



SEMI E86-0999

PROVISIONAL SPECIFICATION FOR CIM FRAMEWORK FACTORY LABOR COMPONENT

This specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on April 15, 1999. Initially available on SEMI OnLine August 1999; to be published September 1999.

1 Purpose

1.1 The Factory Labor Component provides the capability to support the qualification and management of available, qualified human resources for manufacturing operations.

1.2 The technical content of this ballot addresses the portion of the CIM Framework domain concerned with management of the availability of qualified factory personnel. This specification provides the interfaces required by Manufacturing Execution Systems to:

- Configure the Person resource model.
 - Each Person's name (or appropriate alias), identification, department, role, etc. There may be cases where identifying information should not be used to associate a person with the required factory labor records.
 - Each Person's initial skills and authorization to perform jobs and access data.
 - Each Person's initial assignment to other factory resources (such as specific machines or factory areas).
 - Each Person's medical qualifications.
- Define skill maintenance tasks and what triggers them.
- Track and report the Person's skills, authorization, and assignments and record these in the Person's histories.
- Support assignment of Persons to factory jobs and monitor and record job progress (from the perspective of the Person's role).
- Monitor skill maintenance triggers (such as expired skill certification) and recommend skill maintenance jobs.
- Execute triggered skill maintenance jobs.
 - Change Person capabilities (before and after training).
 - Execute and monitor skill maintenance jobs (training, certification testing) and report job progress.

- Record skill maintenance triggers and job results in Person's skill maintenance histories.
- Collect and report Person and skill performance (utilization and effectiveness).
- Maintain relationships between Persons and system security for identity authentication and authorization.

2 Scope

2.1 Specifically, this component provides the services to:

- Manage the assignment of qualified and available labor resources for factory operations.
- Manage the training and medical requirements for factory operations.
- Support the maintenance of personnel information (full name, employee number, department, shift assignment, machines qualified to operate, course and medical examination records).
- Support the maintenance of personnel skill qualification histories (courses and medical examinations successfully completed) of factory personnel.

2.2 The Factory Labor Component is divided into two subcomponents:

- Person Management, which tracks, logs, and maintains availability and task assignment for factory operations personnel within the manufacturing facility. Also provides support for the management of personnel information, training, and medical qualifications.
- Skill Management, which provides support for the management of training and medical requirements for factory operations.

2.3 Figure 1 depicts the relationships between these two components and their interaction with other factory components.

2.4 This standard does not purport to address all of the safety issues associated with its use. It is the responsibility of the users of this standard to establish



appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.

3 Limitations

3.1 This specification is designated as provisional due to known areas that need to be completed. The following items summarize the deficiencies of the provisional specification to be addressed before a subsequent ballot to upgrade it to full standard status.

3.2 Provisional Status

3.2.1 The following items need to be completed before conducting a subsequent ballot to upgrade the Factory Labor Component to full standard status.

- Define skill maintenance tasks and what triggers them.
- Monitor skill maintenance triggers (such as expired skill certification) and recommend skill maintenance jobs.
- Maintain relationships between persons and system security for identity authentication and authorization.

3.2.2 It is anticipated that a future document defining the Resource Abstract Interface will satisfy the first two items.

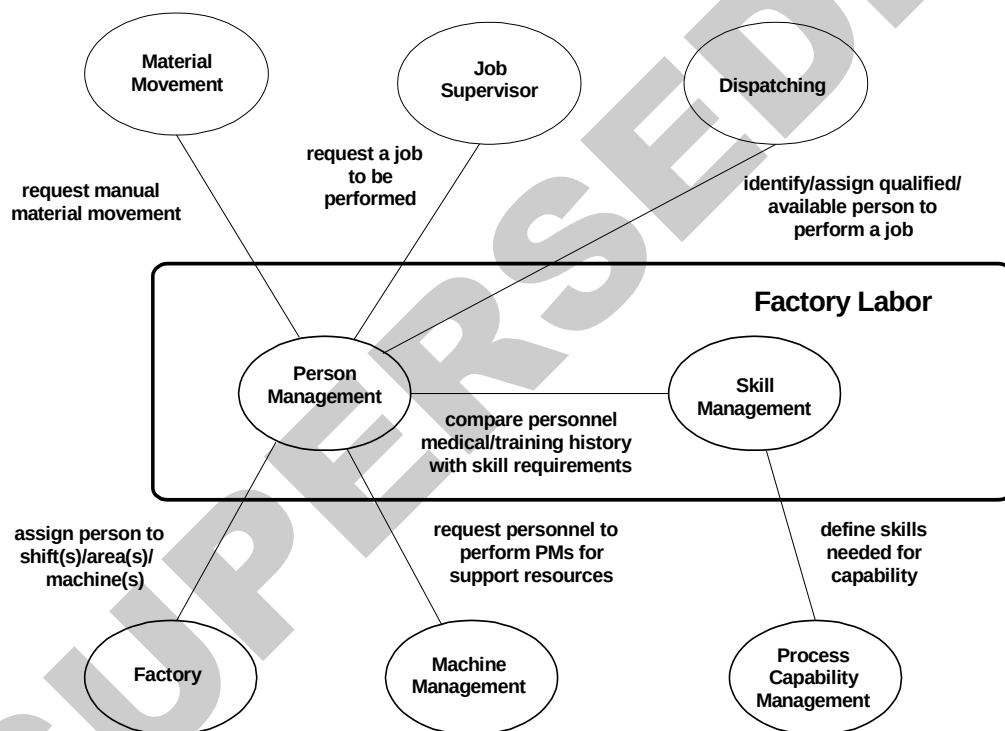


Figure 1
Factory Labor Component Relationships



4 Referenced Standards

4.1 ISO/IEC Standard¹

ISO/IEC 14750 (also ITU-T Recommendation X.920) — Information Technology – Open Distributed Processing – Interface Definition Language

4.2 OMG Standard²

UML Notation Guide, Version 1.1, document number ad/97-08-05

NOTE: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

person — personnel performing manufacturing operations within a factory.

person responsibility — the concept of “responsibility” implies that factory personnel are authorized to perform, operate or access a specific factory object (but it does not imply ownership).

skill — skill is an attribute of factory personnel denoting that they are qualified to assist a process resource or job in the performance of a process capability or some other factory operation.

skill qualification — a skill qualification indicates that factory personnel meet a requirement (or all requirements) necessary to perform some factory operation.

skill requirement — a skill requirement is the training or medical examination(s) required to achieve a specific skill.

6 Requirements

6.1 The following sections provide the specification of interfaces that comprise the Factory Labor Component. The model notation used is the Unified Modeling Language (UML). UML is documented in the UML Notation Guide. The details of each interface are supplied using standard IT-ODP-IDL. Conformant implementations of the Factory Labor Component shall provide support for all elements of the IDL specifications and the semantics provided in the UML models. Conformant implementations are not constrained to any specific implementation approach, provided that a mapping is provided to the IDL specification.

6.2 External Interfaces, Global Type Definitions, Global Exceptions and Events

6.2.1 The following interfaces (external to this component), global type definitions, global exceptions, and events are required to support the Factory Labor component specification.

6.2.2 Abstract and Base Interfaces

6.2.2.1 Base interfaces provide a suite of interfaces that are used throughout the CIM Framework Specification. The Abstract Interfaces provide a hierarchy of abstract interfaces that are used as “Inherited Interfaces” in the rest of the framework. These interfaces are expected to be specialized in a specific implementation.

6.2.2.2 This section identifies base and abstract interfaces required by the Factory Labor component specification.

```
interface NamedEntity {  
};  
  
interface OwnedEntity : NamedEntity {  
};  
  
interface Resource : OwnedEntity {  
};
```

¹ ISO Central Secretariat, 1, rue de Varembe, Case postale 56, CH-1211 Genève 20, Switzerland

² Object Management Group, Inc., Framingham Corporate Center, 492 Old Connecticut Path, Framingham, MA 01701, 1.508.820.4300,
<http://www.omg.org/cgi-bin/doclist.pl>



```
interface ComponentManager : Resource {  
};
```

```
interface Job {  
} ;
```

6.2.3 Other CIM Framework Interfaces

6.2.3.1 This section identifies interfaces within other CIM Framework components that are needed to satisfy relationships with the Factory Labor component.

```
interface HistoryCollection { // from Factory Services  
} ;
```

```
interface History { // from Factory Services  
} ;
```

```
interface MESFactory { // from Factory Management  
} ;
```

6.2.4 Global Type Definitions

6.2.4.1 The CIM Framework uses IT-ODP-IDL *typedefs* in two ways:

- To define aliases for compound object types (*structs*), which can then be used in IDL specifications and
- To define aliases for basic object types, but with additional implied semantics. (e.g., the *units* typedef defines a *string*, whose contents conform to definitions found in SEMI E5).

6.2.4.2 This section identifies the global type definitions required by the Factory Labor component specification.

```
typedef sequence<string> stringSequence ;
```

```
typedef struct HistoryEvent_body {  
    string subject ;  
} HistoryEvent ;
```

```
struct ulonglong {  
    unsigned long low ;  
    unsigned long high ;  
} ;
```

```
typedef ulonglong TimeT ;
```

```
typedef TimeT TimeStamp ;
```

```
struct IntervalT {  
    TimeT lower_bound ;  
    TimeT upper_bound ;  
} ;
```

```
typedef IntervalT TimeWindow ;
```

```
typedef struct Shift_Structure {
```



```
        string name ;
        // ...
    } Shift ;

typedef struct PersonResponsibilitystruct {
    string responsibilityCategory ;
    any responsibility ;
} PersonResponsibility;

typedef sequence<ProcessCapability> ProcessCapabilitySequence ;
typedef stringSequence CategorySequence ;
typedef sequence<HistoryEvent> HistoryEventSequence ;
typedef sequence<Job> JobSequence ;
typedef sequence<Machine> MachineSequence ;
typedef sequence<Person> PersonSequence ;
typedef sequence<QualificationData> QualificationDataSequence ;
typedef sequence<Resource> ResourceSequence ;
typedef sequence<SkillRequirement> RequirementSequence ;
typedef sequence<PersonResponsibility> ResponsibilitySequence ;
typedef sequence<Skill> SkillSequence ;

typedef string PropertyName;
struct Property
{
    PropertyName property_name;
    any property_value;
};
typedef sequence<Property> Properties;
```

6.2.5 Global Exception Declarations

6.2.5.1 An exception is an infrastructure mechanism used to notify a calling client of an operation that an unusual condition occurred in carrying out the operation. This section identifies the global exception declarations required by the Factory Labor component specification.

```
exception FrameworkErrorSignal {
    unsigned long errorCode ;
    any errorInformation ;
} ;

exception InvalidStateTransitionSignal {
} ;

exception SetValueOutOfRangeSignal {
} ;
```

6.2.6 Events

6.2.6.1 An event is an asynchronous message denoting the occurrence of some incident of importance. For example, state change or new object created. This section identifies the events defined by the Factory Labor component specification. Note that “//” or “/* ...*/” infers a comment.

```
/* Person lifecycle event definition. */
const string PersonLifecycleSubject =
    "/PersonManagement/PersonManager/PersonLifecycle" ;
```



```
typedef struct PersonLifecycleFilters_Structure {
    Property      name;           // "Name", aPerson's name
    Property      lifecycleEvent; // "LifecycleEvent", LifecycleState
} PersonLifecycleFilters;
typedef struct PersonLifecycleEvent_Structure {
    string          eventSubject;
    TimeStamp       eventTimeStamp;
    PersonLifecycleFilters eventFilterData;
    Properties      eventNews;
    Person          aPerson;
                // Except when "Delete" where value will be nil
} PersonLifecycleEvent;

/* Person capacity changed event definition. */
const string PersonCapacityChangedSubject =
    "/PersonManagement/Person/CapabilityChanged" ;
typedef struct PersonCapacityChangedFilters_Structure {
    Property      name;           // "Name", aPerson's name
} PersonCapacityChangedFilters;
typedef struct PersonCapacityChangedEvent_Structure {
    string          eventSubject;
    TimeStamp       eventTimeStamp;
    PersonCapacityChangedFilters eventFilterData;
    Properties      eventNews;
    Person          aPerson;
} PersonCapacityChangedEvent;

/* Person state changed event definition. */
const string PersonStateChangedSubject = "/PersonManagement/Person/StateChanged";
enum PersonState {PersonDefined, PersonOffShift, PersonOnShift,
    PersonAvailableForWork, PersonNotAvailableForWork, PersonUnassignedToMachines,
    PersonAssignedToJobs, PersonAssignedToMachines, PersonAvailableForMoreAssignments,
    PersonAssignmentAtCapacityExceeded, PersonIdleWithJob, PersonBusyWithJob };
typedef struct PersonStateChangedFilters_Structure {
    Property      name;           // "Name", aPerson's name
    Property      formerState;    // "FormerState", PersonState
    Property      newState;       // "NewState", PersonState
} PersonStateChangedFilters;
typedef struct PersonStateChangedEvent_Structure {
    string          eventSubject;
    TimeStamp       eventTimeStamp;
    PersonStateChangedFilters eventFilterData;
    Properties      eventNews;
    Person          aPerson;
} PersonStateChangedEvent;

/* Person shift worked changed event definition. */
const string PersonShiftChangedSubject = "/PersonManagement/Person/ShiftChanged";
typedef struct PersonShiftChangedFilters_Structure {
    Property      name;           // "Name", aPerson's name
    Property      fromShift;      // "FromShift", former shift
    Property      toShift;        // "ToShift", new shift
} PersonShiftChangedFilters;
```




6.3.3 *PersonManager Interface*

Interface: **PersonManager**

Inherited Interface: **ComponentManager**

Description: This is the interface between the Person Management Component and the rest of the CIM environment. The **PersonManager** creates instances of Persons within the factory and manages those Persons (shift, job assignment and status information). There are no limitations to the number of **PersonManagers** within a factory.

Exceptions:

/* This signal is raised with attempted creation of Person through the use of a Person identifier that already exists. */

```
exception PersonDuplicateSignal {string identificationNumber;;};
```

/* This signal is raised when a retrieval operation fails because the Person specified by the argument could not be located. */

```
exception PersonNotFoundSignal {string identificationNumber;;};
```

/* This signal is raised when a removal operation fails because the Person specified by the argument is not assigned to the **PersonManager**. */

```
exception PersonNotAssigned { Person unknownPerson;;};
```

```
exception PersonRemovalFailedSignal { };
```

Published Events:

PersonLifecycleEvent

Provided Services:

/* Create an instance of Person with a unique identifier and add to the list managed by **PersonManager**. */

```
Person createPersonWithIdentifier (in string identificationNumber)
    raises (FrameworkErrorSignal, PersonDuplicateSignal);
```

/* Delete the Person identified by the argument from the list managed by **PersonManager**. */

```
void removePerson (in Person aPerson)
    raises (FrameworkErrorSignal, PersonRemovalFailedSignal,
           PersonNotAssignedSignal);
```

/* Return a set of Persons managed by the **PersonManager**. */

```
PersonSequence allPersons ( ) raises (FrameworkErrorSignal);
```

/* Search for an instance of Person using its unique identifier. Raises an exception if the Person is not found. */

```
Person findByIdentifier (in string identificationNumber)
    raises (FrameworkErrorSignal, PersonNotFoundSignal);
```

/* Return a set of all Persons logged on to the specified shift. */

```
PersonSequence allOnShiftPersons (in shift aShift) raises (FrameworkErrorSignal);
```

/* Return a set of all Persons available for work. */

```
PersonSequence allAvailableForWorkPersons ( ) raises (FrameworkErrorSignal);
```

/* Return a set of all Persons not available for work. */

```
PersonSequence allNotAvailableForWorkPersons ( ) raises (FrameworkErrorSignal);
```

```
/****** Manage Person / Machine relationship. *****/
```



/* Return a set of Persons qualified to operate a process or material handling machine. The Person is qualified to operate the machine if the machine is noted in the Person's skill list or if a comparison of the person's course and medical exam histories satisfy the requirements defined within the Skill Management Component for the noted machine. This method does not check the availability of the person to actually perform a task. */

```
PersonSequence allPersonsWithSkill (in Skill aSkill) raises (FrameworkErrorSignal);
```

/* Return a set of Persons available ("on duty" on the current shift) to operate the specified machine. This service assumes that a Person is qualified to operate the machine. */

```
PersonSequence allPersonsAvailableForMachine (in Machine aMachine)  
    raises (FrameworkErrorSignal);
```

/* Return a set of all Persons currently assigned to the specified machine. */

```
PersonSequence allPersonsAssignedToMachine (in Machine aMachine)  
    raises (FrameworkErrorSignal);
```

/* Return a set of all Persons currently assigned to machines. */

```
PersonSequence allAssignedToMachinesPersons ( ) raises (FrameworkErrorSignal);  
/***** Manage Person/Job relationship. *****/
```

/* Return a set of Persons qualified to perform a process or material handling job. The Person is qualified for the job if the job qualification is noted in the person's skill list or if a comparison of the person's course and medical exam histories satisfy the requirements defined within the Skill Management Component for the noted job. This method does not check the availability of the person to actually perform a task. */

```
PersonSequence allPersonsQualifiedForJob (in Job aJob)  
    raises (FrameworkErrorSignal);
```

/* Return a set of Persons available ("on duty" on the current shift) to perform a process or material movement job. This service assumes that a Person is qualified to perform the job. */

```
PersonSequence allPersonsAvailableForJob (in Job aJob)  
    raises (FrameworkErrorSignal);
```

/* Return a set of Persons currently assigned to the specified job. */

```
PersonSequence allPersonsAssignedToJob (in Job aJob) raises (FrameworkErrorSignal);
```

/* Return a set of all Persons currently assigned to jobs. */

```
PersonSequence allAssignedToJobsPersons ( ) raises (FrameworkErrorSignal);  
Contracted Services:    None.
```

Dynamic Model: Same as inherited interface (Component Manager).

6.3.4 *Person Interface*

Interface: Person

Inherited Interface: Resource

Description: This interface represents factory personnel. As currently defined, Person refers specifically to operating personnel within the manufacturing facility. Instances of this interface hold the following information pertaining to these individuals: employee information (full name, identification number, department, shift assignment); training and medical certification; and factory operations (e.g., operate a machine, perform a process) qualified to perform.

Exceptions:



/* This signal is raised when an addition operation fails because the Qualification specified by the argument already exists. */

```
exception QualificationDuplicateSignal {  
    QualificationData qualificationInformation;;
```

/* This signal is raised when a removal operation fails because the Qualification specified by the argument could not be located. */

```
exception QualificationNotAssignedSignal {string identificationNumber;;
```

/* This signal is raised when a retrieval operation fails because the Qualification specified by the argument could not be located. */

```
exception QualificationNotFoundSignal {string identificationNumber;;
```

/* This signal is raised when an addition operation fails because the Responsibility specified by the argument already exists. */

```
exception ResponsibilityDuplicateSignal {any responsibility ;  
    string responsibilityCategory;;
```

/* This signal is raised when an operation fails because the Responsibility category specified by the argument could not be located. */

```
exception ResponsibilityCategoryNotFoundSignal {  
    string responsibilityCategory;;  
exception ResponsibilityCategoryRemovalFailedSignal { };
```

/* This signal is raised when a removal operation fails because the Responsibility specified by the argument could not be located. */

```
exception ResponsibilityRemovalFailedSignal {  
    any responsibility;  
    string responsibilityCategory;;
```

/* This signal is raised when an addition operation fails because the skill specified by the argument already exists. */

```
exception SkillDuplicateSignal {Skill aSkill;;
```

/* This signal is raised when a removal operation fails because the skill specified by the argument could not be located. */

```
exception SkillRemovalFailedSignal {Skill aSkill;;  
exception SkillNotAssignedSignal {Skill aSkill;;
```

Published Events:

/* This event is posted when a Person's capacity to do work has changed. This includes assignment to/deassignment from machines and jobs. It also includes changing the number of jobs and/or machines to which a Person may be assigned. */

PersonCapacityChangedEvent

/* This event is posted when a Person's availability has changed. This includes logging on/off a shift, leaving/returning to a work station, or capacity exceeded. */

PersonStateChangedEvent

/* This event is posted when a Person has been reassigned to a different shift. */

PersonShiftChangedEvent