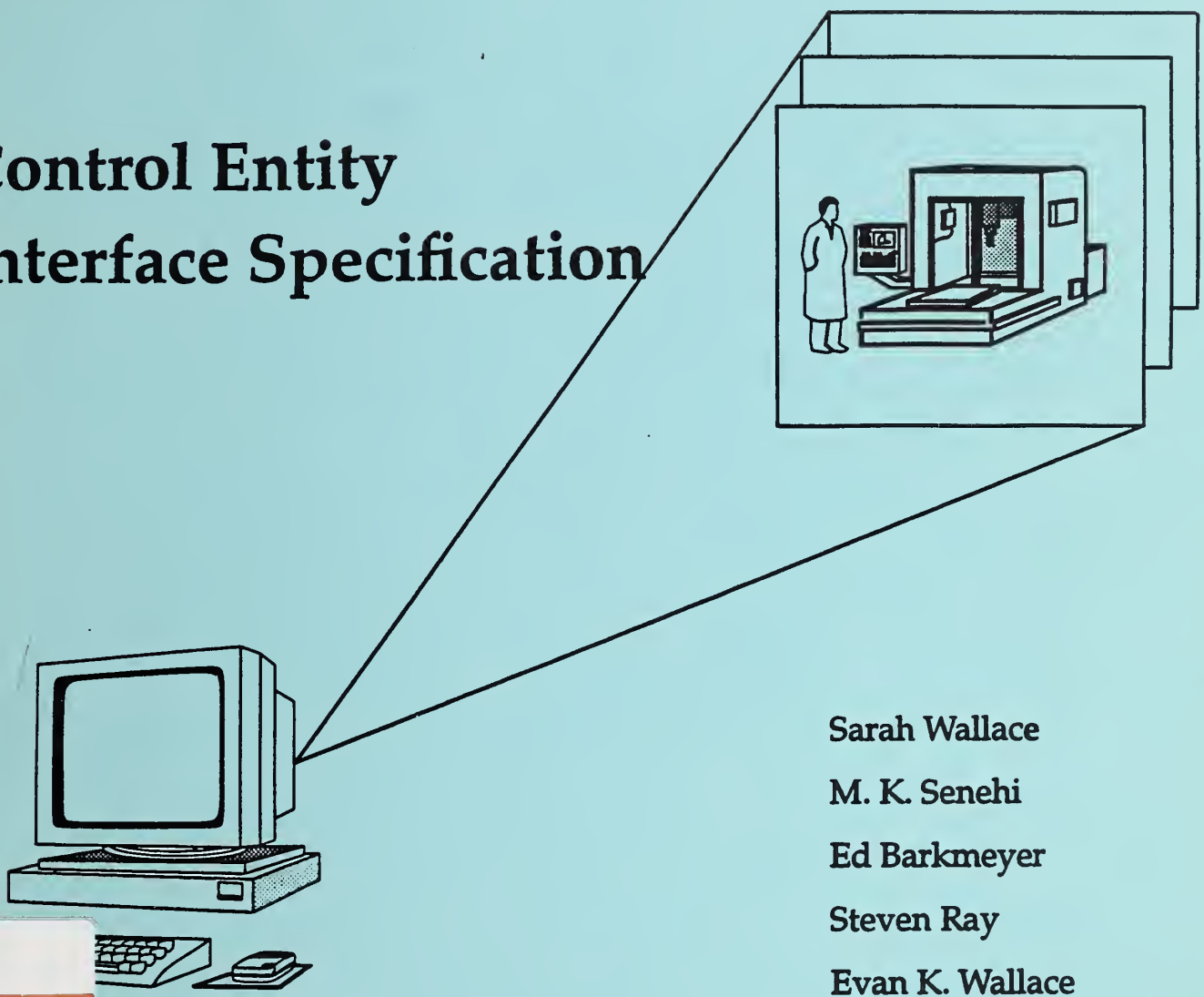
NISTIR 5272
September, 1993

Manufacturing Systems Integration

Control Entity Interface Specification



Sarah Wallace
M. K. Senehi
Ed Barkmeyer
Steven Ray
Evan K. Wallace

United States Department of Commerce
Technology Administration
National Institute of Standards and Technology
Manufacturing Engineering Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 5272
September, 1993

Manufacturing Systems Integration

Control Entity Interface Specification

Sarah Wallace

M. K. Senehi

Ed Barkmeyer

Steven Ray

Evan K. Wallace

U.S. Department of Commerce
Ronald H. Brown, *Secretary*
Technology Administration
Mary L. Good, *Under Secretary for Technology*
National Institute of Standards and Technology
Arati Prabhakar, *Director*

Table Of Contents

List of Figures

List of Tables

1 Introduction

A major activity of the National Institute of Standards and Technology (NIST) Manufacturing Systems Integration (MSI) project is the development of an open architecture that incorporates an integrated production planning and control environment with provisions for the detection and recovery from anomalous situations. This document is concerned with defining the details of the interfaces for control entities which are incorporated into an integrated system that conforms to the NIST MSI architectural model as revised in 1992.

The purpose of this document is twofold: to document the progress and current status of the MSI architecture's control entity interfaces and to provide designers and implementors with specifications for an MSI architectural compliant control entity. The concepts presented in this document were developed by the MSI architecture committee and represent the consensus of that committee as of June 1992. In 1992 the architecture committee members were Ed Barkmeyer, Steven Ray, M. Kate Senehi, Evan Wallace and Sarah Wallace.

The remainder of this document proceeds as follows:

- Section 2 provides an overview of the MSI project, a discussion of pertinent aspects of the MSI architecture, and a discussion of the assumptions which have been made concerning the environment within which the system is operating,
- Section 3 discusses production plans in the context of the interaction between planners and job controllers and the recovery from anomalous situations,
- Section 4 discusses generic data objects which are common to several of the 5 control entity interfaces,
- Sections 5 - 9 detail the 5 different control entity interfaces: the planning interface, the job control interface, the planning-to-job-control interface, the guardian planning interface and the guardian job control interface,
- Section 10 gives an example scenario of how the different messages from the 5 interfaces work together to provide for the detection of and recovery from anomalous situations,
- Section 11 concludes this document by discussing future extensions to this specification.

2 Background

This section provides an overview of the MSI project (its direction, goals, and current status) and discusses several aspects of the MSI architecture which impact the control entity interface concepts presented in this document. The architectural issues discussed include hierarchical control, plans and plan decomposition, the planning paradigm, the job control paradigm, provisions for anomaly detection and recovery, and the communications paradigm.

2.1 The MSI Project

The primary goal of the MSI project is to develop an open architecture which incorporates an integrated production planning and control environment with provisions for the detection and recovery from anomalous situations in order to enable flexible integration of manufacturing systems. There are three main thrusts within the MSI project to achieve this goal:

- the development of an open architecture which details the form and function of identified systems within the architecture, and the nature of interaction between them,
- the definition of information models describing the information to be shared among the different systems identified by the architecture, and
- the definition of communication interfaces detailing how and when systems within the architecture exchange and share information.

A cyclic three-phased approach has been used for the development of the architecture. An initial architecture – including information models and communication interfaces – was developed, implemented and tested in 1990 and 1991 (see [Senehi, 1991a] and [Senehi, 1991b] for details). This architecture incorporated an integrated production planning and control environment with no support for the detection or recovery from anomalous situations. In 1991 and 1992, a revised architecture was developed (incorporating the results of testing the initial architecture) and expanded to provide for the detection and recovery from anomalous situations. Work is currently underway to implement and test this revised architecture.

The reader is referred to [Senehi, 1992] for an overview of the revised architecture; a more detailed paper is forthcoming. [Barkmeyer, 1993] and [Ray, 1992b] present the information models developed as part of the revised architecture. This paper documents the communication interfaces that have been developed as part of the revised architecture.

2.2 Hierarchical Control

In the MSI architecture, a control entity is required to exist within a control hierarchy.

2.2.1 Control Hierarchy

The primary functions of a control hierarchy are to provide reliable channels for directing the start up and shut down of control entities, to localize planning and replanning activity, and to coordinate the manufacturing activity of interrelated equipment. A control entity may have, at most, one supervisor and any number of subordinates. A subsystem consists of a specific control entity and all of its subordinates.

A control hierarchy consists of three logical levels:

- The equipment level is the lowest level of control to which the MSI architecture applies. Within this level, there is a software complex which drives physical equipment. An

equipment control entity is responsible for the planning and execution of tasks by an individual device; it may execute only one task at a time. It may have internal subordinate software elements which perform primitive control tasks, but the characteristics of these internal interfaces are not specified by the MSI architecture.

- A workcell is a subsystem consisting of a collection of equipment viewed as a functional unit. It is coordinated by a single supervisory control entity designated the workcell control entity. The grouping of equipment into workcells is based upon many factors and is highly facility dependent. A workcell control entity may supervise equipment control entities directly, or subordinate workcell control entities, or some combination thereof. There may be any number of "levels" of workcell management within a control hierarchy.
- The top level control entity is the shop control entity. The subsystem it supervises consists of a set of workcell control entities in a manufacturing shop which are part of the current factory configuration. At present, the MSI architecture is limited to a single shop and there is only one control entity at this level. The primary difference between the shop control entity and other control entities is that the shop control entity processes orders; all other control entities process commands.

Each control entity will exhibit at most two interfaces to its supervisor and each of its subordinates: one for a planning function and one for a job control function. It is possible for an entire control hierarchy to support only the planning function, in which case each control entity exhibits only the planning interface to its supervisor and subordinates. Such a control hierarchy would be used to plan future shifts or to test different factory configurations by allowing what-if planning. If a control hierarchy exhibits the job control function, it must also exhibit the planning function.

2.2.2 Control Entities

control entity is a pair of different functional architectural entities which share the same area of focus; the two functions that a control entity performs are a planning function and a job control function. An entity which performs the planning function for a control entity is denoted a "planner" and an entity which performs the job control function for a control entity is denoted a "job controller". As currently defined by the MSI architecture, a planner supports resource allocation and scheduling; it may, but is not required to, support process planning and batching. In order to accommodate as many types of control entities as possible, the MSI architecture identifies a limited set of compatibility requirements for interfacing control entities.

Integrating a control entity into the MSI architecture may require five types of interfaces:

- the planning interface (commands and status for scheduling),
- the job control interface (commands and status for task execution),
- the guardian interface to planning (commands and status for human intervention and monitoring of scheduling activity)
- the guardian interface to job control (commands and status for human intervention and monitoring of task execution) and
- the planning-to-job-control interface (commands and status for rescheduling of tasks and anomaly recovery).

The job control interface and both guardian interfaces are required for any control entity to be integrated into the MSI architecture. If the control entity is implemented as two separate entities, one corresponding to each function, then the planning-to-job-control interface is required. If the

planning function for a control entity is implemented separately from the planning function for other control entities, then the planning interface is required for that control entity. A conceptual diagram depicting all five control entity interfaces is shown in Figure 1.. A more detailed discussion of when each interface is required can be found in Section 2.2.3 on page 12.

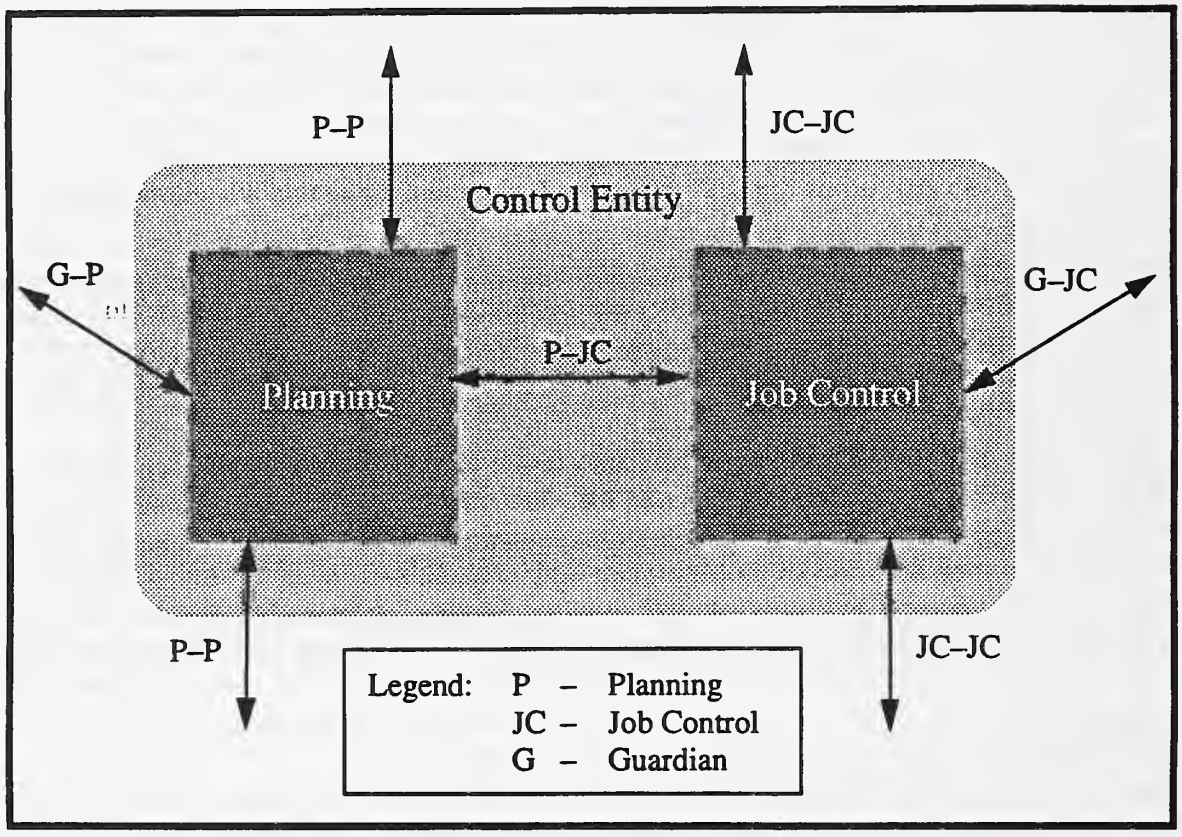


Figure 1. Control Entity Diagram

2.2.3 The Physical Architecture of a Control Hierarchy

The topology of the control hierarchy (i.e., the number of control entities and their inter-relationships) is determined by constraints and coordination requirements on the physical equipment and job controllers. There are no constraints placed on the planning function which require that it be implemented as a distributed system, or that it be implemented to physically correspond one-to-one to the job controllers, or that each control entity's planning function be embedded within the control entity. The only requirements placed upon an implementation of a control entity's planning function are:

- Each job controller in the control hierarchy has some agent which is responsible for creating and maintaining its schedule.
- Each job controller in the control hierarchy has some agent which is responsible for addressing planning errors which the job controller detects.

Given these two constraints on an implementation of a control entity's planning function, several valid implementations may be discussed.

Embedded Planner. It is valid within the MSI architecture for the planning function for each control entity to be embedded within the control entity itself. In this example, each planner is required to exhibit the planning interface. Each control entity is not required to exhibit the

planning-to-job-control interface because the control entity is not distributed; there is no external interface between the planning and job control functions. The planning and job control functions, however, must support the semantics of the planning-to-job-control interface. An example of such a system is depicted in as Figure 2.

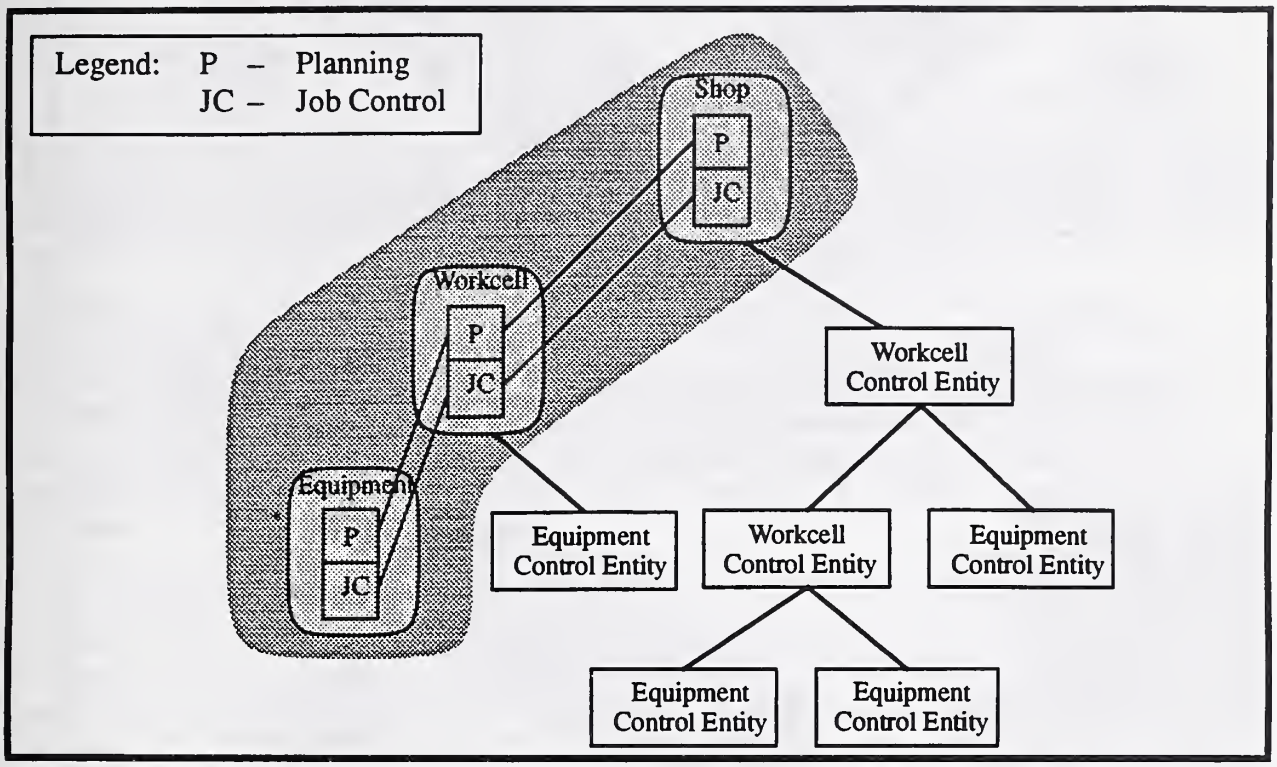


Figure 2. An MSI Architectural Compliant System with Embedded Planners

Distributed Planning Hierarchy. It is valid within the MSI architecture to have a distributed planning hierarchy which mirrors the control hierarchy (i.e., a planner for each job controller). In this example, the planner is required to exhibit both the planning interface and the planning-to-job-control interface because both the planning function and the control entity are distributed. An example of such a system is depicted in Figure 3.

Centralized Planner. It is valid within the MSI architecture to have a centralized planner which plans for every job controller in the control hierarchy. In this example, the planner is not required to exhibit the planning interface because it is not a distributed implementation. The planner is required to support the planning-to-job-control interface because each control entity is distributed; it must support one planning-to-job-control interface for each job controller in the control hierarchy. Each control entity is a conceptual entity only; there is no physical correspondence. An example of such a system is depicted in Figure 4.

Each of the systems discussed describes a homogeneous implementation of all control entities' planning functions (i.e., every control entity's planning function is implemented in an identical manner). It is also valid within the MSI architecture to have hybrid combinations of these systems.

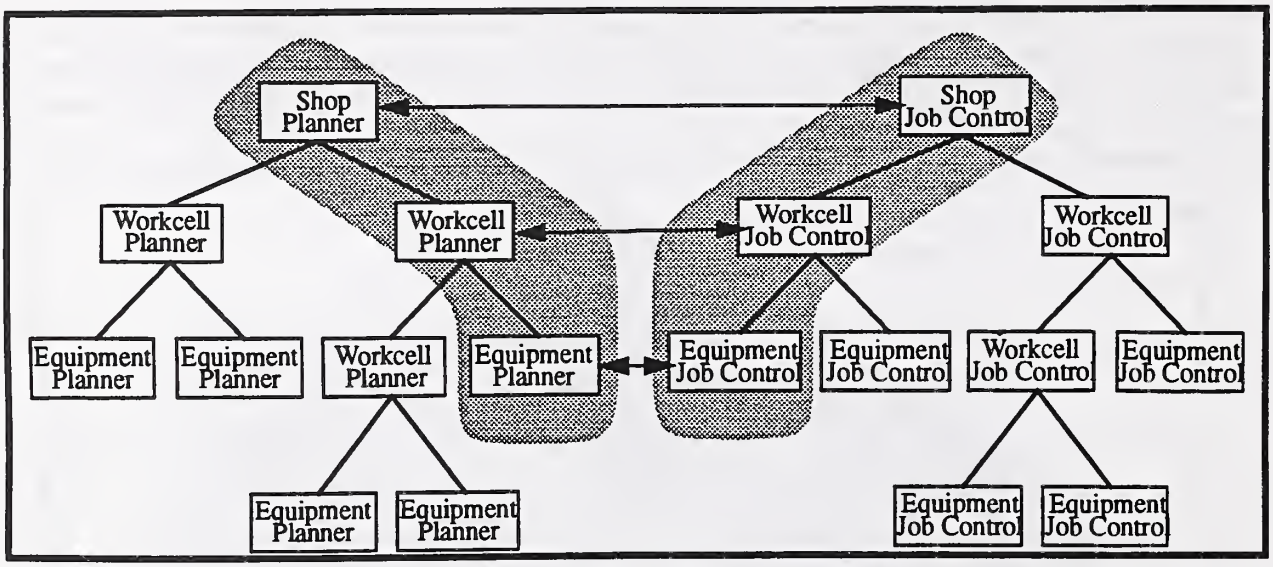


Figure 3. An MSI Architectural Compliant System with a Planning Hierarchy

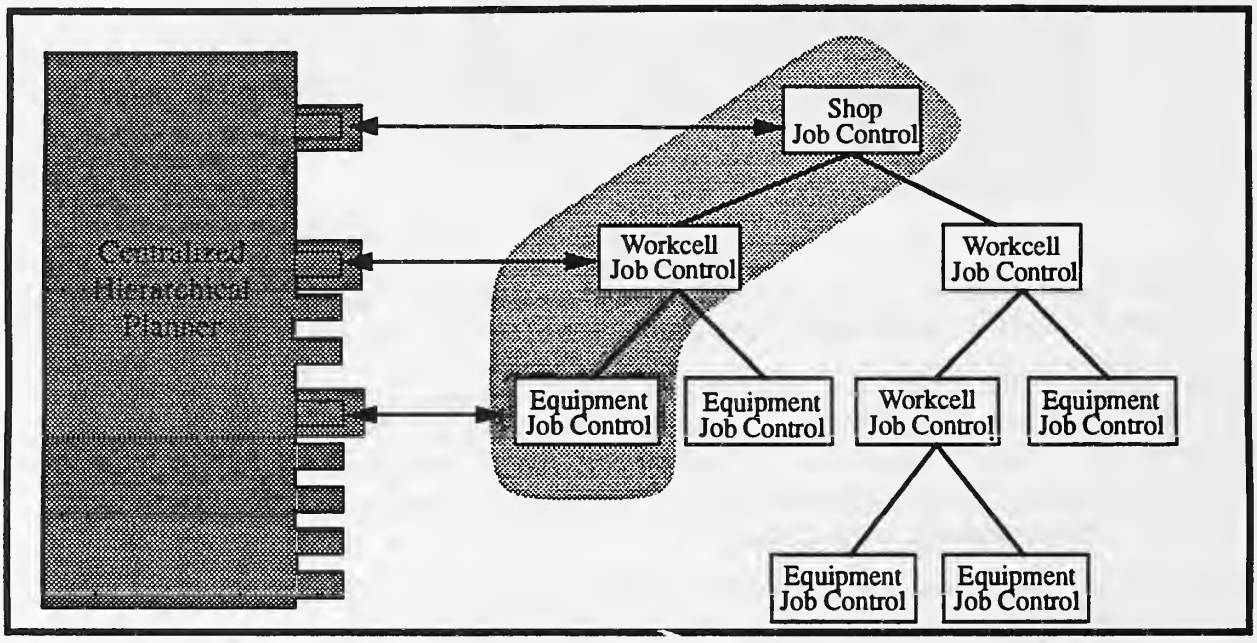


Figure 4. An MSI Architectural Compliant System with a Centralized Planner

2.3 Plans and Plan Decomposition

A plan is a recipe for performing a procedure; it contains a set of steps which provides sequencing information and details how to perform each operation within the procedure. In the MSI architecture, every manufacturing process begins with a plan and culminates in its execution.

2.3.1 Types of Plans

The MSI architecture recognizes three types of plans[1]: process plans, production-managed plans and production plans. A process plan is a generic recipe describing how to perform some

1. For more detailed information about plans, their representation and their decomposition, see [Catron, 1991] and [Ray, 1992a].

procedure in support of the production of some number (usually one) of products. A production-managed plan is an expansion of a process plan which supports the production of a required number of products using a given factory configuration; a production-managed plan reflects decisions about part mix, batching, lot sizes and material handling. A production plan is a refinement of a production-managed plan which identifies specific resources for each step and the times of their usage for that step.

Production-managed plans are the principle input into the planning function; production plans are the principle output. It is the planning function's responsibility to create production plans that correspond to the production-managed plans by choosing appropriate branches of the production-managed plan, selecting proper equipment and scheduling that equipment to perform selected steps.

Production plans are the primary input into the job control function, and the execution of the planned operation is the primary output. Production plans detail for the job control function how and when to perform a specified procedure. The job controller converts the plan into a sequence of commands to subordinate controllers or equipment which ultimately result in the motion of equipment and the execution of a manufacturing process. In anomalous situations, a planner may modify in-execution production plans in order to maintain a feasible schedule for the job controller.

2.3.2 Plan Decomposition

Because the MSI architecture dictates a control hierarchy and because production-managed plans and production plans assume a specific factory configuration, production-managed plans and production plans must be hierarchical. There will be a shop level production-managed plan to perform a specific procedure which will be scheduled by the shop level control entity creating a shop level production plan. The shop level production-managed plan will contain steps which detail how to perform the specified procedure in terms of workcell operations; each step may refer to a (subordinate) workcell and a subordinate production-managed plan which will be scheduled by some subordinate workcell to create a production plan for executing that step in that workcell. Each workcell production-managed plan, in turn, will contain steps which refer to subordinate workcell or equipment control entities and subordinate production-managed plans. This process of plan decomposition (or refinement for a lower level of control) will continue until, at the equipment level, a plan is developed which contains only primitive operations. The hierarchical production plans will then be executed by the job controllers who have been allocated to perform specific steps. During execution, each of the subordinate plans will be activated by a message on the job control interface.

2.4 The Planning Function

There are two primary functions of a planner: to schedule the execution of tasks for specific job controller(s) and to assist in anomaly recovery on behalf of those job controller(s).

2.4.1 Planner Negotiation

When more than one planning system is involved in the scheduling of an operation, planners use a negotiation scheme to schedule and reschedule the execution of steps. This negotiation scheme is similar to the contract negotiations that occur in the business world. A control entity, acting as a contracting party, will issue a request for bid to a collection of its subordinates who have the

capability to perform a specified operation. Each subordinate, acting as a potential contractor, will determine when it will be able to perform the requested operation and what other activities will be affected (this may require the subordinate to negotiate with its subordinates); it will propose a bid to its supervisor. The supervisor will choose to accept one of the proposed bids based upon its own internal knowledge and inform that subordinate that its bid is being accepted (this is similar to the contracting party signing the contract). The chosen subordinate will then finalize the bid as a contract if it is still possible to perform the operation within the proposed time frame (this is similar to the contractor signing the contract). At this point a contract exists between the two control entities to perform a specified operation within a specified time frame. For the remainder of this document, both bids and the resulting contracts are referred to as covenants.

During the negotiation process, a production plan is developed to refine a manufacturing process. Thus each covenant is associated with a production plan and vice versa. When the plan contains steps which involve the actions of subordinates, a subcovenant is associated with each such step. A covenant represents the scheduling information (for the associated production plan) which is shared between a planner and its supervisor. A subcovenant represents the scheduling information related to a specific step within a production plan which is shared between a planner and one of its subordinates. A subcovenant is the supervisor's view of a subordinate's covenant.

2.4.2 Planner Recovery From Anomalous Situations

The MSI architecture specifies a functional interface between a planner and a job controller to address anomalous situations which are detected by a job controller. If a job controller determines that a task is running late, cannot be started, or cannot be continued, it will request the planner to replan the affected production plan. The planner will attempt to replan the production plan. Properly, this may involve rescheduling steps, adding, removing, or modifying steps, or even taking different production alternatives. With current technology, it is likely that a planner will only be able to resolve problems that can be solved by rescheduling steps. Even in most rescheduling cases, a planner will have to renegotiate contracts and subcontracts with its subordinates and possibly its supervisor in order to resolve problems; a planner may have to resolve a problem by instructing the job controller to abnormally complete a task.

Planner recovery is discussed in more detail in Section 7 on page 63.

2.5 The Job Control Function

The primary function of a job controller is to perform manufacturing operations. In the MSI architecture, a job controller may only receive task requests from its supervisor; it will accept or reject these requests based upon its administrative state, the validity of the message parameters, and possibly other internal information.

In the MSI architecture, the job controllers use a 'token of control' scheme to monitor and execute tasks. A control entity will execute those steps in a production plan that are assigned to it. When it encounters a step which refers to a subordinate control entity, it will pass the token of control for that task to the subordinate control entity by instructing it to perform a task that corresponds to that step[2]. The supervisor job controller will then monitor the execution of its task by requesting and receiving status information from its subordinates. The token of control will return to the

2. There will be multiple tokens of control for a task if there are parallel or concurrent paths in that production plan (one for each path).

supervisor when the subordinate has normally or abnormally completed the execution of its task. It is also possible for the supervisor to influence the outcome of a subordinate's task by issuing messages specific to the task. Section 6 on page 42 provides a more detailed explanation of the job control task execution scheme.

During execution, every contract made by the planner becomes a task to be executed by the job controller by the direction of its supervisor. Thus, a task is associated with the corresponding production plan. When the job controller is ready to initiate a step XXX by a subcovenant, a corresponding subtask is created and associated with that step. A task represents the execution-specific information related to a production plan which is shared between a job controller and its supervisor. A subtask represents the execution-specific information related to a specific step within a production plan which is shared between a job controller and one of its subordinates. A subtask is the supervisor's view of a subordinate's task. In addition, for each task, there is a corresponding covenant in the planning domain and for each subtask associated with a step in the production plan, there is a corresponding subcovenant in the planning domain.

2.6 Detection and Recovery from Anomalous Situations

Anomalies can be grouped into three different classes based upon their cause: resource anomalies, task anomalies and tooling anomalies. A resource anomaly occurs when a piece of equipment, whose control entity is part of the control hierarchy, becomes impaired or exhibits unexpected behavior. A task anomaly is an anomaly which affects a specific task only (e.g., task lateness, workpiece damage, unanticipated task abortion, etc.); the resource on which the task is currently being performed is unaffected. In general, a resource anomaly will result in a task anomaly, but a task anomaly may not cause a resource anomaly. A tooling anomaly occurs when a tool is damaged. Tools, as defined in the MSI architecture[3], differ from other resources in that they are not permanently associated with any member of the control hierarchy but may be moved from workcell to workcell as needed. The interfaces defined by the MSI architecture currently support the detection and reporting of all three types of anomalies, but only addresses recovery from resource and task anomalies.

The MSI architecture provides two mechanisms for addressing resource and task anomalies: human monitoring and intervention, and replanning. Resource anomalies are, in general, resolved by the human monitoring and intervention mechanism because the resource's availability may be affected and the resource is likely to require maintenance. Task anomalies may be resolved by either mechanism. Each mechanism will be discussed in turn.

2.6.1 Human Monitoring and Intervention

The MSI architecture provides two interfaces to support external monitoring and intervention in the otherwise automatic operation of a control entity: the guardian and the watchdog.

2.6.1.1 The Guardian Interface

The guardian interface is provided so that a control entity may receive assistance from an operator to resolve planning or job control errors. The entity with which a control entity establishes a connection via the guardian interface is called a guardian. The guardian is a software complex with decision-making intelligence (provided by a human operator or an automated intelligent

3. See [Barkmeyer, 1993] and [Ray, 1992b].

entity). There are two types of guardian interfaces: the guardian planning interface through which a control entity reports planning errors, and the guardian job control interface through which a control entity reports job control errors. In addition, each of the two guardian interfaces may be either active or passive. A passive guardian interface provides only monitoring capabilities, whereas an active guardian interface provides monitoring and intervention capabilities. A control entity may have any number of established passive guardian planning and guardian job control connections; it may have no more than one established active guardian planning connection and one established active guardian job control connection. Whether a guardian connection is passive or active is determined when the connection is established. Sections 8 and 9 discuss, in detail, the messages for the guardian planning interface and the guardian job control interface, respectively.

2.6.1.2 The Watchdog Interface

To allow a control entity to run without active guardian connections, a primitive guardian interface, referred to as the 'watchdog', is presumed always to exist for a running control entity.

The watchdog interface is a mechanism by which an operator is informed that a control entity needs assistance to resolve an anomalous situation encountered by the planner or the job controller (that is, either the planner or job controller needs assistance on its active guardian connection). The watchdog can be described conceptually as a pair of red lights. One red light denotes the control entity needing assistance to resolve a planning error (denoted 'the planning red light'); the other, that the control entity needs assistance to resolve a job control error (denoted 'the job control red light').

Specific messages in the guardian planning interface cause the control entity to turn on the planning red light. These messages are Subordinate Problem, Subcovenant Error Occurred, and Administrative Status messages which contain abnormal information. If a control entity wishes to issue one of these messages, it will turn on the planning red light whether or not the control entity has an active guardian planning connection. The planning red light remains turned on until all outstanding planning problems requiring guardian assistance have been resolved.

Specific messages in the guardian job control interface cause the control entity to turn on the job control red light. These messages are Subordinate Problem, Task Error Occurred, Subtask Error Occurred, and Administrative Status messages which contain abnormal information. If a control entity wishes to issue one of these messages, it will turn on the job control red light whether or not the control entity has an active guardian job control connection. The job control red light remains turned on until all outstanding job control problems requiring guardian assistance have been resolved.

2.6.2 Replanning

Replanning, or the dynamic modification of production plans, is the automated mechanism by which a control entity (or a segment of the shop) recovers from anomalous situations. Since all anomalous situations (other than planner failures) are detected by the job controller, replanning is initiated by a message from the job controller to the planner within the control entity. A brief discussion of replanning was presented in Section 2.4.2 on page 16. A more detailed discussion of replanning is presented in Section 7 on page 63.

2.7 Communications Paradigm

Each of the interfaces defined in this specification assumes a guaranteed point-to-point messaging communications paradigm. This section describes the paradigm in more detail and discusses the consequences of using this paradigm.

Because a connection exists between exactly two entities, if an entity wishes to share the same information with more than one entity, it must create multiple connections, one to each interested entity, and it must explicitly transmit messages across each connection.

Because the communications paradigm supports guaranteed message delivery, an entity must support message queuing: it is possible for an entity to receive messages much faster than it can process them. An entity must support some level of message queuing and some mechanism to handle a full message queue.

Because the communications paradigm supports guaranteed message delivery, an entity may take advantage of incremental status reporting. An entity may report only on those items that have changed since the last time status was reported because the receiving entity is guaranteed to receive every message and in the order they were issued.

2.7.1 The Nature of a Control Entity Interface

A connection is an instance of a specific interface between exactly two entities. There are several valid operations that an entity may perform on an interface: connect, disconnect and transmit:

- Connecting to an interface results in the creation of a connection between exactly two entities.
- Disconnecting from an interface causes an existing connection to cease to exist. (Connection recovery is the process of building a new connection and getting its information state to match the state of a previous connection.)
- Transmitting across an interface occurs when one entity participating in a connection sends a message (a collection of information) to the other entity participating in a connection.

2.7.2 Types of Messages

There are three types of messages used in defining the interfaces in the MSI architecture: requests, responses and unconfirmed messages.

A request is a message by which one entity asks another entity to perform some operation or provide some information.

A response is a message sent in reply to a particular request. A positive response (denoted response+ in the text) either indicates that the receiving entity agrees to perform the operation (or has completed it), or provides the requested information. A negative response (denoted response– in the text) indicates that the receiving entity is rejecting the request and (usually) explains why.

An unconfirmed message is a message by which one entity provides information to another entity without a request being made. An unconfirmed message may be thought of as a response to an assumed request for information or as a notification that something may need to be done, rather than a request to do something in particular.

Formally, to every request message there is exactly one response message. Some requests, however, may cause the receiving entity to initiate a fairly long-term operation, so that the formal

response only indicates acceptance of the operation request. Other messages, both unconfirmed and request/response pairs, concerning the same operation may be exchanged during the course of the operation.

An entity which issues a request or unconfirmed message is said to be the "issuing entity". An entity which receives an unconfirmed message or request and issues a response is said to be the "receiving entity".

2.7.3 Message Format

Each type of request or unconfirmed message is presented in a separate subsection within this document. Responses are described under the corresponding request.

Each subsection presenting the semantics of a request follows the following format:

- the name of the request (or the unconfirmed message),
- a definition of the message from the issuing entity's perspective,
- a list of the parameters to the request
- a discussion of the parameters and variations in the request,
- a list of the possible responses and their parameters,
- a discussion of the parameters and variations in the responses, and
- a definition of the request from the receiving entity's perspective — what steps are to be taken under what circumstances (usually presented in a table).

2.7.4 Types of Parameters

Each message may or may not have parameters associated with it. There are four types of parameters: required, optional, conditional, and choice.

A required parameter is a parameter that must be specified. A parameter is required unless otherwise labelled.

An optional parameter is a parameter that may or may not be specified. In general, optional parameters are parameters such as text strings describing an error code. An optional parameter is denoted by the word 'optional' following the parameter name. For example:

error-text (OPTIONAL)

A conditional parameter is a parameter that is required in some situations and absent in others. A conditional parameter is denoted by the word 'conditional' following the parameter name. For example:

Task Status Object (CONDITIONAL)

A choice parameter is a parameter which has a single function but may take values of different data types. For example, some requests for information have a choice parameter which designates the objects to be reported on. The parameter value may be an object type, indicating that all objects of that type should be reported, or it may be a list of object identifiers, indicating that exactly those objects are to be reported on. A choice parameter is denoted by the word 'CHOICE OF' and then a collection of alternative parameters or values. For example:

```
covenants:    CHOICE OF {
                  list of supervisor-covenant-identifier
              OR All-Bids
              OR All-Contracts
              OR All-Covenants}
```

3 Production Plans

This section discusses production plans and their steps in the context of the interaction between planners and job controllers and the recovery from anomalous situations.

3.1 Production Plans and their States

Production plans are accessed and modified by both a planner and job controller — sometimes simultaneously. As a result, they are subject to change and refinement (evolution) both before and during execution. As they change, they make transitions through a set of evolutionary states. These states are In-Scheduling, Bidden, Scheduled, Executing, Completed, Deferred, Terminated and Aborted. Table 1. explains each of the plan-states and the constraints placed on both a planner and job controller with respect to accessing and modifying a production plan in that state.

Plan State	Valid Planner Actions	Valid Job Controller Actions
In-Scheduling, Bidden	A planner is in the process of performing resource allocation and scheduling for this plan via negotiations with the supervisor and subordinates.	A job controller should never access a production plan which is in one of these states.
Scheduled[a]	A planner may modify a production plan's plan-state from the Scheduled plan-state to either the In-Scheduling or Bidden plan-state. It may not make any other modifications to a production plan in this state.	A job controller may modify a production plan's plan-state from the Scheduled plan-state to the Executing plan-state. It may not make any other modifications to a production plan in this state.
Executing	If a production plan is in the Executing plan-state, the rules governing the interaction between a planner and job controller are defined on a step by step basis. See the discussion of step-states below.	If a production plan is in the Executing plan-state, the rules governing the interaction between a planner and job controller are defined on a step by step basis. See the discussion of step-states below.
Completed, Aborted, Deferred, Terminated	A planner may make any valid modifications it wishes to a production plan in these states. In general, it will do nothing more than dispose of the production plan.	If a production plan is in any of these plan-states, a job controller has normally or abnormally completed executing the production plan and no longer needs to modify the production plan.

Table 1. Production Plan States

a. The transitions from Scheduled to In-Scheduling, Bidden or Executing must have a semaphore mechanism to prevent both a planner and a job controller from checking the current plan-state of a production plan and then each of them setting the plan-state to a different value.

Valid plan-state transitions are depicted in Figure 5. on page 22. Each transition specifies which entity can make that transition (the planner [p] or the job controller [c]).

3.2 Steps and Their States

Because both a planner and job controller can access and modify a production plan simultaneously, there not only need to be rules for accessing the production plan as an entity, but also rules governing the access and modification of the individual steps within a production plan. As a result, a collection of step-states have been defined, along with a collection of rules defining the constraints placed on both a planner and job controller with respect to accessing and modifying a step in that state. Valid step-states are Unscheduled, In-Scheduling, Scheduled, Not-

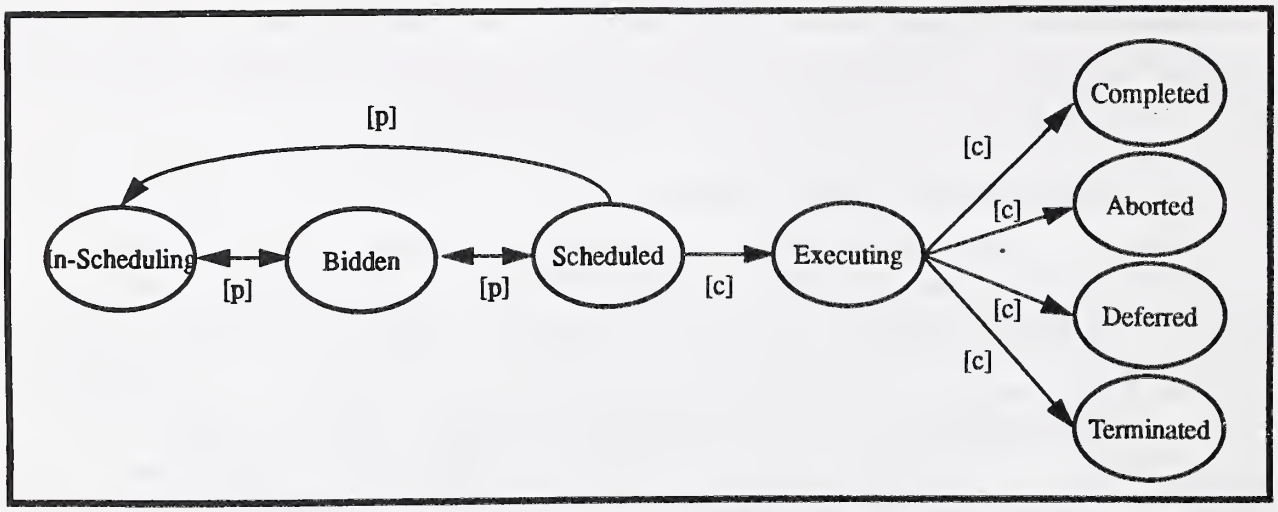


Figure 5. Plan-state Transition Diagram

Yet-Executing, Executing and Completed. Table 2. explains each of the step-states and the rules associated with each state.

Step-state	Valid Planner Actions	Valid Job Controller Actions
Unscheduled, In-Scheduling	A planner may make any valid modifications to a step which is in one of these states as long as the modifications do not violate planner negotiations.	A job controller should never access a step which is in one of these states.
Scheduled[a]	A planner may modify a step's state from the Scheduled step-state to either the Unscheduled or In-Scheduling step-state.	A job controller may modify a step's state from the Scheduled step-state to the Not-Yet-Executing step-state.
Not-Yet-Executing[b]	A planner may modify a step's state from the Not-Yet-Executing step-state to the Unscheduled, In-Scheduling or Scheduled step-states. It may also update the scheduled-start-time or scheduled-completion-time of a step in this state. In all cases, it must issue a Recheck Step message to the job controller. See Section 7.2.1 on page 72 for the discussion of Recheck Step.	A job controller may modify a step's state from the Not-Yet-Executing step-state to the Executing or Scheduled step-state.
Executing	A planner may update only the scheduled-completion-time of a step in this state.	A job controller may make any valid modifications to a step in a production plan in this state. It may modify a step's state to the Scheduled step-state.
Completed	A planner may dispose of a step in a production plan which is in this state.	A job controller may make any valid modifications to a step in this state. It may modify a step's state to the Scheduled step-state.

Table 2. Production Plan Step-states

a. The transitions from Scheduled to Unscheduled, In-Scheduling, or Executing must have a semaphore mechanism to prevent both a planner and a job controller from checking the current step-state of a step in a production plan and then each of them setting the step-state to a different value.
b. The transitions from Not-Yet-Executing to Unscheduled, In-Scheduling, Scheduled or Executing must have a semaphore mechanism to prevent both a planner and a job controller from checking the current step-state of a step within a production plan and then each of them setting the step-state to a different value.

Valid step-state transitions are depicted in Figure 6. Each transition specifies which entity can make that transition (the planner [p], the job controller [c] or either [p/c]).

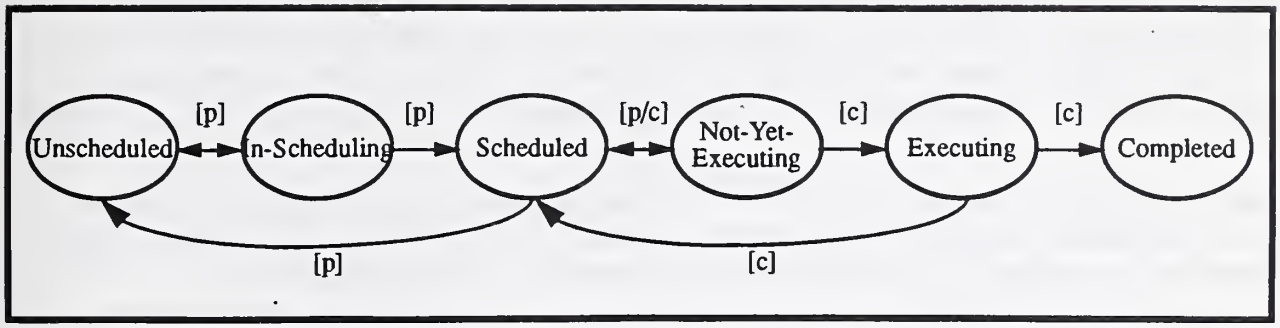


Figure 6. Step-state Transition Diagram

3.3 Checkpoints

In order to facilitate recovery from anomalous situations, the MSI architecture supports the notion of specific steps within a production plan being checkpoints. A checkpoint is a step within a production plan where, upon completion of that step, the manufacturing process may be temporarily or indefinitely suspended in a safe manner without damage to either the equipment of the workpieces. The shop level planner and job controller and all workcell planners and job controllers are required to understand checkpoints. In most cases, the start and end of a production plan, at least, will be checkpoints. A less trivial example of a checkpoint would be a machining step which drills a hole. After the hole is completed and the cutter is withdrawn, in some cases it may be safe to stop the plan at this step and resume at a later time. The MSI architecture does not require that equipment job controllers understand the notion of checkpoint.

4 Generic Data Objects

There are several simple and complex data objects used in the five control entity interfaces. A simple data object is an object which contains a single piece of information; a complex data object is an object which contains several pieces of information, each of which may be simple or complex. A definition of each simple data object can be found in the glossary (See Section 12 on page 110).

Several of the complex data objects are generic and used in several of the control entity interfaces. Each of the generic complex data objects are presented in this section. Those complex data objects which are unique to a single interface are presented in the appropriate section.

4.1 Plan Identifier Object

The Plan Identifier Object is a data object which uniquely identifies a production-managed plan or a production plan. It contains the following data elements:

plan-location (CONDITIONAL)
plan-identifier
plan-version

The parameter plan-location has a value if the plan is located in some database other than a planner's default plan database.

4.2 Plan Parameter Object

The Plan Parameter Object is a data object which specifies a name, value pair denoting either an input or output parameter. It contains the following data elements:

parameter-name
parameter-value (CONDITIONAL)

All input parameters will provide a parameter-value; output parameters will only specify a parameter-value if it is known at the time the Plan Parameter Object is created.

4.3 Error Object

The Error Object is a data object which contains information describing an error which has occurred. It contains the following data elements:

error-code
error-text (OPTIONAL)

The Error Object is used in all 5 of the control entity interfaces presented in Sections 5 – 9. Each interface, however, has its own set of valid error-codes.

Valid error-codes within the planning interface include:

- deactivate failed — connected to job controller
- deactivate failed — outstanding bids
- invalid administrative state
- invalid covenant identifier
- invalid covenant state
- invalid parameter specified

- invalid plan identifier
- invalid planning strategy
- planning horizon exceeded
- subordinate not responding

Valid error-codes within the job control interface include:

- cannot pause task within requested pausing duration
- deactivate failed — outstanding tasks
- invalid administrative state
- invalid parameter specified
- invalid plan identifier
- invalid task identifier
- invalid task state
- subordinate not responding
- terminate task failed — task not paused

Valid error-codes within the planning/job control interface include:

- invalid plan state
- invalid production plan identifier
- invalid step identifier

Valid error-codes within the guardian planning interface include all valid planning error-codes plus the following:

- cannot connect to subordinate
- invalid subordinate
- subordinate reporting anomalous covenant reports

Valid error-codes within the guardian job control interface include all valid job control error-codes plus the following:

- cannot connect to planner
- cannot connect to subordinate
- invalid subordinate
- invalid subtask identifier
- subordinate reporting anomalous administrative status reports
- subordinate reporting anomalous task reports

4.4 Acceptance Object

The Acceptance Object is a data object which either confirms an operation or specifies an error condition. It contains the following data elements:

acceptance-code
acceptance-text (OPTIONAL)

For each interface, the set of valid acceptance-codes is the set of valid error-codes plus the acceptance-code which specifies that no error has occurred and the operation has been accepted.

5 Planning Interface

This section presents the planning interface, including:

- a general discussion of the planning paradigm and how the messages provided by the interface support that paradigm,
- state models for both a planner's administrative state and its covenants,
- the data objects used by the planning interface, and
- the messages which are provided by the planning interface — administrative planning requests, covenant requests issued by a supervising planner and covenant requests issued by a subordinate planner.

5.1 The Planning Model

This section discusses how a planner performs its planning and replanning functions by using internal planning strategies and the messages provided by the planning interface.

In most scenarios where a planner is requested to plan or replan an operation, the sequence of steps that the planner follows are similar. As a result, a general discussion of what planning strategies are and how they affect the planning function, and scenarios describing how a planner plans and replans will aid the reader in more fully understanding the planning interface.

5.1.1 Planning Strategies

Each planner supports some number of planning strategies. A planning strategy determines the approach that a planner will take when trying to schedule an operation; the planning strategy specifies a set of constraints which the planner should meet and a set of goals which the planner should achieve. A time window, specified by an earliest-start-time and a latest-completion-time, may also be specified to further constrain a planner's options. If either limit of the time window is not specified, the assumed earliest-start-time is the current time and the assumed latest-completion-time is the end of the planner's planning horizon.

Planning strategies include:

- Priority Scheduling — within a given time window, schedule an operation anywhere within the window using its priority to cancel previously scheduled operations.
- First-Come-First-Serve Scheduling — within a given time window, schedule an operation anywhere within the window without disrupting or cancelling previously scheduled operations.
- As-Soon-As-Possible Priority Scheduling — within a given time window, schedule an operation as soon as possible (as close to the earliest-start-time of the time window as possible) using its priority to cancel previously scheduled operations.
- As-Soon-As-Possible First-Come-First-Serve Scheduling — within a given time window, schedule an operation as soon as possible (as close to the earliest-start-time of the time window as possible) without disrupting or cancelling previously scheduled operations.
- As-Late-As-Possible Priority Scheduling — within a given time window, schedule an operation as late as possible (as close to the latest-completion-time of the time window as possible) using its priority to cancel previously scheduled operations.

- As-Late-As-Possible First-Come-First-Serve Scheduling — within a given time window, schedule an operation as late as possible (as close to the latest-completion-time of the time window as possible) without disrupting or cancelling previously scheduled operations.

When a planner supports more than one planning strategy, the supervisor will specify which planning strategy to use when it requests a planner to schedule an operation.

5.1.2 Scheduling an Operation

When a planner schedules an operation described by a production-managed plan, the planner follows a certain sequence of steps as it creates the production plan. For a workcell or equipment level planner, this sequence of steps is initiated by the receipt of a Request for Bid request from the supervisor. For the shop planner, this sequence of steps is initiated by new orders (with associated production-managed plans) entering the shop.

When requested to schedule an operation, a planner should perform the following steps:

1. For each step in the production-managed plan (in precedence order), the planner should do the following:
   (a) If the step does not require the use of any subordinate, schedule the step according to internal procedures and the specified planning strategy. Proceed to the next step in the plan.
   (b) Otherwise, issue Request for Bid requests to some collection of subordinates who can perform the step.
   (c) After receiving some number of Covenant Status messages from subordinates advertising their bids, select one bid which should be accepted.
   (d) Issue an Accept Bid request to the subordinate whose bid was selected.
   (e) If an Accept Bid response– is received from that subordinate, then choose another bid to accept and repeat steps (d) and (e). If there are no more bids to accept, then change the parameters to the Request for Bid and repeat starting at step (c).
   (f) If an Accept Bid response+ is received from a subordinate, then issue Remove Covenants requests to each of the other subordinates who have provided a bid but were not issued an Accept Bid request. Proceed to the next step in the plan.
2. After all steps in the production-managed plan have been scheduled and a production plan has been created, if the planner has a supervisor (i.e., the planner is not the shop planner), issue a Covenant Status message to the supervisor advertising the bid.

5.1.3 Rescheduling In-Execution Operations

When an anomalous situation occurs which results in a production plan's specified schedule becoming infeasible, a job controller will request the planner to resolve the situation. In these cases, the planner will attempt to reschedule the production plan.

The steps that a planner follows in attempting to reschedule a production plan are nearly identical to those that the planner performs when scheduling an operation. When a planner is scheduling an operation, it schedules every step in the production plan, whereas when a planner is rescheduling a production plan, it only reschedules those steps which are directly causing the schedule's infeasibility and those steps which are directly or indirectly affected by resolving this infeasibility. The planner will reschedule the production plan starting with the steps causing the infeasibility; the planner will then examine each subsequent step to determine if its scheduled execution time overlaps with the rescheduled execution of the previous steps. If there is an overlap, the planner

will reschedule those steps. This will continue until the end of the plan or until a step is reached which is not affected by the rescheduling of the previous steps.

When requested to reschedule a production plan, a planner should perform the following steps:

1. Starting with the step(s) which causes the schedule's infeasibility, and continuing until the end of the plan or until a step is reached whose scheduled execution time does not overlap with the scheduled execution time of the rescheduled previous step(s), the planner should do the following:
   (a) If the step does not require the use of any subordinate, schedule the step according to internal procedures and the specified planning strategy. Proceed to the next step in the plan.
   (b) Otherwise, issue a Remove Covenants request to the subordinate currently specified to perform the step.
   (c) Issue Request for Bid requests to some collection of subordinates who can perform the step.
   (d) After receiving some number of Covenant Status messages from subordinates advertising their bids, select one bid which should be accepted.
   (e) Issue an Accept Bid request to the subordinate whose bid was selected.
   (f) If an Accept Bid response– is received from that subordinate, then choose another bid to accept and repeat steps (d) and (e). If there are no more bids to accept, then change the parameters to the Request for Bid and repeat starting at step (c).
   (g) If an Accept Bid response+ is received from a subordinate, then issue Remove Covenants requests to each of the other subordinates who have provided a bid but were not issued an Accept Bid request. Proceed to the next step in the plan.
2. The planner should continue with the semantics of the scenario which caused it to need to reschedule.

5.2 Planner Administrative States

Each planner has a notion of administrative state. Table 3. enumerates and defines the valid external administrative states[4] for a planner.

State	Definition
Available	The planner has either 1. started cold and is ready to establish a connection to the supervisor, or 2. received and processed a Deactivate message, released its connection to the supervisor, deactivated its subordinates and is ready to be halted.
Active	The planner is 1. connected to and activated by the supervisor, 2. connected to or in the process of activating its subordinates, 3. ready to accept and process or already processing covenant requests from the supervisor, and 4. ready to accept and process or already processing replanning requests from the job controller.

Table 3. Planner Administrative-state Definitions

Valid transitions among the planner administrative states and the planning messages that cause those transitions are depicted in Figure 7.

4. It is possible for a planner to have any number of internal intermediate administrative states, for example, to specify whether it is connected to the supervisor, connected to the job controller(s) or connected to its subordinates.

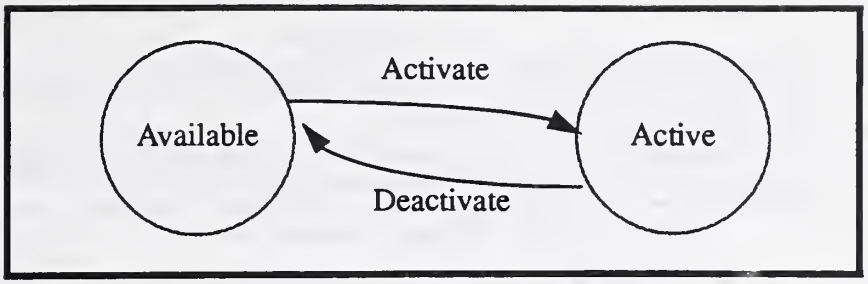


Figure 7. Planner Administrative-state Transition Diagram

5.3 Covenant States

Each covenant that a planner is currently negotiating or has already negotiated but not disposed of (i.e., the job controller has not executed a task to fulfill that covenant) has an associated covenant-state. A covenant-state denotes the current state of a covenant; valid covenant-states are:

- Accepting
- Bidding
- Bidden
- Breaking Bid
- Breaking Contract
- Contracted
- Executing
- Executed
- No Covenant
- Pre-Bidding
- Removing Covenant

All covenant-states are transitional states except No Covenant and Executed. The covenant-states address not only the negotiation stages of a covenant, but also the execution of a task associated with that covenant. Only after the task has completed execution (normally or abnormally) does the covenant cease to exist. Table 4. defines each of the covenant-states.

Covenant-state	Definition
Pre-Bidding	The planner has received a Request for Bid request from the supervisor but has not yet responded. It is determining whether or not it will formulate a bid.
Bidding	The planner has received and responded positively to a Request for Bid from the supervisor. A covenant has been created but the planner has not completed formulating a bid.
Bidden	The planner has formulated a bid and submitted it to the supervisor. The planner is waiting for the supervisor to confirm the bid or to remove the bid (it may be removed for several reasons).
Accepting	The planner has received an Accept Bid request from the supervisor for a bid and it is in the process of converting the bid into a contract.
Contracted	The planner and the supervisor have agreed that the covenant will be honored by both parties to perform a task within a specified time frame.
Executing	A task has been created by a job controller to fulfill the covenant.
Executed	The covenant has been fulfilled (or abandoned).
Breaking Bid	The planner has requested permission from the supervisor to cancel this covenant in order to formulate a bid for some other request. A response has not yet been received from the supervisor.
Breaking Contract	The planner has requested permission from the supervisor to cancel this covenant in order to formulate a bid for some other request. A response has not yet been received from the supervisor.

Table 4. Covenant-state Definitions

Covenant-state	Definition
Removing Contract	The planner has received a Remove Covenant request from the supervisor to remove the covenant but has not yet disposed of it.
No Covenant	The planner does not intend to formulate a bid as requested by the supervisor or a planner has just disposed of the covenant as requested by the supervisor.

Table 4. Covenant-state Definitions

Valid transitions among the covenant-states and the messages (planner messages or planning-to-job-control messages) which cause those transitions are depicted in Figure 8..on page 31.

5.4 Complex Data Objects Used in the Planning Interface

In addition to the data objects described in Section 4 on page 24, there is only one other complex data object used in the planning interface: the Covenant Status Object.

The Covenant Status Object is a data object which contains information describing a single covenant. It is used to convey covenant status information between supervisory and subordinate planners[5].

The Covenant Status Object contains the following data elements:

planning-strategy
priority
earliest-start-time (CONDITIONAL)
latest-completion-time (CONDITIONAL)
covenant-state
production-plan: Plan Identifier Object
scheduled-start-time
scheduled-completion-time
conflicting-covenants: list of supervisor-covenant-identifier (CONDITIONAL)

The parameters earliest-start-time and latest-completion-time have values only if they are specified by the supervisor in the Request for Bid request (see Section 5.6.4 on page 37). The parameter conflicting-covenants specifies the list of covenants that would have to be cancelled in order for the current covenant to be contracted. If there are no such covenants, then the list is empty. When the covenant is contracted, such covenants are broken and the list is always empty.

5.5 Planner Administrative Requests

The planner issues administrative requests to one of its subordinates to alter its administrative status or request information about its administrative status. The planner administrative requests are:

- Activate
- Deactivate
- Identify

5. In addition, it is used to report information to a guardian through the guardian planning interface. See Section 8.1.3 on page 75 for more information about its use in the guardian planning interface.

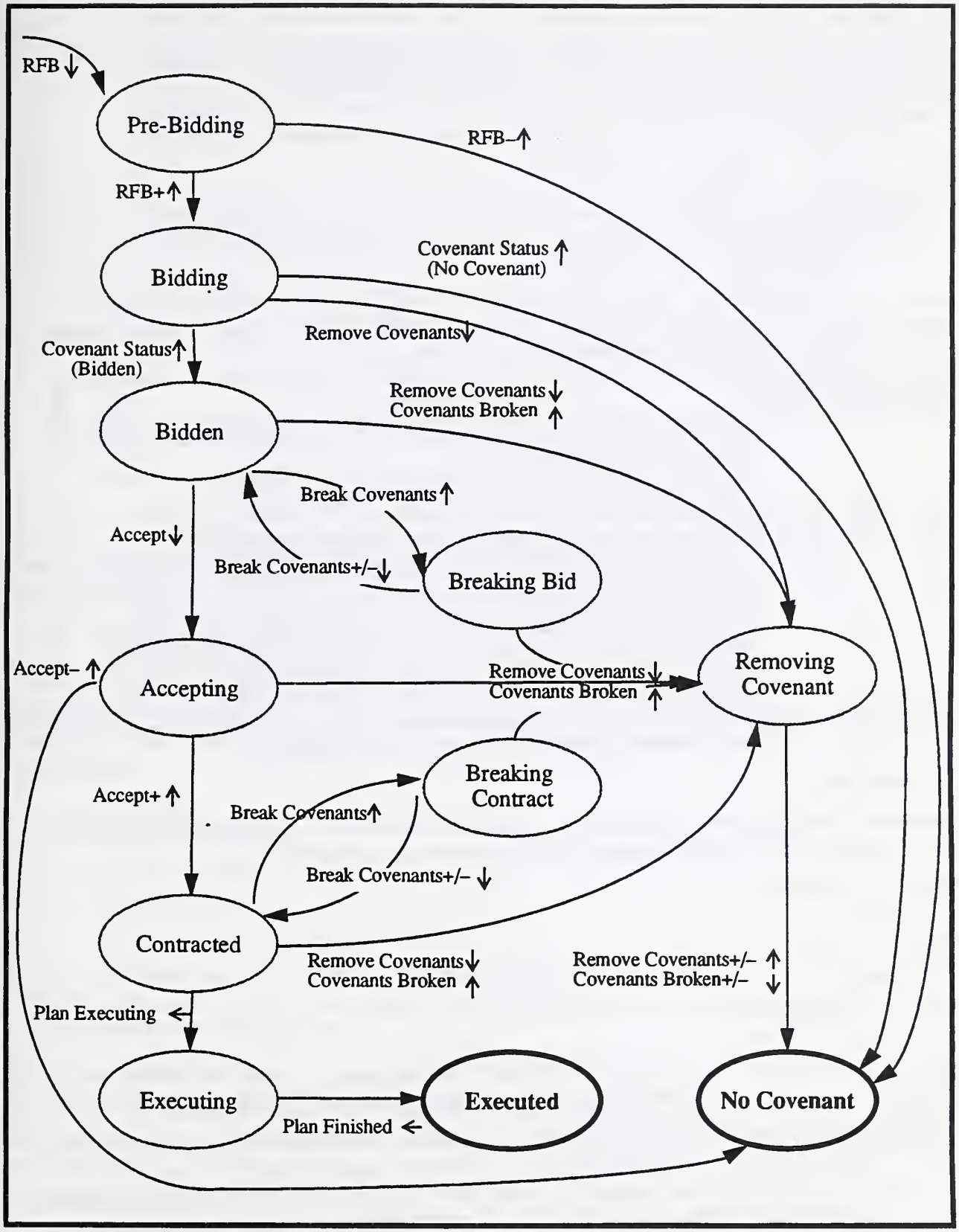


Figure 8. Covenant-state Transition Diagram

5.5.1 Activate

An Activate message is used by a planner to connect to and initialize a subordinate. The parameters for both the Activate request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner wishing to activate one of its subordinates should issue it an Activate request. The Activate response+ results in the connection to be established. Table 5. details what a planner should do upon receiving an Activate request from the supervisor.

Administrative State	Action
Available	1. Upon receiving an Activate request, perform any internal initialization and switch to the Active administrative state. 2. If the initialization fails or the planner cannot switch to the Active administrative state, then issue an Activate response–. No further action is necessary. 3. Otherwise, issue an Activate response+. 4. If necessary, query the database to determine who its subordinates are. 5. Attempt to activate each subordinate by issuing an Activate request to each one. 6. If the planner receives an Activate response– from any of its subordinates, contact the active guardian and proceed according to its instructions.
Active	Receiving an Activate request while in the Active administrative state causes a connection to be established between the receiving planner and the supervisor. If the planner already has a connection to the supervisor, this is a protocol violation; issue an Activate response–. Otherwise, accept the connection by issuing an Activate response+.

Table 5. Semantics of the Planner Activate Message

5.5.2 Deactivate

A Deactivate message is used by a planner to disconnect from and shut down a subordinate. The parameters for both the Deactivate request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner wishing to deactivate one of its subordinates should issue it a Deactivate request. Upon receiving a Deactivate request from the supervisor, a planner should perform the following steps:

1. If the planner's administrative state is Available, a protocol violation has occurred; ignore the request. No further action is necessary.
2. If the planner has any outstanding bids, issue a Deactivate response– and continue negotiating covenants. No further action is necessary.
3. If the planner has no subordinates, respond Deactivate response+ and switch to the Available administrative state. The planner is now ready to be halted.

4. If the planner has subordinates, issue a Deactivate response+. This will result in the connection with the supervisor being released. Then the planner should issue a Deactivate request to each of its subordinates. If any subordinate responds with Deactivate response–, contact the active guardian and proceed according to its instructions. If all subordinates respond with Deactivate response+, switch to the Available administrative state. The planner is now ready to be halted.

5.5.3 Identify

An Identify message is issued by a planner to determine the characteristics of a subordinate. The parameters for both the Identify request and its response are as follows:

Parameters on Request:

<none>

Parameters on Response:

make
model
version
level
list of planning-strategy

A planner wishing to know the characteristics of one of its subordinates should issue that subordinate an Identify request. Upon receiving an Identify request from the supervisor, a planner should issue an Identify response, specifying its make, model and version, its level and which planning strategies it supports.

5.6 Covenant Requests Issued By A Supervising Planner

Covenant Requests are requests which a planner may issue to either the supervisor or one of its subordinates to perform some action to a set of covenants or to obtain information about a set of covenants. This section presents the covenant requests which a planner may issue to one of its subordinates:

- Accept Bid
- Remove Covenants
- Report Covenants
- Request for Bid
- Update Covenant Constraints

These requests may only be issued to a subordinate planner which is in the Active administrative state. If a subordinate planner receives one of these requests while in the Available administrative state, the request should be ignored.

5.6.1 Accept Bid

An Accept Bid message is issued by a planner to a subordinate; it requests a bid to be accepted resulting in a contract being established between a planner and that subordinate. The parameters for both the Accept Bid request and its responses are as follows:

Parameters on Request:

subordinate-covenant-identifier

Parameters on Response+:

supervisor-covenant-identifier

Parameters on Response–:

supervisor-covenant-identifier
error: Error Object
Covenant Status Object (CONDITIONAL)

If a planner issues an Accept Bid response–, the conditional parameter Covenant Status Object is specified if it contains information related to rejecting the request.

If a planner wishes to generate a contract for a step in a production plan (for which the planner has a collection of bids and has determined which bid it would like to accept), it should issue an Accept Bid request to the subordinate who advertised the selected bid.

1. If a planner receives an Accept Bid request for an invalid covenant, it should issue an Accept Bid response–. No further action is necessary.
2. If the covenant (or production plan supporting the covenant) is not intact (i.e. steps are currently being rescheduled and the planner expects that the rescheduling of the steps will impact the start or completion time of its advertised bid), do the following:
   (a) Issue an Accept Bid response– to the supervisor.
   (b) If the planner is at the equipment level, dispose of the covenant and associated production plan. No further action is necessary.
   (c) If the planner has subordinates, issue Remove Covenants requests to all subordinates with subcovenants in support of the covenant, and then dispose of the covenant and associated production plan. No further action is necessary.
3. If the covenant (and production plan supporting the covenant) is still intact, issue a Covenants Broken request for all covenants that need to be broken in order to accept the specified covenant (this list was specified in the original bid). After receiving a response to the Covenants Broken request from the supervisor, issue an Accept Bid response+ to the supervisor. No further action is necessary.

5.6.2 Remove Covenants

A Remove Covenants message is issued by a planner to a subordinate; it requests a set of covenants which a planner has negotiated with that subordinate to be cancelled. The parameters for both the Remove Covenants request and its response are as follows:

Parameters on Request:

covenants: CHOICE OF {
list of subordinate-covenant-identifier
OR All-Bids
OR All-Contracts
OR All-Covenants}

Parameters on Response:

list of [
subordinate-covenant-identifier
supervisor-covenant-identifier
acceptance: Acceptance Object
] (CONDITIONAL)

If a Remove Covenants request specifies All-Bids, All-Contracts or All-Covenants, the response will have no parameters.

If a planner wishes to cancel one or more covenants which currently exist between itself and one of its subordinates, it should issue a Remove Covenants request to that subordinate. The planner has a choice of specifying a list of covenants to remove or specifying to remove all bids, all contracts or all covenants. Upon receiving a Remove Covenants request from the supervisor, a planner should perform the following steps:

1. If the Remove Covenants request was issued with the parameter All-Bids, then for each covenant in the Pre-Bidding covenant-state, issue a Request for Bid response–. For each covenant in the Bidding, Bidden, Breaking Bid or Accepting covenant-states, do the following:
   Switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and the associated production plan. If the planner has subordinates, disposing of the selected covenants may require issuing Remove Covenants requests to subordinates for negotiated subcovenants. After the covenant is disposed of, switch the covenant to the No Covenant covenant-state. Issue a Remove Covenants response with no parameters. No further action is necessary.
2. If the Remove Covenants request was issued with the parameter All-Contracts, then for each covenant in the Breaking Contract or Contracted covenant-state, do the following:
   Switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and the associated production plan. If the planner has subordinates, disposing of the selected covenants may require issuing Remove Covenants requests to subordinates for negotiated subcovenants. After the covenant is disposed of, switch the covenant to the No Covenant covenant-state. Issue a Remove Covenants response with no parameters. No further action is necessary.
3. If the Remove Covenants request was issued with the parameter All-Covenants, then for each covenant in the Pre-Bidding covenant-state, issue a Request for Bid response–. For each covenant in the Bidding, Bidden, Breaking Bid, Accepting, Breaking Contract or Contracted covenant-state, do the following:
   Switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and the associated production plan. If the planner has subordinates, disposing of the selected covenants may require issuing Remove Covenants requests to subordinates for negotiated subcovenants. After the covenant is disposed of, switch the covenant to the No Covenant covenant-state. Issue a Remove Covenants response with no parameters. No further action is necessary.
4. If the Remove Covenants request was issued with a list of covenants, then for each covenant in the list that is in the Bidding, Bidden, Breaking Bid, Accepting, Breaking Contract or Contracted state, do the following:
   Switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and the associated production plan. If the planner has subordinates, disposing of the selected covenants may require issuing Remove Covenants requests to subordinates for negotiated subcovenants. After the covenant is disposed of, switch the covenant to the No Covenant covenant-state. Issue a Remove Covenants response with no parameters. It is an error to remove any other covenants. Issue a Remove Covenants response specifying the error codes for those that are in error and an acceptance code for those that were removed.

5.6.3 Report Covenants

A Report Covenants message is issued by a planner to a subordinate; it requests that subordinate to report the current status of a set of covenants. The parameters for both the Report Covenants request and its response are as follows:

Parameters on Request:

```
covenants:    CHOICE OF {
                  list of subordinate-covenant-identifier
              OR All-Bids
              OR All-Contracts
              OR All-Covenants}
```

Parameters on Response:

```
list of [
    subordinate-covenant-identifier
    supervisor-covenant-identifier
    CHOICE OF {
        error: Error Object
    OR status: Covenant Status Object}
] (CONDITIONAL)
```

If a Report Covenants request specifies All-Bids, All-Contracts or All-Covenants and the subordinate planner currently has no outstanding bids, contracts, or covenants (respectively), the response will have no parameters.

A planner wishing to receive updated status for a collection of its subcovenants should issue a Report Covenants request to each of its subordinates whose covenants correspond to those subcovenants. Upon receiving a Report Covenants request from the supervisor, a planner should perform the following steps:

1. If the Report Covenants request was issued with the parameter All-Bids, then for each covenant in the Bidding, Bidden, Breaking Bid or Accepting covenant-states, collect the supervisor-covenant-identifier, local-covenant-identifier and Covenant Status Object information and issue a Report Covenants response.
2. If the Report Covenants request was issued with the parameter All-Contracts, then for each covenant in the Contracted, Breaking Contract or Executing covenant-states, collect the supervisor-covenant-identifier, local-covenant-identifier and Covenant Status Object information and issue a Report Covenants response.
3. If the Report Covenants request was issued with the parameter All-Covenants, then for each covenant in the Bidding, Bidden, Breaking Bid, Accepting, Contracted, Breaking Contract, Removing Covenant or Executing covenant-states, collect the supervisor-covenant-identifier, local-covenant-identifier and Covenant Status Object information and issue a Report Covenants response.
4. If the Report Covenants request was issued with a list of covenants, then for each covenant that is in the list, but not in the Executed covenant-state, collect the supervisor-covenant-identifier, local-covenant-identifier and Covenant Status Object information. For all other covenants in the list, identify the appropriate error and issue a Report Covenants response which includes the error information for invalid covenants and the covenant status information for valid covenants.

5.6.4 Request for Bid

A Request for Bid message is issued by a planner to a subordinate; it requests that subordinate to estimate when it is able to perform a requested operation. The parameters for both the Request for Bid request and its responses are as follows:

Parameters on Request:

supervisor-covenant-identifier
global-step-reference
supervisor-manufacturing-unit-identifier (CONDITIONAL)
planning-strategy
priority
earliest-start-time (CONDITIONAL)
latest-completion-time (CONDITIONAL)
terminated-flag
input-plan: CHOICE OF {
plan: Plan Identifier Object
OR work-element-identifier}
list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)

Parameters on Response+:

supervisor-covenant-identifier
subordinate-covenant-identifier
covenant-state

Parameters on Response–:

supervisor-covenant-identifier
error: Error Object

If a planner wishes to schedule a specific operation, it should issue a Request for Bid request to some collection of its subordinates who are capable of performing the operation. The planner specifies a local identifier, a manufacturing unit which identifies the workpieces on which to perform the operation, if any, a planning strategy to use in planning the operation, a priority, optionally an earliest-start-time and latest-completion-time between which the operation should be scheduled, a plan or work element which details the steps to perform the operation and any input parameters which are required by the plan.

If the subordinate planner issues a Request for Bid response+, it will specify a local identifier for the covenant. In all future communications regarding this covenant, the supervisor will always provide the subordinate's local identifier for the covenant (subordinate-covenant-identifier) and the subordinate will always provide the supervisor's local identifier for the covenant (supervisor-covenant-identifier).

Upon receiving a Request for Bid request from the supervisor, a planner should perform the following steps:

1. If a planner receives a Request for Bid request with invalid parameters, it should issue a Request for Bid response–. No further action is necessary.
2. If a planner receives a Request for Bid request and determines that it cannot or will not bid, it should issue a Request for Bid response+ with a covenant state of No Covenant.
3. Otherwise, the planner should issue a Request for Bid response+ with a covenant-state of Bidding.

4. If the terminated-flag parameter in the original Request for Bid request is set to true, then the plan which is specified in the request is a production plan and not a production-managed plan (the production plan was partially executed and is being replanned due to an error situation). If this is the case, then in steps 5. and 6., the planner should not schedule steps which are in the Executed step-state; it should only schedule the portion of the plan which was not executed.

5. In general the planner should:

   (a) Use its internal planning algorithm to schedule the operation to produce a production plan. This may involve issuing Break Covenant requests for covenants which need to be broken in order to bid this operation.

   (b) Issue a Covenant Status message to the supervisor advertising its bid.

6. If the planner has subordinate planners, it should traverse the specified production-managed plan and perform the following steps for each step in the plan that requires scheduling or resource allocation (generating a production plan):

   (a) Issue Request for Bid requests to some collection of its subordinates who can perform the step.

   (b) After receiving some number of Covenant Status messages from subordinates advertising their bids, select one bid which should be accepted.

   (c) Issue an Accept Bid request to the subordinate whose bid was selected.

   (d) If an Accept Bid response– is received from that subordinate, then choose another bid and repeat steps (c) and (d). If there are no more bids to accept, then change the parameters to the Request for Bid and repeat starting with step (a).

   (e) If an Accept Bid response+ is received, then issue Remove Covenant requests to each of the other subordinates who provided a bid but were not issued an Accept Bid request.

   (f) Start scheduling the next step in the plan. If there are no more steps in the plan, then formulate a bid and issue a Covenant Status message to the supervisor advertising its bid.

5.6.5 Update Covenant Constraints

An Update Covenant Constraints message is issued by a planner to a subordinate. It allows the planner to update the set of constraints governing an outstanding covenant with that subordinate. The parameters for both the Update Covenant Constraints request and its responses are as follows:

Parameters on Request:

subordinate-covenant-identifier
planning-strategy
priority
earliest-start-time (CONDITIONAL)
latest-completion-time (CONDITIONAL)

Parameters on Response+:

supervisor-covenant-identifier
Covenant Status Object

Parameters on Response–:

supervisor-covenant-identifier
error: Error Object
Covenant Status Object (CONDITIONAL)

If one of the conditional parameters in the Update Covenant Constraints request is omitted, then that constraint is removed. These semantics are consistent with that of the Request for Bid: existent constraints should be specified and constraints to be removed should be omitted from the request. If a planner issues an Update Covenant Constraints response–, the conditional parameter Covenant Status Object is specified if it contains information related to rejecting the request.

An Update Covenant Constraints request is to change all of the planning parameters which were specified in the Request for Bid request: planning-strategy, priority, earliest-start-time or latest-completion-time. This information may be used by the subordinate planner to reschedule a covenant. If a planner wishes to give a subordinate flexibility to reschedule a covenant within certain constraints but without supervised interaction, it may issue that subordinate an Update Covenant Constraints request.

Upon receiving an Update Covenant Constraints request from the supervisor, a planner should perform the following steps:

1. If a planner receives an Update Covenant Constraints request for an invalid covenant, it should issue an Update Covenant Constraints response–. No further action is necessary.
2. If any of the constraints are invalid (e.g., unsupported planning-strategy, specified times violate covenant as currently specified, times extend past planning horizon), issue an Update Covenant Constraints response– and ignore ALL new constraint values specified in the request.
3. Otherwise, update the covenant information to reflect the new constraints. The planner is required to accept all new constraints except the planning-strategy.
4. The planner **may** wish to reschedule the covenant at this time. If so, it must meet the newly accepted constraints. See Section 5.1.2 on page 27 for how to schedule a covenant. It must complete rescheduling the covenant prior to responding to the Update Covenant Constraints request.
5. The planner should issue an Update Covenant Constraints response+.
6. At any point in the future the planner may modify any covenant such that it continues to satisfy the constraints. The most likely modification of a covenant will be shuffling it around in order to accept or bid on other operations. If the planner modifies any covenant, it must issue a Covenant Status message to inform the supervisor of the changes.

5.7 Covenant Requests Issued By A Subordinate Planner

Covenant Requests are requests which a planner may issue to either the supervisor or one of its subordinates to perform some action to a set of covenants or to obtain information about a set of covenants. This section presents the covenant requests which a planner may issue to the supervisor:

- Break Covenants
- Covenant Status
- Covenants Broken

These requests may only be issued to a supervising planner which is in the Active administrative state. If a supervising planner receives one of these requests while in the Available administrative state, the request should be ignored.

5.7.1 Break Covenants

A Break Covenants message is issued by a planner to the supervisor to request permission to cancel a set of covenants in order to formulate a specified other covenant (the causing-covenant), or to, in general, cancel all bids, contracts or covenants. The parameters for the Break Covenants request and its response are as follows:

Parameters on Request:

covenants: CHOICE OF{
causing-covenant: supervisor-covenant-identifier
list of supervisor-covenant-identifier
OR All-Bids
OR All-Contracts
OR All-Covenants}

Parameters on Response:

acceptance: Acceptance Object

The Break Covenants message at no time implies that the set of covenants has already been cancelled or is in the process of being cancelled; if the supervisor grants permission, it merely implies that it will allow those covenants to be cancelled in order to formulate a specified other covenant[6]. A planner wishing to request permission to cancel a set of covenants should issue a Break Covenants request to the supervisor. If the supervisor denies permission, then the planner may ***not*** overlap those covenants with the specified new covenant; it must formulate a bid for a different time. If the supervisor grants permission, then the planner may overlap those covenants with the specified new covenant. If the planner does overlap those covenants with the specified new covenant, those overlapped covenants must be specified in the conflicting-covenants parameter of the Covenant Status Object when the new bid is reported to the supervisor.

The algorithms that a planner goes through to determine if a set of covenants may be broken is beyond the scope and intention of this specification. Only the external behavior and semantics need to be preserved. If the planner determines that it is acceptable to break the specified set of covenants, it should issue a Break Covenants response granting permission. If the planner determines that it is not acceptable to break the specified set of covenants and wishes to deny permission, it should issue a Break Covenants response specifying an error code.

5.7.2 Covenant Status

A Covenant Status message is issued by a planner to inform the supervisor of the current status of a covenant. The Covenant Status message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

supervisor-covenant-identifier
Covenant Status Object
list of output-plan-parameter: Plan Parameter Object (CONDITIONAL)

---

6. The Covenants Broken message is used to actually cancel those covenants. It is issued by a planner at the time the supervisor requests the planner to accept that covenant.

If a planner issues a Covenant Status message, the conditional parameter list of output-plan-paramete, is specified if the production-managed plan used to schedule the covenant requires output plan parameters.

A Covenant Status message should be issued any time one of the parameters in the covenant's status changes (i.e., if any of the parameters of the Covenant Status Object or the output plan parameters have changed). It is invalid to receive a Covenant Status message while in the Available administrative state; the message should be ignored.

5.7.3 Covenants Broken

A Covenants Broken message is issued by a planner to the supervisor; it notifies the supervisor that a set of covenants have been cancelled. The parameters for both the Covenants Broken request and its response are as follows:

Parameters on Request:

covenants: CHOICE OF{
list of supervisor-covenant-identifier
OR All-Bids
OR All-Contracts
OR All-Covenants}

Parameters on Response:

<none>

A planner wishing to cancel a set of covenants it has with the supervisor should issue a Covenants Broken request to the supervisor. This message is used to cancel covenants in order to accept another covenant or because the resource for which the planner is planning has had a change in availability and a set of covenants can no longer be fulfilled at the negotiated time. In addition to informing the supervisor that a set of covenants are broken, the planner issuing the request must also dispose of those covenants and associated production plans.

Upon receiving a Covenants Broken request from a subordinate planner, the planner should issue a Covenants Broken response (this response is required in order to synchronize planner activity). It should then attempt to replan its affected covenants (those covenants with subcovenants that were broken).

6 Job Control Interface

This section presents the job control interface, including:

- state models for both a job controller's administrative state and its tasks,
- the data objects used by the job control interface, and
- the messages which are provided by the job control interface — administrative job control messages, task requests issued by a supervising job controller and task requests issued by a subordinate job controller.

6.1 Job Controller Administrative States

Each job controller has a notion of administrative state. Table 6. enumerates and defines the valid administrative states for a job controller.

State	Definition
Available	The job controller has either 1. started cold and is ready to establish a connection to the supervisor, or 2. received and processed a Deactivate message from the supervisor, has released its connection to the supervisor, deactivated its subordinates and is ready to be halted.
Active	The job controller is up. It is 1. connected to and activated by the supervisor, 2. connected to or in the process of activating its subordinates, 3. ready to accept and process task message, and 4. accepting and initiating new tasks.
Pausing	The job controller is up. It has received a Pause All Tasks message from the supervisor or the active guardian. It is accepting new task requests and executing them until a checkpoint is reached. It is bringing currently active tasks to the next checkpoint and suspending their execution.
Paused	The job controller is up. All tasks are suspended at a checkpoint. It is accepting but not initiating new task requests; it will not resume execution of outstanding tasks until directed to do so.
Terminating	The job controller is up. It has received a Terminate All Tasks message from the supervisor or the active guardian. It is in the process of terminating all outstanding tasks. It will not accept any new task requests.
Terminated	The job controller is up. It has terminated all outstanding tasks and is not accepting any new task requests.
E-stopped	The job controller has received and processed an Emergency Stop message from the supervisor or the active guardian. The job controller is in an unknown state: it is no longer processing any tasks and any peripheral equipment directly under its control has been moved to a safe state. It may or may not be connected to the supervisor and subordinates.

Table 6. Job Controller Administrative-state Definitions

Valid transitions among the job controller administrative states and the job control messages that cause those transitions are depicted in Figure 9. on page 43.

6.2 Task States and Task Management States

Each outstanding task that a job controller is currently executing has an associated task-state and task-management-state.

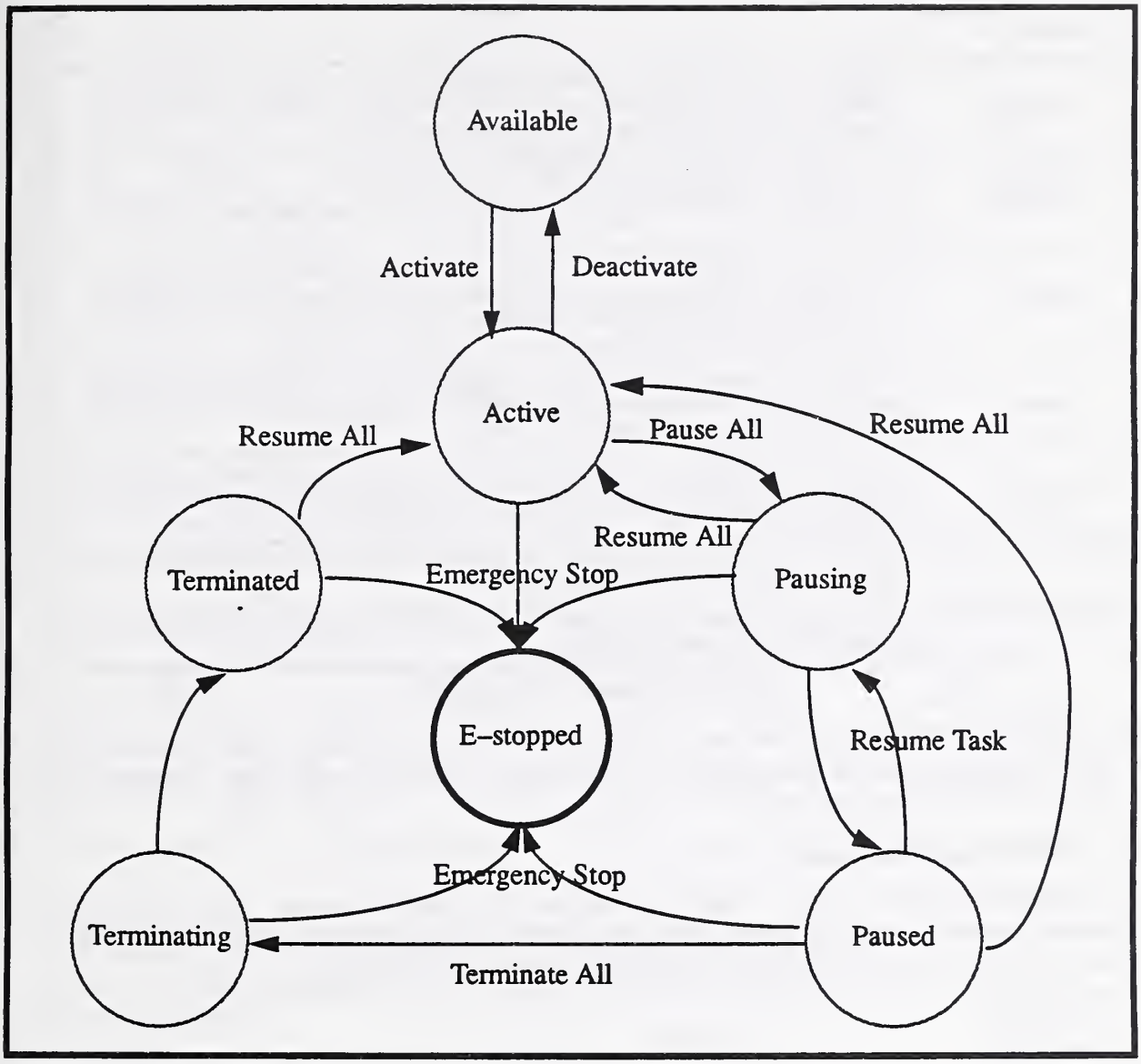


Figure 9. Job Controller Administrative-state Transition Diagram

6.2.1 Task States

A task-state denotes the current state of a task; valid task-states are:

- Aborted
- Active
- Completed
- Deferred
- Paused
- Terminated
- Waiting for Guardian
- Waiting for Planner

A task-state is either a transitional task-state or a terminal task-state. Active, Paused, Waiting for Guardian and Waiting for Planner are transitional task-states. Aborted, Completed, Deferred and Terminated are terminal task-states. Table 7. on page 44 defines each of the task-states.

Task-state	Definition
Aborted	The task was abnormally terminated as quickly as possible by the direction of the supervisor, the active guardian or the planner. The state of the workpieces and tools involved in the task is unknown. This is a terminal state.
Active	The task is currently executing. This is a transitional state.
Completed	The task has completed execution normally; no errors occurred. This is a terminal state.
Deferred	The task was abnormally terminated as quickly as possible by the direction of the supervisor, the active guardian or the planner. The task is known to be completely repeatable from the beginning without damage to the workpieces or replacement of the tools. This is a terminal state.
Paused	The task's execution has been temporarily suspended at a checkpoint, pending some external event or a message from the supervisor or the active guardian to resume the task. This is a transitional state.
Terminated	The task was terminated at a checkpoint so that execution may continue at some later point. The state of the workpieces and tools involved in the task is known to be as specified for completion of the checkpointed step. This is a terminal state.
Waiting for Guardian	The task's execution has been temporarily suspended, pending instructions from the active guardian. This is a transitional state.
Waiting for Planner	The task's execution has been temporarily suspended, pending instructions from the planner. This is a transitional state.

Table 7. Task-state Definitions

Valid transitions among the task-states are depicted in Figure 10. on page 45. The messages that cause task-state transitions are many and their effects inter-related. As a result, the messages that cause each transition are not shown.

6.2.2 Task Management States

A task-management-state further refines the notion of a task's current state. The task-management-state denotes that a task is in the process of switching from a transitional task-state to either a terminal task-state or the Paused task-state. Valid task-management-states are:

- Aborting
- Deferring
- Normal (no transition is currently in-process)
- Pausing
- Terminating

Table 8. on page 45 defines the valid task-management-states.

6.3 Complex Data Objects Used in the Job Control Interface

In addition to the data objects described in Section 4 on page 24, there is only a single additional complex data object used in the job control interface: the Task Status Object.

The Task Status Object is a data object which contains information describing a single task. It is used to convey task status information between supervisory and subordinate job controllers[7].

7. In addition, it is used to report information to a guardian through the guardian job control interface. See Section 8.1.3 on page 75 for more information about its use in the guardian job control interface.

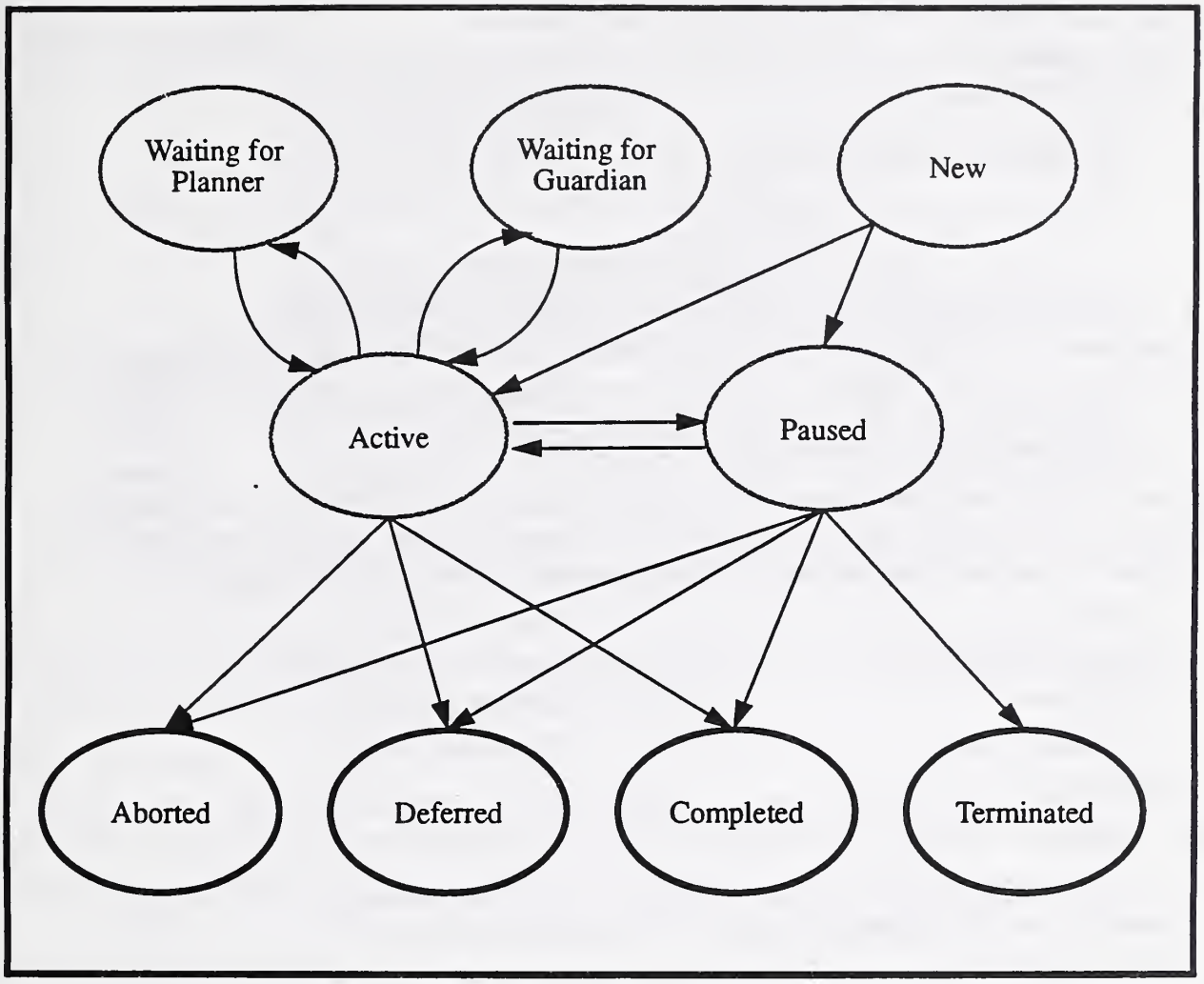


Figure 10. Task-state Transition Diagram

Task-management-state	Definition
Aborting	The job controller has received and is processing an Abort Task directive from either the supervisor, the active guardian or the planner. It has not completed aborting the task.
Deferring	The job controller has received and is processing a Defer Task directive from either the supervisor, the active guardian or the planner. It has not completed deferring the task.
Normal	The job controller has received no directives from the supervisor, the active guardian or the planner which affects the state of the task.
Pausing	The job controller has received and is processing a Pause Task or Pause All Tasks directive from either the supervisor or the active guardian. It has not completed pausing the task
Terminating	The job controller has received and is processing a Terminate Task or Terminate All Tasks directive from either the supervisor, the active guardian or the planner. It has not completed terminating the task.

Table 8. Task-management-state Definitions

The Task Status Object contains the following data elements:

task-state
task-management-state
late
execution-time: CHOICE OF{
start-time
completion-time
OR estimated-remaining-duration}
list of output-plan-parameter: Plan Parameter Object (CONDITIONAL)

The parameter list of output-plan-parameter is only specified in the Task Status Object if the production plan associated with the task contains output plan parameters; otherwise it is empty.

6.4 Job Controller Administrative Requests

The job controller administrative requests are requests which a job controller may issue to the supervisor or one of its subordinates to alter its administrative status or request/provide information about its administrative status. The job controller administrative requests are:

- Activate
- Administrative Status
- Deactivate
- Emergency Stop
- Identify
- Pause All Tasks
- Report Administrative Status
- Resume All Tasks
- Terminate All Tasks

6.4.1 Activate

An Activate message is issued by a job controller to connect to and initialize a subordinate. The parameters for both the Activate request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A job controller wishing to activate one of its subordinates should issue it an Activate request. The Activate response+ results in the connection being established. Table 9. details what a job controller should do upon receiving an Activate request from the supervisor.

6.4.2 Administrative Status

An Administrative Status message is issued by a job controller to inform the supervisor of any change in its administrative status. The Administrative Status message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

administrative-state
link-to-supervisor-status

Administrative State	Action
Available	1. Upon receiving an Activate request, perform any internal initialization and switch to the Active administrative state. 2. If the initialization fails or the job controller cannot switch to the Active administrative state, then issue an Activate response–. No further action is necessary. 3. Otherwise, issue an Activate response+. 4. If necessary, query the database to determine who the planner and subordinates are. 5. Attempt to establish a connection to the planner. 6. Attempt to activate each subordinate by issuing an Activate request to each one. 7. If the job controller receives an Activate response– from any of its subordinates, contact the active guardian and proceed according to its instructions.
Active, Pausing, Paused, Terminating, Terminated, E-stopped	Receiving an Activate request while in the Active, Pausing, Paused, Terminating or Terminated administrative state causes a connection to be established between the receiving job controller and the supervisor. If the job controller already has a connection to the supervisor, this is a protocol violation; issue an Activate response–. Otherwise, accept the connection by issuing an Activate response+.

Table 9. Semantics of the Job Controller Activate Message

An Administrative Status message should be issued any time one of the parameters in the administrative status changes, (i.e., if the administrative state or the status of the connection between the job controller and the supervisor has changed).

6.4.3 Deactivate

A Deactivate message is issued by a job controller to disconnect from and shut down a subordinate. The parameters for both the Deactivate request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A job controller wishing to deactivate one of its subordinates should issue it a Deactivate request. The Deactivate response results in the connection being terminated. Table 10. details what a job controller should do upon receiving a Deactivate request from the supervisor.

Administrative State	Action
Available	It is a protocol violation to receive a Deactivate request while in the Available administrative state. Issue a Deactivate response–.

Table 10. Semantics of the Job Controller Deactivate Message

Administrative State	Action
Active, Pausing, Paused, Terminating, Terminated, E-stopped	1. If the job controller has any outstanding tasks, issue a Deactivate response– and continue executing the tasks. No further action is necessary. 2. If the job controller directly controls equipment, move any equipment to a safe position and respond Deactivate response+. If the equipment cannot be moved to a safe position, issue a Deactivate response–. No further action is necessary. 3. If the job controller has subordinates, issue a Deactivate response+. This will result in the connection with the supervisor being released. 4. Issue Deactivate requests to each of its subordinates. 5. If any subordinate responds with Deactivate response–, contact the active guardian and proceed according to its instructions. 6. If all subordinates respond with Deactivate response+, switch to the Available administrative state. 7. The job controller is now ready to be halted.

Table 10. Semantics of the Job Controller Deactivate Message

6.4.4 Emergency Stop

An Emergency Stop message is issued by a job controller to a subordinate; it requests the subordinate to immediately stop all physical activity. The Emergency Stop message is an unconfirmed message; it has no response. There are no parameters to the Emergency Stop message.

Table 11. details what a job controller should do upon receiving an Emergency Stop message from the supervisor.

Administrative State	Action
Available	It is a protocol violation to receive an Emergency Stop message while in the Available administrative state. Since this message does not have a response, no action is necessary.
Active, Pausing, Paused, Terminating, Terminated	1. If the job controller has subordinates, issue Emergency Stop messages to each subordinate. 2. If the job controller has any equipment directly under its control, abort all tasks and move that equipment to a safe position. 3. Switch to the E-stopped administrative state. 4. Optionally, move any necessary internal data to local or global databases, disconnect from the supervisor and subordinates and halt.
E-stopped	Receiving an Emergency Stop message while in the E-stopped administrative state has no affect on the job controller. Ignore the message.

Table 11. Semantics of the Emergency Stop Message

6.4.5 Identify

An Identify message is issued by a job controller to determine the characteristics of a subordinate. The parameters for both the Identify request and its response are as follows:

Parameters on Request:

<none>

Parameters on Response:

make
model
version
level

A job controller wishing to know the characteristics of one of its subordinates should issue that subordinate an Identify request. Upon receiving an Identify request from the supervisor, a job controller should issue an Identify response, specifying the make, model and version of the job controller, what level of job controller it is (shop, workcell, or equipment level) and any vendor extensions that are supported.

6.4.6 Pause All Tasks

A Pause All Tasks message is issued by a job controller to a subordinate; it requests the subordinate to switch to the Pausing (and eventually to the Paused) administrative state and to suspend the execution of all outstanding tasks. The parameters for both the Pause All Tasks request and its responses are as follows:

Parameters on Request:

time-frame: CHOICE OF{
desired-pausing-duration
OR as-soon-as-possible}
expected-paused-duration

Parameters on Response+:

estimated-pausing-duration

Parameters on Response–:

error: Error Object
estimated-pausing-duration (CONDITIONAL)

If a job controller issues a Pause All Tasks response–, the conditional parameter estimated-pausing-duration is specified if the reason for rejecting the request is that all activity cannot be paused within the requested desired-pausing-duration.

The purpose of this message is to temporarily halt all activity within a subordinate subsystem such that neither the equipment nor any of the tasks' workpieces are damaged. The intention is that the execution of the suspended tasks will either be resumed or terminated at some point in the future. A job controller wishing to pause a subordinate subsystem should issue the subordinate a Pause All Tasks request. The job controller may specify either a time interval within which the subordinate should suspend all activity (parameter desired-pausing-duration), or that all activity should be paused as soon as possible. In addition, the job controller will estimate how long it expects the subordinate subsystem will remain paused (parameter expected-paused-duration). This parameter is specified so that the subordinate may adjust its scheduled to accommodate suspending all activity. Table 12. details what a job controller should do upon receiving a Pause All Tasks request from the supervisor.

Administrative State	Action
Available, E-stopped	It is a protocol violation to receive a Pause All Tasks request while in the Available or E-stopped administrative state.

Table 12. Semantics of the Pause All Tasks Message

Administrative State	Action
Active, Pausing	1. Switch to the Pausing administrative state. 2. Examine each currently executing task. For each currently executing task do the following: (a) If the task is in a terminal state (Aborted, Deferred, Completed,or Terminated) or if the task's management state is Aborting, Deferring or Terminating, then no action is necessary to pause that task. (b) Identify the nearest checkpoint for each remaining task and determine the earliest-pausing-time for each task. The earliest-pausing-time is the earliest time in which that task's execution can reach the identified checkpoint. 3. Determine the earliest time in which all tasks can be paused (the maximum earliest-pausing-time of all tasks), denoted earliest-pause-all-time. 4. If the job controller is at the equipment level, the job controller should: (a) If the Pause All Tasks request was issued with the parameter as-soon-as-possible, issue a Pause All Tasks response+ with an estimated-pausing-duration of the earliest-pause-all-time. (b) If the Pause All Tasks request was issued with the parameter desired-pausing-duration and the earliest-pause-all-time is within that desired-pausing-duration, issue a Pause All Tasks response+ with an estimated-pausing-duration of the earliest-pause-all-time. (c) Otherwise, issue a Pause All Tasks response– with an estimated-pausing-duration of the earliest-pause-all-time, and switch to the Active administrative state. Continue execution of the sole outstanding task. No further action is necessary. (d) Continue executing the sole task until its identified checkpoint is reached. Suspend execution of the task, switch the task to the Paused task-state. (e) After the sole task is either in a terminal task-state or in the Paused task-state, switch to the Paused administrative state and issue an Administrative Status message. 5. If the job controller has subordinates, the job controller should: (a) If the Pause All Tasks request was issued with the parameter as-soon-as-possible, issue a Pause All Tasks response+ with an estimated-pausing-duration of the earliest-pause-all-time and then do the following: Issue Pause All Tasks requests to each of its subordinates so that its subordinates will pause all activity as soon as possible. After ***all*** subordinates issue a Pause All Tasks response+ continue executing each task until its identified checkpoint is reached (either by the workcell job controller or its subordinates reporting that the task is Paused). Switch the task to the Paused task-state. After all tasks are in the Paused task-state and all subordinates are in the Paused administrative state, switch to the Paused administrative state and issue an Administrative Status message. (b) If the Pause All Tasks request was issued with the parameter desired-pausing-duration and the earliest-pause-all-time is within that desired-pausing-duration, issue a Pause All Tasks response+ with an estimated-pausing-duration of the earliest-pause-all-time. Continue executing each task until its identified checkpoint is reached. Switch the task to the Paused task-state. After all tasks are in the Paused task-state, switch to the Paused administrative state and issue an Administrative Status message. (c) If the Pause All Tasks request was issued with the parameter desired-pausing-duration and the earliest-pause-all-time is not within the desired-pausing-duration, then: Issue Pause All Tasks requests to each of its subordinates to determine if ***all*** of its subordinates can pause their tasks within the desired-pausing-duration. If any subordinate issues a Pause All Tasks response–, then issue a Pause All Tasks response– to the supervisor with the longest estimated-pausing-duration received from its subordinates or the earliest-pause-all-time it calculated earlier. Issue Resume All Tasks requests to all subordinates who responded positively to the Pause All Tasks requests. Switch back to the Active administrative state. Continue execution of all tasks. No further action is necessary. If ***all*** subordinates issue a Pause All Tasks response+ then issue a Pause All Tasks response+ to the supervisor with the longest estimated-pausing-duration received from its subordinates. Continue executing each task until its identified checkpoint is reached (either by the workcell job controller or its subordinates reporting that the task is Paused). Switch the task to the Paused task-state. After all tasks are in the Paused task-state and all subordinates are in the Paused administrative state, switch to the Paused administrative state. 6. Continue to monitor the estimated-completion-time of each task to determine if it becomes late by using the expected-pause-duration specified in the Pause All Tasks request.

Table 12. Semantics of the Pause All Tasks Message

Administrative State	Action
Paused	When a job controller receives a Pause All Tasks request while in the Paused administrative state, it should update its information concerning when it is expected to continue executing its outstanding tasks (based upon the parameter expected-pause-duration). The job controller should issue a Pause All Tasks response+ with an estimated-pausing-duration of 0. It should continue to monitor the estimated-completion-time for each task to determine if it becomes late. It should use the newly received expected-pause-duration to determine if tasks are late.
Terminating, Terminated, E-stopped	It is a protocol violation to receive a Pause All Tasks request while in the Terminating, Terminated or E-stopped administrative state.

Table 12. Semantics of the Pause All Tasks Message

6.4.7 Report Administrative Status

The Report Administrative Status message is issued by a job controller to request administrative status information from a subordinate. The parameters for both the Report Administrative Status request and its response are as follows:

Parameters on Request:

<none>

Parameters on Response:

administrative-state
link-to-supervisor-status

A job controller wishing to receive updated administrative status from one of its subordinates should issue it a Report Administrative Status request. Upon receiving a Report Administrative Status request from the supervisor, a job controller should issue a Report Administrative Status response specifying its current administrative state and the current state of the connection between itself and the supervisor.

6.4.8 Resume All Tasks

A Resume All Tasks message is issued by a job controller to a subordinate; it requests the subordinate to switch from the Pausing, Paused or Terminated administrative state to the Active administrative state and resume the execution of all outstanding tasks. The parameters for both the Resume All Tasks request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

The purpose of this message is to resume the activity of a subordinate subsystem which has previously been issued a Pause All Tasks or Terminate All Tasks request. A job controller wishing to resume the activity of a subordinate subsystem should issue the subordinate a Resume All Tasks request. Table 13. on page 52 details what a job controller should do upon receiving a Resume All Tasks request from the supervisor.

Administrative State	Action
Available, E-stopped	It is a protocol violation to receive a Resume All Tasks while in the Available or E-stopped administrative state.
Active	1. Issue a Resume All Tasks response+ to the supervisor. 2. If the job controller has subordinates, issue a Resume All Tasks request to each subordinate.
Pausing, Paused	1. Switch to the Active administrative state. 2. Issue a Resume All Tasks response+ to the supervisor. 3. If the job controller has subordinates, issue a Resume All Tasks request to each subordinate. 4. For each task that is in the Paused task-state and the Normal task-management-state, switch the task to the Active task-state. Resume the execution of these outstanding tasks.
Terminating	It is a protocol violation to receive a Resume All Tasks while in the Terminating administrative state.
Terminated	1. Switch to the Active administrative state. 2. Issue a Resume All Tasks response+ to the supervisor. 3. If the job controller has subordinates, issue a Resume All Tasks request to each subordinate. 4. Resume the execution of new tasks.

Table 13. Semantics of the Resume All Tasks Message

6.4.9 Terminate All Tasks

A Terminate All Tasks message is issued by a job controller to a subordinate; it requests the subordinate to switch to the Terminating (and eventually to the Terminated) administrative state, to indefinitely suspend the execution of all outstanding tasks and to no longer accept Execute Task requests. The parameters for both the Terminate All Tasks request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A job controller wishing to terminate the activity of a subordinate should issue it a Terminate All Tasks request. Table 14. details what a job controller should do upon receiving a Terminate All Tasks request from the supervisor.

Administrative State	Action
Available, Active, Pausing, E-stopped	It is a protocol violation to receive a Terminate All Tasks request while in the Available, Active, Pausing or E-stopped administrative state.

Table 14. Semantics of the Job Controller Terminate All Tasks Message

Administrative State	Action
Paused	1. Switch to the Terminating administrative state. 2. Issue a Terminate All Tasks response+ to the supervisor. 3. If the job controller is at the equipment level and the sole task is in the Paused task-state, then do the following: Abort the task according to internal procedures, Switch the task to the Terminated task-state and issue a Plan Finished request to the planner. Switch to the Terminated administrative state. 4. If the job controller is at the equipment level and there is no outstanding task or the sole task is not in the Paused task-state (then it must be in a terminal task-state), switch to the Terminated administrative state. Issue an Administrative Status message to the supervisor. 5. If the job controller has subordinates, issue a Terminate All Tasks request to each subordinate. For each task that is in the Paused task-state, do the following: Abort any local subtasks according to internal procedures. After all subtasks have been terminated (either by the workcell or some subordinate), switch the task to the Terminated task-state and issue a Plan Finished request to the planner. After receiving an Administrative Status message from each subordinate advertising a Terminated administrative state, switch to the Terminated administrative state and issue an Administrative Status message to the supervisor.
Terminating, Terminated	Receiving a Terminate All Tasks request while in the Terminating or Terminated administrative state will have no affect on the internal state of the job controller. Issue a Terminate All Tasks response+.

Table 14. Semantics of the Job Controller Terminate All Tasks Message

6.5 Task Requests Issued By A Supervising Job Controller

Task Requests are requests which a job controller may issue to either the supervisor or one of its subordinates to perform some action to a task or to obtain information about a set of tasks. This section presents the task requests which a job controller may issue to one of its subordinates:

- Abort Task
- Defer Task
- Execute Task
- Pause Task
- Report Tasks
- Resume Task
- Terminate Task

These requests may not be issued to a subordinate job controller which is in the Available or E-stopped administrative state. If a subordinate job controller receives one of these requests while in the Available or E-stopped administrative state, the request should be ignored and an Administrative Status message should be issued to the supervisor.

6.5.1 Abort Task

An Abort Task message is issued by a job controller to a subordinate; it requests that a specified subordinate task be aborted. When aborting a task, no consideration is given to preserving the state of the workpiece or possibly even the replaceable tools; the only concern is not to damage the equipment. The parameters for both the Abort Task request and its responses are as follows:

Parameters on Request:

subordinate-task-identifier

Parameters on Response+:

supervisor-task-identifier

Parameters on Response–:

supervisor-task-identifier
error: Error Object
Task Status Object (CONDITIONAL)

If a job controller issues an Abort Task response–, the conditional parameter Task Status Object is specified if it contains information related to rejecting the request.

A job controller wishing to abort one of its subtasks should issue an Abort Task request to the subordinate whose task corresponds to the job controller's subtask.

Upon receiving an Abort Task request from the supervisor, the job controller should perform the following steps:

1. If the job controller's administrative state is Terminated, issue an Abort Task response–. No further action is necessary.
2. If the specified task is an invalid task, issue an Abort Task response–. No further action is necessary.
3. If the specified task is in the Aborted task-state, issue an Abort Task response+ to the supervisor.
4. If the specified task is in some other terminal task-state, issue a Task Status message to the supervisor specifying its terminal state and then issue an Abort Task response– to the supervisor. No further action is necessary.
5. If the specified task's management state is Aborting, proceed to step 10.
6. Otherwise, set the specified task's management state to Aborting.
7. If the Abort Task request invalidates any outstanding requests for the specified task, respond negatively to those outstanding requests. If the Abort Task request invalidates any requests that the job controller has issued, then cancel those requests.
8. If the job controller is at the equipment level, abort the task according to internal procedures.
9. If the job controller has subordinates, issue Abort Task requests to each subordinate with subtasks in support of the specified task and wait for responses.
   If any subordinate responds with an Abort Task response–, issue an Abort Task response– to the supervisor and contact the active guardian. Proceed according to the active guardian's instructions.
   If all subordinates respond with Abort Task responses+, then the task has been successfully aborted.
10. After the task is aborted, switch the task into the Aborted task-state and clear the task's management state.
11. Issue a Plan Finished request to the planner.
12. Issue an Abort Task response+ to the supervisor.

6.5.2 Defer Task

A Defer Task message is issued by a job controller to a subordinate; it requests that a specified subordinate task be deferred. A task may be deferred if it is completely repeatable, from the beginning, without damage to the workpiece(s). The parameters for both the Defer Task request and its responses are as follows:

Parameters on Request:

subordinate-task-identifier

Parameters on Response+:

supervisor-task-identifier

Parameters on Response–:

supervisor-task-identifier
error: Error Object
Task Status Object (CONDITIONAL)

If a job controller issues a Defer Task response–, the conditional parameter Task Status Object is specified if it contains information related to rejecting the request.

A job controller wishing to defer one of its subtasks[8] should issue a Defer Task request to the subordinate whose task corresponds to the job controller's subtask.

Upon receiving a Defer Task request from the supervisor, the job controller should perform the following steps:

1. If the job controller's administrative state is Terminating or Terminated, issue a Defer Task response–. No further action is necessary.
2. If the specified task is an invalid task, issue a Defer Task response–. No further action is necessary.
3. If the specified task is in the Deferred task-state, issue a Defer Task response+. No further action is necessary.
4. If the specified task is in some other terminal task-state, issue a Task Status message to the supervisor specifying its terminal state and then issue a Defer Task response– to the supervisor. No further action is necessary.
5. If the specified task's management state is Aborting or Terminating, issue a Defer Task response–. No further action is necessary.
6. If the specified task's management state is Deferring, proceed to step 11.
7. Otherwise, set the task's management state to Deferring.
8. A Defer Task request will invalidate any received or issued Pause Task requests. If any such request was received, issue a response–. If any such requests were issued, cancel them.
9. If the job controller is at the equipment level, abort the task according to internal procedures.
10. If the job controller has subordinates, then do the following:
    (a) Abort any locally executing subtasks according to internal procedures.
    (b) Pause each subtask which is being executed by a subordinate (see Section 6.5.4 on page 57).
    (c) After all subtasks are paused, Terminate each subtask which is being executed by a subordinate (see Section 6.5.7 on page 61).
    (d) If any subtask cannot be paused or terminated, issue a Defer Task response– to the supervisor and contact the active guardian for assistance. Proceed according to its instructions.

8. It is unlikely that a job controller will have the intelligence to determine whether or not a task or subtask is deferrable. It is expected that the active guardian will instruct a job controller to defer a task,.

(e) After all locally executing subtasks are aborted and all subtasks executed by subordinates have been terminated, the task is deferred.

11. After the task is deferred, switch the task into the Deferred task-state and set the task's management state to Normal.
12. Issue a Plan Finished request to the planner.
13. Issue a Defer Task response+ to the supervisor.

6.5.3 Execute Task

An Execute Task message is issued by a job controller to a subordinate; it requests the creation and execution of a new task. The parameters for both the Execute Task request and its responses are as follows:

Parameters on Request:

supervisor-task-identifier
production-plan: Plan Identifier Object
list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)

Parameters on Response+:

supervisor-task-identifier
subordinate-task-identifier

Parameters on Response–:

supervisor-task-identifier
error: Error Object

A job controller wishing to begin the execution of one of its subtasks which is scheduled to be executed by a subordinate should issue an Execute Task request to that subordinate. The job controller specifies a local identifier for the task, the production plan to be executed (parameters production-plan-identifier and production-plan-version) and any input plan parameters that the production plan may require.

If the subordinate job controller issues an Execute Task response+, it will specify a local identifier for that task. In all future communications regarding this task, the supervisor will always provide the subordinate's local identifier for the task (subordinate-task-identifier) and the subordinate will always provide the supervisor's local identifier for the task (supervisor-task-identifier).

Upon receiving an Execute Task request from the supervisor, the job controller should perform the following steps:

1. If the job controller's administrative state is Terminating or Terminated, issue an Execute Task response–. No further action is necessary.
2. If the specified task already exists (i.e., the supervisor-task-id is identical to a currently outstanding task), issue an Execute Task response–. No further action is necessary.
3. Validate the specified production plan. If it is invalid, issue an Execute Task response– No further action is necessary.
4. Issue an Execute Task response+.
5. If the job controller's administrative state is Active, then do the following:
   (a) Create a new task with a task-state of Active and a task-management state of Normal.
   (b) Execute the task according to the steps in the production plan and the job controller's internal task execution algorithms.

6. If the job controller's administrative state is Pausing or Paused, then do the following:
   (a) Create a new task with a task-state of Active and a task-management-state of Pausing.
   (b) If the task can be paused (i.e., a checkpoint can be reached) without the movement of any equipment (assuming that any request to a subordinate implies equipment movement), then execute the task until a checkpoint is reached and then switch the task to the Paused task-state. No further action is necessary.
   (c) If the task cannot be paused without performing manufacturing steps, then switch the subtask to the Waiting for Guardian task-state and contact the active guardian. Proceed according to the active guardian's instructions.

6.5.4 Pause Task

A Pause Task message is issued by a job controller to a subordinate; it requests a specified subordinate task's execution to be suspended. The parameters for both the Pause Task request and its responses are as follows:

Parameters on Request:

subordinate-task-identifier
time-frame: CHOICE OF {
desired-pausing-duration
OR as-soon-as-possible}
expected-paused-duration

Parameters on Response+:

supervisor-task-identifier
estimated-pausing-duration

Parameters on Response–:

supervisor-task-identifier
error: Error Object
estimated-pausing-duration (CONDITIONAL)
Task Status Object (CONDITIONAL)

If a job controller issues a Pause Task response–, the conditional parameter estimated-pausing-duration is specified if the reason for rejecting the request is because the specified task cannot be paused within the requested desired-pausing-duration. The conditional parameter Task Status Object is specified if it contains information related to rejecting the request.

A job controller wishing to suspend the execution of one of its subtasks should issue a Pause Task request to the subordinate whose task corresponds to the job controller's subtask. The job controller may specify either a time interval within which its subtask should reach the paused task-state (parameter desired-pausing-duration), or that its subtask should be paused as soon as possible. In addition, the job controller will estimate how long its subtask's execution is expected to be suspended (parameter expected-paused-duration). This parameter is specified so that the subordinate may adjust its schedule to accommodate the suspended execution.

Upon receiving a Pause Task request from the supervisor, a job controller should perform the following steps:

1. If the job controller's administrative state is Terminating or Terminated, issue a Pause Task response–. No further action is necessary.

2. If the specified task is an invalid task, issue a Pause Task response–. No further action is necessary.
3. If the specified task is in a terminal task-state (Aborted, Deferred, Completed or Terminated) or if the task's management state is Aborting, Deferring or Terminating, then issue a Pause Task response–. No further action is necessary.
4. If the job controller's administrative state is Paused, then do the following:
   (a) Update the information associated with that task specifying when it is expected to continue executing that task (based upon the parameter expected-pause-duration).
   (b) Continue to monitor the estimated-completion-time of the task to determine if it becomes late. Use the newly received expected-pause-duration to determine if it becomes late.
5. If the job controller's administrative state is Active or Pausing, then do the following:
   (a) Set the specified task's management state to Pausing.
   (b) Identify the nearest future checkpoint for the specified task and determine the earliest time in which it can pause the task (denoted 'earliest-pausing-time').
   (c) If the job controller has no subordinates or the specified task does not involve any subordinates, it should:
      >> If the Pause Task request was issued with the parameter as-soon-as-possible, issue a Pause Task response+ with an estimated-pausing-duration of the earliest-pausing-time.
      >> If the Pause Task request was issued with the parameter desired-pausing-duration and the earliest-pausing-time is within that desired-pausing-duration, issue a Pause Task response+ with an estimated-pausing-duration of the earliest-pausing-time.
      >> Otherwise, issue a Pause Task response– with an estimated-pausing-duration of the earliest-pausing-time, clear the task's management state and continue execution of the task. No further action is necessary.
      >> Continue execution of the task until the identified checkpoint is reached, and then suspend execution of the task, switch the task to the Paused task-state and clear the task's management state.
   (d) If the job controller has subordinates, it should:
      >> If the Pause Task request was issued with the parameter as-soon-as-possible, then issue a Pause Task response+ with an estimated-pausing-duration of the earliest-pausing-time. Then do the following:
         >>> Issue Pause Task requests to each subordinate that is executing a subtask in support of that task in order to pause the task as soon as possible.
         >>> Continue execution of the task until the identified checkpoint(s) are reached (either by this job controller or its subordinates reporting Paused), then switch the task to the Paused task-state and clear the task's management state. Issue a Task Status message to the supervisor.
      >> If the Pause Task request was issued with the parameter desired-pausing-duration and the earliest-pausing-time is within that desired-pausing-duration, then issue a Pause Task response+ with an estimated-pausing-duration of the earliest-pausing-time. The do the following:
         >>> Continue execution of the task until the identified checkpoint(s) are reached, then switch the task to the Paused task-state and clear the task's management state. Issue a Task Status message to the supervisor.
      >> If the Pause Task request was issued with the parameter desired-pausing-duration and the earliest-pausing-time is not within that desired-pausing-duration, then do the following:

>>> Issue Pause Task requests to each subordinate that is executing a subtask in support of that task to determine if all subordinates can pause their tasks within the desired-pausing-duration.

>>> If all of the involved subordinates issue Pause Task response+, then the job controller should issue a Pause Task response+ to the supervisor with the longest estimated-pausing-duration (received by a subordinate or determined locally).

>>> If all of the subordinates that responded with Pause Task response– specified an estimated-pausing-duration (i.e., more time is required to pause the task), the job controller should issue a Pause Task response– to the supervisor with the longest estimated-pausing-duration (received by a subordinate or determined locally), issue Resume Task requests to all subordinates who responded positively to the Pause Task requests, clear the task's management state and continue execution of the task. No further action is necessary.

>>> If any subordinate responded Pause Task response– for any other reason, then the job controller should issue a Pause Task response– to the supervisor, issue Resume Task requests to all subordinates who responded positively, clear the task's management state and continue execution of the task. No further action is necessary.

>>> Continue execution of the task until the identified checkpoint(s) are reached (either by the workcell job controller or its subordinates reporting Paused), then switch the task to the Paused task-state and clear the task's management state.

(e) Continue to monitor the estimated-completion-time of the task to determine if it becomes late by using the expected-pause-duration specified in the Pause Task request.

6.5.5 Report Tasks

A Report Tasks message is issued by a job controller to a subordinate; it requests status information to be reported on a set of specified subordinate tasks. The parameters for both the Report Tasks request and its response are as follows:

Parameters on Request:

```
Tasks: CHOICE OF {
            list of subordinate-task-identifier
        OR All-Tasks}
```

Parameters on Response:

```
list of [
    subordinate-task-identifier
    supervisor-task-identifier
    status:     CHOICE OF{
                    error: Error Object
                OR status: Task Status Object}
] (CONDITIONAL)
```

If a job controller issues a Report Tasks request and it wishes to receive status on all outstanding tasks, All-Tasks is specified. If All-Tasks is specified in a Report Tasks request and a job controller does not currently have any outstanding tasks, then the Report Tasks response has an empty list.

A job controller wishing to receive updated status for a collection of its subtasks should issue a Report Tasks request to each of its subordinates whose tasks correspond to the job controller's subtasks.

Upon receiving a Report Tasks request from the supervisor, the job controller should perform the following steps:

1. If the Report Tasks request specified All-Tasks, then for each outstanding task, collect the supervisor-task-identifier, local-task-identifier, and Task Status Object information, and issue a Report Tasks response.
2. If the Report Tasks request specified a list of tasks, for each invalid local-task-identifier in the list of tasks, specify the error-code of invalid task identifier. For each valid local-task-identifier in the list of tasks, collect the supervisor-task-identifier, local-task-identifier and Task Status Object information. Issue a Report Tasks response which includes the error information for invalid tasks and the task status information for valid tasks.

6.5.6 Resume Task

A Resume Task message is issued by a job controller to a subordinate; it requests a suspended subordinate task's execution to continue. The parameters for both the Resume Task request and its responses are as follows:

Parameters on Request:

subordinate-task-identifier

Parameters on Response+:

supervisor-task-identifier

Parameters on Response–:

supervisor-task-identifier
error: Error Object
Task Status Object (CONDITIONAL)

If a job controller issues a Resume Task response–, the conditional parameter Task Status Object is specified if it contains information related to rejecting the request.

A job controller wishing to resume the execution of one of its subtasks should issue a Resume Task request to the subordinate whose task corresponds to the job controller's subtask.

Upon receiving a Resume Task request from the supervisor, the job controller should perform the following steps:

1. If the job controller's administrative state is Terminating or Terminated, issue a Resume Task response–. No further action is necessary.
2. If the specified task is an invalid task, issue a Resume Task response–. No further action is necessary.
3. If the specified task is in a terminal task-state (Aborted, Deferred, Completed or Terminated) or the task's management state is Aborting, Deferring or Terminating, issue a Resume Task response–. No further action is necessary.
4. If the job controller's administrative state is Active or Pausing, then do the following:
   (a) If the specified task's state is not Active, switch it to the Active task-state. Set the task-management-state to Normal.
   (b) Issue a Resume Task response+.

(c) Resume the execution of local subtasks supporting the task, switching each to the Active task-state and setting its task-management-state to Normal.

(d) If the job controller has subordinates supporting the task, then for each subtask in execution by a subordinate, issue that subordinate a Resume Task request. If any subordinate responds with Resume Task response–, contact the active guardian and proceed according to its instructions.

5. If the job controller's administrative state is Paused, then do the following:
   (a) Switch to the Pausing administrative state.
   (b) Proceed according to step 4.

6.5.7 Terminate Task

A Terminate Task message is issued by a job controller to a subordinate; it requests a specified subordinate task's execution to be indefinitely suspended. The parameters for both the Terminate Task request and its responses are as follows:

Parameters on Request:

subordinate-task-identifier

Parameters on Response+:

supervisor-task-identifier

Parameters on Response–:

supervisor-task-identifier
error: Error Object
Task Status Object (CONDITIONAL)

If a job controller issues a Terminate Task response–, the conditional parameter Task Status Object is specified if it contains information related to rejecting the request.

A job controller wishing to terminate one of its subtasks should issue a Terminate Task request to the subordinate whose task corresponds to the job controller's subtask.

Upon receiving a Terminate Task request from the supervisor, the job controller should perform the following steps:

1. If the specified task is an invalid task, issue a Terminate Task response–. No further action is necessary.
2. If the specified task is in the Terminated task-state, issue a Terminate Task response+. No further action is necessary.
3. If the specified task is in some other terminal task-state, issue a Task Status message to the supervisor specifying its terminal state and then issue a Terminate Task response– to the supervisor. No further action is necessary.
4. If the specified task is not currently in the Paused task-state or the task-management-state is Aborting or Deferring, the task cannot be terminated; issue a Terminate Task response–. No further action is necessary.
5. If the specified task's management state is Terminating, proceed to step 9.
6. Otherwise, set the task-management-state to Terminating.
7. If the job controller is at the equipment level, abort the task according to internal procedures. Issue a Terminate Task response+.

8. If the job controller has subordinates, then abort any local subtasks and issue Terminate Task requests to all subordinates with subtasks in support of the specified task, and wait for responses.

   (a) If any subordinate responds with a Terminate Task response– for a reason other than the task being in the Completed or Terminated task-state, then do the following:

   >> If a subordinate task is in the Aborted task-state, issue a Terminate Task response–. No further action is necessary.

   >> Otherwise, contact the active guardian. Proceed according to the active guardian's instructions.

   (b) If all subordinates respond with Terminate Task responses+ or if all subordinates that respond with Terminate Task response– do so because the task is already in the Completed or Terminated task-state, then the task has been successfully terminated.

9. After the task is terminated, mark all Executing or Not-Yet-Executing steps in the production plan as Scheduled, switch the task into the Terminated task-state and clear the task-management-state.
10. Issue a Task Status message to the supervisor.
11. Issue a Plan Finished request to the planner.

6.6 Task Requests Issued By A Subordinate Job Controller

Task Requests are requests which a job controller may issue to either the supervisor or one of its subordinates to perform some action to a task or to obtain information about a set of tasks. This section presents the task requests which a job controller may issue to the supervisor: Task Status. This request may not be issued to the supervisor if the job controller is in the Available or E-stopped administrative state. If a supervising job controller receives a Task Status request from a subordinate which is in the Available or E-stopped administrative state, the request should be ignored.

6.6.1 Task Status

A Task Status message is issued by a job controller to inform the supervisor of the current status of a task. The Task Status message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

supervisor-task-identifier
Task Status Object

A Task Status message should be issued any time one of the parameters in the task's status changes (i.e., if any of the parameters of the Task Status Object have changed).

7 Planning To Job Control Interface

Due to the close relationship and frequent need for interaction between a job controller and the planner, a special interface is specified to support task completion and error recovery. Due to the nature of the planning-to-job-control interface, most messages are initiated by the job controller.

7.1 Job Controller Initiated Messages

There are three types of messages supported by the planning-to-job-control interface: connection-management messages, execution-related messages and error-related messages.

The connection-management messages are:

- Connect
- Disconnect

The execution-related messages are:

- Plan Executing
- Plan Finished

The error-related messages are:

- Replan Aborted Step
- Replan Deferred Step
- Replan Late Plan
- Replan Late Step
- Replan Terminated Step

It is invalid for a planner to receive an execution-related or error-related message from a job controller while in the Available administrative state. If a planner does receive one of these messages while in the Available administrative state, the message should be ignored.

The dialogues and actions which occur between a job controller and the planner in any error-related dialogue are similar:

> A job controller encounters a situation with a task which requires rescheduling. It issues a request to the planner to reschedule the production plan associated with the offending task. The planner attempts (and perhaps succeeds) to modify the production plan and responds to the job controller with an action which affects the disposition of the task. The job controller carries out the instructed action.

There are several valid actions that a planner may instruct the job controller to perform: abort the task, defer the task, terminate the task, or retry executing the task according to a modified production plan.

The reaminder of this section discusses the complex interactions between a job controller and planner, and is structured as follows. First, there is a discussion of the connection-management and execution-related messages. Next, there is a discussion of the error-related messages, including a description of the circumstances which cause a job controller to issue each message, the planner actions upon receiving each message, and the planner's response to the job controller. Finally, there is a discussion of what a job controller does as a result of receiving a response from the planner for each valid action (i.e., abort the task, defer the task, terminate the task, or retry executing the task according to a modified production plan).

7.1.1 Connect

A Connect message is issued by a job controller to the planner to establish a connection via the planning-to-job-control interface. This is the first message a job controller may issue to the planner; the connection must be established before issuing any other messages. The parameters for both the Connect request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner which receives a Connect request should determine if it is capable of accepting the connection request, and if so, issue a Connect response+. The Connect response+ results in the connection being established. A planner may reject a Connect request because it is not in the Active administrative state or because the job controller requesting a connection is not a job controller for which the receiving planner plans.

7.1.2 Disconnect

A Disconnect message is issued by a job controller to the planner to terminate a connection which is currently established via the planning-to-job-control interface. This is the last message a job controller may issue the planner without re-establishing a connection. The parameters for both the Disconnect request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner which receives a Disconnect request should determine if it should terminate the connection, and if so, issue a Disconnect response+. The Disconnect response+ results in the connection being terminated. In general, the primary reason for rejecting a Disconnect request is that the planner has outstanding requests from the job controller.

7.1.3 Plan Executing

A Plan Executing message is issued by a job controller to inform the planner that a production plan has begun execution. The Plan Executing message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

production-plan: Plan-Identifier Object

If a planner receives a Plan Executing message which specifies an invalid plan, it should ignore the message. Otherwise, it should switch the covenant that corresponds to that production plan to the Executing covenant-state and issue a Covenant Status message to the supervisor. Once a

covenant has entered the Executing covenant-state, the planner may no longer alter the covenant; all discrepancies and errors associated with that covenant are reported via the job controllers.

7.1.4 Plan Finished

A Plan Finished message is issued by a job controller to inform the planner that a production plan has completed execution. The parameters for both the Plan Finished request and its response are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
plan-state
local-task-identifier

Parameters on Response:

local-task-identifier

If a planner receives a Plan Finished request and the plan is invalid or the specified plan-state is Not-Yet-Executing or Executing, the planner should ignore the request and issue a Plan Finished response. If the plan-state is Aborted, Completed or Deferred, discard the production plan and its corresponding covenant. If the plan-state is Terminated, discard only the covenant that corresponds to the production plan; save the production plan because it is needed in order to reschedule remaining ***unexecuted*** steps in the production plan. In the case of Aborted, Deferred or Terminated, deleting the covenant may involve issuing Remove Covenants requests to subordinates for the unexecuted subcovenants. Issue a Plan Finished response only after completing the previous steps.

7.1.5 Replan Aborted Step

A Replan Aborted Step message is issued by a job controller to the planner if one of its subtasks has been unexpectedly aborted and the step needs to be replanned. In general, this will occur if a job controller receives a Task Status message from one of its subordinates specifying that a subtask has been Aborted and the job controller did not instruct its subordinate to Abort the subtask. In such cases, the job controller will switch the specified task to the Waiting for Planner task-state and issue a Replan Aborted Step request to the planner specifying the production plan and the step which was aborted. The parameters for both the Replan Aborted Step request and its responses are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
step-identifier
local-task-identifier

Parameters on Response+:

local-task-identifier
action ∈ {Abort Task, Defer Task, Resume, Terminate Task}

Parameters on Response–:

local-task-identifier
error: Error Object

Upon receiving a Replan Aborted Step request from the job controller, the planner should perform the following steps:

1. If the specified production plan or step is invalid, issue a Replan Aborted Step response–. No further action is necessary.
2. If the planner can modify the plan to recover from the anomaly, it should do so and do one of the following:

   (a) Issue a Replan Aborted Step response+ with action 'Resume'. This is issued if the job controller can continue execution of the task according to the modified production plan at the place where it encountered an aborted subtask. No further action is necessary.

   (b) Issue Replan Aborted Step response+ with action 'Defer Task'. This is issued if the planner's recovery from the anomaly requires the job controller to start the task from the beginning again. This will require rescheduling, so the job controller is requested to defer the task. No further action is necessary.
3. If the planner cannot modify the production plan to recover from the anomaly and the workpiece(s) are affected or the task's execution cannot continue until a checkpoint is reached, then the planner should issue a Replan Aborted Step response+ with action 'Abort Task'. No further action is necessary.
4. Otherwise, the planner should issue a Replan Aborted Step response+ with action 'Terminate Task'.

7.1.6 Replan Deferred Step

A Replan Deferred Step message is issued by a job controller to the planner if one of its subtasks has been unexpectedly deferred and the step needs to be replanned. In general, this will occur if a job controller receives a Task Status message from one of its subordinates specifying that a subtask has been Deferred and the job controller did not instruct its subordinate to Defer the subtask. In such cases, the job controller will switch the specified task to the Waiting for Planner task-state and issue a Replan Deferred Step request to the planner specifying the production plan and the step which was deferred. The parameters for both the Replan Deferred Step request and its responses are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
step-identifier
local-task-identifier

Parameters on Response+:

local-task-identifier
action ∈ {Abort Task, Defer Task, Resume, Terminate Task}

Parameters on Response–:

local-task-identifier
error: Error Object

Upon receiving a Replan Deferred Step request from the job controller, the planner should perform the following steps:

1. If the specified production plan or step is invalid, issue a Replan Deferred Step response–. No further action is necessary.
2. Otherwise, the planner should reschedule the production plan (see Section 5.1.3 on page 27) starting with the step which was deferred.

3. If the planner succeeds in rescheduling the plan, then the planner should issue a Replan Deferred Step response+ with action 'Resume'. Note that if the rescheduled plan violates the current covenant constraints, this information will be conveyed via the job controller's estimated-completion-time for the associated task, which may result in rescheduling at the supervisor level.
4. If the planner cannot reschedule the plan, then issue a Subcovenant Error Occurred request to the active guardian.
   (a) If the guardian responds with Subcovenant Error Occurred response with action 'Retry', then repeat starting with step 2.
   (b) If the guardian responds with Subcovenant Error Occurred response with another action, issue a Replan Deferred Step response+ to the job controller with the same action which the guardian specified.

7.1.7 Replan Late Plan

A Replan Late Plan message is issued by an equipment-level job controller to the planner if the identified task's estimated-completion-time is later than its scheduled-completion-time. In such cases, the job controller will switch the specified task to the Waiting for Planner task-state and issue a Replan Late Plan request to the planner, specifying the production plan and the current estimated-completion-time for the task associated with the production plan. The parameters for both the Replan Late Plan request and its responses are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
estimated-completion-time
local-task-identifier

Parameters on Response+:

local-task-identifier
action ∈ {Abort Task, Defer Task, Resume, Terminate Task}

Parameters on Response–:

local-task-identifier
error: Error Object

Upon receiving a Replan Late Plan request from the job controller, the planner should perform the following steps:

1. If the estimated-completion-time is past the planner's planning horizon, contact the active guardian for assistance and proceed according to its instructions.
2. If the specified production plan is invalid, issue a Replan Late Plan response–. No further action is necessary.
3. Otherwise the planner should determine if extending the current plan to accommodate the lateness will affect any existing covenants for this planner.
4. If no covenants will be affected, then do the following:
   (a) Extend the scheduled-completion-time for the production plan to the estimated-completion-time specified in the request
   (b) Issue a Replan late Plan response+ with action 'Resume'. Do not issue a Covenant Status message to the supervisor planner; this notification will be issued by the job controllers.
   (c) No further action is necessary.
5. If there are covenants that will be affected, then do the following:

(a) Issue a Break Covenants request to the supervisor to determine if those covenants may be broken so that the late one may be extended.

(b) If the supervisor grants permission to break those covenants, then do the following:

>> Issue a Covenants Broken request for the covenants to be broken and remove the covenants.

>> Extend the scheduled-completion-time for the production plan to the estimated-completion-time specified in the request.

>> Issue a Replan Late Plan response+ with action 'Resume'. Do not issue a Covenant Status message to the supervisor; this notification will be issued by the job controller.

>> No further action is necessary.

(c) If the supervisor does not grant permission to break all the necessary covenants, then do the following:

>> Identify the first future checkpoint prior to the completion of the plan. Issue a Break Covenants request to the supervisor for the covenants that need to be broken in order to reach the checkpoint. If the supervisor grants permission to break the covenants that need to be broken in order to reach the identified checkpoint, then:

>>> Issue a Covenants Broken request for the covenants for which permission was granted,

>>> Extend the scheduled-completion-time of the production plan, and

>>> Issue a Replan Late Plan response+ with action 'Terminate Task'.

>>> No further action is necessary.

>> If no checkpoint can be reached or the supervisor will not grant permission to break the necessary covenants to reach a checkpoint ***and*** the operation which the plan defines is completely repeatable with no ill consequence, then issue a Replan Late Plan response+ with action 'Defer Task'. No further action is necessary.

>> Otherwise, issue a Replan Late Plan response+ with action 'Abort Task'.

>> No further action is necessary.

7.1.8 Replan Late Step

A Replan Late Step message is issued by a job controller to the planner if one of its subordinates is reporting an estimated-completion-time later than the scheduled-start-time of a subsequent step. In such cases, the job controller will switch the specified task to the Waiting for Planner task-state and issue a Replan Late Step request to the planner specifying the production plan, the step which is late and that step's current estimated-completion-time. The parameters for both the Replan Late Step request and its responses are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
step-identifier
estimated-completion-time
local-task-identifier

Parameters on Response+:

local-task-identifier
action ∈ {Abort Task, Defer Task, Resume, Terminate Task}

Parameters on Response–:

local-task-identifier
error: Error Object

Upon receiving a Replan Late Step request from the job controller, the planner should perform the following steps:

1. If the estimated-completion-time is past the planner's planning horizon, contact the active guardian and proceed according to its instructions.
1. If the specified production plan or step is invalid, issue a Replan Late Step response–. No further action is necessary.
2. Otherwise, the planner should update the scheduled-completion-time for the specified step.
3. Then, the planner should reschedule the production plan (see Section 5.1.3 on page 27) starting with the step(s) following the step which was late.
4. If the planner succeeds in rescheduling the plan, then the planner should issue a Replan Late Step response+ with action 'Resume'. Note that if the rescheduled plan violates the current covenant constraints, this information will be conveyed via the job controller's estimated-completion-time for the associated task, which may result in rescheduling to occur at the supervisor level.
5. If the planner cannot reschedule the plan, then issue a Subcovenant Error Occurred request to the active guardian.
   (a) If the guardian responds with Subcovenant Error Occurred response with action 'Retry', then repeat starting with step 3.
   (b) If the guardian responds with Subcovenant Error Occurred response with another option, issue a Replan Deferred Step response+ to the job controller with the same action which the guardian specified.

7.1.9 Replan Terminated Step

A Replan Terminated Step message is issued by a job controller to the planner if one of its subtasks has been unexpectedly terminated and the step needs to be replanned. In general, this will occur if a job controller receives a Task Status message from one of its subordinates specifying that a subtask has been Terminated and the job controller did not instruct its subordinate to terminate the subtask. In such cases, the job controller will switch the task to the Waiting for Planner task-state and issue a Replan Terminated Step request to the planner specifying the production plan and step which was terminated. The parameters for both the Replan Terminated Step request and its responses are as follows:

Parameters on Request:

production-plan: Plan Identifier Object
step-identifier
local-task-identifier

Parameters on Response+:

local-task-identifier
action ∈ {Abort Task, Defer Task, Resume, Terminate Task}

Parameters on Response–:

local-task-identifier
error: Error Object

Upon receiving a Replan Terminated Step request from the job controller, the planner should perform the following steps:

1. If the specified production plan or step is invalid, issue a Replan Terminated Step response–. No further action is necessary.
2. Otherwise the planner should do the following:
   (a) Issue Request for Bid requests to some collection of subordinates who can perform the terminated step. In the request parameters, specify the production plan that was terminated (NOT the production-managed plan) and set the terminated-flag to true.
   (b) After receiving some number of Covenant Status messages from subordinates advertising their bids, select one bid which should be accepted.
   (c) Issue an Accept Bid request to the subordinate whose bid was selected.
   (d) If an Accept Bid response– is received from the subordinate, then choose another bid to accept and repeat steps (c) and (d). If there are no more bids to accept, then change the parameters to the Request for Bid and repeat starting with step (a).
   (e) If an Accept Bid response+ is received from the subordinate, then issue Remove Covenant requests to each of the subordinates who provided a bid but were not issued an Accept Bid request.
3. Then, the planner should reschedule the production plan (see Section 5.1.3 on page 27) starting with the step(s) following the step which was terminated.
4. If the planner succeeds in rescheduling the plan, then the planner should issue a Replan Late Step response+ with action 'Resume'. Note that if the rescheduled plan violates the current covenant constraints, this information will be conveyed via the job controller's estimated-completion-time for the associated task, which may result in rescheduling to occur at the supervisor level.
5. If the planner cannot reschedule the plan, then issue a Subcovenant Error Occurred request to the active guardian.
   (a) If the guardian responds with Subcovenant Error Occurred response with action 'Retry', then repeat starting with step 3.
   (b) If the guardian responds with Subcovenant Error Occurred response with another option, issue a Replan Deferred Step response+ to the job controller with the same action which the guardian specified.

7.1.10 Planner Instructs Job Controller to Abort Task

Upon receipt from the planner of a response+ with action 'Abort Task' to one of Replan Aborted Step, Replan Deferred Step, Replan Late Plan, Replan Late Step or Replan Terminated Step, the job controller should perform the following steps:

1. If the identified task is invalid or is not in the Waiting for Planner task-state, ignore the message. No further action is necessary.
2. If the specified task is already in a terminal task-state, issue a Task Status message to the supervisor and a Plan Finished message to the planner with the appropriate plan-state. No further action is necessary.
3. Otherwise, follow steps 5. – 11. in Section 6.5.1 on page 53.
4. Issue a Task Status message to the supervisor.

7.1.11 Planner Instructs Job Controller to Defer Task

Upon receiving from the planner a response+ with action 'Defer Task' to one of Replan Aborted Step, Replan Deferred Step, Replan Late Plan, Replan Late Step or Replan Terminated Step, the job controller should perform the following steps:

1. If the identified task is invalid or is not in the Waiting for Planner task-state, ignore the message. No further action is necessary.
2. If the specified task is already in a terminal task-state, issue a Task Status message to the supervisor and a Plan Finished message to the planner with the appropriate plan-state. No further action is necessary.
3. If the specified task's management state is Aborting or Terminating, ignore the message. No further action is necessary.
4. Otherwise, follow steps 7. – 12. in Section 6.5.2 on page 54.
5. Issue a Task Status message to the supervisor.

7.1.12 Planner Instructs Job Controller to Terminate Task

Upon receiving from the planner a response+ with action 'Terminate Task' to one of Replan Aborted Step, Replan Deferred Step, Replan Late Plan, Replan Late Step or Replan Terminated Step, the job controller should perform the following steps:

1. If the identified task is invalid or is not in the Waiting for Planner task-state, ignore the message. No further action is necessary.
2. If the specified task is already in a terminal task-state, issue a Task Status message to the supervisor and a Plan Finished message to the planner with the appropriate plan-state. No further action is necessary.
3. If the job controller is at the equipment level, it should:
   (a) Pause the task (see Section 6.5.4 on page 57).
   (b) Abort the task according to internal procedures.
   (c) Mark all Executing or Not-Yet-Executing steps in the production plan as Scheduled.
   (d) Issue a Plan Finished request to the planner.
   (e) After receiving a response from the planner, issue a Task Status message to the supervisor specifying that the task has been Terminated.
4. If the job controller has subordinates, the job controller must consider each subtask individually:
   (a) For each subtask which is Deferred, Completed or already Terminated, no action is necessary.
   (b) If any subtask of the task is in the Aborted task-state or Aborting task-management-state, contact the active guardian with Subtask Error Occurred.
   (c) For each subtask which is in the Active task-state, issue Pause Task requests (with parameter as-soon-as-possible) to the subordinate executing that subtask. See Section 6.5.4 on page 57 for the details of the Pause Task message. If all active subtasks cannot be paused, then contact the active guardian with Subtask Error Occurred.
   (d) After all subordinates who were attempting to pause tasks have issued a Task Status message specifying that their tasks are Paused, issue a Terminate Task request for all subtasks which are in the Paused state. When all subtasks have been Terminated, the task is Terminated.
   (e) After the task has been terminated, mark all Executing and Not-Yet-Executing steps in the production plan as Scheduled.
   (f) Issue a Plan Finished request to the planner.
   (g) After receiving a response from the planner, issue a Task Status message to the supervisor specifying that the task has been terminated.

7.1.13 Planner Instructs Job Controller to Resume the Task

Upon receiving from the planner a response+ with action 'Resume' to one of Replan Aborted Step, Replan Deferred Step, Replan Late Plan, Replan Late Step or Replan Terminated Step, the job controller should perform the following steps:

1. If the identified task is invalid or not in the Waiting for Planner task-state, ignore the message. No further action is necessary.
2. Otherwise, retrieve an updated copy of the step which was specified in the original request to the planner and continue execution of the task (according to administrative state).

7.2 Planner Initiated Messages

There is only a single message which a planner may send to the job controller: Recheck Step. It is discussed below.

7.2.1 Recheck Step

The Recheck Step message is issued by a planner to inform the job controller that a step within an executing production plan has been modified and the job controller should verify that it has the modified information. The Recheck Step message should only be issued if the step which is modified is in the Not-Yet-Executing step-state. The Recheck Step message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

production-plan: Plan Identifier Object
step-identifier

8 Guardian Planning Interface

The MSI architecture provides the guardian planning interface to allow external monitoring and intervention of planning functions. The guardian planning interface supports some of the same messages as the planning interface. It provides additional messages for error reporting and recovery not provided by the planning interface.

There are two major differences between a guardian planning message and the planning message with the same name:

- In the covenant-related messages of the planning interface, each request has a subordinate-covenant-identifier parameter which identifies the covenant; its responses have a supervisor-covenant-identifier parameter which denotes the supervisor's identifier for that covenant. In the guardian planning interface, each request has a local-covenant-identifier parameter which is defined to be equivalent in value to the subordinate-covenant-identifier; its responses have the same local-covenant-identifier parameter as in the request.
- Some of the covenant related messages in the planning interface contain a Covenant Status Object parameter in the responses. In the guardian planning interface, this parameter is replaced with the Guardian Covenant Status Object. The default detail-level is used for the Guardian Covenant Status Object.

If the semantics of a message in the guardian planning interface are identical to the message in the planning interface with the same name, the reader will be referred to the planning interface specification for that message. In these cases, the following substitutions should be applied when reading those sections for the semantics of the guardian planning message:

- The response issued for the original request to the supervisor should be issued to the requesting guardian instead. All other supervisor correspondence should remain as specified.
- Any mention of the subordinate planner refers to the planner which was issued the message.

A planner may have at most one established active guardian planning connection at a time; it may have any number of established passive guardian planning connections. The set of messages which are valid for the passive guardian planning interface are:

- Administrative Status
- Clear Covenant Notification
- Connect
- Disconnect
- Identify
- Notify Covenant
- Report Administrative Status
- Report Covenants
- Set Covenant Notification

The set of messages which are valid for the active guardian planning interface are all valid passive guardian planning messages plus the following:

- Availability Alert
- Clear Schedule
- Configuration Alert
- Die
- Ignore Subordinate
- Reactivate Subordinate
- Remove All Bids
- Subcovenant Error Occurred
- Subordinate Problem
- Update Covenant Constraints

8.1 Data Objects

In addition to the data objects discussed in Section 4 on page 24, there are two complex data objects used in the guardian planning interface: the Guardian Administrative Status Object for reporting administrative status and the Guardian Covenant Status Object for reporting covenant status. Both data objects employ a similar mechanism to allow different levels of detail to be specified.

The Guardian Covenant Status Object has three component complex data objects: Covenant Name Object, Covenant Status Object, and Subcovenant Status Object. The following subsections describe all of these data objects in detail.

8.1.1 Guardian Administrative Status Object

The Guardian Administrative Status Object is a data object which contains information describing a planner's administrative status. Detail-levels 0 and 1 provide increasingly more detailed information about a planner's administrative status. The default detail-level is 0.

The Guardian Administrative Status Object contains the following data elements:

```
if (detail-level = 0) [
    administrative-state
    link-to-supervisor-status
    planner-job-control-link-status (CONDITIONAL)
]
if (detail-level = 1) [
    administrative-state
    link-to-supervisor-status
    planner-job-control-link-status (CONDITIONAL)
    list of [
        subordinate-planner-identifier
        administrative-state
        link-to-subordinate-status
    ] (CONDITIONAL)
]
```

If the current configuration specifies that a control entity exhibits the planning function only (no job control function is supported), then the parameter planner-job-control-link-status is omitted. If the planner does not have any subordinates, then the list of subordinate information should be omitted.

8.1.2 Covenant Name Object

The Covenant Name Object is a component of the Guardian Covenant Status Object; it contains high level information about a single covenant. Its primary use is to provide the minimal information necessary for a guardian to identify a covenant.

The Covenant Name Object contains the following data elements:

```
local-covenant-identifier
supervisor-covenant-identifier
global-plan-reference
```

plan: CHOICE OF{
production-managed-plan: Plan Identifier Object
OR work-element-identifier}

8.1.3 Covenant Status Object

The Covenant Status Object is a data object which is used in both the planning interface and the guardian planning interface to describe detailed information about a single covenant. The reader is referred to Section 5.4 on page 30 for a description of the Covenant Status Object. Within the guardian planning interface, the Covenant Status Object is a component of the Guardian Covenant Status Object.

8.1.4 Subcovenant Status Object

The Subcovenant Status Object is a component of the Guardian Covenant Status Object; it contains detailed information about a subcovenant for a step in a production plan.

The Subcovenant Status Object contains the following data elements:

local-subcovenant-identifier
step-identifier
subcovenant-state
subordinate-planner-identifier (CONDITIONAL)
subordinate-subcovenant-identifier (CONDITIONAL)

If a subcovenant is being scheduled by the same planner responsible for scheduling the covenant to which it belongs, then the parameters subordinate-planner-identifier and subordinate-subcovenant-identifier are omitted.

8.1.5 Guardian Covenant Status Object

The Guardian Covenant Status Object is a data object which specifies information about a single covenant. Detail-levels 0, 1, 2 and 3 provide increasingly more detailed information about a covenant. If the detail-level is 2, the list of subcovenant (Subcovenant Status Object), contains information about only those subcovenants that are currently in the process of being scheduled. If the detail-level is 3, the list of subcovenant (Subcovenant Status Object), contains information about ***all*** subcovenants of a covenant. The default detail-level is 1.

The Guardian Covenant Status Object contains the following data elements:

if (detail-level = 0) [
Covenant Name Object
]
if (detail-level = 1) [
Covenant Name Object
Covenant Status Object
list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)
]
if (detail-level = 2) [
Covenant Name Object
Covenant Status Object
list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)
list of subcovenant :Subcovenant Status Object
]

```
if (detail-level = 3) [
    Covenant Name Object
    Covenant Status Object
    list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)
    list of subcovenant: Subcovenant Status Object
]
```

The parameter list of input-plan-parameter is only specified if the source-plan used to schedule the covenant requires input plan parameters; otherwise it is omitted.

8.2 Guardian Initiated Messages

The guardian initiated messages in the guardian planning interface are:

- Availability Alert
- Clear Covenant Notification
- Clear Schedule
- Configuration Alert
- Connect
- Die
- Disconnect
- Identify
- Ignore Subordinate
- Reactivate Subordinate
- Remove All Bids
- Report Administrative Status
- Report Covenants
- Set Covenant Notification
- Update Covenant Constraints

This section describes each of these messages in more detail.

8.2.1 Availability Alert

The Availability Alert message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner if the resource for which that planner is scheduling has had a change in availability. The Availability Alert message is an unconfirmed message; it has no response. There are no parameters to the Availability Alert message.

A planner needs to be constantly informed of changes in its resource's availability so that it may accurately schedule plans for the job controller to execute; it must be aware of when that resource is expected to be down for maintenance and if that resource is currently unavailable due to a failure. If the availability information changes, an Availability Alert message should be issued to the planner.

It is a protocol violation for a planner to receive an Availability Alert while in the Available administrative state; the message should be ignored. Otherwise, upon receiving an Availability Alert message, a planner should update its local information about the resource's availability. If the new availability information conflicts with previously scheduled covenants, the planner should issue a Covenants Broken message to the supervisor for the affected covenants (see Section 5.7.3 on page 41 for the details of this message).

8.2.2 Clear Covenant Notification

The Clear Covenant Notification message is valid for both the active and passive guardian planning interfaces. It is issued by a guardian to a planner; it requests the planner to discontinue dynamically reporting covenant status to the issuing guardian for a set of covenants. The parameters for both the Clear Covenant Notification request and its response are as follows:

Parameters on Request:

detail-level
covenants: CHOICE OF{
list of local-covenant-identifier
OR All-Bids
OR All-Contracts
OR All-Covenants}

Parameters on Response:

list of [
local-covenant-identifier
acceptance: Acceptance Object
] (CONDITIONAL)

If a planner receives a Clear Covenant Notification request which specifies All-Bids, All-Contracts or All-Covenants, the Clear Covenant Notification response will have no parameters.

If a guardian wishes to discontinue dynamic status reporting for a set of covenants, it should issue a Clear Covenant Notification request to the planner with those covenants.

It is a protocol violation to receive a Clear Covenant Notification request while in the Available administrative state. The planner should ignore the request and issue an Administrative Status message.

If the planner is in the Active administrative state and the guardian specifies all bids, all contracts or all covenants in the Clear Covenant Notification request, then the planner should clear notification for all outstanding bids, contracts or covenants (respectively) which are marked to be reported to the requesting guardian, and issue a Clear Covenant Notification response with no parameters.

If the planner is in the Active administrative state and the guardian specifies a list of covenants on which to clear notification, then the planner should determine the validity of each covenant specified by the guardian, and if valid, clear notification for that covenant. The planner should issue a Clear Covenant Notification response, enumerating each specified covenant and denoting whether notification was cleared or not.

8.2.3 Clear Schedule

The Clear Schedule message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner; it requests the planner to dispose of all covenants which overlap with a specified time interval. The parameters for both the Clear Schedule request and its responses are as follows:

Parameters on Request:

start-time
completion-time

Parameters on Response:

acceptance: Acceptance Object

If the active guardian wishes to clear the schedule (remove all contracted and bidden operations) of a given planner within a given time interval, it should issue a Clear Schedule request to that

planner. Upon receiving a Clear Schedule request from the active guardian, the planner should perform the following steps:

1. It is a protocol violation to receive a Clear Schedule request while in the Available administrative state. The planner should ignore the request and issue an Administrative Status message to the active guardian.
2. If the specified time interval is completely beyond the planner's planning horizon, issue a Clear Schedule response–. No further action is necessary.
3. If the specified time interval is in the past, issue a Clear Schedule response–. No further action is necessary.
4. If the specified start-time is later than the specified completion-time, issue a Clear Schedule response–. No further action is necessary.
5. For each covenant which partially or completely overlaps with the specified time interval, do the following:
   (a) Switch the covenant to the Removing Covenant covenant-state.
   (b) Dispose of the covenant and its associated production plan. If the planner has subordinates, this may involve issuing Remove Covenant requests to subordinates.
6. Issue a Covenants Broken message to the supervisor specifying the covenants which have been broken.
7. Issue a Clear Schedule response+.

8.2.4 Configuration Alert

The Configuration Alert is valid for the active guardian planning interface only. It is issued by the active guardian to inform a planner that the baseline configuration[9] has changed. The Configuration Alert message is an unconfirmed message; it has no response. There are no parameters to the Configuration Alert message.

It is a protocol violation to receive a Configuration Alert while in the Available administrative state; the message should be ignored. If a planner receives a Configuration Alert from the active guardian while in the Active administrative state, it should consult the current baseline configuration to determine if the supervisor and/or subordinates have changed or if its relationship to one or more job controllers has changed. It should deactivate each planner which it is currently supervising but not a subordinate in the modified configuration. It should activate each of its subordinates in the modified configuration if it is not currently supervising that planner.

8.2.5 Connect

The Connect message is valid for both the active and passive guardian planning interfaces. It is issued by a guardian to a planner to establish an active or passive guardian planning connection via the guardian planning interface. This is the first message a guardian may issue to a planner; the connection is required to be established prior to issuing any other messages. The parameters for both the Connect request and its responses are as follows:

Parameters on Request:

<none>

9. A baseline configuration is the original specification of where a control entity exists within the control hierarchy, who the supervisor is, who its subordinates are, and whether the job control function exists.

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner who receives a Connect request should determine if it is capable of accepting the connection request, and if so, issue a Connect response+. The Connect response+ results in the connection being established. In general, the primary reason for rejecting a connection request from a guardian is that the planner currently has an established active guardian planning connection and another request for the active guardian planning connection is received.

8.2.6 Die

The Die message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner; it requests the planner to immediately exit regardless of administrative state or the existence of outstanding bids. The Die message is an unconfirmed message; it has no response. There are no parameters to the Die message.

An active guardian may issue a Die message to a planner which is exhibiting anomalous behavior or in other situations where such extreme actions are necessary. When a planner receives a Die message, the only action it is ***required*** to perform is to exit; it may perform additional actions. It should be noted that for the shop and workcell planners, this will result in planners without supervisors.

8.2.7 Disconnect

The Disconnect message is valid for both the active and passive guardian planning interfaces. It is issued by a guardian to a planner to terminate a connection which is currently established via the guardian planning interface. This is the last message a guardian may issue to a planner without re-establishing a connection. The parameters for both the Disconnect request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A planner which receives a Disconnect request should determine if it should terminate the connection, and if so, issue a Disconnect response+. The Disconnect response+ results in the connection being terminated. In general, the primary reason for rejecting a Disconnect request is that the planner has outstanding requests to that guardian.

8.2.8 Identify

The Identify message is valid for both the active and passive guardian planning interfaces. It is issued by a guardian to a planner to determine its characteristics. The parameters for both the Identify request and its response are as follows:

Parameters on Request:

<none>

Parameters on Response:

make
model
version
level
list of planning-strategy

If a guardian wishes to know a particular planner's characteristics, it should issue that planner an Identify request. The steps that a planner should follow upon receiving an Identify request from a guardian are identical to those of receiving an Identify request from the supervisor. See Section 5.5.3 on page 33 for an explanation of these steps.

8.2.9 Ignore Subordinate

The Ignore Subordinate message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner; it requests the planner to indefinitely suspend both communicating with and supervising one of its subordinates. The parameters for both the Ignore Subordinate request and its responses are as follows:

Parameters on Request:

subordinate-planner-identifier

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

This message is used when a planner is exhibiting anomalous behavior which is interfering with the supervisor's performance; the active guardian may decide to issue an Ignore Subordinate request instructing the supervising planner to ignore that subordinate.

Upon receiving an Ignore Subordinate request from the active guardian, the planner should perform the following steps:

1. The planner has not checked the baseline configuration to determine its subordinates when in the Available administrative state. If the planner is in the Available administrative state, issue an Ignore Subordinate response–. No further action is necessary.
2. If the specified subordinate does not exist, issue an Ignore Subordinate response–. No further action is necessary.
3. Terminate the connection (if one exists) with the specified subordinate.
4. Issue an Ignore Subordinate response+.
5. Do not attempt to establish a new connection.
6. Continue to report in Guardian Administrative Status messages that the specified subordinate exists and that its link status is unconnected. This is done so that the guardian will remain aware that this subordinate is being ignored.

8.2.10 Reactivate Subordinate

The Reactivate Subordinate message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner; it requests the planner to resume communicating with and supervising a subordinate that it was previously instructed to ignore. The parameters for both the Reactivate Subordinate request and its responses are as follows:

Parameters on Request:

subordinate-planner-identifier

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

Upon receiving a Reactivate Subordinate request from the active guardian, the planner should perform the following steps:

1. A planner has not checked the baseline configuration to determine its subordinates when in the Available administrative state. If the planner is in the Available administrative state, issue a Reactivate Subordinate response–. No further action is necessary.
2. If the specified subordinate does not exist, issue a Reactivate Subordinate response–. No further action is necessary.
3. Otherwise, issue a Reactivate Subordinate response+.
4. Attempt to establish a connection with the specified subordinate. If the planner cannot establish a connection with the specified subordinate, it should issue an Administrative Status message (see Section 8.3.1 on page 85).
5. If a connection can be established, the planner should resume its supervisory responsibilities with respect to that subordinate.

8.2.11 Remove All Bids

The Remove All Bids message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner; it requests the planner to dispose of all covenants which are in the Pre-Bidding, Bidding, Bidden, Breaking Bid or Accepting covenant-state so that the planner may be deactivated. The parameters for both the Remove All Bids request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

acceptance: Acceptance Object

If the active guardian wishes to deactivate a planner, it should first issue that planner a Remove All Bids request. Upon receiving a Remove All Bids request from the active guardian, the planner should perform the following steps:

1. It is a protocol violation to receive a Remove All Bids request while in the Available administrative state. The planner should ignore the request and issue an Administrative Status message to the requesting guardian.
2. For each covenant in the Pre-Bidding covenant-state, issue a Request for Bid response–. Dispose of the covenant and its associated production plan.
3. For each covenant in the Accepting covenant-state, issue an Accept Bid response– Switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and its associated production plan. If the planner has subordinates, this may involve issuing Remove Covenant requests to subordinates.

4. For each covenant in the Bidding, Bidden, or Breaking Bid covenant-state, switch the covenant to the Removing Covenant covenant-state. Dispose of the covenant and its associated production plan. If the planner has subordinates, this may involve issuing Remove Covenant requests to subordinates.
5. Issue a Remove All Bids response+.

8.2.12 Report Administrative Status

The Report Administrative Status message is valid for both the active and passive guardian planning interface. It is issued by a guardian to a planner to request its current administrative status. The parameters for both the Report Administrative Status request and its responses are as follows:

Parameters on Request:

detail-level

Parameters on Response+:

Guardian Administrative Status Object

Parameters on Response–:

error: Error Object

If a guardian wishes to receive updated administrative status from a planner, it should issue a Report Administrative Status request to that planner. The detail-level parameter allows a guardian to request administrative information about only the planner or the planner's subsystem. See Section 8.1.1 on page 74 for a discussion of the Guardian Administrative Status Object which explains the information provided for each detail-level.

Upon receiving a Report Administrative Status request from a guardian, a planner should issue a Report Administrative Status response– if the detail-level is invalid. Otherwise, it should issue a Report Administrative Status response+ specifying the required information as defined by the Guardian Administrative Status Object and detail-level.

8.2.13 Report Covenants

The Report Covenants message is valid for both the active and passive guardian planning interface. It is issued by a guardian to a planner; it requests status information to be reported on a set of covenants. The parameters for both the Report Covenants request and its response are as follows:

Parameters on Request:

```
covenants:    CHOICE OF{
                  list of local-covenant-identifier
              OR All-Bids
              OR All-Contracts
              OR All-Covenants}
detail-level
```

Parameters on Response:

```
list of [
    local-covenant-identifier
    status:   CHOICE {
                  error: Error Object
              OR status: Guardian Covenant Status Object}
] (CONDITIONAL)
```

If a Report Covenants request specified All-Bids, All-Contracts or All-Covenants and a planner currently has no outstanding bids, contracts or covenants (respectively), then the Report Covenants response will have no parameters.

If a guardian wishes to receive updated status for some collection of covenants, it should issue a Report Covenants request to the planner with those covenants. The detail-level parameter allows a guardian to request more or less detailed information about the covenants. See Section 8.1.5 on page 75 for a discussion of the Guardian Covenant Status Object which explains the information provided for each detail-level. Upon receiving a Report Covenants request from the guardian, the planner should perform the following steps:

1. It is a protocol violation to receive a Report Covenants request while in the Available administrative state. The planner should ignore the request and issue an Administrative Status message to the requesting guardian.
2. If the Report Covenants request was issued with the parameter All-Bids, then for each covenant in the Bidding, Bidden, Breaking Bid or Accepting covenant-state, compile the Guardian Covenant Status Object and issue a Report Covenants response.
3. If the Report Covenants request was issued with the parameter All-Contracts, then for each covenant in the Contracted, Breaking Contract or Executing covenant-state, compile a Guardian Covenant Status Object and issue a Report Covenants response.
4. If the Report Covenants request was issued with the parameter All-Covenants, then for each covenant in the Bidding, Bidden, Breaking Bid, Accepting, Contracted, Breaking Contract, Removing Covenant or Executing covenant-state, compile a Guardian Covenant Status Object and issue a Report Covenants response.
5. If the Report Covenants request was issued with a list of covenants, then for each covenant that is in the list, but not in the Executed covenant-state, compile a Guardian Covenant Status Object. For all other covenants in the list, identify the appropriate error and issue a Report Covenants response which includes the error information for invalid covenants and the covenant status information for valid covenants.

8.2.14 Set Covenant Notification

The Set Covenant Notification message is valid for both the active and passive guardian planning interfaces. It is issued by a guardian to a planner; it requests the planner to dynamically report covenant status to the issuing guardian for a set of covenants. The parameters for both the Set Covenant Notification request and its response are as follows:

Parameters on Request:

covenants: CHOICE OF{
list of local-covenant-identifier
OR All-Bids
OR All-Contracts
OR All-Covenants}
detail-level

Parameters on Response:

list of [
local-covenant-identifier
acceptance: Acceptance Object
] (CONDITIONAL)

If a planner receives a Set Covenant Notification request which specifies All-Bids, All-Contracts or All-Covenants, then the Set Covenant Notification response will have no parameters.

If a guardian wishes to receive dynamic covenant status reports for a set of covenants, it should issue a Set Covenant Notification request to the planner with those covenants. This message differs from the Report Covenants message in that the Report Covenants message provides a snapshot view of the status for a set of covenants; the Set Covenant Notification message provides a continuous view of the status for a set of covenants — the planner should issue status messages any time the current status of one of the specified covenants changes.

The detail-level parameter allows a guardian to request more or less detailed information about the covenants. See Section 8.1.5 on page 75 for a discussion of the Guardian Covenant Status Object which explains the information provided for each detail-level.

It is a protocol violation to receive a Set Covenant Notification while in the Available administrative state. The planner should ignore the request and issue an Administrative Status message to the requesting guardian.

If the planner is in the Active administrative state and all bids, all contracts or all covenants is specified in the request, then the planner should set notification for all outstanding bids, contracts or covenants, respectively, and issue a Set Covenant Notification response with no parameters.

If the planner is in the Active administrative state and a list of covenants is specified in the request, then the planner should determine the validity of each covenant specified by the guardian, and if valid, set notification for that covenant. The planner should issue a Set Covenant Notification response, enumerating each specified covenant and denoting whether notification was set or not.

8.2.15 Update Covenant Constraints

The Update Covenant Constraints message is valid for the active guardian planning interface only. It is issued by the active guardian to a planner to assist the planner in planning by changing the constraints associated with a covenant. The parameters for both the Update Covenant Constraints request and its responses are as follows:

Parameters on Request:

local-covenant-identifier
planning-strategy
priority
earliest-start-time (CONDITIONAL)
latest-completion-time (CONDITIONAL)

Parameters on Response+:

local-covenant-identifier

Parameters on Response–:

local-covenant-identifier
error: Error Object
Guardian Covenant Status Object (CONDITIONAL)

If one of the conditional parameters is the Update Covenant Constraints request is omitted, then that constraint is removed. These semantics are consistent with that of the Request for Bid: existent constraints should be specified, constraints to be removed should be omitted from the

request. If a planner issues an Update Covenant Constraints response–, the conditional parameter Guardian Covenant Status Object is specified if it contains information related to rejecting the request.

An Update Covenant Constraints message may be used to change any combination of the following parameters of a covenant: planning-strategy, priority, earliest-start-time or latest-completion-time. Only the parameters which are to be changed should be specified in the request. This information may be used by the planner to reschedule a covenant. If the active guardian wishes to intervene with the scheduling of an operation, it may issue an Update Covenant Constraints request to a planner. The steps that a planner should follow upon receiving an Update Covenant Constraints request from the active guardian are identical to those of receiving an Update Covenant Constraints request from the supervisor. See Section 5.6.5 on page 38 for an explanation of these steps.

8.3 Planner Initiated Messages

The planner initiated messages in the guardian planning interface are:

- Administrative Status
- Notify Covenant
- Subcovenant Error Occurred
- Subordinate Problem

This section describes each of these messages in more detail.

8.3.1 Administrative Status

An Administrative Status message is valid for both the active and passive guardian planning interface. It is issued to each guardian by a planner to inform that guardian of any change in its administrative status. A planner should issue an Administrative Status message to each connected guardian every time one of the parameters in its administrative status changes. If the planner does not currently have an established guardian planning connection and it wishes to issue an Administrative Status message to report an abnormal situation, it should turn on the planning watchdog and after establishing a connection to a guardian, it should issue an Administrative Status message. The Administrative Status message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

administrative-state
link-to-supervisor-status
operational-status
list of [
    subordinate-planner-identifier
    administrative-state
    link-to-subordinate-status
] (CONDITIONAL)

If a planner issuing an Administrative Status message has no subordinates, then the list of subordinate information will not be specified.

8.3.2 Notify Covenant

The Notify Covenant messages is valid for both the active and passive guardian planning interfaces. A Notify Covenant message is issued by a planner to a guardian to report the current status of a covenant. The Notify Covenant message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

local-covenant-identifier
Guardian Covenant Status Object

When in the Active administrative state, a planner should, for each outstanding covenant, issue Notify Covenant messages to each guardian which has set notification for that covenant each time any parameter of the Guardian Covenant Status Object for that covenant changes. The detail-level used in determining how much information to provide in the Guardian Covenant Status Object is that which was specified by that guardian in its Set Covenant Notification request.

8.3.3 Subcovenant Error Occurred

The Subcovenant Error Occurred message is valid for the active guardian planning interface only. It is issued by a planner to inform the active guardian that an error has occurred in attempting to reschedule a production plan. When the planner wishes to issue this message to the active guardian, the planner should turn on its watchdog red-light. The parameters for both the Subcovenant Error Occurred request and its response are as follows:

Parameters on Request:

local-covenant-identifier
step-identifier
global-plan-reference
error: Error Object

Parameters on Response:

local-covenant-identifier
action ∈ {Abort Covenant, Defer Covenant, Retry, Terminate Covenant}

Most subcovenant scheduling errors arise from either competing priority covenants or when the production plan associated with the covenant is already executing and therefore the planner is restricted regarding what it can do to resolve a conflict. The active guardian may respond with one of four options: Abort, Defer, Retry or Terminate. Abort, Defer and Terminate cause the planner to instruct the job controller to Abort, Defer or Terminate (respectively) the corresponding subtask. Retry causes the planner to re-attempt to resolve the conflict; in general, the guardian has changed the parameters — either at this level of control or some higher level of control — associated with the covenants so that the conflict will no longer exist.

8.3.4 Subordinate Problem

The Subordinate Problem message is valid for the active guardian planning interface only. It is issued by a planner to inform the active guardian that a subordinate is exhibiting anomalous behavior. When the planner wishes to issue this message to the active guardian, the planner should turn on its watchdog red-light. The Subordinate Problem message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

subordinate-planner-identifier
error: Error Object

Examples of when a Subordinate Problem message would be issued include when a subordinate is not responding to requests, when a subordinate is consistently issuing inappropriate messages, or when a planner is unable to establish a connection to its subordinate.

9 Guardian Job Control Interface

The MSI architecture provides the guardian job control interface to allow for external monitoring and intervention of job control functions. The guardian job control interface supports most of the same messages as the job control interface. It provides additional messages for error reporting and recovery not provided by the job control interface.

There are two major differences between a guardian job control message and a job control message with the same name:

- In the task-related messages of the job control interface, each request has a subordinate-task-identifier parameter which identifies the task, and its responses have a supervisor-task-identifier parameter which denotes the supervisor's identifier for that task. In the guardian job control interface, each request has a local-task-identifier parameter which is defined to be equivalent in value to the subordinate-task-identifier, and its responses have the same local-task-identifier parameter as in the request.
- Several of the task related messages in the job control interface contain a Task Status Object parameter in the responses. In the guardian job control interface, this parameter is replaced with the Guardian Task Status Object. The default detail-level is used for the Guardian Task Status Object.

If the semantics of a message in the guardian job control interface are identical to the message in the job control interface with the same name, the reader will be referred to the job control interface specification for that message. In these cases, the following substitutions should be applied when reading those sections for the semantics of the guardian job control message:

- Any response issued to the supervisor should be issued to the requesting guardian instead.
- Any mention of the subordinate job controller refers to the job controller which was issued the message.

A job controller may have at most one established active guardian job control connection at a time; it may have any number of established passive guardian job control connections. The set of messages which are valid for the passive guardian job control interface are:

- Administrative Status
- Clear Task Notification
- Connect
- Disconnect
- Identify
- Notify Task
- Report Administrative Status
- Report Tasks
- Set Task Notification

The set of messages which are valid for the active guardian job control interface are all valid passive guardian job control messages plus the following:

- Abort Task
- Call Task Complete
- Change Subtask Status
- Configuration Alert
- Defer Task
- Die
- Emergency Stop
- Pause Task
- Reactivate Subordinate
- Resume All Tasks
- Resume Task
- Subordinate Problem
- Subtask Error Occurred
- Task Error Occurred

- Ignore Subordinate
- Pause All Tasks
- Terminate All Tasks
- Terminate Task

9.1 Data Objects

In addition to the data objects discussed in Section 4 on page 24, there are two complex data objects used in the guardian job control interface: the Guardian Administrative Status Object for reporting administrative status and the Guardian Task Status Object for reporting task status. Both data objects employ a similar mechanism to allow different levels of detail to be specified.

The Guardian Task Status Object has three component data objects: Task Name Object, Task Status Object, and Task Detail Status Object. The following subsections describe all of these data objects in detail.

9.1.1 Guardian Administrative Status Object

The Guardian Administrative Status Object is a data object which contains information describing a job controller's administrative status. Detail-levels 0 and 1 provide increasingly more detailed information about a job controller's administrative status. The default detail-level is 0.

The Guardian Administrative Status Object contains the following data elements:

```
if (detail-level = 0) [
    administrative-state
    link-to-supervisor-status
    planner-job-control-link-status
]
if (detail-level = 1) [
    administrative-state
    link-to-supervisor-status
    planner-job-control-link-status
    list of [
        subordinate-job-controller-identifier
        administrative-state
        link-to-subordinate-status
    ] (CONDITIONAL)
]
```

If a job controller has no subordinates, then the list of subordinate information should be omitted.

9.1.2 Task Name Object

The Task Name Object is a component of the Guardian Task Status Object which contains high-level information about a single task. Its primary use is to provide the minimal information necessary for a guardian to identify a task.

The Task Name Object contains the following data elements:

```
local-task-identifier
supervisor-task-identifier
production-plan: Plan Identifier Object
global-plan-reference
```

9.1.3 Task Status Object

The Task Status Object is a data object which is used in both the job control interface and the guardian job control interface to describe detailed information about a single task. The reader is referred to Section 6.3 on page 44 for a description of the Task Status Object. Within the guardian job control interface, the Task Status Object is a component of the Guardian Task Status Object.

9.1.4 Subtask Status Object

The Subtask Status Object is a component of the Guardian Task Status Object which contains detailed information about a subtask belonging to a single task at the addressed level of control.

The Subtask Status Object contains the following data elements:

local-subtask-identifier
step-identifier
subtask-state
subtask-management-state
subordinate-job-controller-identifier (CONDITIONAL)
subordinate-subtask-identifier (CONDITIONAL)

If a subtask is being executed by the same job controller responsible for executing the task to which it belongs, then the parameters subordinate-job-controller-identifier and subordinate-subtask-identifier are omitted.

9.1.5 Guardian Task Status Object

The Guardian Task Status Object is a data object which specifies information about a single task. Depending on what the detail-level is, the Guardian Task Status Object will provide more or less detailed information. Detail-levels 0, 1 and 2 provide increasingly more detailed information about a task. The default detail-level is 1.

The Guardian Task Status Object contains the following data elements:

```
if (detail-level = 0) [
    Task Name Object
]
if (detail-level = 1) [
    Task Name Object
    Task Status Object
    list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)
]
if (detail-level = 2) [
    Task Name Object
    Task Status Object
    list of input-plan-parameter: Plan Parameter Object (CONDITIONAL)
    list of subtask: Subtask Status Object
]
```

The parameter list of input-plan-parameter is specified if the production plan requires input plan parameters.

9.2 Guardian Initiated Messages

The guardian initiated messages in the guardian job control interface are:

- Abort Task
- Call Task Completed
- Change Subtask Status
- Clear Task Notification
- Configuration Alert
- Connect
- Defer Task
- Die
- Disconnect
- Emergency Stop
- Identify
- Ignore Subordinate
- Pause All Tasks
- Pause Task
- Reactivate Subordinate
- Report Administrative Status
- Report Tasks
- Resume All Tasks
- Resume Task
- Set Task Notification
- Terminate All Tasks
- Terminate Task

This section describes each of these messages in detail.

9.2.1 Abort Task

The Abort Task message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a specified task to be aborted. The parameters for both the Abort Task request and its responses are as follows:

Parameters on Request:

local-task-identifier

Parameters on Response+:

local-task-identifier

Parameters on Response–:

local-task-identifier
error: Error Object
Guardian Task Status Object (CONDITIONAL)

If a job controller issues an Abort Task response–, the conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

If the active guardian wishes to abort a task, it should issue an Abort Task request to the job controller currently executing the task. The steps that a job controller should follow upon receiving an Abort Task request from the active guardian are identical to those of receiving an Abort Task request from the supervisor. See Section 6.5.1 on page 53 for an explanation of these steps.

9.2.2 Call Task Completed

The Call Task Completed message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a job controller to abort a specific task, but to report that it has successfully completed. The parameters for both the Call Task Completed request and its responses are as follows:

Parameters on Request:

local-task-identifier

Parameters on Response+:

local-task-identifier

Parameters on Response–:

local-task-identifier
error: Error Object
Guardian Task Status Object (CONDITIONAL)

If the job controller issues a Call Task Completed response–, the conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

This message is used to address an error condition when a guardian operator intervenes in the execution of a task which results in the task's completion. In such a situation, the active guardian should issue a Call Task Completed request to the job controller executing the task so that the job controller will treat the task as completed.[10]

Upon receiving a Call Task Completed request from the active guardian, the job controller should perform the following steps:

1. If the job controller is in the Available or E-stopped administrative state, the request should be ignored and an Administrative Status message should be issued to the requesting guardian. No further action is necessary.
2. If the specified task is an invalid task, issue a Call Task Completed response–. No further action is necessary.
3. If the specified task is already in the Aborted, Deferred or Terminated task-state, issue a Call Task Completed response–. No further action is necessary.
4. If the specified task is already in the Completed task-state, issue a Call Task Completed response+. No further action is necessary.
5. Abort any locally executing subtasks according to internal procedures.
6. If the job controller has subordinates, issue Abort Task requests to all subordinates executing subtasks in support of the specified task.
7. After all subtasks have been aborted (both local and non-local), the task is aborted.
8. Issue a Plan Finished request to the planner specifying that the production plan has reached the Completed plan-state.
9. Issue a Task Status message to the supervisor specifying that the task has completed (i.e., the task-state is Completed).
10. Issue a Call Task Completed response+.

9.2.3 Change Subtask Status

The Change Subtask Status message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller. It requests the job controller to update the status information associated with an outstanding subtask which is being executed by a subordinate (or

10. For example, a robot encounters an error which results in it being unable to complete a move-part operation. The guardian operator intervenes by moving the part to the correct destination location. The operator would then issues a Call Task Completed request to the robot job controller,

possibly executing locally). The parameters for both the Change Subtask Status request and its responses are as follows:

Parameters on Request:

local-task-identifier
local-subtask-identifier
Task Status Object

Parameters on Response+:

local-task-identifier
local-subtask-identifier

Parameters on Response–:

local-task-identifier
local-subtask-identifier
error: Error Object

Upon receiving a Change Subtask Status request from the active guardian, the job controller should perform the following steps:

1. If the job controller is in the Available or E-stopped administrative state, the request should be ignored and an Administrative Status message should be issued to the requesting guardian. No further action is necessary.
2. If the specified task is an invalid task or the specified subtask is an invalid subtask, issue a Change Subtask Status response–. No further action is necessary.
3. If the specified task is already in the Aborted, Completed, Deferred or Terminated task-state, issue a Change Subtask Status response–. No further action is necessary.
4. Update the internal information about the specified subtask to reflect the information specified in the Task Status Object.
5. Proceed executing the task according to the status information provided.
6. Issue a Change Subtask Status response+.

9.2.4 Clear Task Notification

The Clear Task Notification message is valid for both the active and passive guardian job control interfaces. It is issued by a guardian to a job controller; it requests the dynamic reporting of status information to cease for a set of tasks. The parameters for both the Clear Task Notification request and its response are as follows:

Parameters on Request:

tasks: CHOICE OF {
list of local-task-identifier (CONDITIONAL)
OR All-Tasks

Parameters on Response:

list of [
local-task-identifier
acceptance: Acceptance Object
] (CONDITIONAL)

If a guardian issues a Clear Task Notification request and wishes to clear notification for all tasks with notification currently set, the request will have no parameters. If no list of tasks is specified in a Clear Task Notification request, the Clear Task Notification response will have no parameters.

If a guardian wishes to no longer receive dynamically updated status information for a set of tasks, it should issue a Clear Task Notification request to the job controller currently executing those tasks.

It is a protocol violation to receive a Clear Task Notification request while in the Available or E-stopped administrative state. The job controller should ignore the request and issue an Administrative Status message to the requesting guardian.

For all other administrative states, if the guardian does not specify a list of tasks in the Clear Task Notification request, then the job controller should clear notification for all outstanding tasks and the Clear Task Notification response will have no parameters. If the guardian specifies a list of tasks on which to clear notification, then the job controller should determine the validity of each task specified by the guardian, and if valid, clear notification for that task. The job controller should issue a Clear Task Notification response, enumerating each specified task and denoting whether notification was cleared or not.

9.2.5 Configuration Alert

The Configuration Alert is valid for the active guardian job control interface only. It is issued by the active guardian to inform a job controller that the baseline configuration[11] has changed. The Configuration Alert message is an unconfirmed message; it has no response. There are no parameters to the Configuration Alert message.

It is a protocol violation to receive a Configuration Alert while in the Available or E-stopped administrative state; the message should be ignored. Otherwise, the job controller should consult the current baseline configuration to determine if the supervisor and/or subordinates have changed or if its relationship to the planner has changed. It should deactivate each job controller which it is currently supervising but which is not a subordinate in the modified configuration. It should activate each of its subordinates in the modified configuration if it is not currently supervising that job controller. If the planner has changed, it should disconnect from its current planner and connect to the new planner.

9.2.6 Connect

The Connect message is valid for both the active and passive guardian job control interfaces. It is issued by a guardian to a job controller to establish a connection via the guardian job control interface. This is the first message a guardian may issue to a job controller; a connection must be established before any other messages are issued. The parameters for both the Connect request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

11. A baseline configuration is the original specification of where a control entity exists within the control hierarchy, who the supervisor is, who its subordinates are, and whether the job control function exists.

A job controller who receives a Connect request should determine if it is capable of accepting the connection request, and if so, issue a Connect response+. The Connect response+ results in the connection being established. In general, the primary reason for rejecting a connection request from a guardian is that the job controller currently has the active guardian job control connection and another request for the active guardian job control connection is received.

9.2.7 Defer Task

The Defer Task message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a specified task to be deferred. The parameters for both the Defer Task request and its responses are as follows:

Parameters on Request:

local-task-identifier

Parameters on Response+:

local-task-identifier

Parameters on Response–:

local-task-identifier
error: Error Object
Guardian Task Status Object (CONDITIONAL)

If a job controller issues a Defer Task response–, the conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

If the active guardian wishes to defer a task, it should issue a Defer Task request to the job controller currently executing the task. The steps that a job controller should follow upon receiving a Defer Task request from the active guardian are identical to those of receiving a Defer Task from the supervisor. See Section 6.5.2 on page 54 for an explanation of these steps.

9.2.8 Die

The Die message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to exit regardless of administrative state or the existence of outstanding tasks. The Die message is an unconfirmed message; it has no response. There are no parameters to the Die message.

An active guardian may issue a Die message to a job controller which is exhibiting anomalous behavior or in other situations where such extreme actions are necessary. When a job controller receives a Die message, the only action it is ***required*** to perform is to exit; it may perform other actions. It should be noted that for higher level job controllers, this will result in job controllers without supervisors.

9.2.9 Disconnect

The Disconnect message is valid for both the active and passive guardian job control interfaces. It is issued by a guardian to a job controller to terminate a connection which is currently established via the guardian job control interface. This is the last message a guardian may issue to a job controller without re-establishing a connection. The parameters for both the Disconnect request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

A job controller which receives a Disconnect request should determine if it should terminate the connection, and if so, issue a Disconnect response+. The Disconnect response+ results in the connection being terminated. In general, the primary reason for rejecting a Disconnect request is that the job controller has outstanding requests to that guardian.

9.2.10 Emergency Stop

The Emergency Stop message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to immediately stop all physical activity. The Emergency Stop message is an unconfirmed message; it has no response. There are no parameters to the Emergency Stop message.

The steps that a job controller should follow upon receiving an Emergency Stop message from the active guardian are identical to those of receiving an Emergency Stop message from the supervisor. See Table 11. on page 48 for an explanation of these steps.

9.2.11 Identify

The Identify message is valid for both the active and passive guardian job control interfaces. It is issued to determine the characteristics of a job controller. The parameters for both the Identify request and its response are as follows:

Parameters on Request:

<none>

Parameters on Response:

make
model
version
level

If a guardian wishes to know the characteristics of a job controller, it should issue the job controller an Identify request. The steps that a job controller should follow upon receiving an Identify request from a guardian are identical to those of receiving an Identify request from the supervisor. See Section 6.4.5 on page 48 for an explanation of these steps.

9.2.12 Ignore Subordinate

The Ignore Subordinate message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to indefinitely suspend both communicating with and supervising one of its subordinates. The parameters for both the Ignore Subordinate request and its responses are as follows:

Parameters on Request:

subordinate-job-controller-identifier

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

This message is used when a job controller is exhibiting anomalous behavior which is interfering with the supervisor's performance; the active guardian may decide to issue an Ignore Subordinate request instructing the supervising job controller to ignore that subordinate.

Upon receiving an Ignore Subordinate request from the active guardian, the job controller should perform the following steps:

1. The job controller has not checked the baseline configuration to determine its subordinates when in the Available administrative state. If the planner is in the administrative state, issue an Ignore Subordinate response–. No further action is necessary.
2. If the specified subordinate does not exist, issue an Ignore Subordinate response–. No further action is necessary.
3. Terminate the connection (if one exists) with the specified subordinate.
4. Issue an Ignore Subordinate response+.
5. Do not attempt to establish a new connection.
6. Continue to report, in Guardian Administrative Status messages, that the specified subordinate exists and that its link status is unconnected. This is done so that the guardian will remain aware that this subordinate is being ignored.

9.2.13 Pause All Tasks

The Pause All Tasks message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to switch to the Pausing (and eventually to the Paused) administrative state and to suspend the execution of all outstanding tasks. The parameters for both the Pause All Tasks request and its responses are as follows:

Parameters on Request:

time-frame: CHOICE OF {
desired-pausing-duration
OR as-soon-as-possible}
expected-paused-duration

Parameters on Response+:

estimated-pausing-duration

Parameters on Response–:

error: Error Object
estimated-pausing-duration (CONDITIONAL)

If a job controller issues a Pause All Tasks response–, the conditional parameter estimated-pausing-duration is specified if the reason for rejecting the request is that all activity cannot be paused within the requested desired-pausing-duration.

The semantics of the Pause All Tasks message in the guardian job control interface are identical to those of the Pause All Tasks message in the job control interface. See Section 6.4.6 on page 49 for an explanation of these semantics.

9.2.14 Pause Task

The Pause Task message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a specified task's execution to be suspended. The parameters for both the Pause Task request and its responses are as follows:

Parameters on Request:

local-task-identifier
time-frame: CHOICE OF {
desired-pausing-duration
OR as-soon-as-possible}
expected-paused-duration

Parameters on Response+:

local-task-identifier
estimated-pausing-duration

Parameters on Response–:

local-task-identifier
error: Error Object
estimated-pausing-duration (CONDITIONAL)
Guardian Task Status Object (CONDITIONAL)

If a job controller issues a Pause Task response–, the conditional parameter estimated-pausing-duration is specified if the reason for rejecting the request is that the specified task cannot be paused within the requested desired-pausing-duration. The conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

The semantics of the Pause Task message in the guardian job control interface are identical to those of the Pause Task message in the job control interface. See Section 6.4.5 on page 48 for an explanation of these semantics.

9.2.15 Reactivate Subordinate

The Reactivate Subordinate message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to resume communication with and supervision of a subordinate that it was previously instructed to ignore. The parameters for both the Reactivate Subordinate request and its responses are as follows:

Parameters on Request:

subordinate-job-controller-identifier

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

Upon receiving a Reactivate Subordinate request from the active guardian, the job controller should perform the following steps:

1. A planner has not checked the baseline configuration to determine its subordinates when in the Available administrative state. If the job controller is in the Available administrative state, issue a Reactivate Subordinate response–. No further action is necessary.

2. If the specified subordinate does not exist, issue a Reactivate Subordinate response–. No further action is necessary.
3. Otherwise, issue a Reactivate Subordinate response+.
4. Attempt to establish a connection with the specified subordinate. If it cannot establish a connection with the specified subordinate, it should specify that in an Administrative Status message.
5. If a connection can be established, the job controller should resume its supervisory responsibilities with respect to that subordinate.

9.2.16 Report Administrative Status

The Report Administrative Status message is valid for both the active and passive guardian job control interfaces. It is issued by a guardian to a job controller to request current administrative status. The parameters for both the Report Administrative Status request and its responses are as follows:

Parameters on Request:

detail-level

Parameters on Response+:

Guardian Administrative Status Object

Parameters on Response–:

error: Error Object

If a guardian wishes to receive updated administrative status about a job controller, it should issue a Report Administrative Status request to that job controller. The detail-level parameter allows a guardian to specify whether it wants information about only the job controller or the job controller's subsystem as well. See Section 9.1.1 on page 89 for a discussion of the Guardian Administrative Status Object, which explains the information provided for each detail-level.

Upon receiving a Report Administrative Status request from a guardian, a job controller should issue a Report Administrative Status response– if the detail-level is invalid. Otherwise, it should issue a Report Administrative Status response+ specifying the required information as defined by the Guardian Administrative Status Object and the detail-level.

9.2.17 Report Tasks

The Report Tasks message is valid for both the active and passive guardian job control interface. It is issued by a guardian to a job controller to request status information on a set of tasks. The parameters for both the Report Tasks request and its response are as follows:

Parameters on Request:

tasks: CHOICE OF {
list of local-task-identifier (CONDITIONAL)
OR All-Tasks
detail-level

Parameters on Response:

list of [
local-task-identifier

status: CHOICE OF {
error: Error Object
OR status: Guardian Task Status Object}
] (CONDITIONAL)

If a guardian issues a Report Tasks request and wishes to receive status on all outstanding tasks, the conditional parameter list of local-task-identifier will not be specified. If All-Tasks is specified in a Report Tasks request and a job controller does not currently have any outstanding tasks, then the Report Tasks response will have no parameters.

If a guardian wishes to receive updated status for a collection of tasks, it should issue a Report Tasks request to the job controller currently executing those tasks. The detail-level parameter allows a guardian to specify whether it wants more or less detailed information about the tasks. See Section 9.1.5 on page 90 for a discussion of the Guardian Task Status Object which explains the information provided for each detail-level.

The steps that a job controller should follow upon receiving a Report Tasks request from a guardian are identical to those of receiving a Report Tasks request from the supervisor. See Section 6.5.5 on page 59 for an explanation of these steps.

9.2.18 Resume All Tasks

The Resume All Tasks message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to switch from the Pausing, Paused or Terminated administrative state to the Active administrative state and resume the execution of all outstanding tasks. The parameters for both the Resume All Tasks request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

If the active guardian wishes to resume the activity of a job controller which has previously been instructed to suspend or terminate all activity, it should issue the job controller a Resume All Tasks request. The steps that a job controller should follow upon receiving a Resume All Tasks request from the active guardian are identical to those of receiving a Resume All Tasks request from the supervisor. See Table 13. on page 52 for an explanation of these steps.

9.2.19 Resume Task

The Resume Task message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a suspended task's execution to continue. The parameters for both the Resume Task request and its responses are as follows:

Parameters on Request:

local-task-identifier

Parameters on Response+:

local-task-identifier

Parameters on Response–:

local-task-identifier
error: Error Object
Guardian Task Status Object (CONDITIONAL)

If a job controller issues a Resume Task response–, the conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

If the active guardian wishes to resume the execution of a specific task, it should issue a Resume Task request to the job controller executing the task. The steps that a job controller should follow upon receiving a Resume Task request from the active guardian are identical to those of receiving a Resume Task request from the supervisor. See Section 6.5.6 on page 60 for an explanation of these steps.

9.2.20 Set Task Notification

The Set Task Notification message is valid for both the active and passive guardian job control interfaces. It is issued by a guardian to a job controller; it requests task status information to be dynamically reported for a set of tasks. The parameters for both the Set Task Notification request and its response are as follows:

Parameters on Request:

tasks: CHOICE OF {
list of local-task-identifier (CONDITIONAL)
OR All-Tasks
detail-level

Parameters on Response:

list of [
local-task-identifier
acceptance: Acceptance Object
] (CONDITIONAL)

If a guardian issues a Set Task Notification request and wishes to set notification for all outstanding tasks, the conditional parameter list of local-task-identifier will not be specified. If a guardian specifies All-Tasks, the Set Task Notification response will have no parameters.

If a guardian wishes to receive dynamically updated status information about a set of tasks, it should issue a Set Task Notification request to the job controller with those tasks. This message differs from the Report Tasks message in that the Report Tasks message provides a snapshot view of the status for a set of tasks; the Set Task Notification message results in the job controller issuing status messages any time the status parameters change for any of the specified tasks. The detail-level parameter allows a guardian to specify whether it wants more or less detailed information about the tasks. See Section 9.1.5 on page 90 for a discussion of the Guardian Task Status Object which explains the information provided for each detail-level.

It is a protocol violation to receive a Set Task Notification request while in the Available or E-stopped administrative state. The job controller should ignore the request and issue an Administrative Status message to the requesting guardian.

For all other administrative states, if no list of tasks is specified in the request, then the job controller will set notification for all outstanding tasks and the Set Task Notification response will have no parameters. If a list of tasks is specified in the request, then the job controller should determine the validity of each task specified by the guardian, and if valid, set notification for that

task. The job controller should issue a Set Task Notification response, enumerating each specified task and denoting whether notification was set or not.

For each task which has notification set, the job controller should issue Notify Task messages (see Section 9.3.2 on page 105) to the guardian any time any parameter of the status information changes. The detail-level used in determining how much information to provide in the Notify Task message is that which was specified in the Set Task Notification request. The job controller should continue to issue Notify Task messages for each task with notification set until a Clear Task Notification message (see Section 9.2.4 on page 93) is received which specifies that task.

9.2.21 Terminate All Tasks

The Terminate All Tasks message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests the job controller to switch to the Terminating (and eventually to the Terminated) administrative state, to indefinitely suspend the execution of all outstanding tasks, and to no longer accept Execute Task requests. The parameters for both the Terminate All Tasks request and its responses are as follows:

Parameters on Request:

<none>

Parameters on Response+:

<none>

Parameters on Response–:

error: Error Object

If the active guardian wishes to terminate the activity of a job controller, it should issue a Terminate All Tasks request to that job controller. The steps that a job controller should follow upon receiving a Terminate All Tasks request from the active guardian are identical to those of receiving a Terminate All Tasks request from the supervisor. See Table 14. on page 52 for an explanation of these steps.

9.2.22 Terminate Task

The Terminate Task message is valid for the active guardian job control interface only. It is issued by the active guardian to a job controller; it requests a specified task's execution to be indefinitely suspended. The parameters for both the Terminate Task request and its responses are as follows:

Parameters on Request:

local-task-identifier

Parameters on Response+:

local-task-identifier

Parameters on Response–:

local-task-identifier
error: Error Object
Guardian Task Status Object (CONDITIONAL)

If a job controller issues a Terminate Task response–, the conditional parameter Guardian Task Status Object is specified if it contains information related to rejecting the request.

If the active guardian wishes to indefinitely suspend the execution of a job controller's task, it should issue that job controller a Terminate Task request for that task.

Upon receiving a Terminate Task request from the active guardian, the job controller should perform the following steps:

1. If the job controller is in the Available or E-stopped administrative state, the job controller should ignore the request and issue an Administrative Status message to the requesting guardian. No further action is necessary.
2. If the specified task is an invalid task, issue a Terminate Task response–. No further action is necessary.
3. If the specified task is not in the Waiting for Guardian or Paused task-state, issue a Terminate Task response–. No further action is necessary.
4. If the task is in the Paused task-state, do the following:
   (a) Switch the task's management state to Terminating.
   (b) If the job controller has subordinates executing subtasks in support of this task, issue Terminate Task requests to those subordinates and wait for Terminate Task response+.
   (c) Abort any subtasks which are executing locally according to internal procedures.
   (d) After all subtasks (local and non-local) have been terminated, the task is terminated; switch the task to the Terminated task-state.
   (e) Issue a Terminate Task response+.
   (f) Mark all Executing and Not-Yet-Executing steps in the production plan as Scheduled.
   (g) Issue a Plan Finished request to the planner.
   (h) Issue a Task Status message to the supervisor.
   (i) No further action is necessary.
5. If the job controller is at the equipment level and the task is in the Waiting for Guardian task-state, do the following:
   (a) Switch the task's management state to Terminating.
   (b) Continue executing the task until a checkpoint is reached.
   (c) Abort the task according to internal procedures and switch the task to the Terminated task-state.
   (d) Issue a Terminate Task response+.
   (e) Issue a Plan Finished request to the planner.
   (f) Issue a Task Status message to the supervisor.
   (g) No further action is necessary.
6. If the job controller has subordinates and the task is in the Waiting for Guardian task-state, each subtask must be considered individually:
   (a) For each subtask which is Deferred, Completed or Terminated, no action is necessary.
   (b) If any subtask is in the Aborted task-state or Aborting task-management-state, issue a Terminate Task response–. Continue execution of the task if possible. No further action is necessary.
   (c) Switch the task's management state to Terminating.
   (d) For each subtask which is in the Active task-state, issue a Pause Task request (with parameter as-soon-as-possible) to the subordinate executing that subtask.
   (e) If all subordinates do not respond positively to the Pause Task request, then issue a Terminate Task response– to the guardian. Issue Resume Task requests to all subordinates who issued Pause Task responses+. Continue execution of the task if possible. No further action is necessary.

(f) If all subordinates respond positively to the Pause Task requests, then issue a Terminate Task response+ to the guardian.

(g) After all subordinates who were attempting to pause tasks have issued a Task Status message specifying that their tasks are Paused, issue a Terminate Task request for all subtasks which are in the Paused task-state. When all subtasks have been Terminated, the task is Terminated.

(h) If at any time a subtask enters an unexpected state, contact the active guardian with Task or Subtask Error Occurred.

(i) Mark all Executing and Not-Yet-Executing steps in the production plan as Scheduled.

7. Issue a Plan Finished request to the planner.
8. Issue a Task Status message to the supervisor.

9.3 Job Controller Initiated Messages

The job controller initiated messages in the guardian job control interface are:

- Administrative Status
- Notify Task
- Subordinate Problem
- Subtask Error Occurred
- Task Error Occurred

This section describes each of these messages in detail.

9.3.1 Administrative Status

The Administrative Status message is valid for both the active and passive guardian job control interfaces. It is issued by a job controller to inform a guardian of any change in its administrative status. The Administrative Status message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

administrative-state
planner-job-control-link-status
link-to-supervisor-status
operational-status
list of [
    subordinate-job-controller-identifier
    administrative-state
    link-to-subordinate-status
] (CONDITIONAL)

If a job controller issuing an Administrative Status message has no subordinates, then the list of subordinate information will not be specified.

For each established guardian job control connection, a job controller should issue Administrative Status messages each time one of the parameters in its administrative status changes. If a job controller does not currently have an established guardian job control connection and it wishes to issue an Administrative Status message to report an abnormal situation, it should turn on the job

control watchdog, and after establishing a connection to a guardian, it should issue an Administrative Status message.

9.3.2 Notify Task

The Notify Task message is valid for both the active and passive guardian job control interfaces. A Notify Task message is issued by a job controller to a guardian to report the current status of a task. The Notify Task message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

local-task-identifier
Guardian Task Status Object

For each outstanding task, a job controller should issue Notify Task messages to each guardian which has set notification for that task each time any parameter of the Guardian Task Status Object for that covenant changes. The detail-level used in determining how much information to provide in the Guardian Task Status Object is that which was specified by that guardian in its Set Task Notification request.

9.3.3 Subordinate Problem

The Subordinate Problem message is valid for the active guardian job control interface only. It is issued by a job controller to inform the active guardian that a subordinate is exhibiting anomalous behavior. When the job controller wishes to issue this message to the active guardian, the job controller should turn on the job control watchdog. The Subordinate Problem message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

subordinate-job-controller-identifier
error: Error Object

Examples of when a Subordinate Problem message would be issued include when a subordinate is not responding to requests, when a subordinate is consistently issuing inappropriate messages, or when a job controller is unable to establish a connection to its subordinate.

9.3.4 Subtask Error Occurred

The Subtask Error Occurred message is valid for the active guardian job control interface only. It is issued by a job controller to inform the active guardian that an error has occurred which affects one of its subtasks or an error has occurred to one of its subtasks which affects its task. Most subtask errors are generated by receiving an unexpected status report from a subordinate which the job controller (and the planner) cannot resolve. There are several messages that the active guardian may issue to resolve a subtask error. They include Abort Task, Call Task Completed, Change Subtask Status, Defer Task, Pause Task, Resume Task, and Terminate Task.

When the job controller wishes to issue this message to the active guardian, the job controller should turn on the job control watchdog.The Subtask Error Occurred message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

local-task-identifier
local-subtask-identifier

subordinate-job-controller-identifier
error: Error Object

9.3.5 Task Error Occurred

The Task Error Occurred message is valid for the active guardian job control interface only. It is issued by a job controller to inform the active guardian that an error has occurred which is either intimately related to a specific task or which affects the job controller's ability to complete the execution of a task. In general, all equipment failures will cause a job controller to issue a Task Error Occurred message. Errors specific to a task may also cause a job controller to issue a Task Error Occurred (e.g., workpieces or tools not available for current task). There are several messages that the active guardian may issue to resolve a task error. They include Abort Task, Call Task Completed, Defer Task, Pause Task, Resume Task, and Terminate Task.

When the job controller wishes to issue this message to the active guardian, the job controller should turn on the job control watchdog. The Task Error Occurred message is an unconfirmed message; it has no response. The parameters are as follows:

Parameters:

local-task-identifier
error: Error Object

10 Example Error Recovery Scenario

This section presents an example scenario of task lateness to show how the messages from the different interfaces interact. The messages which are used in this scenario are Break Covenants, Covenants Broken, Replan Late Plan, Replan Late Step, and Task Status. Figure 11. provides a pictorial view of the scenario.

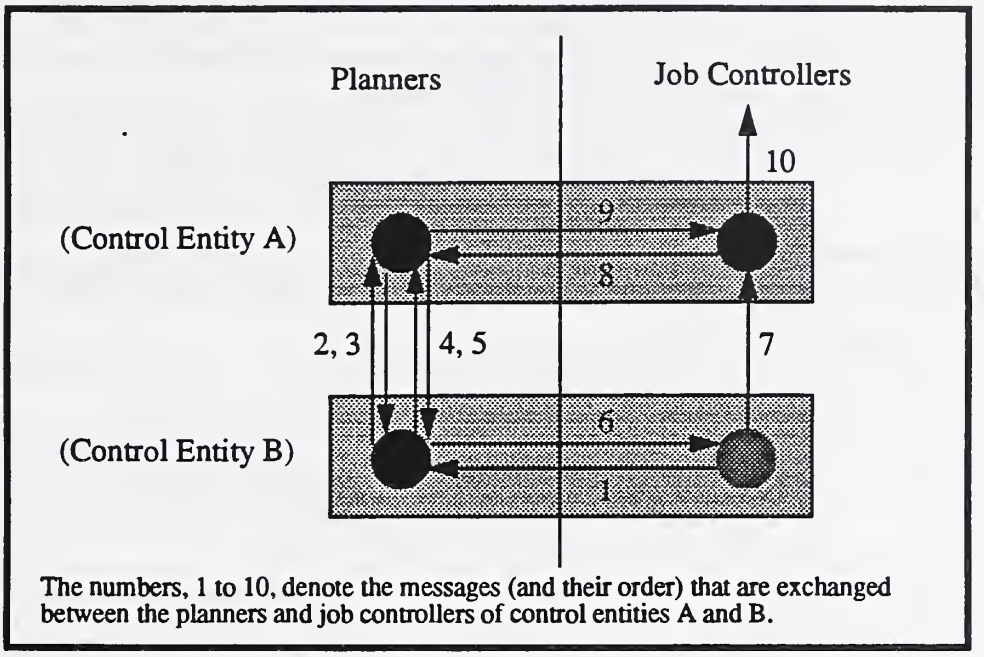


Figure 11. Task Late Scenario

The following situation currently exists:

- Control Entity B is an equipment level control entity which is supervised by Control Entity A.
- Control Entity A's planner had previously scheduled a covenant for the operation 'DrillHole99821' with Control Entity B's planner to be executed at August 23, 1993 10:00:00 for 15 minutes.
- At August 23, 1993 10:00:00, Control Entity A's job controller issued Control Entity B's job controller an Execute Task request for the operation 'DrillHole99821'.
- Control Entity B's job controller has begun executing the task.
- The current time is August 23, 1993 10:13:00.
- Control Entity B's job controller re-calculates the estimated-completion-time for the operation 'DrillHole99821' to be August 23, 1993 10:21:00. It realizes that this estimate exceeds the scheduled-completion-time (August 23, 1993 10:15:00).

Table 15. on page 108 presents the dialog which occurs among the planners and job controllers of control entities A and B.[12]

12. P_A denotes Control Entity A's planner (similarly for P_B) and JC_A denotes Control Entity A's job controller (similarly for JC_B).

	Issuing Entity	Receiving Entity	Action
1	JC_B	P_B	JC_B issues P_B a *Replan Late Plan request* for the operation 'DrillHole99821' specifying an estimated-completion-time of August 23, 1993 10:21:00.
2	P_B	P_A	P_B consults B's schedule to determine if this task's lateness will affect any subsequent operations for the resource. It determines that the operation 'DrillHole8827' will be affected. P_B issues a *Break Covenants request* to P_A, requesting to break the contract for operation 'DrillHole8827' in order to complete the current task, 'DrillHole99821'.
3	P_A	P_B	P_A decides to grant permission to break the contract for 'DrillHole8827' in order to complete the currently executing 'DrillHole99821' because it is easier to reschedule an operation than terminate an executing operation and resume it at some future time. P_A issues a *Break Covenants response* to P_B granting permission.
4	P_B	P_A	Because P_A gave P_B permission to break the contract for 'DrillHole8827', P_B issues a *Covenants Broken request* to P_A, specifying the covenant for operation 'DrillHole8827'.
5	P_A	P_B	P_A issues a *Covenants Broken response* to P_B. It then attempts to reschedule that operation for another time, and possibly with another subordinate.
6	P_B	JC_B	P_B, upon receiving the *Covenants Broken response* from P_A, will remove the covenant and production plan associated with operation 'DrillHole8827'. It will extend the scheduled-completion-time for 'DrillHole99821' to the estimated-completion-time provided by JC_B (August 23, 1993 10:21:00) and issue a *Replan Late Plan response+* with action 'Resume' to JC_B.
7	JC_B	JC_A	After receiving the *Replan Late Plan response+* from P_B, JC_B will issue a *Task Status* message to JC_A, specifying the estimated-completion-time for the task.
8	JC_A	P_A	After receiving the *Task Status* message from JC_B, JC_A determines that the new estimated-completion-time for the step 'DrillHole99821' in its production plan 'MakeWidget3820' conflicts with the scheduled-start-time for the subsequent step in that production plan, 'DeliverToDeburr1022'. JC_A will issue a *Replan Late Step request* to P_A, specifying the production plan 'MakeWidget3820' and the step 'DrillHole99821'.
9	P_A	JC_A	PA will attempt to reschedule the steps in the production plan 'MakeWidget3820', starting with the step following 'DrillHole99821'. It will reschedule each step, in turn, until a step is reached whose scheduled-completion-time does not overlap with the scheduled-start-time of the subsequent steps. After completing this, P_A issues a Replan Late Step response+ with action 'Resume' to JC_A.
10	JC_A	JC_A's supervisor	After receiving the *Replan Late Step response+* from P_A, JC_A will continue monitoring its subtask 'DrillHole99821'. It will issue a *Task Status* message to the supervisor.

Table 15. Task Late Scenario

11 Summary

This specification is the second iteration defining the control entity interfaces for a control entity conforming to the MSI architecture.

11.1 Issues Addressed

As currently specified, this specification supports finite capacity scheduling and rescheduling, task execution and monitoring, human monitoring and intervention at all levels of control within a shop, error detection and limited error recovery. This specification is sufficient to support recovery from errors which require rescheduling only (lateness and repeatable tasks).

11.2 Issues Remaining

This specification needs to be extended to support recovery that involves dynamically re-allocating resources, dynamic material handling, and eventually dynamic process planning.

12 Glossary

acceptance-code: an integral value denoting acceptance or an error-code. See Section 4.4 on page 25.

Acceptance Object: see Section 4.4 on page 25.

acceptance-text: a textual description of an acceptance code.

action: an instruction which a planner may issue to a job controller to resolve a replanning error or anomaly.

administrative-state: the current state of a planner (see Section 5.2 on page 28) or job controller (see Section 6.1 on page 42).

All-Bids: a value which denotes all covenants which are currently in the Pre-Bidding, Bidding, Bidden, Breaking Bid, or Accepting covenant-state.

All-Contracts: a value which denotes all covenants which are currently in the Contracted, Breaking Contract, or Executing covenant-state.

All-Covenants: a value which denotes all covenants which are currently in the Pre-Bidding, Bidding, Bidden, Breaking Bid, Removing Covenant, Accepting, Contracted, or Breaking Contract or Executing covenant-state.

All-Tasks: a value which denotes all outstanding tasks.

as-soon-as-possible: a value which denotes 'as soon as possible'.

completion-time: an absolute time which is used to specify the end of a time interval.

Covenant Name Object: see Section 8.1.2 on page 74.

Covenant Status Object: see Section 5.4 on page 30.

covenant-state: see Section 5.3 on page 29.

desired-pausing-duration: a time duration which specifies the maximum time a supervisor job controller or a guardian wishes to wait for a set of tasks to reach the Paused task-state.

detail-level: a parameter which indicates the desired level of detail to be reported in a given message. Detail-level is used in the guardian planning and guardian job control interface for administrative status reporting, covenant status reporting and task status reporting.

earliest-start-time: an absolute time which indicates the earliest acceptable time at which an operation may be begun.

error-code: an integral value denoting an error condition. See Section 4.3 on page 24.

Error Object: see Section 4.3 on page 24.

error-text: a textual description of an error-code.

estimated-completion-time: an absolute time which specifies a job controller's estimate of when a task will complete execution.

estimated-pausing-duration: a time duration which specifies a job controller's estimate of how much time is needed for a set of tasks to reach the Paused task-state.

estimated-remaining-duration: a time duration which specifies a job controller's estimate of how much time is needed to complete the execution of a task.

expected-paused-duration: a time duration which specifies how much time a supervisor job controller or a guardian expects a set of tasks to remain in the Paused task-state.

global-plan-reference: an operator-understandable globally unique identifier for a plan.

global-step-reference: an operator-understandable globally unique identifier for a step in a plan.

Guardian Administrative Status Object: see Section 8.1.1 on page 74 and Section 9.1.1 on page 89.

Guardian Covenant Status Object: see Section 8.1.5 on page 75.

Guardian Task Status Object: see Section on page 90.

late: a boolean parameter which indicates whether a given task's execution is expected to complete later than its scheduled-completion-time. 'True' indicates that the task is expected to be late; 'false' indicates that the task's execution is expected to complete early or on-time.

latest-completion-time: an absolute time which indicates the latest acceptable time for an operation to be completed.

level: a parameter which denotes at what level within the control hierarchy a given planner or job controller exists. Possible values are equipment, workcell and shop.

link-to-subordinate-status: a parameter which indicates the status of a logical connection between either a job controller and one of its subordinates or a planner and one of its subordinates. Possible values are unconnected, listening, connecting, connected, and disconnecting.

link-to-supervisor-status: a parameter which indicates the status of a logical connection between either a job controller and the supervisor or a planner and the supervisor. Possible values are unconnected, listening, connecting, connected and disconnecting.

local-covenant-identifier: a unique identifier for a covenant which is specified by and meaningful to the planner responsible for planning the covenant.

local-subcovenant-identifier: a unique identifier for a subcovenant which is specified by and meaningful to the planner responsible for planning the covenant of which the subcovenant is a component.

local-subtask-identifier: a unique identifier for a subtask which is specified by and meaningful to the job controller responsible for executing the task of which the subtask is a component.

local-task-identifier: a unique identifier for a task which is specified by and meaningful to the job controller responsible for executing the task.

make: a parameter used in the Identify message which indicates the manufacturer of a planner or job controller software.

model: a parameter used in the Identify message which indicates the model type of a planner or job controller software.

operational-status:vendor-specific information detailing status information about a job controller (e.g., coolant-level, joint-position, spindle-position).

parameter-name: the name of a plan parameter (input or output).

parameter-value: the value of a plan parameter (input or output).

plan-identifier: an attribute of a plan, which along with plan-version and plan-location uniquely identifies a plan.

Plan Identifier Object:see Section 4.1 on page 24.

plan-location: an attribute of a plan which specifies where a plan is located. Along with plan-identifier and plan-version, it uniquely identifies a plan.

Plan Parameter Object: see Section 4.2 on page 24.

plan-state: see Section 3.1 on page 21.

plan-version: an attribute of a plan, which along with plan-identifier and plan-location uniquely identifies a plan.

planner-job-control-link-status: a parameter which indicates the status of a logical connection between a job controller and a planner. Possible values are unconnected, listening, connecting, connected and disconnecting.

planning-strategy: a parameter which denotes an algorithm which a planner may use to schedule a covenant (see Section 5.1.1 on page 26).

priority: a parameter which specifies the relative importance of a covenant to the shop's commitments. It is used in some planning-strategies to determine which existing covenants may be modified in order to accommodate a new covenant.

scheduled-completion-time: an absolute time which specifies a planner's estimate of when a task's or subtask's execution should complete. It may also specify the time at which a resource is no longer available to perform a specified task or subtask.

scheduled-start-time: an absolute time which specifies a planner's estimate of when a task's or subtask's execution should begin. It may also specify the time at which a resource becomes available to perform a specified task or subtask.

start-time: an absolute time which is used to specify the beginning of a time interval. Used in the Clear Schedule guardian planning message.

step-identifier: a unique identifier for a step within a plan.

subcovenant-state: the current state of a subcovenant. See Section 5.3 on page 29.

Subcovenant Status Object: see Section 8.1.4 on page 75.

subordinate-covenant-identifier: a unique identifier for a planner's covenant which is specified by and meaningful to the planner responsible for scheduling the covenant.

subordinate-job-controller-identifier: a unique identifier for a subordinate job controller.

subordinate-planner-identifier: a unique identifier for a subordinate planner.

subordinate-subcovenant-identifier: a unique identifier for a planner's subcovenant which is specified by and meaningful to that planner's subordinate.

subordinate-subtask-identifier: a unique identifier for a job controller's subtask which is specified by and meaningful to that job controller's subordinate.

subordinate-task-identifier: a unique identifier for a job controller's task which is specified and meaningful to the job controller responsible for executing the task.

subtask-management-state: the current management state of a subtask. See Section 6.2 on page 42).

subtask-state: the current state of a subtask. See Section 6.2 on page 42.

Subtask Status Object: see Section 9.1.4 on page 90.

supervisor-covenant-identifier: a unique identifier for a planner's covenant which is specified by and meaningful to the supervisor of the planner responsible for scheduling the covenant.

supervisor-manufacturing-unit-identifier: an identifier for a logical unit which represents the workpieces on which to perform a specified operation.

supervisor-task-identifier: a unique identifier for a job controller's task which is specified by and meaningful to the supervisor of the job controller responsible for executing the task.

Task Name Object: see Section 9.1.2 on page 89.

Task Status Object: see Section 6.3 on page 44.

task-management-state: the current management state of a task. See Section 6.2 on page 42.

task-state: the current state of a task. See Section 6.2 on page 42.

terminated-flag: a boolean parameter specified in a Request for Bid request which denotes whether the specified plan is a production-managed plan to be scheduled, or a partially executed production plan which should be rescheduled.

version: a parameter used in the Identify message which denotes the version of the planner or job controller software.

work-element-identifier: a unique identifier for an operation which a job controller innately knows how to perform. In general, only equipment job controller's have work-elements.

References

[Barkmeyer, 1993] Barkmeyer, E., S. Wallace, S. Ray, E. Wallace and M.K. Senehi, "Manufacturing Systems Integration: Information Models", National Institute of Standards and Technology Interagency Report, forthcoming (to be available from the National Technical Information Service, Springfield, VA 22161).

[Catron, 1991] Catron, B. and S. Ray, "ALPS - A Language for Process Specification," *International Journal of Computer Integrated Manufacturing*, Volume 4, Number 2, pp. 105-113, 1991.

[Ray, 1992a] Ray, S., "Using the ALPS Process Plan Model," Proceedings of the ASME Manufacturing International Conference, Dallas, TX, 1992.

[Ray, 1992b] Ray, S. and S. Wallace, "A Production Management Information Model for Discrete Manufacturing," submitted to *Production Planning and Control Journal*, September, 1992.

[Senehi, 1991a] Senehi, M.K., E. Barkmeyer, M.E. Luce, S.R. Ray, E.K. Wallace, and S. Wallace, "Manufacturing Systems Integration Initial Architecture Document", National Institute of Standards and Technology, Interagency Report 91-4682, September 1991 (Available from the National Technical Information Service, Springfield, VA 22161).

[Senehi, 1991b] Senehi, M.K., S. Wallace, E. Barkmeyer, M.E. Luce, S.R. Ray, and E.K. Wallace, "Manufacturing Systems Integration Control Entity Interface Document", National Institute of Standards and Technology, Interagency Report 91-4626, June 1991 (Available from the National Technical Information Service, Springfield, VA 22161).

[Senehi, 1992] Senehi, M.K., S. Wallace, and M.E. Luce, "An Architecture for Manufacturing Systems Integration", Proceedings of the ASME Manufacturing International Conference, Dallas, TX, 1992.