

SUPRA BOUNDARY WEB COMPOSITOR APPARATUSES, METHODS AND SYSTEMS

[0001] This application for letters patent disclosure document describes inventive aspects that include various novel innovations (hereinafter “disclosure”) and contains material that is subject to copyright, mask work, and/or other intellectual property protection. The respective owners of such intellectual property have no objection to the facsimile reproduction of the disclosure by anyone as it appears in published Patent Office file/records, but otherwise reserve all rights.

PRIORITY CLAIM

[0002] Applicant hereby claims benefit to priority under 35 USC §119 as a non-provisional conversion of: US provisional patent application serial no. 62/396,796, filed September 19, 2016, entitled “Supra Boundary Web Compositor Apparatuses, Methods and Systems,” (attorney docket no. LINKSTORM0100PV); US provisional patent application serial no. 62/396,799, filed September 19, 2016, entitled “Supra Boundary Web Compositor Apparatuses, Methods and Systems,” (attorney docket no. LINKSTORM0100PV1).

[0003] The entire contents of the aforementioned applications are herein expressly incorporated by reference.

FIELD

[0004] The present innovations generally address web frame and graphic compositing, and more particularly, include Supra Boundary Web Compositor Apparatuses, Methods and Systems.

[0005] However, in order to develop a reader's understanding of the innovations, disclosures have been compiled into a single description to illustrate and clarify how aspects of these innovations operate independently, interoperate as between individual innovations, and/or

1 cooperate collectively. The application goes on to further describe the interrelations and
2 synergies as between the various innovations; all of which is to further compliance with 35
3 U.S.C. §112.

4 BACKGROUND

5 [0006] Web technologies allow for graphic display. Web servers host web page data that is
6 transferred to web browsers operated by users. The web browsers interpret the web page data
7 for rendering both graphics and text.

8 BRIEF DESCRIPTION OF THE DRAWINGS

9 [0007] Appendices and/or drawings illustrating various, non-limiting, example, innovative
10 aspects of the Supra Boundary Web Compositor Apparatuses, Methods and Systems
11 (hereinafter “SBWC”) disclosure, include:

12 [0008] FIGURES 1A-1B show a datagraph diagram illustrating embodiments of a data flow
13 for the SBWC;

14 [0009] FIGURE 2 shows a logic flow diagram illustrating embodiments of a frame escaping
15 (FE) component for the SBWC;

16 [0010] FIGURE 3 shows a logic flow diagram illustrating embodiments of a frame busting
17 (FB) component for the SBWC;

18 [0011] FIGURE 4 shows a logic flow diagram illustrating embodiments of a top level listening
19 (TLL) component for the SBWC;

20 [0012] FIGURE 5 shows a logic flow diagram illustrating embodiments of a frame querying
21 (FQ) component for the SBWC;

22 [0013] FIGURE 6 shows a logic flow diagram illustrating embodiments of an expandable
23 payload rebuilding (EPR) component for the SBWC;

24 [0014] FIGURE 7 shows a screenshot diagram illustrating embodiments of the SBWC;

1 [0015] FIGURE 8 shows a screenshot diagram illustrating embodiments of the SBWC; and
2 [0016] FIGURE 9 shows a block diagram illustrating embodiments of a SBWC controller.
3 [0017] Generally, the leading number of each citation number within the drawings indicates
4 the figure in which that citation number is introduced and/or detailed. As such, a detailed
5 discussion of citation number 101 would be found and/or introduced in Figure 1. Citation
6 number 201 is introduced in Figure 2, etc. Any citation and/or reference numbers are not
7 necessarily sequences but rather just example orders that may be rearranged and other orders
8 are contemplated.

DETAILED DESCRIPTION

1
2 **[0018]** The Supra Boundary Web Compositor Apparatuses, Methods and Systems (hereinafter
3 “SBWC”) transforms webpage request inputs, via SBWC components (e.g., FE, FB, TLL, FQ,
4 EPR, etc. components), into composited expandable element outputs. The SBWC
5 components, in various embodiments, implement advantageous features as set forth below.

Introduction

6
7 **[0019]** Advertisers on the Internet have utilized “Rich Media Expandable Ads” as one of the
8 many choices of ad formats available to them. According to the industry trade association the
9 IAB (Interactive Advertising Bureau), “Rich Media” is defined as “advertisements with which
10 users can interact (as opposed to solely animate) in a web page format.” Further, an
11 “Expandable” ad is one that increases in size beyond its original dimensions when it is selected
12 via a mouse-over or a click or a tap, so that it can present a larger footprint with additional
13 information or additional interactive functionality.

14 **[0020]** Expandable ads are very desirable for advertisers because they provide an ad presence
15 that can be made larger, more attention-getting, more engaging, and more interactive – and in
16 the case of SBWC, more useful to the customer because it can present a cascading navigation
17 menu (e.g., of hundreds of clickable choices) – which typically drives higher performance in
18 terms of customer engagement, click-through and conversion (e.g., online sales or lead-
19 generations). In SBWC’s case, these improvements have ranged from 2x to 17x in click-
20 throughs and conversions, as measured by many rigorous A/B tests (e.g., 50/50 alternating
21 rotations of a SBWC ad versus a competitor’s ad on the same websites, same page placements,
22 same times of day, etc.). See Figures 7 and 8 for examples of how navigable menus may expand
23 out of a preset advertising block on a web page.

24 **[0021]** However, one major obstacle to the widespread use of expandable ads is that many
25 “publisher websites” (e.g., websites that permit ads to appear on them) use a technique for
26 hosting ads known as an iframe, which limits the ad to remaining within its original borders and
27 prevents any expansion. When an expandable (e.g., JavaScript) ad is served in an iframe, it is

1 subject to the cross origin restrictions browsers have in place for security purposes. When a
2 JavaScript ad is served it is placed in a child iframe on a different domain (e.g., www.A.com)
3 than the parent publisher webpage (e.g., www.B.com). The two mismatched domains prevent a
4 JavaScript based ad from communicating with the parent publisher webpage. This security
5 feature of browsers prevents execution of JavaScript ad code on the parent publisher website.
6 The reason that many publishers employ iframes is to protect the user experience on their
7 website by restricting the ad from communicating technologically with the rest of the page,
8 thus preventing the ad from adversely affecting the publisher's page (e.g., by replacing the
9 publisher's content with unauthorized content). But an unintended consequence of this is that
10 the ad cannot expand, even if the user deliberately tries to initiate the expansion by mousing
11 over the ad or clicking on it or tapping on it.

12 **[0022]** Many websites do want to accept expandable ads, since their business models are ad-
13 supported and they know that many advertisers want to utilize expandable ads, so that the
14 more ad types they can accept, the more competition they will have amongst the ads bidding
15 for their available ad inventory, and the more money they can make. For example, a very large
16 number of the publisher sites that have signed up with Google's Ad Exchange ("AdX") have
17 declared to Google that they accept expandable ads – but unfortunately they have not actually
18 installed the software kit that Google provides them that will allow expandable ads to work
19 even though their sites use iframes. By some estimates, as many as 80% of Google's publishers
20 do not in fact support expandable ads.

21 **[0023]** The SBWC provides a more elegant solution for supporting expandable ads. In one
22 embodiment, the SBWC provides a mechanism that allows publishers to support expandable
23 ads without having to install anything locally (e.g., a file on the publisher's server) and facilitates
24 hosting of expandable ads (e.g., distributed by an ad exchange such as AdX, OpenX, Index
25 Exchange, AppNexus) without having to worry whether their iframes will thwart the
26 expansion functionality. This also saves ad exchanges the time and effort to enforce
27 compliance by mass-mailing reminders to all publishers, not to mention the problem that as a
28 practical matter it would be nearly impossible to achieve 100% compliance (e.g., every
29 publisher would have to install the file on every domain (e.g., www.B.com) and subdomain

1 (e.g., subdomain1.B.com, subdomain2.B.com, subdomain3.B.com) from which they serve
2 webpages).

3 **[0024]** One reason it is not practical (e.g., for Rich Media vendors, for advertisers, for ad
4 distributors/exchanges) to rely on publishers installing a file on their servers in a programmatic
5 advertising environment (a.k.a. “Real-Time Bidding” or “RTB” environment) is because in that
6 environment the advertiser seldom knows in advance where its ads are going to appear: instead
7 the ads are placed via an “exchange” environment (e.g., like a stock exchange), which
8 automatically matches bids and offers from many different sources and then places an ad
9 automatically on any website where the bid and the offer match each other (e.g., and where the
10 match otherwise conforms to the targeting parameters specified in advance by the parties
11 involved). Thus, unless all publishers install the file on their servers, the advertiser cannot be
12 sure that its expandable ad will be able to expand. Therefore, it is a significant breakthrough to
13 devise a mechanism that universally enables support for expandable ads without requiring any
14 deliberate action on the part of any publishers.

15 **[0025]** In one implementation, this mechanism is a centralized mechanism that allows ads to
16 expand because it allows an ad to communicate with the rest of the publisher’s page and
17 therefore expand beyond its original borders, instead of being locked into the iframe. For
18 example, it may be served centrally by an ad exchange through one or more existing
19 mechanisms (e.g., the iframe escaping solution code may be added to any of Google’s AdX
20 scripts publishers install on their pages when they want to serve Google ads, the iframe
21 escaping solution code may be added to the ad tag script Google creates within their publisher
22 console), and added (e.g., a short snippet of JavaScript code) to the top level frame of the
23 publisher’s webpage. It is secure because it may communicate with either a dedicated frame
24 buster domain or with an ad exchange domain (e.g., if the ad exchange is hosting the code) to
25 allow the iframe escape. That keeps communication within the trusted domains.

26 **[0026]** In one embodiment, the iframe escaping solution uses a technique to establish
27 communication with the parent publisher page called post messaging. Messages can be sent
28 between iframes on different domain names using post messages. In one implementation, the
29 publisher page may be listening for post messages from its child iframes (e.g., via the short
30 snippet of JavaScript code). The publisher page may trust the post messages it is receiving

1 because they are accepted if they are from a trusted domain name. Any post messages not
2 from the trusted domain name may be ignored. The post messages are sent back and forth
3 between the child iframe and the parent top level frame until the ad iframe is located on the
4 parent publisher page. Once the ad iframe has been identified it is then able to execute
5 JavaScript code on the publisher page as it may be trusted. In one implementation, the ad may
6 be rebuilt in the top level frame directly above (e.g., via the CSS z-index property) the iframe
7 on the publisher's webpage. Being on top of the iframe instead of inside it allows the ad to
8 expand.

9 [0027] The significance of the SBWC's extra-territorial compositing mechanism goes far
10 beyond ad distributors/exchanges (e.g., AppNexus, Google, Index Exchange, OpenX). This
11 approach could be utilized not only by vendors of expandable ads, but by information portals
12 and widget makers, or anyone who wants to provide data compositing to an area larger than
13 originally allocated.

14

SBWC

15 [0028] FIGURES 1A-1B show a datagraph diagram illustrating embodiments of a data flow
16 for the SBWC. In Figures 1A-1B, dashed lines indicate data flow elements that may be more
17 likely to be optional. In Figures 1A-1B, a client 102 may send a webpage request 121 to a
18 publisher server 107 to obtain a content page (e.g., a weather forecast, a news story). For
19 example, the client may be a desktop, a laptop, a tablet, a smartphone, and/or the like that is
20 executing a client application. In one implementation, the webpage request may include data
21 such as a request identifier, a content page identifier, and/or the like. In one embodiment, the
22 client may provide the following example webpage request, substantially in the form of a
23 (Secure) Hypertext Transfer Protocol ("HTTP(S)") POST message including eXtensible
24 Markup Language ("XML") formatted data, as provided below:

```
25 POST /authrequest.php HTTP/1.1
26 Host: www.server.com
27 Content-Type: Application/XML
28 Content-Length: 667
29 <?XML version = "1.0" encoding = "UTF-8"?>
30 <auth_request>
```

```

1      <timestamp>2020-12-31 23:59:59</timestamp>
2      <user_accounts_details>
3          <user_account_credentials>
4              <user_name>JohnDaDoeDoeDoooe@gmail.com</user_name>
5              <password>abc123</password>
6              //OPTIONAL <cookie>cookieID</cookie>
7              //OPTIONAL <digital_cert_link>www.mydigitalcertificate.com/
8 JohnDoeDaDoeDoe@gmail.com/mycertifcate.dc</digital_cert_link>
9              //OPTIONAL <digital_certificate>_DATA_</digital_certificate>
10         </user_account_credentials>
11     </user_accounts_details>
12     <client_details> //iOS Client with App and Webkit
13         //it should be noted that although several client details
14         //sections are provided to show example variants of client
15         //sources, further messages will include only on to save
16         //space
17         <client_IP>10.0.0.123</client_IP>
18         <user_agent_string>Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_1 like Mac OS
19 X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D201
20 Safari/9537.53</user_agent_string>
21         <client_product_type>iPhone6,1</client_product_type>
22         <client_serial_number>DNXXX1X1XXXX</client_serial_number>
23         <client_UDID>3XXXXXXXXXXXXXXXXXXXXD</client_UDID>
24         <client_OS>iOS</client_OS>
25         <client_OS_version>7.1.1</client_OS_version>
26         <client_app_type>app with webkit</client_app_type>
27         <app_installed_flag>true</app_installed_flag>
28         <app_name>SBWC.app</app_name>
29         <app_version>1.0 </app_version>
30         <app_webkit_name>Mobile Safari</client_webkit_name>
31         <client_version>537.51.2</client_version>
32     </client_details>
33     <client_details> //iOS Client with Webbrowser
34         <client_IP>10.0.0.123</client_IP>
35         <user_agent_string>Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_1 like Mac OS
36 X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D201
37 Safari/9537.53</user_agent_string>
38         <client_product_type>iPhone6,1</client_product_type>
39         <client_serial_number>DNXXX1X1XXXX</client_serial_number>
40         <client_UDID>3XXXXXXXXXXXXXXXXXXXXD</client_UDID>
41         <client_OS>iOS</client_OS>
42         <client_OS_version>7.1.1</client_OS_version>
43         <client_app_type>web browser</client_app_type>

```

```

1      <client_name>Mobile Safari</client_name>
2      <client_version>9537.53</client_version>
3  </client_details>
4  <client_details> //Android Client with Webbrowser
5      <client_IP>10.0.0.123</client_IP>
6      <user_agent_string>Mozilla/5.0 (Linux; U; Android 4.0.4; en-us; Nexus S
7  Build/IMM76D) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile
8  Safari/534.30</user_agent_string>
9      <client_product_type>Nexus S</client_product_type>
10     <client_serial_number>YXXXXXXXXZ</client_serial_number>
11     <client_UDID>XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</client_UDID>
12     <client_OS>Android</client_OS>
13     <client_OS_version>4.0.4</client_OS_version>
14     <client_app_type>web browser</client_app_type>
15     <client_name>Mobile Safari</client_name>
16     <client_version>534.30</client_version>
17 </client_details>
18 <client_details> //Mac Desktop with Webbrowser
19     <client_IP>10.0.0.123</client_IP>
20     <user_agent_string>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3)
21 AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3
22 Safari/537.75.14</user_agent_string>
23     <client_product_type>MacPro5,1</client_product_type>
24     <client_serial_number>YXXXXXXXXZ</client_serial_number>
25     <client_UDID>XXXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</client_UDID>
26     <client_OS>Mac OS X</client_OS>
27     <client_OS_version>10.9.3</client_OS_version>
28     <client_app_type>web browser</client_app_type>
29     <client_name>Mobile Safari</client_name>
30     <client_version>537.75.14</client_version>
31 </client_details>
32 <webpage_request>
33     <request_identifier>ID_request_1</request_identifier>
34
35     <content_page_identifier>https://www.publisher.com/webpage.html</content_page_
36 identifier>
37 </webpage_request>
38 </auth_request>
39

```

40 [0029] The publisher server may send a webpage response 125 with the requested content page
41 to the client. For example, the client may display the content page to a user of the client. The
42 content page may comprise a plurality of elements including a parent webpage top level (e.g., a

1 top level frame) 103 and a child frame (e.g., an iframe) 104. In one implementation, the
2 webpage top level may include a top level listening (TLL) component (e.g., a short snippet of
3 JavaScript code).

4 **[0030]** The frame may send an expandable payload request 129 to an ad server 108. For
5 example, this iframe may request an expandable display payload (e.g., an expandable ad) from a
6 URL specified in its src attribute (e.g., <https://www.ads.com/ad.php?adID=12345>).

7 **[0031]** The ad server may send an expandable payload response 133 with the requested
8 expandable ad page to the client. For example, the client may display the expandable ad page to
9 the user of the client in the frame. The expandable ad page may be restricted to remaining
10 within the borders of the iframe and may not be able to expand.

11 **[0032]** A frame escaping (FE) component 135 (e.g., provided via the expandable payload
12 response) may instantiate a frame buster (e.g., a child iframe of the frame) element 105, process
13 post messages from the webpage top level, send post messages to the frame buster, and/or the
14 like to facilitate the iframe escaping solution. See Figure 2 for additional details regarding the
15 FE component.

16 **[0033]** The frame buster may send a frame buster payload request 137 to a buster server 109.
17 For example, this iframe may request a frame buster payload (e.g., from a trusted frame buster
18 domain) from a URL specified in its src attribute (e.g.,
19 <https://www.FrameBuster.com/buster.html?adID=12345>). The buster server may send a
20 frame buster payload response 141 with the requested frame buster page to the client.

21 **[0034]** A frame busting (FB) component 145 (e.g., provided via the frame buster payload
22 response) may process post messages from the frame, send post messages to the webpage top
23 level, and/or the like to facilitate the iframe escaping solution. See Figure 3 for additional
24 details regarding the FB component.

25 **[0035]** The frame buster may send an expandable payload identifying message 149 to the
26 webpage top level to facilitate querying frames to determine which frame contains the
27 expandable ad. In one implementation, the expandable payload identifying message may be a
28 post message that includes data such as an origin (e.g., domain associated with the frame
29 sending the post message), target origin (e.g., domain associated with the frame that should

1 receive the post message), data (e.g., data to be sent to the target frame), and/or the like. For
 2 example, the data field may include a frame querying (FQ) component (e.g., which may include
 3 data regarding the frame such as a unique frame identifier, a source, and/or the like). See
 4 Figure 5 for additional details regarding the FQ component. In one embodiment, the frame
 5 buster may provide the following example expandable payload identifying message including
 6 XML-formatted data, as provided below:

```

7   <post_message>
8       <origin>https://www.FrameBuster.com</origin>
9       <targetOrigin>https://www.publisher.com</targetOrigin>
10      <data>executable FQ component</data>
11  </post_message>
12

```

13 **[0036]** The TLL component 153 may listen for post messages from the frame buster and/or
 14 execute components provided via post messages. See Figure 4 for additional details regarding
 15 the TLL component.

16 **[0037]** The webpage top level may send a frame query message 157 to the frame to facilitate
 17 determining which frame contains the expandable ad. In one implementation, the frame query
 18 message may be a post message that includes data such as an origin (e.g., domain associated
 19 with the frame sending the post message), target origin (e.g., domain associated with the frame
 20 that should receive the post message), data (e.g., data to be sent to the target frame), and/or
 21 the like. For example, the data field may include a NodeList identifier, a unique frame
 22 identifier, and/or the like. In one embodiment, the webpage top level may provide the
 23 following example frame query message including XML-formatted data, as provided below:

```

24  <post_message>
25      <origin>https://www.publisher.com</origin>
26      <targetOrigin>https://www.ads.com</targetOrigin>
27      <data>
28          <node_list_id>NodeList ID of the frame</node_list_id>
29          <unique_frame_id>unique frame ID of the frame</unique_frame_id>
30      </data>
31  </post_message>
32

```

33 **[0038]** The frame may send a frame identifying message 161 to the frame buster to provide the
 34 matching NodeList identifier of the frame to the frame buster. In one implementation, the

1 frame identifying message may be a post message that includes data such as an origin (e.g.,
 2 domain associated with the frame sending the post message), target origin (e.g., domain
 3 associated with the frame that should receive the post message), data (e.g., data to be sent to
 4 the target frame), and/or the like. For example, the data field may include a NodeList
 5 identifier, and/or the like. In one embodiment, the frame may provide the following example
 6 frame identifying message including XML-formatted data, as provided below:

```

7   <post_message>
8       <origin>https://www.ads.com</origin>
9       <targetOrigin>https://www.FrameBuster.com</targetOrigin>
10      <data>
11          <node_list_id>NodeList ID of the frame</node_list_id>
12      </data>
13  </post_message>
14

```

15 **[0039]** The frame buster may send an expandable payload data message 165 to the webpage
 16 top level to facilitate providing expandable display payload for the user. In one
 17 implementation, the expandable payload data message may be a post message that includes
 18 data such as an origin (e.g., domain associated with the frame sending the post message), target
 19 origin (e.g., domain associated with the frame that should receive the post message), data (e.g.,
 20 data to be sent to the target frame), and/or the like. For example, the data field may include an
 21 expandable payload rebuilding (EPR) component (e.g., which may include data regarding the
 22 frame). See Figure 6 for additional details regarding the EPR component. In one embodiment,
 23 the frame buster may provide the following example expandable payload data message
 24 including XML-formatted data, as provided below:

```

25   <post_message>
26       <origin>https://www.FrameBuster.com</origin>
27       <targetOrigin>https://www.publisher.com</targetOrigin>
28       <data>executable EPR component</data>
29   </post_message>
30

```

31 **[0040]** The webpage top level may send an expandable payload request 169 to the ad server to
 32 facilitate rebuilding the expandable ad in the webpage top level (e.g., above the frame). For
 33 example, if the expandable ad page was not cached, it may be retrieved from the ad server. In
 34 another example, additional ad data may be retrieved.

1 [0041] The ad server may send an expandable payload response 173 with the requested data to
 2 the client. For example, the client may display the expandable ad page to the user of the client
 3 in the webpage top level. The expandable ad page may expand (e.g., if the user tries to initiate
 4 the expansion by mousing over the expandable ad or clicking on it or tapping on it).

5 [0042] FIGURE 2 shows a logic flow diagram illustrating embodiments of a frame escaping
 6 (FE) component for the SBWC. In Figure 2, a frame escaping request may be obtained at 201.
 7 For example, the frame escaping request may be obtained when an initialization function of an
 8 expandable display payload (e.g., of an expandable ad (e.g., a JavaScript object)) calls the FE
 9 component to facilitate the iframe escaping solution.

10 [0043] A determination may be made at 205 whether to attempt to escape from a frame. For
 11 example, an attempt to escape may be made if it is determined that the expandable ad is in an
 12 iframe. In one implementation, the determination whether to attempt to escape from the frame
 13 may be implemented as follows:

```

14 //bool that determines whether to attempt to escape
15 var go = false;
16
17 //Determine the top level domain name and the first level nested iframe
18 //the expandable ad lives in on the top frame. This code can go 4 nested iframes
19 deep.
20 if (this.checkIframeAccess(parent)){
21     if (parent.parent == top){
22         this.parent_domain = window.location.protocol+"//"+
23             this.extractDomain(parent.document.referrer);
24         this.first_level_iframe = parent;
25         go = true
26     } else {
27         if (this.checkIframeAccess(parent.parent)){
28             if (parent.parent.parent == top){
29                 this.parent_domain = window.location.protocol+
30                     "://" + this.extractDomain(parent.parent.document.referrer);
31                 this.first_level_iframe = parent.parent;
32                 go = true
33             } else {
34                 if (this.checkIframeAccess(parent.parent.parent)){
35                     if (parent.parent.parent.parent == top){
36                         this.parent_domain = window.location.protocol+"//"+

```

```
1         this.extractDomain(parent.parent.parent.document.referrer);
2         this.first_level_iframe = parent.parent.parent;
3         go = true
4     }
5 }
6 }
7 }
8 }
9 }
10
11
12 this.checkIframeAccess = function(ifr) {
13     try {
14         var l = ifr.location.toString();
15         if(typeof l == 'undefined')
16             return false
17         else
18             return true;
19     } catch( e ) {
20         return false;
21     }
22 }
23
24
25 this.extractDomain = function(url){
26     var domain;
27     //find & remove protocol (http, ftp, etc.) and get domain
28     if (url.indexOf("://") > -1)
29         domain = url.split('/')[2];
30     else
31         domain = url.split('/')[0];
32
33     //find & remove port number
34     domain = domain.split(':')[0];
35     return domain;
36 }
```

38 [0044] If it is determined that escape should be attempted, a unique frame identifier of the
39 frame (e.g., an iframe that contains the expandable ad) may be generated at 209. For example,
40 the unique frame identifier may be a random number. In one implementation, the unique
41 frame identifier may be generated as follows:

```

1 //create a random number ID between 1 and a billion used to identify
2 //the iframe this ad is sitting in.
3 var iframe_id = Math.floor((Math.random() * 1000000000) + 1);
4 //set the id to the string type
5 iframe_id = iframe_id.toString();
6

```

7 **[0045]** A frame buster (e.g., a child iframe of the expandable ad iframe) may be instantiated at 8 213. For example, the frame buster may be instantiated to facilitate utilizing a FB component. 9 The frame buster may be provided (e.g., via the window.name attribute) with data such as the 10 unique frame identifier, the source of the first level expandable ad iframe, the domain of the 11 webpage top level, the source of the initialization function of the expandable ad object, and/or 12 the like. The src attribute of the frame buster may be set to a webpage at a trusted frame buster 13 domain containing code of the FB component. In one implementation, the frame buster may 14 be instantiated as follows:

```

15 //create a hidden iframe that will hold a buster iframe
16 this.buster_frame = document.createElement('iframe');
17
18 //Set the variables to send to the buster iframe in the window.name attribute.
19 //Use a custom |L-|||-S| delimiter to parse out the variables later in the buster
20 iframe.
21 this.buster_frame.name =
22     iframe_id+"|L-|||-S|" +
23     this.first_level_iframe.location.href+"|L-|||-S|" +
24     this.parent_domain+"|L-|||-S|" +
25     SB.initscript.src+"|L-|||-S|"; //SB is the expandable ad object containing
26 associated ad data and/or functions
27
28 //set the buster frame source to the buster domain and pass in the adID
29 this.buster_frame.src = window.location.protocol+
30     'http://www.FrameBuster.com/buster.html?adID='+SB.adID;
31
32 //add the buster iframe to the expandable ad iframe's body
33 document.body.appendChild(this.buster_frame);
34

```

35 **[0046]** An event listener for webpage top level post messages may be added (e.g., to the first 36 level expandable ad iframe) at 217. For example the event listener may process messages 37 received from the webpage top level. In one implementation, the event listener may be added 38 as follows:

```

1  //used for JavaScript scoping.
2  var that = this;
3
4  // First level nested child frame receives a post message from the webpage top
5  level
6  // Publisher frame. The ls$iframes NodeList id and iframe_id are passed into
7  // this listener. Look at the iframe_id passed in the message data and check for
8  // a match to this expandable ad's iframe_id.
9  this.first_level_iframe.addEventListener('message',function(e){
10     //split the data to get the frame id and the NodeList id
11     var data_parts = e.data.split("|");
12
13     //check if the iframe_id passed back from the webpage top level matches the
14     generated unique
15     //frame identifier. If there is a match, send the buster iframe a message with
16     the webpage top
17     //level ls$iframes NodeList id. Set the allowed target to www.FrameBuster.com
18     to prevent elements
19     //other than the buster iframe from accepting this message.
20     if (data_parts[1] == iframe_id)
21         that.buster_frame.contentWindow.postMessage(
22             data_parts[0], //webpage top level ls$iframes NodeList array id that
23             matches the ad's first level iframe
24             window.location.protocol+'//www.FrameBuster.com' //available to the buster
25             domain, but not other domains
26         );
27     });
28

```

29 **[0047]** A determination may be made at 221 whether a post message was received from the
30 webpage top level. For example, the event listener may be configured to process received post
31 messages (e.g., as shown in the example above). If a post message was not received, the FE
32 component may wait at 225 until a post message is received.

33 **[0048]** If a post message was received, a unique frame identifier specified in the post message
34 may be determined at 229. In one implementation, data provided in the post message may be
35 parsed (e.g., using JavaScript commands) to determine the unique frame identifier.

36 **[0049]** A determination may be made at 233 whether the specified unique frame identifier
37 matches the unique frame identifier of the iframe that contains the expandable ad. If the frame
38 identifiers match, a NodeList identifier of the first level expandable ad iframe may be

1 determined at 237. In one implementation, data provided in the post message may be parsed
2 (e.g., using JavaScript commands) to determine the NodeList identifier.

3 **[0050]** A post message with the determined NodeList identifier may be sent to the frame
4 buster at 241. In one implementation, the post message may be sent using the
5 window.postMessage() JavaScript function.

6 **[0051]** FIGURE 3 shows a logic flow diagram illustrating embodiments of a frame busting
7 (FB) component for the SBWC. In Figure 3, a frame busting request may be obtained at 301.
8 For example, the frame busting request may be obtained when a script of a frame buster
9 payload is executed (e.g., when a frame buster page is loaded into a frame buster iframe
10 associated with an expandable ad).

11 **[0052]** A unique frame identifier for a parent frame (e.g., an expandable ad iframe) may be
12 determined at 305. For example, the window.name attribute may be parsed (e.g., using
13 JavaScript commands) to determine the unique frame identifier. In one implementation, the
14 unique frame identifier may be determined as follows:

```
15 //Split up the window name attribute into a usable array. Do this by
16 //splitting the data with the custom |L-|||-S| delimiter.
17 var Data = self.name.split("|L-|||-S|");
18 var data_frame_id = Data[0];
19
```

20 **[0053]** A source for the parent frame (e.g., a first level expandable ad iframe) may be
21 determined at 309. For example, the window.name attribute may be parsed (e.g., using
22 JavaScript commands) to determine the source. In one implementation, the source may be
23 determined as follows:

```
24 var data_src = Data[1];
25
```

26 **[0054]** A webpage top level domain may be determined at 313. For example, the window.name
27 attribute may be parsed (e.g., using JavaScript commands) to determine the domain of the
28 webpage top level. In one implementation, the webpage top level domain may be determined
29 as follows:

```
30 var data_top_level_domain = Data[2];
31
```

1 **[0055]** An executable frame querying (FQ) component may be generated at 317. For example,
 2 the FQ component may facilitate querying frames to determine which frame contains the
 3 expandable ad. In one implementation, the FQ component may be generated as follows:

```

4 //Create a string of JavaScript code to send to the webpage top level Publisher
5 //frame. This code (FQ component) is executed when the Publisher frame receives
6 the message.
7 var passed_escape_js_code =
8 //set the LSESCAPE variable to tag this code as buster code
9 "var LSESCAPE = '";
10 //set the src variable to the source of the first level iframe the expandable
11 ad is living in.
12 "var src = '"+data_src+"'";
13 //loop through the ls$iframes NodeList set in a TLL component
14 "for (var node_list_id = 0; node_list_id < ls$iframes.length; node_list_id++)
15 {"+
16 //Check if the source of the first level iframe matches the source of the
17 iframe
18 //in this loop iteration. Continue to next iteration if the source does not
19 match.
20 //Do this to prevent unwanted messages from going to iframes that do not
21 contain the expandable ad.
22 "if (src != ls$iframes[node_list_id].src) continue;"+
23 //send the node_list id and the unique frame identifier back into a child
24 iframe to determine
25 //if this iteration is the expandable ad iframe.
26 "ls$iframes[node_list_id].contentWindow.postMessage("+
27 "node_list_id+'|'+data_frame_id+'', '*' "+
28 "); "+
29 "}";
30
```

31 **[0056]** A post message with the generated FQ component may be sent to the webpage top
 32 level at 321. In one implementation, the post message may be sent using the
 33 window.postMessage() JavaScript function as follows:

```

34 //send the post message to the webpage top level Publisher Frame
35 //set the target origin to the publisher domain name to prevent other elements
36 from receiving the message
37 window.top.postMessage(passed_escape_js_code, data_top_level_domain);
38
```

1 [0057] An event listener for frame post messages may be added (e.g., to the buster iframe) at
 2 325. For example the event listener may process messages received from the parent frame. In
 3 one implementation, the event listener may be added as follows:

```

4   var data_expandable_ad_initscript = Data[3];
5
6   /**
7    * Add a Post Message Listener to the buster iframe.
8    * This listener is waiting for a message telling it the iframe was located.
9    * The data value passed into this listener is the ls$iframes NodeList id that
10  was matched.
11  */
12  window.addEventListener('message',function(e){
13      //Create a string of JavaScript code to send to the webpage top level Publisher
14      //frame. This code (EPR component) is executed when the Publisher frame
15  receives the message.
16      var passed_escape_js_code =
17          //set the LSESCAPE variable to tag this code as buster code.
18          "var LSESCAPE = '";
19          //set the node_list_id from the message data sent in from the parent frame
20          "var node_list_id = '"+e.data+"'";
21          //set a random id used by the expandable ad to find itself when being added
22  to a page
23          "var new_id = Math.floor((Math.random() * 10000000) + 1);"+
24          //create a JavaScript HTML script tag
25          "var s = window.top.document.createElement('script');"+
26          //set the type to text/javascript
27          "s.type='text/javascript';"+
28          //give this script a classname of ls_escaped - used by the expandable ad to
29  determine it has
30          //escaped an iframe.
31          "s.className = 'ls_escaped';"+
32          "s.setAttribute('escaped', 'true');"+
33          "s.id = new_id;"+
34          //set the src of the script tag to the expandable ad Init File
35          "var src = '"+data_expandable_ad_initscript+"'";
36          "src += '&friendly=false&escape_id='+new_id;"+
37          "s.src = src;"+
38          //Using the node_list_id add the expandable ad Init Script in front of the
39  iframe
40          //that matched in the ls$iframes NodeList the expandable ad was originally
41  placed in. The expandable

```

```

1      //ad then executes its normal process on the webpage top level Publisher
2      frame and is able to load its menu
3      //for full expansion.
4
5      "ls$iframes[node_list_id].parentNode.insertBefore(s,ls$iframes[node_list_id]);";
6
7      //send the post message to the webpage top level Publisher Frame
8      //set the target origin to the publisher domain name to prevent other elements
9      from receiving the message
10     window.top.postMessage(passed_escape_js_code, data_top_level_domain);
11     });
12

```

13 **[0058]** A determination may be made at 329 whether a post message was received from the
14 parent frame. For example, the event listener may be configured to process received post
15 messages (e.g., as shown in the example above). If a post message was not received, the FB
16 component may wait at 333 until a post message is received.

17 **[0059]** If a post message was received, a NodeList identifier for the parent frame specified in
18 the post message may be determined at 337. In one implementation, data provided in the post
19 message may be parsed (e.g., using JavaScript commands) to determine the NodeList identifier
20 for the parent frame.

21 **[0060]** Expandable payload data may be determined at 341. In one implementation, the source
22 of the expandable ad initialization script may be determined. For example, the window.name
23 attribute may be parsed (e.g., using JavaScript commands) to determine the source of the
24 expandable ad initialization script. In another implementation, other expandable ad attributes
25 (e.g., passed in from the parent frame and utilized for click tracking) may be determined.

26 **[0061]** An executable expandable payload rebuilding (EPR) component may be generated at
27 345. For example, the EPR component may facilitate rebuilding the expandable ad in the
28 webpage top level (e.g., above the parent frame).

29 **[0062]** A post message with the generated EPR component may be sent to the webpage top
30 level at 349. In one implementation, the post message may be sent using the
31 window.postMessage() JavaScript function.

32 **[0063]** FIGURE 4 shows a logic flow diagram illustrating embodiments of a top level listening
33 (TLL) component for the SBWC. In Figure 4, a top level listening request may be obtained at

1 401. For example, the top level listening request may be obtained when a script of a webpage
2 top level is executed (e.g., when the webpage top level is loaded).

3 **[0064]** A NodeList object of first level frames may be created at 405. For example, the
4 NodeList object may include first level iframes located on the webpage top level. In one
5 implementation, the NodeList object may be created as follows:

```
6 // Create a HTML NodeList object of the iframes located on the webpage top level
7 // Publisher page.
8 var ls$iframes = window.top.document.getElementsByTagName('iframe');
9
```

10 **[0065]** An event listener for frame buster post messages (e.g., from the buster domain) may be
11 added (e.g., to the webpage top level) at 409. For example, the event listener may process
12 messages received from a frame buster. In one implementation, the event listener may be
13 added as follows:

```
14 /**
15  * Add a Post Message Listener to the webpage top level Publisher page.
16  * This function is listening for messages from nested cross origin iframes
17  * associated with expandable ads.
18  */
19 window.addEventListener('message',function(e){
20     //Check the origin of the post message to verify that the post message is from
21     the www.FrameBuster.com
22     //domain name. This code prevents unauthorized use of this message listener.
23     if (e.origin != window.location.protocol+"//www.FrameBuster.com") {
24         //stop processing if the origin of the message was not www.FrameBuster.com
25         return;
26     }
27
28     //check if the data received in the message payload is of the type string
29     //stop processing if is not a string
30     if (typeof e.data != 'string') {
31         return;
32     }
33
34     //check if the message contains the string "LSEscape"
35     //this prevents processing of post messages happening within the expandable ads
36     //themselves that do not relate to iframe escaping.
37     if (e.data.indexOf("LSEscape") == -1) {
38         return;
```

```

1      }
2
3      //Turn the post message data into an executable JavaScript function.
4      var ls$escape = Function(e.data);
5
6      //Run the new JavaScript Function.
7      ls$escape();
8  });
9

```

10 **[0066]** A determination may be made at 413 whether a post message was received from a
 11 frame buster. For example, the event listener may be configured to process received post
 12 messages (e.g., as shown in the example above). If a post message was not received, the TLL
 13 component may wait at 417 until a post message is received.

14 **[0067]** If a post message was received, its origin may be checked to verify that it originated
 15 from the buster domain at 421. In one implementation, origin data provided in the post
 16 message may be checked against a specified buster domain name (e.g., www.FrameBuster.com)
 17 to verify the origin of the post message.

18 **[0068]** An executable component (e.g., FQ component, EPR component) provided in the post
 19 message may be executed at 425. In one implementation, the executable component may be
 20 turned into an executable JavaScript function using the Function() JavaScript function, and the
 21 resulting executable JavaScript function may be executed. See Figure 5 for additional details
 22 regarding the FQ component. See Figure 6 for additional details regarding the EPR
 23 component.

24 **[0069]** FIGURE 5 shows a logic flow diagram illustrating embodiments of a frame querying
 25 (FQ) component for the SBWC. In Figure 5, the source of the first level expandable ad iframe
 26 associated with the frame buster that sent the post message may be determined at 501. In one
 27 implementation, the source of the first level expandable ad iframe may be specified in the post
 28 message.

29 **[0070]** A determination may be made at 505 whether there remain first level frames to process.
 30 In one implementation, each of the first level iframes in the NodeList object may be processed.
 31 If there remain first level iframes to process, the next first level iframe in the NodeList object
 32 may be selected at 509.

1 [0071] A determination may be made at 513 whether the source of the selected first level
2 iframe matches the source of the first level expandable ad iframe. If it is determined that the
3 frame sources match, a new post message with the NodeList identifier (e.g., item index) of the
4 selected first level iframe and the unique frame identifier associated with the first level
5 expandable ad iframe (e.g., specified in the post message) may be sent to the selected first level
6 iframe. In one implementation, the post message may be sent using the `window.postMessage()`
7 JavaScript function.

8 [0072] FIGURE 6 shows a logic flow diagram illustrating embodiments of an expandable
9 payload rebuilding (EPR) component for the SBWC. In Figure 6, an element (e.g., a JavaScript
10 HTML script tag) for the expandable display payload (e.g., the expandable ad) associated with
11 the frame buster that sent the post message may be created at 601. In one implementation, the
12 element may be created using the `document.createElement()` JavaScript function.

13 [0073] The element may be configured as an escaped element at 605. For example, configuring
14 the element as an escaped element may inform the expandable ad (e.g., an expandable ad
15 JavaScript object containing associated ad data and/or functions) that it has escaped its iframe
16 and is being created on the webpage top level. In one implementation, the element may be
17 configured as an escaped element based on a class name (e.g., `ls_escaped`). In another
18 implementation, the element may be configured as an escaped element based on an attribute
19 value (e.g., `escaped = true`).

20 [0074] A determination may be made at 609 whether the expandable display payload (e.g., data
21 associated with the expandable ad) is cached. If the data is cached, the cached data may be
22 determined at 613. For example, the expandable ad object containing associated ad data
23 and/or functions may be identified. If the data is not cached, expandable ad data may be
24 obtained from an ad server at 617. For example, an expandable ad object may be obtained via
25 an expandable payload request.

26 [0075] The expandable ad may be provided for the user at 621. In one implementation, the
27 source of the element may be set to the location of the expandable ad object initialization
28 script and/or the element may be added to the webpage top level. The expandable ad may be
29 rendered using the element, which is on the webpage top level, and the expandable ad may

1 expand (e.g., if a user tries to initiate the expansion by mousing over the expandable ad or
2 clicking on it or tapping on it).

3 **[0076]** FIGURE 7 shows a screenshot diagram illustrating embodiments of the SBWC. In
4 figure 7, screen 701 shows an example of an expandable ad for Home Depot on weather.com
5 prior to expansion. Screen 710 shows the same ad after expansion. Widget 712 illustrates that
6 the expanded ad may provide a user with a multilevel menu that allows the user to browse
7 products, make purchases, utilize a store finder, and/or the like.

8 **[0077]** FIGURE 8 shows a screenshot diagram illustrating embodiments of the SBWC. In
9 Figure 8, screen 801 shown an example of an expandable ad for Kinky Boots on nytimes.com
10 prior to expansion. Screen 810 shows the same ad after expansion. Widget 812 illustrates that
11 the expanded ad may provide a user with a set of media player controls for an associated media
12 (e.g., a preview video) that allows the user to pause, play, rewind, fast forward, make full
13 screen, mute, and/or the like the associated media.

14 SBWC Controller

15 **[0078]** FIGURE 9 shows a block diagram illustrating embodiments of a SBWC controller. In
16 this embodiment, the SBWC controller 901 may serve to aggregate, process, store, search,
17 serve, identify, instruct, generate, match, and/or facilitate interactions with a computer through
18 web frame and graphic compositing technologies, and/or other related data.

19 **[0079]** Typically, users, which may be people and/or other systems, may engage information
20 technology systems (e.g., computers) to facilitate information processing. In turn, computers
21 employ processors to process information; such processors 903 may be referred to as central
22 processing units (CPU). One form of processor is referred to as a microprocessor. CPUs use
23 communicative circuits to pass binary encoded signals acting as instructions to enable various
24 operations. These instructions may be operational and/or data instructions containing and/or
25 referencing other instructions and data in various processor accessible and operable areas of
26 memory 929 (e.g., registers, cache memory, random access memory, etc.). Such communicative
27 instructions may be stored and/or transmitted in batches (e.g., batches of instructions) as
28 programs and/or data components to facilitate desired operations. These stored instruction

1 codes, e.g., programs, may engage the CPU circuit components and other motherboard and/or
2 system components to perform desired operations. One type of program is a computer
3 operating system, which, may be executed by CPU on a computer; the operating system
4 enables and facilitates users to access and operate computer information technology and
5 resources. Some resources that may be employed in information technology systems include:
6 input and output mechanisms through which data may pass into and out of a computer;
7 memory storage into which data may be saved; and processors by which information may be
8 processed. These information technology systems may be used to collect data for later
9 retrieval, analysis, and manipulation, which may be facilitated through a database program.
10 These information technology systems provide interfaces that allow users to access and operate
11 various system components.

12 **[0080]** In one embodiment, the SBWC controller 901 may be connected to and/or
13 communicate with entities such as, but not limited to: one or more users from peripheral
14 devices 912 (e.g., user input devices 911); an optional cryptographic processor device 928;
15 and/or a communications network 913.

16 **[0081]** Networks are commonly thought to comprise the interconnection and interoperation of
17 clients, servers, and intermediary nodes in a graph topology. It should be noted that the term
18 “server” as used throughout this application refers generally to a computer, other device,
19 program, or combination thereof that processes and responds to the requests of remote users
20 across a communications network. Servers serve their information to requesting “clients.” The
21 term “client” as used herein refers generally to a computer, program, other device, user and/or
22 combination thereof that is capable of processing and making requests and obtaining and
23 processing any responses from servers across a communications network. A computer, other
24 device, program, or combination thereof that facilitates, processes information and requests,
25 and/or furthers the passage of information from a source user to a destination user is
26 commonly referred to as a “node.” Networks are generally thought to facilitate the transfer of
27 information from source points to destinations. A node specifically tasked with furthering the
28 passage of information from a source to a destination is commonly called a “router.” There are
29 many forms of networks such as Local Area Networks (LANs), Pico networks, Wide Area
30 Networks (WANs), Wireless Networks (WLANs), etc. For example, the Internet is generally

1 accepted as being an interconnection of a multitude of networks whereby remote clients and
2 servers may access and interoperate with one another.

3 [0082] The SBWC controller 901 may be based on computer systems that may comprise, but
4 are not limited to, components such as: a computer systemization 902 connected to memory
5 929.

6 Computer Systemization

7 [0083] A computer systemization 902 may comprise a clock 930, central processing unit
8 (“CPU(s)” and/or “processor(s)” (these terms are used interchangeable throughout the
9 disclosure unless noted to the contrary)) 903, a memory 929 (e.g., a read only memory (ROM)
10 906, a random access memory (RAM) 905, etc.), and/or an interface bus 907, and most
11 frequently, although not necessarily, are all interconnected and/or communicating through a
12 system bus 904 on one or more (mother)board(s) 902 having conductive and/or otherwise
13 transportive circuit pathways through which instructions (e.g., binary encoded signals) may
14 travel to effectuate communications, operations, storage, etc. The computer systemization may
15 be connected to a power source 986; e.g., optionally the power source may be internal.
16 Optionally, a cryptographic processor 926 may be connected to the system bus. In another
17 embodiment, the cryptographic processor, transceivers (e.g., ICs) 974, and/or sensor array
18 (e.g., accelerometer, altimeter, ambient light, barometer, global positioning system (GPS)
19 (thereby allowing SBWC controller to determine its location), gyroscope, magnetometer,
20 pedometer, proximity, ultra-violet sensor, etc.) 973 may be connected as either internal and/or
21 external peripheral devices 912 via the interface bus I/O 908 (not pictured) and/or directly via
22 the interface bus 907. In turn, the transceivers may be connected to antenna(s) 975, thereby
23 effectuating wireless transmission and reception of various communication and/or sensor
24 protocols; for example the antenna(s) may connect to various transceiver chipsets (depending
25 on deployment needs), including: Broadcom BCM4329FKUBG transceiver chip (e.g.,
26 providing 802.11n, Bluetooth 2.1 + EDR, FM, etc.); a Broadcom BCM4752 GPS receiver with
27 accelerometer, altimeter, GPS, gyroscope, magnetometer; a Broadcom BCM4335 transceiver
28 chip (e.g., providing 2G, 3G, and 4G long-term evolution (LTE) cellular communications;
29 802.11ac, Bluetooth 4.0 low energy (LE) (e.g., beacon features)); a Broadcom BCM43341

1 transceiver chip (e.g., providing 2G, 3G and 4G LTE cellular communications; 802.11 g/,
2 Bluetooth 4.0, near field communication (NFC), FM radio); an Infineon Technologies X-Gold
3 618-PMB9800 transceiver chip (e.g., providing 2G/3G HSDPA/HSUPA communications); a
4 MediaTek MT6620 transceiver chip (e.g., providing 802.11a/ac/b/g/n, Bluetooth 4.0 LE, FM,
5 GPS; a Lapis Semiconductor ML8511 UV sensor; a maxim integrated MAX44000 ambient
6 light and infrared proximity sensor; a Texas Instruments WiLink WL1283 transceiver chip
7 (e.g., providing 802.11n, Bluetooth 3.0, FM, GPS); and/or the like. The system clock typically
8 has a crystal oscillator and generates a base signal through the computer systemization's circuit
9 pathways. The clock is typically coupled to the system bus and various clock multipliers that
10 will increase or decrease the base operating frequency for other components interconnected in
11 the computer systemization. The clock and various components in a computer systemization
12 drive signals embodying information throughout the system. Such transmission and reception
13 of instructions embodying information throughout a computer systemization may be
14 commonly referred to as communications. These communicative instructions may further be
15 transmitted, received, and the cause of return and/or reply communications beyond the instant
16 computer systemization to: communications networks, input devices, other computer
17 systemizations, peripheral devices, and/or the like. It should be understood that in alternative
18 embodiments, any of the above components may be connected directly to one another,
19 connected to the CPU, and/or organized in numerous variations employed as exemplified by
20 various computer systems.

21 **[0084]** The CPU comprises at least one high-speed data processor adequate to execute
22 program components for executing user and/or system-generated requests. The CPU is often
23 packaged in a number of formats varying from large supercomputer(s) and mainframe(s)
24 computers, down to mini computers, servers, desktop computers, laptops, thin clients (e.g.,
25 Chromebooks), netbooks, tablets (e.g., Android, iPads, and Windows tablets, etc.), mobile
26 smartphones (e.g., Android, iPhones, Nokia, Palm and Windows phones, etc.), wearable
27 device(s) (e.g., watches, glasses, goggles (e.g., Google Glass), etc.), and/or the like. Often, the
28 processors themselves will incorporate various specialized processing units, such as, but not
29 limited to: integrated system (bus) controllers, memory management control units, floating
30 point units, and even specialized processing sub-units like graphics processing units, digital

1 signal processing units, and/or the like. Additionally, processors may include internal fast
2 access addressable memory, and be capable of mapping and addressing memory 929 beyond
3 the processor itself; internal memory may include, but is not limited to: fast registers, various
4 levels of cache memory (e.g., level 1, 2, 3, etc.), RAM, etc. The processor may access this
5 memory through the use of a memory address space that is accessible via instruction address,
6 which the processor can construct and decode allowing it to access a circuit path to a specific
7 memory address space having a memory state. The CPU may be a microprocessor such as:
8 AMD's Athlon, Duron and/or Opteron; Apple's A series of processors (e.g., A5, A6, A7, A8,
9 etc.); ARM's application, embedded and secure processors; IBM and/or Motorola's
10 DragonBall and PowerPC; IBM's and Sony's Cell processor; Intel's 80X86 series (e.g., 80386,
11 80486), Pentium, Celeron, Core (2) Duo, i series (e.g., i3, i5, i7, etc.), Itanium, Xeon, and/or
12 XScale; Motorola's 680X0 series (e.g., 68020, 68030, 68040, etc.); and/or the like processor(s).
13 The CPU interacts with memory through instruction passing through conductive and/or
14 transportive conduits (e.g., (printed) electronic and/or optic circuits) to execute stored
15 instructions (i.e., program code) according to conventional data processing techniques. Such
16 instruction passing facilitates communication within the SBWC controller and beyond through
17 various interfaces. Should processing requirements dictate a greater amount speed and/or
18 capacity, distributed processors (e.g., see Distributed SBWC below), mainframe, multi-core,
19 parallel, and/or super-computer architectures may similarly be employed. Alternatively, should
20 deployment requirements dictate greater portability, smaller mobile devices (e.g., Personal
21 Digital Assistants (PDAs)) may be employed.

22 **[0085]** Depending on the particular implementation, features of the SBWC may be achieved by
23 implementing a microcontroller such as CAST's R8051XC2 microcontroller; Intel's MCS 51
24 (i.e., 8051 microcontroller); and/or the like. Also, to implement certain features of the SBWC,
25 some feature implementations may rely on embedded components, such as: Application-
26 Specific Integrated Circuit ("ASIC"), Digital Signal Processing ("DSP"), Field Programmable
27 Gate Array ("FPGA"), and/or the like embedded technology. For example, any of the SBWC
28 component collection (distributed or otherwise) and/or features may be implemented via the
29 microprocessor and/or via embedded components; e.g., via ASIC, coprocessor, DSP, FPGA,
30 and/or the like. Alternately, some implementations of the SBWC may be implemented with

1 embedded components that are configured and used to achieve a variety of features or signal
2 processing.

3 [0086] Depending on the particular implementation, the embedded components may include
4 software solutions, hardware solutions, and/or some combination of both hardware/software
5 solutions. For example, SBWC features discussed herein may be achieved through
6 implementing FPGAs, which are a semiconductor devices containing programmable logic
7 components called "logic blocks", and programmable interconnects, such as the high
8 performance FPGA Virtex series and/or the low cost Spartan series manufactured by Xilinx.
9 Logic blocks and interconnects can be programmed by the customer or designer, after the
10 FPGA is manufactured, to implement any of the SBWC features. A hierarchy of
11 programmable interconnects allow logic blocks to be interconnected as needed by the SBWC
12 system designer/administrator, somewhat like a one-chip programmable breadboard. An
13 FPGA's logic blocks can be programmed to perform the operation of basic logic gates such as
14 AND, and XOR, or more complex combinational operators such as decoders or mathematical
15 operations. In most FPGAs, the logic blocks also include memory elements, which may be
16 circuit flip-flops or more complete blocks of memory. In some circumstances, the SBWC may
17 be developed on regular FPGAs and then migrated into a fixed version that more resembles
18 ASIC implementations. Alternate or coordinating implementations may migrate SBWC
19 controller features to a final ASIC instead of or in addition to FPGAs. Depending on the
20 implementation all of the aforementioned embedded components and microprocessors may be
21 considered the "CPU" and/or "processor" for the SBWC.

22

Power Source

23 [0087] The power source 986 may be of any standard form for powering small electronic
24 circuit board devices such as the following power cells: alkaline, lithium hydride, lithium ion,
25 lithium polymer, nickel cadmium, solar cells, and/or the like. Other types of AC or DC power
26 sources may be used as well. In the case of solar cells, in one embodiment, the case provides an
27 aperture through which the solar cell may capture photonic energy. The power cell 986 is
28 connected to at least one of the interconnected subsequent components of the SBWC thereby
29 providing an electric current to all subsequent components. In one example, the power source
30 986 is connected to the system bus component 904. In an alternative embodiment, an outside

1 power source 986 is provided through a connection across the I/O 908 interface. For example,
2 a USB and/or IEEE 1394 connection carries both data and power across the connection and
3 is therefore a suitable source of power.

4 Interface Adapters

5 [0088] Interface bus(es) 907 may accept, connect, and/or communicate to a number of
6 interface adapters, conventionally although not necessarily in the form of adapter cards, such
7 as but not limited to: input output interfaces (I/O) 908, storage interfaces 909, network
8 interfaces 910, and/or the like. Optionally, cryptographic processor interfaces 927 similarly
9 may be connected to the interface bus. The interface bus provides for the communications of
10 interface adapters with one another as well as with other components of the computer
11 systemization. Interface adapters are adapted for a compatible interface bus. Interface adapters
12 conventionally connect to the interface bus via a slot architecture. Conventional slot
13 architectures may be employed, such as, but not limited to: Accelerated Graphics Port (AGP),
14 Card Bus, (Extended) Industry Standard Architecture ((E)ISA), Micro Channel Architecture
15 (MCA), NuBus, Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express,
16 Personal Computer Memory Card International Association (PCMCIA), and/or the like.

17 [0089] Storage interfaces 909 may accept, communicate, and/or connect to a number of
18 storage devices such as, but not limited to: storage devices 914, removable disc devices, and/or
19 the like. Storage interfaces may employ connection protocols such as, but not limited to:
20 (Ultra) (Serial) Advanced Technology Attachment (Packet Interface) ((Ultra) (Serial) ATA(PI)),
21 (Enhanced) Integrated Drive Electronics ((E)IDE), Institute of Electrical and Electronics
22 Engineers (IEEE) 1394, fiber channel, Small Computer Systems Interface (SCSI), Universal
23 Serial Bus (USB), and/or the like.

24 [0090] Network interfaces 910 may accept, communicate, and/or connect to a
25 communications network 913. Through a communications network 913, the SBWC controller
26 is accessible through remote clients 933b (e.g., computers with web browsers) by users 933a.
27 Network interfaces may employ connection protocols such as, but not limited to: direct
28 connect, Ethernet (thick, thin, twisted pair 10/100/1000/10000 Base T, and/or the like),
29 Token Ring, wireless connection such as IEEE 802.11a-x, and/or the like. Should processing

1 requirements dictate a greater amount speed and/or capacity, distributed network controllers
2 (e.g., see Distributed SBWC below), architectures may similarly be employed to pool, load
3 balance, and/or otherwise decrease/increase the communicative bandwidth required by the
4 SBWC controller. A communications network may be any one and/or the combination of the
5 following: a direct interconnection; the Internet; Interplanetary Internet (e.g., Coherent File
6 Distribution Protocol (CFDP), Space Communications Protocol Specifications (SCPS), etc.); a
7 Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as
8 Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network (WAN);
9 a wireless network (e.g., employing protocols such as, but not limited to a cellular, WiFi,
10 Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A network
11 interface may be regarded as a specialized form of an input output interface. Further, multiple
12 network interfaces 910 may be used to engage with various communications network types
13 913. For example, multiple network interfaces may be employed to allow for the
14 communication over broadcast, multicast, and/or unicast networks.

15 **[0091]** Input Output interfaces (I/O) 908 may accept, communicate, and/or connect to user,
16 peripheral devices 912 (e.g., input devices 911), cryptographic processor devices 928, and/or
17 the like. I/O may employ connection protocols such as, but not limited to: audio: analog,
18 digital, monaural, RCA, stereo, and/or the like; data: Apple Desktop Bus (ADB), IEEE 1394a-
19 b, serial, universal serial bus (USB); infrared; joystick; keyboard; midi; optical; PC AT; PS/2;
20 parallel; radio; touch interfaces: capacitive, optical, resistive, etc. displays; video interface: Apple
21 Desktop Connector (ADC), BNC, coaxial, component, composite, digital, Digital Visual
22 Interface (DVI), (mini) displayport, high-definition multimedia interface (HDMI), RCA, RF
23 antennae, S-Video, VGA, and/or the like; wireless transceivers: 802.11a/ac/b/g/n/x;
24 Bluetooth; cellular (e.g., code division multiple access (CDMA), high speed packet access
25 (HSPA(+)), high-speed downlink packet access (HSDPA), global system for mobile
26 communications (GSM), long term evolution (LTE), WiMax, etc.); and/or the like. One typical
27 output device may include a video display, which typically comprises a Cathode Ray Tube
28 (CRT) or Liquid Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry
29 and cable) that accepts signals from a video interface, may be used. The video interface
30 composites information generated by a computer systemization and generates video signals

1 based on the composited information in a video memory frame. Another output device is a
2 television set, which accepts signals from a video interface. Typically, the video interface
3 provides the composited video information through a video connection interface that accepts a
4 video display interface (e.g., an RCA composite video connector accepting an RCA composite
5 video cable; a DVI connector accepting a DVI display cable, etc.).

6 **[0092]** Peripheral devices 912 may be connected and/or communicate to I/O and/or other
7 facilities of the like such as network interfaces, storage interfaces, directly to the interface bus,
8 system bus, the CPU, and/or the like. Peripheral devices may be external, internal and/or part
9 of the SBWC controller. Peripheral devices may include: antenna, audio devices (e.g., line-in,
10 line-out, microphone input, speakers, etc.), cameras (e.g., gesture (e.g., Microsoft Kinect)
11 detection, motion detection, still, video, webcam, etc.), dongles (e.g., for copy protection,
12 ensuring secure transactions with a digital signature, and/or the like), external processors (for
13 added capabilities; e.g., crypto devices 528), force-feedback devices (e.g., vibrating motors),
14 infrared (IR) transceiver, network interfaces, printers, scanners, sensors/sensor arrays and
15 peripheral extensions (e.g., ambient light, GPS, gyroscopes, proximity, temperature, etc.),
16 storage devices, transceivers (e.g., cellular, GPS, etc.), video devices (e.g., goggles, monitors,
17 etc.), video sources, visors, and/or the like. Peripheral devices often include types of input
18 devices (e.g., cameras).

19 **[0093]** User input devices 911 often are a type of peripheral device 512 (see above) and may
20 include: card readers, dongles, finger print readers, gloves, graphics tablets, joysticks,
21 keyboards, microphones, mouse (mice), remote controls, security/biometric devices (e.g.,
22 fingerprint reader, iris reader, retina reader, etc.), touch screens (e.g., capacitive, resistive, etc.),
23 trackballs, trackpads, styluses, and/or the like.

24 **[0094]** It should be noted that although user input devices and peripheral devices may be
25 employed, the SBWC controller may be embodied as an embedded, dedicated, and/or
26 monitor-less (i.e., headless) device, wherein access would be provided over a network interface
27 connection.

28 **[0095]** Cryptographic units such as, but not limited to, microcontrollers, processors 926,
29 interfaces 927, and/or devices 928 may be attached, and/or communicate with the SBWC

1 controller. A MC68HC16 microcontroller, manufactured by Motorola Inc., may be used for
2 and/or within cryptographic units. The MC68HC16 microcontroller utilizes a 16-bit multiply-
3 and-accumulate instruction in the 16 MHz configuration and requires less than one second to
4 perform a 512-bit RSA private key operation. Cryptographic units support the authentication
5 of communications from interacting agents, as well as allowing for anonymous transactions.
6 Cryptographic units may also be configured as part of the CPU. Equivalent microcontrollers
7 and/or processors may also be used. Other commercially available specialized cryptographic
8 processors include: Broadcom's CryptoNetX and other Security Processors; nCipher's nShield;
9 SafeNet's Luna PCI (e.g., 7100) series; Semaphore Communications' 40 MHz Roadrunner 184;
10 Sun's Cryptographic Accelerators (e.g., Accelerator 6000 PCIe Board, Accelerator 500
11 Daughtercard); Via Nano Processor (e.g., L2100, L2200, U2400) line, which is capable of
12 performing 500+ MB/s of cryptographic instructions; VLSI Technology's 33 MHz 6868;
13 and/or the like.

Memory

14
15 ~~[0096]~~ Generally, any mechanization and/or embodiment allowing a processor to affect the
16 storage and/or retrieval of information is regarded as memory 929. However, memory is a
17 fungible technology and resource, thus, any number of memory embodiments may be
18 employed in lieu of or in concert with one another. It is to be understood that the SBWC
19 controller and/or a computer systemization may employ various forms of memory 929. For
20 example, a computer systemization may be configured wherein the operation of on-chip CPU
21 memory (e.g., registers), RAM, ROM, and any other storage devices are provided by a paper
22 punch tape or paper punch card mechanism; however, such an embodiment would result in an
23 extremely slow rate of operation. In a typical configuration, memory 929 will include ROM
24 906, RAM 905, and a storage device 914. A storage device 914 may be any conventional
25 computer system storage. Storage devices may include: an array of devices (e.g., Redundant
26 Array of Independent Disks (RAID)); a drum; a (fixed and/or removable) magnetic disk drive;
27 a magneto-optical drive; an optical drive (i.e., Blu-ray, CD ROM/RAM/Recordable
28 (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); RAM drives; solid state memory
29 devices (USB memory, solid state drives (SSD), etc.); other processor-readable storage

1 mediums; and/or other devices of the like. Thus, a computer systemization generally requires
2 and makes use of memory.

3 **Component Collection**

4 **[0097]** The memory 929 may contain a collection of program and/or database components
5 and/or data such as, but not limited to: operating system component(s) 915 (operating system);
6 information server component(s) 916 (information server); user interface component(s) 917
7 (user interface); Web browser component(s) 918 (Web browser); database(s) 919; mail server
8 component(s) 921; mail client component(s) 922; cryptographic server component(s) 920
9 (cryptographic server); the SBWC component(s) 935; and/or the like (i.e., collectively a
10 component collection). These components may be stored and accessed from the storage
11 devices and/or from storage devices accessible through an interface bus. Although non-
12 conventional program components such as those in the component collection, typically, are
13 stored in a local storage device 914, they may also be loaded and/or stored in memory such as:
14 peripheral devices, RAM, remote storage facilities through a communications network, ROM,
15 various forms of memory, and/or the like.

16 **Operating System**

17 **[0098]** The operating system component 915 is an executable program component facilitating
18 the operation of the SBWC controller. Typically, the operating system facilitates access of I/O,
19 network interfaces, peripheral devices, storage devices, and/or the like. The operating system
20 may be a highly fault tolerant, scalable, and secure system such as: Apple's Macintosh OS X
21 (Server); AT&T Plan 9; Be OS; Blackberry's QNX; Google's Chrome; Microsoft's Windows
22 7/8; Unix and Unix-like system distributions (such as AT&T's UNIX; Berkley Software
23 Distribution (BSD) variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux
24 distributions such as Red Hat, Ubuntu, and/or the like); and/or the like operating systems.
25 However, more limited and/or less secure operating systems also may be employed such as
26 Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows
27 2000/2003/3.1/95/98/CE/Millennium/Mobile/NT/Vista/XP (Server), Palm OS, and/or the
28 like. Additionally, for robust mobile deployment applications, mobile operating systems may be
29 used, such as: Apple's iOS; China Operating System COS; Google's Android; Microsoft

1 Windows RT/Phone; Palm's WebOS; Samsung/Intel's Tizen; and/or the like. An operating
2 system may communicate to and/or with other components in a component collection,
3 including itself, and/or the like. Most frequently, the operating system communicates with
4 other program components, user interfaces, and/or the like. For example, the operating system
5 may contain, communicate, generate, obtain, and/or provide program component, system,
6 user, and/or data communications, requests, and/or responses. The operating system, once
7 executed by the CPU, may enable the interaction with communications networks, data, I/O,
8 peripheral devices, program components, memory, user input devices, and/or the like. The
9 operating system may provide communications protocols that allow the SBWC controller to
10 communicate with other entities through a communications network 913. Various
11 communication protocols may be used by the SBWC controller as a subcarrier transport
12 mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast,
13 and/or the like.

14 Information Server

15 ~~[0099]~~ An information server component 916 is a stored program component that is executed
16 by a CPU. The information server may be a conventional Internet information server such as,
17 but not limited to Apache Software Foundation's Apache, Microsoft's Internet Information
18 Server, and/or the like. The information server may allow for the execution of program
19 components through facilities such as Active Server Page (ASP), ActiveX, (ANSI) (Objective-)
20 C (++), C# and/or .NET, Common Gateway Interface (CGI) scripts, dynamic (D) hypertext
21 markup language (HTML), FLASH, Java, JavaScript, Practical Extraction Report Language
22 (PERL), Hypertext Pre-Processor (PHP), pipes, Python, wireless application protocol (WAP),
23 WebObjects, and/or the like. The information server may support secure communications
24 protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer
25 Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL),
26 messaging protocols (e.g., America Online (AOL) Instant Messenger (AIM), Application
27 Exchange (APEX), ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger
28 Service, Presence and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's
29 (IETF's) Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging
30 Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol

1 (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and Presence
2 Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The information server
3 provides results in the form of Web pages to Web browsers, and allows for the manipulated
4 generation of the Web pages through interaction with other program components. After a
5 Domain Name System (DNS) resolution portion of an HTTP request is resolved to a
6 particular information server, the information server resolves requests for information at
7 specified locations on the SBWC controller based on the remainder of the HTTP request. For
8 example, a request such as `http://123.124.125.126/myInformation.html` might have the IP
9 portion of the request "123.124.125.126" resolved by a DNS server to an information server at
10 that IP address; that information server might in turn further parse the http request for the
11 `"/myInformation.html"` portion of the request and resolve it to a location in memory
12 containing the information "myInformation.html." Additionally, other information serving
13 protocols may be employed across various ports, e.g., FTP communications across port 21,
14 and/or the like. An information server may communicate to and/or with other components in
15 a component collection, including itself, and/or facilities of the like. Most frequently, the
16 information server communicates with the SBWC database 919, operating systems, other
17 program components, user interfaces, Web browsers, and/or the like.

18 **[00100]** Access to the SBWC database may be achieved through a number of database bridge
19 mechanisms such as through scripting languages as enumerated below (e.g., CGI) and through
20 inter-application communication channels as enumerated below (e.g., CORBA, WebObjects,
21 etc.). Any data requests through a Web browser are parsed through the bridge mechanism into
22 appropriate grammars as required by the SBWC. In one embodiment, the information server
23 would provide a Web form accessible by a Web browser. Entries made into supplied fields in
24 the Web form are tagged as having been entered into the particular fields, and parsed as such.
25 The entered terms are then passed along with the field tags, which act to instruct the parser to
26 generate queries directed to appropriate tables and/or fields. In one embodiment, the parser
27 may generate queries in standard SQL by instantiating a search string with the proper
28 join/select commands based on the tagged text entries, wherein the resulting command is
29 provided over the bridge mechanism to the SBWC as a query. Upon generating query results
30 from the query, the results are passed over the bridge mechanism, and may be parsed for

1 formatting and generation of a new results Web page by the bridge mechanism. Such a new
2 results Web page is then provided to the information server, which may supply it to the
3 requesting Web browser.

4 [00101] Also, an information server may contain, communicate, generate, obtain, and/or
5 provide program component, system, user, and/or data communications, requests, and/or
6 responses.

7 User Interface

8 [00102] Computer interfaces in some respects are similar to automobile operation interfaces.
9 Automobile operation interface elements such as steering wheels, gearshifts, and speedometers
10 facilitate the access, operation, and display of automobile resources, and status. Computer
11 interaction interface elements such as buttons, check boxes, cursors, menus, scrollers, and
12 windows (collectively and commonly referred to as widgets) similarly facilitate the access,
13 capabilities, operation, and display of data and computer hardware and operating system
14 resources, and status. Operation interfaces are commonly called user interfaces. Graphical user
15 interfaces (GUIs) such as the Apple's iOS, Macintosh Operating System's Aqua; IBM's OS/2;
16 Google's Chrome (e.g., and other webbrowser/cloud based client OSs); Microsoft's Windows
17 varied UIs 2000/2003/3.1/95/98/CE/Millennium/Mobile/NT/Vista/XP (Server) (i.e., Aero,
18 Surface, etc.); Unix's X-Windows (e.g., which may include additional Unix graphic interface
19 libraries and layers such as K Desktop Environment (KDE), mythTV and GNU Network
20 Object Model Environment (GNOME)), web interface libraries (e.g., ActiveX, AJAX,
21 (D)HTML, FLASH, Java, JavaScript, etc. interface libraries such as, but not limited to, Dojo,
22 jQuery(UI), MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of
23 which may be used and) provide a baseline and means of accessing and displaying information
24 graphically to users.

25 [00103] A user interface component 917 is a stored program component that is executed by a
26 CPU. The user interface may be a conventional graphic user interface as provided by, with,
27 and/or atop operating systems and/or operating environments such as already discussed. The
28 user interface may allow for the display, execution, interaction, manipulation, and/or operation
29 of program components and/or system facilities through textual and/or graphical facilities.

1 The user interface provides a facility through which users may affect, interact, and/or operate a
2 computer system. A user interface may communicate to and/or with other components in a
3 component collection, including itself, and/or facilities of the like. Most frequently, the user
4 interface communicates with operating systems, other program components, and/or the like.
5 The user interface may contain, communicate, generate, obtain, and/or provide program
6 component, system, user, and/or data communications, requests, and/or responses.

7 Web Browser

8 [00104] A Web browser component 918 is a stored program component that is executed by a
9 CPU. The Web browser may be a conventional hypertext viewing application such as Apple's
10 (mobile) Safari, Google's Chrome, Microsoft Internet Explorer, Mozilla's Firefox, Netscape
11 Navigator, and/or the like. Secure Web browsing may be supplied with 128bit (or greater)
12 encryption by way of HTTPS, SSL, and/or the like. Web browsers allowing for the execution
13 of program components through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java,
14 JavaScript, web browser plug-in APIs (e.g., FireFox, Safari Plug-in, and/or the like APIs),
15 and/or the like. Web browsers and like information access tools may be integrated into PDAs,
16 cellular telephones, and/or other mobile devices. A Web browser may communicate to and/or
17 with other components in a component collection, including itself, and/or facilities of the like.
18 Most frequently, the Web browser communicates with information servers, operating systems,
19 integrated program components (e.g., plug-ins), and/or the like; e.g., it may contain,
20 communicate, generate, obtain, and/or provide program component, system, user, and/or data
21 communications, requests, and/or responses. Also, in place of a Web browser and information
22 server, a combined application may be developed to perform similar operations of both. The
23 combined application would similarly affect the obtaining and the provision of information to
24 users, user agents, and/or the like from the SBWC enabled nodes. The combined application
25 may be nugatory on systems employing standard Web browsers.

26 Mail Server

27 [00105] A mail server component 921 is a stored program component that is executed by a
28 CPU 903. The mail server may be a conventional Internet mail server such as, but not limited
29 to: dovecot, Courier IMAP, Cyrus IMAP, Maildir, Microsoft Exchange, sendmail, and/or the

1 like. The mail server may allow for the execution of program components through facilities
2 such as ASP, ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, CGI scripts, Java,
3 JavaScript, PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may
4 support communications protocols such as, but not limited to: Internet message access
5 protocol (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft
6 Exchange, post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like.
7 The mail server can route, forward, and process incoming and outgoing mail messages that
8 have been sent, relayed and/or otherwise traversing through and/or to the SBWC.
9 Alternatively, the mail server component may be distributed out to mail service providing
10 entities such as Google's cloud services (e.g., Gmail and notifications may alternatively be
11 provided via messenger services such as AOL's Instant Messenger, Apple's iMessage, Google
12 Messenger, SnapChat, etc.).

13 [00106] Access to the SBWC mail may be achieved through a number of APIs offered by the
14 individual Web server components and/or the operating system.

15 [00107] Also, a mail server may contain, communicate, generate, obtain, and/or provide
16 program component, system, user, and/or data communications, requests, information,
17 and/or responses.

18 Mail Client

19 [00108] A mail client component 922 is a stored program component that is executed by a CPU
20 903. The mail client may be a conventional mail viewing application such as Apple Mail,
21 Microsoft Entourage, Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird,
22 and/or the like. Mail clients may support a number of transfer protocols, such as: IMAP,
23 Microsoft Exchange, POP3, SMTP, and/or the like. A mail client may communicate to and/or
24 with other components in a component collection, including itself, and/or facilities of the like.
25 Most frequently, the mail client communicates with mail servers, operating systems, other mail
26 clients, and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide
27 program component, system, user, and/or data communications, requests, information,
28 and/or responses. Generally, the mail client provides a facility to compose and transmit
29 electronic mail messages.

Cryptographic Server

1
2 [00109] A cryptographic server component 920 is a stored program component that is executed
3 by a CPU 903, cryptographic processor 926, cryptographic processor interface 927,
4 cryptographic processor device 928, and/or the like. Cryptographic processor interfaces will
5 allow for expedition of encryption and/or decryption requests by the cryptographic
6 component; however, the cryptographic component, alternatively, may run on a conventional
7 CPU. The cryptographic component allows for the encryption and/or decryption of provided
8 data. The cryptographic component allows for both symmetric and asymmetric (e.g., Pretty
9 Good Protection (PGP)) encryption and/or decryption. The cryptographic component may
10 employ cryptographic techniques such as, but not limited to: digital certificates (e.g., X.509
11 authentication framework), digital signatures, dual signatures, enveloping, password access
12 protection, public key management, and/or the like. The cryptographic component will
13 facilitate numerous (encryption and/or decryption) security protocols such as, but not limited
14 to: checksum, Data Encryption Standard (DES), Elliptical Curve Encryption (ECC),
15 International Data Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way
16 hash operation), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet
17 encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest,
18 Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure Socket Layer
19 (SSL), Secure Hypertext Transfer Protocol (HTTPS), Transport Layer Security (TLS), and/or
20 the like. Employing such encryption security protocols, the SBWC may encrypt all incoming
21 and/or outgoing communications and may serve as node within a virtual private network
22 (VPN) with a wider communications network. The cryptographic component facilitates the
23 process of “security authorization” whereby access to a resource is inhibited by a security
24 protocol wherein the cryptographic component effects authorized access to the secured
25 resource. In addition, the cryptographic component may provide unique identifiers of content,
26 e.g., employing and MD5 hash to obtain a unique signature for an digital audio file. A
27 cryptographic component may communicate to and/or with other components in a
28 component collection, including itself, and/or facilities of the like. The cryptographic
29 component supports encryption schemes allowing for the secure transmission of information
30 across a communications network to enable the SBWC component to engage in secure

1 transactions if so desired. The cryptographic component facilitates the secure accessing of
2 resources on the SBWC and facilitates the access of secured resources on remote systems; i.e.,
3 it may act as a client and/or server of secured resources. Most frequently, the cryptographic
4 component communicates with information servers, operating systems, other program
5 components, and/or the like. The cryptographic component may contain, communicate,
6 generate, obtain, and/or provide program component, system, user, and/or data
7 communications, requests, and/or responses.

8

The SBWC Database

9 [00110] The SBWC database component 919 may be embodied in a database and its stored
10 data. The database is a stored program component, which is executed by the CPU; the stored
11 program component portion configuring the CPU to process the stored data. The database
12 may be a conventional, fault tolerant, relational, scalable, secure database such as MySQL,
13 Oracle, Sybase, etc. may be used. Additionally, optimized fast memory and distributed
14 databases such as IBM's Netezza, MongoDB's MongoDB, opensource Hadoop, opensource
15 VoltDB, SAP's Hana, etc. Relational databases are an extension of a flat file. Relational
16 databases consist of a series of related tables. The tables are interconnected via a key field. Use
17 of the key field allows the combination of the tables by indexing against the key field; i.e., the
18 key fields act as dimensional pivot points for combining information from various tables.
19 Relationships generally identify links maintained between tables by matching primary keys.
20 Primary keys represent fields that uniquely identify the rows of a table in a relational database.
21 Alternative key fields may be used from any of the fields having unique value sets, and in some
22 alternatives, even non-unique values in combinations with other fields. More precisely, they
23 uniquely identify rows of a table on the "one" side of a one-to-many relationship.

24 [00111] Alternatively, the SBWC database may be implemented using various standard data-
25 structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table,
26 and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In
27 another alternative, an object-oriented database may be used, such as Frontier, ObjectStore,
28 Poet, Zope, and/or the like. Object databases can include a number of object collections that
29 are grouped and/or linked together by common attributes; they may be related to other object

1 collections by some common attributes. Object-oriented databases perform similarly to
2 relational databases with the exception that objects are not just pieces of data but may have
3 other types of capabilities encapsulated within a given object. If the SBWC database is
4 implemented as a data-structure, the use of the SBWC database 919 may be integrated into
5 another component such as the SBWC component 935. Also, the database may be
6 implemented as a mix of data structures, objects, and relational structures. Databases may be
7 consolidated and/or distributed in countless variations (e.g., see Distributed SBWC below).
8 Portions of databases, e.g., tables, may be exported and/or imported and thus decentralized
9 and/or integrated.

10 **[00112]** In one embodiment, the database component 919 includes several tables 919a-i:

11 **[00113]** An accounts table 919a includes fields such as, but not limited to: an accountID,
12 accountOwnerID, accountContactID, assetIDs, deviceIDs, paymentIDs, transactionIDs,
13 userIDs, accountType (e.g., agent, entity (e.g., corporate, non-profit, partnership, etc.),
14 individual, etc.), accountCreationDate, accountUpdateDate, accountName, accountNumber,
15 routingNumber, linkWalletsID, accountPrioritAccountRatio, accountAddress, accountState,
16 accountZIPcode, accountCountry, accountEmail, accountPhone, accountAuthKey,
17 accountIPAddress, accountURLAccessCode, accountPortNo, accountAuthorizationCode,
18 accountAccessPrivileges, accountPreferences, accountRestrictions, and/or the like;

19 **[00114]** A users table 919b includes fields such as, but not limited to: a userID, userSSN, taxID,
20 userContactID, accountID, assetIDs, deviceIDs, paymentIDs, transactionIDs, userType (e.g.,
21 agent, entity (e.g., corporate, non-profit, partnership, etc.), individual, etc.), namePrefix,
22 firstName, middleName, lastName, nameSuffix, DateOfBirth, userAge, userName, userEmail,
23 userSocialAccountID, contactType, contactRelationship, userPhone, userAddress, userCity,
24 userState, userZIPCode, userCountry, userAuthorizationCode, userAccessPrivileges,
25 userPreferences, userRestrictions, and/or the like (the user table may support and/or track
26 multiple entity accounts on a SBWC);

27 **[00115]** An devices table 919c includes fields such as, but not limited to: deviceID, sensorIDs,
28 accountID, assetIDs, paymentIDs, deviceType, deviceName, deviceManufacturer,
29 deviceModel, deviceVersion, deviceSerialNo, deviceIPAddress, deviceMACaddress,

1 device_ECID, deviceUUID, deviceLocation, deviceCertificate, deviceOS, appIDs,
 2 deviceResources, deviceSession, authKey, deviceSecureKey, walletAppInstalledFlag,
 3 deviceAccessPrivileges, devicePreferences, deviceRestrictions, hardware_config,
 4 software_config, storage_location, sensor_value, pin_reading, data_length,
 5 channel_requirement, sensor_name, sensor_model_no, sensor_manufacturer, sensor_type,
 6 sensor_serial_number, sensor_power_requirement, device_power_requirement, location,
 7 sensor_associated_tool, sensor_dimensions, device_dimensions,
 8 sensor_communications_type, device_communications_type, power_percentage,
 9 power_condition, temperature_setting, speed_adjust, hold_duration, part_actuation, and/or
 10 the like. Device table may, in some embodiments, include fields corresponding to one or more
 11 Bluetooth profiles, such as those published at [https://www.bluetooth.org/en-](https://www.bluetooth.org/en-us/specification/adopted-specifications)
 12 [us/specification/adopted-specifications](https://www.bluetooth.org/en-us/specification/adopted-specifications), and/or other device specifications, and/or the like;

13 **[00116]** An apps table 919d includes fields such as, but not limited to: appID, appName,
 14 appType, appDependencies, accountID, deviceIDs, transactionID, userID, appStoreAuthKey,
 15 appStoreAccountID, appStoreIPAddress, appStoreURLaccessCode, appStorePortNo,
 16 appAccessPrivileges, appPreferences, appRestrictions, portNum, access_API_call,
 17 linked_wallets_list, and/or the like;

18 **[00117]** An assets table 919e includes fields such as, but not limited to: assetID, accountID,
 19 userID, distributorAccountID, distributorPaymentID, distributorOwnerID, assetOwnerID,
 20 assetType, assetSourceDeviceID, assetSourceDeviceType, assetSourceDeviceName,
 21 assetSourceDistributionChannelID, assetSourceDistributionChannelType,
 22 assetSourceDistributionChannelName, assetTargetChannelID, assetTargetChannelType,
 23 assetTargetChannelName, assetName, assetSeriesName, assetSeriesSeason, assetSeriesEpisode,
 24 assetCode, assetQuantity, assetCost, assetPrice, assetValue, assetManufacturer, assetModelNo,
 25 assetSerialNo, assetLocation, assetAddress, assetState, assetZIPcode, assetState, assetCountry,
 26 assetEmail, assetIPAddress, assetURLaccessCode, assetOwnerAccountID, subscriptionIDs,
 27 assetAuthorizationCode, assetAccessPrivileges, assetPreferences, assetRestrictions, assetAPI,
 28 assetAPIconnectionAddress, and/or the like;

29 **[00118]** A payments table 919f includes fields such as, but not limited to: paymentID,
 30 accountID, userID, couponID, couponValue, couponConditions, couponExpiration,

1 paymentType, paymentAccountNo, paymentAccountName,
 2 paymentAccountAuthorizationCodes, paymentExpirationDate, paymentCCV,
 3 paymentRoutingNo, paymentRoutingType, paymentAddress, paymentState, paymentZIPcode,
 4 paymentCountry, paymentEmail, paymentAuthKey, paymentIPAddress,
 5 paymentURLaccessCode, paymentPortNo, paymentAccessPrivileges, paymentPreferences,
 6 paymentRestrictions, and/or the like;

7 **[00119]** An transactions table 919g includes fields such as, but not limited to: transactionID,
 8 accountID, assetIDs, deviceIDs, paymentIDs, transactionIDs, userID, merchantID,
 9 transactionType, transactionDate, transactionTime, transactionAmount, transactionQuantity,
 10 transactionDetails, productsList, productType, productTitle, productsSummary,
 11 productParamsList, transactionNo, transactionAccessPrivileges, transactionPreferences,
 12 transactionRestrictions, merchantAuthKey, merchantAuthCode, and/or the like;

13 **[00120]** An merchants table 919h includes fields such as, but not limited to: merchantID,
 14 merchantTaxID, merchantName, merchantContactUserID, accountID, issuerID, acquirerID,
 15 merchantEmail, merchantAddress, merchantState, merchantZIPcode, merchantCountry,
 16 merchantAuthKey, merchantIPAddress, portNum, merchantURLaccessCode,
 17 merchantPortNo, merchantAccessPrivileges, merchantPreferences, merchantRestrictions,
 18 and/or the like;

19 **[00121]** An ads table 919i includes fields such as, but not limited to: adID, advertiserID,
 20 adMerchantID, adNetworkID, adName, adTags, advertiserName, adSponsor, adTime, adGeo,
 21 adAttributes, adFormat, adProduct, adText, adMedia, adMediaID, adChannelID, adTagTime,
 22 adAudioSignature, adHash, adTemplateID, adTemplateData, adSourceID, adSourceName,
 23 adSourceServerIP, adSourceURL, adSourceSecurityProtocol, adSourceFTP, adAuthKey,
 24 adAccessPrivileges, adPreferences, adRestrictions, adNetworkXchangeID,
 25 adNetworkXchangeName, adNetworkXchangeCost, adNetworkXchangeMetricType (e.g.,
 26 CPA, CPC, CPM, CTR, etc.), adNetworkXchangeMetricValue, adNetworkXchangeServer,
 27 adNetworkXchangePortNumber, publisherID, publisherAddress, publisherURL, publisherTag,
 28 publisherIndustry, publisherName, publisherDescription, siteDomain, siteURL, siteContent,
 29 siteTag, siteContext, siteImpression, siteVisits, siteHeadline, sitePage, siteAdPrice,
 30 sitePlacement, sitePosition, bidID, bidExchange, bidOS, bidTarget, bidTimestamp, bidPrice,

1 bidImpressionID, bidType, bidScore, adType (e.g., mobile, desktop, wearable, largescreen,
2 interstitial, etc.), assetID, merchantID, deviceID, userID, accountID, impressionID,
3 impressionOS, impressionTimeStamp, impressionGeo, impressionAction, impressionType,
4 impressionPublisherID, impressionPublisherURL, and/or the like.

5 **[00122]** In one embodiment, the SBWC database may interact with other database systems. For
6 example, employing a distributed database system, queries and data access by search SBWC
7 component may treat the combination of the SBWC database, an integrated data security layer
8 database as a single database entity (e.g., see Distributed SBWC below).

9 **[00123]** In one embodiment, user programs may contain various user interface primitives, which
10 may serve to update the SBWC. Also, various accounts may require custom database tables
11 depending upon the environments and the types of clients the SBWC may need to serve. It
12 should be noted that any unique fields may be designated as a key field throughout. In an
13 alternative embodiment, these tables have been decentralized into their own databases and
14 their respective database controllers (i.e., individual database controllers for each of the above
15 tables). Employing standard data processing techniques, one may further distribute the
16 databases over several computer systemizations and/or storage devices. Similarly,
17 configurations of the decentralized database controllers may be varied by consolidating and/or
18 distributing the various database components 919a-i. The SBWC may be configured to keep
19 track of various settings, inputs, and parameters via database controllers.

20 **[00124]** The SBWC database may communicate to and/or with other components in a
21 component collection, including itself, and/or facilities of the like. Most frequently, the SBWC
22 database communicates with the SBWC component, other program components, and/or the
23 like. The database may contain, retain, and provide information regarding other nodes and
24 data.

25 The SBWCs

26 **[00125]** The SBWC component 935 is a stored program component that is executed by a CPU.
27 In one embodiment, the SBWC component incorporates any and/or all combinations of the
28 aspects of the SBWC that was discussed in the previous figures. As such, the SBWC affects
29 accessing, obtaining and the provision of information, services, transactions, and/or the like

1 across various communications networks. The features and embodiments of the SBWC
2 discussed herein increase network efficiency by reducing data transfer requirements the use of
3 more efficient data structures and mechanisms for their transfer and storage. As a
4 consequence, more data may be transferred in less time, and latencies with regard to
5 transactions, are also reduced. In many cases, such reduction in storage, transfer time,
6 bandwidth requirements, latencies, etc., will reduce the capacity and structural infrastructure
7 requirements to support the SBWC's features and facilities, and in many cases reduce the costs,
8 energy consumption/requirements, and extend the life of SBWC's underlying infrastructure;
9 this has the added benefit of making the SBWC more reliable. Similarly, many of the features
10 and mechanisms are designed to be easier for users to use and access, thereby broadening the
11 audience that may enjoy/employ and exploit the feature sets of the SBWC; such ease of use
12 also helps to increase the reliability of the SBWC. In addition, the feature sets include
13 heightened security as noted via the Cryptographic components 920, 926, 928 and throughout,
14 making access to the features and data more reliable and secure

15 **[00126]** The SBWC transforms webpage request inputs, via SBWC components (e.g., FE, FB,
16 TLL, FQ, EPR), into composited expandable element outputs.

17 **[00127]** The SBWC component enabling access of information between nodes may be
18 developed by employing standard development tools and languages such as, but not limited to:
19 Apache components, Assembly, ActiveX, binary executables, (ANSI) (Objective-) C (++), C#
20 and/or .NET, database adapters, CGI scripts, Java, JavaScript, mapping tools, procedural and
21 object oriented development tools, PERL, PHP, Python, shell scripts, SQL commands, web
22 application server extensions, web development environments and libraries (e.g., Microsoft's
23 ActiveX; Adobe AIR, FLEX & FLASH; AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI);
24 MooTools; Prototype; script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject;
25 Yahoo! User Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the
26 SBWC server employs a cryptographic server to encrypt and decrypt communications. The
27 SBWC component may communicate to and/or with other components in a component
28 collection, including itself, and/or facilities of the like. Most frequently, the SBWC component
29 communicates with the SBWC database, operating systems, other program components,

1 and/or the like. The SBWC may contain, communicate, generate, obtain, and/or provide
2 program component, system, user, and/or data communications, requests, and/or responses.

3 Distributed SBWCs

4 **[00128]** The structure and/or operation of any of the SBWC node controller components may
5 be combined, consolidated, and/or distributed in any number of ways to facilitate
6 development and/or deployment. Similarly, the component collection may be combined in any
7 number of ways to facilitate deployment and/or development. To accomplish this, one may
8 integrate the components into a common code base or in a facility that can dynamically load
9 the components on demand in an integrated fashion. As such a combination of hardware may
10 be distributed within a location, within a region and/or globally where logical access to a
11 controller may be abstracted as a singular node, yet where a multitude of private, semiprivate
12 and publically accessible node controllers (e.g., via dispersed data centers) are coordinated to
13 serve requests (e.g., providing private cloud, semi-private cloud, and public cloud computing
14 resources) and allowing for the serving of such requests in discrete regions (e.g., isolated, local,
15 regional, national, global cloud access).

16 **[00129]** The component collection may be consolidated and/or distributed in countless
17 variations through standard data processing and/or development techniques. Multiple
18 instances of any one of the program components in the program component collection may be
19 instantiated on a single node, and/or across numerous nodes to improve performance through
20 load-balancing and/or data-processing techniques. Furthermore, single instances may also be
21 distributed across multiple controllers and/or storage devices; e.g., databases. All program
22 component instances and controllers working in concert may do so through standard data
23 processing communication techniques.

24 **[00130]** The configuration of the SBWC controller will depend on the context of system
25 deployment. Factors such as, but not limited to, the budget, capacity, location, and/or use of
26 the underlying hardware resources may affect deployment requirements and configuration.
27 Regardless of if the configuration results in more consolidated and/or integrated program
28 components, results in a more distributed series of program components, and/or results in
29 some combination between a consolidated and distributed configuration, data may be

1 communicated, obtained, and/or provided. Instances of components consolidated into a
2 common code base from the program component collection may communicate, obtain,
3 and/or provide data. This may be accomplished through intra-application data processing
4 communication techniques such as, but not limited to: data referencing (e.g., pointers), internal
5 messaging, object instance variable communication, shared memory space, variable passing,
6 and/or the like. For example, cloud services such as Amazon Data Services, Microsoft Azure,
7 Hewlett Packard Helion, IBM Cloud services allow for SBWC controller and/or SBWC
8 component collections to be hosted in full or partially for varying degrees of scale.

9 **[00131]** If component collection components are discrete, separate, and/or external to one
10 another, then communicating, obtaining, and/or providing data with and/or to other
11 component components may be accomplished through inter-application data processing
12 communication techniques such as, but not limited to: Application Program Interfaces (API)
13 information passage; (distributed) Component Object Model ((D)COM), (Distributed) Object
14 Linking and Embedding ((D)OLE), and/or the like), Common Object Request Broker
15 Architecture (CORBA), Jini local and remote application program interfaces, JavaScript Object
16 Notation (JSON), Remote Method Invocation (RMI), SOAP, process pipes, shared files,
17 and/or the like. Messages sent between discrete component components for inter-application
18 communication or within memory spaces of a singular component for intra-application
19 communication may be facilitated through the creation and parsing of a grammar. A grammar
20 may be developed by using development tools such as lex, yacc, XML, and/or the like, which
21 allow for grammar generation and parsing capabilities, which in turn may form the basis of
22 communication messages within and between components.

23 **[00132]** For example, a grammar may be arranged to recognize the tokens of an HTTP post
24 command, e.g.:

25 w3c -post http://... Value1
26

27 **[00133]** where Value1 is discerned as being a parameter because “http://” is part of the
28 grammar syntax, and what follows is considered part of the post value. Similarly, with such a
29 grammar, a variable “Value1” may be inserted into an “http://” post command and then sent.
30 The grammar syntax itself may be presented as structured data that is interpreted and/or

1 otherwise used to generate the parsing mechanism (e.g., a syntax description text file as
2 processed by lex, yacc, etc.). Also, once the parsing mechanism is generated and/or
3 instantiated, it itself may process and/or parse structured data such as, but not limited to:
4 character (e.g., tab) delineated text, HTML, structured text streams, XML, and/or the like
5 structured data. In another embodiment, inter-application data processing protocols
6 themselves may have integrated and/or readily available parsers (e.g., JSON, SOAP, and/or
7 like parsers) that may be employed to parse (e.g., communications) data. Further, the parsing
8 grammar may be used beyond message parsing, but may also be used to parse: databases, data
9 collections, data stores, structured data, and/or the like. Again, the desired configuration will
10 depend upon the context, environment, and requirements of system deployment.

11 **[00134]** For example, in some implementations, the SBWC controller may be executing a PHP
12 script implementing a Secure Sockets Layer (“SSL”) socket server via the information server,
13 which listens to incoming communications on a server port to which a client may send data,
14 e.g., data encoded in JSON format. Upon identifying an incoming communication, the PHP
15 script may read the incoming message from the client device, parse the received JSON-
16 encoded text data to extract information from the JSON-encoded text data into PHP script
17 variables, and store the data (e.g., client identifying information, etc.) and/or extracted
18 information in a relational database accessible using the Structured Query Language (“SQL”).
19 An exemplary listing, written substantially in the form of PHP/SQL commands, to accept
20 JSON-encoded input data from a client device via a SSL connection, parse the data to extract
21 variables, and store the data to a database, is provided below:

```
22 <?PHP
23 header('Content-Type: text/plain');
24
25 // set ip address and port to listen to for incoming data
26 $address = '192.168.0.100';
27 $port = 255;
28
29 // create a server-side SSL socket, listen for/accept incoming communication
30 $sock = socket_create(AF_INET, SOCK_STREAM, 0);
31 socket_bind($sock, $address, $port) or die('Could not bind to address');
32 socket_listen($sock);
33 $client = socket_accept($sock);
34
```

```
1 // read input data from client device in 1024 byte blocks until end of message
2 do {
3     $input = "";
4     $input = socket_read($client, 1024);
5     $data .= $input;
6 } while($input != "");
7
8 // parse data to extract variables
9 $obj = json_decode($data, true);
10
11 // store input data in a database
12 mysql_connect("201.408.185.132",$DBserver,$password); // access database server
13 mysql_select("CLIENT_DB.SQL"); // select database to append
14 mysql_query("INSERT INTO UserTable (transmission)
15 VALUES ($data)"); // add data to UserTable table in a CLIENT database
16 mysql_close("CLIENT_DB.SQL"); // close connection to database
17 ?>
```

18
19 **[00135]** Also, the following resources may be used to provide example embodiments regarding
20 SOAP parser implementation:

```
21 http://www.xav.com/perl/site/lib/SOAP/Parser.html
22 http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.I
23 BMDI.doc/referenceguide295.htm
```

24
25 and other parser implementations:

```
26 http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.I
27 BMDI.doc/referenceguide259.htm
```

28
29 all of which are hereby expressly incorporated by reference.

30
31 **[00136]** Additional embodiments include:

1 1. A webpage frame expanding display apparatus, comprising:
2 a memory;
3 a component collection in the memory;
4 a processor disposed in communication with the memory, and configured to issue a plurality of
5 processing instructions from the component collection stored in the memory,
6 wherein the processor issues instructions from the component collection, stored in the memory,
7 to:
8 provide, via at least one processor, a top level listening component to a top level of a webpage,
9 wherein upon webpage loading the top level listening component is configured to
10 create a node list of first level frames in the webpage;
11 configure, via at least one processor, a first frame in the webpage to obtain an expandable
12 display payload that includes a frame escaping component, wherein the frame escaping
13 component is configured to:
14 generate, via at least one processor, a unique frame identifier associated with the
15 expandable display payload;
16 create, via at least one processor, a second frame configured to obtain a frame buster
17 payload that includes a frame busting component; and
18 add, via at least one processor, a top level event listener for messages from the top level
19 listening component;
20 wherein the top level event listener is configured, upon receiving a message from the top
21 level listening component, to:
22 determine, via at least one processor, a unique frame identifier specified in the message
23 from the top level listening component;
24 verify, via at least one processor, that the unique frame identifier specified in the
25 message from the top level listening component matches the generated unique frame
26 identifier;
27 determine, via at least one processor, a node list identifier of a first level frame in the
28 webpage specified in the message from the top level listening component; and
29 send, via at least one processor, a message to the frame busting component with the
30 determined node list identifier;
31 wherein the frame busting component is configured to:
32 determine, via at least one processor, the unique frame identifier associated with the
33 expandable display payload;

1 generate, via at least one processor, an executable frame querying component
2 configured to facilitate querying first level frames in the webpage to determine which
3 first level frame contains the expandable display payload using the unique frame
4 identifier associated with the expandable display payload;

5 send, via at least one processor, a message to the top level listening component with
6 the generated executable frame querying component; and

7 add, via at least one processor, a frame event listener for messages from the frame
8 escaping component;

9 wherein the frame event listener is configured, upon receiving a message from the
10 frame escaping component, to:

11 determine, via at least one processor, the node list identifier specified in the
12 message from the frame escaping component;

13 generate, via at least one processor, an executable expandable payload rebuilding
14 component configured to facilitate rebuilding the expandable display payload in the top
15 level of the webpage using the node list identifier specified in the message from the
16 frame escaping component; and

17 send, via at least one processor, a message to the top level listening component
18 with the generated executable expandable payload rebuilding component;

19 wherein the top level listening component is further configured to:

20 add, via at least one processor, a buster event listener for messages from the frame busting
21 component;

22 wherein the buster event listener is configured, upon receiving a message from the frame
23 busting component, to:

24 execute, via at least one processor, an executable component provided in the message
25 from the frame busting component.

26
27 2. The apparatus of embodiment 1, wherein a frame is an iframe.

28 3. The apparatus of embodiment 1, wherein a message is a post message.

29 4. The apparatus of embodiment 1, wherein a node list identifier is an item index in the node list.

30 5. The apparatus of embodiment 1, wherein the expandable display payload is an expandable ad.

31 6. The apparatus of embodiment 1, wherein the frame escaping component is further configured to:

32 determine, via at least one processor, whether escape from the first frame should be attempted.

- 1 7. The apparatus of embodiment 1, wherein the unique frame identifier associated with the expandable
2 display payload is a randomly generated number.
- 3 8. The apparatus of embodiment 1, wherein the frame busting component is further configured to:
4 determine, via at least one processor, the unique frame identifier associated with the expandable
5 display payload by parsing a name attribute.
- 6 9. The apparatus of embodiment 1, wherein the frame busting component is further configured to:
7 determine, via at least one processor, a source associated with a first level frame containing the
8 expandable display payload;
9 wherein the generated executable frame querying component is further configured to:
10 select, via at least one processor, a first level frame from the node list;
11 determine, via at least one processor, that the first level frame's source matches the source
12 associated with the first level frame containing the expandable display payload; and
13 send, via at least one processor, a message to the selected first level frame with a node list
14 identifier of the selected first level frame and the unique frame identifier associated
15 with the expandable display payload.
- 16 10. The apparatus of embodiment 1, wherein the frame busting component is further configured to:
17 determine, via at least one processor, a domain associated with the top level of the webpage;
18 wherein the message to the top level listening component with the generated executable frame
19 querying component is a post message with target origin set to the determined domain.
- 20 11. The apparatus of embodiment 1, wherein the frame busting component is further configured to:
21 determine, via at least one processor, a source of an initialization script associated with the
22 expandable display payload;
23 wherein the generated executable expandable payload rebuilding component is further configured
24 to:
25 create, via at least one processor, an element for the expandable display payload;
26 set, via at least one processor, the element's source to the source of the initialization script
27 associated with the expandable display payload; and
28 add, via at least one processor, the element to the top level of the webpage.
- 29 12. The apparatus of embodiment 11, wherein the generated executable expandable payload rebuilding
30 component is further configured to:
31 configure, via at least one processor, the element as an escaped element.
- 32 13. The apparatus of embodiment 1, wherein the top level listening component is further configured
33 to:

1 identify, via at least one processor, a buster domain;

2 wherein the buster event listener is further configured to:

3 determine, via at least one processor, an origin of the message from the frame busting
4 component; and

5 verify, via at least one processor, that the origin of the message from the frame busting
6 component matches the identified buster domain.

7 14. The apparatus of embodiment 1, wherein the expandable display payload is configured to display in
8 the webpage beyond boundaries of the first frame.

9 15. The apparatus of embodiment 1, wherein the expandable display payload is configured to expand
10 to display one of: a multilevel menu, a set of media player controls.

11
12
13 16. A webpage frame expanding display non-transient physical medium storing processor-executable
14 components, the components, comprising:

15 a component collection stored in the medium:

16 wherein the component collection, stored in the medium, includes processor-issuable instructions
17 to:

18 provide, via at least one processor, a top level listening component to a top level of a webpage,
19 wherein upon webpage loading the top level listening component is configured to
20 create a node list of first level frames in the webpage;

21 configure, via at least one processor, a first frame in the webpage to obtain an expandable
22 display payload that includes a frame escaping component, wherein the frame escaping
23 component is configured to:

24 generate, via at least one processor, a unique frame identifier associated with the
25 expandable display payload;

26 create, via at least one processor, a second frame configured to obtain a frame buster
27 payload that includes a frame busting component; and

28 add, via at least one processor, a top level event listener for messages from the top level
29 listening component;

30 wherein the top level event listener is configured, upon receiving a message from the top
31 level listening component, to:

32 determine, via at least one processor, a unique frame identifier specified in the message
33 from the top level listening component;

1 verify, via at least one processor, that the unique frame identifier specified in the
2 message from the top level listening component matches the generated unique frame
3 identifier;

4 determine, via at least one processor, a node list identifier of a first level frame in the
5 webpage specified in the message from the top level listening component; and

6 send, via at least one processor, a message to the frame busting component with the
7 determined node list identifier;

8 wherein the frame busting component is configured to:

9 determine, via at least one processor, the unique frame identifier associated with the
10 expandable display payload;

11 generate, via at least one processor, an executable frame querying component
12 configured to facilitate querying first level frames in the webpage to determine which
13 first level frame contains the expandable display payload using the unique frame
14 identifier associated with the expandable display payload;

15 send, via at least one processor, a message to the top level listening component with
16 the generated executable frame querying component; and

17 add, via at least one processor, a frame event listener for messages from the frame
18 escaping component;

19 wherein the frame event listener is configured, upon receiving a message from the
20 frame escaping component, to:

21 determine, via at least one processor, the node list identifier specified in the
22 message from the frame escaping component;

23 generate, via at least one processor, an executable expandable payload rebuilding
24 component configured to facilitate rebuilding the expandable display payload in the top
25 level of the webpage using the node list identifier specified in the message from the
26 frame escaping component; and

27 send, via at least one processor, a message to the top level listening component
28 with the generated executable expandable payload rebuilding component;

29 wherein the top level listening component is further configured to:

30 add, via at least one processor, a buster event listener for messages from the frame busting
31 component;

32 wherein the buster event listener is configured, upon receiving a message from the frame
33 busting component, to:

- 1 execute, via at least one processor, an executable component provided in the message
2 from the frame busting component.
3
- 4 17. The medium of embodiment 16, wherein a frame is an iframe.
- 5 18. The medium of embodiment 16, wherein a message is a post message.
- 6 19. The medium of embodiment 16, wherein a node list identifier is an item index in the node list.
- 7 20. The medium of embodiment 16, wherein the expandable display payload is an expandable ad.
- 8 21. The medium of embodiment 16, wherein the frame escaping component is further configured to:
9 determine, via at least one processor, whether escape from the first frame should be attempted.
- 10 22. The medium of embodiment 16, wherein the unique frame identifier associated with the
11 expandable display payload is a randomly generated number.
- 12 23. The medium of embodiment 16, wherein the frame busting component is further configured to:
13 determine, via at least one processor, the unique frame identifier associated with the expandable
14 display payload by parsing a name attribute.
- 15 24. The medium of embodiment 16, wherein the frame busting component is further configured to:
16 determine, via at least one processor, a source associated with a first level frame containing the
17 expandable display payload;
18 wherein the generated executable frame querying component is further configured to:
19 select, via at least one processor, a first level frame from the node list;
20 determine, via at least one processor, that the first level frame's source matches the source
21 associated with the first level frame containing the expandable display payload; and
22 send, via at least one processor, a message to the selected first level frame with a node list
23 identifier of the selected first level frame and the unique frame identifier associated
24 with the expandable display payload.
- 25 25. The medium of embodiment 16, wherein the frame busting component is further configured to:
26 determine, via at least one processor, a domain associated with the top level of the webpage;
27 wherein the message to the top level listening component with the generated executable frame
28 querying component is a post message with target origin set to the determined domain.
- 29 26. The medium of embodiment 16, wherein the frame busting component is further configured to:
30 determine, via at least one processor, a source of an initialization script associated with the
31 expandable display payload;
32 wherein the generated executable expandable payload rebuilding component is further configured
33 to:

1 create, via at least one processor, an element for the expandable display payload;
2 set, via at least one processor, the element's source to the source of the initialization script
3 associated with the expandable display payload; and
4 add, via at least one processor, the element to the top level of the webpage.

5 27. The medium of embodiment 26, wherein the generated executable expandable payload rebuilding
6 component is further configured to:

7 configure, via at least one processor, the element as an escaped element.

8 28. The medium of embodiment 16, wherein the top level listening component is further configured
9 to:

10 identify, via at least one processor, a buster domain;

11 wherein the buster event listener is further configured to:

12 determine, via at least one processor, an origin of the message from the frame busting
13 component; and

14 verify, via at least one processor, that the origin of the message from the frame busting
15 component matches the identified buster domain.

16 29. The medium of embodiment 16, wherein the expandable display payload is configured to display in
17 the webpage beyond boundaries of the first frame.

18 30. The medium of embodiment 16, wherein the expandable display payload is configured to expand
19 to display one of: a multilevel menu, a set of media player controls.
20
21

22 31. A processor-implemented webpage frame expanding display system, comprising:

23 a component collection means, to:

24 provide, via at least one processor, a top level listening component to a top level of a webpage,
25 wherein upon webpage loading the top level listening component is configured to
26 create a node list of first level frames in the webpage;

27 configure, via at least one processor, a first frame in the webpage to obtain an expandable
28 display payload that includes a frame escaping component, wherein the frame escaping
29 component is configured to:

30 generate, via at least one processor, a unique frame identifier associated with the
31 expandable display payload;

32 create, via at least one processor, a second frame configured to obtain a frame buster
33 payload that includes a frame busting component; and

1 add, via at least one processor, a top level event listener for messages from the top level
2 listening component;

3 wherein the top level event listener is configured, upon receiving a message from the top
4 level listening component, to:

5 determine, via at least one processor, a unique frame identifier specified in the message
6 from the top level listening component;

7 verify, via at least one processor, that the unique frame identifier specified in the
8 message from the top level listening component matches the generated unique frame
9 identifier;

10 determine, via at least one processor, a node list identifier of a first level frame in the
11 webpage specified in the message from the top level listening component; and

12 send, via at least one processor, a message to the frame busting component with the
13 determined node list identifier;

14 wherein the frame busting component is configured to:

15 determine, via at least one processor, the unique frame identifier associated with the
16 expandable display payload;

17 generate, via at least one processor, an executable frame querying component
18 configured to facilitate querying first level frames in the webpage to determine which
19 first level frame contains the expandable display payload using the unique frame
20 identifier associated with the expandable display payload;

21 send, via at least one processor, a message to the top level listening component with
22 the generated executable frame querying component; and

23 add, via at least one processor, a frame event listener for messages from the frame
24 escaping component;

25 wherein the frame event listener is configured, upon receiving a message from the
26 frame escaping component, to:

27 determine, via at least one processor, the node list identifier specified in the
28 message from the frame escaping component;

29 generate, via at least one processor, an executable expandable payload rebuilding
30 component configured to facilitate rebuilding the expandable display payload in the top
31 level of the webpage using the node list identifier specified in the message from the
32 frame escaping component; and

1 send, via at least one processor, a message to the top level listening component
2 with the generated executable expandable payload rebuilding component;
3 wherein the top level listening component is further configured to:
4 add, via at least one processor, a buster event listener for messages from the frame busting
5 component;
6 wherein the buster event listener is configured, upon receiving a message from the frame
7 busting component, to:
8 execute, via at least one processor, an executable component provided in the message
9 from the frame busting component.

10

11 32. The system of embodiment 31, wherein a frame is an iframe.

12 33. The system of embodiment 31, wherein a message is a post message.

13 34. The system of embodiment 31, wherein a node list identifier is an item index in the node list.

14 35. The system of embodiment 31, wherein the expandable display payload is an expandable ad.

15 36. The system of embodiment 31, wherein the frame escaping component is further configured to:
16 determine, via at least one processor, whether escape from the first frame should be attempted.

17 37. The system of embodiment 31, wherein the unique frame identifier associated with the expandable
18 display payload is a randomly generated number.

19 38. The system of embodiment 31, wherein the frame busting component is further configured to:
20 determine, via at least one processor, the unique frame identifier associated with the expandable
21 display payload by parsing a name attribute.

22 39. The system of embodiment 31, wherein the frame busting component is further configured to:
23 determine, via at least one processor, a source associated with a first level frame containing the
24 expandable display payload;

25 wherein the generated executable frame querying component is further configured to:
26 select, via at least one processor, a first level frame from the node list;
27 determine, via at least one processor, that the first level frame's source matches the source
28 associated with the first level frame containing the expandable display payload; and
29 send, via at least one processor, a message to the selected first level frame with a node list
30 identifier of the selected first level frame and the unique frame identifier associated
31 with the expandable display payload.

32 40. The system of embodiment 31, wherein the frame busting component is further configured to:
33 determine, via at least one processor, a domain associated with the top level of the webpage;

wherein the message to the top level listening component with the generated executable frame querying component is a post message with target origin set to the determined domain.

41. The system of embodiment 31, wherein the frame busting component is further configured to:

determine, via at least one processor, a source of an initialization script associated with the expandable display payload;

wherein the generated executable expandable payload rebuilding component is further configured to:

create, via at least one processor, an element for the expandable display payload;

set, via at least one processor, the element's source to the source of the initialization script associated with the expandable display payload; and

add, via at least one processor, the element to the top level of the webpage.

42. The system of embodiment 41, wherein the generated executable expandable payload rebuilding component is further configured to:

configure, via at least one processor, the element as an escaped element.

43. The system of embodiment 31, wherein the top level listening component is further configured to:

identify, via at least one processor, a buster domain;

wherein the buster event listener is further configured to:

determine, via at least one processor, an origin of the message from the frame busting component; and

verify, via at least one processor, that the origin of the message from the frame busting component matches the identified buster domain.

44. The system of embodiment 31, wherein the expandable display payload is configured to display in the webpage beyond boundaries of the first frame.

45. The system of embodiment 31, wherein the expandable display payload is configured to expand to display one of: a multilevel menu, a set of media player controls.

46. A processor-implemented webpage frame expanding display method, comprising:

executing processor-implemented component collection instructions to:

provide, via at least one processor, a top level listening component to a top level of a webpage,

wherein upon webpage loading the top level listening component is configured to

create a node list of first level frames in the webpage;

1 configure, via at least one processor, a first frame in the webpage to obtain an expandable
2 display payload that includes a frame escaping component, wherein the frame escaping
3 component is configured to:

4 generate, via at least one processor, a unique frame identifier associated with the
5 expandable display payload;

6 create, via at least one processor, a second frame configured to obtain a frame buster
7 payload that includes a frame busting component; and

8 add, via at least one processor, a top level event listener for messages from the top level
9 listening component;

10 wherein the top level event listener is configured, upon receiving a message from the top
11 level listening component, to:

12 determine, via at least one processor, a unique frame identifier specified in the message
13 from the top level listening component;

14 verify, via at least one processor, that the unique frame identifier specified in the
15 message from the top level listening component matches the generated unique frame
16 identifier;

17 determine, via at least one processor, a node list identifier of a first level frame in the
18 webpage specified in the message from the top level listening component; and

19 send, via at least one processor, a message to the frame busting component with the
20 determined node list identifier;

21 wherein the frame busting component is configured to:

22 determine, via at least one processor, the unique frame identifier associated with the
23 expandable display payload;

24 generate, via at least one processor, an executable frame querying component
25 configured to facilitate querying first level frames in the webpage to determine which
26 first level frame contains the expandable display payload using the unique frame
27 identifier associated with the expandable display payload;

28 send, via at least one processor, a message to the top level listening component with
29 the generated executable frame querying component; and

30 add, via at least one processor, a frame event listener for messages from the frame
31 escaping component;

32 wherein the frame event listener is configured, upon receiving a message from the
33 frame escaping component, to:

1 determine, via at least one processor, the node list identifier specified in the
2 message from the frame escaping component;
3 generate, via at least one processor, an executable expandable payload rebuilding
4 component configured to facilitate rebuilding the expandable display payload in the top
5 level of the webpage using the node list identifier specified in the message from the
6 frame escaping component; and
7 send, via at least one processor, a message to the top level listening component
8 with the generated executable expandable payload rebuilding component;
9 wherein the top level listening component is further configured to:
10 add, via at least one processor, a buster event listener for messages from the frame busting
11 component;
12 wherein the buster event listener is configured, upon receiving a message from the frame
13 busting component, to:
14 execute, via at least one processor, an executable component provided in the message
15 from the frame busting component.

16
17 47. The method of embodiment 46, wherein a frame is an iframe.

18 48. The method of embodiment 46, wherein a message is a post message.

19 49. The method of embodiment 46, wherein a node list identifier is an item index in the node list.

20 50. The method of embodiment 46, wherein the expandable display payload is an expandable ad.

21 51. The method of embodiment 46, wherein the frame escaping component is further configured to:

22 determine, via at least one processor, whether escape from the first frame should be attempted.

23 52. The method of embodiment 46, wherein the unique frame identifier associated with the expandable
24 display payload is a randomly generated number.

25 53. The method of embodiment 46, wherein the frame busting component is further configured to:

26 determine, via at least one processor, the unique frame identifier associated with the expandable
27 display payload by parsing a name attribute.

28 54. The method of embodiment 46, wherein the frame busting component is further configured to:

29 determine, via at least one processor, a source associated with a first level frame containing the
30 expandable display payload;

31 wherein the generated executable frame querying component is further configured to:

32 select, via at least one processor, a first level frame from the node list;

determine, via at least one processor, that the first level frame's source matches the source associated with the first level frame containing the expandable display payload; and send, via at least one processor, a message to the selected first level frame with a node list identifier of the selected first level frame and the unique frame identifier associated with the expandable display payload.

55. The method of embodiment 46, wherein the frame busting component is further configured to:

determine, via at least one processor, a domain associated with the top level of the webpage; wherein the message to the top level listening component with the generated executable frame querying component is a post message with target origin set to the determined domain.

56. The method of embodiment 46, wherein the frame busting component is further configured to:

determine, via at least one processor, a source of an initialization script associated with the expandable display payload;

wherein the generated executable expandable payload rebuilding component is further configured to:

create, via at least one processor, an element for the expandable display payload;

set, via at least one processor, the element's source to the source of the initialization script associated with the expandable display payload; and

add, via at least one processor, the element to the top level of the webpage.

57. The method of embodiment 56, wherein the generated executable expandable payload rebuilding component is further configured to:

configure, via at least one processor, the element as an escaped element.

58. The method of embodiment 46, wherein the top level listening component is further configured to:

identify, via at least one processor, a buster domain;

wherein the buster event listener is further configured to:

determine, via at least one processor, an origin of the message from the frame busting component; and

verify, via at least one processor, that the origin of the message from the frame busting component matches the identified buster domain.

59. The method of embodiment 46, wherein the expandable display payload is configured to display in the webpage beyond boundaries of the first frame.

60. The method of embodiment 46, wherein the expandable display payload is configured to expand to display one of: a multilevel menu, a set of media player controls.

1 [00137]

2 [00138] In order to address various issues and advance the art, the entirety of this application
3 for Supra Boundary Web Compositor Apparatuses, Methods and Systems (including the
4 Cover Page, Title, Headings, Field, Background, Summary, Brief Description of the Drawings,
5 Detailed Description, Claims, Abstract, Figures, Appendices, and otherwise) shows, by way of
6 illustration, various embodiments in which the claimed innovations may be practiced. The
7 advantages and features of the application are of a representative sample of embodiments only,
8 and are not exhaustive and/or exclusive. They are presented only to assist in understanding
9 and teach the claimed principles. It should be understood that they are not representative of all
10 claimed innovations. As such, certain aspects of the disclosure have not been discussed herein.
11 That alternate embodiments may not have been presented for a specific portion of the
12 innovations or that further undescribed alternate embodiments may be available for a portion
13 is not to be considered a disclaimer of those alternate embodiments. It will be appreciated that
14 many of those undescribed embodiments incorporate the same principles of the innovations
15 and others are equivalent. Thus, it is to be understood that other embodiments may be utilized
16 and functional, logical, operational, organizational, structural and/or topological modifications
17 may be made without departing from the scope and/or spirit of the disclosure. As such, all
18 examples and/or embodiments are deemed to be non-limiting throughout this disclosure.
19 Further and to the extent any financial and/or investment examples are included, such
20 examples are for illustrative purpose(s) only, and are not, nor should they be interpreted, as
21 investment advice. Also, no inference should be drawn regarding those embodiments discussed
22 herein relative to those not discussed herein other than it is as such for purposes of reducing
23 space and repetition. For instance, it is to be understood that the logical and/or topological
24 structure of any combination of any program components (a component collection), other
25 components, data flow order, logic flow order, and/or any present feature sets as described in
26 the figures and/or throughout are not limited to a fixed operating order and/or arrangement,
27 but rather, any disclosed order is exemplary and all equivalents, regardless of order, are
28 contemplated by the disclosure. Similarly, descriptions of embodiments disclosed throughout
29 this disclosure, any reference to direction or orientation is merely intended for convenience of
30 description and is not intended in any way to limit the scope of described embodiments.

1 Relative terms such as “lower,” “upper,” “horizontal,” “vertical,” “above,” “below,” “up,”
2 “down,” “top” and “bottom” as well as derivative thereof (e.g., “horizontally,” “downwardly,”
3 “upwardly,” etc.) should not be construed to limit embodiments, and instead, again, are offered
4 for convenience of description of orientation. These relative descriptors are for convenience
5 of description only and do not require that any embodiments be constructed or operated in a
6 particular orientation unless explicitly indicated as such. Terms such as “attached,” “affixed,”
7 “connected,” “coupled,” “interconnected,” and similar may refer to a relationship wherein
8 structures are secured or attached to one another either directly or indirectly through
9 intervening structures, as well as both movable or rigid attachments or relationships, unless
10 expressly described otherwise. Furthermore, it is to be understood that such features are not
11 limited to serial execution, but rather, any number of threads, processes, services, servers,
12 and/or the like that may execute asynchronously, concurrently, in parallel, simultaneously,
13 synchronously, and/or the like are contemplated by the disclosure. As such, some of these
14 features may be mutually contradictory, in that they cannot be simultaneously present in a
15 single embodiment. Similarly, some features are applicable to one aspect of the innovations,
16 and inapplicable to others. In addition, the disclosure includes other innovations not presently
17 claimed. Applicant reserves all rights in those presently unclaimed innovations including the
18 right to claim such innovations, file additional applications, continuations, continuations in
19 part, divisions, and/or the like thereof. As such, it should be understood that advantages,
20 embodiments, examples, functional, features, logical, operational, organizational, structural,
21 topological, and/or other aspects of the disclosure are not to be considered limitations on the
22 disclosure as defined by the claims or limitations on equivalents to the claims. It is to be
23 understood that, depending on the particular needs and/or characteristics of a SBWC
24 individual and/or enterprise user, database configuration and/or relational model, data type,
25 data transmission and/or network framework, syntax structure, and/or the like, various
26 embodiments of the SBWC, may be implemented that enable a great deal of flexibility and
27 customization. For example, aspects of the SBWC may be adapted for any web frame
28 expansion (e.g., dashboard/widget expansion beyond the initial frame bounds). While various
29 embodiments and discussions of the SBWC have included web frame and graphic
30 compositing, however, it is to be understood that the embodiments described herein may be

1 readily configured and/or customized for a wide variety of other applications and/or
2 implementations.

CLAIMS

1

2 What is claimed is:

3 1. A webpage frame expanding display apparatus, comprising:

4 a memory;

5 a component collection in the memory;

6 a processor disposed in communication with the memory, and configured to issue a plurality of
7 processing instructions from the component collection stored in the memory,

8 wherein the processor issues instructions from the component collection, stored in the memory,
9 to:

10 provide, via at least one processor, a top level listening component to a top level of a webpage,
11 wherein upon webpage loading the top level listening component is configured to
12 create a node list of first level frames in the webpage;

13 configure, via at least one processor, a first frame in the webpage to obtain an expandable
14 display payload that includes a frame escaping component, wherein the frame escaping
15 component is configured to:

16 generate, via at least one processor, a unique frame identifier associated with the
17 expandable display payload;

18 create, via at least one processor, a second frame configured to obtain a frame buster
19 payload that includes a frame busting component; and

20 add, via at least one processor, a top level event listener for messages from the top level
21 listening component;

22 wherein the top level event listener is configured, upon receiving a message from the top
23 level listening component, to:

24 determine, via at least one processor, a unique frame identifier specified in the message
25 from the top level listening component;

26 verify, via at least one processor, that the unique frame identifier specified in the
27 message from the top level listening component matches the generated unique frame
28 identifier;

29 determine, via at least one processor, a node list identifier of a first level frame in the
30 webpage specified in the message from the top level listening component; and

1 send, via at least one processor, a message to the frame busting component with the
2 determined node list identifier;

3 wherein the frame busting component is configured to:

4 determine, via at least one processor, the unique frame identifier associated with the
5 expandable display payload;

6 generate, via at least one processor, an executable frame querying component
7 configured to facilitate querying first level frames in the webpage to determine which
8 first level frame contains the expandable display payload using the unique frame
9 identifier associated with the expandable display payload;

10 send, via at least one processor, a message to the top level listening component with
11 the generated executable frame querying component; and

12 add, via at least one processor, a frame event listener for messages from the frame
13 escaping component;

14 wherein the frame event listener is configured, upon receiving a message from the
15 frame escaping component, to:

16 determine, via at least one processor, the node list identifier specified in the
17 message from the frame escaping component;

18 generate, via at least one processor, an executable expandable payload rebuilding
19 component configured to facilitate rebuilding the expandable display payload in the top
20 level of the webpage using the node list identifier specified in the message from the
21 frame escaping component; and

22 send, via at least one processor, a message to the top level listening component
23 with the generated executable expandable payload rebuilding component;

24 wherein the top level listening component is further configured to:

25 add, via at least one processor, a buster event listener for messages from the frame busting
26 component;

27 wherein the buster event listener is configured, upon receiving a message from the frame
28 busting component, to:

29 execute, via at least one processor, an executable component provided in the message
30 from the frame busting component.

31

32 2. The apparatus of claim 1, wherein a frame is an iframe.

33 3. The apparatus of claim 1, wherein a message is a post message.

- 1 4. The apparatus of claim 1, wherein a node list identifier is an item index in the node list.
- 2 5. The apparatus of claim 1, wherein the expandable display payload is an expandable ad.
- 3 6. The apparatus of claim 1, wherein the frame escaping component is further configured to:
- 4 determine, via at least one processor, whether escape from the first frame should be attempted.
- 5 7. The apparatus of claim 1, wherein the unique frame identifier associated with the expandable display
- 6 payload is a randomly generated number.
- 7 8. The apparatus of claim 1, wherein the frame busting component is further configured to:
- 8 determine, via at least one processor, the unique frame identifier associated with the expandable
- 9 display payload by parsing a name attribute.
- 10 9. The apparatus of claim 1, wherein the frame busting component is further configured to:
- 11 determine, via at least one processor, a source associated with a first level frame containing the
- 12 expandable display payload;
- 13 wherein the generated executable frame querying component is further configured to:
- 14 select, via at least one processor, a first level frame from the node list;
- 15 determine, via at least one processor, that the first level frame's source matches the source
- 16 associated with the first level frame containing the expandable display payload; and
- 17 send, via at least one processor, a message to the selected first level frame with a node list
- 18 identifier of the selected first level frame and the unique frame identifier associated
- 19 with the expandable display payload.
- 20 10. The apparatus of claim 1, wherein the frame busting component is further configured to:
- 21 determine, via at least one processor, a domain associated with the top level of the webpage;
- 22 wherein the message to the top level listening component with the generated executable frame
- 23 querying component is a post message with target origin set to the determined domain.
- 24 11. The apparatus of claim 1, wherein the frame busting component is further configured to:
- 25 determine, via at least one processor, a source of an initialization script associated with the
- 26 expandable display payload;
- 27 wherein the generated executable expandable payload rebuilding component is further configured
- 28 to:
- 29 create, via at least one processor, an element for the expandable display payload;
- 30 set, via at least one processor, the element's source to the source of the initialization script
- 31 associated with the expandable display payload; and
- 32 add, via at least one processor, the element to the top level of the webpage.

- 1 12. The apparatus of claim 11, wherein the generated executable expandable payload rebuilding
2 component is further configured to:
3 configure, via at least one processor, the element as an escaped element.
- 4 13. The apparatus of claim 1, wherein the top level listening component is further configured to:
5 identify, via at least one processor, a buster domain;
6 wherein the buster event listener is further configured to:
7 determine, via at least one processor, an origin of the message from the frame busting
8 component; and
9 verify, via at least one processor, that the origin of the message from the frame busting
10 component matches the identified buster domain.
- 11 14. The apparatus of claim 1, wherein the expandable display payload is configured to display in the
12 webpage beyond boundaries of the first frame.
- 13 15. The apparatus of claim 1, wherein the expandable display payload is configured to expand to
14 display one of: a multilevel menu, a set of media player controls.
15
16
- 17 16. A webpage frame expanding display non-transient physical medium storing processor-executable
18 components, the components, comprising:
19 a component collection stored in the medium:
20 wherein the component collection, stored in the medium, includes processor-issuable instructions
21 to:
22 provide, via at least one processor, a top level listening component to a top level of a webpage,
23 wherein upon webpage loading the top level listening component is configured to
24 create a node list of first level frames in the webpage;
25 configure, via at least one processor, a first frame in the webpage to obtain an expandable
26 display payload that includes a frame escaping component, wherein the frame escaping
27 component is configured to:
28 generate, via at least one processor, a unique frame identifier associated with the
29 expandable display payload;
30 create, via at least one processor, a second frame configured to obtain a frame buster
31 payload that includes a frame busting component; and
32 add, via at least one processor, a top level event listener for messages from the top level
33 listening component;

1 wherein the top level event listener is configured, upon receiving a message from the top
2 level listening component, to:
3 determine, via at least one processor, a unique frame identifier specified in the message
4 from the top level listening component;
5 verify, via at least one processor, that the unique frame identifier specified in the
6 message from the top level listening component matches the generated unique frame
7 identifier;
8 determine, via at least one processor, a node list identifier of a first level frame in the
9 webpage specified in the message from the top level listening component; and
10 send, via at least one processor, a message to the frame busting component with the
11 determined node list identifier;

12 wherein the frame busting component is configured to:

13 determine, via at least one processor, the unique frame identifier associated with the
14 expandable display payload;
15 generate, via at least one processor, an executable frame querying component
16 configured to facilitate querying first level frames in the webpage to determine which
17 first level frame contains the expandable display payload using the unique frame
18 identifier associated with the expandable display payload;
19 send, via at least one processor, a message to the top level listening component with
20 the generated executable frame querying component; and
21 add, via at least one processor, a frame event listener for messages from the frame
22 escaping component;

23 wherein the frame event listener is configured, upon receiving a message from the
24 frame escaping component, to:

25 determine, via at least one processor, the node list identifier specified in the
26 message from the frame escaping component;

27 generate, via at least one processor, an executable expandable payload rebuilding
28 component configured to facilitate rebuilding the expandable display payload in the top
29 level of the webpage using the node list identifier specified in the message from the
30 frame escaping component; and

31 send, via at least one processor, a message to the top level listening component
32 with the generated executable expandable payload rebuilding component;

33 wherein the top level listening component is further configured to:

1 add, via at least one processor, a buster event listener for messages from the frame busting
2 component;

3 wherein the buster event listener is configured, upon receiving a message from the frame
4 busting component, to:

5 execute, via at least one processor, an executable component provided in the message
6 from the frame busting component.

7
8
9 17. A processor-implemented webpage frame expanding display system, comprising:

10 a component collection means, to:

11 provide, via at least one processor, a top level listening component to a top level of a webpage,
12 wherein upon webpage loading the top level listening component is configured to
13 create a node list of first level frames in the webpage;

14 configure, via at least one processor, a first frame in the webpage to obtain an expandable
15 display payload that includes a frame escaping component, wherein the frame escaping
16 component is configured to:

17 generate, via at least one processor, a unique frame identifier associated with the
18 expandable display payload;

19 create, via at least one processor, a second frame configured to obtain a frame buster
20 payload that includes a frame busting component; and

21 add, via at least one processor, a top level event listener for messages from the top level
22 listening component;

23 wherein the top level event listener is configured, upon receiving a message from the top
24 level listening component, to:

25 determine, via at least one processor, a unique frame identifier specified in the message
26 from the top level listening component;

27 verify, via at least one processor, that the unique frame identifier specified in the
28 message from the top level listening component matches the generated unique frame
29 identifier;

30 determine, via at least one processor, a node list identifier of a first level frame in the
31 webpage specified in the message from the top level listening component; and

32 send, via at least one processor, a message to the frame busting component with the
33 determined node list identifier;

1 wherein the frame busting component is configured to:

2 determine, via at least one processor, the unique frame identifier associated with the
3 expandable display payload;

4 generate, via at least one processor, an executable frame querying component
5 configured to facilitate querying first level frames in the webpage to determine which
6 first level frame contains the expandable display payload using the unique frame
7 identifier associated with the expandable display payload;

8 send, via at least one processor, a message to the top level listening component with
9 the generated executable frame querying component; and

10 add, via at least one processor, a frame event listener for messages from the frame
11 escaping component;

12 wherein the frame event listener is configured, upon receiving a message from the
13 frame escaping component, to:

14 determine, via at least one processor, the node list identifier specified in the
15 message from the frame escaping component;

16 generate, via at least one processor, an executable expandable payload rebuilding
17 component configured to facilitate rebuilding the expandable display payload in the top
18 level of the webpage using the node list identifier specified in the message from the
19 frame escaping component; and

20 send, via at least one processor, a message to the top level listening component
21 with the generated executable expandable payload rebuilding component;

22 wherein the top level listening component is further configured to:

23 add, via at least one processor, a buster event listener for messages from the frame busting
24 component;

25 wherein the buster event listener is configured, upon receiving a message from the frame
26 busting component, to:

27 execute, via at least one processor, an executable component provided in the message
28 from the frame busting component.

29
30
31 18. A processor-implemented webpage frame expanding display method, comprising:

32 executing processor-implemented component collection instructions to:

1 provide, via at least one processor, a top level listening component to a top level of a webpage,
2 wherein upon webpage loading the top level listening component is configured to
3 create a node list of first level frames in the webpage;
4 configure, via at least one processor, a first frame in the webpage to obtain an expandable
5 display payload that includes a frame escaping component, wherein the frame escaping
6 component is configured to:
7 generate, via at least one processor, a unique frame identifier associated with the
8 expandable display payload;
9 create, via at least one processor, a second frame configured to obtain a frame buster
10 payload that includes a frame busting component; and
11 add, via at least one processor, a top level event listener for messages from the top level
12 listening component;
13 wherein the top level event listener is configured, upon receiving a message from the top
14 level listening component, to:
15 determine, via at least one processor, a unique frame identifier specified in the message
16 from the top level listening component;
17 verify, via at least one processor, that the unique frame identifier specified in the
18 message from the top level listening component matches the generated unique frame
19 identifier;
20 determine, via at least one processor, a node list identifier of a first level frame in the
21 webpage specified in the message from the top level listening component; and
22 send, via at least one processor, a message to the frame busting component with the
23 determined node list identifier;
24 wherein the frame busting component is configured to:
25 determine, via at least one processor, the unique frame identifier associated with the
26 expandable display payload;
27 generate, via at least one processor, an executable frame querying component
28 configured to facilitate querying first level frames in the webpage to determine which
29 first level frame contains the expandable display payload using the unique frame
30 identifier associated with the expandable display payload;
31 send, via at least one processor, a message to the top level listening component with
32 the generated executable frame querying component; and

1 add, via at least one processor, a frame event listener for messages from the frame
2 escaping component;

3 wherein the frame event listener is configured, upon receiving a message from the
4 frame escaping component, to:

5 determine, via at least one processor, the node list identifier specified in the
6 message from the frame escaping component;

7 generate, via at least one processor, an executable expandable payload rebuilding
8 component configured to facilitate rebuilding the expandable display payload in the top
9 level of the webpage using the node list identifier specified in the message from the
10 frame escaping component; and

11 send, via at least one processor, a message to the top level listening component
12 with the generated executable expandable payload rebuilding component;

13 wherein the top level listening component is further configured to:

14 add, via at least one processor, a buster event listener for messages from the frame busting
15 component;

16 wherein the buster event listener is configured, upon receiving a message from the frame
17 busting component, to:

18 execute, via at least one processor, an executable component provided in the message
19 from the frame busting component.