CSR FAQs

Frequently Asked Questions about the CSR

Q: What is the CSR?
A: The CSR is a review body for changes being made in JDK releases. The letters "CSR" stand for "compatibility and specification review"; therefore the CSR focuses on reviewing specifications (as opposed to implementations) with an emphasis on long-term compatibility impact. Besides compatibly review, review of a specification includes but is not limited to abiding by naming conventions, clear description of semantics, appropriate use of language features, and so on. The compatibility review is not strictly limited to specifications; some implementation-only changes with compatibility impact merit CSR review as well.

Q: Who are the CSR group members?
A: The current membership of the CSR group is listed in the OpenJDK census. The members of the CSR are experienced in JDK development and have deep knowledge of one or more areas of the platform. Multiple members of the CSR group have more than one releases of "double triple" bug fixing activity, authoring more than 100 changes and reviewing more than 100 changes by others. Updating the membership of the CSR group will follow the OpenJDK group procedures.

Q: What kinds of compatibly does the CSR look after?
A: The CSR looks after source, binary, and behavioral compatibility. Binary compatibility is the ability of existing binaries to link. Source compatibility concerns whether or not existing code still compiles and if it still compiles, if it compiles into an equivalent binary. Behavioral compatibility involves operational equivalence; with "the same" inputs, does a program behave "the same way" before and after a change. More detailed and nuanced discussion of these compatibility concerns can be found in "OpenJDK Developers' Guide, Version 0.777".

Q: What sort of changes require CSR review?
A: Any change to a JDK interface meant to be used outside of the JDK itself requires CSR review. In this context "interface" isn't limited to the Java programing language definition of an interface, but encompasses the broader concept of a protocol between the JDK and users of the JDK. Examples of interfaces by this definition include:

- Changes to public exported APIs in java.* and javax.* packages.
- Changes to public and exported APIs in jdk.*packages.
- New language updates to the Java Programming Language
- New structures in the Java Virtual Machine Specification
- Adding or removing a command in $JDK/bin
- Adding, removing, or changing a command line option; for details about evolving HotSpot command line options see Hotspot Command-line Flags: Kinds, Lifecycle and the CSR Process
- Using or defining an environment variable
- Using or defining a new file format or wire format
- Changing or defining a new system or security property

Interfaces that are experimental or for diagnostic purposes do not need to go through CSR process, but the CSR process may be employed if feedback from the CSR reviewers is desired.

Q: What are common types of feedback for the CSR to give?
A: Common outcomes for a proposal after review by the CSR include:

- Approved with no comments; this is most likely for small requests.
- Deferring CSR review until additional technical reviewers examine the proposal.
- Approving the proposal, but with comments affecting the code review. For example, the CSR may find issues with the javadoc that do not change the specification, etc.
- Approving the proposal, on the condition that a release note is written for the change.
- Substantive comments that require more extensive changes to the request.

Q: Timing wise, when do I need to file a CSR request?
A: A CSR request needs to be filed and approved *before* the corresponding change is pushed to a JDK release line of development. In exceptional circumstances, the need for a CSR review may be recognized only after a push has already occurred. In such cases, a retroactive CSR review can be conducted. The results of such a retroactive review may require updates to the change, up to and including complete removal of the change.

Q: How long does a CSR review take?
A: The size of CSR requests varies over several orders of magnitudes. Large requests should expected to take longer to review than small ones. The CSR strives to complete its initial review of a proposal within one week. If the CSR has feedback, the time its takes for an engineer working on the proposal to respond to and act upon the feedback may take more than one week after the proposal was submitted to the CSR.

Q: I have a deadline coming up. Can I ask for the CSR for an expedited review?
A: Engineers are responsible for factoring in reasonable amount of time for a CSR review ahead of any integration deadline, anticipating that the CSR might have feedback which requires the proposal to be updated. While engineers are free to request an expedited review from the CSR, the CSR is free to decline such requests.

Q: If my change needs a CSR review and a code review, which should I do first?
A: To take a common case of a Java API change, there is some overlap between the factors considered in a general code review and the factors considered by the CSR when reviewing the specification and compatibility impact. (CSR members often participate in code reviews in addition to their reviews in CSR roles.) An engineer may choose to run the CSR process and code review in parallel, but feedback from either channel may be received which requires updates to the proposal in the other channel. If an engineer chooses to sequence code review and CSR review, to minimize latency the process expected to provide more feedback should be run first.

Q: Who should be a reviewer on a CSR proposal?
A: One or more engineers with expertise in the areas impacted by the proposed change should review the CSR request and be listed as a reviewer before the proposal is reviewed by the CSR membership. (These engineers may or may not be Reviewers on the corresponding JDK project.) It is appropriate to ask a CSR member to review a request in a area where he or she has expertise, but it is not necessary for a CSR member to review a request before the CSR body considers it. To encourage wider reviews, it is preferable if the CSR chair is not the only reviewer of a CSR request. The CSR may request a proposal be reviewed by additional engineers before further considering the request.

Q: Should people providing feedback via CSR be listed a reviewers when a changeset is pushed?
A: If CSR reviews prompts modifications to what gets pushed, it is reasonable to include as the CSR members providing the feedback among the reviewers of the changeset.

Q: If I don't agree with the outcome of CSR review, what recourse do I have?
A: The CSR review strives to reach a consensus on the appropriate engineering outcome for a proposal, including what modifications to a proposal are necessary to bring the proposal in line with overall JDK goals. If such a consensus is not reached, since the CSR process is used at the request of the Lead for a JDK release project, appeals about the CSR's determination of a request for a particular JDK release can be made to the Lead of the release in question.

Q: If the text of the javadoc of a public exported API is changing, is a CSR request needed?
A: A CSR request is required if the *specification* of a public exported API. Not all javadoc updates are specification changes. For example, typo fixes and rephrasings that do not alter the semantics of the API in question do *not* require CSR review.

Q: What is the JDK's policy on deprecating items?
A: As of JDK 9, JDK APIs use enhanced deprecation (JEP 277). In summary, if an API is problematic in one or more ways, it can be marked as @Deprecated to discourage its use. If there are plans to remove the API in the following release, then it should be marked as @Deprecated(forRemoval=true). If an item is deprecated for removal in the next release, the bug to implement the removal should be filed by the time the CSR request is approved. Deprecated items should have a corresponding @deprecated javadoc tag. The @deprecated text should include a reference to a recommended replacement API, if one exists, and note any salient semantic differences with a replacement.

Q: What is the relationship between a CSR and a JEP?
A: A JEP (JDK Enhancement-Proposal) initially describes a project to update some aspect of the JDK. The exact details of the updates are usually not yet known when the JEP begins. As the JEP matures, the updates to the JDK associated with the JEP are pushed under on more more changesets. Each changesets that involves a specification change (or sufficiently large compatibility impact) would also require CSR review.

Q: Do incubating modules shipped with the JDK require CSR review?
A: An incubating module (JEP 11: Incubator Modules) contains non-final APIs shipped with the JDK to gather feedback. While such APIs don't have the same long-term prospects as Java SE APIs, they are still widely usable and subject to CSR review.

Q: How many CSR requests are typically in a release?
A: For JDK 9, there were approximately 900 CCC requests (the CCC being the internal predecessor to the CSR), approximately 90 JEPs, and over 14,000 bug fixes. For JDK 10, there were about 120 CSR requests and over 2000 bug fixes in the release. Having a single-digit percentage of fixes need to go through compatibility and specification review has been consistent over time.

Q: I found a case where the specification and implementation disagree; how is this resolved?
A: The proper resolution depends on the nature of the discrepancy. However, if it is technically reasonable for either the specification to be changed to match the implementation or for the implementation to be changed to match the specification, the preference is to change the specification to match the implementation. This preference better preserves behavioral compatibility for users of the API in question. Note this is only a preference and at times it is unreasonable to allow erroneous behavior of the implementation, once recognized, to persist.

Q: How do I create a CSR ?
A: Do not directly create a CSR from the Create Menu. JIRA will let you do this right up until the moment you try to save it and find your typing was in vain.
Instead you should go to the target bug, select "More", and from the drop down menu select "Create CSR". This is required to properly associate the CSR with the main bug, just as is done for backports.

Q: How should I get CSR review for multiple release trains?
A: Say you want to get an interface change into JDK N and later decide the change is also desirable and appropriate for the JDK (N-1) update release and that update release is using the CSR process. First a CSR should be created from the main bug targeted at JDK N. Afterward, if a backport of the main bug covering JDK (N-1) does not already exist, a backport of the main bug covering JDK (N-1) should be created. Then, a CSR can be created from that backport.

Q: When does a CSR need to be filed for a purely behavioral change?
A: Using qualitative terms, a CSR for a behavioral change should be filed if it is estimated enough developers or users would be sufficiently impacted by the change that it should get additional consideration or documentation. A judgment call is involved. If assistance is needed in determining whether or not a CSR needs to be filed, ask the CSR representative for that area or the CSR chair.

Q: How do I get my pre-formatted text (etc) to display properly ?
A: You can use markdown as is used in the JEP2 process. The cheat sheet recommended by JEP 2 may be found here : `http://daringfireball.net/projects/markdown/basics`

(Future FAQ items will discuss the logistics of working with CSR issues in JBS.)