SONiC Utilities Repository

Complete Guide

Table of Content

Introduction

This repository contains code to provide Command line interface. The SONiC-CLI, which is based upon a CLICK Python library, gives a SONiC-centric and user-friendly way to interact with the operating system, making it easier to configure network modules, view status information, troubleshoot network settings, and access operational data such as interfaces, routing tables, and system logs.

A Python wheel package, containing all the Python source code for the command-line utilities. This package is linked with CONFIG_DB. It is considered as the user data fetching db from CLI.

Setting up a build/test environment

The sonic-utilities package depends on a number of other packages, many of which are available via PyPI, but some are part of the SONiC codebase. When building/testing the package, setuptools/pip will attempt to install the packages available from PyPI. However, you will need to manually build and install the SONiC dependencies before attempting to build or test the package.

A convenient alternative is to let the SONiC build system configure a build enviroment for you. This can be done by cloning the sonic-buildimage repo, building the sonic-utilities package inside the Debian Buster slave container, and staying inside the container once the build finishes. During the build process, the SONiC build system will build and install all the necessary dependencies inside the container.

In order to build sonic-utilities locally, we have offical guide We have internally our own utilities-build-system guides as well as below:

1. Building a SONiC Utilities.docx

Repository Structure

The SONiC utilities repository comprises of a total of thirty six folders, only the main key folders will be discussed as follows:

- **Clear**
- **Config**
- **Connect**
- **Consutil**
- **Debug**
- **Doc**
- **Dump**
- **Generic_config_update**
- **Scripts**
- **Show**
- **Sonic-utilities-data**
- **Sonic-cli-gen**
- **Sonic-installer**
- **Sonic-package-manager**
- **Tests**
- **Utilities_common**

sonic-utilites/clear

The **clear/** directory within the sonic-utilities repository contains scripts and utilities that are specifically designed for clearing or resetting configurations and network states on SONiC devices. This folder provides functionality related to clearing network configurations, interface states, and other settings.

Key Files and Their Roles in the clear/ Directory

1. **__init__.py**

- **Purpose**: Initializes the clear directory as a Python package. This file may contain setup code for initializing the module or managing imports within the package.

2. **aliases.ini**

- **Purpose**: Contains alias definitions for simplifying and customizing commands in the CLI. This file allows for shorthand or custom commands to trigger specific clear operations.

3. **bgp_frr_v6.py**

- **Purpose**: Clears BGP-related configurations for **FRR (Free Range Routing)** in **IPv6** networks.

4. **bgp_quagga_v4.py**

- **Purpose**: Clears **IPv4** BGP configurations when using the **Quagga** routing daemon.

5. **bgp_quagga_v6.py**

- **Purpose**: Clears **IPv6** BGP configurations for the **Quagga** routing daemon.

6. **main.py**

- **Purpose**: Likely serves as the main entry point for executing clear commands via the CLI. This file would handle command parsing and execution of core logic to clear various configurations or reset settings.

sonic-utilites/config

The **config/** directory contains scripts and utilities responsible for managing configuration-related tasks. These tasks include configuring network settings, managing system configurations, and enabling specific features such as VLANs, BGP, DNS, etc. The directory provides functionality for setting and modifying various configurations using the CLI.

Directory Structure and File Details

Files within the **config/** directory, organized by their function:

1. **__init__.py**

- **Purpose**: This file initializes the config directory as a Python package, allowing it to be imported and used by other parts of the SONiC CLI utilities. It sets up the basic configuration handling and structure for the package.

2. **aaa.py**

- **Purpose**: Handles CLI commands related to **AAA (Authentication, Authorization, and Accounting)** configurations. This file adds support for **LDAP** (Lightweight Directory Access Protocol), which allows the integration of external authentication services into SONiC.
- **Key Functions**:
  - LDAP configuration commands.
  - Authentication-related CLI operations.

3. **bgp_cli.py**

- **Purpose**: Provides configuration commands related to **BGP (Border Gateway Protocol)**, including commands to configure BGP peers, routes, and related settings for routing protocols on SONiC devices.
- **Key Functions**:
  - Configuring BGP settings.
  - Adding or removing BGP peers.
  - Managing BGP routes.

4. chassis_modules.py

- **Purpose**: Offers configuration capabilities for **chassis modules**, including commands for managing startup and shutdown operations of modules in the chassis.
- **Key Functions**:
    - Start/stop chassis modules.
    - Manage chassis configuration.

5. config_mgmt.py

- **Purpose**: Manages configuration operations, including multi-ASIC support and **Golden Config** overrides. It is used for managing configurations in systems with multiple ASICs (Application-Specific Integrated Circuits).
- **Key Functions**:
    - Multi-ASIC configuration management.
    - Golden Config overrides and fixes.

6. console.py

- **Purpose**: Provides functionalities related to console management and includes YANG validation for configurations such as **MCLAG**, **NAT**, and **MUXCABLE** tables.
- **Key Functions**:
    - Console management commands.
    - YANG validation for configuration changes.

7. dns.py

- **Purpose**: Implements configuration and show commands for managing **static DNS** settings, allowing users to configure DNS servers and related settings on the SONiC device.
- **Key Functions**:
    - Configuring static DNS entries.
    - Viewing current DNS configuration.

8. **fabric.py**

- **Purpose**: Deals with **fabric capacity monitoring** and provides CLI commands for monitoring and managing fabric settings in SONiC, especially in chassis or multi-ASIC configurations.
- **Key Functions**:
    - Managing fabric-related settings.
    - Monitoring fabric capacity and configurations in multi-ASIC setups.

9. **feature.py**

- **Purpose**: Handles YANG validation for specific configuration updates, including features like **PORT_STORM_CONTROL** and **PORT_QOS_MA**. It validates changes made to **ConfigDB** to ensure they follow the correct YANG models.
- **Key Functions**:
    - YANG validation for ConfigDB updates.
    - Validating specific features like **PORT_QOS_MA**.

10. **flow_counters.py**

- **Purpose**: Provides a CLI for managing **route flow counters**, enabling the monitoring of flow statistics related to routing in SONiC.
- **Key Functions**:
    - Route flow counter management.
    - Viewing and modifying flow counter settings.

11. **kdump.py**

- **Purpose**: Manages **kdump** configuration settings. This file includes functionality for controlling how crash dumps are handled, including the option to disable automatic saving of kdump configurations to startup configurations.
- **Key Functions**:
    - Managing crash dump (kdump) settings.
    - Modifying startup configuration related to kdump.

12. **kube.py**

- **Purpose**: Similar to console.py, it provides YANG validation for configurations specific to **MCLAG**, **NAT**, and **MUXCABLE** tables, particularly in a Kubernetes or containerized environment.
- **Key Functions**:
    - YANG validation for Kubernetes-related configurations.

    - Managing MCLAG, NAT, and MUXCABLE configurations in a Kubernetes context.

13. **main.py**

- **Purpose**: Likely the main entry point for executing configuration-related commands via the CLI. This file handles command parsing and execution logic for various configuration operations in SONiC.
- **Key Functions**:
    - Main interface for running configuration commands.
    - Parsing and executing configuration changes.

14. **mclag.py**

- **Purpose**: Contains functionality for configuring **MCLAG** (Multi-Chassis Link Aggregation) features, including managing link aggregation across multiple chassis for redundancy and load balancing.
- **Key Functions**:
    - Configuring MCLAG settings.
    - Managing multi-chassis link aggregation.

15. **muxcable.py**

- **Purpose**: Deals with **MUXCABLE** configurations, likely related to multi-tenant cables or specific hardware configurations for high-performance networking.
- **Key Functions**:
    - Managing MUXCABLE settings.

16. **nat.py**

- **Purpose**: Provides configuration commands for managing **NAT** (Network Address Translation), allowing users to configure how traffic is routed between different network segments, especially for internal and external communications.
- **Key Functions**:
    - Configuring NAT settings.
    - Managing IP address translations.

17. **switchport.py**

- **Purpose**: Configures **switchport mode** and related settings, which control the way ports on SONiC devices handle traffic.
- **Key Functions**:
    - Configuring switchport modes.
    - Managing port settings.

18. **syslog.py**

- **Purpose**: Implements commands for managing **syslog** settings, including the ability to update log levels and refresh the configuration.
- **Key Functions**:
  - Updating syslog settings.
  - Refreshing syslog configuration dynamically.

19. **utils.py**

- **Purpose**: Contains utility functions used across the configuration CLI commands. One notable function is splitting **VLAN** configurations into separate files for easier management.
- **Key Functions**:
  - Utility functions for managing VLAN configurations.
  - Supporting functions for other config operations.

20. **validated_config_db_connector.py**

- **Purpose**: Enhances performance by making patch sorting optional in **Golden Config Updates** (GCU). It connects to ConfigDB and improves configuration performance by controlling patch sorting.
- **Key Functions**:
  - Improving patch sorting performance.
  - Managing validated connections to ConfigDB.

21. **vlan.py**

- **Purpose**: Provides CLI commands for configuring **VLANs** and related settings, including managing switchport modes for different VLANs.
- **Key Functions**:
  - Configuring VLANs.
  - Managing VLAN-related switchport settings.

22. **vxlan.py**

- **Purpose**: Provides commands related to **VXLAN** (Virtual Extensible LAN) configuration, likely used for managing virtual network overlays or tunnels.
- **Key Functions**:
  - Configuring VXLAN tunnels and network overlays.

sonic-utilites/connect

The **connect/** directory in the **sonic-utilities** repository contains utilities related to **network connection management** and the **interaction** with various services or devices. This includes connecting to various SONiC components, clearing configurations, and handling the underlying connections for network management tasks. It provides CLI commands to manage connectivity and related operations on the system.

Directory Structure and File Overview

Here’s a detailed overview of the files found in the **connect/** directory:

1. **__init__.py**

- **Purpose**: This file is used to initialize the connect module and define it as part of the Python package. It is responsible for setting up the necessary imports, functions, and structures needed for the connection utilities in SONiC.
- **Last Commit**: "Console show/clear/connect skeleton (#278)."

2. **main.py**

- **Purpose**: This is likely the entry point for handling **connection**-related operations in the **connect** CLI module. It includes the core logic for managing connections, processing commands, and executing connection-related tasks. The file is responsible for parsing user input and providing necessary outputs related to network connections, such as establishing or terminating connections.
- **Key Functions**:
  - Managing connections to devices or services.
  - Parsing and executing connect-related commands.
  - Handling exceptions or errors related to network connectivity

sonic-utilites/consutil

The **consutil/** directory contains utilities related to managing and interacting with the **console** interface in SONiC. It includes functions for clearing configurations, displaying console outputs, and performing other console management tasks in SONiC devices.

Directory Structure and File Overview

Here’s an overview of the files in the **consutil/** directory:

1. **__init__.py**

- **Purpose**: Initializes the **consutil** module and defines it as a Python package. It enables other parts of the SONiC system to import and utilize the console utilities within this package.

2. **lib.py**

- **Purpose**: Contains supporting functions and utilities related to **console operations**. This file handles the underlying logic for interacting with the console interface, such as clearing console settings, handling input/output, and managing console configurations.

3. **main.py**

- **Purpose**: Acts as the main entry point for handling **console-related commands** via the **CLI**. It parses user input and invokes functions from **lib.py** to perform tasks like clearing the console, displaying the status, and interacting with the system through the console interface.
- **Key Functions**:
  - Parsing and executing console-related commands.
  - Managing input/output to and from the console.
  - Coordinating with other parts of the system to carry out console operations.

sonic-utilites/debug

The **debug/** directory in the **sonic-utilities** repository is responsible for providing command-line utilities that assist with debugging tasks. This includes functionalities for displaying debug information, enabling or disabling specific debug levels, and managing other debug-related operations .

Directory Structure and File Overview

Here's an overview of the files in the **debug/** directory:

1. **__init__.py**

- **Purpose**: This file is used to initialize the **debug** module, marking it as a Python package and allowing it to be imported into other parts of the SONiC CLI utilities. It also helps set up the required functions and imports for debugging operations.

2. **main.py**

- **Purpose**: This is the main entry point for executing debugging commands via the **CLI**. It contains the logic for parsing user input, executing debug-related operations, and providing the necessary outputs related to debugging.
- **Key Functions**:
  - Enabling and disabling debug features.
  - Showing debugging logs or information about the current system state.
  - Managing the execution of various debug-related commands, such as **debug** and **undebug**.

sonic-utilites/doc

The **doc/** directory in the **sonic-utilities** repository is dedicated to documentation. File contains the **command reference** for the **sonic-utilities** CLI tools. It documents the available commands, their syntax, usage, options, and examples for users to interact with SONiC through the command line. It serves as an essential guide for understanding how to configure, monitor, and troubleshoot SONiC devices using the provided CLI tools.

sonic-utilites/dump

The **dump/** directory in the **sonic-utilities** repository is focused on utilities for **dumping system state** and configuration data. This is typically used for troubleshooting and diagnostic purposes. The utilities in this directory help extract detailed information about the system's current state, including various module states and network configuration details.

Directory Structure and File Overview

1. __init__.py

- **Purpose**: Initializes the **dump** module, making it a Python package. This allows other parts of the SONiC utilities to import and use the functions defined in this directory.

2. helper.py

- **Purpose**: Contains utility functions that assist with dumping specific parts of the SONiC system. It is responsible for handling the extraction and formatting of the dumped information, as well as applying any necessary filtering.

3. main.py

- **Purpose**: The **main.py** file is the primary entry point for executing dump-related commands from the **CLI**. It controls the execution of various dump utilities, orchestrating the collection and presentation of system state data.
- **Key Functions**:
  - Handling **dump state** CLI commands.
  - Extracting data from various modules and formatting it for display.

4. match_helper.py

- **Purpose**: Provides support for pattern matching or filtering during the dump process, especially useful when dealing with large data sets or when specific pieces of information need to be isolated from the full dump.

5. match_infra.py

- **Purpose**: Defines the infrastructure and functions needed for matching and filtering data during the dump operation, particularly when working with complex configurations such as network rules or ACLs.

sonic-utilites/generic_config_updater

The **generic_config_updater/** directory in the **sonic-utilities** repository is focused on managing and updating **generic configurations** for SONiC devices. The utilities in this directory allow for applying configuration patches, rolling back changes, and managing checkpoints, providing a way to update system configurations safely and effectively.

Directory Structure and File Overview

1. **__init__.py**

- **Purpose**: Initializes the **generic_config_updater** module, allowing other parts of the SONiC utilities to import and utilize its functions. This file likely sets up the necessary configurations and dependencies for the module.

2. **change_applier.py**

- **Purpose**: Handles the application of configuration changes to the system. This file manages the logic for applying patches and updates and may also handle rollback functionality and checkpoint creation.

3. **field_operation_validators.py**

- **Purpose**: Contains validation functions to ensure the correctness of the configuration fields being updated. It checks the integrity of the field-level operations before any updates are applied.

4. **gcu_field_operation_validators.conf.json**

- **Purpose**: A JSON configuration file that defines the rules and settings used for field-level validation during configuration updates.

5. **gcu_services_validator.conf.json**

- **Purpose**: A JSON configuration file used for validating service-related configuration updates. It ensures that services are configured correctly before applying changes.

6. **generic_updater.py**

- **Purpose**: The central script for handling **generic configuration updates**. It manages patch application, namespace checking, and multi-ASIC configurations.

7. **gu_common.py**

- **Purpose**: Provides common functions and utilities used throughout the **generic_config_updater** module. These are helper functions that streamline tasks like logging, error handling, and configuration management.

8. patch_sorter.py

- **Purpose**: This file manages the sorting of patches before they are applied, ensuring the correct order for patch application to avoid conflicts.

9. services_validator.py

- **Purpose**: Validates the configuration changes related to system services. It ensures that services are properly configured and compatible with the intended changes before they are applied.

sonic-utilites/scripts

The /scripts/ folder in the sonic-utilities repository is a collection of various utility scripts used for managing, troubleshooting, and interaction. These scripts provide functionalities for performing system checks, managing hardware components, gathering statistics, and supporting network troubleshooting on SONiC-based devices, typically running on switches.

The folder contains a variety of Python and shell scripts that cover different use cases.

Types of Scripts in /scripts/ Folder:

1. **System Checks and Configuration:**
    a. **check_db_integrity.py**: Checks the integrity of the database used by SONiC (such as Redis or other databases).
    b. **configlet**: Used for managing configuration templates.
    c. **route_check.py**: Checks the network routes, and the most recent commit improved the performance of this script (#3544).
2. **Diagnostics and Troubleshooting:**
    a. **coredump-compress.py**: Compresses core dump files for diagnostic purposes.
    b. **dualtor_neighbor_check.py**: Checks neighbor status in dual-tor (two top-of-rack) network configurations.
    c. **debug_voq_chassis_packet_drops.sh**: Helps in debugging packet drops in VoQ (Virtual Output Queues) on chassis-based systems.
    d. **techsupport_cleanup.py**: Cleans up files and configurations for auto-techsupport features to help in troubleshooting issues like core dumps or system errors.
3. **Network and Hardware Statistics:**
    a. **fdbshow**: Displays Forwarding Database (FDB) entries.
    b. **fibshow**: Shows the routing information, both IPv4 and IPv6 forwarding information.
    c. **portstat**: Shows statistics related to the network ports on the device.
    d. **sensorshow**: Displays voltage and current sensor data.
4. **System Reboot and Boot Management:**
    a. **warm-reboot**: Performs a warm reboot on a device, which typically retains certain operational states while restarting the system.
    b. **fast-reboot**: A faster reboot process for the system.
    c. **verify_image_sign.sh**: Verifies the signature of an image file before performing an upgrade to ensure security.
5. **Performance Monitoring and Configurations:**
    a. **storm_control.py**: Configures storm control for broadcast, unknown unicast, and multicast traffic on a network switch.
    b. **pfcstat**: Provides statistics related to Priority Flow Control (PFC), typically used in Ethernet networks to manage congestion.
    c. **vnet_route_check.py**: Performs checks on virtual network routes, especially for platforms running virtualized network functions.
6. **Specialized Features and Support:**

   a. **generate_dump.py**: Generates dumps of system state or logs, useful for troubleshooting.
   b. **dpu-tty.py**: Utility for interacting with the DPU (Data Processing Unit) console.
   c. **db_migrator.py**: Migrates database versions, ensuring compatibility with newer versions of SONiC or its components.

sonic-utilites/show

The /show/ directory in the sonic-utilities repository contains scripts that implement CLI commands for displaying various types of system and network information. These scripts are part of the SONiC system's show command functionality, which is designed to provide users with real-time information about different components of the network device

Overview of Key Files and Their Functions:

1. **General Purpose:**
   a. **__init__.py**: This file is used to initialize the show module. It essentially sets up the structure for the other show commands to work under this module.
   b. **main.py**: A key file for managing CLI commands related to the banner (e.g., showing the current banner and configuring it). The most recent commit in October 2024 added functionality for configuring and displaying the banner message.
2. **Network and System Information:**
   a. **bgp_cli.py & bgp_common.py**: Handle BGP (Border Gateway Protocol) related commands like showing IP routes (show ip bgp) and BGP neighbor status (show ip bgp nei). The files have undergone several changes, such as modifying the output for multi-ASIC platforms and hiding unnecessary interfaces.
   b. **dns.py**: Implements commands for DNS (Domain Name System) configuration and display, including static DNS settings (show dns).
   c. **fabric.py**: Provides CLI commands for fabric-related information, such as showing fabric rates (show fabric rate).
   d. **vlan.py**: Contains commands for VLAN (Virtual Local Area Network) configurations and displays VLAN-related information.
   e. **vnet.py**: Implements commands related to virtual networks, particularly focused on virtual network routing (show vnet routes).
3. **Hardware and Platform Information:**
   a. **chassis_modules.py**: Displays the status of chassis modules, including the serial field in the chassis module status (show chassis modules).
   b. **platform.py**: Provides CLI commands for platform-related statistics such as voltage and current sensors (show platform).
   c. **muxcable.py**: Includes commands to display multiplex cable configuration (show mux config), recently updated to support new SOC configurations like soc_ipv6.
   d. **p4_table.py**: Adds commands for interacting with P4 tables, a key component of programmable networking.
   e. **reboot_cause.py**: Provides user-friendly information about the cause of a reboot, especially useful for kernel panic or other reboot-related events.
4. **Traffic and Flow Statistics:**

   a. **flow_counters.py**: Implements the show flow-counters command, providing statistics related to flow monitoring in SONiC devices.
   b. **sflow.py**: Displays statistics and configuration related to sFlow (sampled flow monitoring) including support for egress sFlow (show sflow).
5. **Process and System Health Information:**
   a. **processes.py**: Provides commands for displaying the status of processes running on the SONiC device.
   b. **system_health.py**: Displays system health information, useful for monitoring the overall status and well-being of the device.
6. **Reboot and Restart Information:**
   a. **warm_restart.py**: Likely provides commands to show or configure warm reboot or restart behavior for the system.
   b. **reboot_cause.py**: Displays the cause of the last system reboot, especially useful for troubleshooting unexpected restarts.

sonic-utilites/sonic-utilities-data

The /sonic-utilities-data/ directory in the sonic-utilities repository seems to be focused on supporting data, templates, and configuration files for the utility tools, rather than containing scripts that provide direct CLI commands or system information.

Overview of Key Files and Folders:

1. **bash_completion.d/**:
   a. Contains **bash completion scripts** for SONiC CLI commands. These files are used to provide auto-completion when typing commands in the SONiC CLI, improving the user experience by suggesting available options or completing command arguments.
2. **debian/**:
   a. Contains files related to **Debian package creation** for SONiC utilities. This folder likely includes configurations and scripts necessary to package sonic-utilities as a Debian package for installation on Debian-based Linux distributions.
3. **templates/**:
   a. Contains **template files** that are likely used by various scripts in the repository. These templates could be used for generating configuration files, reports, or other required data structures in the SONiC environment.

sonic-utilites/sonic_cli_gen

The /sonic_cli_gen/ directory is focused on **automating the generation of SONiC CLI commands**, likely based on YANG data models. It contains Python scripts that define the tools and functionality needed to automatically generate configuration commands for SONiC, which can significantly streamline the process of managing and interacting with SONiC devices. The use of YANG models suggests a sophisticated approach to managing network configurations in a standardized and scalable way.

Overview of Key Files:

1. **__init__.py**:
    a. This is an **initialization file** for the sonic_cli_gen module. It marks the directory as a Python package, allowing the module to be imported into other Python scripts or projects.
2. **generator.py**:
    a. This script likely handles the **generation of SONiC CLI commands**. It is part of the functionality that automates the creation of CLI commands based on specific parameters or configurations. It may define functions that generate the command syntax or the logic for creating command-line utilities automatically.
3. **main.py**:
    a. This is probably the **main entry point** for the sonic_cli_gen module. It could provide the logic for starting the command generation process and may include user-facing functions for interacting with the tool. It might also handle configurations or input from users to customize the generated commands.
4. **yang_parser.py**:
    a. This file likely contains logic for **parsing YANG models**. YANG is a data modeling language used in networking to define data structures and configurations. This parser would help extract relevant information from YANG models and use that data to generate corresponding SONiC CLI commands.

sonic-utilites/sonic_installer

The /sonic_installer/ directory in the **sonic-utilities** repository contains utilities related to the installation and management.

Key Files in /sonic_installer/:

1. **__init__.py**:
    a. Marks the directory as a Python package. This file enables other Python scripts or utilities to import the functions or classes from the sonic_installer module.
2. **bootloader/**:
    a. This directory likely contains scripts or tools related to the **bootloader** of SONiC. The **latest commit** (Sep 16, 2024) indicates a change from using bootctl to mokutil for checking the **Secure Boot** status. This suggests that the bootloader-related utilities deal with secure boot settings and configurations for SONiC devices.
3. **aliases.ini**:

    a. This file contains **configuration data** for the sonic_installer utility, possibly related to aliases or simplified command names for different install or boot tasks. It helps in simplifying and streamlining command invocations in the SONiC installer.

4. **common.py**:
    a. This script likely contains **common utility functions** shared across different components of the sonic_installer. The latest commit indicates improvements in exception handling by introducing **notes** to make error messages more informative.
5. **exception.py**:
    a. This file defines the **exception handling logic** for the sonic_installer. It appears to be focused on improving the error reporting system by adding more detailed and user-friendly notes to the exceptions, which can be helpful for troubleshooting during the installation or upgrade process.

6. **main.py**:
    a. The main entry point for the sonic_installer utility. This script likely drives the **installation or upgrade process** and may provide user interfaces or CLI commands for managing SONiC installations. It probably coordinates the different tasks needed for installing or updating SONiC on a device

sonic-utilites/sonic_package_manager/

The /sonic_package_manager/ directory in the **sonic-utilities** repository is dedicated to managing **SONiC packages** and their associated components. It houses the utility responsible for **installing, upgrading, and managing packages** within a SONiC system. This includes both system packages and potentially third-party containerized applications.

Key Files in /sonic_package_manager/:

1. **__init__.py**:
    a. Marks the directory as a Python package. This enables the various modules in the package to be imported and used by other parts of the system.
2. **service_creator**:
    a. Likely contains logic related to **creating services** related to SONiC packages. The latest commit (Sep 16, 2024) involves a cleanup of the **timers auto-generation logic**, possibly affecting how services and package timers are configured or managed.
3. **constraint.py**:
    a. Contains logic related to **constraints** for packages, possibly related to compatibility or dependency management. The recent commit suggests that there was an issue with mutable default values in dataclasses, which has been fixed. This could be part of how package dependencies or configuration options are handled.
4. **database.py**:
    a. Likely manages interactions with a **database** for storing metadata about the installed or available packages. It is central to keeping track of package versions, statuses, and possibly configurations for installed SONiC packages.

5. **dockerapi.py**:
    a. This file likely contains logic for interacting with **Docker containers**. Since SONiC supports containerized applications, this module would help manage container lifecycles, image management, and interactions with Docker when dealing with packages that are containerized.
6. **errors.py**:
    a. Manages the **error handling** for the package manager. This would define custom error types, provide helpful error messages, and possibly log errors related to package installation, upgrades, or removals.
7. **logger.py**:
    a. Responsible for **logging** operations within the package manager. It would capture logs related to package management actions, such as installations, upgrades, or failures, to aid in troubleshooting.
8. **main.py**:
    a. Acts as the entry point for the **SONiC package manager** utility. This file likely orchestrates the entire package management process, such as installing or upgrading packages, and may contain the main logic for interacting with the user or other services.
9. **manager.py**:
    a. Likely handles the **core management tasks** for SONiC packages, including installation, upgrade, and removal of packages. It would interact with other modules like the database and Docker API to perform these tasks.
10. **manifest.py**:
    a. Deals with the **manifest** files associated with packages. These files typically contain metadata about the package, including versioning, dependencies, and other relevant details. This module may validate or parse these files when handling packages.
11. **metadata.py**:
    a. Manages **metadata** related to SONiC packages. It might store details about package versions, authors, descriptions, and other attributes necessary for the package manager.
12. **package.py**:
    a. Contains logic related to individual **packages** themselves. This would likely define how to install, upgrade, and configure a package, along with dependencies or any special configurations that need to be handled.
13. **progress.py**:
    a. Tracks the **progress** of package management tasks. This could be useful for providing users with feedback on the installation or upgrade progress.
14. **reference.py**:
    a. Likely contains **reference** data or constants used throughout the package management system. It could store predefined values or configurations related to how packages are managed.

15. **registry.py**:
    a. Manages the **registry** of available or installed packages. It could interact with a central repository or local registry to track package availability and versions. A recent commit (Aug 27, 2022) involved dropping the expires_in field, which may have been related to package expiration or versioning.
16. **source.py**:
    a. Likely deals with the **sources** of the packages, such as repositories or locations from which packages can be fetched or installed.
17. **utils.py**: Contains various **utility functions** for supporting the package manager.
18. **version.py**:
    a. This script manages versioning for the **sonic-package-manager** utility, helping to keep track of version numbers and ensuring compatibility with packages and their versions.

sonic-utilites/tests/

The tests directory in the sonic-utilities repository contains a wide range of test files for verifying different parts of the Sonic utilities and its CLI functionalities.