```
#ifndef _EDU_WSU_GRIDSTAT_COMMAND_STRUCTS
#define _EDU_WSU_GRIDSTAT_COMMAND_STRUCTS

#include "commConstants.idl"
#include "hierConstants.idl"

module edu
{
module wsu
{
  module gridstat
  {
    module command
    {
      // Sequence of base types
      typedef sequence<unsigned long,
                       CommConstants::MAX_MODES> IntervalSeq;
      typedef sequence<unsigned short,
                       CommConstants::MAX_MODES> PrioritySeq;

      // Holder for the interval
      struct IntervalInfo
      {
        unsigned long intervalLow;
        unsigned long intervalHigh;
      };

      // Holder for QoS requirement
      struct QoSInfo
      {
        // These are the desired QoS properties desired
        unsigned short desiredSecurity;
        unsigned short desiredRedundancy;
        unsigned short desiredPriority;
        unsigned short desiredTimeliness;

        // Subscription Interval properties
        IntervalHolder interval;

        // These are the worst case QoS properties
        unsigned short minimumSecurity;
        unsigned short minimumRedundancy;
        unsigned short minimumPriority;
        unsigned short minimumTimeliness;
      };

      // Sequence to hold QoS requirements for each mode
      typedef sequence<QoSInfo, CommConstants::MAX_MODES> QoSSeq;

      // Holder for information that a publisher provides when
      // publishing a status variable
      // NOTE: Total message length = messageLength+patternLength
      struct PublicationInfo
      {
        string pubName;    // The name of the publisher
        string variableName;
        unsigned short type; // Static, dynamic, or compound
        unsigned short dataType; // Int, Float, Boolean, User defined
        unsigned long userDataType; // If user defined the data type
        unsigned long messageLength; // How many bytes
        unsigned long patternLength; // How many bytes
        unsigned short numPattern; // How many patterns
        unsigned short priority; // Alert=0, 1-5 High-Low priority
        IntervalInfo interval;
      };

      // Holder information an identifier for a single subscription
      struct SubscriptionIdInfo
      {
        string srcCloud;
        string pubName;
        string variableName;
        string dstCloud;
        string subName;
        unsigned short redundantId;
      };

      // Holder information for a single subscription
      struct SubscriptionInfo
      {
        unsigned long variableId;
        string dstSRName;
        unsigned long pathId;
        unsigned short redundantId;
        IntervalSeq subInterval;
        unsigned short dataType; // Int, Float, Boolean, User defined
        unsigned long userDataType; // If user defined the data type
        unsigned long messageLength; // How many bytes
        PrioritySeq priority;
        unsigned long modes;
      };

      // Holder for information that a subscriber provides when
      // subscribing to a status variable
      struct SubscribeInfo
      {
        string subName;
        string pubName;
        string variableName;
        unsigned long modes;
        unsigned short dataType; // Int, Float, Boolean, User defined
        unsigned long userDataType; // If user defined the data type
        QoSHolder QoS;
      };

      // Holder information for the id of a event channel
      struct EventChannelIdInfo
      {
        string srcSRName;
        string dstSRName;
      };

      // Holder for information for a event channel between two SRs
      struct EventChannelInfo
      {
        EventChannelIdInfo linkId;
        string srcAdr;
        unsigned long srcPort;
        string dstAdr;
        unsigned long dstPort;
        unsigned long level;
        unsigned short bandwidth;
        unsigned long latency;
      };

      // Sequence to hold a number of subscriptions for a cond. fun
      typedef sequence<SubscribeInfo,
                       CommConstants::MAX_SUB_COND> SubscribeSeq;
      typedef sequence<SubscriptionInfo,
                       CommConstants::MAX_SUB_COND> SubscriptionSeq;

      // Holder for information for a condensation function
      struct CondensationRequestInfo
      {
        string creator;
        string placementEdge; // Request cond fun be placed here
        PublishHolder pubHolder;
        boolean inFilterLow;
        float inFilterLowValue;
        boolean inFilterHigh;
        float inFilterHighValue;
        boolean outFilterLow;
        float outFilterLowValue;
        boolean outFilterHigh;
        float outFilterHighValue;
        string calculatorURI;
        string calculatorClassName;
        unsigned short triggerType;
        unsigned long triggerVar1;
        unsigned long triggerVar2;
        SubscribeSeq subSeq;
      };

      // Holder for information for a condensation function
      struct CondensationSetupInfo
      {
        string creator;
        PublishHolder pubHolder;
        boolean inFilterLow;
        float inFilterLowValue;
        boolean inFilterHigh;
        float inFilterHighValue;
        boolean outFilterLow;
        float outFilterLowValue;
        boolean outFilterHigh;
        float outFilterHighValue;
        string calculatorURI;
        string calculatorClassName;
        unsigned short triggerType;
        unsigned long triggerVar1;
        unsigned long triggerVar2;
        SubscriptionSeq subSeq;
      };
    };
  };
};
};

#endif // define _EDU_WSU_GRIDSTAT_COMMAND_STRUCTS
```