Build Asset Registry

The database containing build asset version information, channels, and subscriptions used by DARC, Maestro, and MC

Requirements

- Builds can be assigned to one or more Channel(s)
- Channels and Builds can be either private or public
- Public Repositories cannot subscribe to private Channels
- Private Builds cannot be assigned to a Public Channel
- MC can get full dependency graph of a Build

Scenarios

1. Query
2. Add
3. Update

Query

1. Get latest Build of a repository for a channel
   - Used by Darc to determine if a dependency needs to be updated
2. Get repositories that subscribe to updates from a [Repository] intended for a Channel
   - Used by Darc to flow dependencies when a Build is assigned to a Channel
3. Get a Specific Build
   - Used by MC
4. Get dependencies of a Build
   - Used by MC to show dependency tree for a Build

Add

1. Push new Asset and Build information
   - Used by repository build once new assets are ready for consumption
2. Create new Channel(s) and Subscription(s)
   - Used by Maestro and Darc for product release, feature branch creation, and dev PR scenarios

Update

1. Assign existing Build to a Channel
   - Used by Maestro based on repo policy
   - Used by Darc when asked by devs
2. Create/Edit/Delete existing Subscription(s), Channel(s) and Build(s)
   - Used by users

Definitions

Subscription

Represents a specific repository branch's desire to receive updates from another repository on a specific Channel.

```
class Subscription {
    string ChannelName;
    string SourceRepository;
    string TargetRepository;
    string TargetBranch;
    SubscriptionPolicy Policy;
}
```

Subscription Policy

Represents the policies that are applied to a Subscription.

```
class SubscriptionPolicy {
    UpdateFrequency UpdateFrequency; // "every day", "every build", ...
    MergePolicy MergePolicy; // "when green", "unit tests passed", "never", ...
}
```

Channel

A Channel contains a set of builds

```
class Channel {
    string Name;
    string Classification; // "Official", "DevBuild", "PR Validation", ...
}
```

Build

Identifies a specific build of a repository.

```
class Build {
    string Repository;
    string Branch;
    string Commit;
    string BuildNumber;
    DateTimeOffset DateProduced;
    List<Channel> Channels;
    List<Asset> Assets;
    List<Build> Dependencies;
}
```

DefaultChannel

Identifies a default channel that will be associated with new builds from a specific repository/branch.

```
class DefaultChannel {
    string Repository;
    string Branch;
    Channel Channel;
}
```

Asset

A specific Asset including when it was produced and where it can be found.

```
class Asset {
  string Name;
  string Version;
  Build ProducedBy;
  List<AssetLocation> Locations;
}
```

Asset Location

A feed or other publish mechanism where a specific Asset can be found.

```
class AssetLocation {
    string Location;
    LocationType Type; // "NuGet Feed", "Blob Container", ...
}
```

Was this helpful?