MAPL Type System and Message Types Specification

1. Primitive Types

Basic Types

```
Boolean     ::= true | false
Number      ::= Integer | Float | Double
String      ::= "..." // UTF-8 encoded
Timestamp   ::= ISO8601 formatted string
UUID        ::= RFC4122 formatted string
Byte        ::= 8-bit value
Null        ::= null
```

Extended Primitives

```
Duration    ::= Number + Unit    // e.g., "30s", "5m", "1h"
URI         ::= RFC3986 formatted string
Hash        ::= String[32|64]    // SHA-256/512 hash
PublicKey   ::= String           // X.509 encoded
```

2. Composite Types

Collection Types

```
Array<T>    ::= [T]              // Ordered list of type T
Set<T>      ::= {T}              // Unique unordered collection
Map<K,V>    ::= {K: V}           // Key-value pairs
Queue<T>    ::= Queue[T]         // FIFO structure
Stack<T>    ::= Stack[T]         // LIFO structure
```

Complex Types

```
Option<T>   ::= T | null         // Optional value
Result<T,E> ::= Ok(T) | Err(E) // Success/failure container
Future<T>   ::= Promise<T>       // Asynchronous value
Stream<T>   ::= Infinite<T>      // Continuous data flow
```

3. Protocol-Specific Types

Identity Types

```
AgentID    ::= UUID
SessionID  ::= UUID
TaskID     ::= UUID
ResourceID ::= UUID
```

State Types

```
Status     ::= INIT | ACTIVE | WAITING | COMPLETED | ERROR | CANCELLED
Priority   ::= HIGH | MEDIUM | LOW
Permission ::= READ | WRITE | EXECUTE | ADMIN
Health     ::= HEALTHY | DEGRADED | FAILED
```

Security Types

```
Credential ::= {
    type: "jwt" | "x509" | "oauth",
    value: String,
    expiry: Timestamp
}

Permission ::= {
    resource: ResourceID,
    action: String,
    constraints: Map<String, Any>
}
```

4. Message Types

Base Message Structure

```
Message    ::= {
    id: UUID,
    type: MessageType,
    timestamp: Timestamp,
    sender: AgentID,
    receiver: AgentID,
    priority: Priority,
    payload: Any,
    metadata: Map<String, Any>
}
```

System Messages

```
HeartbeatMessage ::= {
    extends: Message,
    payload: {
        status: Health,
        metrics: Map<String, Number>
    }
}

ErrorMessage ::= {
    extends: Message,
    payload: {
        code: String,
        description: String,
        stackTrace: Option<String>,
        severity: HIGH | MEDIUM | LOW
    }
}

AckMessage ::= {
    extends: Message,
    payload: {
        referenceId: UUID,
        status: SUCCESS | FAILURE
    }
}
```

Task-Related Messages

```
TaskAssignment ::= {
  extends: Message,
  payload: {
    taskId: TaskID,
    description: String,
    parameters: Map<String, Any>,
    constraints: {
      deadline: Timestamp,
      resources: Array<ResourceID>,
      priority: Priority
    }
  }
}

TaskProgress ::= {
  extends: Message,
  payload: {
    taskId: TaskID,
    progress: Number,  // 0-100
    status: Status,
    metrics: Map<String, Number>,
    timestamp: Timestamp
  }
}

TaskResult ::= {
  extends: Message,
  payload: {
    taskId: TaskID,
    status: Status,
    result: Result<Any, Error>,
    metrics: Map<String, Number>,
    resources: Map<ResourceID, Usage>
  }
}
```

Control Messages

```
SessionControl ::= {
  extends: Message,
  payload: {
    action: START | STOP | PAUSE | RESUME,
    sessionId: SessionID,
    parameters: Map<String, Any>
  }
}

ResourceRequest ::= {
  extends: Message,
  payload: {
    resourceType: String,
    quantity: Number,
    constraints: Map<String, Any>,
    priority: Priority,
    deadline: Timestamp
  }
}

StateSync ::= {
  extends: Message,
  payload: {
    state: Map<String, Any>,
    version: Number,
    timestamp: Timestamp,
    checksum: Hash
  }
}
```

5. Type Constraints

Validation Rules

1. All UUIDs must be valid RFC4122 v4
2. Timestamps must be UTC
3. Arrays must be homogeneous
4. Maps must have String keys
5. Priority must be one of defined enum values
6. Status must be one of defined enum values

Type Extensions

```
CustomType  ::= type Name extends BaseType {
    fields: Map<String, Type>,
    validators: Array<Validator>,
    serializer: Option<Function>,
    deserializer: Option<Function>
}
```