|  |  |
| --- | --- |
| **Class Name:** Action | |
| **Description:** Contains the operations to perform with a Formatter’s Rule(s) | |
| **Contracts:**  1. Manage Action  *Add Action to Rule*  addAction(Rule<>, Action)  Description: Receives a Rule and an Action, adds a new Action to the Rule.  Pre: The action belongs to one of hiding, renaming, annotating or creating a hook.  Post: The addition of the specified action to the rule.    *Delete Action from Rule*  deleteAction(Rule<>, Action)  Description: Receives an action and deletes it.  Pre: The Action is in the Rule and is the last added to the Rule.  Post: The deletion of the Action.  2. Manage Field  *Hide Field*  hideField(Field)  Description: Receives a field currently displayed and hides it.  Pre: The field belongs to one of A and is not hidden.  Post: The field is hidden.    *Unhide Field*  showField(Field)  Description: Receives a field currently hidden and displays it.  Pre: The field belongs to one of A and is not hidden.  Post: The field is displayed.  *Update Field*  updateField(String, Field)  updateField(Boolean, Field)  Description: Receives either a string or a Boolean along with a field, and updates a field value according to the value provided.  Pre: The field belongs to one of A.  Post: The field element value gets changed to the new value.  **Private Responsibilities:** | **Collaborations:**  Rule (23), Hook (13)  Rule (23), Hook (13)  Field (5)  Field (5)  Field (6), Hook (15) |
| **Comments:**  Contains the steps by which to: Hide, Rename, Annotate, Create Hook | |

|  |  |
| --- | --- |
| **Class Name:** AnnotationAction | |
| **Description:** Contains metadata of the annotation and annotation itself for a Field | |
| **Contracts:**  3. Manage Annotation  *Add Annotation*  addAnnotation(String, Rule)  Description: Receives a String adds a new Action to the Rule.  Pre: The Action is  Post: The addition of the specified annotation to the action.  *Delete Annotation*  deleteAnnotation(AnnotationAction, Rule)  Description: Receives an AnnotationAction and deletes it.  Pre: The AnnotationAction is the last Action added to the Rule.  Post:  **Private Responsibilities:** | **Collaborations:**  Rule (23)  Rule (23) |
| **Comments:**  Attributes include user name, time added, time modified, and the annotation itself | |

|  |  |
| --- | --- |
| **Class Name:** Expression | |
| **Description:** The string by which to Filter views on the Proto Element, Field, or Rules | |
| **Contracts:**  4. Manage Primitives  *Add Primitive*  addPrimitive(String)  Description: Receives a primitive (composed from the string) and adds it to this Expression  Pre: The primitive is in the Berkeley Packet Filter (BPF) syntax.  Post: The addition of a primitive to this Expression.  *Remove Primitive*  deletePrimitive(String)  Description: Deletes a specified primitive.  Pre:  Post: The deletion of the primitive from this Expression.  **Private Responsibilities:** | **Collaborations:** |
| **Comments:**  Contains one or more primitives defined by Berkeley Packet Filter (BPF) syntax | |

|  |  |
| --- | --- |
| **Class Name:** Field | |
| **Description:** Element of ProtoElement that is editable Rule(s) in a given Protocol | |
| **Contracts:**  5. Visualize Field  *Show Field*  showField(Field)  Description: Receives a field currently hidden and displays it.  Pre: The field belongs to one of A and is not hidden.  Post: The field is displayed.  *Hide Field*  hideField(Field)  Description: Receives a field currently displayed and hides it.  Pre: The field belongs to one of A and is not hidden.  Post: The field is hidden.  6. Modify Field  *Rename Field*  renameField(String, Action, Field)  Description: Receives a string and a field to rename using the provided string, adding to the Action  Pre: The Field exists within the PDML  Post:  *Add Hook to Field*  addHook(Action, Field)  Description: Adds a Hook to this Field in this Action.  Pre: The Field is editable by this Formatter.  Post: The Hook is added to the Action for this Field with a Script attached to it.  *Annotate Field*  addAnnotation(Action, Annotation)  Description: Receives an action and an annotation, adds a new annotation to the action.  Pre: The action is either hiding, renaming, annotating or creating a hook.  Post: The addition of the specified annotation to the action.  *Update Field*  updateField(String, Field)  updateField(Boolean, Field)  Description: Receives either a string or a Boolean along with a field, and updates a field value according to the value provided.  Pre: The Field exists within the PDML  Post:  **Private Responsibilities:** | **Collaborations:**  Action (2)  Action (2)  Action (1)  Action (1)  AnnotationAction (3)  Hook (15) |
| **Comments:**  Contains the following optional and required attributes:  Optional attributes – hidden (Boolean), mask (Bytes), showdtl (String), showmap (String), showvalue (Integer)  Required attributes – longname (String), name (String), pos (Bytes), show, showname (String), size (Bytes), unmaskedvalue (*required only if the length of the current field is not an integer number of bytes*), value (Hex, String) | |

|  |  |
| --- | --- |
| **Class Name:** File | |
| **Description:** Represents the Script, Historical Copy, and PDML | |
| **Contracts:**  7. Create Historical Copy  *Convert in tShark*  convertPCAP(File)  Description: Receives a PCAP file to be sent to tShark for conversion to receive a PDML file.  Pre: The file is of type PCAP.  Post:  *Create PDML File*  createHistCopy(File)  Description: Receives a PDML file and creates a historical copy of the PDML file.  Pre: The file is a PDML file.  Post:  **Private Responsibilities:**  - Create new File | **Collaborations:**  tShark (27) |
| **Comments:**  Abstract class which handles creation, opening, saving, and modifications to PDML original and Script as well as creation of Historical Copies | |

|  |  |
| --- | --- |
| **Class Name:** Filter | |
| **Description:** Contains Expression(s) by which to visualize and modify the PDML capture | |
| **Contracts:**  8. Manage Filter  *Add Filter to Rule*  addFilter(Rule, Filter)  Description: Receives a filter and a rule, adds the filter to the rule.  Pre: The Rule exists.  Post:  *Delete Filter from Rule*  deleteFilter(Filter)  Descritpion: Receives a filter and deletes it.  Pre: The Filter is the last added to the Rule.  Post:  9. Manage Expression  *Add Primitive to Expression*  addPrimitive(String,Expression)  Description: Receives a primitive (composed from the string) and adds it to the Expression  Pre: The Expression is in the Berkeley Packet Filter (BPF) syntax.  Post: The addition of a primitive to the Expression for this Filter.  *Remove Primitive from Expression*  deletePrimitive(String, Expression)  Description: Deletes a specified primitive.  Pre: The primitive is contained in this Expression.  Post: The deletion of the primitive from the Expression for this Filter.  **Private Responsibilities:** | **Collaborations:**  Rule (22)  Rule (22)  Expression (4)  Expression (4) |
| **Comments:**  Contains one Berkeley Packet Filter (BPF) expression | |

|  |  |
| --- | --- |
| **Class Name:** Formatter | |
| **Description:** Allows to apply rules to Fields in a Protocol for inspection | |
| **Contracts:**  10. Manage Formatter  *Create Formatter*  createFormatter(Rule<>)  Description: Receives a list of rules to build a new formatter.  Pre: The list of rules is empty  Post:  *Apply Rules*  applyRules(Formatter, Rule<>)  Description: Receives a Formatter and a set of Rules and applies the Rules to the Formatter.  Pre: Rule<> is not empty, and the Formatter exists  Post:  11. Manage Rules  *Add a New Rule*  addRule(Formatter, Rule)  Description: Receives a formatter and a rule, adds the rule to the formatter.  Pre: The Rule is not empty and the Formatter exists.  Post:  *Delete the Last Rule*  deleteRule(Rule)  Description: Receives a rule and deletes it.  Pre: The Rule is the last one added to the Formatter  Post: | **Collaborations:**  PDML (17)  Rule (22, 23)  Rule (22, 23) |
| **Comments:**  Contains Filter and Action rules specific to one Protocol; type of File | |

|  |  |
| --- | --- |
| **Class Name:** GUI | |
| **Description:** Interfaces to PFS functionality | |
| **Contracts:**  28. Visualize Script Requests  *Accept Script Modifications*  modifyScript(Hook)  Description: Accepts the user input to modify the Script for this Hook  Pre: This Hook exists  Post:  *Accept Input Parameters*  addInputParameters(Hook, String)  Description: Accepts the user input to add input parameters to run this Hook’s Script  Pre: This Hook exists  Post:  *Accept Link to Script*  modifyScriptLink(Hook, String)  Description: Accepts the user input to modify the link to this Hook’s Script  Pre: This Hook exists and this String is a valid system path  Post:  29. Visualize PDML Modifications  *Revert PDML to Historical Copy*  revertToOriginal(PDML)  Description:  Pre: This PDML has a Historical Copy  Post: The Historical Copy remains unmodified.  *Accept PDML Changes*  acceptChanges(PDML)  Description: Saves edits to this PDML File.  Pre:  Post: The Historical Copy remains unmodified and PDML File is saved.  **Private Responsibilities:**  - Accept input  - Visualize changes | **Collaborations:**  Hook (14)  Hook (14)  Hook (14)  PDML (18)  PDML (18) |
| **Comments:**  Accepts inputs and outputs to display | |

|  |  |
| --- | --- |
| **Class Name:** Historical Copy | |
| **Description:** File Class subtype which contains captured network data within PFS | |
| **Contracts:**  12. Manage Historical Copy  *Create PDML File*  createHistCopy(File)  Description: Receives a PDML file and creates a historical copy of the PDML file.  Pre: The file is a PDML file.  Post:  *Store PDML File*  storePDML(File)  Description: Receives a PDML file and stores it on the local filesystem.  Pre: The file has not been stored before on the current local filesystem of the user.  Post:  **Private Responsibilities:** | **Collaborations:**  File (7)  File (7) |
| **Comments:**  Attributes include identification as unmodifiable and metadata; type of File | |

|  |  |
| --- | --- |
| **Class Name:** Hook | |
| **Description:** Steps to update a value of specific attributes of a Field | |
| **Contracts:**  13. Manage Hook  *Create Hook*  createHook(Action, Script)  Description: The Hook is added to the Action  Pre: The Script exists.  Post:  *Remove Hook*  deleteHook(Action)  Description: Deletes the Hook from the Action  Pre:  Post: The Script is not deleted.  14. Modify Hook  *Get Input Parameters*  setScriptParameters(String<>)  Description: Stores the String as the input parameters for this Hook’s Script  Pre: The Script exists, the number of parameters matches for the Script, and the types match.  Post:  *Modify Link to Script*  modifyLinkToScript(String)  Description: Stores the String as the link or file path to the Python Script  Pre: The link is a valid system path to a Python (.PY) File.  Post:  *Modify Script*  modifyScript(String<>)  Description: Updates the Script’s algorithm to this String<>  Pre: String<> is valid, no-error-producing Python code.  Post:  15. Apply Hook  *Run Script*  runScript()  Description: The Script begins execution  Pre: This Script has its parameters and runs without error  Post:  *Get Return Value to Modify Field*  getReturnValue(Field)  Description: Gets the return value from the Script’s execution  Pre: The Script has completed execution and has a return value  Post:  **Private Responsibilities:** | **Collaborations:**  Action (1), Script (24)  Action (1)  GUI (28)  GUI (28)  GUI (28)  Script (26)  Field (6) |
| **Comments:**  Contains path to Script, Field to modify, and input parameters to Script’s algorithm (*associated values of specific attributes defined in the Field*) | |

|  |  |
| --- | --- |
| **Class Name:** Packet | |
| **Description:** Element of the PDML File | |
| **Contracts:**  16. Visualize Packet  *Show Packet*  showPacket(Packet)  Description: Receives a packet currently hidden and displays it.  Pre: The packet exists within the PDML and is hidden.  Post:  *Hide Packet*  hidePacket(Packet)  Description: Receives a packet currently displayed and hides it.  Pre: The packet exists within the PDML and is not hidden.  Post:  **Private Responsibilities:** | **Collaborations:**  Formatter (10)  Formatter (10) |
| **Comments:**  Contains one ProtoElement and the following required attributes: num (Integer), len (Bytes), caplen (Bytes), and timestamp (Float) | |

|  |  |
| --- | --- |
| **Class Name:** PDML | |
| **Description:** Allows by subtype to modify and analyze network data within PFS | |
| **Contracts:**  17. Apply Formatter  *Apply Formatter Rules*  applyFormatter(Formatter)  Description: Formatter to apply its rules to the current PDML  Pre: The Formatter is not currently being edited and the current PDML has Protocol(s) for this Formatter.  Post: The modified PDML will be saved and the Historical Copy will remain unedited.  *Update Field*  updateField(Action, Field)  Description: Receives a Field and an Action to be applied to this Field  Pre: The Field exists within the PDML’s Protocol.  Post: Only this Field will be modified according the specifications of this particular Action.  18. Revert to Historical Copy  *Visualize Changes from PDML and Historical Copy*  findChanges(HistoricalCopy)  Description: Returns a list of the differences between this PDML and itsHistorical Copy.  Pre: This PDML’s Historical Copy does exist within PFS.  Post: The PDML and Historical Copy will not be changed.  *Accept or Reject Changes*  acceptChanges()  Description: Saves the changes applied to this PDML in the PFS  Pre: The Formatter has been applied.  Post: Only this will be saved in the PFS.    rejectChanges(HistoricalCopy)  Description: Reverts the modified PDML to its Historical Copy  Pre: The Historical Copy exists in the PFS.  Post: The modified PDML copy is a deep copy of its Historical Copy.  **Private Responsibilities:**  - Save Changes | **Collaborations:**  Formatter (10)  Field (6)  Historical Copy (12)  GUI (29) |
| **Comments:**  Attributes include identification as modifiable and metadata; type of File | |

|  |  |
| --- | --- |
| **Class Name:** Protocol | |
| **Description:** Contains Packets by which the matching Formatter can modify | |
| **Contracts:**  19. Visualize Protocol  *Show Protocol*  showProtocol(Protocol)  Description: Receives a protocol currently hidden and displays it.  Pre: The protocol exists within the PDML and is hidden.  Post:  *Hide Protocol*  hideProtocol(Protocol)  Description: Receives a protocol currently displayed and hides it.  Pre: The protocol exists within the PDML and is not hidden.  Post:  **Private Responsibilities:** | **Collaborations:**  Formatter (10)  Formatter (10) |
| **Comments:**  Contains one or more Packets | |

|  |  |
| --- | --- |
| **Class Name:** ProtocolIdentifier | |
| **Description:** Scans PDML to identify existing and non-existing Formatters for each Protocol | |
| **Contracts:**  20. Scan PDML  *Identify Protocols*  identifyProtocols(File)  Description: Receives a file and returns a list of all identified protocols from the file.  Pre:  Post: Returns a list of all identified protocols from the file.  *Identify Formatters*  identifyFormatters(File)  Description: Receives a file and returns a list of all identified Formatters from the File.  Pre: The File has been opened by PFS.  Post: Returns a list of all applied Formatters from the File.  *Create Formatter*  Create Formatter(Rule<>)  Description: Receives a list of Rules to build a new Formatter.  Pre: The Rule list is Empty  Post:  **Private Responsibilities:** | **Collaborations:**  Formatter (10)  Formatter (10)  Formatter (10) |
| **Comments:**  Reads through Protocols in a modifiable PDML | |

|  |  |
| --- | --- |
| **Class Name:** ProtoElement | |
| **Description:** Element of each Packet in a PDML File | |
| **Contracts:**  21. Manage ProtoElement  *Edit Action*  addAction(Rule, Action)  Description: Receives a rule and an action, adds a new action to the rule.  Pre: The action is hiding, renaming, annotating or creating a hook.  Post:    deleteAction(Action)  Description: Receives an action and deletes it.  Pre: The action is hiding, renaming, annotating or creating a hook.  Post:  *Edit Filter*  addFilter(Rule, Filter)  Description: Receives a filter and a rule, adds the filter to the rule.  Pre: The Filter does not already exist in the Rule  Post:  deleteFilter(Rule, Filter)  Descritpion: Receives a filter and deletes it.  Pre: The Filter is the last added to the Rule.  Post:  *Edit Field*  renameField(String, Action, Field)  Description: Receives a string and a field to rename using the provided string, adding to the Action  Pre: The field belongs to one of A and the string is unique.  Post:  addHook(Action, Field)  Description: Adds a Hook to this Field in this Action.  Pre: The Field is editable by this Formatter.  Post: The Hook is added to the Action for this Field with a Script attached to it.  addAnnotation(Action, Annotation)  Description: Receives an action and an annotation, adds a new annotation to the action.  Pre: The action is hiding, renaming, annotating or creating a hook.  Post:  deleteAnnotation(Annotation)  Description: Receives an annotation and deletes it.  Pre: The annotation exists with this ProtoElement  Post: Only the annotation is removed from this ProtoElement  updateField(String, Field)  updateField(Boolean, Field)  Description: Receives either a string or a Boolean along with a field, and updates a field value according to the value provided.  Pre: The field exists in the PDML.  Post:  *Visualize Field*  hideField(Field)  Description: Receives a field currently displayed and hides it.  Pre: The field exists in the Protocol of the PDML and is not hidden.  Post:    showField(Field)  Description: Receives a field currently hidden and displays it.  Pre: The field exists in the protocol of the PDML and is not hidden.  Post:  **Private Responsibilities:** | **Collaborations:**  Rule (23)  Rule (22)  Field (6)  Rule (23), Field (6) |
| **Comments:**  Contains one Field and the following attributes: name (String), longname (String), size (Bytes), and pos (Bytes) | |

|  |  |
| --- | --- |
| **Class Name:** Rule | |
| **Description:** Set of Filters and Actions which to build the Formatter to modify the Protocol of a PDML | |
| **Contracts:**  22. Manage Rule’s Filters  *Add a New Filter to Rule*  addFilter(Filter)  Description: Receives a Filter and adds the Filter to the Rule.  Pre:  Post: This Filter is the new last Filter of the Rule.  *Delete the Last Filter from Rule*  deleteFilter(Filter)  Descritpion: Receives a Filter and deletes it.  Pre: The Filter is the last added Filter to the Rule.  Post: The deletion of the Filter.  23. Manage Rule’s Actions  *Add a New Action to Rule*  addAction(Action)  Description: Receives an Action and adds it to the Rule.  Pre:  Post: This Action is the new last Action of the Rule.  *Delete the Last Action from Rule*  deleteAction(Action)  Descritpion: Receives an Action and deletes it.  Pre: The Action is the last added Action to the Rule.  Post:  **Private Responsibilities:** | **Collaborations:**  Filter (8, 9)  Filter (8)  Action (1), AnnotationAction (3), Hook (13)  Action (1), AnnotationAction (3), Hook (13) |
| **Comments:**  Contains Filters and Actions | |

|  |  |
| --- | --- |
| **Class Name:** Script | |
| **Description:** Subtype of File Class which contains the Python algorithm by which to update Field(s) | |
| **Contracts:**  24. Manage Script  *Create a New Script*  createScript(Hook)  Description: Creates a Script for this Hook  Pre:  Post: The Script Python File will be created if it does not already exist.  *Delete Script*  deleteScript(Hook)  Description: Deletes the Script from this Hook  Pre: The Script is associated with this Hook.  Post: The Script File is not deleted.  25. Modify Script  *Update Algorithm*  updateAlgorithm(String<>)  Description: Overwrites the existing algorithm to contain this String<>  Pre: The Script Python File exists.  Post:  *Update File Path*  updatePath(String)  Description: Sets the path to this Script to be this String  Pre: This String is a valid file system path to this Algorithm  Post: The Script is not deleted.  26. Run Script  *Get Return Value*  getReturnValue()  Description: Gets the return value from the Script’s execution  Pre: The Script has completed execution and has a return value  Post:  *Update Field*  updateField(Field)  Description: Updates the Field associated with this Hook  Pre: The Script’s return value type matches the Field’s value type  Post:  **Private Responsibilities:** | **Collaborations:**  Hook (13, 14)  Hook (14)  Hook (14)  Hook (14)  Hook (15)  Hook (15) |
| **Comments:**  Contains editable algorithm and file path; type of File | |

|  |  |
| --- | --- |
| **Class Name:** tShark | |
| **Description:** Handles operations for communications between tShark and Formatter Protocol System (FPS) | |
| **Contracts:**  27. Convert PCAP to PDML  *Send PCAP File*  sendPCAP(File)  Description: Receives a PCAP file to be parsed by tShark tool  Pre: The file has not been stored before on the current local filesystem of the user.  Post: tShark returns no error on accepting the PDML file to parse.  *Return Parsed PDML File*  storePDML()  Description: Receives a parsed PDML file and stores it on the local filesystem.  Pre: tShark parsed the PDML with no errors.  Post: A new file of type PDML is stored on the local filesystem of the user to be opened by PFS.  **Private Responsibilities:** | **Collaborations:**  File (7)  File (7) |
| **Comments:**  External tool to Formatter Protocol System (FPS) | |