```
              "type": "string",
              "description": "IAM group name",
              "name": "group",
              "in": "path",
              "required": true
            }
          ],
          "responses": {
            "200": {
              "description": "OK",
              "schema": {
                "$ref": "#/definitions/api.IAMGroup"
              }
            },
            "400": {
              "description": "Bad Request",
              "schema": {
                "$ref": "#/definitions/server.ValidationErrorResponse"
              }
            },
            "404": {
              "description": "Not Found",
              "schema": {
                "$ref": "#/definitions/server.ValidationErrorResponse"
              }
            }
          }
        },
        "delete": {
          "produces": [
            "application/json"
          ],
          "summary": "Remove the specified user from the specified group. The group will not be
deleted.",
          "operationId": "removeIAMUserFromGroup",
          "parameters": [
            {
              "type": "string",
              "description": "IAM account ID",
              "name": "account",
              "in": "path",
              "required": true
            },
            {
              "type": "string",
              "description": "IAM user name",
              "name": "username",
              "in": "path",
              "required": true
            },
            {
              "type": "string",
              "description": "IAM group name",
              "name": "group",
              "in": "path",
              "required": true
            }
          ],
          "responses": {
            "204": {
              "description": "No Content"
```

```
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/iam/users/{account}/{username}/keys": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List IAM keys for the specified user",
      "operationId": "getIAMUserKeys",
      "parameters": [
        {
          "type": "string",
          "description": "IAM account ID",
          "name": "account",
          "in": "path",
          "required": true
        },
        {
          "type": "string",
          "description": "IAM user name",
          "name": "username",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.IAMUserKey"
            }
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    },
    "post": {
      "produces": [
        "application/json"
      ],
```

```
      "summary": "Create an IAM key for the specified user",
      "operationId": "createIAMUserKey",
      "parameters": [
        {
          "type": "string",
          "description": "IAM account ID",
          "name": "account",
          "in": "path",
          "required": true
        },
        {
          "type": "string",
          "description": "IAM user name",
          "name": "username",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/api.IAMUserKeyCreateResponse"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/iam/users/{account}/{username}/keys/{id}": {
    "delete": {
      "produces": [
        "application/json"
      ],
      "summary": "Delete an IAM key so a new key can be created",
      "operationId": "deleteIAMUserKey",
      "parameters": [
        {
          "type": "string",
          "description": "IAM account ID",
          "name": "account",
          "in": "path",
          "required": true
        },
        {
          "type": "string",
          "description": "IAM user name",
          "name": "username",
          "in": "path",
          "required": true
```

```
      },
      {
        "type": "string",
        "description": "Access key ID",
        "name": "id",
        "in": "path",
        "required": true
      }
    ],
    "responses": {
      "204": {
        "description": "No Content"
      },
      "400": {
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Modify an IAM key",
    "operationId": "updateIAMUserKey",
    "parameters": [
      {
        "type": "string",
        "description": "IAM account ID",
        "name": "account",
        "in": "path",
        "required": true
      },
      {
        "type": "string",
        "description": "IAM user name",
        "name": "username",
        "in": "path",
        "required": true
      },
      {
        "type": "string",
        "description": "Access key ID",
        "name": "id",
        "in": "path",
        "required": true
      },
      {
        "name": "body",
        "in": "body",
```

```
            "required": true,
            "schema": {
              "$ref": "#/definitions/api.IAMUserKeyUpdate"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.IAMUserKey"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/lifecycles": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "List all lifecycles",
        "operationId": "getLifecycles",
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/api.Lifecycle"
              }
            }
          }
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Create a lifecycle",
        "operationId": "createLifecycle",
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "required": true,
```

```
            "schema": {
              "$ref": "#/definitions/api.LifecycleCreate"
            }
          }
        ],
        "responses": {
          "201": {
            "description": "Created",
            "schema": {
              "$ref": "#/definitions/api.Lifecycle"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/lifecycles/{id}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "Get a lifecycle",
        "operationId": "getLifecycle",
        "parameters": [
          {
            "type": "string",
            "description": "Lifecycle ID",
            "name": "id",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.Lifecycle"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      },
      "delete": {
        "produces": [
          "application/json"
        ],
        "summary": "Delete a lifecycle",
        "operationId": "deleteLifecycle",
        "parameters": [
          {
            "type": "string",
```

```
        "description": "Lifecycle ID",
        "name": "id",
        "in": "path",
        "required": true
      }
    ],
    "responses": {
      "204": {
        "description": "No Content"
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Update a lifecycle",
    "operationId": "updateLifecycle",
    "parameters": [
      {
        "type": "string",
        "description": "Lifecycle ID",
        "name": "id",
        "in": "path",
        "required": true
      },
      {
        "name": "body",
        "in": "body",
        "required": true,
        "schema": {
          "$ref": "#/definitions/api.LifecycleUpdate"
        }
      }
    ],
    "responses": {
      "201": {
        "description": "Created",
        "schema": {
          "$ref": "#/definitions/api.Lifecycle"
        }
      },
      "400": {
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
```

```
          }
        }
      }
    }
  },
  "/sl/api/locations": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List all locations",
      "operationId": "getLocations",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.Location"
            }
          }
        }
      }
    },
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Create a location",
      "operationId": "createLocation",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.Location"
          }
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/api.Location"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/locations/{id}": {
    "delete": {
```

```
    "produces": [
      "application/json"
    ],
    "summary": "Delete a location",
    "operationId": "deleteLocation",
    "parameters": [
      {
        "type": "string",
        "description": "Location ID",
        "name": "id",
        "in": "path",
        "required": true
      }
    ],
    "responses": {
      "204": {
        "description": "No Content"
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Update a location",
    "operationId": "updateLocation",
    "parameters": [
      {
        "type": "string",
        "description": "Location ID",
        "name": "id",
        "in": "path",
        "required": true
      },
      {
        "name": "body",
        "in": "body",
        "required": true,
        "schema": {
          "$ref": "#/definitions/api.LocationUpdate"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "$ref": "#/definitions/api.Location"
        }
      },
      "400": {
        "description": "Bad Request",
```

```
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/logsets": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List all logsets stored in Vail's AWS S3 Sphere bucket",
      "operationId": "getLogsets",
      "parameters": [
        {
          "type": "string",
          "description": "Specifies the starting position",
          "name": "marker",
          "in": "query"
        },
        {
          "type": "integer",
          "description": "Maximum number of elements to return",
          "name": "max-keys",
          "in": "query"
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.Logsets"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/logsets/{key}": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get information about the specified logset, including a URL to download it",
      "operationId": "getLogset",
      "parameters": [
        {
          "type": "string",
          "description": "Logset key",
```

```
            "name": "key",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.LogsetURL"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/messages": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "Get all messages",
        "operationId": "getMessages",
        "parameters": [
          {
            "type": "string",
            "description": "Severity of message",
            "name": "severity",
            "in": "query"
          },
          {
            "type": "string",
            "format": "date-time",
            "description": "Start of date/time range",
            "name": "start",
            "in": "query"
          },
          {
            "type": "string",
            "format": "date-time",
            "description": "End of date/time range",
            "name": "end",
            "in": "query"
          },
          {
            "type": "string",
            "description": "Text to search for",
            "name": "search",
            "in": "query"
          },
```

```
      {
        "type": "boolean",
        "description": "Not setting this parameter returns both read and unread messages",
        "name": "read",
        "in": "query"
      },
      {
        "type": "string",
        "description": "Specifies the starting position",
        "name": "marker",
        "in": "query"
      },
      {
        "type": "integer",
        "description": "Maximum number of elements to return",
        "name": "max-keys",
        "in": "query"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "$ref": "#/definitions/api.Messages"
        }
      },
      "400": {
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Update all messages",
    "operationId": "updateMessages",
    "parameters": [
      {
        "name": "body",
        "in": "body",
        "required": true,
        "schema": {
          "$ref": "#/definitions/api.MessageUpdate"
        }
      },
      {
        "type": "string",
        "description": "Specifies the starting position",
        "name": "marker",
        "in": "query"
      },
      {
        "type": "integer",
        "description": "Maximum number of elements to return",
```

```
            "name": "max-keys",
            "in": "query"
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.Messages"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/messages/{id}": {
      "patch": {
        "consumes": [
          "application/merge-patch+json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Update a message",
        "operationId": "updateMessage",
        "parameters": [
          {
            "type": "string",
            "description": "Message ID",
            "name": "id",
            "in": "path",
            "required": true
          },
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/api.MessageUpdate"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.Message"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          },
          "404": {
```

```
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/network/interfaces": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get all configured network interfaces",
      "operationId": "getNetworkInterfaces",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.NetworkInterface"
            }
          }
        }
      }
    }
  },
  "/sl/api/network/interfaces/{name}": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get all properties of the specified network interface",
      "operationId": "getNetworkInterface",
      "parameters": [
        {
          "type": "string",
          "description": "Network interface name",
          "name": "name",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.NetworkInterface"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    },
    "patch": {
      "consumes": [
```

```
        "application/merge-patch+json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Update the configuration of the specified network interface",
      "operationId": "updateNetworkInterface",
      "parameters": [
        {
          "type": "string",
          "description": "Network interface name",
          "name": "name",
          "in": "path",
          "required": true
        },
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.NetworkInterfaceUpdate"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.NetworkInterface"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/performance": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get available endpoint performance tables",
      "operationId": "getPerformanceTables",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.PerformanceTable"
            }
```

```
        }
      }
    }
  }
},
"/sl/api/performance/{table}/{id}": {
  "get": {
    "produces": [
      "application/json"
    ],
    "summary": "Get performance metrics for an endpoint",
    "operationId": "getPerformanceGraph",
    "parameters": [
      {
        "type": "string",
        "description": "Performance data table",
        "name": "table",
        "in": "path",
        "required": true
      },
      {
        "type": "string",
        "description": "Endpoint ID",
        "name": "id",
        "in": "path",
        "required": true
      },
      {
        "type": "string",
        "description": "Length of time in rrd abbreviated format.",
        "name": "length",
        "in": "query"
      },
      {
        "type": "string",
        "description": "Comma-separated list of field names to include.",
        "name": "fields",
        "in": "query"
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/api.PerformanceDataset"
          }
        }
      }
    }
  }
},
"/sl/api/proxy": {
  "get": {
    "produces": [
      "application/json"
    ],
    "summary": "Get the all network proxy configurations",
    "operationId": "getProxyServers",
    "responses": {
```

```
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.HttpProxyResponse"
            }
          }
        }
      }
    },
    "/sl/api/proxy/global": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "Get the global network proxy configuration",
        "operationId": "getProxyServer",
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.HttpProxyResponse"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      },
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Create a global network proxy configuration",
        "operationId": "createProxyServer",
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/api.HttpProxy"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/api.HttpProxyResponse"
            }
          },
          "400": {
```

```
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "delete": {
    "produces": [
      "application/json"
    ],
    "summary": "Delete a network proxy configuration for both http and https",
    "operationId": "deleteProxyServer",
    "responses": {
      "204": {
        "description": "No Content"
      },
      "400": {
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Update a global network proxy configuration",
    "operationId": "updateProxyServer",
    "parameters": [
      {
        "name": "body",
        "in": "body",
        "required": true,
        "schema": {
          "$ref": "#/definitions/api.HttpProxy"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "schema": {
          "$ref": "#/definitions/api.HttpProxyResponse"
```

```
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/reports/audit": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get audit log data",
      "operationId": "getAudits",
      "parameters": [
        {
          "type": "string",
          "description": "Name of Cognito user",
          "name": "username",
          "in": "query"
        },
        {
          "type": "string",
          "format": "date-time",
          "description": "Start time",
          "name": "start",
          "in": "query"
        },
        {
          "type": "string",
          "format": "date-time",
          "description": "End time",
          "name": "end",
          "in": "query"
        },
        {
          "type": "string",
          "description": "Specifies the starting position",
          "name": "marker",
          "in": "query"
        },
        {
          "type": "integer",
          "description": "Maximum number of elements to return",
          "name": "max-keys",
          "in": "query"
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
```

```
          "schema": {
            "$ref": "#/definitions/api.Audits"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/reset/metrics": {
    "post": {
      "produces": [
        "application/json"
      ],
      "summary": "Recalculate metrics based on bucket data",
      "operationId": "trigger",
      "responses": {
        "204": {
          "description": "No Content"
        }
      }
    }
  },
  "/sl/api/s3/buckets": {
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Deprecated S3 bucket listing (use cloud/buckets instead)",
      "operationId": "getS3Buckets",
      "deprecated": true,
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.CloudBucketRequest"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
```

```
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/sphere/activate": {
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Install latest version and prepare for registration.",
      "operationId": "activateNode",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.Activate"
          }
        }
      ],
      "responses": {
        "204": {
          "description": "No Content"
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/sphere/endpoints": {
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Register this endpoint",
      "operationId": "registerNode",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.EndpointRegistration"
          }
        }
      ],
      "responses": {
        "201": {
```

```
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/handlers.NodeRegistrationInfo"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/sphere/locations": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List all locations in the given sphere",
      "operationId": "getSphereLocations",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.Location"
            }
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    },
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Create a location",
      "operationId": "createSphereLocation",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.Location"
          }
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
```

```
          "schema": {
            "$ref": "#/definitions/api.Location"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/sphere/tokens": {
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Get a token from the given sphere",
      "operationId": "createSphereToken",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/rest.Credentials"
          }
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/rest.SphereToken"
          }
        }
      }
    }
  },
  "/sl/api/storage": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List all storage",
      "operationId": "getStorages",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.Storage"
            }
          }
        }
      }
```

```
    },
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Create storage",
      "operationId": "createStorage",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.StorageCreate"
          }
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/api.Storage"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/storage/{id}": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get storage",
      "operationId": "getStorage",
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.Storage"
          }
        },
        "400": {
          "description": "Bad Request",
```

```
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "delete": {
    "produces": [
      "application/json"
    ],
    "summary": "Delete empty storage",
    "operationId": "deleteStorage",
    "parameters": [
      {
        "type": "string",
        "description": "Storage ID",
        "name": "id",
        "in": "path",
        "required": true
      }
    ],
    "responses": {
      "204": {
        "description": "No Content"
      },
      "400": {
        "description": "Bad Request",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      },
      "404": {
        "description": "Not Found",
        "schema": {
          "$ref": "#/definitions/server.ValidationErrorResponse"
        }
      }
    }
  },
  "patch": {
    "consumes": [
      "application/merge-patch+json"
    ],
    "produces": [
      "application/json"
    ],
    "summary": "Update storage",
    "operationId": "updateStorage",
    "parameters": [
      {
        "type": "string",
        "description": "Storage ID",
        "name": "id",
        "in": "path",
        "required": true
```

```
            },
            {
              "name": "body",
              "in": "body",
              "required": true,
              "schema": {
                "$ref": "#/definitions/api.StorageUpdate"
              }
            }
          ],
          "responses": {
            "200": {
              "description": "OK",
              "schema": {
                "$ref": "#/definitions/api.Storage"
              }
            },
            "400": {
              "description": "Bad Request",
              "schema": {
                "$ref": "#/definitions/server.ValidationErrorResponse"
              }
            },
            "404": {
              "description": "Not Found",
              "schema": {
                "$ref": "#/definitions/server.ValidationErrorResponse"
              }
            }
          }
        }
      },
      "/sl/api/storage/{id}/buckets": {
        "get": {
          "produces": [
            "application/json"
          ],
          "summary": "Get buckets for the given storage.  Only applicable for BlackPearl and Cloud
storage.",
          "operationId": "getStorageBuckets",
          "parameters": [
            {
              "type": "string",
              "description": "Storage ID",
              "name": "id",
              "in": "path",
              "required": true
            }
          ],
          "responses": {
            "200": {
              "description": "OK",
              "schema": {
                "type": "array",
                "items": {
                  "type": "string"
                }
              }
            },
            "400": {
              "description": "Bad Request",
```

```
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/storage/{id}/classes": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get supported storage classes for the given storage.",
      "operationId": "getStorageClasses",
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/storage/{id}/metrics": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get metrics for the given storage",
      "operationId": "getStorageMetrics",
```

```
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.Metrics"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/storage/{id}/placement": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get placement for the given storage.  Only applicable for BlackPearl.",
      "operationId": "getStorageBpPlacement",
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.BpPlacement"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        },
        "404": {
```

```
          "description": "Not Found",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/storage/{id}/sync": {
    "post": {
      "produces": [
        "application/json"
      ],
      "summary": "Synchronize the given storage.  Only applicable to linked bucket storage.",
      "operationId": "syncStorageBucket",
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        },
        {
          "type": "string",
          "description": "Process this object instead of the entire bucket.",
          "name": "object",
          "in": "query"
        }
      ],
      "responses": {
        "204": {
          "description": "No Content"
        }
      }
    }
  },
  "/sl/api/storage/{id}/verify": {
    "put": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Trigger verification of clones stored on the given storage.",
      "operationId": "verifyStorage",
      "parameters": [
        {
          "type": "string",
          "description": "Storage ID",
          "name": "id",
          "in": "path",
          "required": true
        },
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.StorageVerification"
```

```
            }
          }
        ],
        "responses": {
          "202": {
            "description": "Accepted"
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      },
      "delete": {
        "produces": [
          "application/json"
        ],
        "summary": "Cancel verification of clones stored on the given storage.",
        "operationId": "cancelVerifyStorage",
        "parameters": [
          {
            "type": "string",
            "description": "Storage ID",
            "name": "id",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "202": {
            "description": "Accepted"
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          },
          "404": {
            "description": "Not Found",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/summary": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "Get Summary Data",
```

```
      "operationId": "getSummary",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.Summary"
          }
        }
      }
    }
  },
  "/sl/api/system": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get information about this system.",
      "operationId": "getSystem",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.System"
          }
        }
      }
    },
    "patch": {
      "consumes": [
        "application/merge-patch+json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Update the system",
      "operationId": "updateSystem",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/api.SystemUpdate"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/api.System"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
```

```
    },
    "/sl/api/system/services": {
      "get": {
        "produces": [
          "application/json"
        ],
        "summary": "Get diagnostics for required services.",
        "operationId": "getServices",
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/api.Service"
              }
            }
          }
        }
      }
    },
    "/sl/api/tokens": {
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Create a JSON Web Token. Create a credential if invalid key/credential
detected.",
        "operationId": "createTokenCredential",
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/rest.Credentials"
            }
          }
        ],
        "responses": {
          "201": {
            "description": "Created",
            "schema": {
              "$ref": "#/definitions/rest.Token"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/updates/endpoints": {
      "get": {
        "produces": [
```

```
        "application/json"
      ],
      "summary": "Returns a list of endpoints and the version each one can update to.",
      "operationId": "getEndpointUpdateVersions",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/*api.EndpointUpdateVersion"
            }
          }
        }
      }
    }
  },
  "/sl/api/updates/vmdk-url": {
    "post": {
      "produces": [
        "application/json"
      ],
      "summary": "Create presigned download URL for the latest available VMDK.",
      "operationId": "createVMDownloadUrl",
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/handlers.PresignedS3Request"
          }
        }
      }
    }
  },
  "/sl/api/usage/cloud/summary": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get cloud usage summary",
      "operationId": "getCloudUsageSummary",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.StorageUsed"
            }
          }
        }
      }
    }
  },
  "/sl/api/usage/entities": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get usage summary for all independent storage entities",
      "operationId": "getEntityUsageSummary",
```

```
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.StorageUsed"
            }
          }
        }
      }
    }
  },
  "/sl/api/usage/locations": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get usage summary for on-premise locations (excludes cloud storage)",
      "operationId": "getLocationUsageSummary",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.StorageUsed"
            }
          }
        }
      }
    }
  },
  "/sl/api/usage/sphere/summary": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "Get sphere on-premise usage summary",
      "operationId": "getSphereUsageSummary",
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.StorageUsed"
            }
          }
        }
      }
    }
  },
  "/sl/api/users": {
    "get": {
      "produces": [
        "application/json"
      ],
      "summary": "List all users",
      "operationId": "getCognitoUsers",
      "responses": {
```

```
        "200": {
          "description": "OK",
          "schema": {
            "$ref": "#/definitions/users.UserCollection"
          }
        }
      }
    },
    "post": {
      "consumes": [
        "application/json"
      ],
      "produces": [
        "application/json"
      ],
      "summary": "Request to create a new user",
      "operationId": "createCognitoUser",
      "parameters": [
        {
          "name": "body",
          "in": "body",
          "required": true,
          "schema": {
            "$ref": "#/definitions/users.User"
          }
        }
      ],
      "responses": {
        "201": {
          "description": "Created",
          "schema": {
            "$ref": "#/definitions/users.User"
          }
        },
        "400": {
          "description": "Bad Request",
          "schema": {
            "$ref": "#/definitions/server.ValidationErrorResponse"
          }
        }
      }
    }
  },
  "/sl/api/users/{username}": {
    "delete": {
      "produces": [
        "application/json"
      ],
      "summary": "Delete's a Cognito user",
      "operationId": "deleteCognitoUser",
      "parameters": [
        {
          "type": "string",
          "description": "User's username",
          "name": "username",
          "in": "path",
          "required": true
        }
      ],
      "responses": {
        "204": {
```

```
            "description": "No Content"
          }
        }
      },
      "patch": {
        "consumes": [
          "application/merge-patch+json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Update a user",
        "operationId": "updateCognitoUser",
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/users.UserPatch"
            }
          },
          {
            "type": "string",
            "description": "User's username",
            "name": "username",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/users.User"
            }
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/users/{username}/forgot_password": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Reset the user's password",
        "operationId": "confirmResetCognitoUserPassword",
        "parameters": [
          {
            "type": "string",
            "description": "User's username",
            "name": "username",
```

```
            "in": "path",
            "required": true
          },
          {
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/api.CognitoUserPasswordReset"
            }
          }
        ],
        "responses": {
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      },
      "post": {
        "produces": [
          "application/json"
        ],
        "summary": "Request a password reset",
        "operationId": "resetCognitoUserPassword",
        "parameters": [
          {
            "type": "string",
            "description": "User's username",
            "name": "username",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "204": {
            "description": "No Content"
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    },
    "/sl/api/users/{username}/password": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "summary": "Change a user's password",
        "operationId": "updateCognitoUserPassword",
        "parameters": [
          {
```

```
            "name": "body",
            "in": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/api.CognitoUserPasswordUpdate"
            }
          },
          {
            "type": "string",
            "description": "User's username",
            "name": "username",
            "in": "path",
            "required": true
          }
        ],
        "responses": {
          "204": {
            "description": "No Content"
          },
          "400": {
            "description": "Bad Request",
            "schema": {
              "$ref": "#/definitions/server.ValidationErrorResponse"
            }
          }
        }
      }
    }
  },
  "definitions": {
    "*api.EndpointUpdateVersion": {},
    "api.ACL": {
      "required": [
        "type",
        "id"
      ],
      "properties": {
        "id": {
          "description": "Canonical user ID or group URI",
          "type": "string"
        },
        "read": {
          "description": "Grant READ permission",
          "type": "boolean"
        },
        "readACP": {
          "description": "Grant READ_ACP permission",
          "type": "boolean"
        },
        "type": {
          "type": "string",
          "enum": [
            "CanonicalUser",
            "Group"
          ]
        },
        "write": {
          "description": "Grant WRITE permission",
          "type": "boolean"
        },
        "writeACP": {
```

```
            "description": "Grant WRITE_ACP permission",
            "type": "boolean"
          }
        }
      },
      "api.Account": {
        "properties": {
          "canonicalId": {
            "description": "AWS canonical ID",
            "type": "string",
            "readOnly": true
          },
          "default": {
            "description": "True if this is the default bucket owner account",
            "type": "boolean",
            "readOnly": true
          },
          "description": {
            "description": "The AWS Account's description.",
            "type": "string"
          },
          "email": {
            "description": "The user's email associated with the AWS Account.",
            "type": "string"
          },
          "externalId": {
            "description": "AWS role external ID",
            "type": "string"
          },
          "id": {
            "description": "AWS account ID",
            "type": "string",
            "readOnly": true
          },
          "roleArn": {
            "description": "AWS role ARN",
            "type": "string"
          },
          "username": {
            "description": "The user's name associated with the AWS Account.",
            "type": "string",
            "readOnly": true
          }
        }
      },
      "api.AccountUpdate": {
        "properties": {
          "description": {
            "description": "The AWS Account's description.",
            "type": "string"
          },
          "email": {
            "description": "The user's email associated with the AWS Account.",
            "type": "string"
          },
          "externalId": {
            "description": "AWS role external ID. This can only be updated if the account has a role
set.",
            "type": "string"
          },
          "roleArn": {
            "description": "AWS role ARN. This can only be updated if the account originally had a
```

```
role set.",
            "type": "string"
          }
        }
      },
      "api.Activate": {
        "required": [
          "key"
        ],
        "properties": {
          "key": {
            "description": "Sphere activation key",
            "type": "string"
          },
          "url": {
            "description": "Alternate activation endpoint",
            "type": "string"
          }
        }
      },
      "api.Audit": {
        "required": [
          "request",
          "server",
          "resource",
          "message",
          "nodeID",
          "hostIP",
          "clientIP",
          "username"
        ],
        "properties": {
          "clientIP": {
            "type": "string"
          },
          "hostIP": {
            "type": "string"
          },
          "id": {
            "description": "Audit identifier",
            "type": "string",
            "readOnly": true
          },
          "message": {
            "type": "string"
          },
          "nodeID": {
            "type": "string"
          },
          "request": {
            "$ref": "#/definitions/api.AuditRequest"
          },
          "resource": {
            "$ref": "#/definitions/api.AuditResource"
          },
          "server": {
            "type": "string"
          },
          "username": {
            "type": "string"
          }
```

```
    }
  },
  "api.AuditRequest": {
    "required": [
      "path",
      "method",
      "time"
    ],
    "properties": {
      "data": {
        "type": "object"
      },
      "method": {
        "type": "string"
      },
      "path": {
        "type": "string"
      },
      "time": {
        "type": "string",
        "format": "date-time"
      }
    }
  },
  "api.AuditResource": {
    "required": [
      "id",
      "name"
    ],
    "properties": {
      "id": {
        "type": "string"
      },
      "name": {
        "type": "string"
      }
    }
  },
  "api.Audits": {
    "required": [
      "data"
    ],
    "properties": {
      "data": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.Audit"
        }
      },
      "isTruncated": {
        "description": "Returns true if list was truncated",
        "type": "boolean",
        "readOnly": true
      },
      "marker": {
        "description": "The marker specified in the request",
        "type": "string",
        "readOnly": true
      },
      "maxKeys": {
        "description": "The maximum number of keys specified in the request",
```

```
        "type": "integer",
        "format": "int32",
        "readOnly": true
      },
      "nextMarker": {
        "description": "The starting ID of the next page",
        "type": "string",
        "readOnly": true
      }
    }
  },
  "api.BlackPearlStatus": {
    "required": [
      "name",
      "serialNumber",
      "cacheUsed",
      "cacheTotal",
      "databaseUsed",
      "databaseTotal",
      "buckets",
      "activeJobs",
      "capacitySummary"
    ],
    "properties": {
      "activeJobs": {
        "description": "Active jobs on the BlackPearl",
        "type": "integer",
        "format": "int64"
      },
      "buckets": {
        "description": "Total BlackPearl buckets",
        "type": "integer",
        "format": "int64"
      },
      "cacheTotal": {
        "description": "Cache capacity in bytes",
        "type": "integer",
        "format": "int64"
      },
      "cacheUsed": {
        "description": "Cache used in bytes",
        "type": "integer",
        "format": "int64"
      },
      "capacitySummary": {
        "description": "Capacity summaries",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.CapacitySummary"
        }
      },
      "databaseTotal": {
        "description": "Database capacity in bytes",
        "type": "integer",
        "format": "int64"
      },
      "databaseUsed": {
        "description": "Database used in bytes",
        "type": "integer",
        "format": "int64"
      },
```

```
          "id": {
            "description": "BlackPearl identifier",
            "type": "string",
            "readOnly": true
          },
          "name": {
            "description": "BlackPearl name",
            "type": "string"
          },
          "serialNumber": {
            "description": "BlackPearl Serial Number",
            "type": "string"
          }
        }
      },
      "api.BpPlacement": {
        "required": [
          "partitions"
        ],
        "properties": {
          "id": {
            "description": "Storage identifier",
            "type": "string",
            "readOnly": true
          },
          "partitions": {
            "description": "The partitions in the bucket data policy",
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        }
      },
      "api.Bucket": {
        "required": [
          "name",
          "created"
        ],
        "properties": {
          "acls": {
            "description": "User and group ACLs",
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.ACL"
            }
          },
          "blockPublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
blocked",
            "type": "boolean"
          },
          "blockPublicPolicy": {
            "description": "True if public policies for this bucket are blocked",
            "type": "boolean"
          },
          "compress": {
            "description": "True if compression should be attempted for all clones of objects stored
in the bucket",
            "type": "boolean",
            "default": true
```

```
          },
          "control": {
            "description": "Bucket ownership control",
            "type": "string",
            "enum": [
              "ObjectWriter",
              "BucketOwnerPreferred",
              "BucketOwnerEnforced"
            ]
          },
          "created": {
            "description": "A timestamp of when the bucket was created.",
            "type": "string",
            "format": "date-time"
          },
          "defaultRetention": {
            "$ref": "#/definitions/api.Retention"
          },
          "encrypt": {
            "description": "True if contents are encrypted",
            "type": "boolean"
          },
          "ignorePublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
ignored",
            "type": "boolean"
          },
          "lifecycle": {
            "description": "The ID of the bucket's lifecycle",
            "type": "string"
          },
          "linkedStorage": {
            "description": "Cloud or BlackPearl storage to use for linking",
            "type": "string"
          },
          "locking": {
            "description": "True if object locking is enabled",
            "type": "boolean"
          },
          "name": {
            "description": "The bucket's name",
            "type": "string",
            "uniqueItems": true
          },
          "owner": {
            "description": "The bucket's owner",
            "type": "string"
          },
          "policy": {
            "$ref": "#/definitions/api.Policy"
          },
          "restore": {
            "description": "True if automatic restore is enabled",
            "type": "boolean"
          },
          "restrictPublicBuckets": {
            "description": "True if public policies for this bucket are restricted",
            "type": "boolean"
          },
          "versioning": {
            "description": "Bucket versioning status",
```

```
            "type": "string",
            "enum": [
              "Enabled",
              "Suspended"
            ]
          }
        }
      },
      "api.BucketCommon": {
        "properties": {
          "acls": {
            "description": "User and group ACLs",
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.ACL"
            }
          },
          "blockPublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
blocked",
            "type": "boolean"
          },
          "blockPublicPolicy": {
            "description": "True if public policies for this bucket are blocked",
            "type": "boolean"
          },
          "compress": {
            "description": "True if compression should be attempted for all clones of objects stored
in the bucket",
            "type": "boolean",
            "default": true
          },
          "control": {
            "description": "Bucket ownership control",
            "type": "string",
            "enum": [
              "ObjectWriter",
              "BucketOwnerPreferred",
              "BucketOwnerEnforced"
            ]
          },
          "defaultRetention": {
            "$ref": "#/definitions/api.Retention"
          },
          "encrypt": {
            "description": "True if contents are encrypted",
            "type": "boolean"
          },
          "ignorePublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
ignored",
            "type": "boolean"
          },
          "lifecycle": {
            "description": "The ID of the bucket's lifecycle",
            "type": "string"
          },
          "locking": {
            "description": "True if object locking is enabled",
            "type": "boolean"
          },
```

```
          "owner": {
            "description": "The bucket's owner",
            "type": "string"
          },
          "policy": {
            "$ref": "#/definitions/api.Policy"
          },
          "restore": {
            "description": "True if automatic restore is enabled",
            "type": "boolean"
          },
          "restrictPublicBuckets": {
            "description": "True if public policies for this bucket are restricted",
            "type": "boolean"
          },
          "versioning": {
            "description": "Bucket versioning status",
            "type": "string",
            "enum": [
              "Enabled",
              "Suspended"
            ]
          }
        }
      },
      "api.BucketCreate": {
        "required": [
          "name"
        ],
        "properties": {
          "acls": {
            "description": "User and group ACLs",
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.ACL"
            }
          },
          "blockPublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
blocked",
            "type": "boolean"
          },
          "blockPublicPolicy": {
            "description": "True if public policies for this bucket are blocked",
            "type": "boolean"
          },
          "compress": {
            "description": "True if compression should be attempted for all clones of objects stored
in the bucket",
            "type": "boolean",
            "default": true
          },
          "control": {
            "description": "Bucket ownership control",
            "type": "string",
            "enum": [
              "ObjectWriter",
              "BucketOwnerPreferred",
              "BucketOwnerEnforced"
            ]
          },
```

```
          "defaultRetention": {
            "$ref": "#/definitions/api.Retention"
          },
          "encrypt": {
            "description": "True if contents are encrypted",
            "type": "boolean"
          },
          "ignorePublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
ignored",
            "type": "boolean"
          },
          "lifecycle": {
            "description": "The ID of the bucket's lifecycle",
            "type": "string"
          },
          "locking": {
            "description": "True if object locking is enabled",
            "type": "boolean"
          },
          "name": {
            "description": "The bucket's name",
            "type": "string",
            "uniqueItems": true
          },
          "owner": {
            "description": "The bucket's owner",
            "type": "string"
          },
          "policy": {
            "$ref": "#/definitions/api.Policy"
          },
          "restore": {
            "description": "True if automatic restore is enabled",
            "type": "boolean"
          },
          "restrictPublicBuckets": {
            "description": "True if public policies for this bucket are restricted",
            "type": "boolean"
          },
          "versioning": {
            "description": "Bucket versioning status",
            "type": "string",
            "enum": [
              "Enabled",
              "Suspended"
            ]
          }
        }
      },
      "api.BucketOwner": {
        "required": [
          "id",
          "name"
        ],
        "properties": {
          "id": {
            "description": "Cloud bucket owner identifier",
            "type": "string"
          },
          "name": {
```

```
            "description": "Cloud bucket owner name",
            "type": "string"
          }
        }
      },
      "api.BucketUpdate": {
        "properties": {
          "acls": {
            "description": "User and group ACLs",
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.ACL"
            }
          },
          "blockPublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
blocked",
            "type": "boolean"
          },
          "blockPublicPolicy": {
            "description": "True if public policies for this bucket are blocked",
            "type": "boolean"
          },
          "compress": {
            "description": "True if compression should be attempted for all clones of objects stored
in the bucket",
            "type": "boolean",
            "default": true
          },
          "control": {
            "description": "Bucket ownership control",
            "type": "string",
            "enum": [
              "ObjectWriter",
              "BucketOwnerPreferred",
              "BucketOwnerEnforced"
            ]
          },
          "defaultRetention": {
            "$ref": "#/definitions/api.Retention"
          },
          "encrypt": {
            "description": "True if contents are encrypted",
            "type": "boolean"
          },
          "ignorePublicAcls": {
            "description": "True if public access control lists for this bucket and its objects are
ignored",
            "type": "boolean"
          },
          "lifecycle": {
            "description": "The ID of the bucket's lifecycle",
            "type": "string"
          },
          "locking": {
            "description": "True if object locking is enabled",
            "type": "boolean"
          },
          "owner": {
            "description": "The bucket's owner",
            "type": "string"
```

```
      },
      "policy": {
        "$ref": "#/definitions/api.Policy"
      },
      "restore": {
        "description": "True if automatic restore is enabled",
        "type": "boolean"
      },
      "restrictPublicBuckets": {
        "description": "True if public policies for this bucket are restricted",
        "type": "boolean"
      },
      "versioning": {
        "description": "Bucket versioning status",
        "type": "string",
        "enum": [
          "Enabled",
          "Suspended"
        ]
      }
    }
  },
  "api.CapacitySummary": {
    "required": [
      "type",
      "allocated",
      "used",
      "total"
    ],
    "properties": {
      "allocated": {
        "description": "Bytes Allocated",
        "type": "integer",
        "format": "int64"
      },
      "total": {
        "description": "Bytes Total",
        "type": "integer",
        "format": "int64"
      },
      "type": {
        "description": "Storage Type",
        "type": "string"
      },
      "used": {
        "description": "Bytes Used",
        "type": "integer",
        "format": "int64"
      }
    }
  },
  "api.Certificate": {
    "required": [
      "issuer",
      "subject",
      "notBefore",
      "notAfter"
    ],
    "properties": {
      "issuer": {
        "description": "The Issuer of the Certificate",
```

```
        "type": "string"
      },
      "notAfter": {
        "description": "A timestamp of the expiration date for the certificate",
        "type": "string",
        "format": "date-time"
      },
      "notBefore": {
        "description": "A timestamp of the start date for the certificate to be valid",
        "type": "string",
        "format": "date-time"
      },
      "subject": {
        "description": "The Subject of the Certificate",
        "type": "string"
      }
    }
  },
  "api.CertificateUpdate": {
    "required": [
      "certPEM",
      "privateKeyPEM",
      "passphrase"
    ],
    "properties": {
      "certPEM": {
        "description": "The Certificate PEM",
        "type": "string"
      },
      "passphrase": {
        "description": "The Passphrase for the Encrypted Private Key",
        "type": "string"
      },
      "privateKeyPEM": {
        "description": "The Private Key PEM",
        "type": "string"
      }
    }
  },
  "api.CloneState": {
    "properties": {
      "processing": {
        "description": "Lifecycle processing is active",
        "type": "boolean"
      },
      "restoring": {
        "description": "A restore request is active",
        "type": "boolean"
      },
      "scheduled": {
        "description": "Scheduled copy time",
        "type": "string",
        "format": "date-time"
      },
      "storage": {
        "description": "Storage clones",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.StorageClone"
        },
        "readOnly": true
```

```
          }
        }
      },
      "api.CloudBucketRequest": {
        "properties": {
          "accessKey": {
            "description": "The account owner's access key, or the Azure storage account",
            "type": "string"
          },
          "arn": {
            "description": "The IAM role arn to use to access the account.  This can be used as an
alternative to providing accessKey and secretKey.",
            "type": "string"
          },
          "cloudProvider": {
            "description": "Provider of cloud services (if applicable)",
            "type": "string",
            "enum": [
              "aws",
              "other",
              "azure",
              "google"
            ]
          },
          "credentials": {
            "description": "Credentials as a single element (e.g. Google JSON file)",
            "type": "string"
          },
          "externalid": {
            "description": "The IAM role's external ID.S",
            "type": "string"
          },
          "region": {
            "description": "The region where the account resides",
            "type": "string"
          },
          "secretKey": {
            "description": "The account owner's secret key",
            "type": "string"
          },
          "url": {
            "description": "S3 data path URL (if applicable)",
            "type": "string"
          }
        }
      },
      "api.CloudBucketUpdate": {
        "properties": {
          "accessKey": {
            "description": "The account owner's access key, or the Azure storage account",
            "type": "string"
          },
          "arn": {
            "description": "The IAM role arn to use to access the account.  This can be used as an
alternative to providing accessKey and secretKey.",
            "type": "string"
          },
          "externalid": {
            "description": "The IAM role's external ID.",
            "type": "string"
          },
```

```
      "secretKey": {
        "description": "The account owner's secret key",
        "type": "string"
      }
    }
  },
  "api.CognitoUserPasswordReset": {
    "required": [
      "confirmationCode",
      "password"
    ],
    "properties": {
      "confirmationCode": {
        "description": "The confirmation code in the password reset email",
        "type": "string"
      },
      "password": {
        "description": "The new password",
        "type": "string"
      }
    }
  },
  "api.CognitoUserPasswordUpdate": {
    "required": [
      "password"
    ],
    "properties": {
      "password": {
        "description": "The new password",
        "type": "string"
      }
    }
  },
  "api.DeleteStatusField": {
    "properties": {
      "status": {
        "description": "Status can be set to deleting to begin background deletion",
        "type": "string",
        "enum": [
          "deleting"
        ]
      }
    }
  },
  "api.Destinations": {
    "required": [
      "storage"
    ],
    "properties": {
      "count": {
        "description": "The number of required destinations. No count means all destinations.",
        "type": "integer",
        "format": "int32"
      },
      "storage": {
        "description": "The list of destination storage.",
        "type": "array",
        "items": {
          "type": "string"
        }
      }
```

```
    }
  },
  "api.Endpoint": {
    "required": [
      "id",
      "type",
      "location",
      "url"
    ],
    "properties": {
      "debug": {
        "description": "debug level (0-9)",
        "type": "integer",
        "format": "int32"
      },
      "hosts": {
        "description": "additional supported hostnames",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "id": {
        "description": "Endpoint identifier",
        "type": "string"
      },
      "location": {
        "description": "ID of physical location",
        "type": "string"
      },
      "managementURL": {
        "description": "URL for DS3 system management (if available)",
        "type": "string"
      },
      "name": {
        "description": "Name of endpoint (hostname is default).",
        "type": "string"
      },
      "profiling": {
        "description": "profiling configuration",
        "$ref": "#/definitions/api.Profiling"
      },
      "status": {
        "description": "Status",
        "type": "string",
        "enum": [
          "ok",
          "updating",
          "degraded",
          "deleting",
          "unavailable"
        ],
        "readOnly": true
      },
      "type": {
        "description": "Type of system running at this endpoint",
        "type": "string",
        "enum": [
          "sphere",
          "bp",
          "vm"
```

```
          ]
        },
        "url": {
          "description": "endpoint S3 URL",
          "type": "string"
        },
        "version": {
          "description": "current version",
          "type": "string"
        }
      }
    },
    "api.EndpointRegistration": {
      "required": [
        "location"
      ],
      "properties": {
        "location": {
          "description": "ID of physical location",
          "type": "string"
        },
        "name": {
          "description": "Name of endpoint (hostname is default).",
          "type": "string"
        },
        "version": {
          "description": "current version",
          "type": "string"
        }
      }
    },
    "api.EndpointStatusField": {
      "properties": {
        "status": {
          "description": "Status",
          "type": "string",
          "enum": [
            "ok",
            "updating",
            "degraded",
            "deleting",
            "unavailable"
          ],
          "readOnly": true
        }
      }
    },
    "api.EndpointUpdate": {
      "properties": {
        "debug": {
          "description": "debug level (0-9)",
          "type": "integer",
          "format": "int32"
        },
        "hosts": {
          "description": "additional supported hostnames",
          "type": "array",
          "items": {
            "type": "string"
          }
        },
```

```
      "location": {
        "description": "ID of physical location",
        "type": "string"
      },
      "profiling": {
        "description": "profiling configuration",
        "$ref": "#/definitions/api.Profiling"
      }
    }
  },
  "api.Geocode": {
    "required": [
      "id",
      "displayName",
      "name",
      "latitude",
      "longitude"
    ],
    "properties": {
      "displayName": {
        "description": "Display name of place (intended for search results)",
        "type": "string"
      },
      "id": {
        "description": "Place identifier",
        "type": "integer",
        "format": "int64"
      },
      "latitude": {
        "description": "Latitude of place",
        "type": "number",
        "format": "double"
      },
      "longitude": {
        "description": "Longitude of place",
        "type": "number",
        "format": "double"
      },
      "name": {
        "description": "Name of place",
        "type": "string"
      }
    }
  },
  "api.HttpProxy": {
    "required": [
      "hostname",
      "port"
    ],
    "properties": {
      "hostname": {
        "description": "Proxy server hostname",
        "type": "string"
      },
      "password": {
        "description": "Proxy server password",
        "type": "string"
      },
      "port": {
        "description": "Proxy server port",
        "type": "integer",
```

```
          "format": "int32"
        },
        "username": {
          "description": "Proxy server username",
          "type": "string"
        }
      }
    },
    "api.HttpProxyResponse": {
      "required": [
        "id",
        "hostname",
        "port"
      ],
      "properties": {
        "hostname": {
          "description": "Proxy server hostname",
          "type": "string"
        },
        "id": {
          "description": "Proxy type",
          "type": "string"
        },
        "password": {
          "description": "Proxy server password",
          "type": "string"
        },
        "port": {
          "description": "Proxy server port",
          "type": "integer",
          "format": "int32"
        },
        "username": {
          "description": "Proxy server username",
          "type": "string"
        }
      }
    },
    "api.IAMGroup": {
      "required": [
        "name",
        "accountid"
      ],
      "properties": {
        "accountid": {
          "description": "The AWS Account the IAM group belongs to.",
          "type": "string"
        },
        "name": {
          "description": "The group's name",
          "type": "string"
        }
      }
    },
    "api.IAMGroups": {
      "required": [
        "data"
      ],
      "properties": {
        "data": {
          "type": "array",
```

```
            "items": {
              "$ref": "#/definitions/api.IAMGroup"
            }
          }
        }
      },
      "api.IAMUser": {
        "required": [
          "username",
          "accountid"
        ],
        "properties": {
          "accountid": {
            "description": "The AWS Account the IAM user belongs to.",
            "type": "string"
          },
          "username": {
            "description": "The user's username",
            "type": "string"
          }
        }
      },
      "api.IAMUserKey": {
        "required": [
          "id"
        ],
        "properties": {
          "id": {
            "description": "The access key ID",
            "type": "string"
          },
          "inactive": {
            "description": "Indicates the key has been disabled.",
            "type": "boolean"
          },
          "initialized": {
            "description": "Indicates the key's secret has been provided, making it available for
use.'",
            "type": "boolean"
          }
        }
      },
      "api.IAMUserKeyCreateResponse": {
        "required": [
          "id",
          "secretAccessKey"
        ],
        "properties": {
          "id": {
            "description": "The access key ID",
            "type": "string"
          },
          "inactive": {
            "description": "Indicates the key has been disabled.",
            "type": "boolean"
          },
          "initialized": {
            "description": "Indicates the key's secret has been provided, making it available for
use.'",
            "type": "boolean"
          },
```

```
          "secretAccessKey": {
            "description": "The created secret access key. This can only be seen when initially
creating the key. It cannot be recovered later.",
            "type": "string"
          }
        }
      },
      "api.IAMUserKeyUpdate": {
        "properties": {
          "inactive": {
            "description": "True if the key is to be disabled.",
            "type": "boolean"
          },
          "secretAccessKey": {
            "description": "The current value of the secret access key on AWS.",
            "type": "string"
          }
        }
      },
      "api.IAMUsers": {
        "required": [
          "data"
        ],
        "properties": {
          "data": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.IAMUser"
            }
          }
        }
      },
      "api.KeyCredentials": {
        "required": [
          "key",
          "secret",
          "token",
          "expiration"
        ],
        "properties": {
          "expiration": {
            "description": "When temporary credential expires",
            "type": "string",
            "format": "date-time"
          },
          "key": {
            "description": "Temporary access key",
            "type": "string"
          },
          "secret": {
            "description": "Temporary secret",
            "type": "string"
          },
          "token": {
            "description": "Temporary token",
            "type": "string"
          }
        }
      },
      "api.Lifecycle": {
        "required": [
```

```
        "id",
        "name",
        "modified"
      ],
      "properties": {
        "description": {
          "description": "The lifecycle's description",
          "type": "string"
        },
        "id": {
          "description": "Lifecycle identifier",
          "type": "string"
        },
        "linkedStorage": {
          "description": "ID of any linked Cloud or BlackPearl storage used as a destination",
          "type": "string"
        },
        "markers": {
          "description": "True if expired delete markers should be deleted",
          "type": "boolean"
        },
        "modified": {
          "description": "The last modified date",
          "type": "string",
          "format": "date-time"
        },
        "name": {
          "description": "The lifecycle's name",
          "type": "string"
        },
        "rules": {
          "description": "The lifecycle's rules",
          "type": "array",
          "items": {
            "$ref": "#/definitions/api.Rule"
          }
        },
        "uploads": {
          "description": "The number of days to wait before deleting incomplete multipart uploads",
          "type": "integer",
          "format": "int32"
        }
      }
    },
    "api.LifecycleCommon": {
      "properties": {
        "description": {
          "description": "The lifecycle's description",
          "type": "string"
        },
        "markers": {
          "description": "True if expired delete markers should be deleted",
          "type": "boolean"
        },
        "rules": {
          "description": "The lifecycle's rules",
          "type": "array",
          "items": {
            "$ref": "#/definitions/api.Rule"
          }
        },
```

```
      "uploads": {
        "description": "The number of days to wait before deleting incomplete multipart uploads",
        "type": "integer",
        "format": "int32"
      }
    }
  },
  "api.LifecycleCreate": {
    "required": [
      "name"
    ],
    "properties": {
      "description": {
        "description": "The lifecycle's description",
        "type": "string"
      },
      "markers": {
        "description": "True if expired delete markers should be deleted",
        "type": "boolean"
      },
      "name": {
        "description": "The lifecycle's name",
        "type": "string"
      },
      "rules": {
        "description": "The lifecycle's rules",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.Rule"
        }
      },
      "uploads": {
        "description": "The number of days to wait before deleting incomplete multipart uploads",
        "type": "integer",
        "format": "int32"
      }
    }
  },
  "api.LifecycleUpdate": {
    "properties": {
      "description": {
        "description": "The lifecycle's description",
        "type": "string"
      },
      "markers": {
        "description": "True if expired delete markers should be deleted",
        "type": "boolean"
      },
      "name": {
        "description": "The lifecycle's name",
        "type": "string"
      },
      "rules": {
        "description": "The lifecycle's rules",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.Rule"
        }
      },
      "uploads": {
        "description": "The number of days to wait before deleting incomplete multipart uploads",
```

```
        "type": "integer",
        "format": "int32"
      }
    }
  },
  "api.ListObjectsResult": {
    "required": [
      "name",
      "maxKeys"
    ],
    "properties": {
      "commonPrefixes": {
        "description": "If a delimiter is specified, this is the analogue of folders",
        "type": "array",
        "items": {
          "$ref": "#/definitions/worker.CommonPrefixResult"
        }
      },
      "contents": {
        "description": "List of object versions",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.Object"
        }
      },
      "delimiter": {
        "description": "Delimiter used in the query",
        "type": "string"
      },
      "isTruncated": {
        "description": "Indicates there are additional entries",
        "type": "boolean"
      },
      "marker": {
        "description": "Marker specified in the query",
        "type": "string"
      },
      "maxKeys": {
        "description": "Maximum number of entries returned in a page",
        "type": "integer",
        "format": "int32"
      },
      "name": {
        "description": "Bucket name",
        "type": "string"
      },
      "nextMarker": {
        "description": "Marker to use for next page (if isTruncated)",
        "type": "string"
      },
      "nextVersionIDMarker": {
        "description": "Version ID narker to use for next page (if isTruncated)",
        "type": "string"
      },
      "prefix": {
        "description": "Prefix used in the query",
        "type": "string"
      },
      "versionIDMarker": {
        "description": "Version ID marker specified in the query",
        "type": "string"
```

```
      },
      "versions": {
        "description": "Indicates version data is present",
        "type": "boolean"
      }
    }
  },
  "api.Location": {
    "required": [
      "name"
    ],
    "properties": {
      "id": {
        "description": "Location identifier",
        "type": "string",
        "readOnly": true
      },
      "latitude": {
        "description": "Location latitude",
        "type": "number",
        "format": "double"
      },
      "longitude": {
        "description": "Location longitude",
        "type": "number",
        "format": "double"
      },
      "name": {
        "description": "Location name",
        "type": "string"
      },
      "status": {
        "description": "Status",
        "type": "string",
        "enum": [
          "ok",
          "updating",
          "degraded",
          "unavailable"
        ],
        "readOnly": true
      }
    }
  },
  "api.LocationStatusField": {
    "properties": {
      "status": {
        "description": "Status",
        "type": "string",
        "enum": [
          "ok",
          "updating",
          "degraded",
          "unavailable"
        ],
        "readOnly": true
      }
    }
  },
  "api.LocationUpdate": {
    "properties": {
```

```
        "latitude": {
          "description": "Location latitude",
          "type": "number",
          "format": "double"
        },
        "longitude": {
          "description": "Location longitude",
          "type": "number",
          "format": "double"
        },
        "name": {
          "description": "Location name",
          "type": "string"
        }
      }
    },
    "api.Logset": {
      "required": [
        "key",
        "created"
      ],
      "properties": {
        "created": {
          "description": "When the logset was created",
          "type": "string",
          "format": "date-time"
        },
        "key": {
          "description": "The logset's name'",
          "type": "string"
        }
      }
    },
    "api.LogsetURL": {
      "required": [
        "key",
        "created",
        "url"
      ],
      "properties": {
        "created": {
          "description": "When the logset was created",
          "type": "string",
          "format": "date-time"
        },
        "key": {
          "description": "The logset's name'",
          "type": "string"
        },
        "url": {
          "description": "URL to download the logset file.",
          "type": "string"
        }
      }
    },
    "api.Logsets": {
      "required": [
        "data"
      ],
      "properties": {
        "data": {
```

```
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.Logset"
        }
      },
      "isTruncated": {
        "description": "Returns true if list was truncated",
        "type": "boolean",
        "readOnly": true
      },
      "marker": {
        "description": "The marker specified in the request",
        "type": "string",
        "readOnly": true
      },
      "maxKeys": {
        "description": "The maximum number of keys specified in the request",
        "type": "integer",
        "format": "int32",
        "readOnly": true
      },
      "nextMarker": {
        "description": "The starting ID of the next page",
        "type": "string",
        "readOnly": true
      }
    }
  },
  "api.Message": {
    "required": [
      "severity",
      "text"
    ],
    "properties": {
      "id": {
        "description": "Message identifier",
        "type": "string",
        "readOnly": true
      },
      "key": {
        "description": "The untranslated message key",
        "type": "string",
        "readOnly": true
      },
      "params": {
        "description": "The message parameters",
        "type": "object",
        "additionalProperties": {
          "type": "string"
        }
      },
      "read": {
        "description": "If the message has been read",
        "type": "boolean"
      },
      "severity": {
        "description": "The severity of the message",
        "type": "string",
        "enum": [
          "unknown",
          "info",
```

```
            "ok",
            "warning",
            "error"
          ]
        },
        "text": {
          "description": "The translated messages",
          "type": "string"
        },
        "time": {
          "description": "The time of the message",
          "type": "string",
          "format": "date-time",
          "readOnly": true
        }
      }
    },
    "api.MessageUpdate": {
      "properties": {
        "read": {
          "description": "Update the read status of the message(s)",
          "type": "boolean"
        }
      }
    },
    "api.Messages": {
      "required": [
        "data",
        "UnreadCount"
      ],
      "properties": {
        "UnreadCount": {
          "type": "string"
        },
        "data": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/api.Message"
          }
        },
        "isTruncated": {
          "description": "Returns true if list was truncated",
          "type": "boolean",
          "readOnly": true
        },
        "marker": {
          "description": "The marker specified in the request",
          "type": "string",
          "readOnly": true
        },
        "maxKeys": {
          "description": "The maximum number of keys specified in the request",
          "type": "integer",
          "format": "int32",
          "readOnly": true
        },
        "maxUnreadSeverity": {
          "description": "The maximum severity of the unread messages",
          "type": "string",
          "enum": [
            "unknown",
```

```
            "info",
            "ok",
            "warning",
            "error"
          ]
        },
        "nextMarker": {
          "description": "The starting ID of the next page",
          "type": "string",
          "readOnly": true
        }
      }
    },
    "api.Metrics": {
      "properties": {
        "count": {
          "description": "Number of items",
          "type": "integer",
          "format": "int64"
        },
        "optional": {
          "description": "Number of bytes of stored optional content",
          "type": "integer",
          "format": "int64"
        },
        "size": {
          "description": "Number of bytes of content",
          "type": "integer",
          "format": "int64"
        },
        "stored": {
          "description": "Number of bytes of content after compression",
          "type": "integer",
          "format": "int64"
        }
      }
    },
    "api.NetworkInterface": {
      "required": [
        "name",
        "mtu"
      ],
      "properties": {
        "addresses": {
          "description": "Bound Network Addresses",
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "auto6": {
          "description": "IPV6 Automatic Configuration Enabled",
          "type": "boolean"
        },
        "dhcp4": {
          "description": "IPV4 DHCP Enabled",
          "type": "boolean"
        },
        "dhcpDns": {
          "description": "Use DHCP for DNS",
          "type": "boolean"
```

```
      },
      "gateway4": {
        "description": "Configured IPV4 Gateway",
        "type": "string"
      },
      "gateway6": {
        "description": "Configured IPV6 Gateway",
        "type": "string"
      },
      "mtu": {
        "description": "Maximum Transmission Unit",
        "type": "integer",
        "format": "int32"
      },
      "name": {
        "description": "Interface Device Name",
        "type": "string"
      },
      "nameServers": {
        "description": "List of DNS Name Servers",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "routes": {
        "description": "Network routes (if any)",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.NetworkRoute"
        }
      },
      "searchDomains": {
        "description": "List of DNS Search Domains",
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "api.NetworkInterfaceCommon": {
    "properties": {
      "addresses": {
        "description": "Bound Network Addresses",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "auto6": {
        "description": "IPV6 Automatic Configuration Enabled",
        "type": "boolean"
      },
      "dhcp4": {
        "description": "IPV4 DHCP Enabled",
        "type": "boolean"
      },
      "dhcpDns": {
        "description": "Use DHCP for DNS",
        "type": "boolean"
```

```
      },
      "gateway4": {
        "description": "Configured IPV4 Gateway",
        "type": "string"
      },
      "gateway6": {
        "description": "Configured IPV6 Gateway",
        "type": "string"
      },
      "nameServers": {
        "description": "List of DNS Name Servers",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "routes": {
        "description": "Network routes (if any)",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.NetworkRoute"
        }
      },
      "searchDomains": {
        "description": "List of DNS Search Domains",
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "api.NetworkInterfaceUpdate": {
    "properties": {
      "addresses": {
        "description": "Bound Network Addresses",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "auto6": {
        "description": "IPV6 Automatic Configuration Enabled",
        "type": "boolean"
      },
      "dhcp4": {
        "description": "IPV4 DHCP Enabled",
        "type": "boolean"
      },
      "dhcpDns": {
        "description": "Use DHCP for DNS",
        "type": "boolean"
      },
      "gateway4": {
        "description": "Configured IPV4 Gateway",
        "type": "string"
      },
      "gateway6": {
        "description": "Configured IPV6 Gateway",
        "type": "string"
      },
```

```
      "mtu": {
        "description": "Maximum Transmission Unit",
        "type": "integer",
        "format": "int32"
      },
      "nameServers": {
        "description": "List of DNS Name Servers",
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "routes": {
        "description": "Network routes (if any)",
        "type": "array",
        "items": {
          "$ref": "#/definitions/api.NetworkRoute"
        }
      },
      "searchDomains": {
        "description": "List of DNS Search Domains",
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  },
  "api.NetworkRoute": {
    "required": [
      "to"
    ],
    "properties": {
      "onLink": {
        "description": "Add route on link",
        "type": "boolean"
      },
      "scope": {
        "description": "Route scope",
        "type": "string"
      },
      "to": {
        "description": "Destination network",
        "type": "string"
      },
      "via": {
        "description": "Route to via the specified address",
        "type": "string"
      }
    }
  },
  "api.NodeCredentials": {
    "required": [
      "KeyCredentials",
      "nodekey",
      "nodesecret",
      "nonce"
    ],
    "properties": {
      "KeyCredentials": {
        "$ref": "#/definitions/api.KeyCredentials"
```

```
          },
          "nodekey": {
            "description": "New node credentials access key",
            "type": "string"
          },
          "nodesecret": {
            "description": "New node credentials encrypted secret",
            "type": "string"
          },
          "nonce": {
            "description": "Nonce for node secret decryption",
            "type": "string"
          }
        }
      },
      "api.Object": {
        "required": [
          "key",
          "lastModified"
        ],
        "properties": {
          "etag": {
            "description": "A unique identifier for the object content",
            "type": "string"
          },
          "isCompliance": {
            "description": "Indicates this object has a compliance lock that can't be bypassed",
            "type": "boolean"
          },
          "isDelete": {
            "description": "Indicates this is a delete marker (which has no etag, size, or owner)",
            "type": "boolean"
          },
          "isLatest": {
            "description": "Indicates this is the current version",
            "type": "boolean"
          },
          "key": {
            "description": "Object name",
            "type": "string"
          },
          "lastModified": {
            "description": "Object creation time",
            "type": "string",
            "format": "date-time"
          },
          "legalHold": {
            "description": "Indicates this object cannot be deleted until the legal hold is removed",
            "type": "boolean"
          },
          "ownerId": {
            "description": "Canonical ID of the account that owns the object",
            "type": "string"
          },
          "ownerName": {
            "description": "Name associated with the owning account",
            "type": "string"
          },
          "restoredUntil": {
            "description": "If present, the object has been restored and the restore expires at this
time",
```

```
        "type": "string",
        "format": "date-time"
      },
      "retainUntil": {
        "description": "If present, an object lock prevents deletion until this time",
        "type": "string",
        "format": "date-time"
      },
      "size": {
        "description": "Size of the object",
        "type": "integer",
        "format": "int64"
      },
      "storageClass": {
        "description": "Storage class",
        "type": "string",
        "enum": [
          "STANDARD",
          "INTELLIGENT_TIERING",
          "STANDARD_IA",
          "ONEZONE_IA",
          "REDUCED_REDUNDANCY",
          "GLACIER_IR",
          "GLACIER",
          "DEEP_ARCHIVE",
          "ANY"
        ]
      },
      "versionId": {
        "description": "Version ID (only present when querying versions)",
        "type": "string"
      }
    }
  },
  "api.ObjectMetadata": {
    "required": [
      "version",
      "clones"
    ],
    "properties": {
      "clones": {
        "$ref": "#/definitions/api.CloneState"
      },
      "version": {
        "type": "string"
      }
    }
  },
  "api.Paginator": {
    "properties": {
      "isTruncated": {
        "description": "Returns true if list was truncated",
        "type": "boolean",
        "readOnly": true
      },
      "marker": {
        "description": "The marker specified in the request",
        "type": "string",
        "readOnly": true
      },
      "maxKeys": {
```

```
        "description": "The maximum number of keys specified in the request",
        "type": "integer",
        "format": "int32",
        "readOnly": true
      },
      "nextMarker": {
        "description": "The starting ID of the next page",
        "type": "string",
        "readOnly": true
      }
    }
  },
  "api.PerformanceDataset": {
    "required": [
      "label",
      "unit",
      "data"
    ],
    "properties": {
      "data": {
        "description": "Performance metric data",
        "type": "array",
        "items": {
          "$ref": "#/definitions/tseries.DataPoint"
        }
      },
      "label": {
        "description": "Performance dataset label",
        "type": "string"
      },
      "unit": {
        "description": "Performance dataset data unit",
        "type": "string"
      }
    }
  },
  "api.PerformanceTable": {
    "required": [
      "name",
      "title"
    ],
    "properties": {
      "name": {
        "description": "Performance table name",
        "type": "string"
      },
      "title": {
        "description": "Descriptive title",
        "type": "string"
      }
    }
  },
  "api.Policy": {
    "required": [
      "Version",
      "Statement"
    ],
    "properties": {
      "Id": {
        "type": "string"
      },
```

```
          "Statement": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/api.Statement"
            }
          },
          "Version": {
            "type": "string"
          }
        }
      },
      "api.Profiling": {
        "properties": {
          "cpu": {
            "description": "CPU profiling frequency (in seconds)",
            "type": "integer",
            "format": "int32"
          },
          "memory": {
            "description": "Memory profiling frequency (in seconds)",
            "type": "integer",
            "format": "int32"
          }
        }
      },
      "api.Retention": {
        "properties": {
          "compliance": {
            "description": "Retention mode is set to Compliance when true, Governance when false",
            "type": "boolean"
          },
          "days": {
            "description": "Number of days to retain locked objects (days or years must be specified,
not both)",
            "type": "integer",
            "format": "int32"
          },
          "years": {
            "description": "Number of years to retain locked objects (days or years must be
specified, not both)",
            "type": "integer",
            "format": "int32"
          }
        }
      },
      "api.Rule": {
        "required": [
          "name"
        ],
        "properties": {
          "apply": {
            "type": "string",
            "default": "all",
            "enum": [
              "all",
              "current",
              "nonCurrent"
            ]
          },
          "deletion": {
            "description": "Deletion removes clones not in the specified destinations",
```

```
            "type": "boolean"
          },
          "destinations": {
            "description": "The destination storage for clones.",
            "$ref": "#/definitions/api.Destinations"
          },
          "exclude": {
            "description": "Regular expression that must not match the object name",
            "type": "string"
          },
          "expiration": {
            "description": "Expiration is a deletion without destinations that deletes the object
version",
            "type": "boolean"
          },
          "include": {
            "description": "Regular expression that must match the object name",
            "type": "string"
          },
          "name": {
            "type": "string"
          },
          "noncurrentVersions": {
            "description": "How many noncurrent versions for the rule to keep. Once this number is
exceeded, the oldest noncurrent version is expired.",
            "type": "integer",
            "format": "int32"
          },
          "schedule": {
            "description": "The rule's schedule. If not scheduled, the rule executes immediately",
            "$ref": "#/definitions/api.RuleSchedule"
          }
        }
      },
      "api.RuleSchedule": {
        "required": [
          "days"
        ],
        "properties": {
          "days": {
            "description": "Number of days to wait before executing the rule.",
            "type": "integer",
            "format": "int32"
          }
        }
      },
      "api.Service": {
        "required": [
          "name"
        ],
        "properties": {
          "delay": {
            "description": "response time (in milliseconds)",
            "type": "integer",
            "format": "int64"
          },
          "name": {
            "description": "Name of the service",
            "type": "string"
          },
          "status": {
```

```
        "description": "Status",
        "type": "string",
        "enum": [
          "ok",
          "unavailable"
        ]
      }
    }
  },
  "api.ServiceStatusField": {
    "properties": {
      "status": {
        "description": "Status",
        "type": "string",
        "enum": [
          "ok",
          "unavailable"
        ]
      }
    }
  },
  "api.Statement": {
    "type": "object"
  },
  "api.Storage": {
    "required": [
      "id",
      "name",
      "type",
      "endpoint"
    ],
    "properties": {
      "alternate": {
        "description": "ID of alternate storage to move clones to during delete",
        "type": "string"
      },
      "archival": {
        "description": "Restore may be required to access data",
        "type": "boolean"
      },
      "bucket": {
        "description": "The external bucket to write too",
        "type": "string"
      },
      "bucketOwner": {
        "description": "The external bucket owner",
        "$ref": "#/definitions/api.BucketOwner"
      },
      "cautionThreshold": {
        "description": "Caution threshold capacity for the storage",
        "type": "integer",
        "format": "int32"
      },
      "cloneRestore": {
        "description": "Create a new clone when restoring this storage",
        "type": "boolean"
      },
      "cloudProvider": {
        "description": "Provider of cloud services (if applicable)",
        "type": "string",
        "enum": [
```

```
        "aws",
        "other",
        "azure",
        "google"
      ]
    },
    "empty": {
      "description": "Storage has no clone data",
      "type": "boolean"
    },
    "endpoint": {
      "description": "The id of the Vail endpoint owning this storage",
      "type": "string"
    },
    "id": {
      "description": "Storage identifier",
      "type": "string"
    },
    "link": {
      "description": "The vail bucket to ingest objects to",
      "type": "string"
    },
    "name": {
      "description": "Storage name",
      "type": "string"
    },
    "oldest": {
      "description": "The Vail version used to write the first data to the pool",
      "type": "string"
    },
    "optionalData": {
      "description": "Percentage of space available for optional data",
      "type": "integer",
      "format": "int32"
    },
    "readOnly": {
      "description": "Storage cannot be modified",
      "type": "boolean"
    },
    "region": {
      "description": "Cloud region (if applicable)",
      "type": "string"
    },
    "status": {
      "description": "Status",
      "type": "string",
      "enum": [
        "ok",
        "degraded",
        "unavailable",
        "deleting"
      ],
      "readOnly": true
    },
    "storageClass": {
      "description": "Storage class",
      "type": "string",
      "enum": [
        "STANDARD",
        "INTELLIGENT_TIERING",
        "STANDARD_IA",
```

```
            "ONEZONE_IA",
            "REDUCED_REDUNDANCY",
            "GLACIER_IR",
            "GLACIER",
            "DEEP_ARCHIVE",
            "ANY"
          ]
        },
        "type": {
          "description": "Storage type",
          "type": "string",
          "enum": [
            "file",
            "bp",
            "s3",
            "azure",
            "google"
          ]
        },
        "url": {
          "description": "S3 data path URL (if applicable)",
          "type": "string"
        },
        "verificationRunning": {
          "description": "Storage verification in progress",
          "type": "boolean"
        },
        "warningThreshold": {
          "description": "Warning threshold capacity for the storage",
          "type": "integer",
          "format": "int32"
        }
      }
    },
    "api.StorageClassField": {
      "properties": {
        "storageClass": {
          "description": "Storage class",
          "type": "string",
          "enum": [
            "STANDARD",
            "INTELLIGENT_TIERING",
            "STANDARD_IA",
            "ONEZONE_IA",
            "REDUCED_REDUNDANCY",
            "GLACIER_IR",
            "GLACIER",
            "DEEP_ARCHIVE",
            "ANY"
          ]
        }
      }
    },
    "api.StorageClassRequired": {
      "required": [
        "storageClass"
      ],
      "properties": {
        "storageClass": {
          "description": "Storage class",
          "type": "string",
```

```
            "enum": [
              "STANDARD",
              "INTELLIGENT_TIERING",
              "STANDARD_IA",
              "ONEZONE_IA",
              "REDUCED_REDUNDANCY",
              "GLACIER_IR",
              "GLACIER",
              "DEEP_ARCHIVE"
            ]
          }
        }
      },
      "api.StorageClone": {
        "required": [
          "id"
        ],
        "properties": {
          "archived": {
            "description": "Clone may require restore before access",
            "type": "boolean"
          },
          "id": {
            "description": "Storage identifier",
            "type": "string"
          },
          "optional": {
            "description": "Clone will be deleted if space is needed",
            "type": "boolean"
          },
          "partial": {
            "description": "Clone is not complete",
            "type": "boolean"
          },
          "restored": {
            "description": "Clone is currently restored",
            "type": "boolean"
          }
        }
      },
      "api.StorageCreate": {
        "required": [
          "name",
          "type",
          "endpoint"
        ],
        "properties": {
          "accessKey": {
            "description": "The account owner's access key, or the Azure storage account",
            "type": "string"
          },
          "arn": {
            "description": "The IAM role arn to use to access the account.  This can be used as an
alternative to providing accessKey and secretKey.",
            "type": "string"
          },
          "bucket": {
            "description": "The external bucket to write to",
            "type": "string"
          },
          "cautionThreshold": {
```

```
      "description": "Caution threshold capacity for the storage",
      "type": "integer",
      "format": "int32"
    },
    "cloneRestore": {
      "description": "Create a new clone when restoring this storage",
      "type": "boolean"
    },
    "cloudProvider": {
      "description": "Provider of cloud services (if applicable)",
      "type": "string",
      "enum": [
        "aws",
        "other",
        "azure",
        "google"
      ]
    },
    "credentials": {
      "description": "Credentials as a single element (e.g. Google JSON file)",
      "type": "string"
    },
    "endpoint": {
      "description": "The id of the Vail endpoint owning this storage",
      "type": "string"
    },
    "externalid": {
      "description": "The IAM role's external ID.S",
      "type": "string"
    },
    "link": {
      "description": "The vail bucket to ingest objects to",
      "type": "string"
    },
    "name": {
      "description": "Storage name",
      "type": "string"
    },
    "optionalData": {
      "description": "Percentage of space available for optional data",
      "type": "integer",
      "format": "int32"
    },
    "region": {
      "description": "The region where the account resides",
      "type": "string"
    },
    "secretKey": {
      "description": "The account owner's secret key",
      "type": "string"
    },
    "storageClass": {
      "description": "Storage class",
      "type": "string",
      "enum": [
        "STANDARD",
        "INTELLIGENT_TIERING",
        "STANDARD_IA",
        "ONEZONE_IA",
        "REDUCED_REDUNDANCY",
        "GLACIER_IR",
```

```
            "GLACIER",
            "DEEP_ARCHIVE",
            "ANY"
          ]
        },
        "type": {
          "description": "Storage type",
          "type": "string",
          "enum": [
            "file",
            "bp",
            "s3",
            "azure",
            "google"
          ]
        },
        "url": {
          "description": "S3 data path URL (if applicable)",
          "type": "string"
        },
        "warningThreshold": {
          "description": "Warning threshold capacity for the storage",
          "type": "integer",
          "format": "int32"
        }
      }
    },
    "api.StorageEntity": {
      "required": [
        "id",
        "storageClass"
      ],
      "properties": {
        "data": {
          "description": "Portion of physical media used by this storage for object data",
          "type": "integer",
          "format": "int64"
        },
        "id": {
          "description": "Storage ID",
          "type": "string"
        },
        "optional": {
          "description": "Portion of physical media used by this storage for cached data",
          "type": "integer",
          "format": "int64"
        },
        "storageClass": {
          "description": "Storage class",
          "type": "string",
          "enum": [
            "STANDARD",
            "INTELLIGENT_TIERING",
            "STANDARD_IA",
            "ONEZONE_IA",
            "REDUCED_REDUNDANCY",
            "GLACIER_IR",
            "GLACIER",
            "DEEP_ARCHIVE"
          ]
        }
```

```
        }
      },
      "api.StorageStatusField": {
        "properties": {
          "status": {
            "description": "Status",
            "type": "string",
            "enum": [
              "ok",
              "degraded",
              "unavailable",
              "deleting"
            ],
            "readOnly": true
          }
        }
      },
      "api.StorageUpdate": {
        "properties": {
          "accessKey": {
            "description": "The account owner's access key, or the Azure storage account",
            "type": "string"
          },
          "alternate": {
            "description": "ID of alternate storage to move clones to during delete",
            "type": "string"
          },
          "arn": {
            "description": "The IAM role arn to use to access the account.  This can be used as an
alternative to providing accessKey and secretKey.",
            "type": "string"
          },
          "cautionThreshold": {
            "description": "Caution threshold capacity for the storage",
            "type": "integer",
            "format": "int32"
          },
          "cloneRestore": {
            "description": "Create a new clone when restoring this storage",
            "type": "boolean"
          },
          "externalid": {
            "description": "The IAM role's external ID.",
            "type": "string"
          },
          "name": {
            "description": "Storage name",
            "type": "string"
          },
          "optionalData": {
            "description": "Percentage of space available for optional data",
            "type": "integer",
            "format": "int32"
          },
          "secretKey": {
            "description": "The account owner's secret key",
            "type": "string"
          },
          "status": {
            "description": "Status can be set to deleting to begin background deletion",
            "type": "string",
```

```
          "enum": [
            "deleting"
          ]
        },
        "storageClass": {
          "description": "Storage class",
          "type": "string",
          "enum": [
            "STANDARD",
            "INTELLIGENT_TIERING",
            "STANDARD_IA",
            "ONEZONE_IA",
            "REDUCED_REDUNDANCY",
            "GLACIER_IR",
            "GLACIER",
            "DEEP_ARCHIVE",
            "ANY"
          ]
        },
        "warningThreshold": {
          "description": "Warning threshold capacity for the storage",
          "type": "integer",
          "format": "int32"
        }
      }
    },
    "api.StorageUsed": {
      "required": [
        "storage"
      ],
      "properties": {
        "data": {
          "description": "Number of bytes of object data stored",
          "type": "integer",
          "format": "int64"
        },
        "endpoint": {
          "description": "Endpoint that hosts this storage",
          "type": "string"
        },
        "entity": {
          "description": "Identifier for underlying physical media",
          "type": "string"
        },
        "label": {
          "description": "Label for bundled capacity information",
          "type": "string"
        },
        "location": {
          "description": "ID of location",
          "type": "string"
        },
        "optional": {
          "description": "Number of bytes of cached data stored",
          "type": "integer",
          "format": "int64"
        },
        "storage": {
          "description": "Space used by each storage that shares this physical media",
          "type": "array",
          "items": {
```

```
          "$ref": "#/definitions/api.StorageEntity"
        }
      },
      "total": {
        "description": "Maximum capacity (if storage has one)",
        "type": "integer",
        "format": "int64"
      },
      "used": {
        "description": "Number of bytes of physical media used (including overhead)",
        "type": "integer",
        "format": "int64"
      }
    }
  },
  "api.StorageVerification": {
    "properties": {
      "alert": {
        "description": "Send alert messages when permanent read errors are encountered",
        "type": "boolean"
      },
      "full": {
        "description": "Perform a full verification of readable clones",
        "type": "boolean"
      }
    }
  },
  "api.Summary": {
    "required": [
      "totalManaged",
      "objectCount"
    ],
    "properties": {
      "objectCount": {
        "description": "Total Number of Managed Objects",
        "type": "integer",
        "format": "int64"
      },
      "totalManaged": {
        "description": "Bytes of Total Managed storage",
        "type": "integer",
        "format": "int64"
      }
    }
  },
  "api.System": {
    "required": [
      "os",
      "type",
      "name"
    ],
    "properties": {
      "id": {
        "description": "System identifier. Only present if the system is registered.",
        "type": "string"
      },
      "key": {
        "description": "The sphere activation key. Only present if the system is registered.",
        "type": "string"
      },
      "monitor": {
```

```
            "description": "True if monitor events are sent to SpectraLogic.",
            "type": "boolean"
          },
          "name": {
            "description": "Name of the system",
            "type": "string"
          },
          "namespace": {
            "description": "The sphere name. Only present if the system is activated.",
            "type": "string"
          },
          "nightly": {
            "description": "Nightly processing time (in UTC).",
            "type": "string"
          },
          "os": {
            "description": "The system's operating system'",
            "type": "string"
          },
          "sequence": {
            "description": "Current sequence unique ID.",
            "type": "string"
          },
          "sphere": {
            "description": "The sphere credentials endpoint. Only present if the system is
activated.",
            "type": "string"
          },
          "time": {
            "description": "Current system time.",
            "type": "string",
            "format": "date-time"
          },
          "type": {
            "description": "Type of system",
            "type": "string",
            "enum": [
              "bp",
              "vm"
            ]
          }
        }
      },
      "api.SystemCommon": {
        "required": [
          "name",
          "os",
          "type"
        ],
        "properties": {
          "id": {
            "description": "System identifier. Only present if the system is registered.",
            "type": "string"
          },
          "key": {
            "description": "The sphere activation key. Only present if the system is registered.",
            "type": "string"
          },
          "monitor": {
            "description": "True if monitor events are sent to SpectraLogic.",
            "type": "boolean"
```

```
          },
          "name": {
            "description": "Name of the system",
            "type": "string"
          },
          "nightly": {
            "description": "Nightly processing time (in UTC).",
            "type": "string"
          },
          "os": {
            "description": "The system's operating system'",
            "type": "string"
          },
          "sequence": {
            "description": "Current sequence unique ID.",
            "type": "string"
          },
          "sphere": {
            "description": "The sphere credentials endpoint. Only present if the system is
activated.",
            "type": "string"
          },
          "time": {
            "description": "Current system time.",
            "type": "string",
            "format": "date-time"
          },
          "type": {
            "description": "Type of system",
            "type": "string",
            "enum": [
              "bp",
              "vm"
            ]
          }
        }
      },
      "api.SystemUpdate": {
        "properties": {
          "key": {
            "description": "The sphere activation key",
            "type": "string"
          },
          "monitor": {
            "description": "True if monitor events are sent to SpectraLogic.",
            "type": "boolean"
          },
          "name": {
            "description": "Name of the system",
            "type": "string"
          },
          "nightly": {
            "description": "Nightly processing time (in UTC).",
            "type": "string"
          }
        }
      },
      "handlers.NodeRegistrationInfo": {
        "required": [
          "system",
          "token"
```

```
    ],
    "properties": {
      "system": {
        "description": "System information",
        "$ref": "#/definitions/api.System"
      },
      "token": {
        "description": "Authentication token for the node GUI",
        "$ref": "#/definitions/rest.Token"
      }
    }
  },
  "handlers.PresignedS3Request": {
    "required": [
      "url"
    ],
    "properties": {
      "url": {
        "type": "string"
      }
    }
  },
  "license.Entitlement": {
    "required": [
      "dimension",
      "value"
    ],
    "properties": {
      "dimension": {
        "description": "Entitlement dimension",
        "type": "string"
      },
      "expires": {
        "description": "Entitlement expiration date",
        "type": "string",
        "format": "date-time"
      },
      "value": {
        "description": "Entitlement value. May be a string, float64, int32, or boolean.",
        "type": "string"
      }
    }
  },
  "license.SignedEntitlements": {
    "required": [
      "id",
      "entitlements",
      "signature"
    ],
    "properties": {
      "entitlements": {
        "description": "Entitlements",
        "type": "array",
        "items": {
          "$ref": "#/definitions/license.Entitlement"
        }
      },
      "id": {
        "description": "Machine ID",
        "type": "string"
      },
```

```
          "signature": {
            "description": "Signature",
            "type": "string"
          }
        }
      },
      "rest.Credentials": {
        "required": [
          "username"
        ],
        "properties": {
          "challengeName": {
            "description": "The challenge to pass.",
            "type": "string"
          },
          "challengeResponses": {
            "description": "The responses for the given challenge.  This is required when
challengeName is supplied.",
            "type": "object",
            "additionalProperties": {
              "type": "string"
            }
          },
          "otp": {
            "description": "One-time password for multi-factor authentication. This is only used for
backends that don't support MFA challenges (e.g. BlackPearl)'.",
            "type": "string"
          },
          "password": {
            "description": "The password for the user.  This is only required when initially trying
to create a token and not when responding to a challenge.",
            "type": "string"
          },
          "session": {
            "description": "The session ID used to pass the challenge.  This is required when
challengeName is supplied.",
            "type": "string"
          },
          "username": {
            "description": "The user login name.",
            "type": "string"
          }
        }
      },
      "rest.SphereToken": {
        "properties": {
          "challengeName": {
            "description": "The challenge that must be passed first.",
            "type": "string",
            "readOnly": true
          },
          "challengeParameters": {
            "description": "The challenge parameters for the given challenge.",
            "type": "object",
            "additionalProperties": {
              "type": "string"
            },
            "readOnly": true
          },
          "session": {
            "description": "Temporary session ID used to pass a challenge.  This is only populated if
ChallengeName is populated.",
```

```
            "type": "string",
            "readOnly": true
          },
          "sphere": {
            "description": "The Sphere that this token is valid for.",
            "type": "string",
            "readOnly": true
          },
          "token": {
            "description": "The JSON Web Token",
            "type": "string",
            "readOnly": true
          },
          "username": {
            "description": "The user login name.",
            "type": "string",
            "readOnly": true
          }
        }
      },
      "rest.Token": {
        "properties": {
          "challengeName": {
            "description": "The challenge that must be passed first.",
            "type": "string",
            "readOnly": true
          },
          "challengeParameters": {
            "description": "The challenge parameters for the given challenge.",
            "type": "object",
            "additionalProperties": {
              "type": "string"
            },
            "readOnly": true
          },
          "session": {
            "description": "Temporary session ID used to pass a challenge.  This is only populated if
ChallengeName is populated.",
            "type": "string",
            "readOnly": true
          },
          "token": {
            "description": "The JSON Web Token",
            "type": "string",
            "readOnly": true
          },
          "username": {
            "description": "The user login name.",
            "type": "string",
            "readOnly": true
          }
        }
      },
      "server.RequestError": {
        "required": [
          "code",
          "message"
        ],
        "properties": {
          "code": {
            "description": "a string identifying the type of error",
```

```
        "type": "string"
      },
      "message": {
        "description": "a textual description of the error",
        "type": "string"
      }
    }
  },
  "server.ValidationErrorResponse": {
    "required": [
      "code",
      "message"
    ],
    "properties": {
      "code": {
        "description": "a string identifying the type of error",
        "type": "string"
      },
      "errors": {
        "description": "a map of attributes and the errors associated with each",
        "type": "object",
        "additionalProperties": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/server.RequestError"
          }
        }
      },
      "message": {
        "description": "a textual description of the error",
        "type": "string"
      }
    }
  },
  "tseries.DataPoint": {
    "required": [
      "x",
      "y"
    ],
    "properties": {
      "x": {
        "description": "time of data point",
        "type": "string",
        "format": "date-time"
      },
      "y": {
        "description": "value of data point",
        "type": "number",
        "format": "double"
      }
    }
  },
  "users.User": {
    "required": [
      "id",
      "username",
      "email"
    ],
    "properties": {
      "email": {
        "description": "The user's email address",
```

```
        "type": "string",
        "uniqueItems": true
      },
      "error": {
        "description": "True if the user wants to receive Error emails",
        "type": "boolean"
      },
      "id": {
        "description": "The user's ID (represented as the username)",
        "type": "string",
        "uniqueItems": true
      },
      "info": {
        "description": "True if the user wants to receive Info emails",
        "type": "boolean"
      },
      "username": {
        "description": "The user's username",
        "type": "string",
        "uniqueItems": true
      },
      "warning": {
        "description": "True if the user wants to receive Warning emails",
        "type": "boolean"
      }
    }
  },
  "users.UserCollection": {
    "required": [
      "data"
    ],
    "properties": {
      "data": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/users.User"
        }
      }
    }
  },
  "users.UserPatch": {
    "properties": {
      "email": {
        "description": "User's email address'",
        "type": "string"
      },
      "error": {
        "description": "True if the user wants to receive Error emails",
        "type": "boolean"
      },
      "info": {
        "description": "True if the user wants to receive Info emails",
        "type": "boolean"
      },
      "warning": {
        "description": "True if the user wants to receive Warning emails",
        "type": "boolean"
      }
    }
  },
  "worker.CommonPrefixResult": {
```

```
        "required": [
          "prefix"
        ],
        "properties": {
          "prefix": {
            "type": "string"
          }
        }
      },
      "worker.UpdateStatus": {
        "required": [
          "update"
        ],
        "properties": {
          "detail": {
            "type": "string"
          },
          "update": {
            "type": "string",
            "format": "int32"
          },
          "value": {
            "type": "integer",
            "format": "int32"
          }
        }
      }
    }
  }
```