# Connect5 Best Practices

This document outlines the best practices for designing solutions using Connect5. Even though this document is specific to request/response integration styles, the practices are applicable to all integrations styles.

1. Solution architects should produce a Visio outlining the routes, formats and chaining of routes. Visio basic flow chart or EIP stencils can be used [//corp-fs01/software/Pilotfish/visio/EIP\_Visio5\_stencil.zip].
2. The number of routes in a given interface should be no more than N+1. Where N is the number of target system. With Processor and XSLT transformation in places at the target and source, almost all the solutions can be designed by applying the rule.
3. When chaining routes, they should share formats. See Figure 1.
4. Source side of the route must have processor to validate all incoming messages; if the messages don’t conform an exception must be thrown. Avoid having targets to route error conditions. Groovy processors can be used which gives scripting ability to write adhoc validations.
5. If there is more than one target avoid using default routes. Instead specifically check for routing predicates in the source, if they are not available throw exception.
6. All routes except for error routes must have transaction monitor registered. Make sure error routes don’t throw an exception.
7. All transaction monitors should be configured to include attributes. Transaction data can be included if you think error route can consume the transaction data.
8. If the protocol supports timeouts then it must be specified in the Listener and Transport configuration.
9. All modules – Listeners, Transports & Processors, must have unique names and the names should be describing the task it does.
10. All XSLT Templates must be cached.
11. If applicable try to re-use processors configuration and formats.
12. Leverage XSLT import/include feature instead of copying templates.
13. Populate message id in the source of the very first route.
14. Xalan java extensions is available with Xalan interpreted template engine; Xalan compiled template engine does not support java extensions.
15. Avoid using Xalan Compiled engine instead use Saxon.

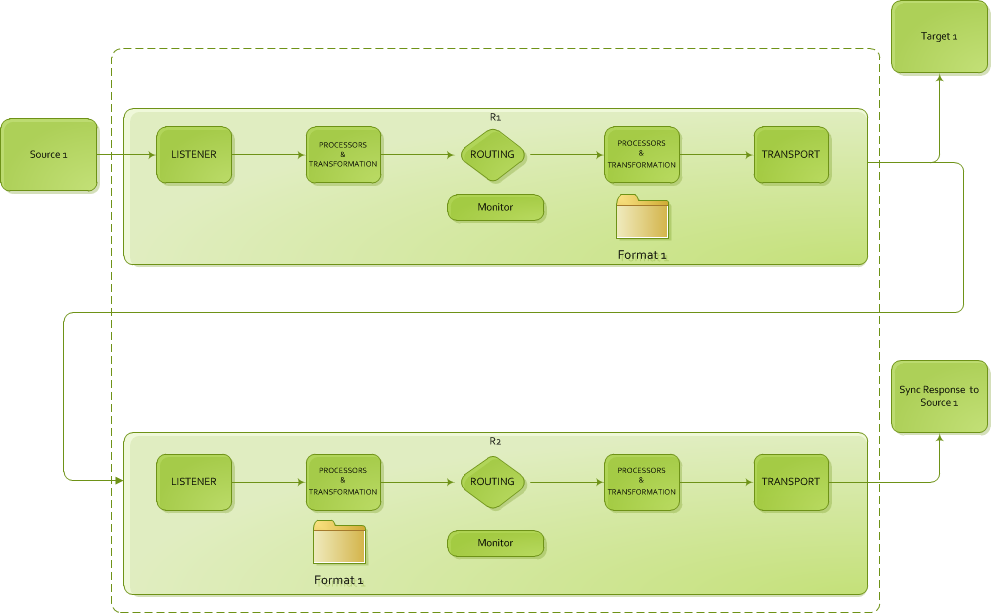


Figure 1