Overview
HDFS-1742 and HDFS-1179 both suggest the need for a mechanism for external applications (e.g. search engines, filesystem mirroring) to keep track of edits to the filesystem state.

Desired Client Interface
Methods are available in DFSClient for a client to specify a root directory to watch. A stream from which edits can be read is returned. (We will need to expose the edit log ops in the public API and ProtoBuf them.) The client can choose to see only certain types of edits and can select a desired starting point for the stream (transaction ID or timestamp). Clients will be unable to read edits to parts of the filesystem to which they do not have read access. More advanced pub-sub functionality (e.g. partitioning) can be achieved by writing a utility to dump edits into a system like Kafka.

Possible Designs
Both of the following design options are highly available. They would both involve maintaining the edit logs for some user-specified period of time (e.g. 24 hours). A user application would be responsible for remembering its position in the edit stream if it crashed. Filters specified by the client can be sent to the servers so that only the desired edits would need to be sent over the wire.

Design Option #1: The client communicates directly with the NameNode.

Pros:
- Security policies are much easier to enforce when the NN is involved
- Should be the easier of the options to implement

Cons:
- Extra load on the NN – may not be too significant since it's a read-only workload that involves minimal processing

Design Option #2: We add some non-voting JournalNodes that are sent edits from the NameNode but are not considered when determining whether a quorum has been reached (they do not need to respond to the NameNode). The client then reads edits from these JNs.

Pros:
- No additional NN load

Cons:
- Requires client to be using QJM – or we need to refactor QJM to support a setting where there are only non-voting members (i.e. QJM is enabled solely for the purpose of inotify and there must be another edit storage system like NFS in place)
- Requires major (and probably undesirable) refactoring to support security policies since JNs know nothing about access privileges now – this may be tolerable if e.g. we only allow the superuser to read edits