.wiki, for cs2:
        ps gets:        '' (empty string - no terminal punctuation)
        sep gets:        ','

]]

local function set_cs2_style (ps, ref)
```

```
        if not is_set (ps) then
-- if |postscript= has not been set, set cs2 default
                ps = cfg.presentation['ps_cs2'];
-- terminate the rendered citation
        end
        if not is_set (ref) then
-- if |ref= is not set
                ref = "harv";
-- set default |ref=harv
        end
        return cfg.presentation['sep_cs2'], ps, ref;
-- element separator
end


--[[--------------------------< G E T _ S E T T I N G S _ F R O M _ C I T E _
C L A S S >----------------------

When |mode= is not set or when its value is invalid, use config.CitationClass
and parameter values to establish
rendered style.

]]

local function get_settings_from_cite_class (ps, ref, cite_class)
        local sep;
        if (cite_class == "citation") then
-- for citation templates (CS2)
                sep, ps, ref = set_cs2_style (ps, ref);
        else
-- not a citation template so CS1
                sep, ps = set_cs1_style (ps);
        end

        return sep, ps, ref
-- return them all
end


--[[--------------------------< S E T _ S T Y L E >--------------------------
----------------------------------

Establish basic style settings to be used when rendering the citation.  Uses
|mode= if set and valid or uses
config.CitationClass from the template's #invoke: to establish style.

]]

local function set_style (mode, ps, ref, cite_class)
        local sep;
        if 'cs2' == mode then
```

```
-- if this template is to be rendered in CS2 (citation) style
                sep, ps, ref = set_cs2_style (ps, ref);
        elseif 'cs1' == mode then
-- if this template is to be rendered in CS1 (cite xxx) style
                sep, ps = set_cs1_style (ps);
        else
-- anything but cs1 or cs2
                sep, ps, ref = get_settings_from_cite_class (ps, ref,
cite_class);                -- get settings based on the template's
CitationClass
        end

        if cfg.keywords_xlate[ps:lower()] == 'none' then
-- if assigned value is 'none' then
                ps = '';
-- set to empty string
        end
        return sep, ps, ref
end


--[=[-------------------------< I S _ P D F >--------------------------------
----------------------------------

Determines if a url has the file extension that is one of the pdf file
extensions used by [[MediaWiki:Common.css]] when
applying the pdf icon to external links.

returns true if file extension is one of the recognized extensions, else
false

]=]

local function is_pdf (url)
        return url:match ('%.pdf$') or url:match ('%.PDF$') or
                url:match ('%.pdf[%?#]') or url:match ('%.PDF[%?#]') or
                url:match ('%.PDF&#035') or url:match ('%.pdf&#035');
end


--[[-------------------------< S T Y L E _ F O R M A T >--------------------
----------------------------------

Applies css style to |format=, |chapter-format=, etc.  Also emits an error
message if the format parameter does
not have a matching url parameter.  If the format parameter is not set and
the url contains a file extension that
is recognized as a pdf document by MediaWiki's commons.css, this code will
set the format parameter to (PDF) with
the appropriate styling.
```

```
]]

local function style_format (format, url, fmt_param, url_param)
	if is_set (format) then
		format = wrap_style ('format', format);
-- add leading space, parentheses, resize
		if not is_set (url) then
			format = format .. set_error( 'format_missing_url',
{fmt_param, url_param} );	-- add an error message
		end
	elseif is_pdf (url) then
-- format is not set so if url is a pdf file then
		format = wrap_style ('format', 'PDF');
-- set format to pdf
	else
		format = '';
-- empty string for concatenation
	end
	return format;
end


--[[--------------------------< G E T _ D I S P L A Y _ N A M E S >----------
----------------------------------

Returns a number that defines the number of names displayed for author and
editor name lists and a boolean flag
to indicate when et al. should be appended to the name list.

When the value assigned to |display-xxxxors= is a number greater than or
equal to zero, return the number and
the previous state of the 'etal' flag (false by default but may have been set
to true if the name list contains
some variant of the text 'et al.').

When the value assigned to |display-xxxxors= is the keyword 'etal', return a
number that is one greater than the
number of authors in the list and set the 'etal' flag true.  This will cause
the list_people() to display all of
the names in the name list followed by 'et al.'

In all other cases, returns nil and the previous state of the 'etal' flag.

inputs:
	max: A['DisplayAuthors'] or A['DisplayEditors']; a number or some
flavor of etal
	count: #a or #e
	list_name: 'authors' or 'editors'
	etal: author_etal or editor_etal

]]
```

```
local function get_display_names (max, count, list_name, etal)
        if is_set (max) then
                if 'etal' == max:lower():gsub("[ '%.]", '') then
-- the :gsub() portion makes 'etal' from a variety of 'et al.' spellings and
stylings
                        max = count + 1;
-- number of authors + 1 so display all author name plus et al.
                        etal = true;
-- overrides value set by extract_names()
                elseif max:match ('^%d+$') then
-- if is a string of numbers
                        max = tonumber (max);
-- make it a number
                        if max >= count then
-- if |display-xxxxors= value greater than or equal to number of
authors/editors
                                add_maint_cat ('disp_name',
cfg.special_case_translation [list_name]);
                        end
                else
-- not a valid keyword or number
                        table.insert( z.message_tail, { set_error(
'invalid_param_val', {'display-' .. list_name, max}, true ) } );
-- add error message
                        max = nil;
-- unset; as if |display-xxxxors= had not been set
                end
        end
        return max, etal;
end


--[[--------------------------< E X T R A _ T E X T _ I N _ P A G E _ C H E C
K >----------------------------

Adds page to Category:CS1 maint: extra text if |page= or |pages= has what
appears to be some form of p. or pp.
abbreviation in the first characters of the parameter content.

check Page and Pages for extraneous p, p., pp, and pp. at start of parameter
value:
        good pattern: '^P[^%.P%l]' matches when |page(s)= begins PX or P# but
not Px where x and X are letters and # is a dgiit
        bad pattern: '^[Pp][Pp]' matches matches when |page(s)= begins pp or
pP or Pp or PP

]]

local function extra_text_in_page_check (page)
        local good_pattern = '^P[^%.Pp]';
-- ok to begin with uppercase P: P7 (pg 7 of section P) but not p123 (page
```

```
123) TODO: add Gg for PG or Pg?
        local bad_pattern = '^[Pp]?[Pp]%.?[ %d]';

        if not page:match (good_pattern) and (page:match (bad_pattern) or
page:match ('^[Pp]ages?')) then
                add_maint_cat ('extra_text');
        end
end


--[=[-------------------------< G E T _ V _ N A M E _ T A B L E >------------
---------------------------------

split apart a |vauthors= or |veditors= parameter.  This function allows for
corporate names, wrapped in doubled
parentheses to also have commas; in the old version of the code, the doubled
parnetheses were included in the
rendered citation and in the metadata.  Individual author names may be
wikilinked

        |vauthors=Jones AB, [[E. B. White|White EB]], ((Black, Brown, and
Co.))

]=]

local function get_v_name_table (vparam, output_table, output_link_table)
        local name_table = mw.text.split(vparam, "%s*,%s*");
-- names are separated by commas
        local wl_type, label, link;
-- wl_type not used here; just a place holder
        local i = 1;
        while name_table[i] do
                if name_table[i]:match ('^%(%(.*[^%)][^%)]$') then
-- first segment of corporate with one or more commas; this segment has the
opening doubled parens
                        local name = name_table[i];
                        i=i+1;
-- bump indexer to next segment
                        while name_table[i] do
                                name = name .. ', ' .. name_table[i];
-- concatenate with previous segments
                                if name_table[i]:match ('^.*%)%)$') then
-- if this table member has the closing doubled parens
                                        break;
-- and done reassembling so
                                end
                                i=i+1;
-- bump indexer
                        end
                        table.insert (output_table, name);
-- and add corporate name to the output table
```

```
                        table.insert (output_link_table, '');
-- no wikilink
                else
                        wl_type, label, link = is_wikilink (name_table[i]);
-- wl_type is: 0, no wl (text in label variable); 1, [[D]]; 2, [[L|D]]
                        table.insert (output_table, label);
-- add this name
                        if 1 == wl_type then
                                table.insert (output_link_table, label);
-- simple wikilink [[D]]
                        else
                                table.insert (output_link_table, link);
-- no wikilink or [[L|D]]; add this link if there is one, else empty string
                        end
                end
                i = i+1;
        end
        return output_table;
end


--[[--------------------------< P A R S E _ V A U T H O R S _ V E D I T O R S >------------------------------

This function extracts author / editor names from |vauthors= or |veditors=
and finds matching |xxxxor-maskn= and
|xxxxor-linkn= in args.  It then returns a table of assembled names just as
extract_names() does.

Author / editor names in |vauthors= or |veditors= must be in Vancouver system
style. Corporate or institutional names
may sometimes be required and because such names will often fail the
is_good_vanc_name() and other format compliance
tests, are wrapped in doubled paranethese ((corporate name)) to suppress the
format tests.

Supports generational suffixes Jr, 2nd, 3rd, 4th–6th.

This function sets the vancouver error when a reqired comma is missing and
when there is a space between an author's initials.

]]

local function parse_vauthors_veditors (args, vparam, list_name)
        local names = {};
-- table of names assembled from |vauthors=, |author-maskn=, |author-linkn=
        local v_name_table = {};
        local v_link_table = {};
-- when name is wikilinked, targets go in this table
        local etal = false;
-- return value set to true when we find some form of et al. vauthors
```

```
parameter
        local last, first, link, mask, suffix;
        local corporate = false;

        vparam, etal = name_has_etal (vparam, etal, true);
-- find and remove variations on et al. do not categorize (do it here because
et al. might have a period)
        v_name_table = get_v_name_table (vparam, v_name_table, v_link_table);
-- names are separated by commas

        for i, v_name in ipairs(v_name_table) do
                first = '';
-- set to empty string for concatenation and because it may have been set for
previous author/editor
                if v_name:match ('^%(%(.+%)%)$') then
-- corporate authors are wrapped in doubled parentheses to supress vanc
formatting and error detection
                        last = v_name:match ('^%(%((.+)%)%)$')
-- remove doubled parntheses
                        corporate = true;
-- flag used in list_people()
                elseif string.find(v_name, "%s") then
                        if v_name:find('[;%.]') then
-- look for commonly occurring punctuation characters;
                                add_vanc_error
(cfg.err_msg_supl.punctuation);
                        end
                        local lastfirstTable = {}
                        lastfirstTable = mw.text.split(v_name, "%s+")
                        first = table.remove(lastfirstTable);
-- removes and returns value of last element in table which should be intials
or generational suffix

                        if not mw.ustring.match (first, '^%u+$') then
-- mw.ustring here so that later we will catch non-latin characters
                                suffix = first;
-- not initials so assume that whatever we got is a generational suffix
                                first = table.remove(lastfirstTable);
-- get what should be the initials from the table
                        end
                        last = table.concat(lastfirstTable, ' ')
-- returns a string that is the concatenation of all other names that are not
initials and generational suffix
                        if not is_set (last) then
                                first = '';
-- unset
                                last = v_name;
-- last empty because something wrong with first
                                add_vanc_error (cfg.err_msg_supl.name);
                        end
                        if mw.ustring.match (last, '%a+%s+%u+%s+%a+') then
```

```
                                add_vanc_error (cfg.err_msg_supl['missing
comma']);                               -- matches last II last; the case
when a comma is missing
                        end
                        if mw.ustring.match (v_name, ' %u %u$') then
-- this test is in the wrong place TODO: move or replace with a more
appropriate test
                                add_vanc_error (cfg.err_msg_supl.name);
-- matches a space between two intiials
                        end
                else
                        last = v_name;
-- last name or single corporate name?  Doesn't support multiword corporate
names? do we need this?
                end
                if is_set (first) then
                        if not mw.ustring.match (first, "^%u?%u$") then
-- first shall contain one or two upper-case letters, nothing else
                                add_vanc_error (cfg.err_msg_supl.initials);
-- too many initials; mixed case initials (which may be ok Romanization);
hyphenated initials
                        end
                        is_good_vanc_name (last, first, suffix);
-- check first and last before restoring the suffix which may have a non-
Latin digit
                        if is_set (suffix) then
                                first = first .. ' ' .. suffix;
-- if there was a suffix concatenate with the initials
                                suffix = '';
-- unset so we don't add this suffix to all subsequent names
                        end
                else
                        if not corporate then
                                is_good_vanc_name (last, '');
                        end
                end

                link = select_one( args, cfg.aliases[list_name .. '-Link'],
'redundant_parameters', i ) or v_link_table[i];
                mask = select_one( args, cfg.aliases[list_name .. '-Mask'],
'redundant_parameters', i );
                names[i] = {last = last, first = first, link = link, mask =
mask, corporate=corporate};                -- add this assembled name to our
names list
        end
        return names, etal;
-- all done, return our list of names
end


--[[-------------------------< S E L E C T _ A U T H O R _ E D I T O R _ S O
```

```
U R C E >-----------------------

Select one of |authors=, |authorn= / |lastn / firstn=, or |vauthors= as the
source of the author name list or
select one of |editors=, |editorn= / editor-lastn= / |editor-firstn= or
|veditors= as the source of the editor name list.

Only one of these appropriate three will be used.  The hierarchy is:
|authorn= (and aliases) highest and |authors= lowest and
similarly, |editorn= (and aliases) highest and |editors= lowest

When looking for |authorn= / |editorn= parameters, test |xxxxor1= and
|xxxxor2= (and all of their aliases); stops after the second
test which mimicks the test used in extract_names() when looking for a hole
in the author name list.  There may be a better
way to do this, I just haven't discovered what that way is.

Emits an error message when more than one xxxxor name source is provided.

In this function, vxxxxors = vauthors or veditors; xxxxors = authors or
editors as appropriate.

]]

local function select_author_editor_source (vxxxxors, xxxxors, args,
list_name)
        local lastfirst = false;
        if select_one( args, cfg.aliases[list_name .. '-Last'], 'none', 1 )
or                -- do this twice incase we have a |first1= without a
|last1=; this ...
                select_one( args, cfg.aliases[list_name .. '-First'], 'none',
1 ) or        -- ... also catches the case where |first= is used with
|vauthors=
                select_one( args, cfg.aliases[list_name .. '-Last'], 'none',
2 ) or
                select_one( args, cfg.aliases[list_name .. '-First'], 'none',
2 ) then
                        lastfirst=true;
        end

        if (is_set (vxxxxors) and true == lastfirst) or
-- these are the three error conditions
                (is_set (vxxxxors) and is_set (xxxxors)) or
                (true == lastfirst and is_set (xxxxors)) then
                        local err_name;
                        if 'AuthorList' == list_name then
-- figure out which name should be used in error message
                                err_name = 'author';
                        else
                                err_name = 'editor';
                        end
```

```
                        table.insert( z.message_tail, { set_error(
'redundant_parameters',
                                {err_name .. '-name-list parameters'}, true )
} );                                -- add error message
        end

        if true == lastfirst then return 1 end;
-- return a number indicating which author name source to use
        if is_set (vxxxxors) then return 2 end;
        if is_set (xxxxors) then return 3 end;
        return 1;
-- no authors so return 1; this allows missing author name test to run in
case there is a first without last
end


--[[--------------------------< I S _ V A L I D _ P A R A M E T E R _ V A L U
E >----------------------------

This function is used to validate a parameter's assigned value for those
parameters that have only a limited number
of allowable values (yes, y, true, live, dead, etc).  When the parameter
value has not been assigned a value (missing
or empty in the source template) the function returns the value specified by
ret_val.  If the parameter value is one
of the list of allowed values returns the translated value; else, emits an
error message and returns the value
specified by ret_val.

]]

local function is_valid_parameter_value (value, name, possible, ret_val)
        if not is_set (value) then
                return ret_val;
-- an empty parameter is ok
        elseif in_array (value, possible) then
                return cfg.keywords_xlate[value];
-- return translation of parameter keyword
        else
                table.insert( z.message_tail, { set_error(
'invalid_param_val', {name, value}, true ) } );        -- not an allowed
value so add error message
                return ret_val;
        end
end


--[[--------------------------< T E R M I N A T E _ N A M E _ L I S T >------
---------------------------------

This function terminates a name list (author, contributor, editor) with a
```

```
separator character (sepc) and a space
when the last character is not a sepc character or when the last three
characters are not sepc followed by two
closing square brackets (close of a wikilink).  When either of these is true,
the name_list is terminated with a
single space character.

]]

local function terminate_name_list (name_list, sepc)
        if (string.sub (name_list,-3,-1) == sepc .. '. ') then
-- if already properly terminated
                return name_list;
-- just return the name list
        elseif (string.sub (name_list,-1,-1) == sepc) or (string.sub
(name_list,-3,-1) == sepc .. ']]') then        -- if last name in list ends
with sepc char
                return name_list .. " ";
-- don't add another
        else
                return name_list .. sepc .. ' ';
-- otherwise terninate the name list
        end
end


--[[-------------------------< F O R M A T _ V O L U M E _ I S S U E >-------
---------------------------------

returns the concatenation of the formatted volume and issue parameters as a
single string; or formatted volume
or formatted issue, or an empty string if neither are set.

]]
local function format_volume_issue (volume, issue, cite_class, origin, sepc,
lower)
        if not is_set (volume) and not is_set (issue) then
                return '';
        end
        if 'magazine' == cite_class or (in_array (cite_class, {'citation',
'map'}) and 'magazine' == origin) then
                if is_set (volume) and is_set (issue) then
                        return wrap_msg ('vol-no', {sepc, volume, issue},
lower);
                elseif is_set (volume) then
                        return wrap_msg ('vol', {sepc, volume}, lower);
                else
                        return wrap_msg ('issue', {sepc, issue}, lower);
                end
        end
```

```
        if 'podcast' == cite_class and is_set (issue) then
                return wrap_msg ('issue', {sepc, issue}, lower);
        end

        local vol = '';
-- here for all cites except magazine
        if is_set (volume) then
                if volume:match ('^[MDCLXVI]+$') or volume:match
('^%d+$')then                           -- volume value is all digits or all
uppercase roman numerals
                        vol = substitute (cfg.presentation['vol-bold'],
{sepc, hyphen_to_dash(volume)});        -- render in bold face
                elseif (4 < mw.ustring.len(volume)) then
-- not all digits or roman numerals and longer than 4 characters
                        vol = substitute (cfg.messages['j-vol'], {sepc,
volume});                       -- not bold
                        add_prop_cat ('long_vol');
                else
-- four or less characters
                        vol = substitute (cfg.presentation['vol-bold'],
{sepc, hyphen_to_dash(volume)});        -- bold
                end
        end
        if is_set (issue) then
                return vol .. substitute (cfg.messages['j-issue'], issue);
        end
        return vol;
end


--[[-------------------------< F O R M A T _ P A G E S _ S H E E T S >-------
----------------------------------

adds static text to one of |page(s)= or |sheet(s)= values and returns it with
all of the others set to empty strings.
The return order is:
        page, pages, sheet, sheets

Singular has priority over plural when both are provided.

]]

local function format_pages_sheets (page, pages, sheet, sheets, cite_class,
origin, sepc, nopp, lower)
        if 'map' == cite_class then
-- only cite map supports sheet(s) as in-source locators
                if is_set (sheet) then
                        if 'journal' == origin then
                                return '', '', wrap_msg ('j-sheet', sheet,
lower), '';
                        else
```

```
                                return '', '', wrap_msg ('sheet', {sepc,
sheet}, lower), '';
                        end
                elseif is_set (sheets) then
                        if 'journal' == origin then
                                return '', '', '', wrap_msg ('j-sheets',
sheets, lower);
                        else
                                return '', '', '', wrap_msg ('sheets', {sepc,
sheets}, lower);
                        end
                end
        end

        local is_journal = 'journal' == cite_class or (in_array (cite_class,
{'citation', 'map', 'interview'}) and 'journal' == origin);
        if is_set (page) then
                if is_journal then
                        return substitute (cfg.messages['j-page(s)'], page),
'', '', '';
                elseif not nopp then
                        return substitute (cfg.messages['p-prefix'], {sepc,
page}), '', '', '';
                else
                        return substitute (cfg.messages['nopp'], {sepc,
page}), '', '', '';
                end
        elseif is_set(pages) then
                if is_journal then
                        return substitute (cfg.messages['j-page(s)'], pages),
'', '', '';
                elseif tonumber(pages) ~= nil and not nopp then
-- if pages is only digits, assume a single page number
                        return '', substitute (cfg.messages['p-prefix'],
{sepc, pages}), '', '';
                elseif not nopp then
                        return '', substitute (cfg.messages['pp-prefix'],
{sepc, pages}), '', '';
                else
                        return '', substitute (cfg.messages['nopp'], {sepc,
pages}), '', '';
                end
        end
        return '', '', '', '';
-- return empty strings
end


--[[--------------------------< I N S O U R C E _ L O C _ G E T >------------
---------------------------------
```

```
returns one of the in-source locators: page, pages, or at.

If any of these are interwiki links to wikisource, returns the label portion
of the interwikilink as plain text
for use in COinS.  This COinS thing is done because here we convert an
interwiki link to and external link and
add an icon span around that; get_coins_pages() doesn't know about the span.
TODO: should it?

TODO: add support for sheet and sheets?; streamline;

TODO: make it so that this function returns only one of the three as the
single in-source (the return value assigned
to a new name)?

]]

local function insource_loc_get (page, pages, at)
	local ws_url, ws_label, coins_pages, L;
-- for wikisource interwikilinks; TODO: this corrupts page metadata (span
remains in place after cleanup; fix there?)

	if is_set (page) then
		if is_set (pages) or is_set(at) then
			pages = '';
-- unset the others
			at = '';
		end
		extra_text_in_page_check (page);
-- add this page to maint cat if |page= value begins with what looks like p.
or pp.

		ws_url, ws_label, L = wikisource_url_make (page);
-- make ws url from |page= interwiki link; link portion L becomes tool tip
label
		if ws_url then
			page = external_link (ws_url, ws_label .. ' ',
'ws link in page');	-- space char after label to move icon away from
in-source text; TODO: a better way to do this?
			page = substitute (cfg.presentation['interwiki-
icon'], {cfg.presentation['class-wikisource'], L, page});
			coins_pages = ws_label;
		end
	elseif is_set (pages) then
		if is_set (at) then
			at = '';
-- unset
		end
		extra_text_in_page_check (pages);
-- add this page to maint cat if |pages= value begins with what looks like p.
or pp.
```

```
                ws_url, ws_label, L = wikisource_url_make (pages);
-- make ws url from |pages= interwiki link; link portion L becomes tool tip
label
                if ws_url then
                        pages = external_link (ws_url, ws_label .. ' ',
'ws link in pages');        -- space char after label to move icon away from
in-source text; TODO: a better way to do this?
                        pages = substitute (cfg.presentation['interwiki-
icon'], {cfg.presentation['class-wikisource'], L, pages});
                        coins_pages = ws_label;
                end
        elseif is_set (at) then
                ws_url, ws_label, L = wikisource_url_make (at);
-- make ws url from |at= interwiki link; link portion L becomes tool tip
label
                if ws_url then
                        at = external_link (ws_url, ws_label .. ' ', 'ws
link in at');        -- space char after label to move icon away from in-
source text; TODO: a better way to do this?
                        at = substitute (cfg.presentation['interwiki-icon'],
{cfg.presentation['class-wikisource'], L, at});
                        coins_pages = ws_label;
                end
        end
        return page, pages, at, coins_pages;
end


--[=[-------------------------< A R C H I V E _ U R L _ C H E C K >----------
---------------------------------
```

Check archive.org urls to make sure they at least look like they are pointing at valid archives and not to the
save snapshot url or to calendar pages. When the archive url is 'https://web.archive.org/save/' (or http://...)
archive.org saves a snapshot of the target page in the url. That is something that Wikipedia should not allow
unwitting readers to do.

When the archive.org url does not have a complete timestamp, archive.org chooses a snapshot according to its own
algorithm or provides a calendar 'search' result. [[WP:ELNO]] discourages links to search results.

This function looks at the value assigned to |archive-url= and returns empty strings for |archive-url= and
|archive-date= and an error message when:
        |archive-url= holds an archive.org save command url
        |archive-url= is an archive.org url that does not have a complete timestamp (YYYYMMDDhhmmss 14 digits) in the
                correct place

```
otherwise returns |archive-url= and |archive-date=

There are two mostly compatible archive.org urls:
        //web.archive.org/<timestamp>...                -- the old form
        //web.archive.org/web/<timestamp>...        -- the new form

The old form does not support or map to the new form when it contains a
display flag.  There are four identified flags
('id_', 'js_', 'cs_', 'im_') but since archive.org ignores others following
the same form (two letters and an underscore)
we don't check for these specific flags but we do check the form.

This function supports a preview mode.  When the article is rendered in
preview mode, this funct may return a modified
archive url:
        for save command errors, return undated wildcard (/*/)
        for timestamp errors when the timestamp has a wildcard, return the
url unmodified
        for timestamp errors when the timestamp does not have a wildcard,
return with timestamp limited to six digits plus wildcard (/yyyymm*/)

]=]

local function archive_url_check (url, date)
        local err_msg = '';
-- start with the error message empty
        local path, timestamp, flag;
-- portions of the archive.or url
        if (not url:match('//web%.archive%.org/')) and (not
url:match('//liveweb%.archive%.org/')) then                -- also deprecated
liveweb Wayback machine url
                return url, date;
-- not an archive.org archive, return ArchiveURL and ArchiveDate
        end

        if url:match('//web%.archive%.org/save/') then
-- if a save command url, we don't want to allow saving of the target page
                err_msg = cfg.err_msg_supl.save;
                url = url:gsub ('(//web%.archive%.org)/save/', '%1/*/', 1);
-- for preview mode: modify ArchiveURL
        elseif url:match('//liveweb%.archive%.org/') then
                err_msg = cfg.err_msg_supl.liveweb;
        else
                path, timestamp, flag =
url:match('//web%.archive%.org/([^%d]*)(%d+)([^/]*)/');        -- split out
some of the url parts for evaluation
                if not is_set(timestamp) or 14 ~= timestamp:len() then
-- path and flag optional, must have 14-digit timestamp here
                        err_msg = cfg.err_msg_supl.timestamp;
                        if '*' ~= flag then
                                url=url:gsub
```

```
('(//web%.archive%.org/[^%d]*%d?%d?%d?%d?%d?%d?)[^/]*', '%1*', 1)        --
for preview, modify ts to be yearmo* max (0-6 digits plus splat)
                        end
                elseif is_set(path) and 'web/' ~= path then
-- older archive urls do not have the extra 'web/' path element
                        err_msg = cfg.err_msg_supl.path;
                elseif is_set (flag) and not is_set (path) then
-- flag not allowed with the old form url (without the 'web/' path element)
                        err_msg = cfg.err_msg_supl.flag;
                elseif is_set (flag) and not flag:match ('%a%a_') then
-- flag if present must be two alpha characters and underscore (requires
'web/' path element)
                        err_msg = cfg.err_msg_supl.flag;
                else
                        return url, date;
-- return ArchiveURL and ArchiveDate
                end
        end
-- if here, something not right so
        table.insert( z.message_tail, { set_error( 'archive_url', {err_msg},
true ) } );        -- add error message and
        if is_set (Frame:preprocess('{{REVISIONID}}')) then
                return '', '';
-- return empty strings for ArchiveURL and ArchiveDate
        else
                return url, date;
-- preview mode so return ArchiveURL and ArchiveDate
        end
end


--[[-------------------------< P L A C E _ C H E C K >----------------------
----------------------------------

check |place=, |publication-place=, |location= to see if these params include
digits.  This function added because
many editors mis-use location to specify the in-source location (|page(s)=
and |at= are supposed to do that)

returns the original parameter value without modification; added maint cat
when parameter value contains digits

]]

local function place_check (param_val)
        if not is_set (param_val) then
-- parameter empty or omitted
                return param_val;
-- return that empty state
        end
        if mw.ustring.find (param_val, '%d') then
```

```
-- not empty, are there digits in the parameter value
                add_maint_cat ('location');
-- yep, add maint cat
        end
        return param_val;
-- and done
end


--[[-------------------------< C I T A T I O N 0 >-------------------------
---------------------------------

This is the main function doing the majority of the citation formatting.

]]

local function citation0( config, args)
        --[[
        Load Input Parameters
        The argument_wrapper facilitates the mapping of multiple aliases to
single internal variable.
        ]]
        local A = argument_wrapper( args );
        local i

        -- Pick out the relevant fields from the arguments.  Different
citation templates
        -- define different field names for the same underlying things.

        local Mode = is_valid_parameter_value (A['Mode'], A:ORIGIN('Mode'),
cfg.keywords_lists['mode'], '');

        local author_etal;
        local a        = {};
-- authors list from |lastn= / |firstn= pairs or |vauthors=
        local Authors;

        local NameListFormat = is_valid_parameter_value (A['NameListFormat'],
A:ORIGIN('NameListFormat'), cfg.keywords_lists['name-list-format'], '');
        local Collaboration = A['Collaboration'];

        do
-- to limit scope of selected
                local selected = select_author_editor_source (A['Vauthors'],
A['Authors'], args, 'AuthorList');
                if 1 == selected then
                        a, author_etal = extract_names (args, 'AuthorList');
-- fetch author list from |authorn= / |lastn= / |firstn=, |author-linkn=, and
|author-maskn=
                elseif 2 == selected then
                        NameListFormat = 'vanc';
```

```
-- override whatever |name-list-format= might be
                        a, author_etal = parse_vauthors_veditors (args,
args.vauthors, 'AuthorList');        -- fetch author list from |vauthors=,
|author-linkn=, and |author-maskn=
                elseif 3 == selected then
                        Authors = A['Authors'];
-- use content of |authors=
                        if 'authors' == A:ORIGIN('Authors') then
-- but add a maint cat if the parameter is |authors=
                                add_maint_cat ('authors');
-- because use of this parameter is discouraged; what to do about the aliases
is a TODO:
                        end
                end
                if is_set (Collaboration) then
                        author_etal = true;
-- so that |display-authors=etal not required
                end
        end

        local Others = A['Others'];

        local editor_etal;
        local e        = {};
-- editors list from |editor-lastn= / |editor-firstn= pairs or |veditors=
        local Editors;

        do
-- to limit scope of selected
                local selected = select_author_editor_source (A['Veditors'],
A['Editors'], args, 'EditorList');
                if 1 == selected then
                        e, editor_etal = extract_names (args, 'EditorList');
-- fetch editor list from |editorn= / |editor-lastn= / |editor-firstn=,
|editor-linkn=, and |editor-maskn=
                elseif 2 == selected then
                        NameListFormat = 'vanc';
-- override whatever |name-list-format= might be
                        e, editor_etal = parse_vauthors_veditors (args,
args.veditors, 'EditorList');        -- fetch editor list from |veditors=,
|editor-linkn=, and |editor-maskn=
                elseif 3 == selected then
                        Editors = A['Editors'];
-- use content of |editors=
                        add_maint_cat ('editors');
-- but add a maint cat because use of this parameter is discouraged
                end
        end

        local translator_etal;
        local t = {};
```

```
-- translators list from |translator-lastn= / translator-firstn= pairs
        local Translators;
-- assembled translators name list
        t = extract_names (args, 'TranslatorList');
-- fetch translator list from |translatorn= / |translator-lastn=, -firstn=, -
linkn=, -maskn=

        local interviewer_etal;
        local interviewers_list = {};
        local Interviewers;
-- used later
        interviewers_list = extract_names (args, 'InterviewerList');
-- process preferred interviewers parameters

        local contributor_etal;
        local c = {};
-- contributors list from |contributor-lastn= / contributor-firstn= pairs
        local Contributors;
-- assembled contributors name list

        local Chapter = A['Chapter'];
-- done here so that we have access to |contribution= from |chapter= aliases
        local Chapter_origin = A:ORIGIN ('Chapter');
        local Contribution;
-- because contribution is required for contributor(s)
                if 'contribution' == A:ORIGIN ('Chapter') then
                        Contribution = A['Chapter'];
-- get the name of the contribution
                end

        if in_array(config.CitationClass, {"book","citation"}) and not
is_set(A['Periodical']) then        -- |contributor= and |contribution= only
supported in book cites
                c = extract_names (args, 'ContributorList');
-- fetch contributor list from |contributorn= / |contributor-lastn=, -
firstn=, -linkn=, -maskn=
                if 0 < #c then
                        if not is_set (Contribution) then
-- |contributor= requires |contribution=
                                table.insert( z.message_tail, { set_error(
'contributor_missing_required_param', 'contribution')});        -- add
missing contribution error message
                                c = {};
-- blank the contributors' table; it is used as a flag later
                        end
                        if 0 == #a then
-- |contributor= requires |author=
                                table.insert( z.message_tail, { set_error(
'contributor_missing_required_param', 'author')});        -- add missing
author error message
                                c = {};
```

```
-- blank the contributors' table; it is used as a flag later
                        end
                end
        else
-- if not a book cite
                if select_one (args, cfg.aliases['ContributorList-Last'],
'redundant_parameters', 1 ) then        -- are there contributor name list
parameters?
                        table.insert( z.message_tail, { set_error(
'contributor_ignored')});        -- add contributor ignored error message
                end
                Contribution = nil;
-- unset
        end

        if is_set (Others) then
                if 0 == #a and 0 == #e then
-- add maint cat when |others= has value and used without |author=, |editor=
                        add_maint_cat ('others');
                end
        end

        local Year = A['Year'];
        local PublicationDate = A['PublicationDate'];
        local OrigYear = A['OrigYear'];
        local Date = A['Date'];
        local LayDate = A['LayDate'];
        ------------------------------------------------- Get title data
        local Title = A['Title'];
        local ScriptTitle = A['ScriptTitle'];
        local BookTitle = A['BookTitle'];
        local Conference = A['Conference'];
        local TransTitle = A['TransTitle'];
        local TransTitle_origin = A:ORIGIN ('TransTitle');
        local TitleNote = A['TitleNote'];
        local TitleLink = A['TitleLink'];
                        link_title_ok (TitleLink, A:ORIGIN ('TitleLink'),
Title, 'title');        -- check for wikimarkup in |title-link= or wikimarkup
in |title= when |title-link= is set

        local Section = '';
-- {{cite map}} only; preset to empty string for concatnation if not used
                if 'map' == config.CitationClass and 'section' == A:ORIGIN
('Chapter') then
                        Section = A['Chapter'];
-- get |section= from |chapter= alias list; |chapter= and the other aliases
not supported in {{cite map}}
                        Chapter = '';
-- unset for now; will be reset later from |map= if present
                end
```

```
        local ScriptChapter = A['ScriptChapter'];
        local ScriptChapter_origin = A:ORIGIN ('ScriptChapter');
        local ChapterLink        -- = A['ChapterLink'];
-- deprecated as a parameter but still used internally by cite episode
        local TransChapter = A['TransChapter'];
        local TransChapter_origin = A:ORIGIN ('TransChapter');
        local TitleType = A['TitleType'];
        local Degree = A['Degree'];
        local Docket = A['Docket'];
        local ArchiveFormat = A['ArchiveFormat'];

        local ArchiveDate;
        local ArchiveURL;

        ArchiveURL, ArchiveDate = archive_url_check (A['ArchiveURL'],
A['ArchiveDate'])
        local UrlStatus = is_valid_parameter_value (A['UrlStatus'],
A:ORIGIN('UrlStatus'), cfg.keywords_lists['url-status'], '');


        local URL = A['URL']
        local URL_origin = A:ORIGIN('URL');
-- get name of parameter that holds URL
        local ChapterURL = A['ChapterURL'];
        local ChapterURL_origin = A:ORIGIN('ChapterURL');
-- get name of parameter that holds ChapterURL
        local ConferenceFormat = A['ConferenceFormat'];
        local ConferenceURL = A['ConferenceURL'];
        local ConferenceURL_origin = A:ORIGIN('ConferenceURL');
-- get name of parameter that holds ConferenceURL

        local Periodical = A['Periodical'];
        local Periodical_origin = '';
                if is_set (Periodical) then
                        Periodical_origin = A:ORIGIN('Periodical');
-- get the name of the periodical parameter
                        local i;
                        Periodical, i = strip_apostrophe_markup (Periodical);
-- strip appostrophe markup so that metadata isn't contaminated
                        if i then
-- non-zero when markup was stripped so emit an error message
                                table.insert( z.message_tail, {set_error
('apostrophe_markup', {Periodical_origin}, true)});
                        end
                end

        if 'mailinglist' == config.CitationClass then
-- special case for {{cite mailing list}}
                if is_set (Periodical) and is_set (A ['MailingList']) then
-- both set emit an error
                        table.insert( z.message_tail, {
```

```
set_error('redundant_parameters', {wrap_style ('parameter',
Periodical_origin) .. ' and ' .. wrap_style ('parameter', 'mailinglist')},
true )});
                end

                Periodical = A ['MailingList'];
-- error or no, set Periodical to |mailinglist= value because this template
is {{cite mailing list}}
                Periodical_origin = A:ORIGIN('MailingList');
        end



        local ScriptPeriodical = A['ScriptPeriodical'];
        local ScriptPeriodical_origin = A:ORIGIN('ScriptPeriodical');
-- web and news not tested for now because of
-- Wikipedia:Administrators%27_noticeboard#Is_there_a_semi-
automated_tool_that_could_fix_these_annoying_"Cite_Web"_errors?
                if not (is_set (Periodical) or is_set (ScriptPeriodical))
then                            -- 'periodical' templates require periodical
parameter
        --                local p = {['journal'] = 'journal', ['magazine'] =
'magazine', ['news'] = 'newspaper', ['web'] = 'website'};        -- for error
message
                        local p = {['journal'] = 'journal', ['magazine'] =
'magazine'};                -- for error message
                        if p[config.CitationClass]  then
                                table.insert( z.message_tail, {set_error
('missing_periodical', {config.CitationClass, p[config.CitationClass]},
true)});
                        end
                end

        local TransPeriodical =  A['TransPeriodical'];
        local TransPeriodical_origin =  A:ORIGIN ('TransPeriodical');

        local Series = A['Series'];
        local Volume;
        local Issue;
        local Page;
        local Pages;
        local At;

        if 'citation' == config.CitationClass then
                if is_set (Periodical) then
                        if not in_array (Periodical_origin, {'website',
'mailinglist'}) then        -- {{citation}} does not render volume for these
'periodicals'
                                Volume = A['Volume'];
-- but does for all other 'periodicals'
                        end
```

```
                elseif is_set (ScriptPeriodical) then
                        if 'script-website' ~= ScriptPeriodical_origin then
-- {{citation}} does not render volume for |script-website=
                                Volume = A['Volume'];
-- but does for all other 'periodicals'
                        end
                else
                        Volume = A['Volume'];
-- and does for non-'periodical' cites
                end
        elseif in_array (config.CitationClass, cfg.templates_using_volume)
then                -- render |volume= for cs1 according to the configuration
settings
                Volume = A['Volume'];
        end

        if 'citation' == config.CitationClass then
                if is_set (Periodical) and in_array (Periodical_origin,
{'journal', 'magazine', 'newspaper', 'periodical', 'work'}) or        --
{{citation}} renders issue for these 'periodicals'
                        is_set (ScriptPeriodical) and in_array
(ScriptPeriodical_origin, {'script-journal', 'script-magazine', 'script-
newspaper', 'script-periodical', 'script-work'}) then -- and these 'script-
periodicals'
                                Issue = hyphen_to_dash (A['Issue']);
                end
        elseif in_array (config.CitationClass, cfg.templates_using_issue)
then                -- conference & map books do not support issue;
{{citation}} listed here because included in settings table
                if not (in_array (config.CitationClass, {'conference', 'map',
'citation'}) and not (is_set (Periodical) or is_set (ScriptPeriodical))) then
                        Issue = hyphen_to_dash (A['Issue']);
                end
        end

        local Position = '';
        if not in_array (config.CitationClass, cfg.templates_not_using_page)
then
                Page = A['Page'];
                Pages = hyphen_to_dash (A['Pages']);
                At = A['At'];
        end

        local Edition = A['Edition'];
        local PublicationPlace = place_check (A['PublicationPlace'],
A:ORIGIN('PublicationPlace'));
        local Place = place_check (A['Place'], A:ORIGIN('Place'));
        local PublisherName = A['PublisherName'];
        local PublisherName_origin = A:ORIGIN('PublisherName');
                if is_set (PublisherName) then
                        local i=0;
```

```
                        PublisherName, i = strip_apostrophe_markup
(PublisherName);                        -- strip appostrophe markup so that
metadata isn't contaminated; publisher is never italicized

                        if i then
-- non-zero when markup was stripped so emit an error message
                                table.insert( z.message_tail, {set_error
('apostrophe_markup', {PublisherName_origin}, true)});
                        end
                end

        local Newsgroup = A['Newsgroup'];
-- TODO: strip apostrophe markup?
        local Newsgroup_origin = A:ORIGIN('Newsgroup');

        if 'newsgroup' == config.CitationClass then
                if is_set (PublisherName) then
-- general use parmeter |publisher= not allowed in cite newsgroup
                        local error_text = set_error ('parameter_ignored',
{PublisherName_origin}, true);
                        if is_set (error_text) then
                                table.insert( z.message_tail, {error_text,
error_state} );
                        end
                end

                PublisherName = nil;
-- ensure that this parameter is unset for the time being; will be used again
after COinS
        end

        local UrlAccess = is_valid_parameter_value (A['UrlAccess'],
A:ORIGIN('UrlAccess'), cfg.keywords_lists['url-access'], nil);
                if not is_set(URL) and is_set(UrlAccess) then
                        UrlAccess = nil;
                        table.insert( z.message_tail, { set_error(
'param_access_requires_param', {'url'}, true ) } );
                end
        local ChapterUrlAccess = is_valid_parameter_value
(A['ChapterUrlAccess'], A:ORIGIN('ChapterUrlAccess'),
cfg.keywords_lists['url-access'], nil);
                if not is_set(ChapterURL) and is_set(ChapterUrlAccess) then
                        ChapterUrlAccess = nil;
                        table.insert( z.message_tail, { set_error(
'param_access_requires_param', {A:ORIGIN('ChapterUrlAccess'):gsub ('%-
access', '')}, true ) } );
                end

        local MapUrlAccess = is_valid_parameter_value (A['MapUrlAccess'],
A:ORIGIN('MapUrlAccess'), cfg.keywords_lists['url-access'], nil);
                if not is_set(A['MapURL']) and is_set(MapUrlAccess) then
```

```
                        MapUrlAccess = nil;
                        table.insert( z.message_tail, { set_error(
'param_access_requires_param', {'map-url'}, true ) } );
                end

        local Via = A['Via'];
        local AccessDate = A['AccessDate'];
        local Agency = A['Agency'];

        local Language = A['Language'];
        local Format = A['Format'];
        local ChapterFormat = A['ChapterFormat'];
        local DoiBroken = A['DoiBroken'];
        local ID = A['ID'];
        local ASINTLD = A['ASINTLD'];
        local IgnoreISBN = is_valid_parameter_value (A['IgnoreISBN'],
A:ORIGIN('IgnoreISBN'), cfg.keywords_lists['yes_true_y'], nil);
        local Embargo = A['Embargo'];
        local Class = A['Class'];
-- arxiv class identifier

        local ID_list = extract_ids( args );
                if is_set (DoiBroken) and not ID_list['DOI'] then
                        table.insert( z.message_tail, { set_error(
'doibroken_missing_doi', A:ORIGIN('DoiBroken'))});
                end
        local ID_access_levels = extract_id_access_levels( args, ID_list );

        local Quote = A['Quote'];

        local LayFormat = A['LayFormat'];
        local LayURL = A['LayURL'];
        local LaySource = A['LaySource'];
        local Transcript = A['Transcript'];
        local TranscriptFormat = A['TranscriptFormat'];
        local TranscriptURL = A['TranscriptURL']
        local TranscriptURL_origin = A:ORIGIN('TranscriptURL');
-- get name of parameter that holds TranscriptURL

        local LastAuthorAmp = is_valid_parameter_value (A['LastAuthorAmp'],
A:ORIGIN('LastAuthorAmp'), cfg.keywords_lists['yes_true_y'], nil);

        local no_tracking_cats = is_valid_parameter_value (A['NoTracking'],
A:ORIGIN('NoTracking'), cfg.keywords_lists['yes_true_y'], nil);

        --local variables that are not cs1 parameters
        local use_lowercase;
-- controls capitalization of certain static text
        local this_page = mw.title.getCurrentTitle();
-- also used for COinS and for language
        local anchor_year;
```

```
-- used in the CITEREF identifier
        local COinS_date = {};
-- holds date info extracted from |date= for the COinS metadata by
Module:Date verification

        local DF = is_valid_parameter_value (A['DF'], A:ORIGIN('DF'),
cfg.keywords_lists['df'], '');
        if not is_set (DF) then
                DF = cfg.global_df;
-- local df if present overrides global df set by {{use xxx date}} template
        end

        local sepc;
-- separator between citation elements for CS1 a period, for CS2, a comma
        local PostScript;
        local Ref = A['Ref'];
        if 'harv' == Ref then
                add_maint_cat ('ref_harv');
-- add maint cat to identify templates that have this now-extraneous param
value
        elseif not is_set (Ref) then
                Ref = 'harv';
-- set as default when not set externally
        end
        sepc, PostScript, Ref = set_style (Mode:lower(), A['PostScript'],
Ref, config.CitationClass);
        use_lowercase = ( sepc == ',' );
-- used to control capitalization for certain static text

        --check this page to see if it is in one of the namespaces that cs1
is not supposed to add to the error categories
        if not is_set (no_tracking_cats) then
-- ignore if we are already not going to categorize this page
                if in_array (this_page.nsText, cfg.uncategorized_namespaces)
then
                        no_tracking_cats = "true";
-- set no_tracking_cats
                end
                for _,v in ipairs (cfg.uncategorized_subpages) do
-- cycle through page name patterns
                        if this_page.text:match (v) then
-- test page name against each pattern
                                no_tracking_cats = "true";
-- set no_tracking_cats
                                break;
-- bail out if one is found
                        end
                end
        end
-- check for extra |page=, |pages= or |at= parameters. (also sheet and sheets
while we're at it)
```

```
        select_one (args, {'page', 'p', 'pp', 'pages', 'at', 'sheet',
'sheets'}, 'redundant_parameters');        -- this is a dummy call simply to
get the error message and category

        local coins_pages;
        Page, Pages, At, coins_pages = insource_loc_get (Page, Pages, At);

        local NoPP = is_valid_parameter_value (A['NoPP'], A:ORIGIN('NoPP'),
cfg.keywords_lists['yes_true_y'], nil);

        if is_set (PublicationPlace) and is_set (Place) then
-- both |publication-place= and |place= (|location=) allowed if different
                add_prop_cat ('location test');
-- add property cat to evaluate how often PublicationPlace and Place are used
together
                if PublicationPlace == Place then
                        Place = '';
-- unset; don't need both if they are the same
                end
        elseif not is_set (PublicationPlace) and is_set (Place) then
-- when only |place= (|location=) is set ...
                PublicationPlace = Place;
-- promote |place= (|location=) to |publication-place
        end

        if PublicationPlace == Place then Place = ''; end
-- don't need both if they are the same
        --[[
        Parameter remapping for cite encyclopedia:
        When the citation has these parameters:
                |encyclopedia and |title then map |title to |article and
|encyclopedia to |title
                |encyclopedia and |article then map |encyclopedia to |title

                |trans-title maps to |trans-chapter when |title is re-mapped
                |url maps to |chapterurl when |title is remapped
        All other combinations of |encyclopedia, |title, and |article are not
modified
        ]]

        local Encyclopedia = A['Encyclopedia'];
-- used as a flag by this module and by ~/COinS

        if is_set (Encyclopedia) then
-- emit error message when Encyclopedia set but template is other than {{cite
encyclopedia}} or {{citation}}
                if 'encyclopaedia' ~= config.CitationClass and 'citation' ~=
config.CitationClass then
                        table.insert (z.message_tail, {set_error
('parameter_ignored', {A:ORIGIN ('Encyclopedia')}, true)});
                        Encyclopedia = nil;
```

```
-- unset because not supported by this template
                end
        end

        if ('encyclopaedia' == config.CitationClass) or ('citation' ==
config.CitationClass and is_set (Encyclopedia)) then
                if is_set (Periodical) and is_set (Encyclopedia) then
-- when both set emit an error
                        table.insert (z.message_tail,
{set_error('redundant_parameters', {wrap_style ('parameter', A:ORIGIN
('Encyclopedia')) .. ' and ' .. wrap_style ('parameter', Periodical_origin)},
true )});
                end

                if is_set (Encyclopedia) then
                        Periodical = Encyclopedia;
-- error or no, set Periodical to Encyclopedia; allow periodical without
encyclopedia
                        Periodical_origin = A:ORIGIN ('Encyclopedia');
                end

                if is_set (Periodical) then
-- Periodical is set when |encyclopedia is set
                        if is_set(Title) or is_set (ScriptTitle) then
                                if not is_set(Chapter) then
                                        Chapter = Title;
-- |encyclopedia and |title are set so map |title to |article and
|encyclopedia to |title
                                        ScriptChapter = ScriptTitle;
                                        ScriptChapter_origin =
A:ORIGIN('ScriptTitle')
                                        TransChapter = TransTitle;
                                        ChapterURL = URL;
                                        ChapterURL_origin = A:ORIGIN('URL')

                                        ChapterUrlAccess = UrlAccess;

                                        if not is_set (ChapterURL) and is_set
(TitleLink) then
                                                Chapter = make_wikilink
(TitleLink, Chapter);
                                        end
                                        Title = Periodical;
                                        ChapterFormat = Format;
                                        Periodical = '';
-- redundant so unset
                                        TransTitle = '';
                                        URL = '';
                                        Format = '';
                                        TitleLink = '';
                                        ScriptTitle = '';
```

```
                                end
                        elseif is_set (Chapter) then
-- |title not set
                                Title = Periodical;
-- |encyclopedia set and |article set so map |encyclopedia to |title
                                Periodical = '';
-- redundant so unset
                        end
                end
        end

        -- Special case for cite techreport.
        if (config.CitationClass == "techreport") then
-- special case for cite techreport
                if is_set(A['Number']) then
-- cite techreport uses 'number', which other citations alias to 'issue'
                        if not is_set(ID) then
-- can we use ID for the "number"?
                                ID = A['Number'];
-- yes, use it
                        else
-- ID has a value so emit error message
                                table.insert( z.message_tail, {
set_error('redundant_parameters', {wrap_style ('parameter', 'id') .. ' and '
.. wrap_style ('parameter', 'number')}, true )});
                        end
                end
        end

        -- Account for the oddity that is {{cite conference}}, before
generation of COinS data.
        if 'conference' == config.CitationClass then
                if is_set(BookTitle) then
                        Chapter = Title;
                        Chapter_origin = 'title';
        --                ChapterLink = TitleLink;
-- |chapterlink= is deprecated
                        ChapterURL = URL;
                        ChapterUrlAccess = UrlAccess;
                        ChapterURL_origin = URL_origin;
                        URL_origin = '';
                        ChapterFormat = Format;
                        TransChapter = TransTitle;
                        TransChapter_origin = TransTitle_origin;
                        Title = BookTitle;
                        Format = '';
        --                TitleLink = '';
                        TransTitle = '';
                        URL = '';
                end
        elseif 'speech' ~= config.CitationClass then
```

```
                Conference = '';
-- not cite conference or cite speech so make sure this is empty string
        end

        -- cite map oddities
        local Cartography = "";
        local Scale = "";
        local Sheet = A['Sheet'] or '';
        local Sheets = A['Sheets'] or '';
        if config.CitationClass == "map" then
                if is_set (Chapter) then
                        table.insert( z.message_tail, { set_error(
'redundant_parameters', {wrap_style ('parameter', 'map') .. ' and ' ..
wrap_style ('parameter', Chapter_origin)}, true ) } );        -- add error
message
                end
                Chapter = A['Map'];
                Chapter_origin = A:ORIGIN('Map');
                ChapterURL = A['MapURL'];
                ChapterURL_origin = A:ORIGIN('MapURL');
                TransChapter = A['TransMap'];
                ScriptChapter = A['ScriptMap']
                ScriptChapter_origin = A:ORIGIN('ScriptMap')

                ChapterUrlAccess = MapUrlAccess;
                ChapterFormat = A['MapFormat'];

                Cartography = A['Cartography'];
                if is_set( Cartography ) then
                        Cartography = sepc .. " " .. wrap_msg ('cartography',
Cartography, use_lowercase);
                end
                Scale = A['Scale'];
                if is_set( Scale ) then
                        Scale = sepc .. " " .. Scale;
                end
        end

        -- Account for the oddities that are {{cite episode}} and {{cite
serial}}, before generation of COinS data.
        if 'episode' == config.CitationClass or 'serial' ==
config.CitationClass then
                local SeriesLink = A['SeriesLink'];

                link_title_ok (SeriesLink, A:ORIGIN ('SeriesLink'), Series,
'series');        -- check for wikimarkup in |series-link= or wikimarkup in
|series= when |series-link= is set

                local Network = A['Network'];
                local Station = A['Station'];
                local s, n = {}, {};
```

```
-- do common parameters first
                if is_set(Network) then table.insert(n, Network); end
                if is_set(Station) then table.insert(n, Station); end
                ID = table.concat(n, sepc .. ' ');
                if 'episode' == config.CitationClass then
-- handle the oddities that are strictly {{cite episode}}
                        local Season = A['Season'];
                        local SeriesNumber = A['SeriesNumber'];

                        if is_set (Season) and is_set (SeriesNumber) then
-- these are mutually exclusive so if both are set
                                table.insert( z.message_tail, { set_error(
'redundant_parameters', {wrap_style ('parameter', 'season') .. ' and ' ..
wrap_style ('parameter', 'seriesno')}, true ) } );                -- add
error message
                                SeriesNumber = '';
-- unset; prefer |season= over |seriesno=
                        end
-- assemble a table of parts concatenated later into Series
                        if is_set(Season) then table.insert(s, wrap_msg
('season', Season, use_lowercase)); end
                        if is_set(SeriesNumber) then table.insert(s, wrap_msg
('seriesnum', SeriesNumber, use_lowercase)); end
                        if is_set(Issue) then table.insert(s, wrap_msg
('episode', Issue, use_lowercase)); end
                        Issue = '';
-- unset because this is not a unique parameter
                        Chapter = Title;
-- promote title parameters to chapter
                        ScriptChapter = ScriptTitle;
                        ScriptChapter_origin = A:ORIGIN('ScriptTitle');
                        ChapterLink = TitleLink;
-- alias episodelink
                        TransChapter = TransTitle;
                        ChapterURL = URL;
                        ChapterUrlAccess = UrlAccess;
                        ChapterURL_origin = A:ORIGIN('URL');
                        Title = Series;
-- promote series to title
                        TitleLink = SeriesLink;
                        Series = table.concat(s, sepc .. ' ');
-- this is concatenation of season, seriesno, episode number

                        if is_set (ChapterLink) and not is_set (ChapterURL)
then                        -- link but not URL
                                Chapter = make_wikilink (ChapterLink,
Chapter);
                        elseif is_set (ChapterLink) and is_set (ChapterURL)
then                        -- if both are set, URL links episode;
                                Series = make_wikilink (ChapterLink, Series);
                        end
```

```
                        URL = '';
-- unset
                        TransTitle = '';
                        ScriptTitle = '';
                else
-- now oddities that are cite serial
                        Issue = '';
-- unset because this parameter no longer supported by the citation/core
version of cite serial
                        Chapter = A['Episode'];
-- TODO: make |episode= available to cite episode someday?
                        if is_set (Series) and is_set (SeriesLink) then
                                Series = make_wikilink (SeriesLink, Series);
                        end
                        Series = wrap_style ('italic-title', Series);
-- series is italicized
                end
        end
        -- end of {{cite episode}} stuff

        -- Account for the oddities that are {{cite arxiv}}, {{cite
biorxiv}}, {{cite citeseerx}}, {{cite ssrn}}, before generation of COinS
data.
        do
                if in_array (config.CitationClass,
whitelist.preprint_template_list) then
                        if not is_set (ID_list[config.CitationClass:upper()])
then                        -- |arxiv= or |eprint= required for cite arxiv;
|biorxiv= & |citeseerx= required for their templates
                                table.insert( z.message_tail, { set_error(
config.CitationClass .. '_missing', {}, true ) } );        -- add error
message
                        end

                        Periodical = ({['arxiv'] = 'arXiv', ['biorxiv'] =
'bioRxiv', ['citeseerx'] = 'CiteSeerX', ['ssrn'] = 'Social Science Research
Network'})[config.CitationClass];
                end
        end

        -- handle type parameter for those CS1 citations that have default
values
        if in_array(config.CitationClass, {"AV-media-notes", "interview",
"mailinglist", "map", "podcast", "pressrelease", "report", "techreport",
"thesis"}) then
                TitleType = set_titletype (config.CitationClass, TitleType);
                if is_set(Degree) and "Thesis" == TitleType then
-- special case for cite thesis
                        TitleType = Degree .. ' ' .. cfg.title_types
['thesis']:lower();
                end
```

```
        end

        if is_set(TitleType) then
-- if type parameter is specified
                TitleType = substitute( cfg.messages['type'], TitleType);
-- display it in parentheses
        -- TODO: Hack on TitleType to fix bunched parentheses problem
        end

        -- legacy: promote PublicationDate to Date if neither Date nor Year
are set.
        local Date_origin;
-- to hold the name of parameter promoted to Date; required for date error
messaging

        if not is_set (Date) then
                Date = Year;
-- promote Year to Date
                Year = nil;
-- make nil so Year as empty string isn't used for CITEREF
                if not is_set (Date) and is_set(PublicationDate) then
-- use PublicationDate when |date= and |year= are not set
                        Date = PublicationDate;
-- promote PublicationDate to Date
                        PublicationDate = '';
-- unset, no longer needed
                        Date_origin = A:ORIGIN('PublicationDate');
-- save the name of the promoted parameter
                else
                        Date_origin = A:ORIGIN('Year');
-- save the name of the promoted parameter
                end
        else
                Date_origin = A:ORIGIN('Date');
-- not a promotion; name required for error messaging
        end

        if PublicationDate == Date then PublicationDate = ''; end
-- if PublicationDate is same as Date, don't display in rendered citation

        --[[
        Go test all of the date-holding parameters for valid MOS:DATE format
and make sure that dates are real dates. This must be done before we do COinS
because here is where
        we get the date used in the metadata.
        Date validation supporting code is in
Module:Citation/CS1/Date_validation
        ]]
        do        -- create defined block to contain local variables
error_message, date_parameters_list, mismatch
                local error_message = '';
```

```
-- AirDate has been promoted to Date so not necessary to check it
                local date_parameters_list = {
                        ['access-date'] = {val=AccessDate, name=A:ORIGIN
('AccessDate')},
                        ['archive-date'] = {val=ArchiveDate, name=A:ORIGIN
('ArchiveDate')},
                        ['date'] = {val=Date, name=Date_origin},
                        ['doi-broken-date'] = {val=DoiBroken, name=A:ORIGIN
('DoiBroken')},
                        ['embargo'] = {val=Embargo, name=A:ORIGIN
('Embargo')},
                        ['lay-date'] = {val=LayDate, name=A:ORIGIN
('LayDate')},
                        ['publication-date'] ={val=PublicationDate,
name=A:ORIGIN ('PublicationDate')},
                        ['year'] = {val=Year, name=A:ORIGIN ('Year')},
                        };
                anchor_year, Embargo, error_message =
dates(date_parameters_list, COinS_date);

-- start temporary Julian / Gregorian calendar uncertainty categorization
                if COinS_date.inter_cal_cat then
                        add_prop_cat ('jul_greg_uncertainty');
                end
-- end temporary Julian / Gregorian calendar uncertainty categorization

                if is_set (Year) and is_set (Date) then
-- both |date= and |year= not normally needed;
                        local mismatch = year_date_check (Year, Date)
                        if 0 == mismatch then
-- |year= does not match a year-value in |date=
                                if is_set (error_message) then
-- if there is already an error message
                                        error_message = error_message .. ',
';                                              -- tack on this additional
message
                                end
                                error_message = error_message .. '&#124;year=
/ &#124;date= mismatch';
                        elseif 1 == mismatch then
-- |year= matches year-value in |date=
                                add_maint_cat ('date_year');
                        end
                end
                if not is_set(error_message) then
-- error free dates only
                        local modified = false;
-- flag
                        if is_set (DF) then
-- if we need to reformat dates
                                modified = reformat_dates
```

```
(date_parameters_list, DF, false);        -- reformat to DF format, use long
month names if appropriate
                        end

                        if true == date_hyphen_to_dash (date_parameters_list)
then                        -- convert hyphens to dashes where appropriate
                                modified = true;
                                add_maint_cat ('date_format');
-- hyphens were converted so add maint category
                        end
        -- for those wikis that can and want to have English date names
translated to the local language,
        -- uncomment these three lines.  Not supported by en.wiki (for
obvious reasons)
        -- set date_name_xlate() second argument to true to translate English
digits to local digits (will translate ymd dates)
        --                if date_name_xlate (date_parameters_list, false)
then
        --                        modified = true;
        --                end

                        if modified then
-- if the date_parameters_list values were modified
                                AccessDate = date_parameters_list['access-
date'].val;                        -- overwrite date holding parameters with
modified values
                                ArchiveDate = date_parameters_list['archive-
date'].val;
                                Date = date_parameters_list['date'].val;
                                DoiBroken = date_parameters_list['doi-broken-
date'].val;
                                LayDate = date_parameters_list['lay-
date'].val;
                                PublicationDate =
date_parameters_list['publication-date'].val;
                        end
                else
                        table.insert( z.message_tail, { set_error(
'bad_date', {error_message}, true ) } );        -- add this error message
                end
        end        -- end of do

        -- Link the title of the work if no |url= was provided, but we have a
|pmc= or a |doi= with |doi-access=free
        -- Here we unset Embargo if PMC not embargoed (|embargo= not set in
the citation) or if the embargo time has expired. Otherwise, holds embargo
date
        Embargo = is_embargoed (Embargo);
        if config.CitationClass == "journal" and not is_set(URL) and not
is_set(TitleLink) then
                if is_set(ID_list['PMC']) and not is_set (Embargo) then
```

```
-- if not embargoed or embargo has expired
                        URL=cfg.id_handlers['PMC'].prefix .. ID_list['PMC'];
-- set url to be the same as the PMC external link if not embargoed
                        URL_origin = cfg.id_handlers['PMC'].parameters[1];
-- set URL_origin to parameter name for use in error message if citation is
missing a |title=
                elseif is_set(ID_list['DOI']) and ID_access_levels['DOI'] ==
"free" then
                        URL=cfg.id_handlers['DOI'].prefix .. ID_list['DOI'];
                        URL_origin = cfg.id_handlers['DOI'].parameters[1];
                end
                if is_set(URL) and is_set(AccessDate) then
-- access date requires |url=; pmc or doi created url is not |url=
                        table.insert( z.message_tail, { set_error(
'accessdate_missing_url', {}, true ) } );
                        AccessDate = '';
-- unset
                end
        end

        -- At this point fields may be nil if they weren't specified in the
template use.  We can use that fact.
        -- Test if citation has no title
        if        not is_set(Title) and not is_set(TransTitle) and not
is_set(ScriptTitle) then        -- has special case for cite episode
                table.insert( z.message_tail, { set_error(
'citation_missing_title', {'episode' == config.CitationClass and 'series' or
'title'}, true ) } );
        end

        if cfg.keywords_xlate[Title] == 'none' and
                        in_array (config.CitationClass, {'journal',
'citation'}) and
                        (is_set (Periodical) or is_set (ScriptPeriodical))
and
                        ('journal' == Periodical_origin or 'script-journal'
== ScriptPeriodical_origin) then        -- special case for journal cites
                                Title = '';
-- set title to empty string
                                add_maint_cat ('untitled');
        end

        check_for_url ({
-- add error message when any of these parameters hold a URL
                ['title']=Title,
                [A:ORIGIN('Chapter')]=Chapter,
                [Periodical_origin] = Periodical,
                [PublisherName_origin] = PublisherName
                });

        -- COinS metadata (see <http://ocoins.info/>) for automated parsing
```

```
of citation information.
        -- handle the oddity that is cite encyclopedia and {{citation
|encyclopedia=something}}. Here we presume that
        -- when Periodical, Title, and Chapter are all set, then Periodical
is the book (encyclopedia) title, Title
        -- is the article title, and Chapter is a section within the article.
So, we remap
        local coins_chapter = Chapter;
-- default assuming that remapping not required
        local coins_title = Title;
-- et tu
        if 'encyclopaedia' == config.CitationClass or ('citation' ==
config.CitationClass and is_set (Encyclopedia)) then
                if is_set (Chapter) and is_set (Title) and is_set
(Periodical) then                -- if all are used then
                        coins_chapter = Title;
-- remap
                        coins_title = Periodical;
                end
        end
        local coins_author = a;
-- default for coins rft.au
        if 0 < #c then
-- but if contributor list
                coins_author = c;
-- use that instead
        end
        -- this is the function call to COinS()
        local OCinSoutput = COinS({
                ['Periodical'] = strip_apostrophe_markup (Periodical),
-- no markup in the metadata
                ['Encyclopedia'] = Encyclopedia,
-- just a flag; content ignored by ~/COinS
                ['Chapter'] = make_coins_title (coins_chapter,
ScriptChapter),                        -- Chapter and ScriptChapter stripped
of bold / italic wikimarkup
                ['Degree'] = Degree;
-- cite thesis only
                ['Title'] = make_coins_title (coins_title, ScriptTitle),
-- Title and ScriptTitle stripped of bold / italic wikimarkup
                ['PublicationPlace'] = PublicationPlace,
                ['Date'] = COinS_date.rftdate,
-- COinS_date has correctly formatted date if Date is valid;
                ['Season'] = COinS_date.rftssn,
                ['Quarter'] = COinS_date.rftquarter,
                ['Chron'] =  COinS_date.rftchron or (not COinS_date.rftdate
and Date) or '',        -- chron but if not set and invalid date format use
Date; keep this last bit?
                ['Series'] = Series,
                ['Volume'] = Volume,
                ['Issue'] = Issue,
```

```
                ['Pages'] = coins_pages or get_coins_pages (first_set
({Sheet, Sheets, Page, Pages, At}, 5)),        -- pages stripped of external
links
                ['Edition'] = Edition,
                ['PublisherName'] = PublisherName or Newsgroup,
-- any apostrophe markup already removed from PublisherName
                ['URL'] = first_set ({ChapterURL, URL}, 2),
                ['Authors'] = coins_author,
                ['ID_list'] = ID_list,
                ['RawPage'] = this_page.prefixedText,
        }, config.CitationClass);

        -- Account for the oddities that are {{cite arxiv}}, {{cite
biorxiv}}, {{cite citeseerx}}, and {{cite ssrn}} AFTER generation of COinS
data.
        if in_array (config.CitationClass, whitelist.preprint_template_list)
then        -- we have set rft.jtitle in COinS to arXiv, bioRxiv, CiteSeerX,
or ssrn now unset so it isn't displayed
                Periodical = '';
-- periodical not allowed in these templates; if article has been published,
use cite journal
        end

        -- special case for cite newsgroup.  Do this after COinS because we
are modifying Publishername to include some static text
        if 'newsgroup' == config.CitationClass and is_set (Newsgroup) then
                PublisherName = substitute (cfg.messages['newsgroup'],
external_link( 'news:' .. Newsgroup, Newsgroup, Newsgroup_origin, nil ));
        end

        -- Now perform various field substitutions.
        -- We also add leading spaces and surrounding markup and punctuation
to the
        -- various parts of the citation, but only when they are non-nil.
        local EditorCount;
-- used only for choosing {ed.) or (eds.) annotation at end of editor name-
list
        do
                local last_first_list;
                local control = {
                        format = NameListFormat,
-- empty string or 'vanc'
                        maximum = nil,
-- as if display-authors or display-editors not set
                        lastauthoramp = LastAuthorAmp,
                        mode = Mode
                };

                do
-- do editor name list first because the now unsupported coauthors used to
modify control table
```

```
                        control.maximum , editor_etal = get_display_names
(A['DisplayEditors'], #e, 'editors', editor_etal);
                        last_first_list, EditorCount = list_people(control,
e, editor_etal);

                        if is_set (Editors) then
                                Editors, editor_etal = name_has_etal
(Editors, editor_etal, false, 'editors');        -- find and remove
variations on et al.
                                if editor_etal then
                                        Editors = Editors .. ' ' ..
cfg.messages['et al'];                          -- add et al. to editors
parameter beause |display-editors=etal
                                end
                                EditorCount = 2;
-- we don't know but assume |editors= is multiple names; spoof to display
(eds.) annotation
                        else
                                Editors = last_first_list;
-- either an author name list or an empty string
                        end

                        if 1 == EditorCount and (true == editor_etal or 1 <
#e) then                -- only one editor displayed but includes etal then
                                EditorCount = 2;
-- spoof to display (eds.) annotation
                        end
                end
                do
-- now do interviewers
                        control.maximum , interviewer_etal =
get_display_names (A['DisplayInterviewers'], #interviewers_list,
'interviewers', interviewer_etal);
                        Interviewers = list_people (control,
interviewers_list, interviewer_etal);
                end
                do
-- now do translators
                        control.maximum , translator_etal = get_display_names
(A['DisplayTranslators'], #t, 'translators', translator_etal);
                        Translators = list_people (control, t,
translator_etal);
                end
                do
-- now do contributors
                        control.maximum , contributor_etal =
get_display_names (A['DisplayContributors'], #c, 'contributors',
contributor_etal);
                        Contributors = list_people (control, c,
contributor_etal);
                end
```

```
                do
-- now do authors
                        control.maximum , author_etal = get_display_names
(A['DisplayAuthors'], #a, 'authors', author_etal);

                        last_first_list = list_people(control, a,
author_etal);

                        if is_set (Authors) then
                                Authors, author_etal = name_has_etal
(Authors, author_etal, false, 'authors');        -- find and remove
variations on et al.
                                if author_etal then
                                        Authors = Authors .. ' ' ..
cfg.messages['et al'];                          -- add et al. to authors
parameter
                                end
                        else
                                Authors = last_first_list;
-- either an author name list or an empty string
                        end
                end
-- end of do
                if is_set (Authors) and is_set (Collaboration) then
                        Authors = Authors .. ' (' .. Collaboration .. ')';
-- add collaboration after et al.
                end

        end

        -- apply |[xx-]format= styling; at the end, these parameters hold
correctly styled format annotation,
        -- an error message if the associated url is not set, or an empty
string for concatenation
        ArchiveFormat = style_format (ArchiveFormat, ArchiveURL, 'archive-
format', 'archive-url');
        ConferenceFormat = style_format (ConferenceFormat, ConferenceURL,
'conference-format', 'conference-url');
        Format = style_format (Format, URL, 'format', 'url');
        LayFormat = style_format (LayFormat, LayURL, 'lay-format', 'lay-
url');
        TranscriptFormat = style_format (TranscriptFormat, TranscriptURL,
'transcript-format', 'transcripturl');

        -- special case for chapter format so no error message or cat when
chapter not supported
        if not (in_array(config.CitationClass, {'web', 'news', 'journal',
'magazine', 'pressrelease', 'podcast', 'newsgroup', 'arxiv', 'biorxiv',
'citeseerx', 'ssrn'}) or
                ('citation' == config.CitationClass and (is_set (Periodical)
or is_set (ScriptPeriodical)) and not is_set (Encyclopedia))) then
```

```
                        ChapterFormat = style_format (ChapterFormat,
ChapterURL, 'chapter-format', 'chapter-url');
        end

        if not is_set(URL) then
                if in_array(config.CitationClass, {"web","podcast",
"mailinglist"}) or                -- |url= required for cite web, cite
podcast, and cite mailinglist
                        ('citation' == config.CitationClass and ('website' ==
Periodical_origin or 'script-website' == ScriptPeriodical_origin)) then
-- and required for {{citation}} with |website= or |script-website=
                                table.insert( z.message_tail, { set_error(
'cite_web_url', {}, true ) } );
                end
                -- do we have |accessdate= without either |url= or |chapter-
url=?
                if is_set(AccessDate) and not is_set(ChapterURL)then
-- ChapterURL may be set when URL is not set;
                        table.insert( z.message_tail, { set_error(
'accessdate_missing_url', {}, true ) } );
                        AccessDate = '';
                end
        end

        local OriginalURL, OriginalURL_origin, OriginalFormat,
OriginalAccess;
        UrlStatus = UrlStatus:lower();
-- used later when assembling archived text
        if is_set( ArchiveURL ) then
                if is_set (ChapterURL) then
-- if chapter-url is set apply archive url to it
                        OriginalURL = ChapterURL;
-- save copy of source chapter's url for archive text
                        OriginalURL_origin = ChapterURL_origin;
-- name of chapter-url parameter for error messages
                        OriginalFormat = ChapterFormat;
-- and original |chapter-format=

                        if 'live' ~= UrlStatus then
                                ChapterURL = ArchiveURL
-- swap-in the archive's url
                                ChapterURL_origin = A:ORIGIN('ArchiveURL')
-- name of archive-url parameter for error messages
                                ChapterFormat = ArchiveFormat or '';
-- swap in archive's format
                                ChapterUrlAccess = nil;
-- restricted access levels do not make sense for archived urls
                        end
                elseif is_set (URL) then
                        OriginalURL = URL;
-- save copy of original source URL
```

```
                        OriginalURL_origin = URL_origin;
-- name of url parameter for error messages
                        OriginalFormat = Format;
-- and original |format=
                        OriginalAccess = UrlAccess;

                        if 'live' ~= UrlStatus then
-- if URL set then archive-url applies to it
                                URL = ArchiveURL
-- swap-in the archive's url
                                URL_origin = A:ORIGIN('ArchiveURL')
-- name of archive url parameter for error messages
                                Format = ArchiveFormat or '';
-- swap in archive's format
                                UrlAccess = nil;
-- restricted access levels do not make sense for archived urls
                        end
                end
        end

        if in_array(config.CitationClass, {'web','news','journal',
'magazine', 'pressrelease', 'podcast', 'newsgroup', 'arxiv', 'biorxiv',
'citeseerx', 'ssrn'}) or        -- if any of the 'periodical' cites except
encyclopedia
                ('citation' == config.CitationClass and (is_set (Periodical)
or is_set (ScriptPeriodical)) and not is_set (Encyclopedia)) then
                        local chap_param;
                        if is_set (Chapter) then
-- get a parameter name from one of these chapter related meta-parameters
                                chap_param = A:ORIGIN ('Chapter')
                        elseif is_set (TransChapter) then
                                chap_param = A:ORIGIN ('TransChapter')
                        elseif is_set (ChapterURL) then
                                chap_param = A:ORIGIN ('ChapterURL')
                        elseif is_set (ScriptChapter) then
                                chap_param = ScriptChapter_origin;
                        else is_set (ChapterFormat)
                                chap_param = A:ORIGIN ('ChapterFormat')
                        end

                        if is_set (chap_param) then
-- if we found one
                                table.insert( z.message_tail, { set_error(
'chapter_ignored', {chap_param}, true ) } );                -- add error
message
                                Chapter = '';
-- and set them to empty string to be safe with concatenation
                                TransChapter = '';
                                ChapterURL = '';
                                ScriptChapter = '';
                                ChapterFormat = '';
```

```
                        end
        else
-- otherwise, format chapter / article title
                local no_quotes = false;
-- default assume that we will be quoting the chapter parameter value
                if is_set (Contribution) and 0 < #c then
-- if this is a contribution with contributor(s)
                        if in_array (Contribution:lower(),
cfg.keywords_lists.contribution) then        -- and a generic contribution
title
                                no_quotes = true;
-- then render it unquoted
                        end
                end

                Chapter = format_chapter_title (ScriptChapter,
ScriptChapter_origin, Chapter, Chapter_origin, TransChapter,
TransChapter_origin, ChapterURL, ChapterURL_origin, no_quotes,
ChapterUrlAccess);                -- Contribution is also in Chapter
                if is_set (Chapter) then
                        Chapter = Chapter .. ChapterFormat ;
                        if 'map' == config.CitationClass and is_set
(TitleType) then
                                Chapter = Chapter .. ' ' .. TitleType;
-- map annotation here; not after title
                        end
                        Chapter = Chapter.. sepc .. ' ';
                elseif is_set (ChapterFormat) then
-- |chapter= not set but |chapter-format= is so ...
                        Chapter = ChapterFormat .. sepc .. ' ';
-- ... ChapterFormat has error message, we want to see it
                end
        end

        -- Format main title
        if is_set (ArchiveURL) and
                (mw.ustring.match (mw.ustring.lower(Title),
cfg.special_case_translation.archived_copy.en) or        -- if title is
'Archived copy' (place holder added by bots that can't find proper title)
                mw.ustring.match (mw.ustring.lower(Title),
cfg.special_case_translation.archived_copy['local'])) then        -- local-
wiki's form
                        add_maint_cat ('archived_copy');
-- add maintenance category before we modify the content of Title
        end

        if Title:match ('^%(%(.*%)%)$') then
-- if keep as written markup:
                Title= Title:gsub ('^%(%((.*)%)%)$', '%1')
-- remove the markup
        else
```

```
                if '...' == Title:sub (-3) then
-- if elipsis is the last three characters of |title=
                        Title = Title:gsub ('(%.%.%.)%.+$', '%1');
-- limit the number of dots to three
                elseif not mw.ustring.find (Title, '%.%s*%a%.$') and
-- end of title is not a 'dot-(optional space-)letter-dot' initialism ...
                        not mw.ustring.find (Title, '%s+%a%.$') then
-- ...and not a 'space-letter-dot' initial (''Allium canadense'' L.)
                                Title = mw.ustring.gsub(Title,
'%'..sepc..'$', '');                                    -- remove any trailing
separator character; sepc and ms.ustring() here for languages that use
multibyte separator characters
                end
        end

        if in_array(config.CitationClass, {'web', 'news', 'journal',
'magazine', 'pressrelease', 'podcast', 'newsgroup', 'mailinglist',
'interview', 'arxiv', 'biorxiv', 'citeseerx', 'ssrn'}) or
                ('citation' == config.CitationClass and (is_set (Periodical)
or is_set (ScriptPeriodical)) and not is_set (Encyclopedia)) or
                ('map' == config.CitationClass and (is_set (Periodical) or
is_set (ScriptPeriodical))) then                -- special case for cite map
when the map is in a periodical treat as an article
                        Title = kern_quotes (Title);
-- if necessary, separate title's leading and trailing quote marks from
Module provided quote marks
                        Title = wrap_style ('quoted-title', Title);
                        Title = script_concatenate (Title, ScriptTitle,
'script-title');        -- <bdi> tags, lang atribute, categorization, etc;
must be done after title is wrapped
                        TransTitle= wrap_style ('trans-quoted-title',
TransTitle );
        elseif 'report' == config.CitationClass then
-- no styling for cite report
                Title = script_concatenate (Title, ScriptTitle, 'script-
title');                -- <bdi> tags, lang atribute, categorization, etc;
must be done after title is wrapped
                TransTitle= wrap_style ('trans-quoted-title', TransTitle );
-- for cite report, use this form for trans-title
        else
                Title = wrap_style ('italic-title', Title);
                Title = script_concatenate (Title, ScriptTitle, 'script-
title');                -- <bdi> tags, lang atribute, categorization, etc;
must be done after title is wrapped
                TransTitle = wrap_style ('trans-italic-title', TransTitle);
        end

        local TransError = "";
        if is_set(TransTitle) then
                if is_set(Title) then
                        TransTitle = " " .. TransTitle;
```

```
                else
                        TransError = " " .. set_error( 'trans_missing_title',
{'title'} );
                end
        end

        if is_set (Title) then
-- TODO: is this the right place to be making wikisource urls?
                if is_set (TitleLink) and is_set (URL) then
                        table.insert( z.message_tail, { set_error(
'wikilink_in_url', {}, true ) } );        -- set an error message because we
can't have both
                        TitleLink = '';
-- unset
                end
                if not is_set (TitleLink) and is_set (URL) then
                        Title = external_link (URL, Title, URL_origin,
UrlAccess) .. TransTitle .. TransError .. Format;
                        URL = '';
-- unset these because no longer needed
                        Format = "";
                elseif is_set (TitleLink) and not is_set (URL) then
                        local ws_url;
                        ws_url = wikisource_url_make (TitleLink);
-- ignore ws_label return; not used here
                        if ws_url then
                                Title = external_link (ws_url, Title ..
' ', 'ws link in title-link');        -- space char after Title to move
icon away from italic text; TODO: a better way to do this?
                                Title = substitute
(cfg.presentation['interwiki-icon'], {cfg.presentation['class-wikisource'],
TitleLink, Title});
                                Title = Title .. TransTitle .. TransError;
                        else
                                Title = make_wikilink (TitleLink, Title) ..
TransTitle .. TransError;
                        end
                else
                        local ws_url, ws_label;
-- Title has italic or quote markup by the time we get here which causes
is_wikilink() to return 0 (not a wikilink)
                        ws_url, ws_label, L = wikisource_url_make
(Title:gsub('[\'"](.-)[\'"]', '%1'));        -- make ws url from |title=
interwiki link (strip italic or quote markup); link portion L becomes tool
tip label
                        if ws_url then
                                Title = Title:gsub ('%b[]', ws_label);
-- replace interwiki link with ws_label to retain markup
                                Title = external_link (ws_url, Title ..
' ', 'ws link in title');        -- space char after Title to move icon
away from italic text; TODO: a better way to do this?
```

```
                                Title = substitute
(cfg.presentation['interwiki-icon'], {cfg.presentation['class-wikisource'],
L, Title});
                                Title = Title .. TransTitle .. TransError;
                        else
                                Title = Title .. TransTitle .. TransError;
                        end
                end
        else
                Title = TransTitle .. TransError;
        end

        if is_set(Place) then
                Place = " " .. wrap_msg ('written', Place, use_lowercase) ..
sepc .. " ";
        end

        if is_set (Conference) then
                if is_set (ConferenceURL) then
                        Conference = external_link( ConferenceURL,
Conference, ConferenceURL_origin, nil );
                end
                Conference = sepc .. " " .. Conference .. ConferenceFormat;
        elseif is_set(ConferenceURL) then
                Conference = sepc .. " " .. external_link( ConferenceURL,
nil, ConferenceURL_origin, nil );
        end

        if not is_set(Position) then
                local Minutes = A['Minutes'];
                local Time = A['Time'];

                if is_set(Minutes) then
                        if is_set (Time) then
                                table.insert( z.message_tail, { set_error(
'redundant_parameters', {wrap_style ('parameter', 'minutes') .. ' and ' ..
wrap_style ('parameter', 'time')}, true ) } );
                        end
                        Position = " " .. Minutes .. " " ..
cfg.messages['minutes'];
                else
                        if is_set(Time) then
                                local TimeCaption = A['TimeCaption']
                                if not is_set(TimeCaption) then
                                        TimeCaption = cfg.messages['event'];
                                        if sepc ~= '.' then
                                                TimeCaption =
TimeCaption:lower();
                                        end
                                end
                                Position = " " .. TimeCaption .. " " .. Time;
```

```
                        end
                end
        else
                Position = " " .. Position;
                At = '';
        end

        Page, Pages, Sheet, Sheets = format_pages_sheets (Page, Pages, Sheet,
Sheets, config.CitationClass, Periodical_origin, sepc, NoPP, use_lowercase);

        At = is_set(At) and (sepc .. " " .. At) or "";
        Position = is_set(Position) and (sepc .. " " .. Position) or "";
        if config.CitationClass == 'map' then
                local Sections = A['Sections'];
-- Section (singular) is an alias of Chapter so set earlier
                local Inset = A['Inset'];
                if is_set( Inset ) then
                        Inset = sepc .. " " .. wrap_msg ('inset', Inset,
use_lowercase);
                end

                if is_set( Sections ) then
                        Section = sepc .. " " .. wrap_msg ('sections',
Sections, use_lowercase);
                elseif is_set( Section ) then
                        Section = sepc .. " " .. wrap_msg ('section',
Section, use_lowercase);
                end
                At = At .. Inset .. Section;
        end

        if is_set (Language) then
                Language = language_parameter (Language);
-- format, categories, name from ISO639-1, etc
        else
                Language="";
-- language not specified so make sure this is an empty string;
        --[[ TODO: need to extract the wrap_msg from language_parameter
        so that we can solve parentheses bunching problem with
Format/Language/TitleType
        ]]
        end

        Others = is_set(Others) and (sepc .. " " .. Others) or "";
        if is_set (Translators) then
                Others = safe_join ({sepc .. ' ', wrap_msg ('translated',
Translators, use_lowercase), Others}, sepc);
        end
        if is_set (Interviewers) then
                Others = safe_join ({sepc .. ' ', wrap_msg ('interview',
Interviewers, use_lowercase), Others}, sepc);
```

```
        end
        TitleNote = is_set(TitleNote) and (sepc .. " " .. TitleNote) or "";
        if is_set (Edition) then
                if Edition:match ('%f[%a][Ee]d%.?$') or Edition:match
('%f[%a][Ee]dition$') then
                        add_maint_cat ('extra_text', 'edition');
                end
                Edition = " " .. wrap_msg ('edition', Edition);
        else
                Edition = '';
        end

        Series = is_set (Series) and wrap_msg ('series', {sepc, Series}) or
"";               -- not the same as SeriesNum
        OrigYear = is_set (OrigYear) and wrap_msg ('origyear', OrigYear) or
'';
        Agency = is_set (Agency) and wrap_msg ('agency', {sepc, Agency}) or
"";
        Volume = format_volume_issue (Volume, Issue, config.CitationClass,
Periodical_origin, sepc, use_lowercase);

        ------------------------------------ totally unrelated data
        Via = is_set (Via) and  wrap_msg ('via', Via) or '';

        if is_set(AccessDate) then
                local retrv_text = " " .. cfg.messages['retrieved']

                AccessDate = nowrap_date (AccessDate);
-- wrap in nowrap span if date in appropriate format
                if (sepc ~= ".") then retrv_text = retrv_text:lower() end
-- if mode is cs2, lower case
                AccessDate = substitute (retrv_text, AccessDate);
-- add retrieved text

                AccessDate = substitute (cfg.presentation['accessdate'],
{sepc, AccessDate});        -- allow editors to hide accessdates
        end
        if is_set(ID) then ID = sepc .." ".. ID; end
        if "thesis" == config.CitationClass and is_set(Docket) then
                ID = sepc .." Docket ".. Docket .. ID;
        end
        if "report" == config.CitationClass and is_set(Docket) then
-- for cite report when |docket= is set
                ID = sepc .. ' ' .. Docket;
-- overwrite ID even if |id= is set
        end

        ID_list = build_id_list( ID_list, {IdAccessLevels=ID_access_levels,
DoiBroken = DoiBroken, ASINTLD = ASINTLD, IgnoreISBN = IgnoreISBN,
Embargo=Embargo, Class = Class} );
```

```
        if is_set(URL) then
                URL = " " .. external_link( URL, nil, URL_origin, UrlAccess
);
        end

        if is_set(Quote) then
                if Quote:sub(1,1) == '"' and Quote:sub(-1,-1) == '"' then
-- if first and last characters of quote are quote marks
                        Quote = Quote:sub(2,-2);
-- strip them off
                end
                Quote = sepc .." " .. wrap_style ('quoted-text', Quote );
-- wrap in <q>...</q> tags
                PostScript = "";
-- cs1|2 does not supply terminal punctuation when |quote= is set
        end
        local Archived
        if is_set(ArchiveURL) then
                local arch_text;
                if not is_set(ArchiveDate) then
                        ArchiveDate = set_error('archive_missing_date');
                end
                if "live" == UrlStatus then
                        arch_text = cfg.messages['archived'];
                        if sepc ~= "." then arch_text = arch_text:lower() end
                        Archived = sepc .. " " .. substitute(
cfg.messages['archived-live'],
                                { external_link( ArchiveURL, arch_text,
A:ORIGIN('ArchiveURL'), nil ) .. ArchiveFormat, ArchiveDate } );
                        if not is_set (OriginalURL) then
                                Archived = Archived .. " " ..
set_error('archive_missing_url');
                        end
                elseif is_set(OriginalURL) then
-- UrlStatus is empty, 'dead', 'unfit', 'usurped', 'bot: unknown'
                        if in_array (UrlStatus, {'unfit', 'usurped', 'bot:
unknown'}) then
                                arch_text = cfg.messages['archived-unfit'];
                                if sepc ~= "." then arch_text =
arch_text:lower() end
                                Archived = sepc .. " " .. arch_text ..
ArchiveDate;                                    -- format already styled
                                if 'bot: unknown' == UrlStatus then
                                        add_maint_cat ('bot:_unknown');
-- and add a category if not already added
                                else
                                        add_maint_cat ('unfit');
-- and add a category if not already added
                                end
                        else
-- UrlStatus is empty, 'dead'
```

```
                                arch_text = cfg.messages['archived-dead'];
                                if sepc ~= "." then arch_text =
arch_text:lower() end
                                Archived = sepc .. " " .. substitute(
arch_text,
                                        { external_link( OriginalURL,
cfg.messages['original'], OriginalURL_origin, OriginalAccess ) ..
OriginalFormat, ArchiveDate } );        -- format already styled
                        end
                else
-- OriginalUrl not set
                        arch_text = cfg.messages['archived-missing'];
                        if sepc ~= "." then arch_text = arch_text:lower() end
                        Archived = sepc .. " " .. substitute( arch_text,
                                { set_error('archive_missing_url'),
ArchiveDate } );
                end
        elseif is_set (ArchiveFormat) then
                Archived = ArchiveFormat;
-- if set and ArchiveURL not set ArchiveFormat has error message
        else
                Archived = ""
        end
        local Lay = '';
        if is_set(LayURL) then
                if is_set(LayDate) then LayDate = " (" .. LayDate .. ")" end
                if is_set(LaySource) then
                        LaySource = " &ndash; ''" ..
safe_for_italics(LaySource) .. "''";
                else
                        LaySource = "";
                end
                if sepc == '.' then
                        Lay = sepc .. " " .. external_link( LayURL,
cfg.messages['lay summary'], A:ORIGIN('LayURL'), nil ) .. LayFormat ..
LaySource .. LayDate
                else
                        Lay = sepc .. " " .. external_link( LayURL,
cfg.messages['lay summary']:lower(), A:ORIGIN('LayURL'), nil ) .. LayFormat
.. LaySource .. LayDate
                end
        elseif is_set (LayFormat) then
-- Test if |lay-format= is given without giving a |lay-url=
                Lay = sepc .. LayFormat;
-- if set and LayURL not set, then LayFormat has error message
        end

        if is_set(Transcript) then
                if is_set(TranscriptURL) then
                        Transcript = external_link( TranscriptURL,
Transcript, TranscriptURL_origin, nil );
```

```
                end
                Transcript = sepc .. ' ' .. Transcript .. TranscriptFormat;
        elseif is_set(TranscriptURL) then
                Transcript = external_link( TranscriptURL, nil,
TranscriptURL_origin, nil );
        end

        local Publisher;
        if is_set(PublicationDate) then
                PublicationDate = wrap_msg ('published', PublicationDate);
        end
        if is_set(PublisherName) then
                if is_set(PublicationPlace) then
                        Publisher = sepc .. " " .. PublicationPlace .. ": "
.. PublisherName .. PublicationDate;
                else
                        Publisher = sepc .. " " .. PublisherName ..
PublicationDate;
                end
        elseif is_set(PublicationPlace) then
                Publisher= sepc .. " " .. PublicationPlace ..
PublicationDate;
        else
                Publisher = PublicationDate;
        end
        -- Several of the above rely upon detecting this as nil, so do it
last.
        if (is_set (Periodical) or is_set (ScriptPeriodical) or is_set
(TransPeriodical)) then
                if is_set(Title) or is_set(TitleNote) then
                        Periodical = sepc .. " " .. format_periodical
(ScriptPeriodical, ScriptPeriodical_origin, Periodical, TransPeriodical,
TransPeriodical_origin);
                else
                        Periodical = format_periodical (ScriptPeriodical,
ScriptPeriodical_origin, Periodical, TransPeriodical,
TransPeriodical_origin);
                end
        end

        --[[
        Handle the oddity that is cite speech.  This code overrides whatever
may be the value assigned to TitleNote (through |department=) and forces it
to be " (Speech)" so that
        the annotation directly follows the |title= parameter value in the
citation rather than the |event= parameter value (if provided).
        ]]
        if "speech" == config.CitationClass then
-- cite speech only
                TitleNote = " (Speech)";
-- annotate the citation
```

```
                if is_set (Periodical) then
-- if Periodical, perhaps because of an included |website= or |journal=
parameter
                        if is_set (Conference) then
-- and if |event= is set
                                Conference = Conference .. sepc .. " ";
-- then add appropriate punctuation to the end of the Conference variable
before rendering
                        end
                end
        end

        -- Piece all bits together at last.  Here, all should be non-nil.
        -- We build things this way because it is more efficient in LUA
        -- not to keep reassigning to the same string variable over and over.

        local tcommon;
        local tcommon2;
-- used for book cite when |contributor= is set
        if in_array(config.CitationClass, {"journal","citation"}) and
is_set(Periodical) then
                if is_set(Others) then Others = safe_join ({Others, sepc .. "
"}, sepc) end                -- add terminal punctuation & space; check for
dup sepc; TODO why do we need to do this here?
                tcommon = safe_join( {Others, Title, TitleNote, Conference,
Periodical, Format, TitleType, Series, Language, Edition, Publisher, Agency,
Volume}, sepc );
        elseif in_array(config.CitationClass, {"book","citation"}) and not
is_set(Periodical) then                -- special cases for book cites
                if is_set (Contributors) then
-- when we are citing foreword, preface, introduction, etc
                        tcommon = safe_join( {Title, TitleNote}, sepc );
-- author and other stuff will come after this and before tcommon2
                        tcommon2 = safe_join( {Conference, Periodical,
Format, TitleType, Series, Language, Volume, Others, Edition, Publisher,
Agency}, sepc );
                else
                        tcommon = safe_join( {Title, TitleNote, Conference,
Periodical, Format, TitleType, Series, Language, Volume, Others, Edition,
Publisher, Agency}, sepc );
                end

        elseif 'map' == config.CitationClass then
-- special cases for cite map
                if is_set (Chapter) then
-- map in a book; TitleType is part of Chapter
                        tcommon = safe_join( {Title, Format, Edition, Scale,
Series, Language, Cartography, Others, Publisher, Volume}, sepc );
                elseif is_set (Periodical) then
-- map in a periodical
                        tcommon = safe_join( {Title, TitleType, Format,
```

```
Periodical, Scale, Series, Language, Cartography, Others, Publisher, Volume},
sepc );
                else
-- a sheet or stand-alone map
                        tcommon = safe_join( {Title, TitleType, Format,
Edition, Scale, Series, Language, Cartography, Others, Publisher}, sepc );
                end
        elseif 'episode' == config.CitationClass then
-- special case for cite episode
                tcommon = safe_join( {Title, TitleNote, TitleType, Series,
Language, Edition, Publisher}, sepc );

        else
-- all other CS1 templates
                tcommon = safe_join( {Title, TitleNote, Conference,
Periodical, Format, TitleType, Series, Language,
                        Volume, Others, Edition, Publisher, Agency}, sepc );
        end
        if #ID_list > 0 then
                ID_list = safe_join( { sepc .. " ",  table.concat( ID_list,
sepc .. " " ), ID }, sepc );
        else
                ID_list = ID;
        end
        local idcommon;
        if 'audio-visual' == config.CitationClass or 'episode' ==
config.CitationClass then        -- special case for cite AV media & cite
episode position transcript
                idcommon = safe_join( { ID_list, URL, Archived, Transcript,
AccessDate, Via, Lay, Quote }, sepc );
        else
                idcommon = safe_join( { ID_list, URL, Archived, AccessDate,
Via, Lay, Quote }, sepc );
        end
        local text;
        local pgtext = Position .. Sheet .. Sheets .. Page .. Pages .. At;

        if is_set(Date) then
                if is_set (Authors) or is_set (Editors) then
-- date follows authors or editors when authors not set
                        Date = " (" .. Date ..")" .. OrigYear .. sepc .. " ";
-- in paranetheses
                else
-- neither of authors and editors set
                        if (string.sub(tcommon,-1,-1) == sepc) then
-- if the last character of tcommon is sepc
                                Date = " " .. Date .. OrigYear;
-- Date does not begin with sepc
                        else
                                Date = sepc .. " " .. Date .. OrigYear;
-- Date begins with sepc
```

```
                        end
                end
        end
        if is_set(Authors) then
                if (not is_set (Date)) then
-- when date is set it's in parentheses; no Authors termination
                        Authors = terminate_name_list (Authors, sepc);
-- when no date, terminate with 0 or 1 sepc and a space
                end
                if is_set(Editors) then
                        local in_text = " ";
                        local post_text = "";
                        if is_set(Chapter) and 0 == #c then
                                in_text = in_text .. cfg.messages['in'] .. "
"
                                if (sepc ~= '.') then
                                        in_text = in_text:lower()
-- lowercase for cs2
                                end
                        end
                        if EditorCount <= 1 then
                                post_text = " (" .. cfg.messages['editor'] ..
")";                                -- be consistent with no-author, no-date
case
                        else
                                post_text = " (" .. cfg.messages['editors']
.. ")";
                        end
                        Editors = terminate_name_list (in_text .. Editors ..
post_text, sepc);        -- terminate with 0 or 1 sepc and a space
                end
                if is_set (Contributors) then
-- book cite and we're citing the intro, preface, etc
                        local by_text = sepc .. ' ' .. cfg.messages['by'] ..
' ';
                        if (sepc ~= '.') then by_text = by_text:lower() end
-- lowercase for cs2
                        Authors = by_text .. Authors;
-- author follows title so tweak it here
                        if is_set (Editors) and is_set (Date) then
-- when Editors make sure that Authors gets terminated
                                Authors = terminate_name_list (Authors,
sepc);                                          -- terminate with 0 or 1 sepc
and a space
                        end
                        if (not is_set (Date)) then
-- when date is set it's in parentheses; no Contributors termination
                                Contributors = terminate_name_list
(Contributors, sepc);                -- terminate with 0 or 1 sepc and a
space
                        end
```

```
                        text = safe_join( {Contributors, Date, Chapter,
tcommon, Authors, Place, Editors, tcommon2, pgtext, idcommon }, sepc );
                else
                        text = safe_join( {Authors, Date, Chapter, Place,
Editors, tcommon, pgtext, idcommon }, sepc );
                end
        elseif is_set(Editors) then
                if is_set(Date) then
                        if EditorCount <= 1 then
                                Editors = Editors .. ", " ..
cfg.messages['editor'];
                        else
                                Editors = Editors .. ", " ..
cfg.messages['editors'];
                        end
                else
                        if EditorCount <= 1 then
                                Editors = Editors .. " (" ..
cfg.messages['editor'] .. ")" .. sepc .. " "
                        else
                                Editors = Editors .. " (" ..
cfg.messages['editors'] .. ")" .. sepc .. " "
                        end
                end
                text = safe_join( {Editors, Date, Chapter, Place, tcommon,
pgtext, idcommon}, sepc );
        else
                if in_array(config.CitationClass, {"journal","citation"}) and
is_set(Periodical) then
                        text = safe_join( {Chapter, Place, tcommon, pgtext,
Date, idcommon}, sepc );
                else
                        text = safe_join( {Chapter, Place, tcommon, Date,
pgtext, idcommon}, sepc );
                end
        end
        if is_set(PostScript) and PostScript ~= sepc then
                text = safe_join( {text, sepc}, sepc );
--Deals with italics, spaces, etc.
                text = text:sub(1,-sepc:len()-1);
        end
        text = safe_join( {text, PostScript}, sepc );

        -- Now enclose the whole thing in a <cite/> element
        local options = {};
        if is_set(config.CitationClass) and config.CitationClass ~=
"citation" then
                options.class = string.format ('%s %s %s', 'citation',
config.CitationClass, is_set (Mode) and Mode or 'cs1');        --
class=citation required for blue highlight when used with |ref=
        else
```

```
                options.class = string.format ('%s %s', 'citation', is_set
(Mode) and Mode or 'cs2');
        end
        if is_set(Ref) and 'none' ~= cfg.keywords_xlate[Ref:lower()] then
                local id = Ref
                if ('harv' == Ref ) then
                        local namelist = {};
-- holds selected contributor, author, editor name list
                        local year = first_set ({Year, anchor_year}, 2);
-- Year first for legacy citations and for YMD dates that require
disambiguation

                        if #c > 0 then
-- if there is a contributor list
                                namelist = c;
-- select it
                        elseif #a > 0 then
-- or an author list
                                namelist = a;
                        elseif #e > 0 then
-- or an editor list
                                namelist = e;
                        end
                        if #namelist > 0 then
-- if there are names in namelist
                                id = anchor_id (namelist, year);
-- go make the CITEREF anchor
                        else
                                id = '';
-- unset
                        end
                end
                options.id = id;
        end
        if string.len(text:gsub("<span[^>/]*>(.-)</span>",
"%1"):gsub("%b<>","")) <= 2 then        -- remove <span> tags and other html-
like markup; then get length of what remains
                z.error_categories = {};
                text = set_error('empty_citation');
                z.message_tail = {};
        end
        local render = {};
-- here we collect the final bits for concatenation into the rendered
citation

        if is_set(options.id) then
-- here we wrap the rendered citation in <cite ...>...</cite> tags
                table.insert (render, substitute (cfg.presentation['cite-
id'], {mw.uri.anchorEncode(options.id), mw.text.nowiki(options.class),
text}));        -- when |ref= is set
        else
```

```
                table.insert (render, substitute (cfg.presentation['cite'],
{mw.text.nowiki(options.class), text}));        -- all other cases
        end

        table.insert (render, substitute (cfg.presentation['ocins'],
{OCinSoutput}));        -- append metadata to the citation

        if 0 ~= #z.message_tail then
                table.insert (render, ' ');
                for i,v in ipairs( z.message_tail ) do
                        if is_set(v[1]) then
                                if i == #z.message_tail then
                                        table.insert (render, error_comment(
v[1], v[2] ));
                                else
                                        table.insert (render, error_comment(
v[1] .. "; ", v[2] ));
                                end
                        end
                end
        end

        if 0 ~= #z.maintenance_cats then
                local maint_msgs = {};
-- here we collect all of the maint messages
                for _, v in ipairs( z.maintenance_cats ) do
-- append maintenance categories
                        local maint = {};
-- here we assemble a maintenence message
                        table.insert (maint, v);
-- maint msg is the category name
                        table.insert (maint, ' (');
-- open the link text
                        table.insert (maint, make_wikilink (':Category:' ..
v, 'link'));        -- add the link
                        table.insert (maint, ')');
-- and close it
                        table.insert (maint_msgs, table.concat (maint));
-- assemble new maint message and add it to the maint_msgs table
                end
                table.insert (render, substitute (cfg.presentation['hidden-
maint'], table.concat (maint_msgs, ' ')));        -- wrap the group of maint
message with proper presentation and save
        end
        if not no_tracking_cats then
                for _, v in ipairs( z.error_categories ) do
                        table.insert (render, make_wikilink ('Category:' ..
v));
                end
                for _, v in ipairs( z.maintenance_cats ) do
-- append maintenance categories
```

```
                        table.insert (render, make_wikilink ('Category:' ..
v));
                end
                for _, v in ipairs( z.properties_cats ) do
-- append properties categories
                        table.insert (render, make_wikilink ('Category:' ..
v));
                end
        end

        return table.concat (render);
end


--[[--------------------------< V A L I D A T E >--------------------------
----------------------------------

Looks for a parameter's name in one of several whitelists.

Parameters in the whitelist can have three values:
        true - active, supported parameters
        false - deprecated, supported parameters
        nil - unsupported parameters
]]

local function validate (name, cite_class)
        local name = tostring (name);
        local state;
        local function state_test (state, name)
-- local function to do testing of state values
                if true == state then return true; end
-- valid actively supported parameter
                if false == state then
                        deprecated_parameter (name);
-- parameter is deprecated but still supported
                        return true;
                end
                return nil;
        end

        if name:find ('#') then
-- # is a cs1|2 reserved character so parameters with # not permitted
                return nil;
        end

        if in_array (cite_class, whitelist.preprint_template_list ) then
-- limited parameter sets allowed for these templates
                state = whitelist.limited_basic_arguments[name];
                if true == state_test (state, name) then return true; end

                state = whitelist.preprint_arguments[cite_class][name];
```

```
-- look in the parameter-list for the template identified by cite_class
                if true == state_test (state, name) then return true; end

-- limited enumerated parameters list
                name = name:gsub("%d+", "#" );
-- replace digit(s) with # (last25 becomes last#) (mw.ustring because non-
Western 'local' digits)
                state = whitelist.limited_numbered_arguments[name];
                if true == state_test (state, name) then return true; end

                return false;
-- not supported because not found or name is set to nil
        end
-- end limited parameter-set templates

        if in_array (cite_class, whitelist.unique_param_template_list) then
-- experiment for template-specific parameters for templates that accept
parameters from the basic argument list
                state = whitelist.unique_arguments[cite_class][name];
-- look in the template-specific parameter-lists for the template identified
by cite_class
                if true == state_test (state, name) then return true; end
        end
-- if here, fall into general validation
        state = whitelist.basic_arguments[name];
-- all other templates; all normal parameters allowed
        if true == state_test (state, name) then return true; end

-- all enumerated parameters allowed
        name = name:gsub("%d+", "#" );
-- replace digit(s) with # (last25 becomes last#) (mw.ustring because non-
Western 'local' digits)
        state = whitelist.numbered_arguments[name];
        if true == state_test (state, name) then return true; end

        return false;
-- not supported because not found or name is set to nil
end


--[[-------------------------< M I S S I N G _ P I P E _ C H E C K >--------
----------------------------------

Look at the contents of a parameter. If the content has a string of
characters and digits followed by an equal
sign, compare the alphanumeric string to the list of cs1|2 parameters.  If
found, then the string is possibly a
parameter that is missing its pipe.  There are two tests made:
        {{cite ... |title=Title access-date=2016-03-17}}        -- the first
parameter has a value and whitespace separates that value from the missing
pipe parameter name
```

```
        {{cite ... |title=access-date=2016-03-17}}                       --
the first parameter has no value (whitespace after the first = is trimmed by
mediawiki)
cs1|2 shares some parameter names with xml/html atributes: class=, title=,
etc.  To prevent false positives xml/html
tags are removed before the search.

If a missing pipe is detected, this function adds the missing pipe
maintenance category.

]]

local function missing_pipe_check (parameter, value)
        local capture;
        value = value:gsub ('%b<>', '');
-- remove xml/html tags because attributes: class=, title=, etc

        capture = value:match ('%s+(%a[%w%-]+)%s*=') or value:match
('^(%a[%w%-]+)%s*=');        -- find and categorize parameters with possible
missing pipes
        if capture and validate (capture) then
-- if the capture is a valid parameter name
                table.insert( z.message_tail, {set_error ('missing_pipe',
parameter)});
        end
end


--[[--------------------------< H A S _ E X T R A N E O U S _ P U N C T >----
---------------------------------

look for extraneous terminal punctuation in most parameter values; parameters
listed in skip table are not checked

]]

local function has_extraneous_punc (param, value)
        if 'number' == type (param) then
                return;
        end
        param = param:gsub ('%d+', '#');
-- enumerated name-list mask params allow terminal punct; normalize
        if cfg.punct_skip[param] then
                return;
-- parameter name found in the skip table so done
        end
        if value:match ('[,;:]$') then
                add_maint_cat ('extra_punct');
-- has extraneous punctuation; add maint cat
        end
end
```

```
--[[--------------------------< C I T A T I O N >----------------------------
--------------------------------

This is used by templates such as {{cite book}} to create the actual citation
text.

]]

local function citation(frame)
	Frame = frame;
-- save a copy incase we need to display an error message in preview mode
	local pframe = frame:getParent()
	local validation, utilities, identifiers, metadata, styles;
	if nil ~= string.find (frame:getTitle(), 'sandbox', 1, true) then
-- did the {{#invoke:}} use sandbox version?
		cfg = mw.loadData
('Module:Citation/CS1/Configuration/sandbox');				-- load sandbox
versions of support modules
		whitelist = mw.loadData
('Module:Citation/CS1/Whitelist/sandbox');
		utilities = require
('Module:Citation/CS1/Utilities/sandbox');
		validation = require
('Module:Citation/CS1/Date_validation/sandbox');
		identifiers = require
('Module:Citation/CS1/Identifiers/sandbox');
		metadata = require ('Module:Citation/CS1/COinS/sandbox');
		styles = 'Module:Citation/CS1/sandbox/styles.css';
	else
-- otherwise
		cfg = mw.loadData ('Module:Citation/CS1/Configuration');
-- load live versions of support modules
		whitelist = mw.loadData ('Module:Citation/CS1/Whitelist');
		utilities = require ('Module:Citation/CS1/Utilities');
		validation = require ('Module:Citation/CS1/Date_validation');
		identifiers = require ('Module:Citation/CS1/Identifiers');
		metadata = require ('Module:Citation/CS1/COinS');
		styles = 'Module:Citation/CS1/styles.css';

	end

	utilities.set_selected_modules (cfg);
-- so that functions in Utilities can see the cfg tables
	identifiers.set_selected_modules (cfg, utilities);
-- so that functions in Identifiers can see the selected cfg tables and
selected Utilities module
	validation.set_selected_modules (cfg, utilities);
-- so that functions in Date validataion can see selected cfg tables and the
selected Utilities module
	metadata.set_selected_modules (cfg, utilities);
-- so that functions in COinS can see the selected cfg tables and selected
```

```
Utilities module

        dates = validation.dates;
-- imported functions from Module:Citation/CS1/Date validation
        year_date_check = validation.year_date_check;
        reformat_dates = validation.reformat_dates;
        date_hyphen_to_dash = validation.date_hyphen_to_dash;
        date_name_xlate = validation.date_name_xlate;

        is_set = utilities.is_set;
-- imported functions from Module:Citation/CS1/Utilities
        in_array = utilities.in_array;
        substitute = utilities.substitute;
        error_comment = utilities.error_comment;
        set_error = utilities.set_error;
        select_one = utilities.select_one;
        add_maint_cat = utilities.add_maint_cat;
        wrap_style = utilities.wrap_style;
        safe_for_italics = utilities.safe_for_italics;
        is_wikilink = utilities.is_wikilink;
        make_wikilink = utilities.make_wikilink;
        strip_apostrophe_markup = utilities.strip_apostrophe_markup;

        z = utilities.z;
-- table of error and category tables in Module:Citation/CS1/Utilities

        extract_ids = identifiers.extract_ids;
-- imported functions from Module:Citation/CS1/Identifiers
        build_id_list = identifiers.build_id_list;
        is_embargoed = identifiers.is_embargoed;
        extract_id_access_levels = identifiers.extract_id_access_levels;
        make_coins_title = metadata.make_coins_title;
-- imported functions from Module:Citation/CS1/COinS
        get_coins_pages = metadata.get_coins_pages;
        COinS = metadata.COinS;

        local args = {};
-- table where we store all of the template's arguments
        local suggestions = {};
-- table where we store suggestions if we need to loadData them
        local error_text, error_state;

        local config = {};
-- table to store parameters from the module {{#invoke:}}
        for k, v in pairs( frame.args ) do
-- get parameters from the {{#invoke}} frame
                config[k] = v;
        --        args[k] = v;
-- crude debug support that allows us to render a citation from module
{{#invoke:}}; skips parameter validation; TODO: keep?
        end
```

```
        local capture;
-- the single supported capture when matching unknown parameters using
patterns
        for k, v in pairs( pframe.args ) do
-- get parameters from the parent (template) frame
                if v ~= '' then
                        if ('string' == type (k)) then
                                k = mw.ustring.gsub (k, '%d',
cfg.date_names.local_digits);                -- for enumerated parameters,
translate 'local' digits to Western 0-9
                        end
                        if not validate( k, config.CitationClass ) then
                                error_text = "";
                                if type( k ) ~= 'string' then
                                        -- Exclude empty numbered parameters
                                        if v:match("%S+") ~= nil then
                                                error_text, error_state =
set_error( 'text_ignored', {v}, true );
                                        end
                                elseif validate( k:lower(),
config.CitationClass ) then
                                        error_text, error_state = set_error(
'parameter_ignored_suggest', {k, k:lower()}, true );        -- suggest the
lowercase version of the parameter
                                else
                                        if nil == suggestions.suggestions
then                                            -- if this table is nil
then we need to load it
                                                if nil ~= string.find
(frame:getTitle(), 'sandbox', 1, true) then                         -- did the
{{#invoke:}} use sandbox version?
                                                        suggestions =
mw.loadData( 'Module:Citation/CS1/Suggestions/sandbox' );        -- use the
sandbox version
                                                else
                                                        suggestions =
mw.loadData( 'Module:Citation/CS1/Suggestions' );                        --
use the live version
                                                end
                                        end
                                        for pattern, param in pairs
(suggestions.patterns) do                -- loop through the patterns to see
if we can suggest a proper parameter
                                                capture = k:match (pattern);
-- the whole match if no caputre in pattern else the capture if a match
                                                if capture then
-- if the pattern matches
                                                        param = substitute
(param, capture);                                -- add the capture to the
suggested parameter (typically the enumerator)
                                                        if validate (param,
```

```
config.CitationClass) then                -- validate the suggestion to make
sure that the suggestion is supported by this template (necessary for limited
parameter lists)
                                                                    error_text,
error_state = set_error ('parameter_ignored_suggest', {k, param}, true);
-- set the suggestion error message
                                                            else
                                                                    error_text,
error_state = set_error( 'parameter_ignored', {param}, true );        --
suggested param not supported by this template
                                                                    v = '';
-- unset
                                                            end
                                                    end
                                            end
                                            if not is_set (error_text) then
-- couldn't match with a pattern, is there an expicit suggestion?
                                                    if suggestions.suggestions[
k:lower() ] ~= nil then
                                                            error_text,
error_state = set_error( 'parameter_ignored_suggest', {k,
suggestions.suggestions[ k:lower() ]}, true );
                                                    else
                                                            error_text,
error_state = set_error( 'parameter_ignored', {k}, true );
                                                            v = '';
-- unset value assigned to unrecognized parameters (this for the limited
parameter lists)
                                                    end
                                            end
                                    end
                                    if error_text ~= '' then
                                            table.insert( z.message_tail,
{error_text, error_state} );
                                    end
                            end

                            args[k] = v;
-- save this parameter and its value

-- crude debug support that allows us to render a citation from module
{{#invoke:}} TODO: keep?
        --        elseif args[k] ~= nil or (k == 'postscript') then
-- when args[k] has a value from {{#invoke}} frame (we don't normally do
that)
        --                args[k] = v;
-- overwrite args[k] with empty string from pframe.args[k] (template frame);
v is empty string here
                end
-- not sure about the postscript bit; that gets handled in parameter
validation; historical artifact?
```

```
        end

        for k, v in pairs( args ) do
                if 'string' == type (k) then
-- don't evaluate positional parameters
                        has_invisible_chars (k, v);
-- look for invisible characters
                end
                has_extraneous_punc (k, v);
-- look for extraneous terminal punctuation in parameter values
                missing_pipe_check (k, v);
-- do we think that there is a parameter that is missing a pipe?
        end

        return table.concat ({citation0( config, args), frame:extensionTag
('templatestyles', '', {src=styles})});
end

--[[--------------------------< E X P O R T E D   F U N C T I O N S >--------
---------------------------------
]]

return {citation = citation};
```

one hundredth of an acre (0.004 ha)

accreted sediment in a river course or estuary, including both lateral (point-bars) and medial (braid-bars). Chars (or sand bars) emerge as islands within the river channel (island chars) or as attached land to the riverbanks (attached chars), create new opportunities for temporary settlements and agriculture.

actions taken to prevent or repair the deterioration of water management infrastructure and to keep the physical components of a water management system in such a state that they can serve their intended function.

Project Management Committee

Namespaces

- Module
- Discussion

Variants

Categories:

- Pages with script errors
- Pages with broken file links
- Templates using TemplateStyles
- Modules subject to page protection
- Modules that add a tracking category

This page was last edited on 23 August 2020, at 06:04.

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.