UKCA Code Management Guidelines

Version 0.99

Purpose

The objective of this guideline is to provide an agreed set of working practices for the collaborative development of UKCA code. By better co-ordination of changes to the UKCA and better communication of our plans we expect that new science will be available across the UKCA community more quickly and in a better tested way. It should be noted though that they are guidelines and if the code management group or the UKCA executive wishes to work outside these guidelines for specific reasons that is possible. However this should be the exception not the rule and requires the agreement of the whole CMG or exec. These guidelines should also be seen as a living document which evolves as we better understand the practicalities of the process or our requirements change. These guidelines will also evolve and develop as time progresses, so all UKCA developers should periodically review this document for updates. If you wish your code to be lodged into the main MetUM code base, you must follow these guidelines. For other changes, you are still encouraged to follow these guidelines as they are considered best practice for developing code but it is not essential.

Responsibilities

A reason for having these code management guidelines is to better define the responsibilities of the code developers, code management group and the Met Office UKCA team. Unless these are clearly defined, agreed and implemented collaboration will be increasingly unmanageable.

Responsibilities of code developers

- Follow the guidelines in this document when developing code. In particular, when developing code, note the need to keep tickets up to date and the coding standards.
- Ensure that code which is of interest to other people gets through the review process and is not left as a branch indefinitely

Responsibilities of code management group

- Have an overview of code changes under development
- Continue to adapt these guidelines in the light of experience
- Prioritise changes to be lodged if there is insufficient time to get all changes in to a particular release
- Report on progress on a regular basis to the Executive. (approximately monthly)
- For each UKCA development version provide validated, supported, scientific configurations which can be used as 'control' models to test new science. These will be called Supported science configurations.

- For each UM version provide jobs used to test the impact of changes being made. These jobs would typically be run for very short integrations to demonstrate that the change has not altered configurations it should not be changing (i.e. if you turn this new change off are the results identical to a job without the branch). These may include some jobs with UKCA off and will be called Standard testing jobs. They are also the basis for any automated testing through the UTF.

- With the Executive identify if there is any duplication of development effort or key science not being developed.

Responsibilities of Met Office UKCA team

- Reviewed code to be lodged in a timely manner

- Provide means of easily testing branches for no change on configurations other than those the change is meant to test. Ideally this will be by means of the User Test Framework (UTF).

- Provide advice and support to NCAS scientists on how to meet these guidelines and how to review code

- Provide code styling scripts and testing tools to make complying with Met Office coding standards easier.

How to make a code change

The steps detailed below are those involved for external collaborators in making a change to the UKCA code. Note that the whole process needs to be adhered to for changes that are destined to be consolidated on the trunk. Other changes can adjust the process appropriately. For full details of some of the technical details, the FCM User Guide is a good reference.

Plan Your Change

It is good practice to first consult the Code Management Group (CMG). This allows them to inform you of any development plans they know of that may have an impact on your work, and also keeps them informed of what you wish to do. Large changes with a big system impact should also be discussed at an early stage with the CMG.

Create a Ticket on the UKCA trac site (on the collaboration wiki)

When a change is being planned a ticket should be created on the UKCA Trac system (hosted on the Met Office collaboration server) describing the purpose of the change. We encourage all UKCA developers to do this once a new piece of science is thought of (even if it cannot be implemented for some time). This allows the rest of the UKCA community to see what others are working on, promotes collaboration and helps avoid duplicated effort (although of course there will be occasions when two different groups will be working on the same issue in parallel). All developers are strongly encouraged request a UKCA Trac account. The purpose of this ticket is to monitor across the whole UKCA community whether in the Met Office, UKCA academia or abroad. The ticket should be assigned to a developer at this stage. Please do not leave this entry empty as tickets can get forgotten about if not assigned. Please assign to yourself initially and if someone else agrees to work

on the ticket for you then assign it to them. The other fields in the ticket (e.g. Priority, Milestone, Keywords etc) should also be set appropriately. The ‘type’ field should be set to

- defect if the change is a bug-fix; correcting an error in existing UM code.
- enhancement if the change is a new development; adding new UM functionality.
- task if the ticket relates to testing, code styling, relocation of routines, etc.

The milestone field is there to help plan the lodging of code in the Met Office trunk. Many pieces of code will initially be opened as ‘Somewhen’ tickets. This means you don’t yet know when (or even if) it will be reviewed for acceptance in the UM trunk. As a piece of code becomes more mature though, you should consider if it will be ready in time for an upcoming UM code freeze. Please use ‘Not for Trunk’ if the related ticket is not intended for the UM trunk (e.g. a branch with diagnostics for debuggin a specific problem).

Priority can be set to major, normal, minor, trivial;

The CMG has the authority to categorize the appropriate priority for each ticket, guidelines for authors are:

- major. A change to the UM that requires significant modifications to both UKCA code and the UMUI e.g. a new chemistry scheme
- normal ; usual ticket status. Small code change to both UM code and UMUI.
- Minor. A change to the UM that only affects the UM code or the UMUI, but not both.
- Trivial. a simple one-line change to the UM, an obvious bug-fix, or purely a documentation change; requiring only a single level of review.

Component; what UKCA area is being altered? For example;

- 'Atmosphere - chemistry scheme':.
- 'Atmosphere - emissions'
- 'Atmosphere - emissions'
- 'Land - interactive emissions'

Each of these areas has a specific subsection expert. It is good practice to at least inform the owner of your plasns as they may be able to offer advice or be aware of similar efforts elsewhere. They will also have to approve the code as suitable for inclusion in the trunk. Details of the subsections and their owners are on the UKCA website. Note that all tickets for a particular milestone are visible either from the Roadmap feature in Trac or via a custom query in View Tickets.

Once you have opened a ticket, if you wish it to be lodged in the main UM trunk you must also inform the CMG who will assess if it is suitable in principle for inclusion.

NOTE: each ticket should in general correspond to a single change to the code and relate to one branch only (which may be upgraded from one UM version to a later one). Packages of unrelated changes are not generally acceptable for inclusion in the trunk. This is because they can have impacts on very many parts of the code making testing difficult and are difficult to review. If your work depends on changes on another branch try and keep the code changes as two separate branch and note this in your ticket. You will be unable to lodge your changes until the other branch is lodged. On rare occasions such as major conflicts, some changes may need to be merged to give a package but you must first seek the permission of the CMG group to do this. Avoid it if at all possible

Open a ticket on your own repository

As well as the centralised tickets on the UKCA trac system, all UM branches will need a ticket to be opened on their own UM trac site to record the details of the branch's development as per standard practice on your system

Starting Work (and Stopping)

A ticket may be a record of development work planned, a bug that needs to be corrected etc.. When a ticket is created, assigned and accepted, it does not necessarily imply that work is actually in progress. Developers should keep the status of their tickets up to date! This enables the CMG to monitor the progress of your ticket and coordinate UKCA work.

Various UM Ticket states are available, including: New, Assigned, In Progress, review, approved and committed.

When work on a ticket begins one should update their ticket with the "start work" action. The ticket state will then become "In Progress".

A "stop work" action is also available for "In Progress" tickets should developers want to 'suspend/hibernate' work and move a ticket state back to "Inactive".

Create a Branch

All development work must be done on a branch. Initially this can be the main FCM repository you use to run the Unified Model (i.e. PUMA for NCAS, UKMO repository for Met Office staff). A development (dev) branch can be created either via the fcm gui, the command line fcm branch --create or via the utility script create_branch if available.. The UM trac ticket number (not the UKCA trac ticket number) should be provided while creating the branch either by using the appropriate field in fcm gui (or create_branch) or by using the --ticket option to the fcm branch command. This makes it much easier to connect code changes to the reasons for those changes and visa versa.

Branches should be created from the latest UKCA development version. When creating branches from a release, use the FCM keyword for that release.

Remember that a branch provides a way for you to manage your changes in a controlled manner without changing the trunk. Changes you commit to your branch are permanently and centrally managed and are viewable via the PUMA/UKMO TRAC systems. It is a good idea to regularly commit your changes from your working copy into your branch as this ensures changes from your working copy won't get lost. You can always go back to a previous revision of a branch if you need to. In addition if you have checked a branch in, someone else can see your changes and run with

When you create branches, please use a meaningful name. "My_Branch" or "Fix" are examples of names to avoid. By using a meaningful name, if someone else is using your branch, it is easier to understand what the branch is for and whether it is appropriate for the experiment they are running.

Also please do not combine multiple changes on a single branch unless absolutely necessary. Ideally each ticket should describe a single change such as a new chemistry scheme or a better wet deposition change and should be associated with a single branch. Sometimes this may not be possible, if for example a change you are developing relies on another change which is not yet lodged and also conflicts with your change. If this is the case please seek the advice of the code management group immediately. By keeping code changes separate they are easier to test, review and manage. In addition, it makes it possible to prioritise some changes over others and prevents a problem with one change causing other updates to be delayed.

Updating to a new release

The Unified model usually has 3 to 4 releases per year. Once a year, the code management group will agree a specific UM version to be the 'UKCA development version' for the coming year and provide standard jobs for testing. You can continue to develop and test the code at the original UM version you created your branch at, but before review you will need to update it to the latest UKCA development version. If the code is at an early stage of development, it may be best to update it straight away. If however, you are in the middle of doing longer integrations, it may be best to wait until you are confident in the results at your original release. In either case update your branch as follows:

- Login to UKCA Trac system, update the Milestone field of the ticket and save.
- Create a new branch from the current stable version of the trunk using the appropriate fcm keyword as described in the previous section.
- Ensure all changes in the working copy of your old branch are committed to your old branch.
- Merge your old branch into the new branch.
- If any files are in conflict (marked by a 'C' in the report of changes during the merge) then you need to resolve these.
- Commit the changes to the new branch.
- The new branch is then ready for further development and testing.

Develop your Change

A few things to bear in mind when developing your changes,

- Follow the appropriate coding standard, see UMDP3. Code which is not written to this standard is not allowed in the trunk and the Met Office does not have resources to rewrite everyone code to this standard. This is a living document and frequently updated so please check the latest version from the collaboration wiki.

- Ensure commit messages to your branch are meaningful. They are visible to everyone using the system.
- You don't have to commit to your branch to test a change. You can compile from a working copy, albeit final testing prior to submission for review should be with code committed to a branch, so it is 'identifiable'. However, frequent committing is encouraged as good practice.

Test your Change

Test your change appropriately and record the results. Standard Jobs are provided to further aid testing/code development. These will include both the key supported UKCA jobs and also some non-UKCA Met Office standard jobs such as limited area models used for NWP.

The User Test Framework (UTF) is provided for testing your changes prior to submission for review. This will cover much of what is required for completion on the Ticket summary. If adding new functionality, remember to test that your new code doesn't change results when it is switched off. If this is a bug fix or optimisation that does change the results of one or more standard jobs, please note this in your ticket.

[More discussion needed. Hand edits, namelists, STASHMasters, AncilMAsters].To achieve this consider the use of:

- Switches in the UMUI to control new code for small changes
- New section-versions for large changes (this is probably not the most suitable solution for most UKCA changes)]

As well as this you will need to test the change to show that it does what is intended when turned on and the impact of this change on results. The extent of the testing needed will depend on the complexity and impact of the change. A small bug fix may only need a small set of tests to show that for example it prevents a particular type of crash but has little impact on the results of a 10 day model run. A more substantial change such as adding a new chemical reaction or aerosol process will require more evaluation.

A normal requirement for code is that the same results can be obtained regardless of how many processors are used or what configuration they are used in. This test often finds bugs which other tests do not such as uninitialised memory or code which exceeds array bounds. At present the Newton-Raphson solver in UKCA does not meet this requirement but it is hoped that it will achieve this in the near future. At present this requirement will be ignored.

Document your Change

Any change will need documenting. If the documentation is fairly limited (i.e. for a small change with strictly limited impact) then the ticket will be an appropriate place for this. If the required documentation is more extensive, then a UKCA Trac wiki page should be used. If you are introducing a substantial scientific change, you should also consider producing a peer reviewed paper in which the change is described and validated. Secondly relevant UM Documentation Papers (UMDPs) source should be updated, by creating a UMDP branch and updating appropriately.

The convention for naming wiki pages associated with tickets is to replace "ticket/xxxx" with "wiki/ticket/xxxx" for ticket number "xxxx".

The wiki page can be created by including the markup wiki:ticket/xxxx (where xxxx is your ticket number) in your ticket and then following the link.

A sample wiki template (header) is provided for documenting large tickets.

Information required for tickets should include (but is not limited to)

- A description of the purpose of the change
- A description of what code is changed and a link to updated UMDP documentation.
- Test results
- A note of whether the change will lose bit-comparison with previous versions if the science options remain unchanged.
- UMUI/STASHmaster/ANCILmaster changes required. As external users cannot easily develop the UMUI, we recommend that the changes needed here are just explained as new namelist items required or items for deletion. A new or modified STASHMaster or ANCILmaster records will need to be detailed in the ticket.
- A note of any impact on memory use or runtimes
- Are you bringing in any code from external sources (e.g. from publicly available libraries)? If so, where has it come from, and what is the situation regarding licensing? Note that we cannot normally accept code into the UM with a GPL licence or from copyrighted sources such as Numerical Recipes.
- Any other information that may help reviewers or later users of the change.
- Please complete a Ticket summary template and append this to your ticket to aid the review process
- Use the wiki filename convention wiki:ticket/xxxx/TicketSummary where xxxx is the ticket number.

Review

Before the Met Office can accept changes from external collaborators for lodging, they will need to be reviewed. The purpose of the review is to check that the code meets the coding standards, is properly tested and is properly documented. Once a ticket has been reviewed and has permission from the code review group for suitability for lodging, it will then go through the Met Office's internal code review. This will normally be dealt with by Met Office staff, at least initially, but it is also possible that NCAS staff with Met Office accounts can also play a role. The code management group will suggest a suitable candidate for this role. Ideally by this point the CMG will also have agreed whether the code is suitable for lodging.

Assign for review to your agreed reviewer.

Make sure that the ticket contains enough information to make the code under review obvious - links to the latest branch revision should be sufficient.

The reviewer should check that:

- The code changes are understood and appropriately made.
- The documentation is sufficient to understand the code change and its impacts.
- Complete a review template and append it to the ticket with the wiki filename convention wiki:ticket/xxxx/SciTechReview where xxxx is the ticket number.

It is likely that the review process will involve iterations between code author and reviewer until documentation (including UMDP updates) and change are agreed to be of sufficient quality. The reviewer has the option to "reject & assign" back to the code author should the documentation/code not meet the required standards and major alterations/improvements are required.

Once the reviewer is satisfied, he/she should complete the approval section of the review template and pass on to the Met Office (if the CMG agree) for lodging. This will then enter the Met Office's code review system.

Close Ticket

Once the code is lodged in the Met Office trunk it can be closed. Before closing the ticket as fixed, the code reviewer should update the ticket with a note stating what UM version the code was lodged at (it may be later than the current development version) and then re-assign the ticket to the original author

Suitability for lodging

Not all developments will be suitable for lodging. The code management group needs to strike a careful balance between giving UKCA world leading capability and the limited resources available for maintenance of the code. Changes intended for the trunk should:

1. be expected to be used by a reasonable number of people (e.g. a chemistry scheme for Venus might be better maintained on a users branch)
2. not add undue complexity to the code
3. be of suitable scientific quality.

Of course the CMG should also be able to periodically delete code which is no longer used (if for example it is seen as obsolete).

Note for Met Office developers of UKCA

Met Office staff who are developing code for UKCA will be able to skip the external review stage if they plan to get the code through the internal Met Office standard sci/tech and code/system reviews themselves. However, they should still initially develop the code at the agreed development version not the latest release (so that NCAS developers can use their code) and also they should test

it with the same set of UKCA jobs as external developers are required to. In addition they will also need to get code management group permission for the development in the same way as external developers do. They will therefore have to open a ticket on the UKCA trac system. This has the advantage that there is a single trac database of ongoing UKCA development both inside and outside the Met Office.

Glossary

Code management group – a group that is responsible for the day to day collaborative code management of UKCA. Consists of Met Office and NCAS staff appointed by the UKCA executive committee. Members will be listed on the UKCA wiki

UKCA Executive –

UM development versions -

UM trunk – the master version of the code held at the Met Office. For code to be accepted into this code base, the Met Office has strict rules on code quality, testing, and documentation. One in the trunk it is available from the next MetUM release for all users.

The User Test Framework – a tool for automatically testing the impact a branch has on multiple UM configurations. Developed at the Met Office.

Additional documents to be prepared

- short and friendly guide to developing changes for students and new developers
- Step by step guide to the templates
- Templates: ticket description, ticket summary, review
- More detail on development versions and supported jobs (especially the amount of testing of the supported jobs)