日志控制

概述

`peer` 和 `orderer` 中的日志是由 `common/flogging` 包提供的，这个包支持：

- 根据消息的严重性进行日志记录控制
- 根据生成消息的软件 *记录器* 的记录控制
- 根据消息的严重性，提供不同的漂亮的打印选项

目前所有日志都输出到 `stderr`。为用户和开发人员提供了按严重性对日志进行全局和日志级别的控制。目前没有针对每个严重性级别提供信息类型的正式规范。提交错误报告时，开发人员可能希望在 DEBUG 级别查看完整的日志

在正常打印的日志中，日志记录级别由不同颜色的四个字符的代码表示，例如，"ERRO" 表示错误（ERROR），"DEBU" 表示调试（DEBUG），等等。在日志记录上下文中，*记录器* 是由开发人员对相关消息组的命名（字符串）。在下面的正常打印的示例，记录器 `ledgermgmt`、`kvledger` 和 `peer` 都在生成日志。

```
2018-11-01 15:32:38.268 UTC [ledgermgmt] initialize -> INFO 002 Initializing ledger mgmt
2018-11-01 15:32:38.268 UTC [kvledger] NewProvider -> INFO 003 Initializing ledger provider
2018-11-01 15:32:38.342 UTC [kvledger] NewProvider -> INFO 004 ledger provider Initialized
2018-11-01 15:32:38.357 UTC [ledgermgmt] initialize -> INFO 005 ledger mgmt initialized
2018-11-01 15:32:38.357 UTC [peer] func1 -> INFO 006 Auto-detected peer address: 172.24.0.3:7051
2018-11-01 15:32:38.357 UTC [peer] func1 -> INFO 007 Returning peer0.org1.example.com:7051
```

可以在运行时创建任意数量的记录器，因此没有记录器的"主列表"，并且日志控件结构无法检查记录器是否确实存在或将生成。

日志规范

`peer` 和 `orderer` 命令的日志记录级别由日志记录规范控制，该规范是通过 ``FABRIC_LOGGING_SPEC`` 环境变量设置的。

完整的日志记录级别规范具有以下形式：

```
[<logger>[,<logger>...]=]<level>[:[<logger>[,<logger>...]=]<level>...]
```

日志记录严重性级别使用不区分大小写的字符串指定

```
FATAL | PANIC | ERROR | WARNING | INFO | DEBUG
```

日志记录级别有默认值。但是，可以使用如下语法来指定某个或某组记录器

```
<logger>[,<logger>...]=<level>
```

示例如下：

```
info                                        - Set default to INFO
warning:msp,gossip=warning:chaincode=info   - Default WARNING; Override for msp, gossip, and chai
chaincode=info:msp,gossip=warning:warning   - Same as above
```

记录格式

`peer` 和 `orderer` 命令的日志记录格式是通过 `FABRIC_LOGGING_FORMAT` 环境变量控制的。可以将其设置为格式化的字符串，例如默认打印为可读的控制台格式：

```
"%{color}%{time:2006-01-02 15:04:05.000 MST} [%{module}] %{shortfunc} -> %{level:.4s} %{id:03x}%{
```

也可以将其设置为 `json`，按照 JSON 格式输出日志。

链码

链码日志是链码开发人员的责任。

作为独立执行的程序，技术上用户提供的链码也可以在 stdout / stderr 上生成输出。这自然对“devmode” 很有用，但在生产环境中通道一般会禁用该模式以减轻有破坏性或恶意代码的滥用。然而，可以通过配置 CORE_VM_DOCKER_ATTACHSTDOUT = true 选项，在每个 Peer 节点管理容器（例如，“netmode”）上启用此输出。

一旦启用，每个链码都将收到以其自身容器 ID 为键的日志记录通道。写入到 stdout 或 stderr 的任何输出都将逐行集成到 Peer 节点的日志中。不建议将其用于生产。

可以使用适用于容器平台的标准命令，从链码容器查看未转发到 Peer 容器的 stdout 和 stderr。

```
docker logs <chaincode_container_id>
kubectl logs -n <namespace> <pod_name>
oc logs -n <namespace> <pod_name>
```