EXHIBIT L

A/PROV

PTO/SB/16 (10-01)
Approved for use through 10/31/2002 OMB 0651-0032
U S Patent and Trademark Office, U S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

12/11/02 JC06 U.S. PTO

JC996 U.S. PTO 60/432255 12/11/02

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

Express Mail Label No.

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Michael T.	McKibben	Westerville, Ohio
Jeff R.	Lamb	Westerville, Ohio

☐ *Additional inventors are being named on the ___ separately numbered sheets attached hereto*

TITLE OF THE INVENTION (500 characters max)

METHOD FOR DYNAMIC ASSOCIATION OF ELECTRONICALLY STORED INFORMATION WITH ITERATIVE WORKFLOW CHANGES

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

[X] Customer Number 25534 → *Place Customer Number Bar Code Label here*

OR *Type Customer Number here*

☐ Firm or Individual Name					
Address					
Address					
City		State		ZIP	
Country		Telephone		Fax	

ENCLOSED APPLICATION PARTS *(check all that apply)*

- [X] Specification *Number of Pages* 18
- [ ] CD(s), Number
- [ ] Drawing(s) *Number of Sheets*
- [ ] Other (specify)
- [ ] Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT

- [X] Applicant claims small entity status. See 37 CFR 1.27.
- [X] A check or money order is enclosed to cover the filing fees
- [ ] The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number:
- [ ] Payment by credit card. Form PTO-2038 is attached.

FILING FEE AMOUNT ($): 80.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government

- [X] No
- [ ] Yes, the name of the U S Government agency and the Government contract number are

Respectfully submitted,

SIGNATURE Frederick N. Samuels

Date 12/11/2002

REGISTRATION NO. *(if appropriate)* 34715

Docket Number: 547.0003P

TYPED or PRINTED NAME Frederick N. Samuels

TELEPHONE 202-331-8777

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C., 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

Plaintiff's Trial Exhibit
PTX-3
Case No. 08-CV-00862

METHOD FOR DYNAMIC ASSOCIATION OF ELECTRONICALLY STORED INFORMATION WITH ITERATIVE WORKFLOW CHANGES

I. Field of the Invention

[0001] This invention relates to management and storage of electronic information. More particularly, this invention relates to new structures and methods for creating relationships between users, applications, files and folders.

II. Background of the Invention

[0002] Digital communications solutions are presently supplied to users in ways that are completely divorced from their business context. A particular item of communication provides little or no inherent understanding of how that communication furthers the purpose and intent of the group or enterprise. In other words, an email inbox collects emails about all topics, business and personal. The email application itself is not discerning about topic, priority or context beyond perhaps rudimentary "message filters" that will look for certain key words or people then place those items in target folders. Generally, it simply presents a sequential list of messages received. Similarly, a fax machine receives fax pages in sequence. A fax machine is not discerning about topic, priority or context. It simply outputs fax pages. Once received, it remains the task of the recipient to sort, categorize and organize these items of communication in ways most meaningful to that person. The organization task generally occurs outside the context of the particular communications tool itself.

[0003] Typical methods for organization of communications are limited and fragmented. For example, for an email, the recipient may either leave all email in the inbox or move it to another electronic folder. For a fax, the recipient is likely to place that fax in a file folder that is identified by project name or name of recipient. These

typical methods of organizing communications are wholly inadequate for a number of reasons:

[0004] 1. **Organization** – the recipient is left to do all the work of organization and categorization of the communications rather than having the systems themselves doing that work for them, automatically.

[0005] 2. **Leadership** – the linkage between business strategy and an individual act of communication is non-existent.

[0006] 3. **Categorization** – the items themselves rarely apply to only one topic of interest. As such, under current systems, the items would need to be manually stored in multiple locations (either electronic or "brick and mortar" folders). For example, a fax letter to a sales manager may contain information about contact addresses, market intelligence data, specific product requests, and financial accounting.

[0007] 4. **Knowledge Sharing** – items often relate to organizational issues for which one or more work groups need access; access that is denied when the recipient "buries" that item in his/her personal filing system, electronic or otherwise.

[0008] 5. **Context** – prior art communications tools do not know the business and/or personal context(s) within which files are created and used. For example, a person may create three files in a word processor, one relating to sales, the second relating to operations and the third relating to his son's football team. However, the word processor itself has no way of knowing to automatically store those three files in at least three different places.

[0009] 6. **Security & Privacy** – the applications and their file storage methods are generally insecure; they do not conform to a single, dependable security model.

[0010] Known software applications create and store files outside of a contextual framework. For example, when a user creates a Microsoft Word (*.doc) file in Microsoft Word 2000, the user must select a single folder within which to store that file. The file may be stored in an existing folder or the user may create a new folder to receive the file. This file management method is known as Lightweight Directory Application Protocol (LDAP). LDAP borrowed the physical world paper file management scheme where a machine/application creates files, stores those files in individual folders and stores those folders in cabinets. Under this scheme, context is completely independent of the application. File context is limited to the decision made by the user about which folder the file should be stored. The user decision does not adequately represent reflect the true context of the file given that the file may contain information that could reasonable be stored in multiple folders.

[0011] Another limitation of LDAP is that little or no information is contained within the file about the user and the context and circumstances of the user at the time the file was created. Current processes designed to add context to files such as the "meta-data" tagging approach, involve having a knowledge officer view files after they have been stored and create meta-data tags with additional key words associated with the file for search purposes.

[0012] Notwithstanding the usefulness of the above-described methods, a need still exists for a communications tool that associates files generated by applications with individuals, groups and topical context.

III. Summary of the Invention

[0013] It is an object of the invention to provide a communication tool that seamlessly facilitates, collects, compiles and distributes communication data.

[0014] It is a further object of the invention to provide a communication tool that links communication data to enterprise leadership priorities.

[0015] It is another objective of the invention to provide a communication tool that performs communications tasks while simultaneously reminding the user of his/her individual work priorities.

[0016] It is still a further object of the invention to provide a communication tool that automatically stores contextual information relating to an item of communication and utilizes that contextual in performance of communication tasks.

[0017] Sill another object of the invention is to provide a communication tool that integrates two or more different communication applications such as telephony, unified messaging, decision support, document management, portals, chat, collaboration, search, vote, relationship management, calendar, personal information management, profiling, directory management, executive information systems, dashboards, cockpits, tasking, meeting, conferencing, etc. into a common application.

[0018] Still a further object of the invention is to provide a structure for defining relationships between complex collections of data.

[0019] Yet another object of the invention is to provide a process for automating workflow between multiple entities.

[0020]

[0021] Given the following enabling description, the invention should become evident to a person of ordinary skill in the art.

IV. Description of the Embodiments

[0022] In the past, intuitive, dynamic, changeable workflow processes have proved to be too dynamic and expensive for automation. The present invention utilizes "boards" and "webs" to automate workflow processes and define relationships between data and applications. As users create and change their contexts, the files and applications automatically follow, dynamically capturing those shifts in context.

[0023] As used herein, a "board" is defined as a collection of data and application functionality related to a user-defined topic. For example, a user defined topic may be a department of a company or a project that involves the company. In the case of a project, the board preferably includes all of the data relating to that project including email, tasks, calendar events, ideas, discussions, meetings, phone calls, files, contact records, people, etc. Data and applications may be grouped in a board based on the identity of the tag.

[0024] As used herein, the term "web" refers to a collection of interelated boards. Boards in a web may have, for example, a parent-child relationship. A given board may have more than one parent and may have more than one child. A board may not be its own child or its own parent. However, boards may have various relationships to each other. For example, a board may be part of a circular relationship of any complexity such as the following: A is parent to B; B is parent to C and C is parent to A.

[0025] In accordance with the invention, webs may be used to maintain the location of content within a complex and changing set of boards and support automation of the

workflow process. Automation of the workflow process may shown by the following example.

Example

The workflow process to be automated is A→B→C. Three different people are assigned to each item. Therefore A(1,2,3) → B(4,5,6) → C(7,8,9). The workflow change desired in this example is A → B/C → C.

In the known environment, LDAP, it is necessary for the automation sequence to predetermine how work data flows from A to B and C. Then, the automation module for inputs to D must be spelled out and rewritten to consolidate split input from B and C. As such, the automation support for this workflow change will always lag behind the ability of the people involved to start working with the new workflow assumptions.

In contrast, in accordance with the present invention, webs and boards are preferably the context for applications, files and folders. Hence, the workflow process may be readily reorganized by making a change to one or more of the webs and boards.

In preferred embodiments, webs may be utilized to maintain the location of content within a complex and changing set of boards. Content is preferably associated with a routing algorithm referred to herein as a webslice. Thus the content has an intelligent quality whereby upon a change of structure of the web, the content knows which board or boards it should be on both before and after the change of structure. In keeping with a preferred aspect of the invention, the location of the content may be

determined at dynamically at run using the routing algorithm. Alternatively, the loction of content may be determined by detecting changes in structure, detecting the temporary location fo the content on the boards in the routing algorithm before and after the change and adjusting the location of the affected content as part of the change in structure.

ATTACHMENT 2

"board" Module

"WEB VERSION 1" WORKING DESCRIPTION

Webs are collections of boards and a collection of parent-child relationships between those boards. Boards in a web may have more than one parent and my have more than one child. A board may not be its own child (and thus may not be its own parent), but may participate in a circular relationship of any complexity (A is parent to B. B is parent to C. C is parent to A).
WebSlices are a way of representing an algorithm that's ultimate output is a set of boards. A webslice consists of a Web, a starting board, and a traversal (of arbitary complexity). Take for example a web of boards a b and c where b and c are children of a. A webslice that referenced this board, started at a and used a traversal of "all cihldren" would return b and c. If the smae traversal on the same web had started at b, the empty set would be the result.
Webs can be utilized to maintain the location of content within a complex and changing set of boards. If content has a webslice associated with it, then any change of structure in the web would still result in the content (with the webslice) knowing what boards it should be on both before and after the change of structure. Actually effecting this change of location can be done by allowing the "location" to be determined dynamically at run time using the webslice or can be accomplished by detecting changes in structure, detecting the (temporary) location of the content on the boards in the slice before and after the change and adjusting the location of the affected content as part of the change in web structure.

CIAP also facilitates a new business workflow process. Workflow automation is currently a site-specific effort. The workflow between A to B to C must be clearly specified in all its variables prior to automation. Automation fixes this workflow in code. Changes to the workflow require manual changes to the code. Predictable, repeatable, transactional and hierarchical workflow processes are best suited to this approach. LDAP and hierarchical storage models work best in this environment. Multiple applications work independently of the storage, generating and reporting data to and from the storage model.

Intuitive, dynamic, changeable workflow processes have proved too dynamic and expensive for automation. CIAP changes that. CIAP is key off users and context, not off of applications and files. As users create and change their contexts, the files and applications automatically follow, dynamically capturing those shifts of context.

Professional services consulting is currently held hostage by a cumbersome, expensive, time-consuming and often dehumanizing process known as "change management." The modus operandi of these firms is to for the implementation of that firm's change model. These models have a variety of names: Balanced Scorecard, Critical Success Factors, Vital Signs, etc. These models are often intended to replace traditional "command and control" models. Generally this is an either/or process. This change in the workflow practices in a company is time consuming. Generally these new processes begin a spate of new automation projects to support these changes. However, as any professional services person knows, the automation, like the change process itself, is iterative. Typically 50% of the changes initially championed will not work. Then 25% of the secondary changes will not work. Then, 12.5 of the third round of changes will not work... and so on. As a consequence, automation always lags behind, many times in terms of years.

CIAP allows professional services providers to support IT automation professionals with an approach to automation support of workflow changes that changes and adapts as the organization learns with little to no change to the underlying IT architecture.

To use a simple example, A → B → C is the workflow process we want to automate. We assign 3 different people to each item, Therefore A(1,2,3) → B(4,5,6) → C(7,8,9).

LDAP Implementation

Persons (1,2,3,4,5,6,7,8,9) → Applications → **A**files, **B**files, **C**files → **A**folders, **B**folders, **C**folders.

Now let's say a workflow change is proposed to look like this: A → B/C → D. In an LDAP environment, before the people involved have any automation support for this change, the automation sequence *pre-determine* how work data flows from A to B & C. Then, the automation module for inputs to D must be *spelled out and rewritten* to consolidate split input from B & C. In other words, the automation support for this change will always lag behind the ability of the people involved to start working with the new workflow assumptions. LDAP structure forces a regimented, minimalistic approach to the automation of workflow processes.

CIAP Implementation

Persons (1,2,3,4,5,6,7,8,9) → Web → **A**board,**B**board,**C**board (incl. Applications, Files, Folders)

Now let's say the workflow changes to A → B/C → D. In a CIAP environment a simple adjustment is made to the webs & boards table and the entire workflow process is reorganized with all the relevant data files appropriate reorganized and available. This should always be the first step in the change process. The first step in the change process should always be the instantaneous reorganization of the people and topic associations along with the communications tools. At this stage in the change, no predictable, repeatable, transactional or hierarchical process can be established. That can only come with time and consistency. Some processes must remain flexible, unpredictable, yet they

[redacted]

are processes nonetheless. CIAP allows for the simultaneous automation of repeatable and dynamic processes.

In CIAP, the People, Webs and Boards become the automatic context for Applications, Files and Folders. In LDAP the Applications, Files and Folders have *no* inherent relationship to the People or their Context. The implications of this difference on the automation of workflow process are profound.

Looking at the code for Web (my comments in []'s):

```
package com.leader.osapplication.board;

import java.util.*;
import com.leader.util.*;
import com.leader.debug.*;
import com.leader.persist.*;
import com.leader.persist.vbsf.*;
import com.leader.osapplication.*;
import com.leader.osapplication.field.*;
import com.leader.osapplication.util.*;
import com.leader.osapplication.actions.*;
import com.leader.osapplication.framework.*;
import com.leader.osapplication.exception.*;
import com.leader.osapplication.interfaces.*;
import com.leader.osapplication.sessionstate.*;

/**
 * A collections of boards with connected relationships tying them
together.
 * The stereotypical example is an org chart in a company where each
person is
 * a node on the web.
 *
 * @author Jeff R. Lamb
 * @author Betsy Foote
 * @author Eric Rosenberg
 */
public class Web extends Content {

  public static final String RELATIONSHIPS_LIST_FIELD_ID =
"existingRelationshipsList";
  public static final String CHILD_BOARD_FIELD_ID = "childBoard";
  public static final String PARENT_BOARD_FIELD_ID = "parentBoard";

[These are the relationships that make up the web. If a board
participates in any relationship in this collection, then they are part
of this web]

  private Collection relationships =
CollectionFactory.getPersistenceCapableCollection();

[Webs are named to allow them to be easy to work with for the users]

  private String name;
```

```
  /**
   * VBSF required no argument constructor.
   */
  private Web(){
    super();
  }

  /**
   * Constructor
   * @param name the name to give this Web
   */
  public Web(String name){
    this();
    this.name = name;
  }

  //CI
  public ContentInterface newContent(Map pairs, RequestState
requestState) throws LeaderException {
    return new Web(TextField.convert("name",pairs));
  }

  //CI
  public void setCurrentValues(Map pairs, RequestState requestState){
    if (pairs.containsKey("webNameTextField")){
      setName((String)pairs.get("webNameTextField"));
    }
  }

  //CI
  public String getValidForAddErrorMessage(){
    String errorMessage = null;
    if(getName() == null || "".equals(getName().trim())){
      errorMessage = "You must designate a name for your Web.";
    }
    return errorMessage;
  }

  //CI
  public int getContentToolCode(){
    return LeaderConstants.BOARD_WEB_TOOL;
  }

  /**SE*/
  public String getName(){
    return name;
  }

  /**SE*/
  public void setName(String name){
    this.name = name;
  }

  /**
   * Add a WebRelationship to the Web.
   * @param relationship The relationship to add.
```

[redacted]

```
     */
    public void addWebRelationship(WebRelationship relationship){
      if(relationship != null){
        relationships.add(relationship);
      }
    }

    /**
     * Remove a WebRelationship from the Web.
     * @param relationship The relationship to remove.
     */
    public void removeWebRelationship(WebRelationship relationship){
      if(relationship != null){
        relationships.remove(relationship);
      }
    }

    /**
     * Remove a WebRelationship from the Web.
     * @param relationshipId The object id of the relationship to remove.
     */
    public void removeWebRelationship(Long relationshipId){
      if(relationshipId != null){
        Iterator iterator = relationships.iterator();
        while(iterator.hasNext()){
          WebRelationship relationship =
(WebRelationship)iterator.next();
          if(relationshipId.equals(relationship.getId())){
            removeWebRelationship(relationship);
          }
        }
      }
    }

    /**
     * Get all the WebRelationships on this Web.  If there are no
relationships,
     * return a 0 length array.
     * @return WebRelationship array.
     */
    private WebRelationship[] getWebRelationships(){
      return (WebRelationship [])new ArrayList(relationships).toArray(new
WebRelationship[relationships.size()]); //WebRelationship
[])relationships.toArray(new WebRelationship[relationships.size()]);
    }

    /**
     * Determine whether a given board is in this web.
     * @param board Board we want to check on.
     * @return boolean True if board is in this web, false otherwise.
     */
    public boolean contains(Board board){
      List webBoards = getBoardsList();
      return webBoards.contains(board);
    }

    /**
```

```
    * Get all the board included in this Web.  If there are no
relationships,
    * and hence no boards, return an empty List.
    * @return Board[] Array of boards in this Web.
    */
   public List getBoardsList(){
     List boardList = new ArrayList();
     WebRelationship[] relations = getWebRelationships();
     for (int i=0; i < relations.length; i++){
       Board parent = relations[i].getParent();
       Board child = relations[i].getChild();
       if (!boardList.contains(parent)) boardList.add(parent);
       if (!boardList.contains(child)) boardList.add(child);
     }
     return boardList;
   }

   /**
    * Get all the Children of a Board on this Web.
    * @param board the board to find children of.
    * @return Set of children Boards.  0 size set if board parameter is
null
    * or when there are no children.
    */
   public Set getChildren(Board board){
     Set childrenSet = new HashSet();
     if(board == null){
       return childrenSet;
     }
     Iterator allRelationships = relationships.iterator();
     while (allRelationships.hasNext()){
       WebRelationship relationship =
(WebRelationship)allRelationships.next();
       if (relationship.getParent().getId().equals(board.getId())){
         childrenSet.add(relationship.getChild());
       }
     }
     return childrenSet;
   }

   /**
    * Get all the Parents of a Board on this Web.
    * @param board the board to find parents of.
    * @return Set of parent Boards.  0 size set if board parameter is
null
    * or when there are no parents.
    */
   public Set getParents(Board board){
     Set parentsSet = new HashSet();
     if(board == null){
       return parentsSet;
     }
     Iterator allRelationships = relationships.iterator();
     while (allRelationships.hasNext()){
       WebRelationship relationship =
(WebRelationship)allRelationships.next();
       if (relationship.getChild().getId().equals(board.getId())){
```

```
            parentsSet.add(relationship.getParent());
          }
        }
        return parentsSet;
      }

      /**
       * Get all the Peers (all children of all parents of the board).
       * @param board the board to find siblings of.
       * @return Set of Boards.  0 size set if board parameter is null
       * or when there are no peers.
       */
      public Set getPeers(Board board){
        Set childrenOfParents = new HashSet();
        if(board == null){
          return childrenOfParents;
        }
        Set parentBoards = getParents(board);
        Iterator parentBoardsIterator = parentBoards.iterator();
        while(parentBoardsIterator.hasNext()){
          Set children = getChildren((Board)parentBoardsIterator.next());
          childrenOfParents.addAll(children);
        }
        childrenOfParents.remove(board);
        return childrenOfParents;
      }

      //CI
      public Field[] getDisplayFields(RequestState requestState) throws
    LeaderException{
        List fields = new ArrayList();
        TextField textField = new TextField("name",getName(), "Web Name");
        textField.setLinkText("(Edit)");
        textField.setUrlId(LeaderConstants.BOARD_WEB_TOOL,""+getId());
        FieldUtilities.makeFieldAToolActivator(textField, requestState,
    this, getContentToolCode(),getContentToolCode());
        fields.add(textField);
        Field[] dateFields = DateField.getComponentFields(new
    DateTimeField(getLastModified()));
        dateFields[0].setTitle("Last Modified Date");
        fields.add(dateFields[0]);
        fields.add(dateFields[1]);
        return (Field[])fields.toArray(new Field[fields.size()]);
      }

      //CI
      public String getDisplayName(){
        return "Web";
      }

      //CI
      public Form getForm(RequestState requestState,int displayCode,int
    toolCode) {
        Debug.println("Web.getForm: for " + this, Debug.DEBUG);
        Form form = new ConcreteForm("webForm", "General Web Attributes");
        int pageIndex = 0;
        int selectedIndex = requestState.getMultiPageIndex();
```

[redacted]

```
        toolCode = getContentToolCode();

        //Web name sub-form.
        Page page = new ConcretePage("createWebPage", pageIndex,
selectedIndex);
        SubForm sub = new ConcreteSubForm("webNameSubForm","Web name");
        sub.add(new TextField("webNameTextField", (getName() != null ?
getName() : ""), "Web name", true));
        page.add(sub);

        //Existing relationships sub-form.
        sub = new ConcreteSubForm("existingWebRelationshipsSubForm",
"Existing Web Relationships");
        sub.add(getWebRelationshipsListField(requestState.getPairsMap()));

        InterfaceAction action = new
InterfaceAction("removeRelationship","Remove Relationship",toolCode,
true);

action.addActionListener(RemoveWebRelationshipActionListener.GLOBAL);
        action.addInterfaceListener(AddInterfaceListener.GLOBAL);
        action.setErrorInterfaceListener(AddInterfaceListener.GLOBAL);
        sub.addAction(action);
        page.add(sub);

        //Add new Relationships sub-form
        sub = new ConcreteSubForm("createRelationshipsSubForm", "Create New
Relationship");
        SingleSelectGroupKeyField boardDropDown = new
BoardKeyField(PARENT_BOARD_FIELD_ID, "Parent Board", null,
requestState.getCurrentUser().getId());
        sub.add(boardDropDown);
        boardDropDown = new BoardKeyField(CHILD_BOARD_FIELD_ID, "Child
Board", null, requestState.getCurrentUser().getId());
        sub.add(boardDropDown);
        action = new InterfaceAction("addRelationship","Add
Relationship",toolCode,true);
        action.addActionListener(AddWebRelationshipActionListener.GLOBAL);
        action.addInterfaceListener(AddInterfaceListener.GLOBAL);
        action.setErrorInterfaceListener(AddInterfaceListener.GLOBAL);
        sub.addAction(action);
        page.add(sub);

        form.add(page);
        return form;
    }

    /**VBSF*/
    private Collection getRelationshipsCollection(){
        return relationships;
    }

    /**VBSF*/
    private void setRelationshipsCollection(Collection collection){
        this.relationships = collection;
    }
```

[redacted]

```
  /**
   * Return a Field representing a list view of the web relationships
in this
   * web. This is used by the getForm method, and by the
MyContextInterface.
   * @param pairs SE
   * @return a Field
   */
  public Field getWebRelationshipsListField(Map pairs){
    Iterator iterator = relationships.iterator();
    List displayFieldsList = new ArrayList();
    Long[] keys = new Long[relationships.size()];
    for(int i=0; iterator.hasNext(); i++){
      WebRelationship relationship = (WebRelationship)iterator.next();
      keys[i] = relationship.getId();
      displayFieldsList.add(relationship.getDisplayFields());
    }
    Long[] selectedKeys =
MultiSelectListKeyField.convert(RELATIONSHIPS_LIST_FIELD_ID, pairs);
    Field[][] displayFields = (Field[][])displayFieldsList.toArray(new
Field[relationships.size()][0]);
    MultiSelectListKeyField relationshipsList = new
MultiSelectListKeyField(RELATIONSHIPS_LIST_FIELD_ID, keys, "Existing
Web Relationships", selectedKeys, displayFields);
    return relationshipsList;
  }
}
```

[END Web.java]

Looking at the code for WebSlice.java:

```
package com.leader.osapplication.board;

import com.leader.osapplication.framework.*;
import com.leader.osapplication.*;
import com.leader.osapplication.util.*;
import com.leader.osapplication.exception.*;
import com.leader.osapplication.sessionstate.*;
import com.leader.debug.*;
import java.util.*;

/**
 * A collection of enough information to isolate a set of boards from
the set
 * of all boards. This is typically codified as a Web to use, a
starting board
 * and a Traversal. The Traversal is then used to travel across the Web
from
 * the starting board and return a list of Boards.
 *
 * @author Jeff R. Lamb
 * @author Eric Rosenberg
 */
public class WebSlice extends AbstractPersistedObject{

  private Web web;
```

```
  private Board board;
  private Traversal traversal;

  /**VBSF*/
  private WebSlice(){
    super();
  }

  /**
   * Constructor
   * @param webToUse which Web is this WebSlice a slice of
   * @param boardToUse when you start moving around the Web, where do
you
   * start from?
   * @param traversalToUse what traversal (strategy) should be used to
   * move around the Web to carve out this WebSlice
   */
  public WebSlice(Web webToUse, Board boardToUse, Traversal
traversalToUse){
    this();
    setWeb(webToUse);
    setBoard(boardToUse);
    setTraversal(traversalToUse);
  }

  /**
   * Return the boards that are currently part of this webslice. This
can
   * change as the web that the webslice lies on is edited.
   * @return the boards that are a member of the slice
   */
  public Board[] getBoards(){
    return getTraversal().getBoards(web, board);
  }

  /**
   * Specify the web that that this webslice is taken from.
   * @param webToUse the web to use if coming up with the set of boards
the
   * web slice represents
   */
  public void setWeb(Web webToUse){ this.web = webToUse; }

  /**
   * Get the web that the webslice is taken from.
   * @return web that the web slice is a part of
   */
  public Web getWeb(){ return this.web;}

  /**
   * Specify the board that is the starting point for this webslice
   * @param boardToUse the board that is the starting point for the
webslice
   * @throws IllegalArgumentException if boardToUse is not in this web
   */
  public void setBoard(Board boardToUse){
```

```
    // These null checks are to bypass the 'contains' check when VBSF
may
    // be using this method with a null value or before setting web.
    if (boardToUse == null || web == null || web.contains(boardToUse)){
      this.board = boardToUse;
    }
    else{

      // throw an IllegalArgumentException if boardToUse is NOT in
      // webToUse.
      throw new IllegalArgumentException("The starting Board of a
WebSlice must be part of the Web.");
    }
  }

  /**
   * Get the board that is the starting point for the webslice
   * @return board that is the starting point for the webslice
   */
  public Board getBoard(){ return this.board;}

  /**
   * Specify the traversal used to get the boards for this webslice
   * @param traversalToUse SE
   */
  public void setTraversal(Traversal traversalToUse){
    this.traversal = traversalToUse;
  }

  /**
   * Get the traversal used to get the boards for this webslice
   * @return traversal used to get the boards for this webslice
   */
  public Traversal getTraversal(){ return this.traversal; }

  /**VBSF*/
  private int getTraversalCode() { return
TraversalFactory.getCode(traversal); }

  /**VBSF*/
  private void setTraversalCode(int code){ this.traversal =
TraversalFactory.getTraversal(code);}

}
```

[redacted]