Client

logProcessMessage(String processName, String message,
MessageCategory category, MessagePriority priority)
logJobProgress(JobID, ProcessName, int percent)
logJobProgress(JobID, ProcessName, int total, int completed)
logJobStart(JobID, ProcessName)
logJobStart(JobID, ProcessName, int totalTasks)
logJobComplete(JobID, ProcessName)
logProcessStart(processName)
logProcessShutdown(processName)
logProcessStateChange(processName, ProcessState state)

UI

Joe's ideas:

GetAllMessages(processID)
GetAllMessagesBasedOnPriority(messagePriority)
GetAllStartupAndShutdownMessages()

GetLastMessage(processID)
GetProgress(processID)
GetAllProcessesSummary()

Devon's Ideas:

List<Message> getAllMessages(processID)
List<Message> getMessagesByPriority(MessagePriority priority)
List<Message> getMessagesByCategory(MessageCategory category)

List<int> getProcesses()
List<int> getJobsInProcess(processName)
Process getProcessById(int ProcessID)
Map<int,string> getProcessIdToProcessNameMap()

Kyle's Ideas

For each exposed method in this section, I give a little explanation of where I think it will be used. I am using the requirements doc to help me create these. I also figure we might as well pass stored procedure arguments so they can be evaluated while the data is fetched. This includes record limits and sorting arguments. I wonder if we should include some kind of "sort by" string argument in the exposed methods. That way we wont need a million exposed methods to sort data different ways.

List<ProcessSummary> getProcessSummaries(int startLimit = 0, int endLimit = 200)

Where used:

This method is for the main page of the UI. Joe wants a summary of all the processes.

What it returns:

A ProcessSummary includes a Process object with it's state (red, yellow, green) and all its other members, a job object with it's status (if a process has logged a job that is being worked on), the last logged message object for this process and anything else we want to include in the summary.

Argument notes:
If there are more than the default endLimit of summaries, I figure that as the page scrolls to the bottom, we just fetch more summaries through an ajax call. Default sorting should be done by ProcessState first, maybe the last message Priority second, and the last message timestamp third?

List<Job> getProcessJobs(int process_id, int startLimit = 0, int endLimit = 10)

Where used:
This method might be called when a user clicks on a summary on the main page or when a user is examining a process in it's view.
What it returns:
Job Objects. Just like the database table. Sorted by Job status, maybe the last message Priority second, and the last message timestamp third?
Argument notes:
Should always fetch by id for fast comparisons. the limits could behave like the getProcessSummaries limits.

List<Message> getMessagesByProcess(int process_id, int startLimit = 0, int endLimit = 30)

Where used:
Called when user clicks on a process?
What it returns:
Returns Message objects like in the database. Sorted by message Priority, and message timestamp next
Argument notes:
Should always fetch by id for fast comparisons. the limits could behave like the getProcessSummaries limits.

List<Message> getMessagesByJob(int job_id, int startLimit = 0, int endLimit = 30)

Where used:
Called when user clicks on a job?
What it returns:
Returns Message objects like in the database. Sorted by message Priority, and message timestamp next
Argument notes:
Should always fetch by id for fast comparisons. the limits could behave like the getProcessSummaries limits.

List<Message> fetchMessages(int startLimit = 0, int endLimit = 30)

Where used:
The message list view
What it returns:
Returns Message objects like in the database. Sorted by message Priority, and message timestamp next
Argument notes:
the limits could behave like the getProcessSummaries limits.

List<Message> fetchMessagesWithCategory(String categories, int startLimit = 0, int endLimit = 30)

Where used:
The message list view and when the user wants to view startup and shutdown messages
What it returns:
Returns Message objects like in the database. Sorted by message Priority, and message timestamp next
Argument notes:
We could pass a string of comma separated categories and fetch Message records that are in that list. the limits could behave like the getProcessSummaries limits.

ENUMS

- MessageCategory
  - ERROR
  - INFORMATION
  - STATE_CHANGE
  - START
  - STOP
  - PROGRESS
- MessagePriority
  - HIGH
  - MEDIUM
  - LOW
- ProcessState
  - RED
  - YELLOW
  - GREEN

Process

- has a list of messages
- has a status
- has a list of jobs
- has a name

Message

- has a category
- has a priorty
- has messageText
- has a ProcessId
- has a date