Tutorial for Parallel Request/Read Functions

If you have connections to a lot of scanners (more than six or so), the total round-trip time for requesting and reading a profile from each one can take a while. Fortunately, there's an easy way to parallelize requests and reads from groups of scanners. The key is to send all the requests at once, and then read all the profiles back at once. This causes all the scanners' processing time and network travel time to happen simultaneously.

To do this during Synchronized Scanning Mode, do something like this:

1. `jsEnterEncoderSyncMode()` or `jsEnterTimeSyncMode()` for each connection to start Synchronized Scanning Mode.
2. `jsSendMultipleProfileRequest()` for each connection.
3. `jsReadMultipleProfiles()` for each connection.
4. `jsExitSyncMode()` for each connection to stop Synchronized Scanning Mode.

For snapshot scanning, care must be taken to ensure that the scanners don't see each other's lasers. Do something like this:

1. Call `jsSendProfileRequestN()` for each connection.
2. Call `jsReadProfileN()` for each connection.

```
#include <vector>
#define JCAM_STATIC_LIB
#include "jcam_dll.h"

using namespace joescan;

//Returns the first failed connection if there is one
JCONNECTION readProfiles(std::vector<JCONNECTION> &connections, std::vector<jsProfile> &profiles)
{
    for(i = 0; i < connections.size(); ++i)
    {
        //Request one profile from each scanner
        switch(jsSendMultipleProfileRequest(connections[i], 1))
        {
            case 0:
                break;
            case INVALID_PARAMETER:
                printf("Invalid parameter to jsSendMultipleProfileRequest().\n");
                return connections[i];
            case SCANNER_FAILURE:
                printf("Scanner failure during jsSendMultipleProfileRequest().\n");
                return connections[i];
        }
    }

    //Make sure there's enough storage space for one profile from each scanner
    profiles.resize(connections.size());

    for(i = 0; i < connections.size(); ++i)
    {
        //Read one profile from each scanner
        switch(jsReadMultipleProfiles(connections[i], &profiles[i], 1))
        {
            case 0:
                break;
            case INVALID_PARAMETER:
                printf("Invalid parameter to jsReadMultipleProfiles().\n");
                return connections[i];
            case SCANNER_FAILURE:
                printf("Scanner failure during jsReadMultipleProfiles().\n");
                return connections[i];
        }
    }

    //No failed connections, return NULL.
    return 0;
}
```